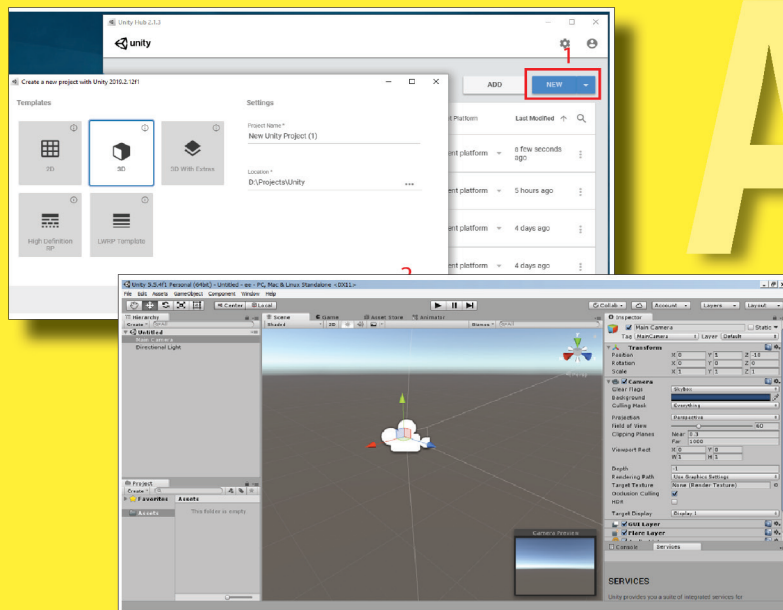


Министерство науки и высшего образования
Российской Федерации
Тольяттинский государственный университет
Институт математики, физики и информационных технологий

О.М. Гущина, А.В. Очеповский

РАЗРАБОТКА AR-ПРИЛОЖЕНИЙ

Электронное учебно-методическое пособие



AR

© ФГБОУ ВО «Тольяттинский государственный университет», 2021

ISBN 978-5-8259-1580-7

УДК 004.9

ББК 3811

Рецензенты:

канд. техн. наук, доцент, доцент кафедры «Управление качеством и инновационные технологии» Поволжского государственного университета сервиса *Д.И. Панюков*;

д-р физ.-мат. наук, профессор кафедры «Прикладная математика и информатика» Тольяттинского государственного университета *А.И. Сафронов*.

Гущина, О.М. Разработка AR-приложений : электронное учебно-методическое пособие / О.М. Гущина, А.В. Очеповский. – Тольятти : Изд-во ТГУ, 2021. – 1 оптический диск. – ISBN 978-5-8259-1580-7.

Учебно-методическое пособие содержит информацию о понятии дополненной реальности и инструментах, с помощью которых можно создавать интерактивные учебные приложения дополненной реальности. В нем описываются теоретические подходы и предлагаются практические задания.

Предназначено для студентов, обучающихся по направлениям подготовки бакалавров 01.03.02 «Прикладная математика и информатика», 02.03.03 «Математическое обеспечение и администрирование информационных систем», 09.03.03 «Прикладная информатика» очной и заочной форм обучения.

Может быть полезно студентам, профессорско-преподавательскому составу высших учебных заведений, а также любому желающему получить знания в области технологии дополненной реальности в качестве практического руководства при разработке мобильных приложений с элементами дополненной реальности.

Текстовое электронное издание.

Рекомендовано к изданию научно-методическим советом Тольяттинского государственного университета.

Минимальные системные требования: IBM PC-совместимый компьютер: Windows XP/Vista/7/8; PIII 500 МГц или эквивалент; 128 Мб ОЗУ; SVGA; CD-ROM; Adobe Acrobat Reader.

© ФГБОУ ВО «Тольяттинский государственный университет», 2021



Редактор *Е.В. Пилясова*
Технический редактор *Н.П. Крюкова*
Компьютерная верстка: *Л.В. Сызганцева*
Художественное оформление,
компьютерное проектирование: *И.И. Шишкина*

Дата подписания к использованию 03.06.2021.
Объем издания 17,3 Мб.
Комплектация издания: компакт-диск,
первичная упаковка.
Заказ № 1-37-20.

Издательство Тольяттинского
государственного университета
445020, г. Тольятти, ул. Белорусская, 14,
тел. 8 (8482) 53-91-47, www.tltsu.ru

Содержание

Введение	5
Создание приложений дополненной реальности с использованием популярных библиотек AR и платформы разработки в реальном времени Unity 3D	6
Создание приложения дополненной реальности с использованием программной платформы EasyAR	17
Создание приложения дополненной реальности с использованием программной платформы Vuforia	30
Создание приложения дополненной реальности с использованием программной платформы ARToolKit	36
Распознавание объектов на Android с помощью модели TensorFlow	44
Библиографический список	57

Введение

Технология дополненной реальности не является новой, хотя сам термин получил популярность относительно недавно. С использованием данной технологии создается множество различных приложений в самых разных сферах, от медиа до военной. На эту тему существует много книг и исследований с использованием различных математических расчетов. Данное же методическое пособие направлено в первую очередь на тех, кто только начал знакомиться с технологией AR (дополненная реальность) и не обладает специальными техническими навыками, позволяющими самостоятельно писать приложения.

В методическом пособии представлено общее описание технологии дополненной реальности и нейронных сетей без углубления в математические расчеты. Также приведены практические примеры, с помощью которых можно самостоятельно запустить AR-приложение и обучить собственную сверточную нейронную сеть.

Также методическое пособие включает практические задания, выполнение которых подкрепит теоретические знания и позволит самостоятельно поработать с такими технологиями, как Unity, EasyAR, Vuforia, ARToolKit.

Создание приложений дополненной реальности с использованием популярных библиотек AR и платформы разработки в реальном времени Unity 3D

Обзор технологии дополненной реальности и алгоритмов ее работы

Реальность, в которой существует каждый из нас, представляет собой большие хаотичные потоки информации, которые мы получаем из компьютера, смартфона, телевизора, радио и других источников. Все эти устройства – часть нашего современного развитого технологического мира. Мира, в котором технологии улучшают повседневную человеческую жизнь.

С приходом все более новых и мощных устройств наступает новое время, а вместе с ним меняется и наша реальность. Современный мир адаптируется под человека, его предпочтения и потребности. Реальность становится гибкой, изменчивой и высокоперсонализированной. Повсюду преодолеваются языковые барьеры и появляются новые сенсорные ощущения, которые делают зрение, слух, прикосновение и вкус совершенно новыми. И все это становится возможным благодаря новой дополненной реальности.

Дополненная реальность (AR) – одна из технологий, вызывающая все больший интерес.

Смешивая виртуальный мир с реальным, дополненная реальность обеспечивает уровень погружения, который не может обеспечить ни одно виртуальное оборудование. Системы AR уже используются во многих областях, таких как хирургия или инженерия. Однако большинство из этих систем работают только внутри помещений и охватывают относительно небольшие площади. Достижения компьютерной, визуальной и беспроводной технологий делают возможным разработку наружных беспроводных систем для комплексного анализа, принятия решений и управления процессами. Использование таких систем имеет следующие преимущества:

- представление данных «на месте»;
- представление 2D-, 3D-информации.

Термин «дополненная реальность» – это не то же самое, что «виртуальная» или «смешанная», между ними есть различия. Для лучшего понимания введем понятия этих терминов.

Дополненная реальность — это интерактивное восприятие среды реального мира, в которой объекты, находящиеся в реальном мире, дополняются с помощью компьютерной информации. То есть получаемые сенсорные данные усиливаются дополнительными сведениями для улучшения восприятия информации. AR можно определить как систему, которая выполняет три основные функции: сочетание реального и виртуального миров, взаимодействие в реальном времени и точное трехмерное объединение виртуальных и реальных объектов. Важно понимать, что дополненная реальность *неразрывно связана со средой реального мира.*

Виртуальная реальность — трехмерная компьютерная среда, которую человек может исследовать и взаимодействовать с ней. Человек становится частью этого виртуального мира или погружается в эту среду и в то же время способен манипулировать объектами или выполнять ряд действий. То есть виртуальная реальность либо *имитирует реальный мир*, либо *полностью его заменяет.*

Смешанная или гибридная реальность — это слияние реальных и виртуальных миров для создания новых сред, где физические и цифровые объекты сосуществуют и взаимодействуют в реальном времени. Смешанная реальность имеет место не только в физическом или виртуальном мире, но представляет собой гибрид реальности и виртуальной реальности, охватывающий как дополненную реальность, так и дополненную виртуальность. То есть *реальные объекты могут взаимодействовать с виртуальными.*

Приложения дополненной реальности можно разделить на три категории:

- решения, использующие встроенные устройства телефона, такие как GPS, компас, акселерометр, для наложения объектов на поверхность входного изображения камеры;
- решения, использующие метки для ориентации и наложения на распознанные метки требуемых элементов. В качестве метки можно привести QR-код, который находится в распечатанном виде на столе или выведен на экран какого-либо устройства. На место опознанного QR-кода выводится информация, которая накладывается поверх входного изображения камеры устройства;

– решения, использующие реальный мир для ориентации. Обычно в подобных приложениях выполняется анализ местности с помощью поступающих изображений с камеры устройства. Далее строится карта полигонов и пользователю предлагается разместить какой-либо 3D-объект на данной сцене. Взаимодействие объекта с реальным миром будет зависеть от заложенных заранее алгоритмов.

Точная фиксация виртуальных объектов на реальной среде является главной проблемой в дополненной реальности (AR). Эта проблема возникает независимо от сложности виртуальных объектов, с помощью которых планируется улучшить реальную среду. Как простые текстовые аннотации, так и сложные виртуальные 3D-объекты реальных вещей должны быть жестко зафиксированы на изображении реальной среды. Система дополненной реальности, в которой отсутствует это требование, будет иметь артефакты, то есть иметь сильное «дрожание» виртуальных объектов на реальной среде, и, следовательно, подобная система не сможет дать пользователю лучшего впечатления от дополненной реальности.

Работу алгоритма дополненной реальности можно подразделить на четыре этапа:

- изображение реального мира захватывается с помощью устройств ввода, то есть камеры;
- алгоритмы распознавания изображения анализируют входящие данные;
- генерируется виртуальная модель методом системы графов;
- сгенерированная модель интегрируется во входящий видеопоток, и итоговое изображение выводится на экран устройства.

Общий алгоритм работы представлен на рис. 1.

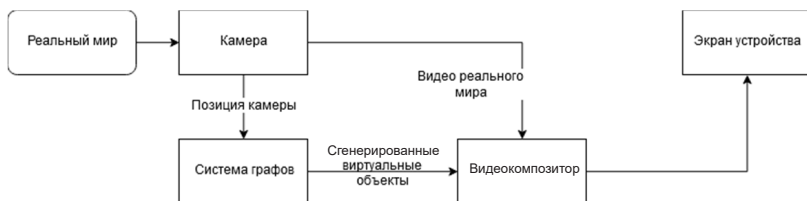


Рис. 1. Алгоритм работы дополненной реальности

Важным аспектом в представленном выше описании работы является то, как работает алгоритм распознавания входящего изображения. Его реализация будет зависеть от человека, его создающего. Чаще всего для анализа входного изображения используют готовую библиотеку OpenCV, так как она создается достаточно давно и предоставляет готовый мощный инструментарий. Для обработки изображений рекомендуется использовать языки C/C++, так как они не используют автоматическую работу с памятью, а передачу данных можно осуществлять посредством указателей, что в свою очередь снижает затраты времени на передачу данных между функциями.

Но на сегодняшний день нет необходимости создавать подобные алгоритмы с нуля. В настоящее время существуют готовые наборы средств для разработки приложений дополненной реальности (SDK – software development kit). Они просты в освоении и имеют понятный программный интерфейс. Подробный обзор готовых библиотек и их использования будет сделан в следующих главах.

Обзор популярных программных платформ для создания приложений дополненной реальности

Прежде чем приступить к разработке приложения дополненной реальности, нужно выбрать одну из двух категорий: приложения с использованием определения местоположения и приложения на основе маркеров. Кратко обсудим различия между ними.

Приложения на основе маркеров базируются на распознавании изображений. Они используют черные и белые маркеры в качестве триггеров для отображения AR-контента. Чтобы увидеть компонент дополненной реальности, пользователь должен навести камеру на маркер. Как только устройство распознает маркер, приложение накладывает цифровые данные на этот маркер, и пользователь может увидеть виртуальный объект поверх сцены реального мира.

При создании приложения на основе маркера разработчик заранее подготавливает изображения маркеров или их дескрипторы, чтобы упростить процесс их поиска при анализе данных камеры. Другими словами, объекты уже запрограммированы в приложении, поэтому их легче обнаружить. Неудивительно, что большинство AR-приложений основаны на маркерах. Они особенно популярны в рекламе.

AR-приложения на основе местоположения работают без маркеров. Они определяют местоположение пользователя с помощью GPS, акселерометра или цифрового компаса и накладывают объекты дополненной реальности поверх реальных физических мест. Самым известным приложением на основе определения местоположения, безусловно, является Pokemon Go.

Эти приложения могут отправлять пользователю уведомления в зависимости от его местоположения, чтобы оповестить о новом контенте AR, связанном с данным местом. Например, приложение может дать рекомендации о лучших ресторанах поблизости и показать, как туда добраться, может помочь найти свой автомобиль на огромной парковке с помощью GPS.

Подробнее остановимся на создании приложения, использующего маркеры, и основных критериях выбора SDK дополненной реальности.

Когда дело доходит до выбора SDK для разработки, легко разочароваться в количестве доступных инструментов. Чтобы выбрать SDK, который лучше всего подходит для проекта, нужно убедиться, что он поддерживает все функции, необходимые приложению.

Далее подробно рассмотрим основные моменты: цена, платформы, распознавание плоских изображений, распознавание 3D-объектов, поддержка Unity (опционально), поддержка облачного распознавания, GPS и SLAM.

Ценообразование является первым отличительным знаком AR SDK. Для тех, кто хочет попробовать AR-разработку в первый раз, лучшими вариантами являются бесплатные AR SDK с открытым исходным кодом, которые открыты для участия в разработке и могут быть расширены новыми функциями, предлагаемыми разработчиками.

Платные SDK в большинстве случаев предлагают несколько тарифных планов в зависимости от потребностей пользователя. Бесплатные планы имеют ограниченные возможности и должны быть «демо-версией» всего продукта. Создание сложного приложения с большим динамическим контентом, вероятно, потребует коммерческой лицензии.

При разработке приложений для iOS или Android проблем с выбором инструментария дополненной реальности не возникнет, по-

скольку почти все они поддерживают их. Между тем выбор инструментов, совместимых с Windows или macOS, невелик. Тем не менее можно создать приложение для компьютеров с Windows или смартфонов, используя комплект разработки дополненной реальности, поддерживающий универсальную платформу Windows (UWP).

Распознавание плоских изображений является обязательным для любого приложения AR, поскольку оно позволяет идентифицировать объекты, места и изображения. Для этого смартфоны и другие устройства используют машинное зрение вместе с камерой и программным обеспечением искусственного интеллекта для отслеживания изображений, которые впоследствии могут быть наложены на анимацию, звук, HTML-контент и т. д.

Распознавание и отслеживание 3D-изображений – одна из самых ценных функций любого AR SDK. Благодаря отслеживанию приложение может «понимать» и расширять большие пространства вокруг пользователя внутри больших зданий, таких как аэропорты, автобусные станции, торговые центры и т. д. Приложения, поддерживающие 3D-распознавание, могут распознавать трехмерные объекты, такие как коробки, чашки, цилиндры, игрушки и т. д.

В настоящее время эта технология широко используется в мобильных играх и электронной коммерции.

Unity, как известно, является самым популярным и мощным игровым движком во всем мире. Хотя он обычно используется для разработки компьютерных игр, его также можно использовать для создания AR-приложений с мощными эффектами. Независимо от того, собираетесь ли вы создать что-то новое или расширить традиционную идею с помощью новых методов, такой многофункциональный инструмент, как Unity, позволит реализовать оба варианта.

При разработке мобильных приложений AR нужно решить, будут ли пользовательские данные храниться локально или в облаке. Это решение в основном зависит от количества маркеров, которые можно создать. Если есть необходимость добавления большого количества маркеров в приложение, то рассматривают возможность хранения всех этих данных в облаке, иначе приложение будет использовать много места на устройстве. Кроме того, важно иметь представление о количестве маркеров, используемых приложением,

поскольку некоторые SDK дополненной реальности поддерживают сто маркеров, а другие поддерживают тысячи.

С другой стороны, локальное хранение маркеров (на устройстве) позволяет пользователям запускать приложение дополненной реальности в автономном режиме, что может быть удобно, поскольку у пользователя не всегда есть Wi-Fi или 3G/4G.

Если нужно создать приложение AR на основе определения местоположения, геолокация является фундаментальной функцией, которая должна поддерживаться используемым AR SDK. GPS можно использовать как в AR-играх, например Pokemon Go, так и в приложениях, созданных для наложения данных в некоторых близлежащих местах (например, для поиска ближайшего ресторана).

SLAM означает одновременную локализацию и сопоставление. Это алгоритм, который отображает среду, в которой находится пользователь, и отслеживает все его движения. Приложения AR, содержащие эту функцию, могут запоминать положение физических объектов в некоторой среде и позиционировать виртуальные объекты в соответствии с их положением и движениями пользователя. SLAM обладает огромным потенциалом и может использоваться во многих видах приложений, не только в приложениях AR. Основным преимуществом этой технологии является возможность использования в помещении, в то время как GPS доступен только на улице.

Для выбора наиболее подходящего SDK можно ознакомиться с табл. 1.

Таблица 1

Характеристики популярных фреймворков дополненной реальности

Фреймворки Характеристики	EasyAR	Vuforia	Google ARCore	Apple ARKit	Wikitude	VOID AR
Бесплатная лицензия	Да	Да	Да	Да	Да	Да
SLAM	Да	Да	Да	Да	Да	Да
Плоские изображения, маркеры	Да	Да	Да	Да	Да	Да
Облачное распознавание	Да	Да	Нет	Да	Да	Да

Фреймворки Характеристики	EasyAR	Vuforia	Google ARCore	Apple ARKit	Wikitude	VOID AR
Отслеживание нескольких целей	Да	Да	Да	Да	Да	Да
Распознавание 3D-объектов	Да	Да	Да	Да	Да	Да
Поддержка Unity 3D	Да	Да	Да	Да	Да	Да
Поддержка GPS	Нет	Да	Да	Да	Да	Да

Установка платформы разработки в реальном времени Unity 3D с использованием менеджера Unity Hub

Unity – это инструмент для разработки двух- и трехмерных приложений и игр, работающий под операционными системами Windows, Linux и OS X. Использование инструмента Unity для создания AR-приложений подходит не только для начинающих разработчиков, но и для опытных специалистов.

Системные требования

Операционная система: Windows 7 SP1+, 8, 10, 64-bit versions only; macOS 10.12+; Ubuntu 16.04, 18.04, and CentOS 7.

Центральный процессор с поддержкой набора инструкций SSE2.

Графический процессор: видеокарта с поддержкой DX10 (версия шейдеров 4.0).

Остальное зависит главным образом от сложности проектов дополненной реальности.

Для установки Unity достаточно загрузить установочный файл из архива на сайте unity3d.com/get-unity/download/archive. С этой страницы можно скачать предыдущие версии Unity Personal или Pro при наличии ее лицензии. Надо заметить, что обратной совместимости версий в Unity нет; проекты, сделанные в версии 5.x, не откроются в версии 4.x.

Для экономии времени и сил рекомендуется скачать Unity Hub.

Unity Hub – это новое приложение, разработанное для упрощения рабочего процесса. Hub – настоящий центр управления Unity-проектами.

С его помощью можно загружать любую версию Unity без установочного файла, а также создавать проекты под эти версии. Для ознакомления с принципами работы платформы Unity Hub содержит множество примеров, что позволит осуществить первый запуск платформы для работы по созданию приложений.

Загрузку Unity Hub можно выполнить с сайта unity3d.com/ru/get-unity/download. По окончании загрузки файла на компьютер нужно запустить установочный файл и произвести установку в любой удобный для пользователя каталог. По окончании установки нужно запустить Unity Hub, чтобы проверить правильность выполненных действий и начать работать с платформой.

При первом запуске отображается главное окно, как показано на рис. 2.

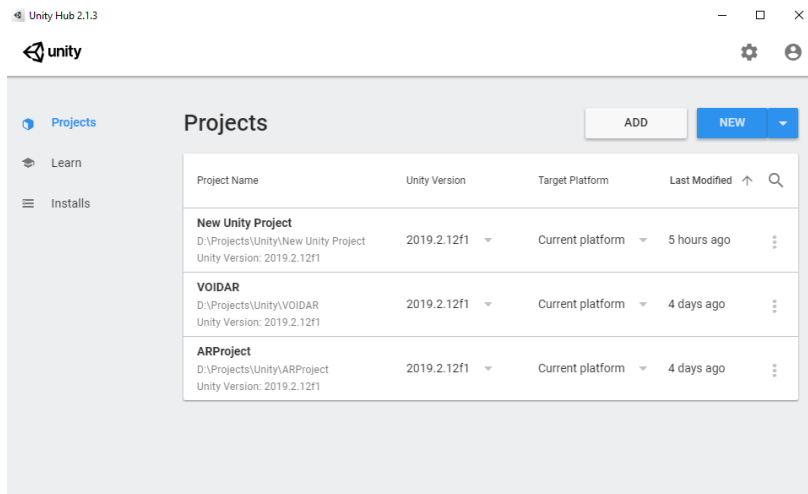



Рис. 2. Главное окно Unity Hub

Для использования бесплатной версии авторизация необязательна. Чтобы активировать персональную лицензию в окне управления лицензиями, перейдите в настройки нажатием на иконку настроек, изображенной следующим образом: , и перейдите в пункт «License Management». Нажмите на кнопку «Activate new license» и добавьте лицензию «Personal». Результат выполнения можно увидеть на рис. 3.

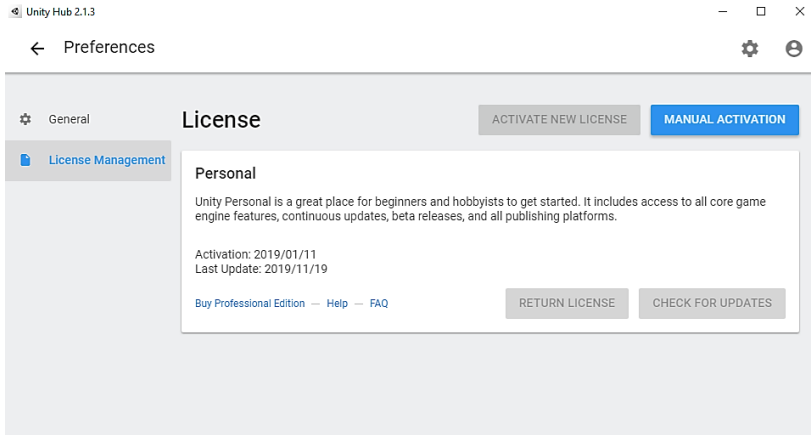


Рис. 3. Окно лицензий Unity Hub

Далее необходимо установить последнюю версию Unity. Для этого перейдите в главное окно, которое отображается при открытии Unity Hub, и откройте раздел «Installs». В данном окне программы станет доступна кнопка «Add», необходимо нажать на нее, после чего откроется новое модальное окно с выбором нужной версии. Не изменяйте выбора, если хотите установить самую последнюю версию. Нажмите на кнопку «Next», после чего вы увидите следующее окно, представленное на рис. 4.

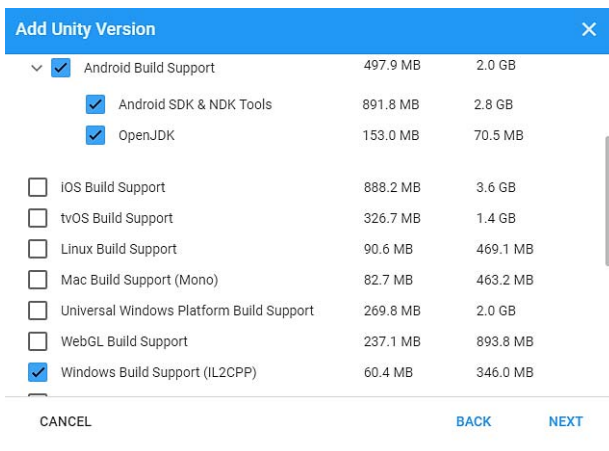


Рис. 4. Окно выбора версий Unity Hub

В данном окне нужно выбрать устанавливаемые модули. Для первого AR-проекта достаточно сделать выбор, как на рис. 4, где «Android Build Support» и его подпункты необходимы для компиляции приложения под операционную систему Android. А пункт «Windows Build Support» служит для конвертации кода IL перед созданием двоичного файла для выбранной платформы (.exe, .apk, .xap), что позволяет повысить безопасность и производительность приложения. После нажмите на кнопку «Next». Начнется установка.

По окончании установки у вас появится возможность создавать проекты.

Создание приложения дополненной реальности с использованием программной платформы EasyAR

EasyAR SDK – это набор средств разработки дополненной реальности. Существует две редакции: EasyAR SDK Basic и EasyAR SDK Pro.

EasyAR SDK Basic бесплатен для коммерческого использования. Не имеет ограничений или водяных знаков. SDK способен распознавать плоские изображения (маркеры, картинки), имеет плавную загрузку (изображение не проявляет коллизий при изменении угла наклона камеры) и способен хранить и обрабатывать более чем 1000 локальных целей. Также имеется возможность воспроизведения видео и отслеживания нескольких целей одновременно.

EasyAR SDK Pro является платным, но имеет бесплатный пробный период в виде ограничения на количество распознаваний – 100 в день. Все функции EasyAR SDK Basic доступны в EasyAR SDK Pro. И по сравнению с EasyAR SDK Basic имеется больше функций, включая распознавание и отслеживание 3D-объектов, SLAM и запись экрана. Информация о ценах и способах оплаты указана на странице продукта EasyAR SDK.

Сравнение EasyAR SDK Basic и EasyAR SDK Pro приведено в табл. 2.

Таблица 2

Сравнение Basic и Pro версий

Версии	EasyAR SDK Basic	EasyAR SDK Pro
Параметры сравнения		
C/C++17/C++ API	Да	Да
Java/Kotlin API для Android	Да	Да
Swift/Objective-C API для iOS	Да	Да
Воспроизведение видео с использованием аппаратного декодирования H.264	Да	Да
Воспроизведение видео	Да	Да
Поддержка Unity 3D	Да	Да
Отслеживание плоских изображений	Да	Да

Версии	EasyAR SDK Basic	EasyAR SDK Pro
Параметры сравнения		
Неограниченное количество распознаваний	Да	Да
Обнаружение и отслеживание нескольких целей	Да	Да
Более 1000 локальных целей	Да	Да
Распознавание с помощью облака	Да	Да
SLAM	Да	Да
Поддержка Android ARM64	Да	Да
Поддержка пользовательских камер	Да	Да
Поддержка внешних алгоритмов	Да	Да
Отслеживание 3D-объектов	Нет	Да
Запись экрана	Нет	Да
Обнаружение и отслеживание целей разных видов (3D и плоские)	Нет	Да
Цена	Бесплатно	\$ 499

Для использования EasyAR необходима регистрация. Зарегистрируйтесь на www.easyar.com, используя свой адрес электронной почты, подтвердите ее и авторизуйтесь на сайте для продолжения работы.

Получение токена авторизации EasyAR

Для инициализации EasyAR SDK требуется ключ. Зайдите на сайт www.easyar.com и перейдите в свой аккаунт, как показано на рис. 5.

В боковом меню вашего профиля выберите пункт «SDK-> SDK Authorization» и нажмите на кнопку «Add SDK License Key».

В открывшемся окне в поле «Type» выберите один из трех типов лицензии:

- EasyAR SDK Basic. Free and no watermark;
- EasyAR SDK Pro. \$499/License key. One-time charge for lifetime use;
- Trail Version of EasyAR SDK Pro. Free, including Basic function, EasyAR SDK Pro can be activated 100 times daily while Basic is not limited.



Рис. 5. Главная страница EasyAR

Add SDK License Key

Type

[View SDK Feature Comparison](#)

- EasyAR SDK Basic
Free and no watermark
- EasyAR SDK Pro
\$499/License key. One-time charge for lifetime use.
- Trail Version of EasyAR SDK Pro
free, including Basic function, EasyAR SDK Pro can be activated 100 times daily while Basic is not limited.

Application Details

App Name

Supported Platforms iOS Android Windows, macOS

Bundle ID
iOS

Package Name
Android

1, SDK License key should be used with Bundle ID or Package Name correspondently
2, Bundle ID, PackageName can be modified after creation

Confirm

Рис. 6. Выбор версии EasyAR SDK

Для создания первого приложения достаточно выбрать EasyAR SDK Basic. Так как стандартная версия не имеет каких-либо ограничений и водяных знаков, накладываемых на выходное изображение. Результат выбора лицензии показан на рис. 6.

Задайте имя приложения в пункте «App Name». Название может быть любым, оно служит лишь для идентификации ключа в вашем меню профиля. В качестве примера введем название «MyFirstEasyARApp».

В пункте «Bundle ID» введите «0». «Bundle ID» используется для сборки приложения под устройства с операционной системой iOS.

«Package Name» – поле, необходимое для идентификации ключа при сборке приложения под устройства Android. Зададим значение этого поля «com.TSU.MyFirstEasyARApp». Важно, чтобы значение этого поля совпадало с «Package Name» приложения. Как задать это поле перед компиляцией в Unity, будет показано дальше.

В итоге заполненные поля должны выглядеть так же, как на рис. 7.

Add SDK License Key

Type

[View SDK Feature Comparison](#)

EasyAR SDK Basic
Free and no watermark

EasyAR SDK Pro
\$499/License key. One-time charge for lifetime use.

Trail Version of EasyAR SDK Pro
free, including Basic function, EasyAR SDK Pro can be activated 100 times daily while Basic is not limited.

Application Details

App Name

Supported Platforms iOS Android Windows, macOS

Bundle ID
iOS

Package Name
Android

1 SDK License key should be used with Bundle ID or Package Name correspondently
2 Bundle ID, PackageName can be modified after creation

[Confirm](#)

Рис. 7. Окно регистрации API-токена

После заполнения полей нажмите на кнопку «Confirm» и после загрузки откройте настройки только что созданного приложения в панели управления EasyAR. В разделе «SDK License Key» будет отображен сгенерированный ключ, который необходимо сохранить либо в текстовый файл, либо в буфер обмена путем нажатия на кнопку «Сору» справа от ключа.

Ключи нельзя использовать с несовпадающими версиями EasyAR SDK. Ключ EasyAR SDK 1.0 можно использовать только на EasyAR 1.0 и других версиях 1.0. Ключ EasyAR SDK 2.0 может использоваться только на EasyAR 2.0 и других версиях 2.0. Чтобы посмотреть ключ для более ранней версии EasyAR SDK, нажмите на кнопку «View Keys of EasyAR SDK 1.x, 2.x».

Ключ EasyAR Basic можно использовать только на EasyAR SDK Basic, но не на EasyAR SDK Pro. Если базовый ключ используется в EasyAR SDK Pro, вы получите ошибку «Invalid Key».

Ключ пробной версии EasyAR Pro можно использовать на EasyAR SDK Basic и EasyAR SDK Pro. Этот ключ позволит инициализировать SDK только 100 раз в день. Когда количество вызовов инициализации превысит ограничение, вы получите ошибку «Invalid Key».

Ключ EasyAR Pro можно использовать на EasyAR SDK Basic и EasyAR SDK Pro без ограничений.

EasyAR SDK Basic и EasyAR SDK Pro поставляются с приставками Basic или Pro, обязательно загрузите и используйте нужный пакет SDK.

Загрузка и настройка EasyAR SDK для Unity 3D

Для загрузки EasyAR SDK необходимо перейти на страницу www.easyar.com/view/download.html. На странице доступны три версии SDK, а именно «EasyAR SDK v*», «EasyAR SDK v* Unity Packages», «EasyAR SDK v* Native Samples», где «v*» обозначает версию SDK, к примеру «EasyAR SDK v3.0.1».

«EasyAR SDK v*» – это архив, который содержит библиотеку EasyAR SDK, скомпилированную для разных платформ (Android, Windows, iOS), а также импортом всех функций C++ в C#.

«EasyAR SDK v* Unity Packages» представляет собой файл формата «.unitypackage», содержащий ресурсы для импорта в Unity 3D. В данном архиве содержатся примеры сцен для быстрого запуска.

«EasyAR SDK v* Native Samples» – архив, который содержит примеры для запуска EasyAR SDK на разных платформах без использования игрового движка Unity.

Для запуска примеров на Unity необходимо загрузить архив вида «EasyAR SDK v* Unity Packages». Пример показан на рис. 8.

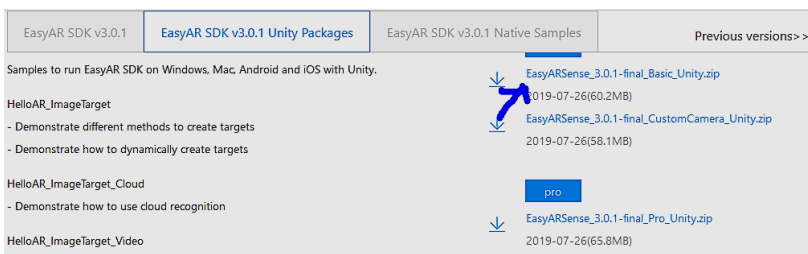


Рис. 8. Окно загрузки плагина Unity

Помимо архива «Basic Unity» на сайте доступен архив «Custom-Camera_Unity», данный пример необходимо использовать при подключении внешней USB-камеры, то есть камеры, которая не встроена в устройство по умолчанию. Для начала вполне подойдет версия «Basic».

После загрузки архив необходимо разархивировать в любое удобное место на компьютере. После распаковки в целевой папке будет находиться файл для импорта в Unity и текстовые файлы описания. Файл импорта будет использоваться после создания проекта.

Для создания проекта воспользуемся редактором Unity Hub, который мы установили ранее. Для этого нажмем кнопку «New», как показано на рис. 9.

После нажатия на кнопку вам будет предложено ввести название проекта, выбрать место размещения, а также шаблон. Путь расположения и название вы можете ввести самостоятельно, так, как вам удобно, данные поля не влияют на дальнейшую работу. Но шаблон приложения изменять не стоит. Для завершения нажмите на кнопку «Create» и дождитесь окончания загрузки.

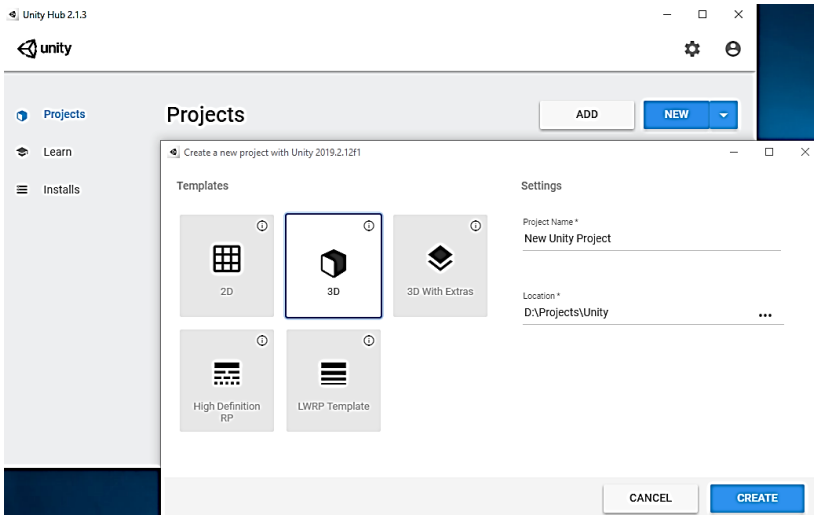


Рис. 9. Создание проекта в Unity

Для импорта пакета EasyAR SDK, загруженного ранее, переместите файл импорта на окно Unity. Вам будет предложено просмотреть импортируемые файлы, как показано на рис. 10.

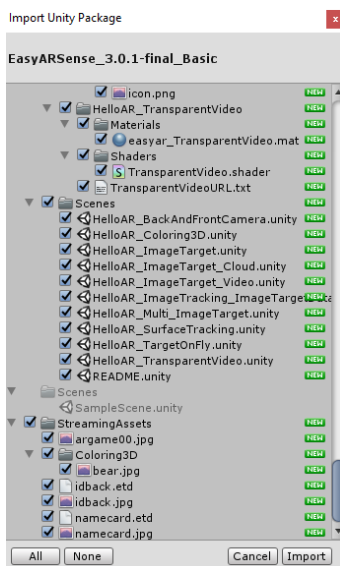


Рис. 10. Окно импорта Unity

Не отменяя выбора с файлов, нажмите на кнопку «Import» и дождитесь окончания загрузки, после чего в файловом менеджере Unity вы увидите новые директории. Структура отображена на рис. 11.

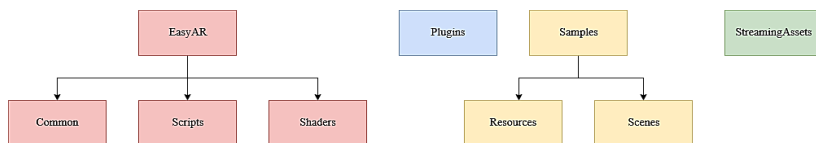


Рис. 11. Структура EasyAR SDK в Unity

EasyAR – содержит ресурсы и скомпилированные библиотеки EasyAR.

Common – общие ресурсы. Обратите внимание: данный раздел содержит EasyARSetting, который используется для ввода ключа.

Scripts – примеры кода ядра EasyAR и файл интерфейса API низкого уровня csapi.cs.

Shaders – шейдер, используемый для вывода фона камеры, префаб куба и видео.

Plugins – двоичные файлы для платформ Android / iOS / Windows / Mac и код взаимодействия.

Samples – примеры ресурсов и кода.

Resources – ресурсы, используемые в сценах примеров.

Scenes – примеры сцен.

StreamingAssets – некомпиллируемые Unity файлы ресурсов, которые можно использовать для загрузки в качестве целевых данных.

Для работы EasyAR необходим лицензионный ключ. Вам нужно найти EasyARKey и вставить ключ в панель инспектора. Процесс получения ключа был описан в пункте «Получение токена авторизации EasyAR».

Для активации SDK внутри Unity откроем директорию «EasyAR-> Common-> Resources», в которой находится ресурс «EasyARKey». Открыв его щелчком мыши, в инспекторе Unity увидим поле, как на рис. 12.

В поле «Easy AR Key» необходимо скопировать созданный ранее ключ.

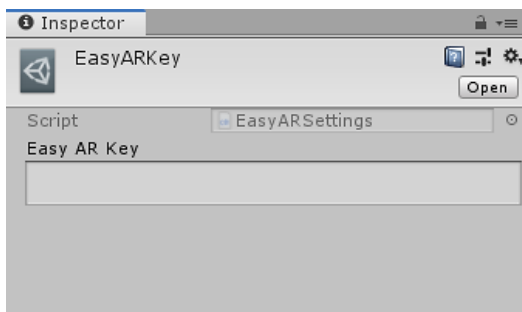


Рис. 12. Поля для импорта API-токена

Запуск и компиляция приложения для операционной системы Android

Чтобы скомпилировать приложение под операционную систему Android, необходимо изменить целевую платформу в Unity. Для этого в открытом проекте откройте пункт «File-> Build Settings». Откроется окно, как показано на рис. 13.

В диалоговом окне выберите пункт «Android» и нажмите на кнопку «Switch platform». Начнется процесс перекомпиляции скриптов под операционную систему Android. По завершении нажмите на кнопку «Player Settings», откроется новое диалоговое окно с настройками проекта. На данном этапе необходимо изменить поля в пункте «Other Settings».

В разделе «Graphics APIs» удалите пункт «Vulkan», должно получиться как на рис. 14.

При компиляции под Android снимите выбор с пункта «Multithreaded Rendering».

В подпункте «Identification» необходимо задать «Package name» и минимальную версию Android API Level. Для EasyAR минимально поддерживаемая версия API – Level 17 и более новые. Чтобы изменить минимальный API-уровень, в выпадающем списке пункта «Minimum API Level» выберите «Android 4.2». Этого будет достаточно для успешной компиляции.

Измените поле «Package Name» в соответствии с тем, что вы вводили при создании ключа EasyAR («com.TSU.MyFirstEasyARApp»). Результат показан на рис. 15.

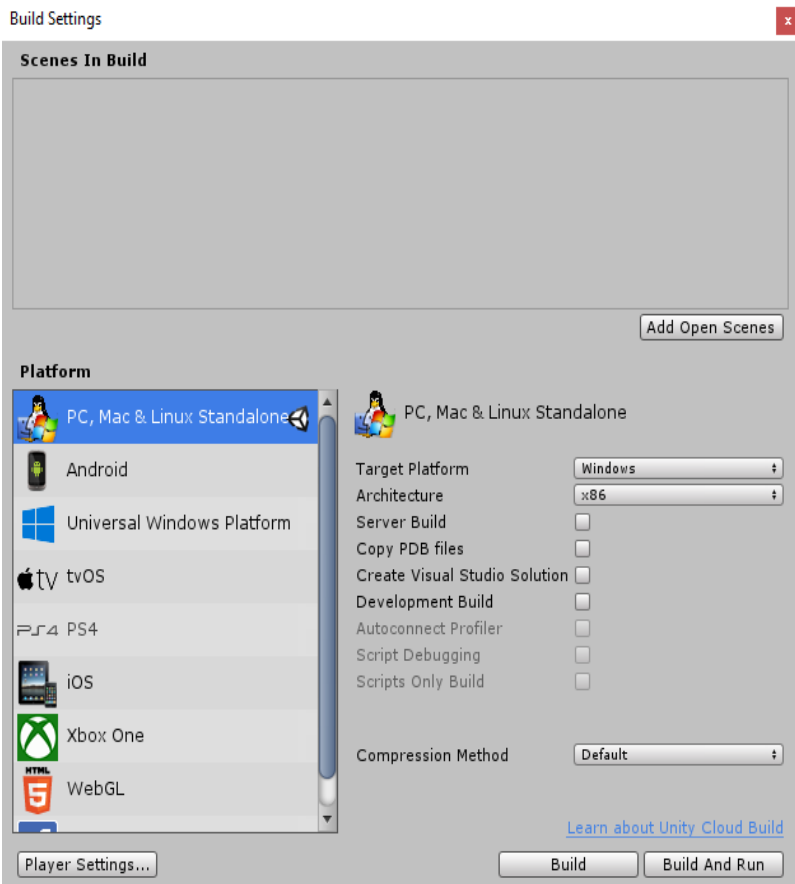


Рис. 13. Окно настройки Unity

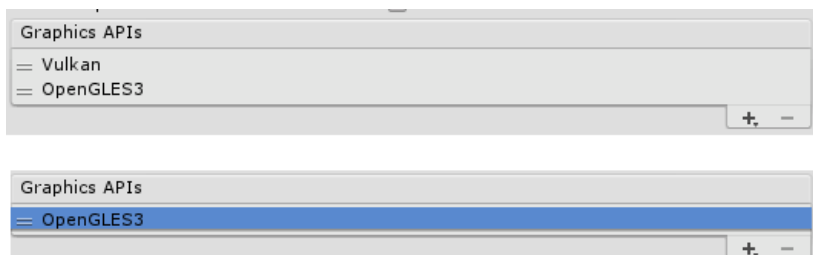


Рис. 14. Окно настройки Graphics APIs

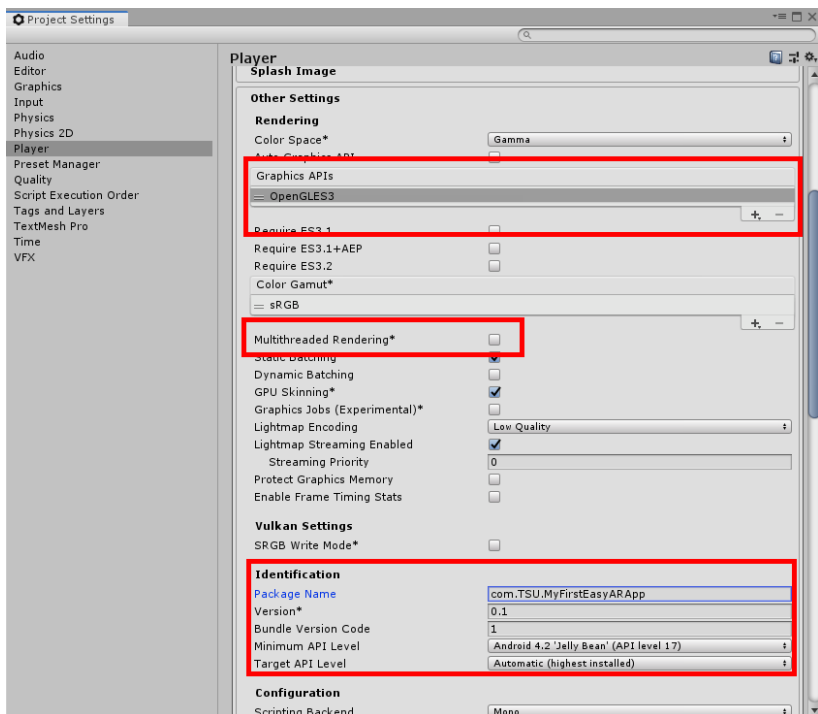


Рис. 15. Окно настройки проекта Unity

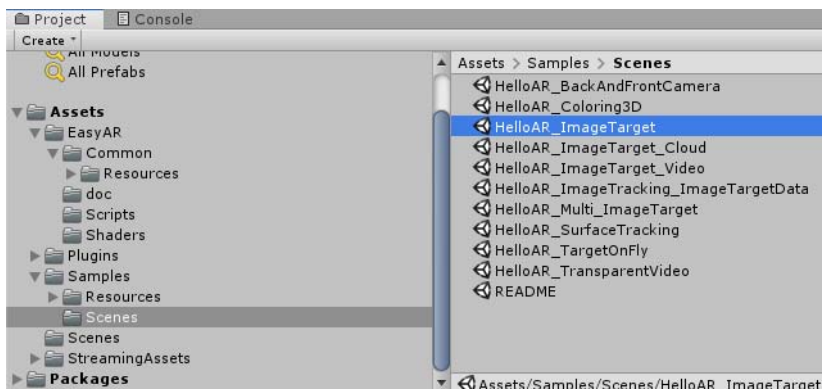


Рис. 16. Выбор сцены Unity

Проект готов к сборке под Android, осталось открыть сцену из раздела с примерами и сделать ее начальной в приложении.

В файловом менеджере выберите пункт «Scenes» и откройте сцену «HelloAR_ImageTarget», как показано на рис. 16.

В меню Unity выберите раздел «File-> Build Settings» и нажмите кнопку «Add Open Scenes». В списке «Scenes In Build» должна появиться открытая сцена. После чего вы можете собрать приложение под ваше устройство и запустить его.

Для обычной сборки приложения в открытом окне «Build Settings» нажмите кнопку «Build», в открывшемся диалоговом окне выберите место сохранения APK-файла и нажмите кнопку «Сохранить». Начнется компиляция приложения. По завершении вы можете скопировать полученный файл на ваше устройство, установить его и запустить.

Если вы хотите собрать приложение и немедленно запустить его на подключенном по USB устройстве, то нажмите кнопку «Build and Run», выполните все действия, описанные в предыдущем абзаце, и ожидайте конца компиляции, после чего на вашем устройстве будет запущено приложение.

Для проверки работоспособности вашего приложения вы можете использовать картинку, представленную на рис. 17. Просто наведите на нее ваше устройство.



Рис. 17. Изображение маркера для тестового стенда EasyAR

После попадания картинка в поле видимости камеры вашего устройства на дисплее должна наблюдаться 3D-фигура, которая была выбрана на предыдущих этапах.

Практическое задание

В данном практическом задании предлагается создать приложение дополненной реальности с использованием платформы разработки в реальном времени Unity совместно с API, предоставляемым EasyAR.

Задание

Создать приложение, которое будет распознавать стандартный маркер EasyAR. Поверх маркера необходимо вывести любую простую трехмерную фигуру (куб, шар, цилиндр и т. д.).

Результатом выполнения задания может являться обычный исполняемый файл .exe или архивный исполняемый файл .apk для Android-устройств.

Для выполнения задания необходимо:

- загрузить файл формата unitypackage с официального цифрового ресурса EasyAR, этот файл далее будет именоваться как «плагин»;
- создать новый проект в Unity и открыть его;
- импортировать загруженный плагин в созданную сцену;
- выполнить регистрацию на сайте EasyAR;
- создать новый токен для активации EasyAR;
- произвести активацию EasyAR внутри Unity;
- выбрать сцену с демонстрацией работы распознавания одного маркера, при необходимости заменить трехмерную фигуру;
- скомпилировать приложение.

Проверить работоспособность приложения можно, используя изображение, представленное на рис. 17.

Создание приложения дополненной реальности с использованием программной платформы Vuforia

Vuforia Engine – это программная платформа для создания приложений дополненной реальности. Разработчики могут легко добавить функциональность компьютерного зрения в любое приложение, позволяя ему распознавать изображения и объекты и взаимодействовать с виртуальным пространством в реальном мире.

Платформа Vuforia Engine поддерживает разработку приложений AR для устройств Android, iOS и UWP.

В Unity 2019.2 и более поздних версиях пакет Vuforia Engine будет автоматически добавлен в ваш проект.

Для начала разработки с использованием Vuforia необходимо создать проект, для этого перейдите в Unity Hub и создайте новый проект, как показано на рис. 18.

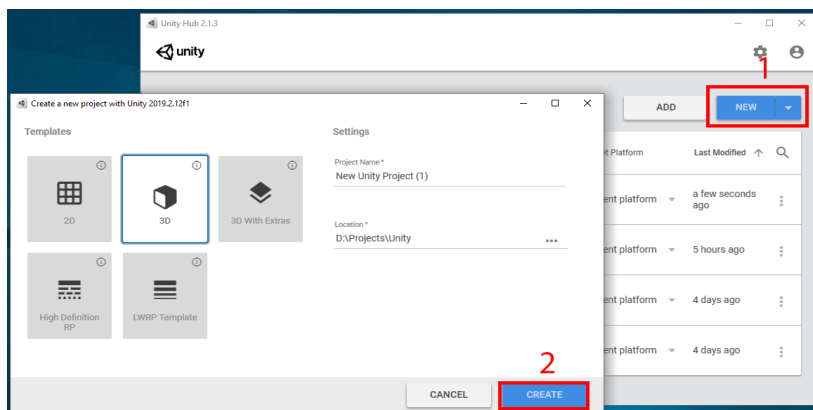


Рис. 18. Создание нового проекта Unity

Откройте только что созданный проект и воспользуйтесь пунктом меню «File-> Build Settings-> Player Settings» для активации нужного пакета. Найдите подпункт «XR Settings» и отметьте пункты, как показано на рис. 19.

Так как целевой платформой является операционная система Android, необходимо выбрать соответствующий пункт в настройках проекта. Для этого воспользуйтесь меню «File-> Build Settings». В новом окне в разделе «Platform» выберите «Android», нажмите

кнопку «Switch Platform» и дождитесь окончания конвертации проекта. По завершении получится результат, представленный на рис. 20.

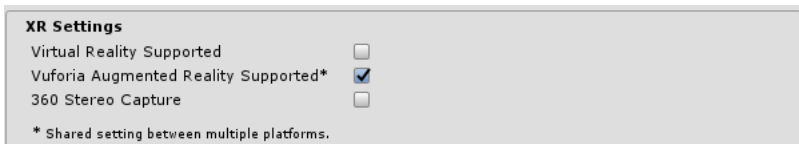


Рис. 19. Окно настройки XR в Unity

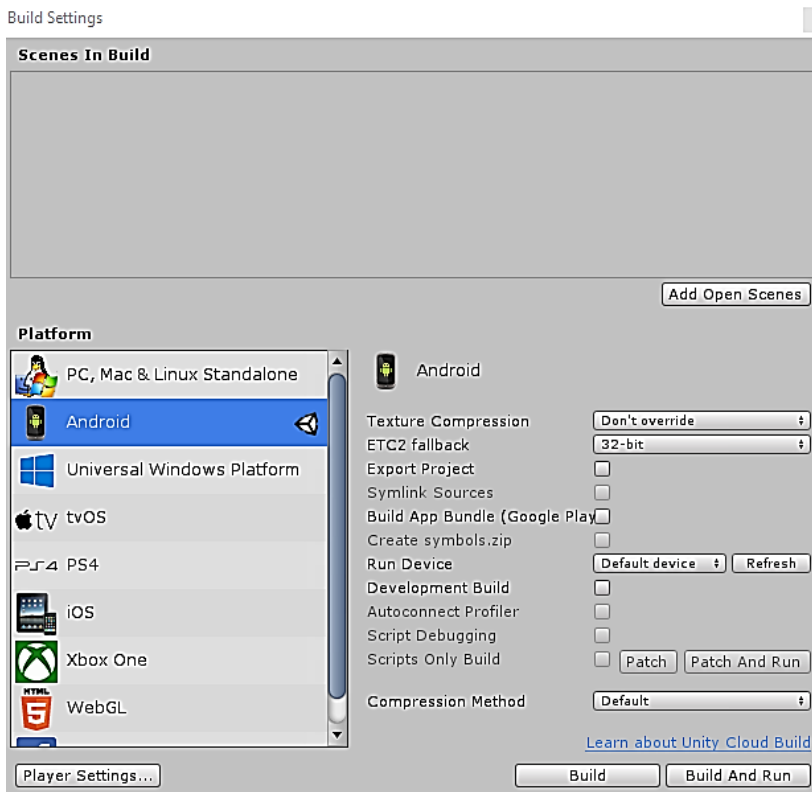


Рис. 20. Настройки сцены Unity

Активация Vuforia Engine завершена, приступим к созданию простой сцены для демонстрации на мобильном устройстве. Вначале необходимо добавить специальную камеру, поставляемую Vuforia

Engine. Для этого перейдите в раздел меню «GameObject-> Vuforia Engine» и выберите «AR Camera», как показано на рис. 21. Если это меню не отображается, это означает, что вы не установили Vuforia с Unity (версии Unity до 2019.2) или не добавили пакет Vuforia Engine в свой проект (Unity 2019.2 и более поздние версии).

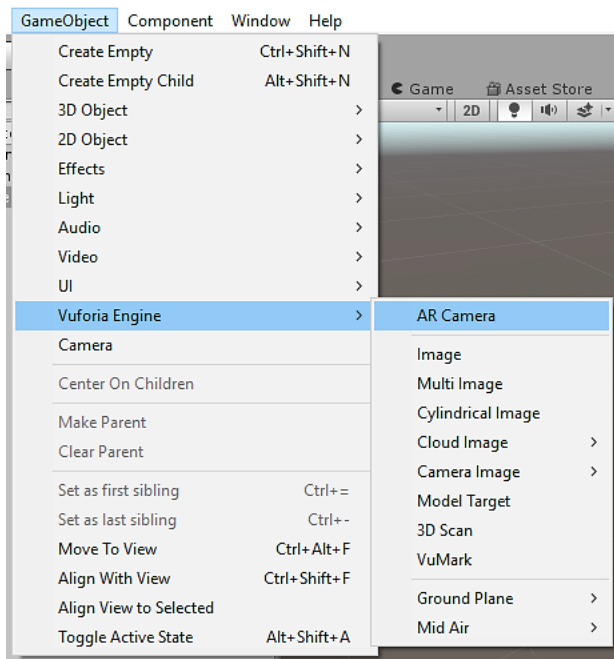


Рис. 21. Окно выбора камеры Unity

Для работы камеры необходимо целевое изображение, которое будет выступать в качестве маркера в будущем приложении. Таким маркером может быть любое изображение, но желательно, чтобы оно содержало как можно больше деталей. Vuforia Engine поставляется с базой данных стандартных изображений, которые мы можем использовать для демонстрации. Воспользуемся ими. Для этого, как в предыдущем пункте, откроем меню «GameObject-> Vuforia Engine-> Image». Начнется процесс импорта, по окончании которого в нашем проекте появятся новые элементы, как показано на рис. 22.

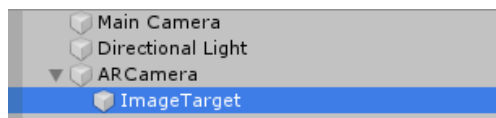


Рис. 22. Выбор элемента в инспекторе

«ImageTarget» в данном случае служит как цель (маркер). Цель и камера имеются, осталось добавить 3D-объект для вывода его поверх распознаваемого изображения. В качестве объекта будет использоваться стандартный куб Unity. Для его добавления достаточно «закрепить» объект под «ImageTarget». Нажмите правой клавишей мыши (ПКМ) на целевое изображение в иерархии объектов, выберите пункт «3D Object-> Cube». На сцене появится белый куб, уменьшите его размеры на ваш вкус. В итоге иерархия и игровая сцена должны выглядеть как на рис. 23.

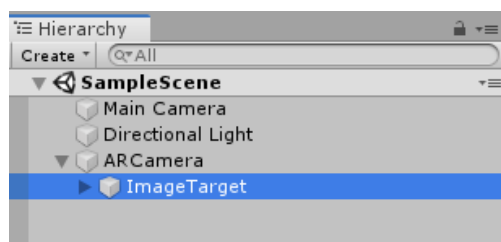


Рис. 23. Окно инспектора Unity

После сборки проекта запустите на вашем устройстве полученное приложение. При наведении на ранее созданный маркер должна появиться 3D-модель, которую также выбирали ранее.

Практическое задание

В данном практическом задании предлагается создать приложение дополненной реальности с использованием платформ разработки в реальном времени Unity и Vuforia.

Задание

Создать приложение, которое будет распознавать стандартные маркеры Vuforia. Поверх маркера необходимо вывести любую простую трехмерную фигуру (куб, шар, цилиндр и т. д.).

Обратите внимание, необходимо создать приложение, способное распознавать несколько маркеров, находящихся в фокусе камеры одновременно, а не один, как в предыдущем задании.

В качестве маркеров необходимо использовать маркеры, представленные на рис. 24 и 25.

Результатом выполнения задания может являться обычный исполняемый файл .exe или архивный исполняемый файл .apk для Android-устройств.

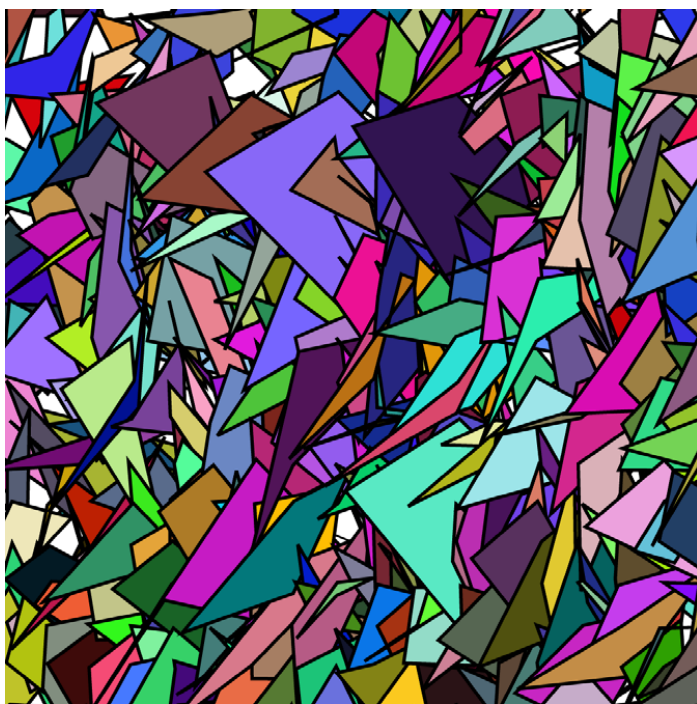


Рис. 24. Первый маркер для практического задания

Так как Vuforia идет в комплекте с Unity, отсутствует необходимость в дополнительных действиях по ее загрузке.

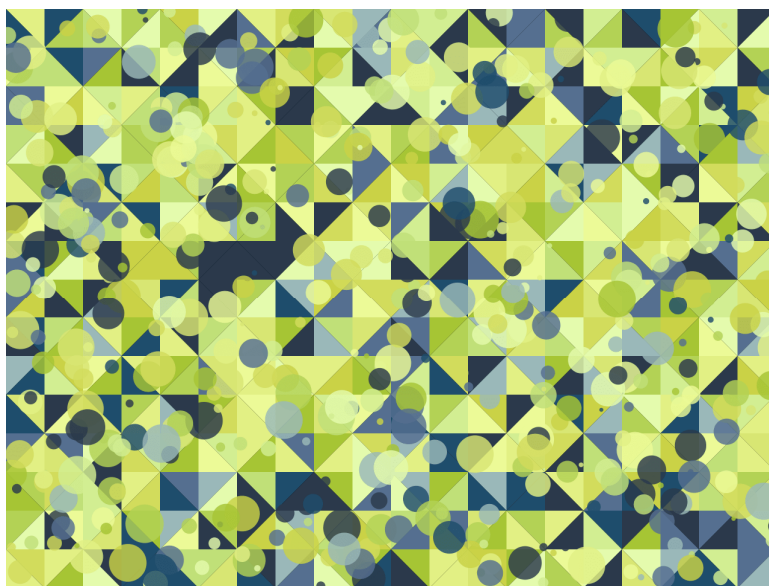


Рис. 25. Второй маркер для практического задания

Проверить работоспособность приложения можно, используя представленные выше изображения.

Чтобы выгрузить маркеры для практического задания из данного файла, нажмите на изображение ПКМ и выберите пункт «Сохранить как рисунок...». Текст пункта может отличаться в зависимости от версии используемого ПО.

Создание приложения дополненной реальности с использованием программной платформы ARToolKit

Для создания приложения необходима *среда разработки* – комплекс программных средств, используемый для создания программных продуктов.

Мы будем использовать *Unity 3D* – платформу для разработки 3D-, 2D-, VR- и AR-игр и приложений.

Нам понадобится версия Unity 5.5.4f1. Для ее установки необходимо:

- зайти на сайт Unity в раздел загрузок разных версий <https://unity3d.com/ru/get-unity/download/archive>;
- выбрать вкладку Unity 5.x и найти версию 5.5.4.

После выполнения указанных действий должна отобразиться страница, как показано на рис. 26.

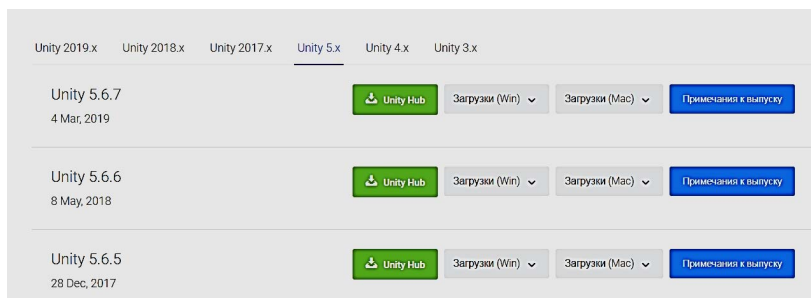


Рис. 26. Архив версий Unity

Далее необходимо:

- выбрать установщик Unity, как показано на рис. 27;
- запустить установщик и установить, ничего не меняя;
- зарегистрироваться на сайте <https://id.unity.com/en/conversations/683851d8-94ea-4514-a07b-0eb8d74452a901bf> и ввести логин и пароль в приложении после запуска;
- создать новый проект, как это показано на рис. 28;
- нажать «New». Ввести любое название и указать любую директорию, где проект будет храниться.

Далее откроется рабочее окно, показанное на рис. 29.

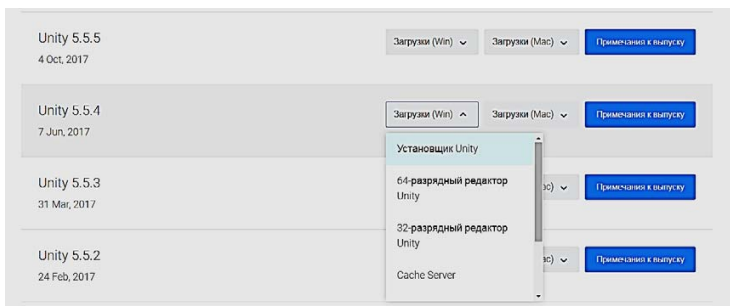


Рис. 27. Окно выбора варианта загрузки

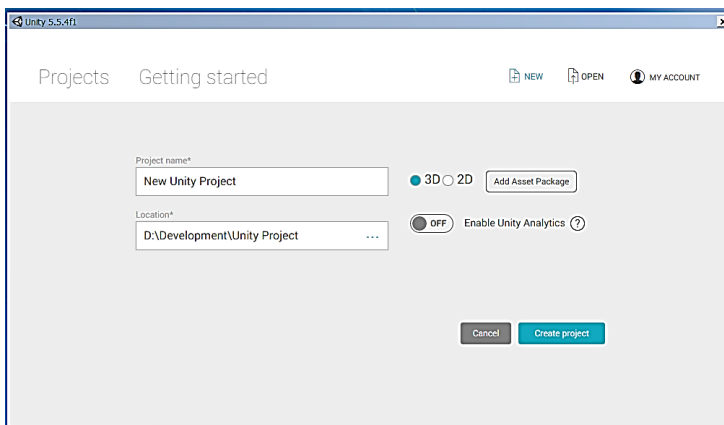


Рис. 28. Окно создания проекта Unity

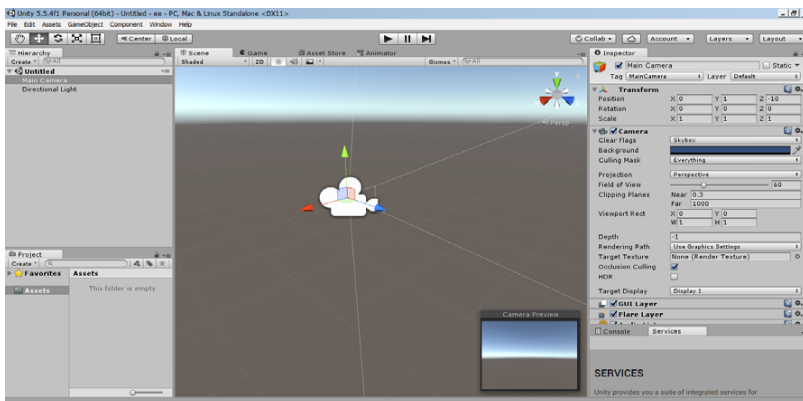


Рис. 29. Главное окно Unity

Среда разработки имеет следующий вид: два окна сцены, на которых располагаются объекты (в одном – списком, в другом – на 3D-плоскости), окно «Console» с системной информацией, окно «Inspector» для редактирования свойств объектов и окно «Project» для манипулирования исходными файлами.

Загрузка и импорт программной платформы ARToolKit

Чтобы создать AR-приложение, понадобится *библиотека для разработки* – готовый программный код, который можно использовать во время разработки. Проще говоря, инструмент.

Мы будем использовать ARToolKit. Его необходимо скачать на сайте <https://github.com/artoolkit/arunity5>.

Нажать кнопку «Clone or Download», затем «Download ZIP». Должно получиться, как показано на рис. 30.

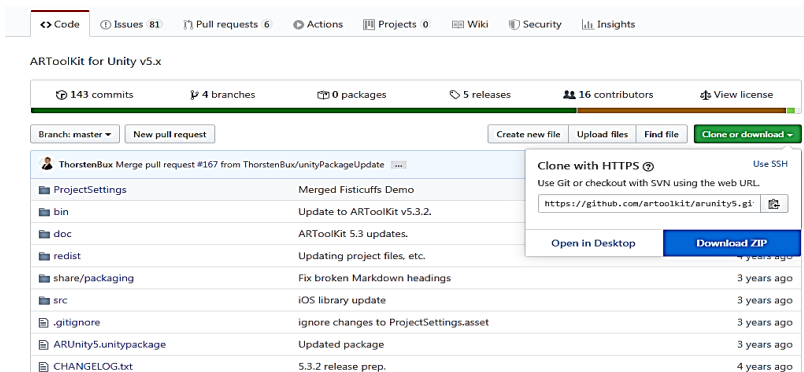


Рис. 30. Резепозиторий проекта ARToolKit

Распаковать папку «arunity5-master» в любое место.

Теперь нужно подключить библиотеку к проекту.

В окне «Project» нажать ПКМ на «Assets», выбрать пункт «Import Package», затем «Custom Package», далее выбрать распакованную папку, файл пакета, как это показано на рис. 31.

В следующем окне нажать кнопку «Import». Откроется окно менеджера, где необходимо выбрать все файлы, как показано на рис. 32. Для удобства можно нажать кнопку «All», что приведет к автоматическому выделению всех имеющихся элементов.

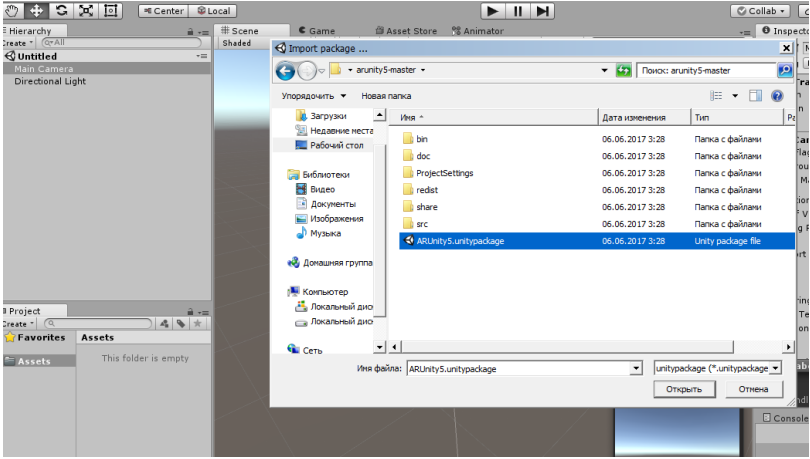


Рис. 31. Окно выбора пакета ARToolKit

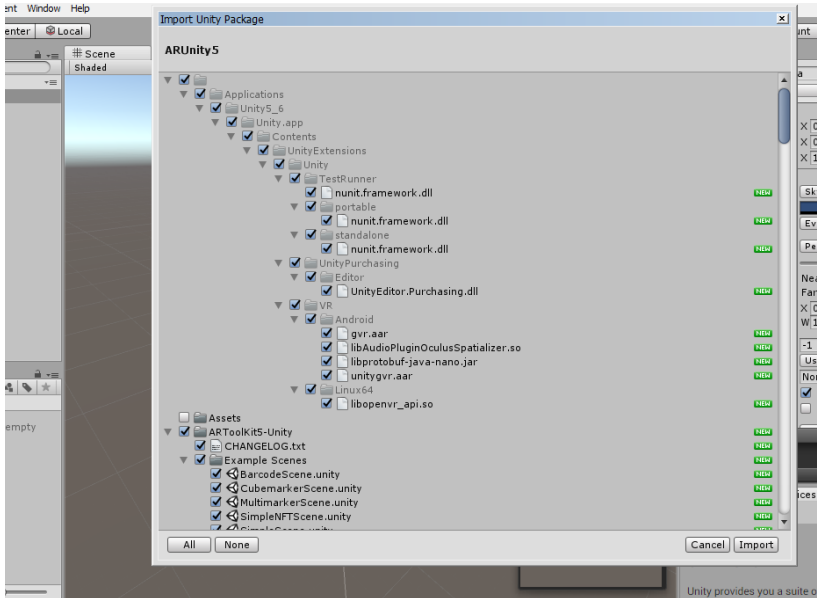


Рис. 32. Окно импорта ARToolKit

После окончания импорта библиотека ARToolKit готова к использованию. Для дальнейшего использования необходимо настроить сцену в Unity 3D, что будет сделано в следующем разделе.

Настройка проекта Unity 3D с использованием программной платформы ARToolKit

В первую очередь удалить со сцены все объекты, создать пустой объект (команда «Create Empty»), назвать ARToolKit. В окне «Inspector» добавить скрипт «AR Controller». Для него в окне «Camera parameters» прописать «camera_raga», в окне «Layer» выбрать пункт «AR background» (если нет в списке, тогда выбрать элемент «Add Layer» и прописать в одном из пустых полей), все остальное оставить по умолчанию. Помимо этого, выбрать пункт «Layer» — «AR background» в самом объекте, в верхней части окна «Inspector». Также надо добавить скрипт «AR Marker», написать тег, по которому мы будем в дальнейшем обращаться к этому маркеру, и в поле «Pattern File» указать сам маркер в формате patt.

Далее создать и поместить внутрь ARToolKit другой пустой объект и назвать «Scene root». К нему добавить скрипт «AR Origin».

Внутри «Scene root» создать объект «Camera» и добавить к нему «AR Camera». В параметре «Culling Mask» установить — «AR Foreground». Там же создать пустой объект «Marker» — это то, что накладывается на маркер. Добавить к нему элемент «AR Tracked Object», в параметрах указать нужный тег маркера из библиотеки «AR Marker» в ARToolKit. Внутри «Marker» можно помещать любой объект, который нужно наложить. Для изменения его размеров или положения относительно маркера следует работать со сценой в виде 3D-плоскости.

После всего выбрать объект «Scene root» и установить в окне «Layer» метку «AR Foreground», согласиться на изменение этого параметра для всех дочерних объектов.

Получается структура, изображенная на рис. 33.

После выполнения указанных действий будет создана сцена, на которой к определенному маркеру будет добавлен 3D-объект. Этих действий достаточно для первого запуска приложения и демонстрации его работы.

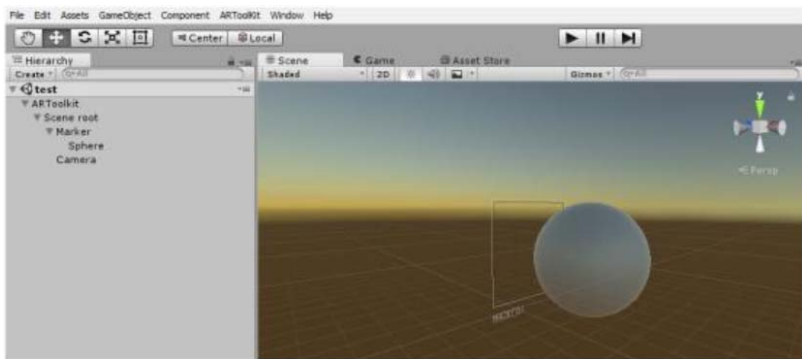


Рис. 33. Структура проекта

Запуск и компиляция приложения для операционной системы Android

Выбрать пункт «File / Build Settings». Выбрать подходящую платформу, выбрать настроенную сцену. В пункте «Player Settings» есть возможность настройки приложения. Начать сборку с помощью опции «Build». Результат работы отображен на рис. 34.

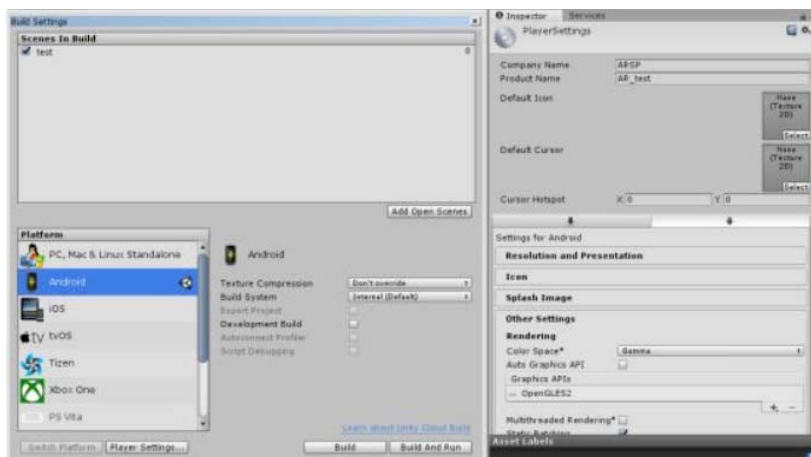


Рис. 34. Окно настройки приложения

В случае с Android-приложением после сборки получается установочный файл (расширение apk). Пример работы приложения отображен на рис. 35.

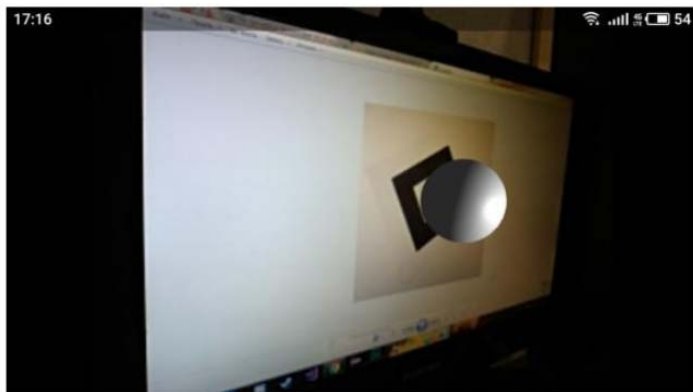


Рис. 35. Пример работы приложения

При загрузке и запуске скомпилированного приложения на вашем устройстве наведите камеру на маркер. Поверх определенного ранее маркера должен выводиться 3D-объект.

На этом этапе настройки инструментов дополненной реальности закончен. Далее необходимо рассмотреть работу с нейронными сетями, для того чтобы перейти от определения заранее заложенных маркеров к модели распознавания объектов. В таком режиме приложение ориентируется не на запрограммированное изображение, а на распознавание объектов в реальном мире. Что дает большую свободу действий, но усложняет процесс разработки. Более подробно работа с нейронными сетями будет рассмотрена в следующей главе.

Практическое задание

В данном практическом задании предлагается создать приложение дополненной реальности с использованием платформ разработки в реальном времени Unity и ARToolKit.

Задание

Создать приложение, которое будет распознавать стандартный маркер ARToolKit. Поверх маркера необходимо вывести любую простую трехмерную фигуру (куб, шар, цилиндр и т. д.).

Данное задание является наиболее сложным, так как присутствует необходимость самостоятельной ручной работы с архивами и исходными файлами библиотеки.

В качестве маркеров необходимо использовать маркер, представленный на рис. 36.

Результатом выполнения задания должен быть архивный исполняемый файл .apk для Android-устройств.



Рис. 36. Маркер для практического задания

Проверить работоспособность приложения можно, используя представленное выше изображение.

Чтобы выгрузить маркеры для практического задания из данного файла, нажмите на изображение ПКМ и выберите пункт «Сохранить как рисунок...». Текст пункта может отличаться в зависимости от версии используемого ПО.

Распознавание объектов на Android с помощью модели TensorFlow

Обучение нейросети распознаванию образов — долгий и ресурсоемкий процесс. Особенно когда под рукой есть только недорогой ноутбук, а не компьютер с мощной видеокартой. В этом случае на помощь придет программа Google Colaboratory, которая предлагает совершенно бесплатно воспользоваться GPU уровня Tesla K80.

Подготовка данных

Нашей целью будет обучить нейросеть распознавать детали компьютера. Соответственно, для начала надо создать набор данных, достаточный для обучения (~200 фото). Пример набора данных отображен на рис. 37.



Рис. 37. Изображения видеокарт

Для обучения воспользуемся моделью «TensorFlow Object Detection API». Все необходимые для обучения данные мы подготовим на ноутбуке. Нам понадобится менеджер управления окружением и зависимостями **conda**. Инструкция по установке взята с сайта <https://docs.conda.io/projects/conda/en/latest/user-guide/install/index.html>.

Создадим окружение для работы:

```
conda create -n object_detection_prepare pip python=3.6
```

И активируем его:

```
conda activate object_detection_prepare
```

Установим зависимости, которые нам понадобятся:

```
pip install --ignore-installed --upgrade tensorflow==1.14
```

```
pip install --ignore-installed pandas
```

```
pip install --ignore-installed Pillow
```

```
pip install lxml
```

```
conda install pyqt=5
```

Создадим папку **object_detection** и положим все наши фотографии в папку **object_detection/images**.

В Google Colab есть ограничение на использование памяти, поэтому перед разметкой данных нужно снизить разрешение фотографий, чтобы в процессе обучения не столкнуться с ошибкой «**tcmalloc: large alloc...**».

Создадим папку **object_detection/preprocessing** и добавим в нее подготовленные скрипты.

Для изменения размера фото используем скрипт

```
python ./object_detection/preprocessing/image_resize.py -i ./object_detection/images --imageWidth=800 --imageHeight=600
```

Этот скрипт пробежится по папке с указанными фото, изменит их размер до 800×600 и сложит их в **object_detection/images/resized**. Теперь можно заменить ими оригинальные фотографии в **object_detection/images**.

Для разметки данных воспользуемся тулзой **labelImg**.

Клонируем репозиторий **labelImg** в **object_detection**.

Переходим в папку **labelImg**

```
cd [FULL_PATH]/object_detection/labelImg
```

и выполняем команду **pyrcc5 -o libs/resources.py resources.qrc**.

После этого можно приступить к разметке данных (это самый долгий и скучный этап):

```
python labelImg.py
```

Откроется окно, как показано на рис. 38.

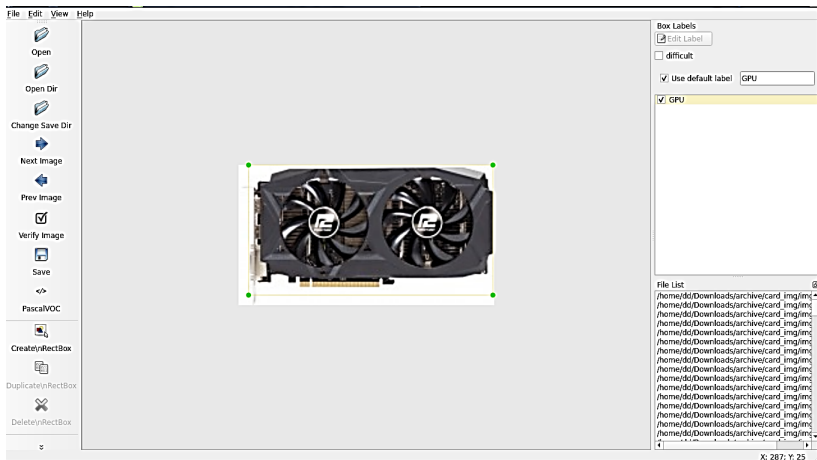


Рис. 38. Окно разметки данных

В поле «Open dir» указываем папку **object_detection/images** и проходим по всем фото, выделяя объекты для распознавания и указывая их класс. В нашем случае это комплектующие компьютера. Сохраним метаданные (файлы *.xml) в той же папке, как показано на рис. 39.

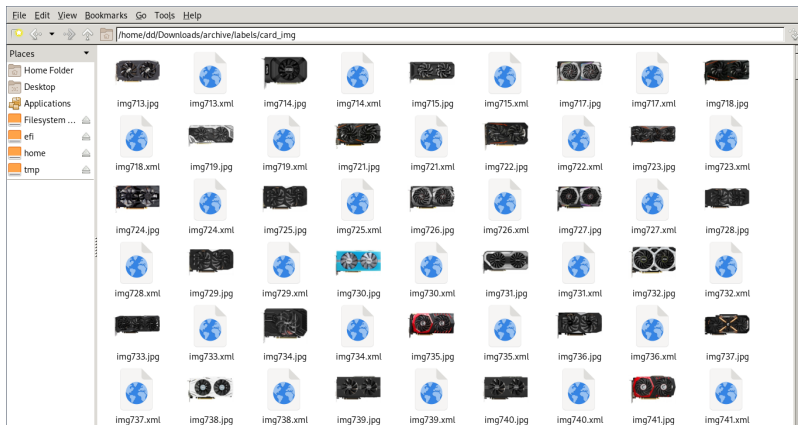


Рис. 39. Окно сохранения метаданных

Создадим папку **object_detection/training_demo**, которую мы чуть позже зальем в Google Colab для обучения.

Разделим наши фото (с метаданными) на тренировочные и тестовые в соотношении **80/20** и переместим их в соответствующие папки: **object_detection/training_demo/images/train** и **object_detection/training_demo/images/test**.

Создадим папку **object_detection/training_demo/annotations**, в которую будем складывать файлы с метаданными, необходимыми для обучения. Первым из них будет **label_map.pbtxt**, в котором укажем отношение класса объекта и целочисленного значения. В нашем случае это **label_map.pbtxt**.

Чтобы использовать метаданные, полученные в процессе разметки данных, для обучения, необходимо конвертировать их в формат TFRecord. Для конвертации воспользуемся скриптами из репозитория: https://github.com/Swaini/object_detection_retraining.

Проведем конвертацию в два этапа: **xml -> csv** и **csv -> record**.

Перейдем в папку **preprocessing**:

cd [FULL_PATH]\object_detection\preprocessing

– Из xml в csv. Тренировочные данные: **python xml_to_csv.py -i [FULL_PATH]/object_detection/training_demo/images/train -o [FULL_PATH]/object_detection/training_demo/annotations/train_labels.csv**

Тестовые данные: **python xml_to_csv.py -i [FULL_PATH]/object_detection/training_demo/images/test -o [FULL_PATH]/object_detection/training_demo/annotations/test_labels.csv**

– Из csv в record. Тренировочные данные: **python generate_tfrecord.py --label_map_path=[FULL_PATH]\object_detection\training_demo\annotations\label_map.pbtxt --csv_input=[FULL_PATH]\object_detection\training_demo\annotations\train_labels.csv --output_path=[FULL_PATH]\object_detection\training_demo\annotations\train.record --img_path=[FULL_PATH]\object_detection\training_demo\images\train**

Тестовые данные: **python generate_tfrecord.py --label_map_path=[FULL_PATH]\object_detection\training_demo\annotations\label_map.pbtxt --csv_input=[FULL_PATH]\object_detection\training_demo\annotations\test_labels.csv --output_path=[FULL_PATH]\object_detection\training_demo\annotations\test.record --img_path=[FULL_PATH]\object_detection\training_demo\images\test**

На этом мы закончили подготовку данных, теперь надо выбрать модель, которую будем обучать.

Сейчас мы выберем модель `ssdlite_mobilenet_v2_coco`, чтобы в дальнейшем запустить обученную модель на Android-устройстве.

Скачиваем архив с моделью и распаковываем его в `object_detection/training_demo/pre-trained-model`.

Должно получиться что-то вроде:

`object_detection/training_demo/pre-trained-model/ssdlite_mobilenet_v2_coco_2018_05_09`.

Из распакованного архива копируем файл `pipeline.config` в `object_detection/training_demo/training` и переименовываем его в `ssdlite_mobilenet_v2_coco.config`.

Далее нам надо настроить его под свою задачу, для этого укажем:

– количество классов `model.ssd.num_classes: 9`;

– размер пакета (количество данных для обучения за одну итерацию), количество итераций и путь к сохраненной модели из архива, который мы скачали:

`train_config.batch_size: 18`

`train_config.num_steps: 20000`

`train_config.fine_tune_checkpoint: "/training_demo/pre-trained-model/ssdlite_mobilenet_v2_coco_2018_05_09/model.ckpt"`;

– количество фото в тренировочном наборе (`object_detection/training_demo/images/train`)

`eval_config.num_examples: 64`;

– путь к набору данных для тренировки

`train_input_reader.label_map_path: "/training_demo/annotations/label_map.pbtxt"`

`train_input_reader.tf_record_input_reader.input_path: "/training_demo/annotations/train.record"`;

– путь к тестовому набору данных

`eval_input_reader.label_map_path: "/training_demo/annotations/label_map.pbtxt"`

`eval_input_reader.tf_record_input_reader.input_path: "/training_demo/annotations/test.record"`.

Далее архивируем папку `training_demo` и получившийся `training_demo.zip` размещаем в Google Drive.

На этом подготовка данных закончена, перейдем к обучению.

Обучение модели

В Google Drive выбираем файл **training_demo.zip**, нажимаем на кнопку **Get shareable link** и из полученной ссылки сохраним себе **id** этого файла:

drive.google.com/open?id=[YOUR_FILE_ID_HERE]

Самый простой способ воспользоваться Google Colab – создать новый блокнот в Google Drive, как показано на рис. 40.

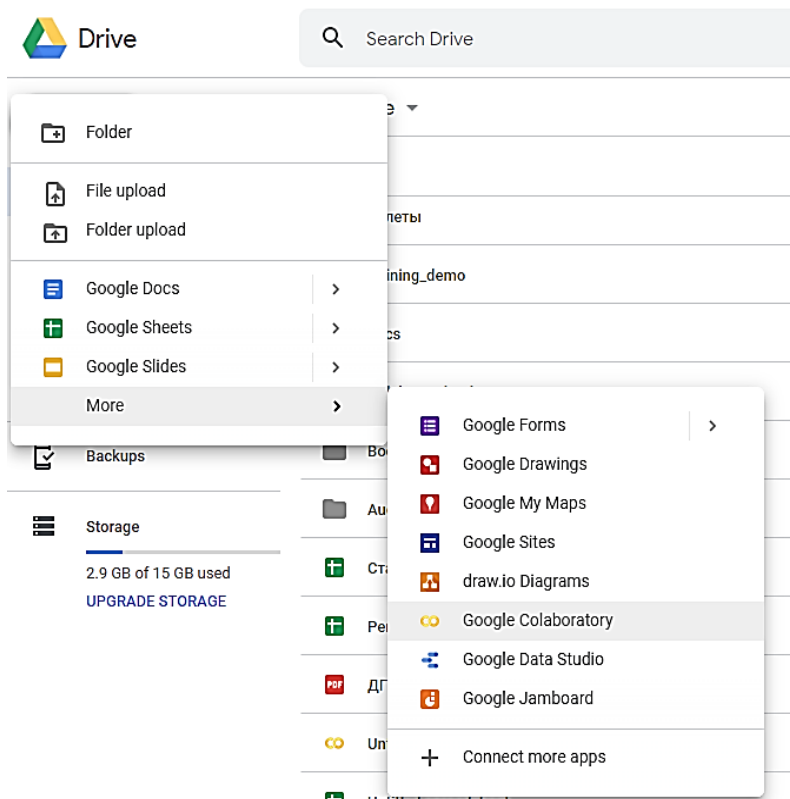


Рис. 40. Окно выбора Google Colab

По умолчанию обучение будет выполняться на CPU. Чтобы использовать GPU, нужно поменять тип runtime, как показано на рис. 41.

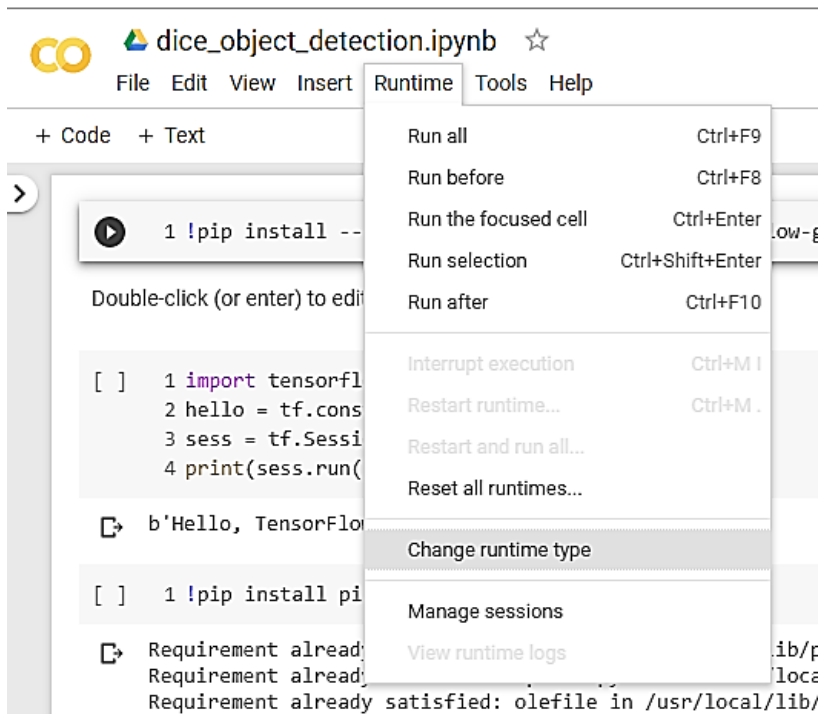


Рис. 41. Выбор runtime в меню

Готовый блокнот можно взять [тут](#).

Обучение состоит из следующих этапов:

- клонируем репозиторий TensorFlow Models;
- `!git clone https://github.com/tensorflow/models.git`
- устанавливаем protobuf и компилируем необходимые файлы

в `object_detection`:

```
!apt-get -qq install libprotobuf-java protobuf-compiler
```

```
%cd ./models/research/
```

```
!protoc object_detection/protos/*.proto --python_out=.
```

```
%cd ../..
```

- добавляем необходимые пути в переменную окружения

`PYTHONPATH`:

```
import os
```

```
os.environ['PYTHONPATH'] += ":/content/models/research/"
```

```
os.environ['PYTHONPATH'] += ":/content/models/research/slim"
```

```
os.environ['PYTHONPATH'] += ":/content/models/research/object_
detection"
```

```
os.environ['PYTHONPATH'] += ":/content/models/research/object_
detection/utils"
```

– для получения файла из Google Drive устанавливаем PyDrive и авторизируемся:

```
!pip install -U -q PyDrive
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from google.colab import auth
from oauth2client.client import GoogleCredentials
auth.authenticate_user()
gauth = GoogleAuth()
gauth.credentials = GoogleCredentials.get_application_default()
drive = GoogleDrive(gauth)
```

– скачиваем архив (нужно указать id вашего файла) и разархивируем его:

```
drive_file_id="[YOUR_FILE_ID_HERE]"
training_demo_zip = drive.CreateFile({'id': drive_file_id})
training_demo_zip.GetContentFile("training_demo.zip")
!unzip training_demo.zip
!rm training_demo.zip
```

– запускаем процесс обучения:

```
!python ./models/research/object_detection/legacy/train.py --logtostderr
--train_dir=./training_demo/training --pipeline_config_path=./training_
demo/training/ssdlite_mobilenet_v2_coco.config
```

– конвертируем результат обучения в frozen graph, который можно будет использовать:

```
!python /content/models/research/object_detection/export_inference_
graph.py --input_type image_tensor --pipeline_config_path /content/
training_demo/training/ssdlite_mobilenet_v2_coco.config --trained_
checkpoint_prefix /content/training_demo/training/model.ckpt-
[CHECKPOINT_NUMBER]
--output_directory /content/training_demo/training/output_inference_
graph_v1.pb
```

Должно получиться как на рис. 42.

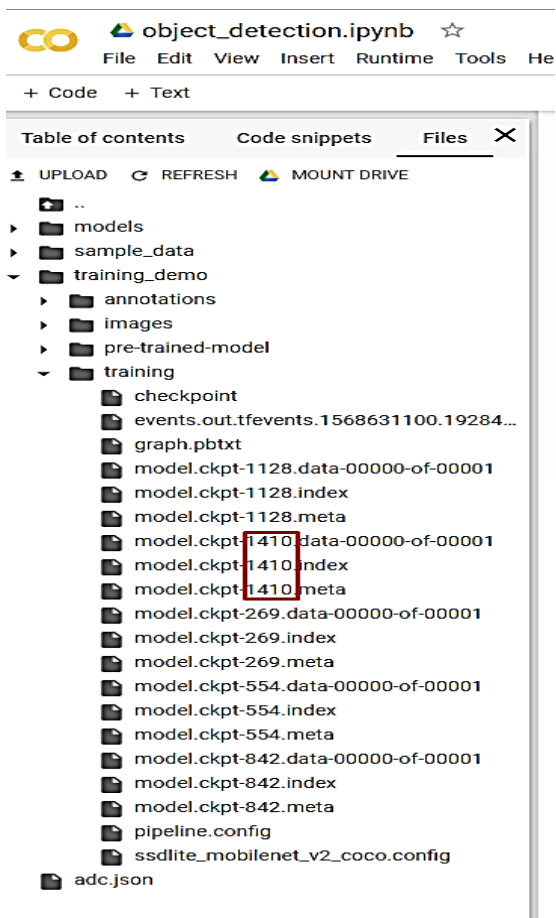


Рис. 42. Полученный результат преобразования

Для визуализации результата обучения в блокноте есть специальный скрипт;

– конвертируем обученную модель в tflite.

Для использования TensorFlow Lite нужно конвертировать модель в формат **tflite**. Для этого конвертируем результат обучения в **frozen graph**, который поддерживает конвертацию в **tflite** (параметры такие же, как при использовании скрипта **export_inference_graph.py**):

```
!python /content/models/research/object_detection/export_tflite_ssd_graph.py --pipeline_config_path /content/training_demo/training/
```

```
ssdlite_mobilenet_v2_coco.config --trained_checkpoint_prefix /content/  
training_demo/training/model.ckpt-[CHECKPOINT_NUMBER]  
--output_directory /content/training_demo/training/output_inference_  
graph_tf_lite.pb
```

Для конвертации в **tfLite** нам нужна дополнительная информация о модели, для ее получения скачиваем модель из **output_inference_graph_tf_lite.pb**, как показано на рис. 43.

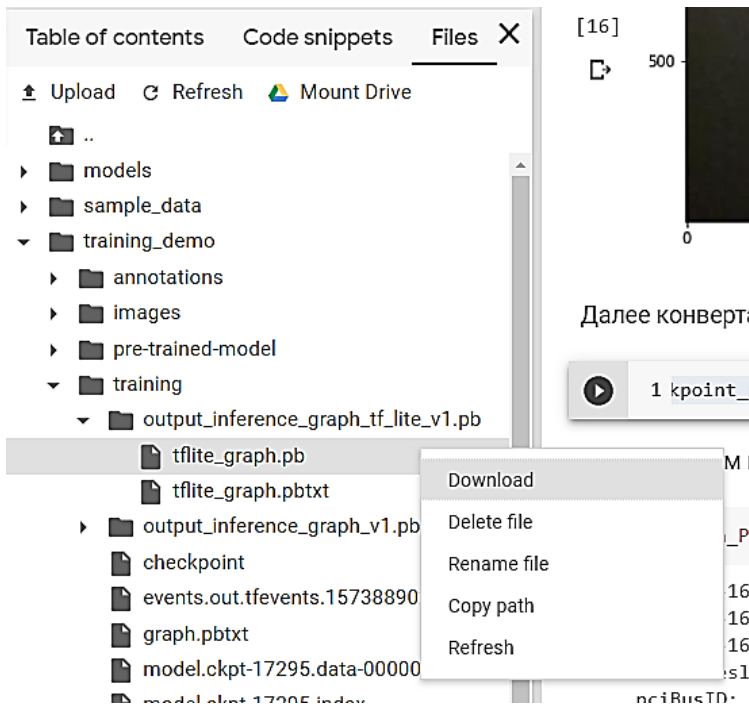


Рис. 43. Окно загрузки модели

И открываем ее в тулзе Netron, как показано на рис. 44. Нас интересуют названия и размерность входного и выходного узлов модели.

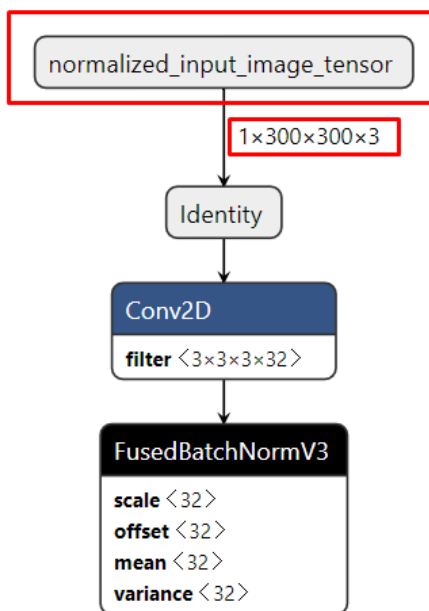


Рис. 44. Формат модели

Зная их, можно конвертировать pb-модель в tflite-формат:

```

!tflite_convert --output_file=/content/training_demo/training/
model_q.tflite --graph_def_file=/content/training_demo/training/
output_inference_graph_tf_lite_v1.pb/tflite_graph.pb --input_
arrays=normalized_input_image_tensor --output_arrays='TFLite_
Detection_PostProcess','TFLite_Detection_PostProcess:1','TFLite_
Detection_PostProcess:2','TFLite_Detection_PostProcess:3'
--input_shapes=1,300,300,3 --enable_select_tf_ops --allow_custom_ops
--inference_input_type=QUANTIZED_UINT8--inference_type=FLOAT
--mean_values=128 --std_dev_values=128
  
```

Архивируем папку с результатами обучения и размещаем ее в Google Drive:

```

!zip -r ./training_demo/training.zip ./training_demo/training/
training_result = drive.CreateFile({'title': 'training_result.zip'})
training_result.SetContentFile('training_demo/training.zip')
training_result.Upload()
  
```

Если будет ошибка Invalid client secrets file, то нужно сделать повторную авторизацию в Google Drive.

Запуск модели на Android-устройстве

В основе Android-приложения был использован официальный гайд по object detection, но он был полностью переписан с использованием kotlin и CameraX.

CameraX уже имеет механизм для анализа входящих фреймов с камеры с помощью ImageAnalysis. Логика по распознаванию находится в ObjectDetectorAnalyzer.

Весь процесс распознавания изображения можно разбить на несколько этапов:

– на вход мы получаем изображение, которое имеет YUV-формат. Для дальнейшей работы его нужно конвертировать в RGB-формат:

```
val rgbArray = convertYuvToRgb(image)
```

– далее нужно трансформировать изображение (повернуть, если есть необходимость, и изменить размер до входных значений модели, в нашем случае это 300×300), для этого отрисуем массив с пикселями на Bitmap и применим на нем трансформацию:

```
val rgbBitmap = getRgbBitmap(rgbArray, image.width, image.height)
```

```
val transformation = getTransformation(rotationDegrees, image.width, image.height)
```

```
Canvas(resizedBitmap).drawBitmap(rgbBitmap, transformation, null)
```

– конвертируем bitmap в массив пикселей и отдадим его на вход детектора:

```
ImageUtil.storePixels(resizedBitmap, inputArray)
```

```
val objects = detect(inputArray)
```

– для визуализации передадим результат распознавания в RecognitionResultOverlayView и преобразуем координаты с соблюдением соотношения сторон:

```
val scaleFactorX = measuredWidth / result.imageWidth.toFloat()
```

```
val scaleFactorY = measuredHeight / result.imageHeight.toFloat()
```

```
result.objects.forEach { obj ->
```

```
val left = obj.location.left * scaleFactorX
```

```
val top = obj.location.top * scaleFactorY
```

```
val right = obj.location.right * scaleFactorX
```

```
val bottom = obj.location.bottom * scaleFactorY
```

```
canvas.drawRect(left, top, right, bottom, boxPaint)
```

```
canvas.drawText(obj.text, left, top - 25f, textPaint)}
```

Чтобы запустить нашу модель в приложении, нужно заменить в assets файл с моделью на **training_demo/training/model_q.tflite** (переименовав ее в **detect.tflite**) и файл с метками **labelmap.txt**, в нашем случае это **detect.tflite**.

Так как в официальном гайде используется **SSD Mobilenet V1**, в котором индексация меток начинается с 1, а не с 0, нужно поменять Label Offset с 1 на 0 в методе collect Detection Result класса Object Detector.

Библиографический список

1. Wes McDermott. Creating 3D Game Art for the iPhone with Unity: Featuring modo and Blender pipelines. – М. : Focal Press, 2011. – 254 с.
2. Торн, А. Основы анимации в Unity / Алан Торн. – Москва : ДМК Пресс, 2016. – 176 с.
3. Joe Hocking. Unity in Action: Multiplatform Game Development in C# with Unity 5. – Eddifor!al : Manning Publications, 2015. – 352 с.
4. Линовес, Д. Виртуальная реальность в Unity / Джонотан Линовес ; пер. с англ. Р.Н. Рагимов. – Москва : ДМК Пресс, 2016. – 316 с.
5. Мэннинг, Д. Unity для разработчика. Мобильные мультиплатформенные игры / Джон Мэннинг, Пэрис Батфилд-Эддисон. – Санкт-Петербург : Питер, 2018. – 352 с.
6. Финни, К. 3D-игры. Все о разработке (+ CD-ROM) / К. Финни. – Москва : Бином. Лаборатория знаний, 2007. – 976 с.
7. Шикин, Е.В. Компьютерная графика. Полигональные модели / Е.В. Шикин, А.В. Боресков. – Москва : Диалог-Мифи, 2001. – 461 с.
8. Терехов, В.А. Нейросетевые системы управления / В.А. Терехов, Д.В. Ефимов, И.Ю. Тюкин. – Москва : ИПРЖР, 2002. – 480 с.
9. Галушкин, А.И. Нейронные сети. Основы теории / А.И. Галушкин. – Москва : Горячая Линия – Телеком, 2012. – 496 с.
10. Энгель, Е.А. Системы анализа, управления, принятия решений и обработки информации / Е.А. Энгель. – Москва : LAP Lambert Academic Publishing, 2011. – 168 с.
11. Турбин, А. Модель нейронного ансамбля на основе уравнения Фоккера-Планка / А. Турбин, А. Чижов. – Москва : LAP Lambert Academic Publishing, 2011. – 128 с.
12. Злобин, В.К. Нейросети и нейрокомпьютеры / В.К. Злобин, В.Н. Ручкин. – Санкт-Петербург : БХВ-Петербург, 2011. – 256 с.