Тимур Машнин

Разработка Android-приложений с Augmented Reality

Тимур Машнин

Разработка Android-приложений с Augmented Reality

«Издательские решения»

Машнин Т.

Разработка Android-приложений с Augmented Reality /
Т. Машнин — «Издательские решения»,

ISBN 978-5-44-838090-7

Дополненная реальность (Augmented Reality) не является какой-то новой технологией, но ее применение было замечено широкой публикой с появлением игры Pokemon GO, которая показала, что технология AR имеет большой потенциал.В книге рассмотрены различные способы разработки приложений с дополненной реальностью, от нативной разработки в Android Studio до использования таких движков, как Unity.

ISBN 978-5-44-838090-7

Содержание

Разработка Android-приложений с Augmented Reality

Тимур Машнин

ISBN 978-5-4483-8090-7
Создано в интеллектуальной издательской системе Ridero

Введение

Пригласить автора в проект admin@tmsoftstudio.com

Дополненная реальность (Augmented Reality) не является какой-то новой технологией, но ее применение было замечено широкой публикой с появлением игры Pokemon GO, которая показала, что технология AR имеет большой потенциал. Помимо игры Pokemon GO, такие технологии как Google Tango и Microsoft HoloLens также находятся на переднем крае AR.

«Дополненная» означает сделать нечто более сложное, добавляя что-то к чему-то. «Реальность» это состояние вещей, как они на самом деле существуют.

Например, сцена изображения камеры дополняется 3D Android логотипом в верхней части. Имейте в виду, что дополненная реальность не ограничивается только изображением, также возможны звук и другие сенсорные усовершенствования.

Дополненная реальность (AR) накладывает куски виртуального мира на реальный мир (в отличие от виртуальной реальности (VR), которая заменяет реальный мир виртуальным миром). Для мобильных устройств, это просто означает улучшение того, что вы можете видеть через камеру устройства. Например, вы можете навести вашу камеру на постер фильма и посмотреть его трейлер, или вы можете навести камеру на звезду в небе и узнать ее имя. Так что, в основном AR сводится к следующим трем основным вопросам: ГДЕ показать ЧТО и КАК.

ГДЕ может включать в себя такие области, как согласование 2-D изображений и их отслеживание, согласование 3-D объектов и их отслеживание, обнаружения лиц и их отслеживание, SLAM (Simultaneous Localization and Mapping), отслеживание местоположения (с помощью GPS, акселерометра, компаса, гироскопа). Иногда, ГДЕ это ничего больше, как некоторые заранее определенные точки местоположения Points of Interest (POIs).

С другой стороны, ЧТО и КАК может использовать рендеринг 3-D модели, анимацию и обнаружение жестов. В общем, ЧТО может быть любой частью цифровой информации (например, текст, изображение, видео), с которыми пользователь мог бы иметь возможность взаимодействовать (например, повернуть или переместить).

Используя смартфон в качестве примера, AR технология работает с помощью приложения, которое выполняет поиск маркера, как правило, черно-белого штрих-кода или другого изображения. После того, как маркер найден, на маркер затем накладывается 3D-объект. С помощью камеры телефона, отслеживая относительное положение устройства и маркера, пользователь может ходить вокруг маркера и просматривать 3D-объект со всех точек зрения. Это занимает много ресурсов, так как телефон должен отслеживать свое положение, а также положение маркеров, чтобы 3D-объект выглядел правильно.

Такие игры, как Pokemon GO, работают немного по-другому. Вместо использования физического маркера с привязкой к нему визуализации объекта, Pokemon GO просто отображает 3D-объект в видоискателе камеры. Используя этот метод, Pokemon GO не предоставляет возможность ходить вокруг покемонов, как традиционное использование технологии AR. На самом деле, в этой игре нет никакого отслеживания дистанции, вы можете свободно ходить вокруг, и покемон все равно останется на таком же расстоянии от вас, до тех пор, пока вы не пойдете в правильном направлении. Этот метод может в конечном итоге быть более общим способом реализации дополненной реальности в мобильном пространстве.

Технология Google Tango обеспечивает более сложную реализацию дополненной реальности для мобильных устройств, так как Tango устройство имеет специальное оборудование для этого. Tango устройство использует компьютерное зрение, чтобы отслеживать движение, имеет глубину восприятия и изучает пространство вокруг вас для самостоятельного исправления деталей. Tango устройство включает в себя стандартную камеру, камеру обнаружения движения рыбий глаз и датчик глубины.

По способу привязки виртуальных объектов к реальному миру, AR системы можно разделить на два типа – основанные на сенсорах и основанные на компьютерном зрении. AR приложения, основанные на сенсорах, используют GPS, акселерометры, магнитометры и гироскопы для определения глобальной позиции пользователя в реальном мире и имеют ограничения по использованию вне помещения и скорости перемещения пользователя из-за запаздывания передачи GPS информации. AR приложения, основанные на компьютерном зрении, используют камеру устройства для компьютерной обработки изображения и регистрации виртуального объекта в реальном мире и имеют ограничения по мощности используемых устройств, так как компьютерная обработка изображения камеры потребляет значительные ресурсы.

Существует несколько способов разработки приложений с дополненной реальностью, от нативной разработки в Android Studio до использования таких движков, как Unity. На сегодняшний день доступны несколько десятков SDK для разработки AR приложений, ниже перечислены некоторые из них:

Vuforia – разработан компанией Qualcomm. Этот SDK компьютерного зрения обеспечивает разработку приложений с дополненной реальностью, основанной на отслеживании маркеров, для Android и ОС IOS с поддержкой Unity. Vuforia поддерживает несколько целей одновременно, Smart Terrain (реконструкция физического мира), а также локальные и облачные базы данных.

FastCV Computer Vision SDK – разработан компанией Qualcomm. Обеспечивает распознавание жестов, обнаружение, слежение и распознавание человеческого лица, распознавание и отслеживание текста, дополненную реальность. Библиотека FastCV представляет собой оптимизированную для мобильных устройств библиотеку компьютерного зрения, включающую в себя наиболее часто используемые функции обработки компьютерного зрения для использования в широком спектре мобильных устройств.

OpenCV (Open Source Computer Vision Library) – библиотека компьютерного зрения и машинного обучения с открытым исходным кодом. OpenCV обеспечивает общую инфраструктуру для приложений компьютерного зрения.

ARToolKit – библиотека компьютерного зрения, обеспечивающая надежное отслеживание маркеров, включая отслеживание изображений Natural Feature Tracking, поддержку калибровки камеры, одновременное отслеживание и поддержку стерео камеры, мультиязычность, оптимизацию для мобильных устройств, полную поддержку Unity3D и OpenSceneGraph.

OpenSpace3D – является «свободным программным обеспечением» для развития проектов виртуальной и дополненной реальности. Цель OpenSpace3D состоит в том, чтобы демократизировать 3D-приложения реального времени и предоставить инструмент для всех творческих умов, а не только разработчиков. OpenSpace3D поддерживает два метода дополненной реальности для создания AR-приложений. Обнаружение маркера, позволяющее отслеживать произвольное изображение с помощью камеры, и обнаружение Aruco реперного маркера, что позволяет делать быстрые приложения с помощью нескольких маркеров, а также использовать их в качестве материального интерфейса.

BeyondAR – платформа предлагает ресурсы для разработки приложений с дополненной реальностью, основанной на георасположении на смартфонах и планшетах. С помощью нескольких строк кода можно создавать 2D-объекты, чтобы увидеть их через камеру.

Beyond Reality Face Nxt – платформа позволяет создавать веб, мобильные и настольные приложения, отслеживающие человеческое лицо, с использованием Actionscript или HTML5/Javascript.

VISION SDK – полностью настраиваемая, легкая в использовании библиотека представлений дополненной реальности, которая позволяет включение AR в любое приложение для Apple или Android устройств без необходимости быть экспертом в программировании. С VISION SDK любое мобильное приложение может быть улучшено благодаря этой технологии, которая интегрирует цифровую информацию в общий вид окружающего пространства естественным образом.

ARLab – больше, чем просто SDK, ARLab также имеет 3D-движок, который может быть использован для создания AR-приложений. ARLab не является бесплатным, и предлагает несколько различных вариантов цен в зависимости от того, какие функции вы хотите включить в ваше приложение. ARLab обеспечивает виртуальные кнопки, отслеживание изображения и сопоставление изображения.

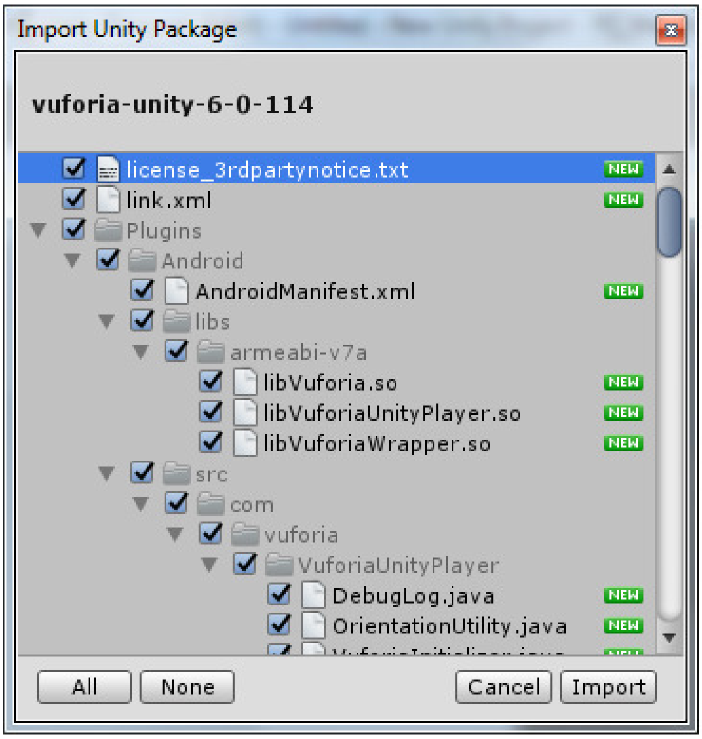
Wikitude SDK – платная, включает в себя распознавание образов и отслеживание, рендеринг 3D моделей, видео с наложением, AR, основанное на местоположении, и многое другое, поддерживает Android, iOS, Smartphone, Tablet, Smart Glasses, Cordova/PhoneGap, Titanium, Xamarin.

Intel RealSense SDK – требует наличия двух камер User Facing (SR300) and World Facing (R200), обеспечивает распознавание жестов, лиц, 3D сканирование стационарных объектов, удаление фона, отслеживание объектов, распознавание речи (требует наличия только микрофона), создание цифрового представления наблюдаемой среды и оценка положения камеры в реальном масштабе времени, улучшение фотографий и видео за счет использования 3D глубины.

Vuforia

Для использования Unity в разработке AR приложений, в качестве первого шага, необходимо скачать платформу Unity. После установки, откройте Unity и создайте новый проект, при этом убедитесь, что выбрана опция «3D». Закройте Unity.

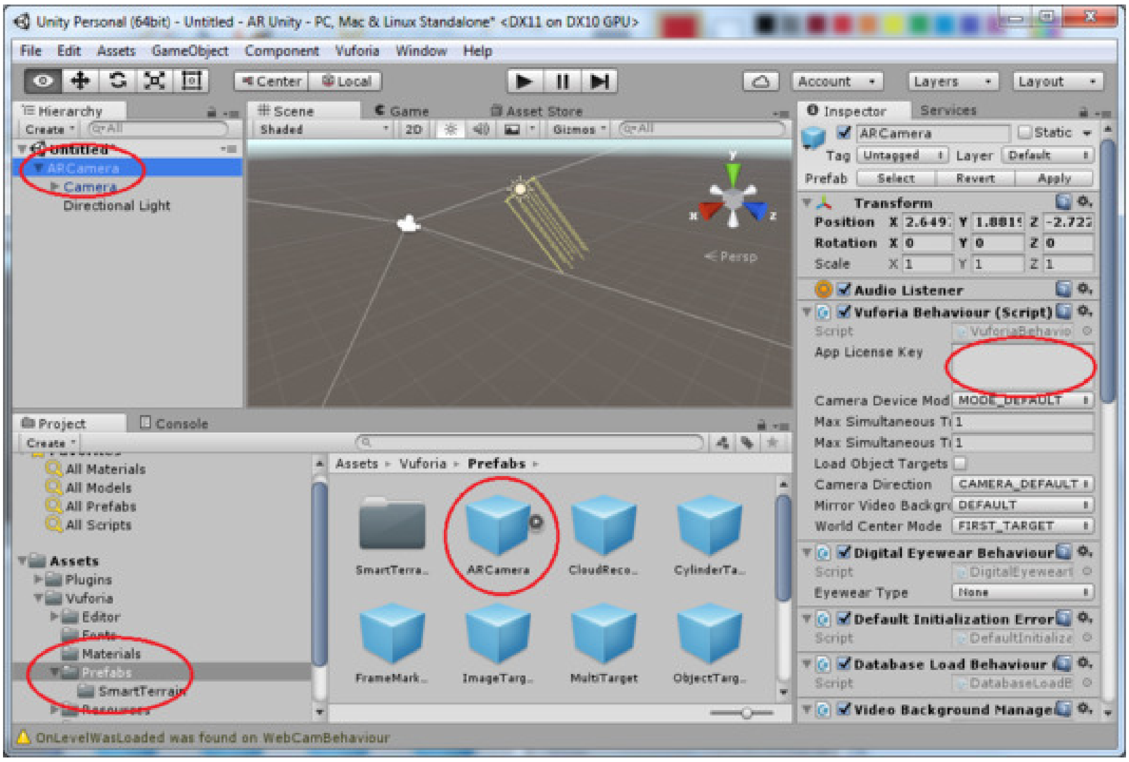
Загрузите Vuforia Unity Extension и дважды кликните на файле *.unitypackage. При открытии Unity, выберете созданный проект. Импортируйте Vuforia Unity Extension в проект.



Удалите из сцены Main Camera и перетащите в сцену камеру ARCamera в папке Assets/Vuforia/Prefabs окна Project.

В окне Inspector откройте свойства ARCamera.

Получите лицензионный ключ для приложения на сайте Vuforia и вставьте его в поле App Lisense Key.

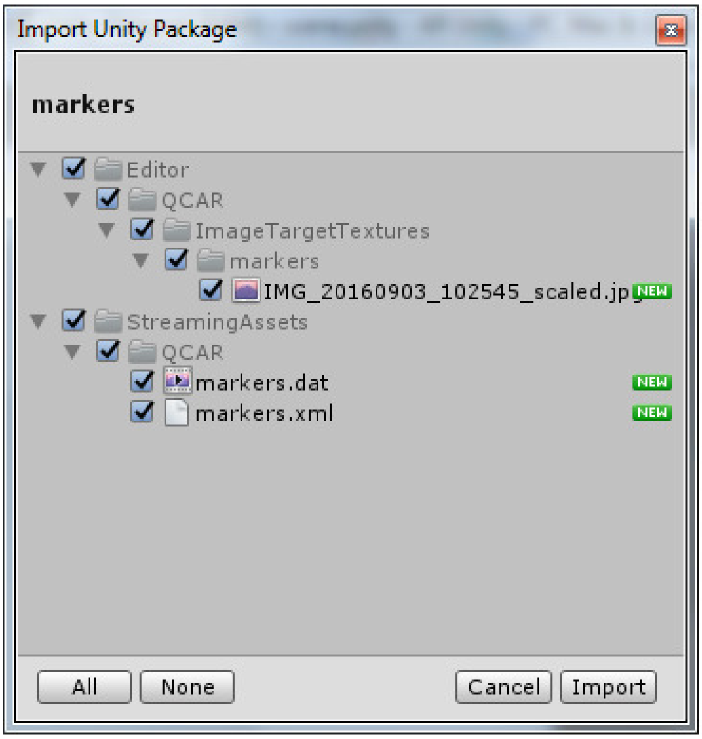


Поместим камеру в начало координат Position 0, 0, 0, Rotation 90, 0, 0.

На сайте Vuforia создайте базу данных маркеров, к которым будут прикрепляться 3D-объекты. Для этого нужно нажать кнопку Add Database во вкладке Develop/Target Manager.

Сделайте фотографию какого-либо предмета и с помощью кнопки Add Target загрузите изображение в базу маркеров.

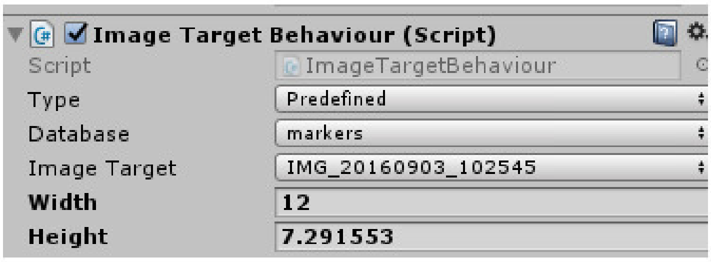
С помощью кнопки Download Database скачайте базу маркеров и дважды кликните на скачанном Unity пакете. Импортируйте маркеры в Unity проект.



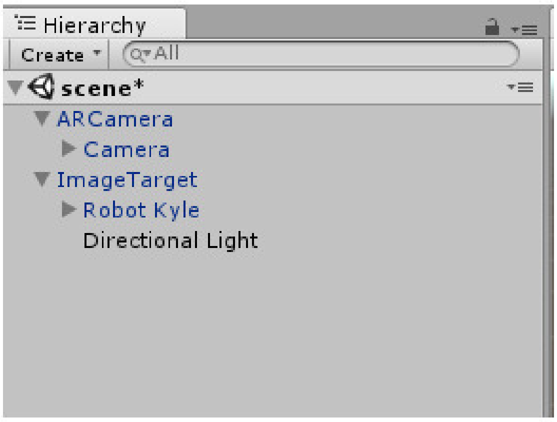
Добавим 3D модель, которую будет прикреплять к маркеру. Для этого в Unity откроем окно Window/Asset Store и скачаем Unity пакет с 3D моделями. Импортируем скачанный пакет в Unity проект.

Из папки Assets/Vuforia/Prefabs окна Project перетащим в сцену объект Image Target в позицию Position 0, -10, 0, Rotation 0, 0, 0.

В свойствах Image Target в окне Inspector в разделе Image Target Behaviour в поле Database выберем импортированную базу маркеров и увидим наше изображение в сцене.



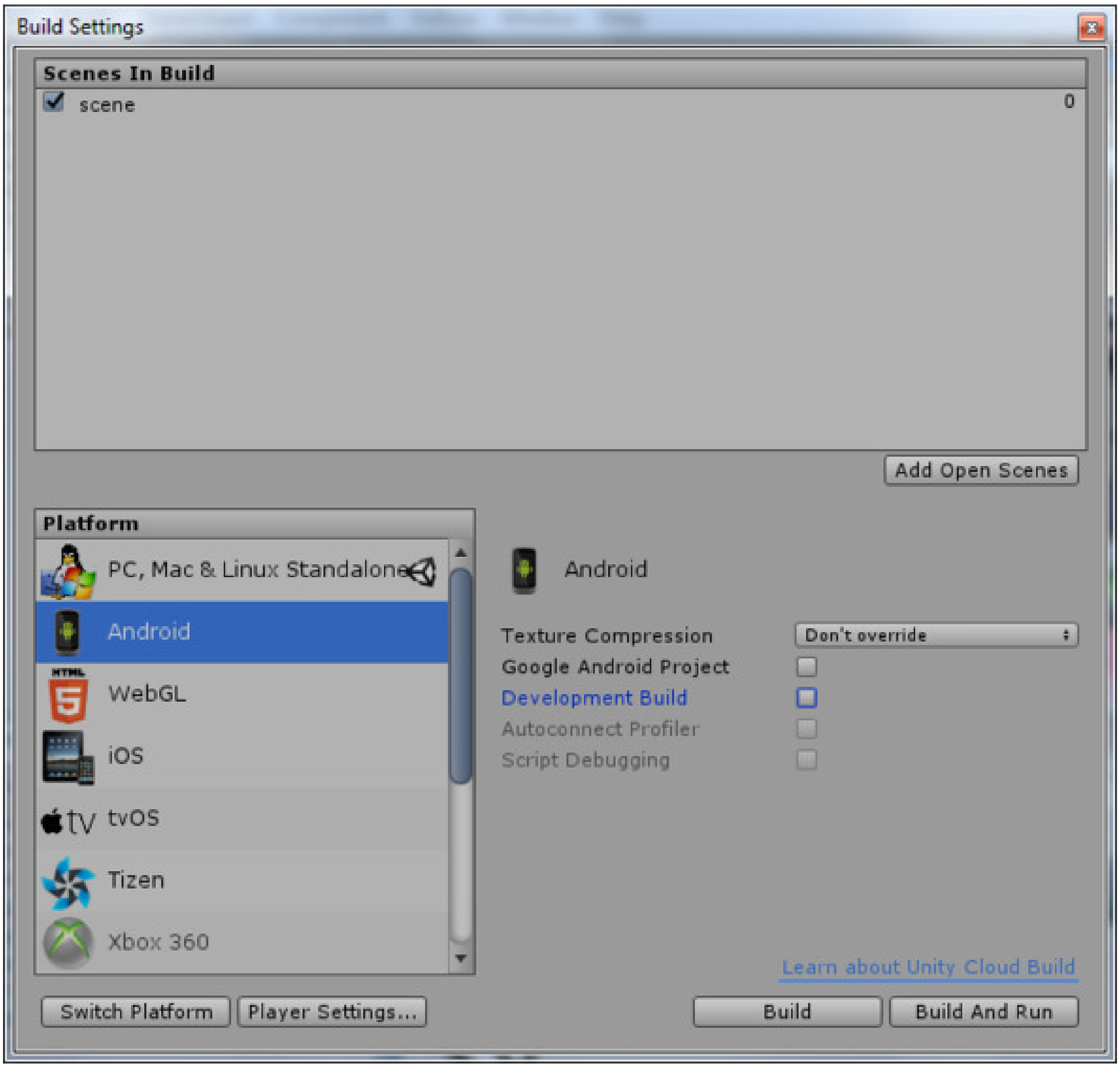
Перетащим в сцену 3D модель, импортированную из Asset Store, в позицию Position 0, 0, 0, Rotation 0, 0, 0. В поле Scale свойств модели подберем масштаб модели относительно нашего маркера. В окне Hierarchy перетащим модель в узел ImageTarget и таким образом сделаем модель дочерним объектом объекта Image Target.



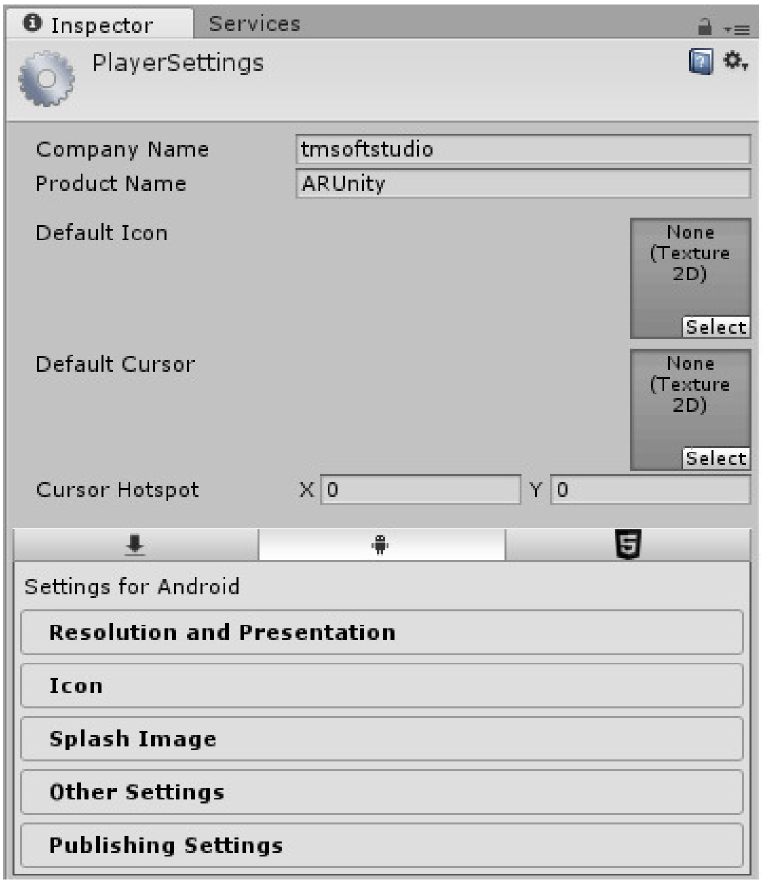
Теперь, когда маркер Image Target будет обнаруживаться камерой мобильного устройства, все дочерние объекты маркера Image Target также будут появляться в камере.

Активируем базу маркеров для камеры ARCamera. Для этого в свойствах ARCamera в окне Inspector в разделе Dataset Load Behaviour выберем Load markers Database и Activate.

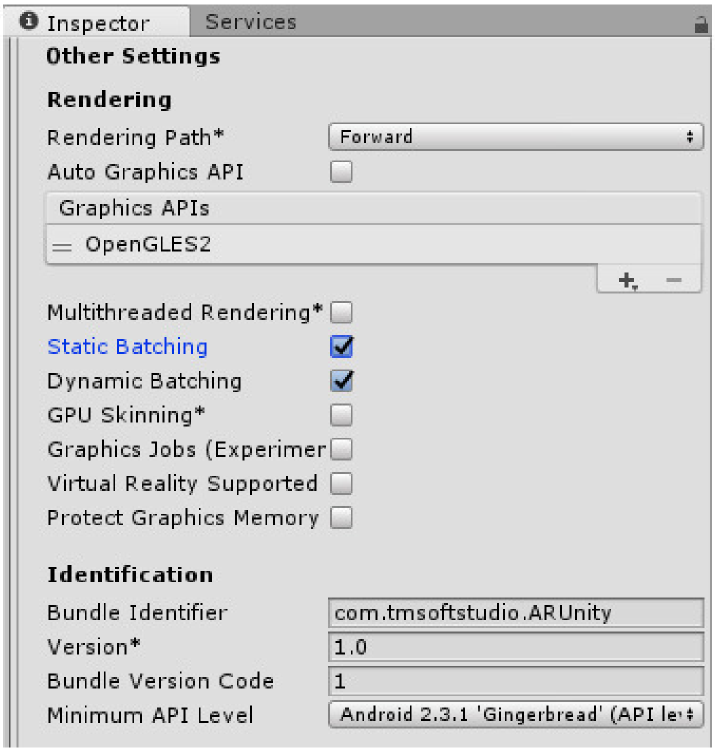
Соберем Unity проект в виде приложения для Android устройства. Для этого в меню File выберем Build Settings, выберем платформу Android и кнопкой Add Open Scenes добавим созданную сцену.



Кнопка Player Settings открывает окно настроек сборки приложения.



В поле Company Name введем свое имя, в разделе Other Settings в поле Bundle Identifier введем имя пакета, состоящее из полей Company Name и Product Name.



Нажмем кнопку Build и сохраним APK файл.

Установим APK файл на мобильное устройство. Теперь при наведении камеры устройства на ранее сфотографированный предмет, будет появляться виртуальная 3D модель.

ARToolKit

ARToolKit является библиотекой компьютерного зрения, которая обеспечивает функциональные возможности отслеживания, необходимые для создания приложений дополненной реальности.

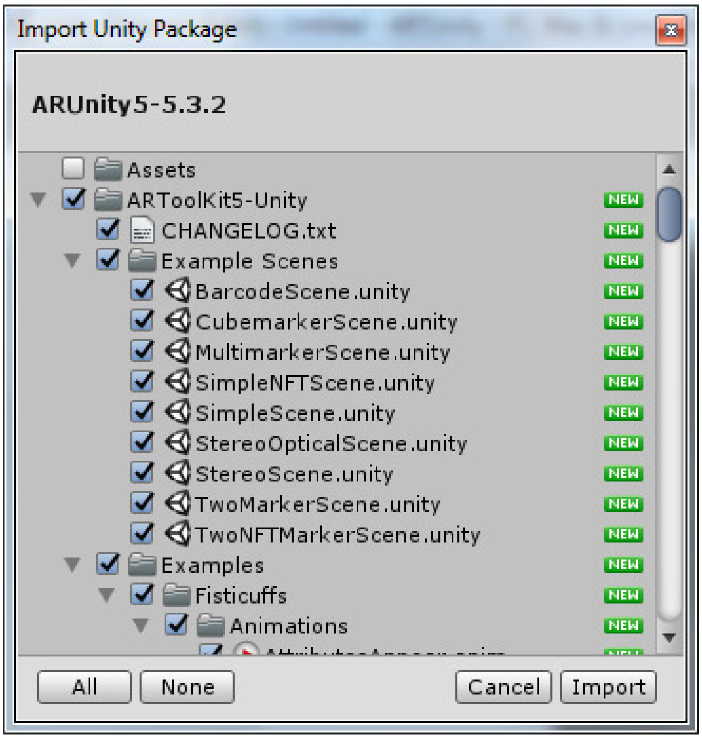
ARToolKit плагин для Unity

ARToolKit плагин для Unity можно скачать на странице https://artoolkit.org/download-artoolkit-sdk#unity.

Более актуальную и стабильную версию плагина для Unity можно скачать на GitHub https://github.com/artoolkit/arunity5.

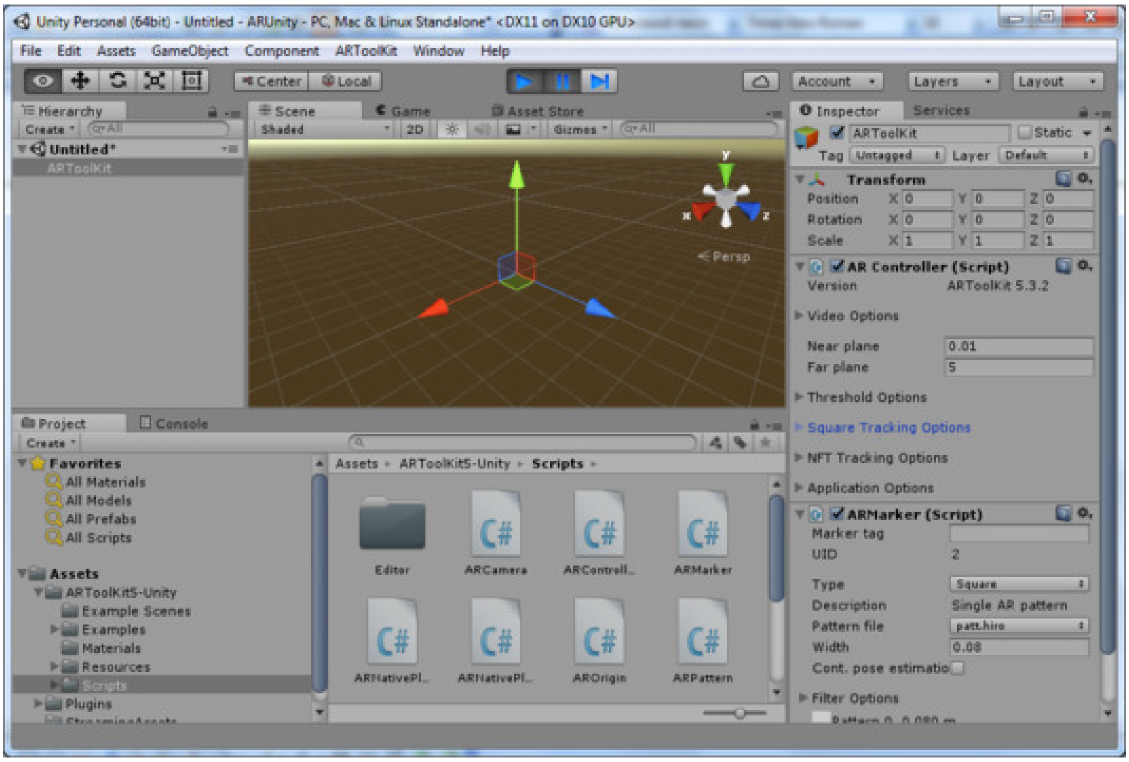
После скачивания пакета откроем Unity и создадим новый 3D проект.

В меню Assets выберем Import Package/Custom Package, откроем и импортируем скачанный пакет.



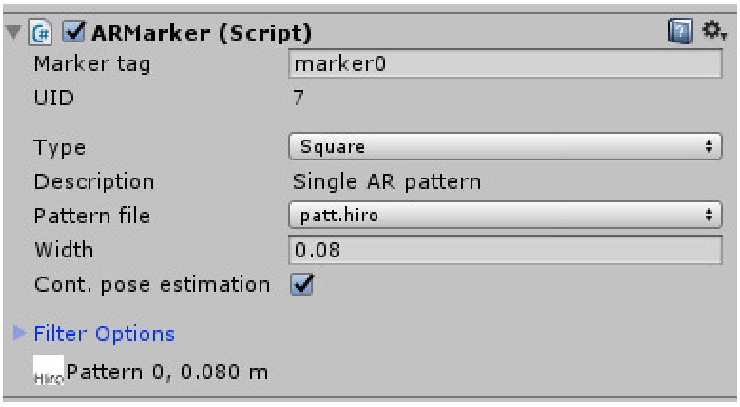
В сцене Unity проекта удалим все объекты и с помощью меню Create/Create Empty создадим объект GameObject, который назовем ARToolKit.

Этот объект будет содержать два AR конфигурационных объекта ARController и ARMarker. Поэтому перетащим их из папки Scripts в объект ARToolKit.



ARController скрипт отвечает за видео-фон и за создание и управление AR отслеживанием.

ARMarker скрипт обеспечивает маркер отслеживания, к которому прикрепляется 3D модель. В поле Marker tag введем идентификатор маркера.



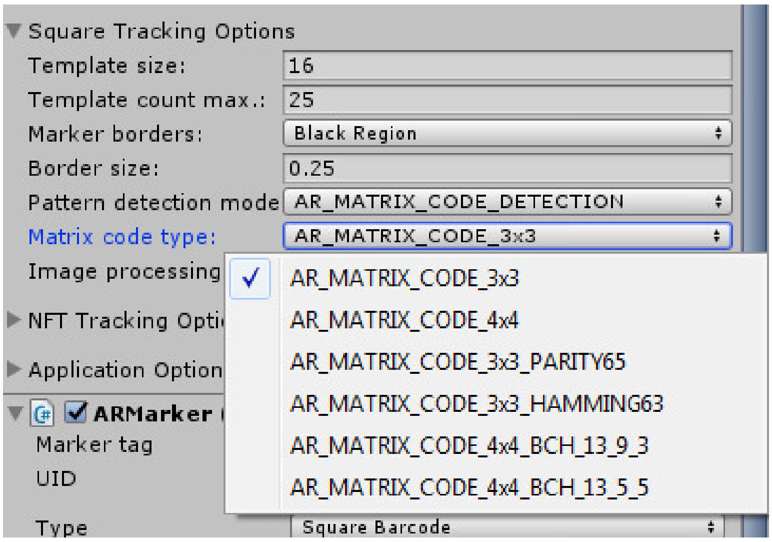
По умолчанию тип маркера Square, и изображение маркера установлено из папки Resources/ardata/markers.

Для использования этого маркера, изображение нужно распечатать из каталога ARUnity-tools\doc\patterns дополнительного набора инструментов Additional Unity Tools, который можно скачать на странице https://artoolkit.org/download-artoolkit-sdk.

Другой тип маркера это Square Barcode. Square Barcode это изображение, имеющее шаблон, предопределенный для распознавания библиотекой ARToolKit, в виде матрицы из черных и белых квадратов. Использование данного типа маркера ускоряет его распознавание камерой и обеспечивает надежность его идентификации.

Для распечатки маркеры Square Barcode находятся в каталоге ARUnity-tools\doc\patterns дополнительного набора инструментов Additional Unity Tools.

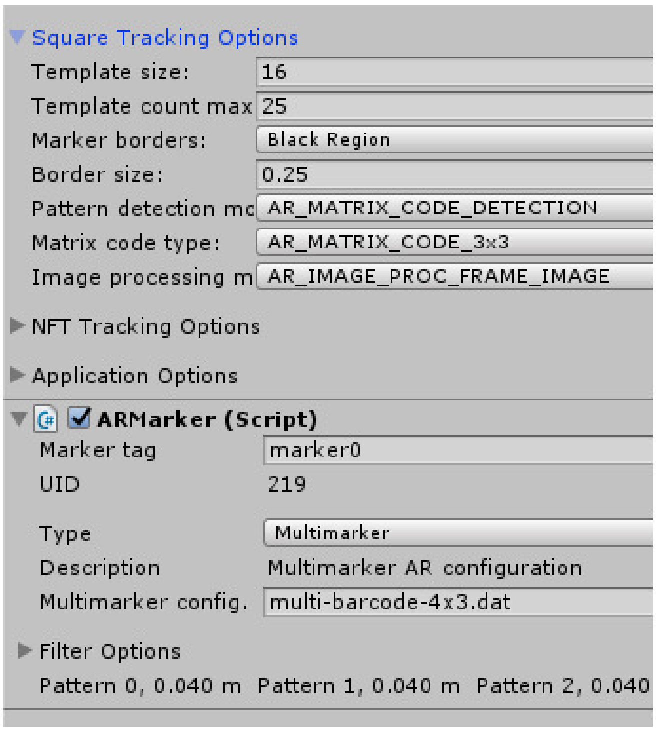
Для использования маркера Square Barcode в свойствах ARController скрипта в разделе Square tracking options в поле Pattern detection mode выберем AR_MATRIX_CODE_DETECTION, а в поле Matrix Code Type выберем тип набора маркеров.



В свойствах ARMarker скрипта в поле Type выберем Square Barcode, в поле Barcode ID введем номер маркера из набора.

Следующий тип маркера это Multimarker, состоящий из множества маркеров Square Barcode для прикрепления к одному 3D объекту. Для распечатки маркеры Multimarker находятся в каталоге ARUnity-tools\doc\patterns дополнительного набора инструментов Additional Unity Tools. Преимущество использования Multimarker состоит в повышенной устойчивости к окклюзии, даже когда один маркер затемняется, другой маркер все равно виден, а также в эффективном покрытии большего оптического угла, что приводит к уменьшению ошибок.

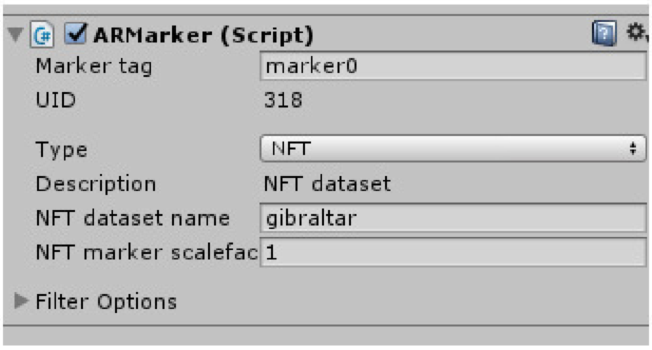
Для использования маркера Multimarker в свойствах ARController скрипта в разделе Square tracking options в поле Pattern detection mode выберем AR_MATRIX_CODE_DETECTION, в свойствах ARMarker скрипта в поле Type выберем Multimarker, в поле Multimarker config. введем имя конфигурационного файла из каталога StreamingAssets.



Следующий тип маркера это NFT маркер. Функция Natural Feature Tracking (NFT) позволяет распознавать и отслеживать обычное изображение, которое не содержит предопределенные маркеры.

Для распечатки примеры NFT маркеров находятся в каталоге ARUnity-tools\doc\Marker images дополнительного набора инструментов Additional Unity Tools.

Для использования NFT маркера в свойствах ARMarker скрипта в поле Type выберем NFT, в поле NFT dataset name введем имя набора данных из каталога StreamingAssets.



Примеры сцен, созданных с использованием различных типов маркеров, можно посмотреть в каталоге Example Scenes.

Создадим свой маркер, к которому будет прикрепляться 3D модель.

Для этого сфотографируем и сохраним jpeg изображение предмета, который будет служить NFT маркером.

Переместим изображение в папку bin каталога дополнительного набора инструментов Additional Unity Tools, в которой находится инструмент genTexData. exe генерации NFT набора данных

В командной строке наберем:

genTexData. exe image.jpeg

Далее в процессе генерации нужно будет ввести уровень извлечения характерных точек изображения, а также диапазон разрешения изображения, в котором характерные точки изображения будут извлекаться при приближении или удалении камеры.

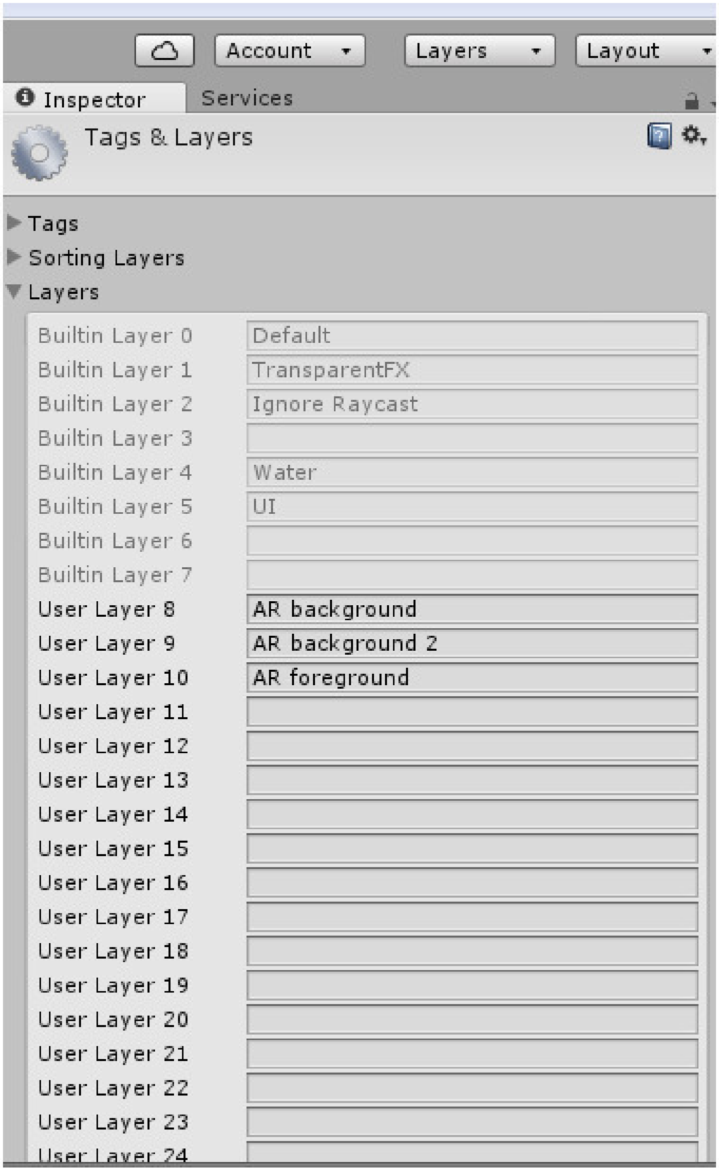
После окончания генерации NFT набора мы получим три файла. iset,.fset и. fset3, которые перетащим в каталог StreamingAssets Unity проекта.

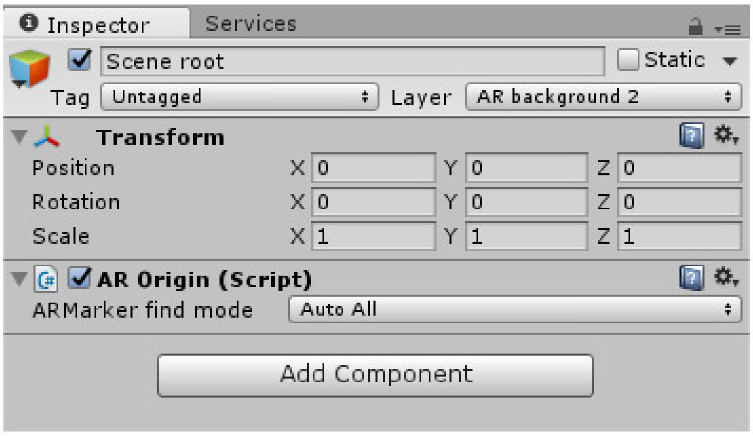
Далее в свойствах ARMarker скрипта в поле Type выберем NFT, в поле NFT dataset name введем имя полученного NFT набора из каталога StreamingAssets.

Для дальнейшего формирования сцены с помощью меню Create/Create Empty создадим объект GameObject, который назовем Scene root. Перетащим из папки Scripts в объект Scene root скрипт AROrigin, представляющий центр ARToolkit мира и являющийся корневым объектом сцены.

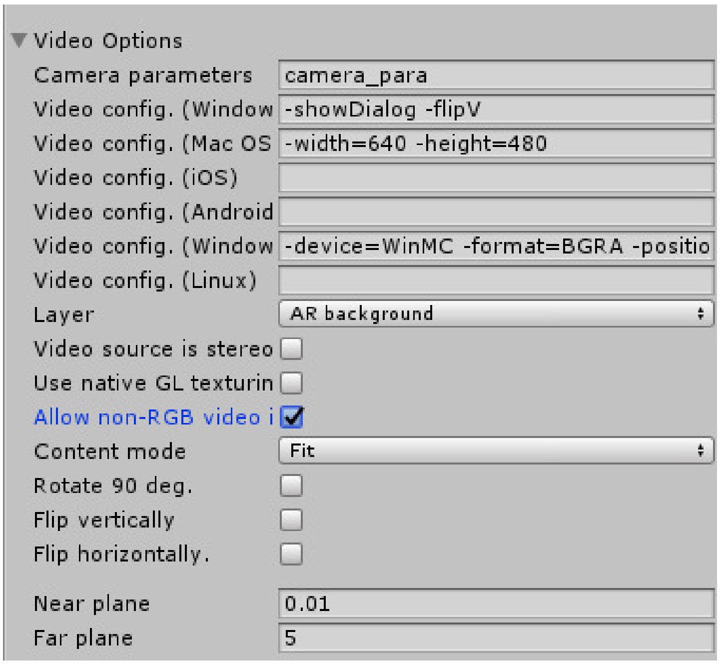
В свойствах объекта Scene root в окне Inspector в поле Layer выберем 9 слой сцены.


Layers
Everything
Nothing
Default
TransparentFX
Ignore Raycast
Water
UI
AR background
AR background 2
AR foreground
Edit Layers...


Account
Layers
Layout
Inspector
Services
Tags & Layers
Tags
Sorting Layers
Layers
Builtin Layer 0
Default
Builtin Layer 1
TransparentFX
Builtin Layer 2
Ignore Raycast
Builtin Layer 3
Builtin Layer 4
Water
Builtin Layer 5
UI
Builtin Layer 6
Builtin Layer 7
User Layer 8
AR background
User Layer 9
AR background 2
User Layer 10
AR foreground
User Layer 11
User Layer 12
User Layer 13
User Layer 14
User Layer 15
User Layer 16
User Layer 17
User Layer 18
User Layer 19
User Layer 20
User Layer 21
User Layer 22
User Layer 23

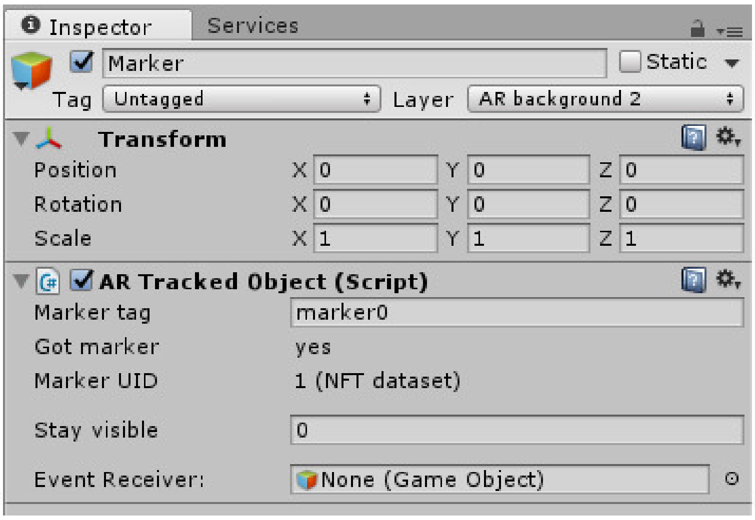


При этом в свойствах ARController скрипта в разделе Video Options в поле Layer должен быть выбран 8 слой сцены, а также выбрана опция Allow non-RGB video.

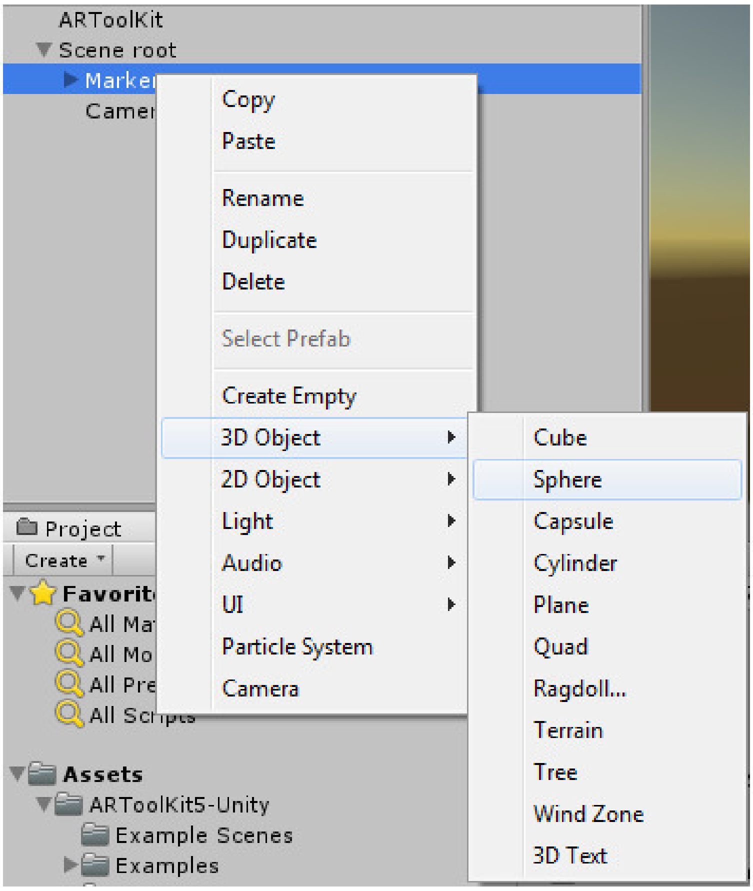


С помощью меню Create/Create Empty Child создадим объект GameObject, дочерний для объекта Scene root, который назовем Marker. Перетащим из папки Scripts в объект Marker скрипт ARTrackedObject, представляющий маркер. Его дочерние объекты будут автоматически прикрепляться к этому маркеру.

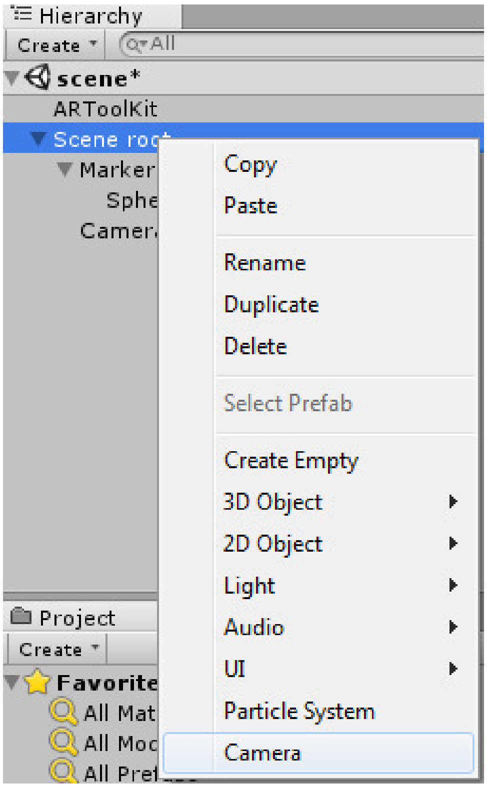
В свойствах скрипта ARTrackedObject в поле Marker tag введем тот же идентификатор, что и в поле Marker tag свойствах скрипта ARMarker.



К объекту Marker прикрепим дочерний объект – 3D модель.

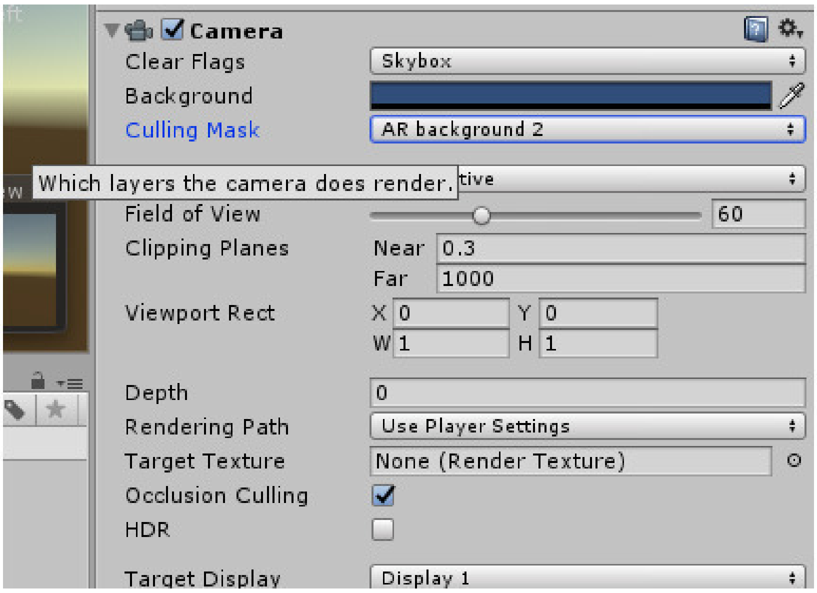


Нажмем правой кнопкой мышки на объекте Scene root и прикрепим к нему дочерний объект Camera.



Перетащим из папки Scripts в объект Camera скрипт ARCamera, связывающий Unity камеру с AR контентом.

В свойствах объекта Camera в поле Culling Mask выберем 9 слой сцены.



Нажмем кнопку проигрывания сцены и поднесем к камере маркер – должен появиться 3D объект.

Для сборки Android приложения в меню File выберем Build Settings, кнопкой Add Open Scenes добавим нашу сцену, выберем платформу Android и нажмем кнопку Player Settings.

В поле Company Name введем имя пакета, в поле Product Name введем имя приложения, в разделе Other Settings в поле Bundle Identifier введем com. [Company Name]. [Product Name]. В других разделах введем остальные настройки приложения.

В каталоге Plugins/Android откроем файл манифеста AndroidManifest. xml и в атрибуте package введем com. [Company Name]. [Product Name]. После этого нажмем Build и соберем APK файл Android приложения.

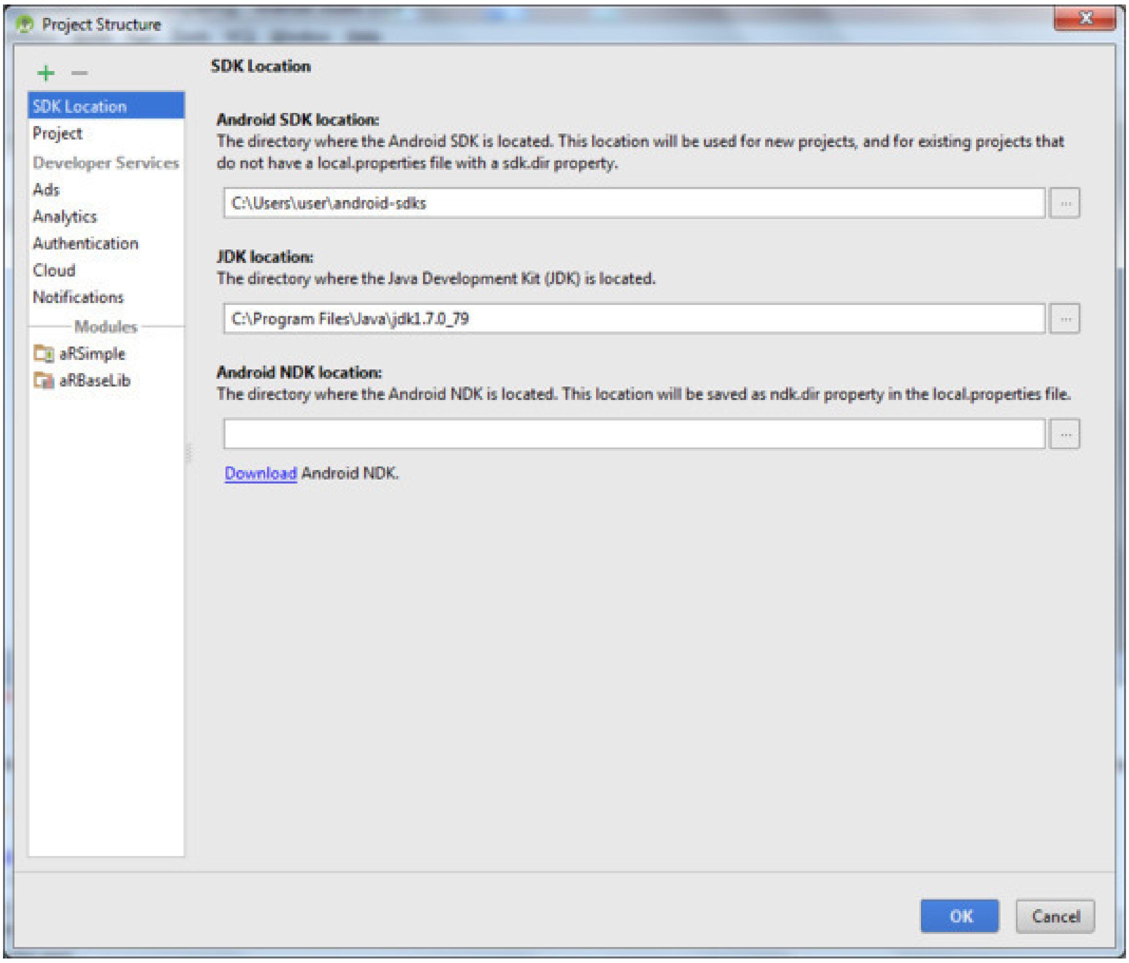
ARToolKit для Android

Скачаем и установим:
Java Development Kit 1.7+
Android Studio IDE 1.5.x+
Android SDK
Git

При установке Git в окне Adjusting your PATH environment выберем Use Git from Git Bash only, в окне Configuring the line ending conversions выберем Checkout Windows-style, commit Unix-style line ends.

Для установки Android NDK откроем Android Studio и в меню File откроем Project Structure.

Воспользуемся ссылкой Download Android NDK.



В результате в каталоге Android SDK будет создана папка ndk-bundle, содержащая Android NDK.

Установим переменные среды.
ANDROID_HOME = …\android-sdks
ANDROID_NDK_ROOT = %ANDROID_HOME%\ndk-bundle
NDK = %ANDROID_HOME%\ndk-bundle
;%NDK%\ добавить в Path

Из GitHub скачаем и распакуем artoolkit5 и в папке artoolkit5\android запустим Git скрипт build.sh и build_native_examples.sh.

Возможно, в скрипте нужно будет вывести переменную $WinsVerNum и поменять ее значение в коде $WinsVerNum = «6.3».

else #Checking for Windows in a non-cygwin dependent way.

WinsOS=

if [[$OS]]; then

WinsVerNum=$ {OS##*-}

echo $WinsVerNum

if [[$WinsVerNum = «10.0» || **$WinsVerNum = «6.3»**]]; then

if [[$WinsVerNum = «10.0»]]; then

WinsOS=«Wins10»

else

WinsOS=«Wins8.1»

fi

echo Building on Microsoft $ {WinsOS} Desktop \ ($ {ARCH} \)

export HOST_OS=«windows»

NDK_BUILD_SCRIPT_FILE_EXT=".cmd»

CPUS=`/usr/bin/nproc`

fi

fi

fi

if [[! $CPUS]]; then

echo **Development platform not supported, exiting script**

read -rsp $«Press enter to continue…\n’

exit 1

В результате будет сгенерирована папка libs с файлами libARWrapper.so и libc++_shared.so для различных CPU архитектур в папке artoolkit5\android, а также в проектах каталога artoolkit5\AndroidStudioProjects.

Откроем Android Studio и откроем проект ARSimpleProj каталога AndroidStudioProjects.

В меню File откроем Project Structure и увидим, что модуль aRSimple имеет зависимость от модуля aRBaseLib, представленного проектом ARBaseLibProj. Если этой зависимости нет, ее нужно добавить с помощью меню File/Project Structure.

Библиотека ARBaseLib предоставляет Java классы ARToolKit, ARActivity и ARRenderer для создания ARToolKit приложения и обеспечивает с помощью JNI связь с нативной C++ библиотекой ARWrapper, представленной файлами libARWrapper.so и libc++_shared.so, которая управляет жизненным циклом ARToolkit приложения, включая инициализацию, добавление маркеров и др.

При создании своего ARToolKit приложения, в каталог src\main проекта нужно включить папку libs с файлами libARWrapper.so и libc++_shared.so для различных CPU архитектур, а также добавить зависимость от модуля aRBaseLib с помощью меню File/Project Structure/Add a new module/Import. JAR/.AAR Package/AndroidStuiodProjects/ARBaseLibProj/arBaseLib/build/outputs/aar/ ARBaseLib. aar.

Документация API библиотеки ARBaseLib находится в папке AndroidStudioProjects\ARBaseLibProj\doc.

При запуске приложения ARSimple, при наведении камеры устройства на маркер, будет появляться 3D объект – куб.

Из предыдущего раздела ARToolKit плагин для Unity возьмем изображение gibraltar.jpg и инструмент genTexData. exe и создадим набор данных маркера.

genTexData. exe gibraltar.jpg

Поместим файлы. iset,.fset и. fset3 в папку AndroidStudioProjects\ARSimpleProj\aRSimple \src\main\assets\Data.

В классе SimpleRenderer изменим код добавления маркера.

```
@Override
public boolean configureARScene () {

markerID = ARToolKit.getInstance().addMarker («nft; Data/gibraltar»);
if (markerID <0) return false;

return true;
}
```

В файле модуля build.gradle увеличим значение versionCode для обновления кэша.

Теперь при наведении камеры устройства на простое изображение, будет появляться 3D объект – куб.

По умолчанию, свойства камеры устройства содержатся в файле camera_para. dat папки src\main\assets\Data проекта ARToolKit приложения. Параметры камеры устройства по умолчанию являются достаточными для базового отслеживания для широкого спектра различных камер.

OpenSpace3D

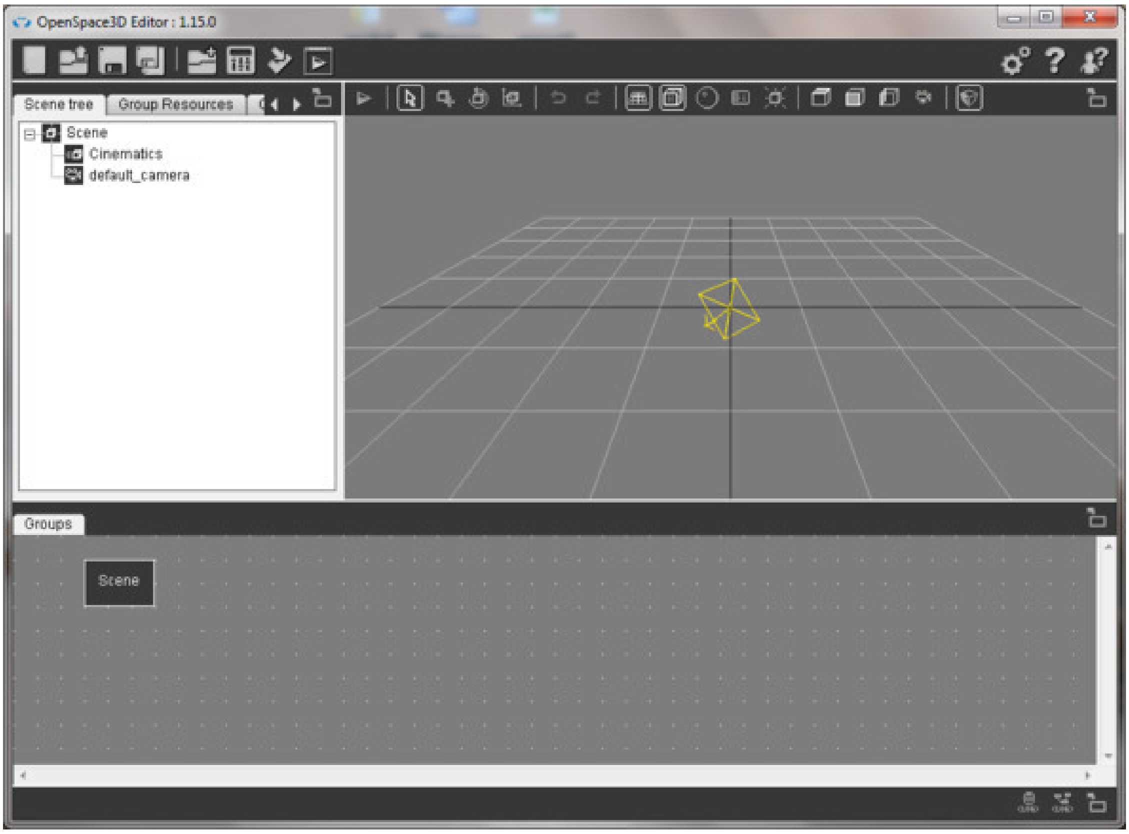
Проект OpenSpace3D, основанный на языке Scol 3D приложений режима реального времени и многопользовательских приложений и 3D-движке SO3Engine, предоставляет набор свободного программного обеспечения для разработки проектов приложений виртуальной и дополненной реальности.

В OpenSpace3D отсутствует возможность монетизации готового приложения и возможность экспорта сцены в распространенные форматы.

В OpenSpace3D есть возможность регистрации маркера динамически, на лету, также имеется Face Tracking.

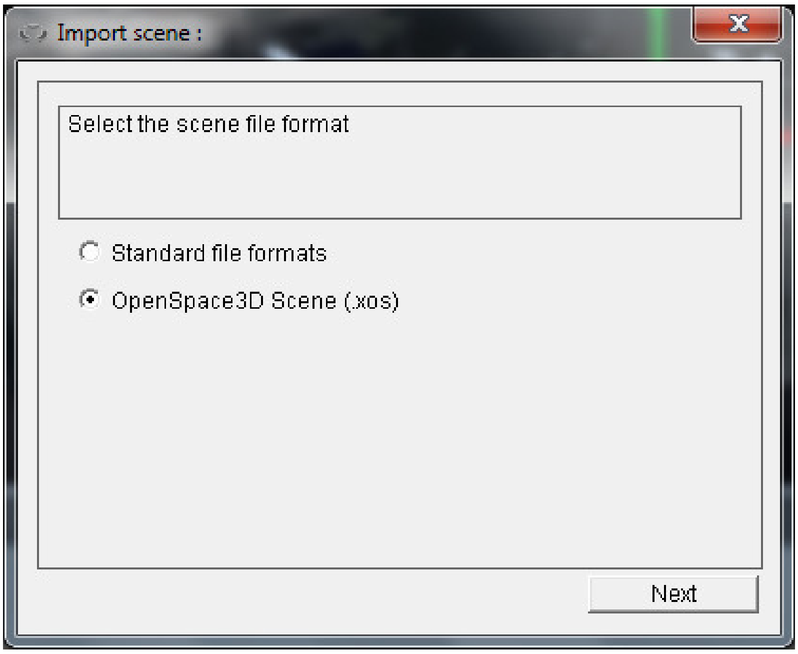
Скачаем и установим OpenSpace3D, а также дополнительные пакеты (http://www.openspace3d.com/lang/en/support/download/). При этом будет создан каталог для OpenSpace3D проектов C:\Users\user\Documents\OpenSpace3D.

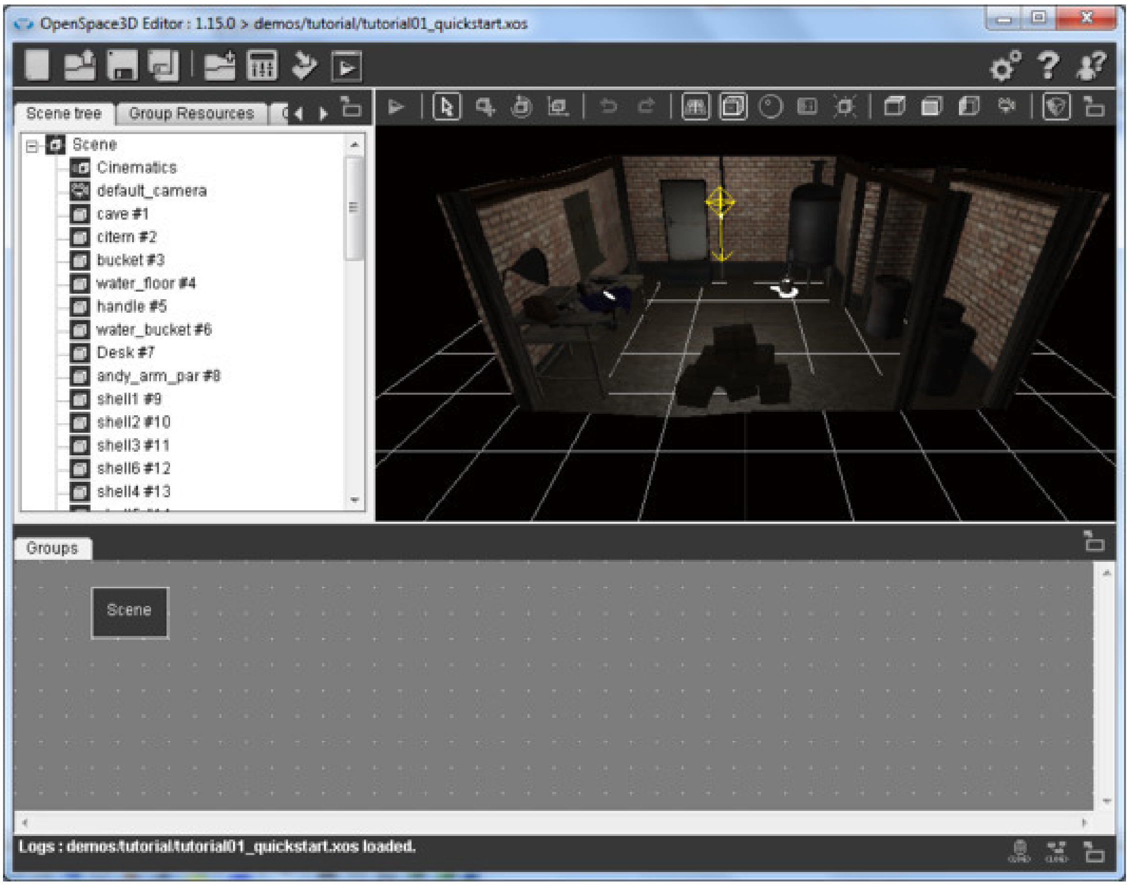
После скачивания и установки дистрибутива OpenSpace3D, откроем OpenSpace3D редактор.



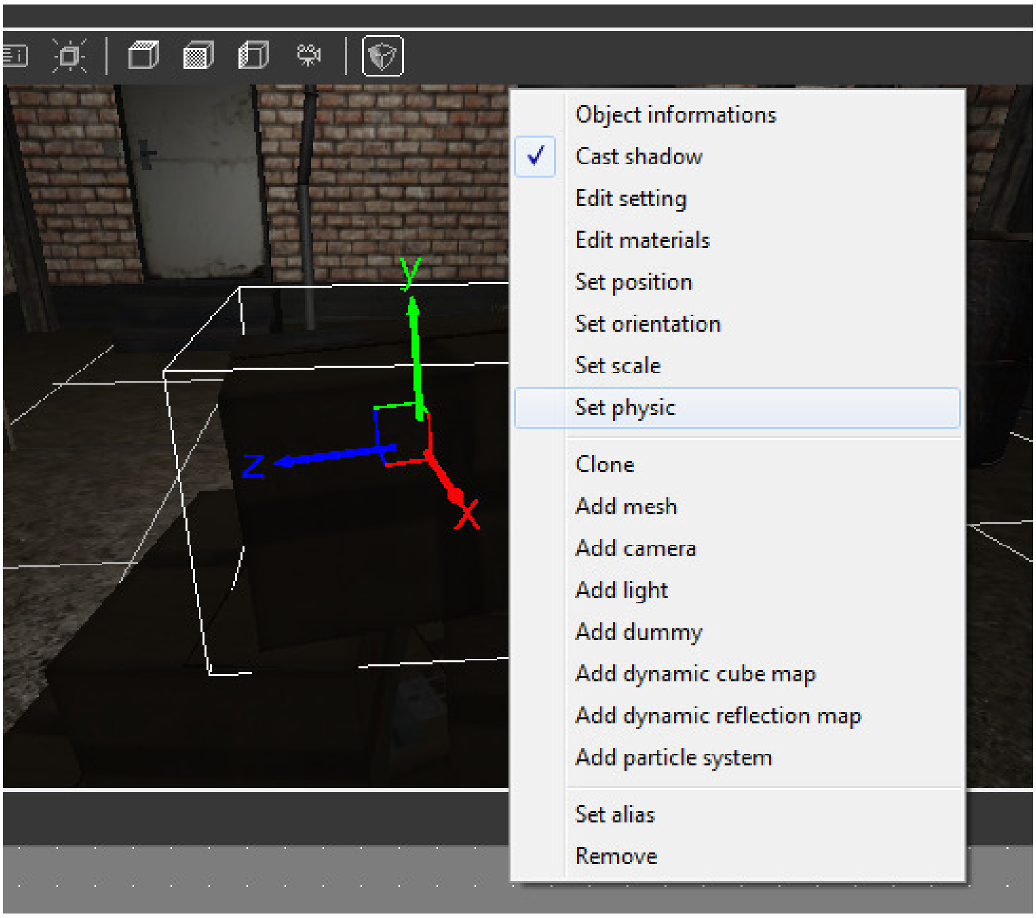
В правом верхнем углу, в меню Preferences выберем язык интерфейса редактора.

В качестве примера откроем готовую сцену tutorial01_quickstart. xos папки OpenSpace3D \demos\tutorial с помощью меню Open scene или с помощью меню Import scene.

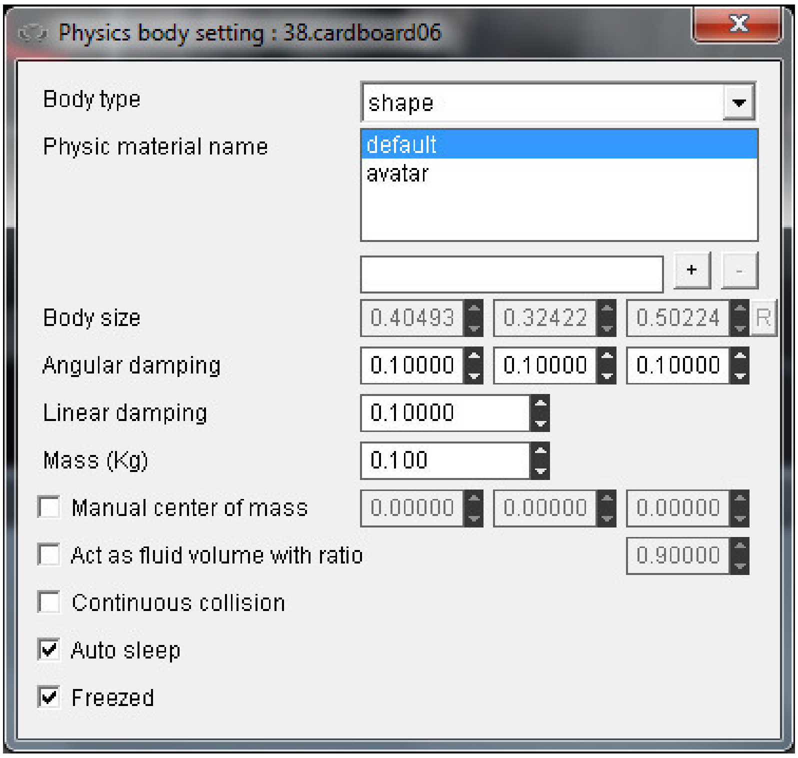




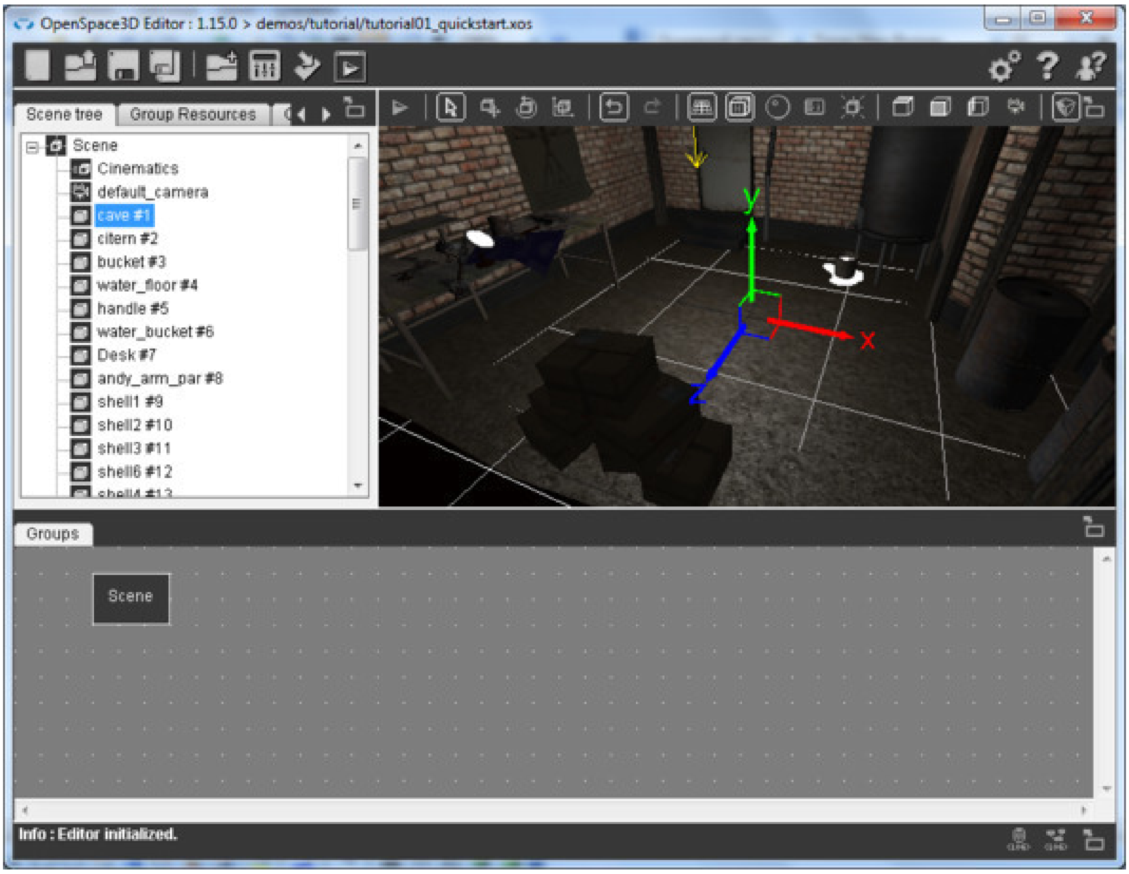
Приблизим ящик сцены – это будет узел cardboard сцены и нажмем правой кнопкой мышки.

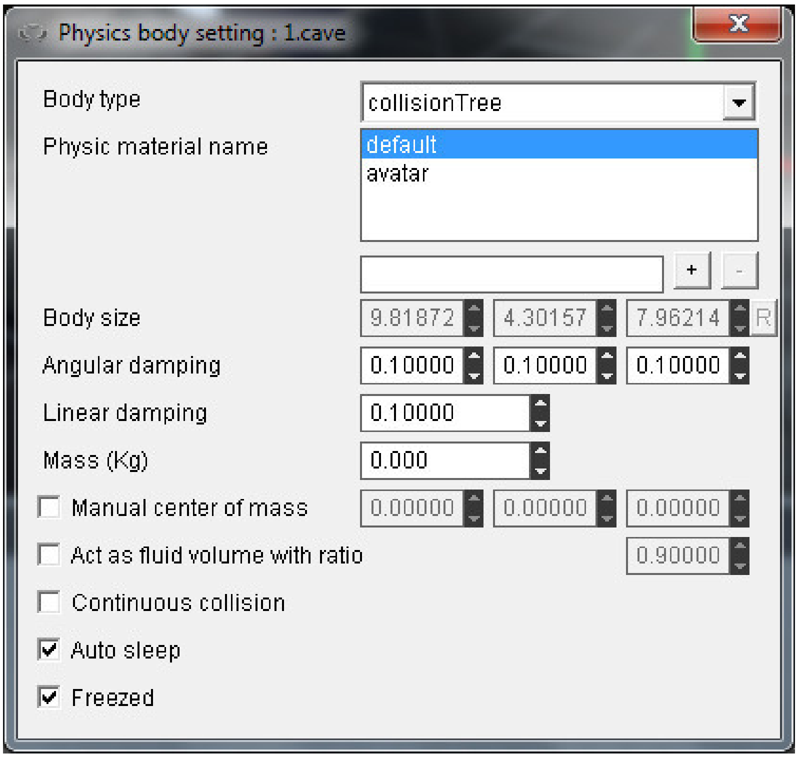


Выберем Set physic.



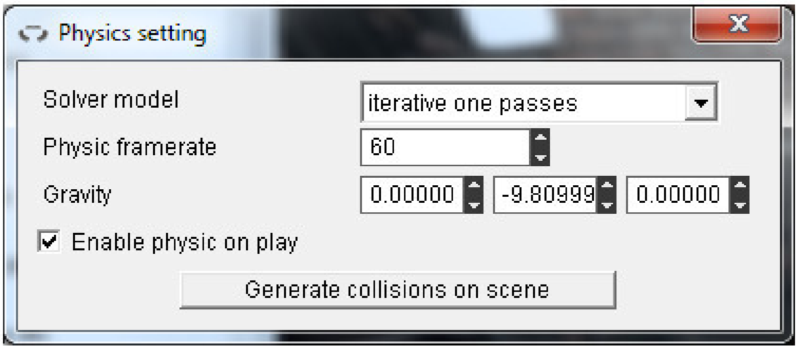
Увидим, что для этого объекта выбран Body type shape и установлена масса 0.1 кг. Выберем узел cave сцены, нажмем правой кнопкой мышки и выберем Set physic.





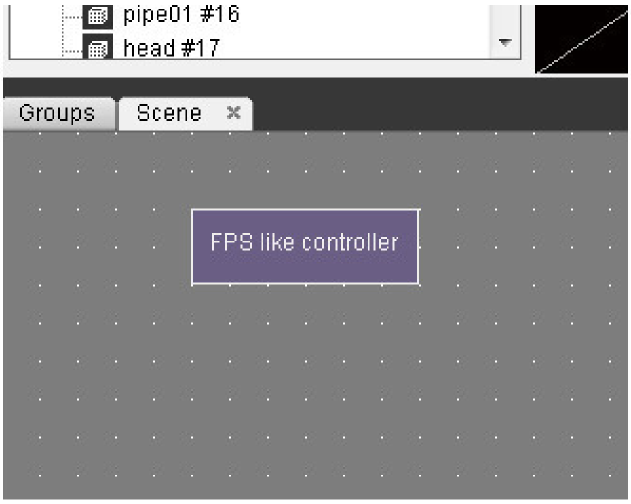
Увидим, что для этого объекта выбран Body type collisionTree.

Нажмем правой кнопкой мышки на узле Scene и выберем Set physic setting.

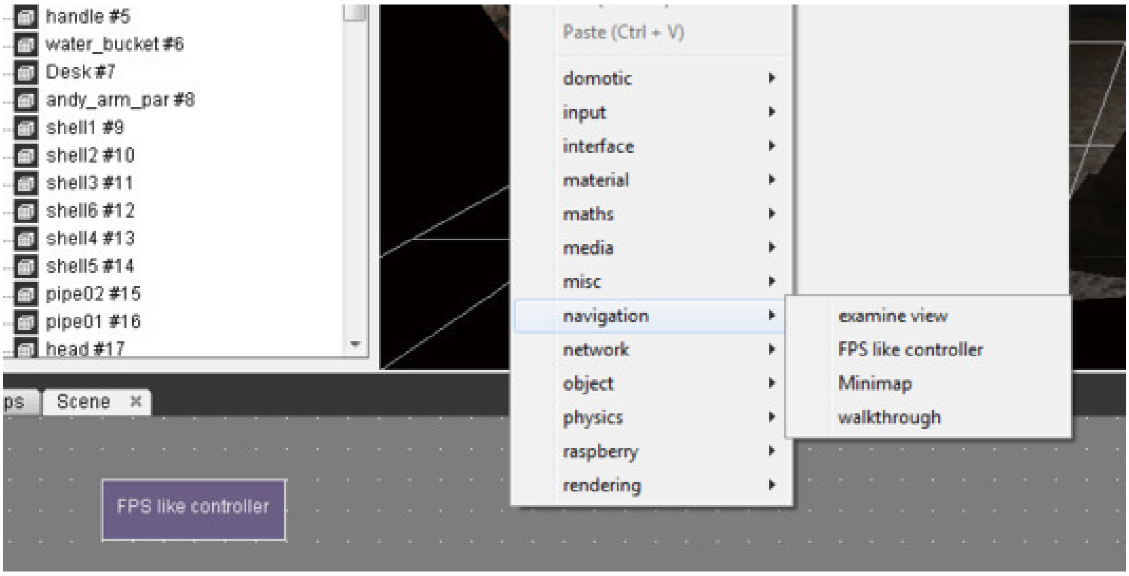


Увидим, что физика включена при проигрывании сцены, а значение ускорения свободного падения установлено как -9.8. Если минус поменять на плюс, тогда ящики окажутся на потолке.

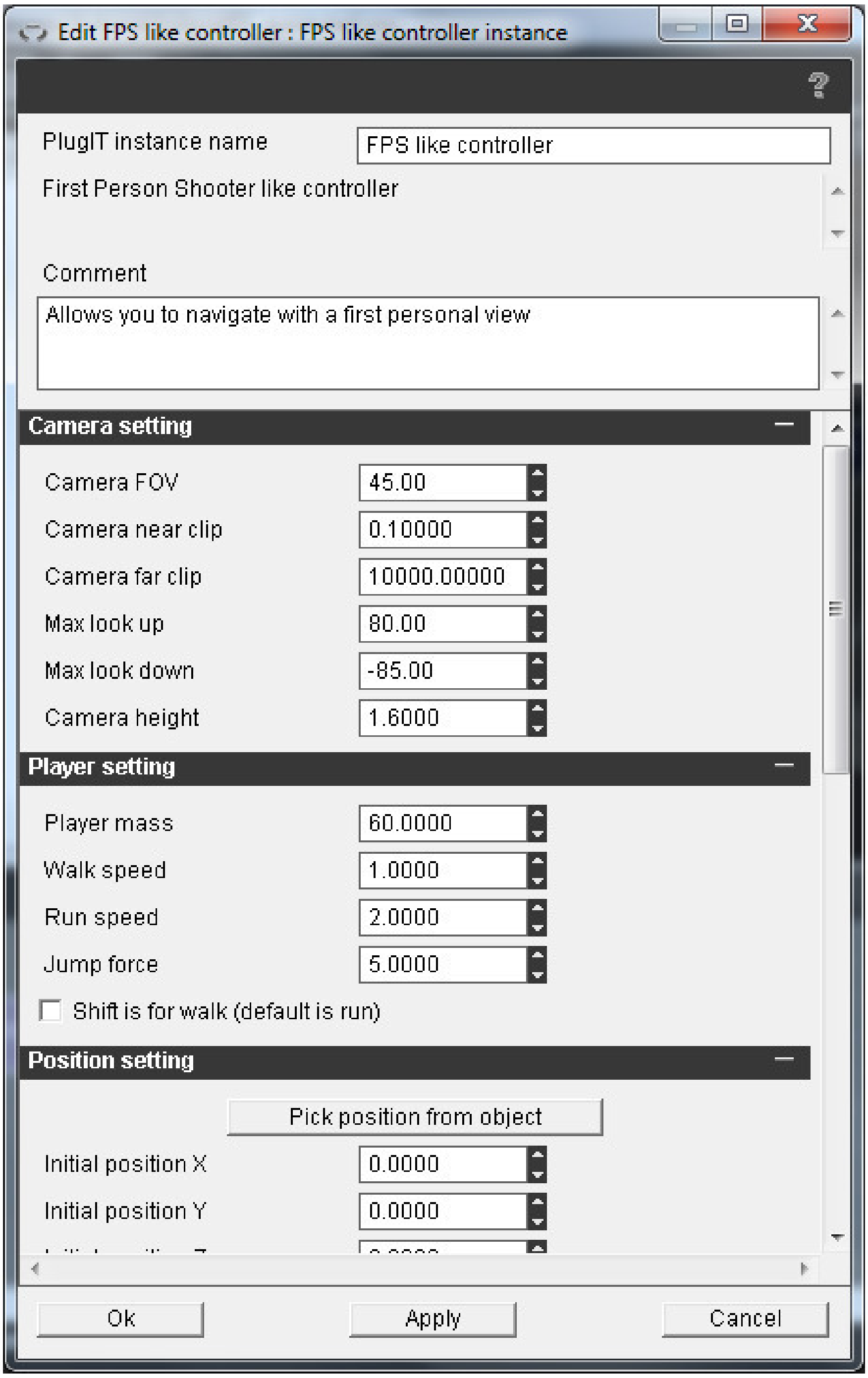
В зоне редактирования функций щелкнем два раза на сцене и во вкладке Scene увидим, что подключен компонент FPS Like Controller, обеспечивающий перемещение в сцене от первого лица.



FPS like Controller PlugIT можно подключить, нажав правой кнопкой мышки в зоне редактирования функций вкладки Scene и выбрав navigation/FPS like controller. Удалить компонент можно кнопкой Delete клавиатуры.



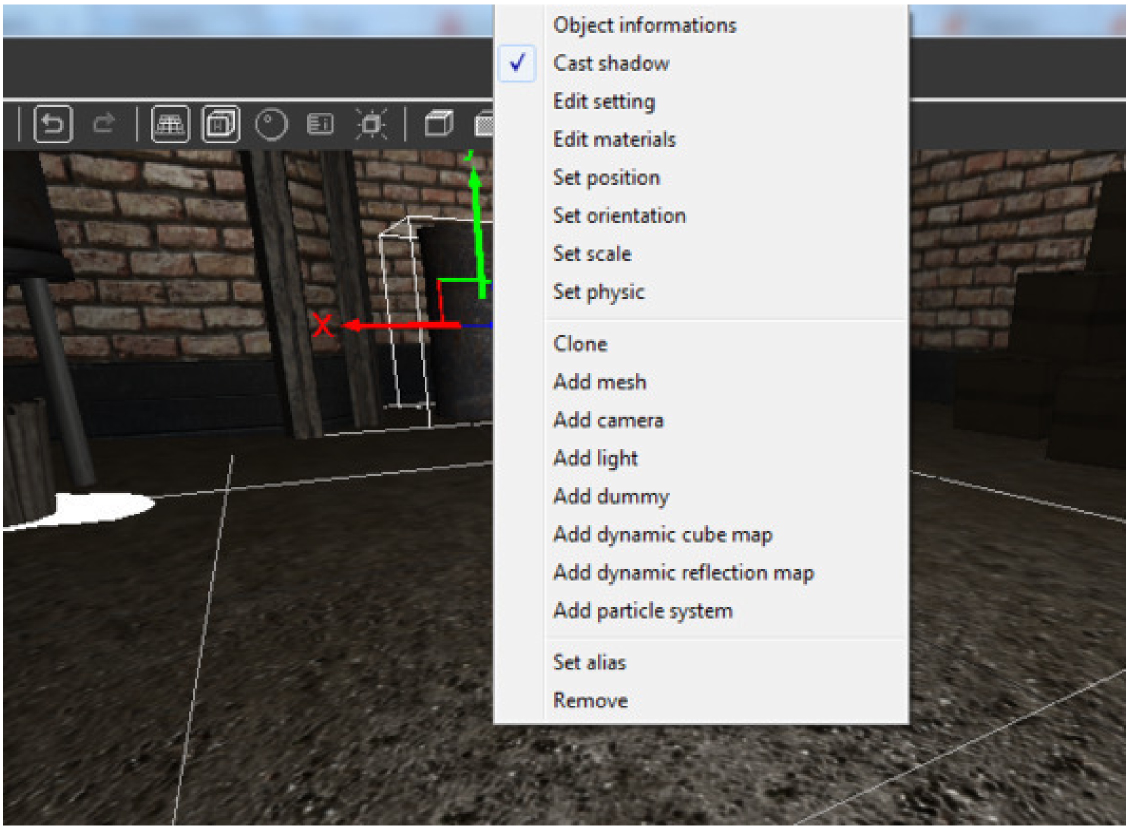
Щелкнув два раза на компоненте FPS like controller, можно установить его свойства, такие как угол камеры, скорость передвижения и др.



Нажав кнопку Play/Stop панели инструментов, походим по сцене и разобьем стопку ящиков.

В 3D представлении сцены можно добавлять, удалять и перемещать 3D объекты, а также редактировать их свойства.

Для выбора объекта сцены кликнем на нем левой кнопкой мышки. После этого для редактирования его свойств или его удаления кликнем правой кнопкой мышки.



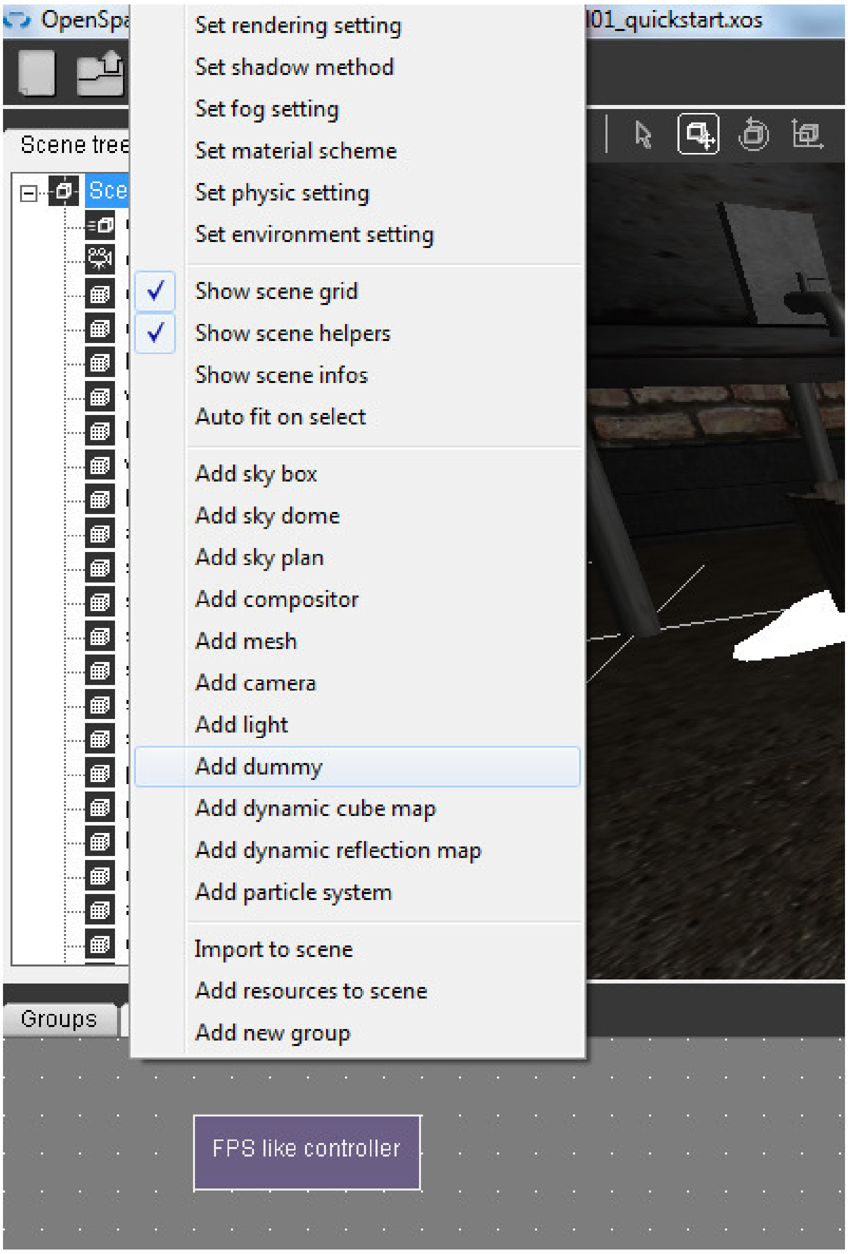
Remove удаляет объект. Clone дублирует объект, после чего нужно потянуть за ось X, чтобы разъединить объекты. Set orientation позволяет положить объект.

Перемещать объекты по сцене можно, нажав кнопку Move панели инструментов и перетаскивая объект за оси.

Кнопка Rotate панели инструментов позволяет вращать объект, а кнопка Scale панели инструментов позволяет масштабировать объект.

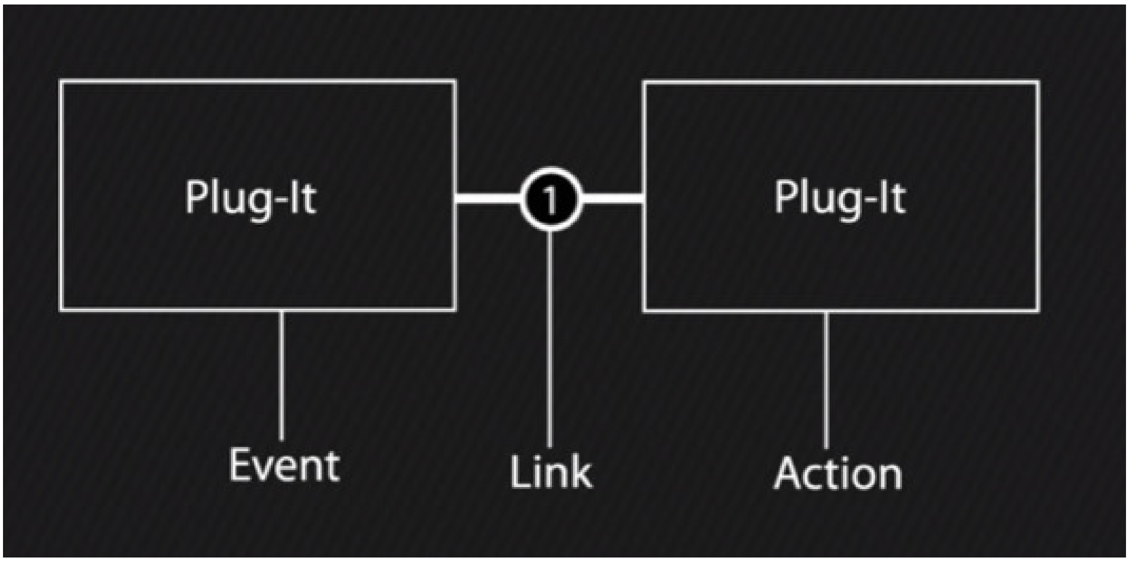
Чтобы придать реальность объекту, в его свойствах выберем Set physic, выберем объекту форму shape и установим его массу, при этом в узле cave сцены должна быть установлена форма collisionTree. Если отмечен флажок Freezed, тогда физика объекта будет ожидать столкновения, перед тем как заработать.

Чтобы объединить объекты в группу, на узле Scene нажмем правой кнопкой мышки и выберем Add dummy, затем в дереве сцены перетащим объекты в созданный узел dummy.



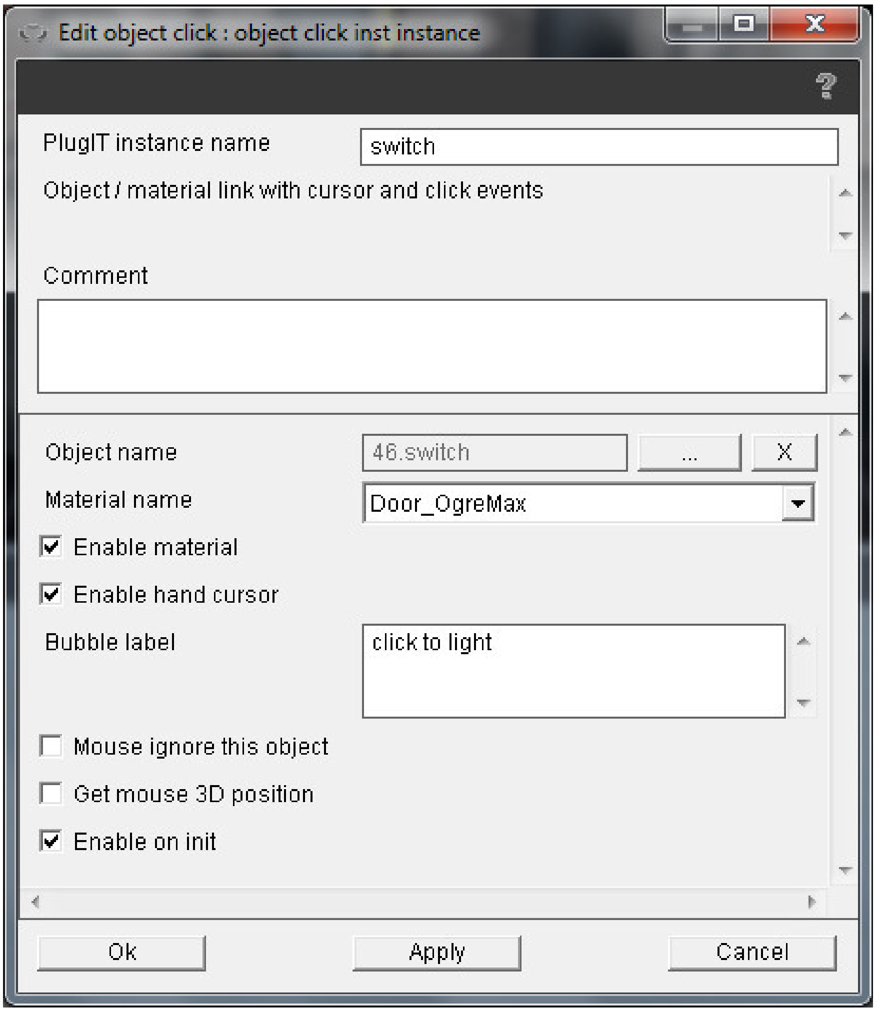
Теперь можно манипулировать сразу группой объектов.

Компоненты PlugIT обеспечивают взаимодействие пользователя со сценой. Связываясь между собой, компоненты PlugIT образуют систему событий и действий.

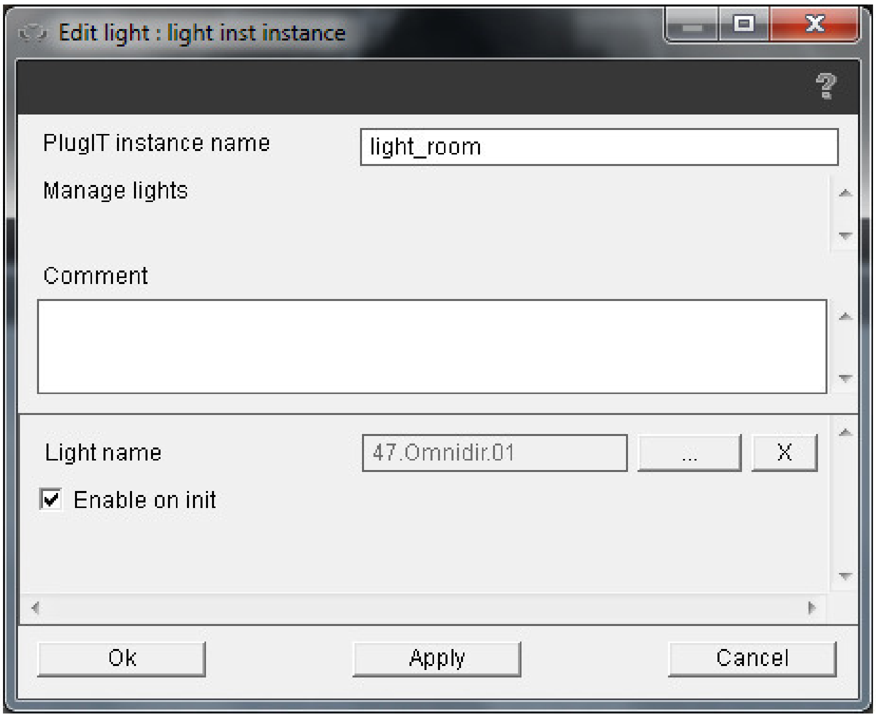


В зоне редактирования функций вкладки Scene удалим PlugIT FPS like Controller с помощью клавиши Delete клавиатуры.

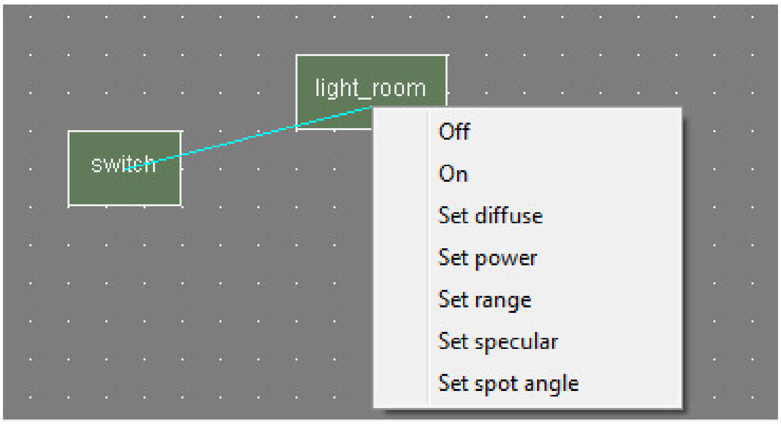
Нажмем правой кнопкой мышки на зоне редактирования функций и выберем object/ object click. Присвоим имя switch экземпляру PlugIT. В поле Object name свяжем экземпляр PlugIT с объектом switch дерева сцены. Отметим флажок Enable hand cursor. В поле Bubble label введем click to light и нажмем Ok.



Нажмем правой кнопкой мышки на зоне редактирования функций и выберем object/light, свяжем этот PlugIT экземпляр с объектом Omnidir.



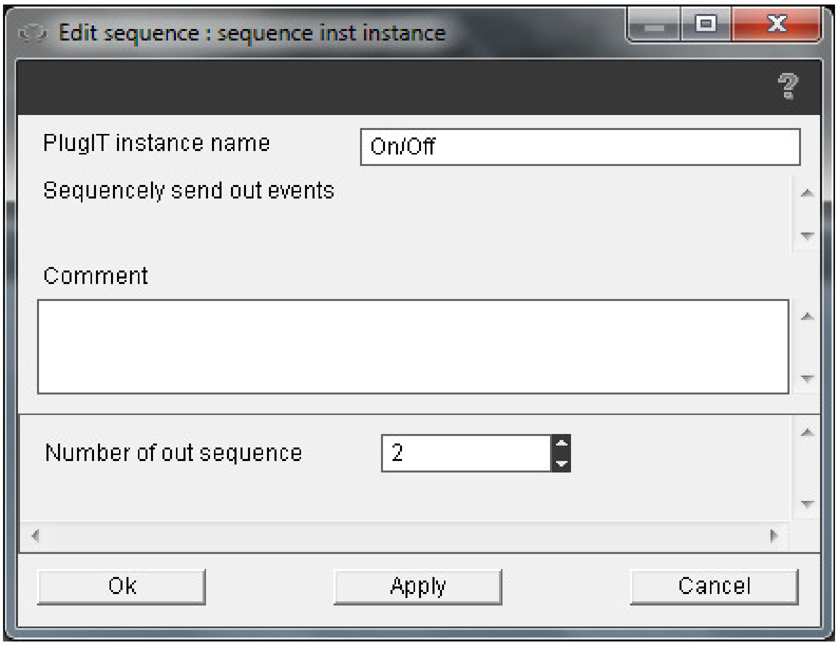
Нажмем правой кнопкой мышки на экземпляре switch и выберем LeftClick, свяжем его с экземпляром light_room, выбрав Off.



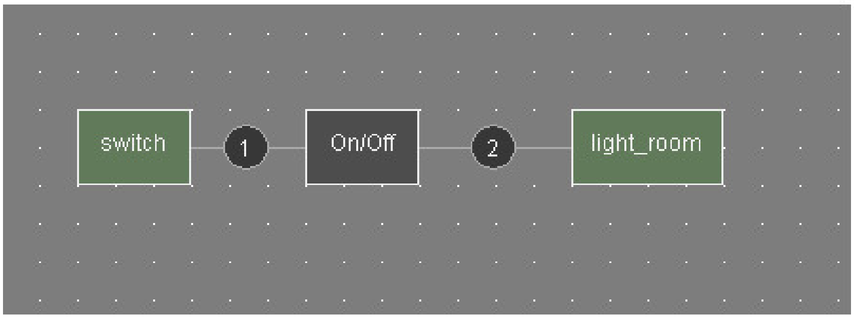
Теперь при проигрывании сцены, при нажатии левой кнопкой мышки на выключателе у двери, лампочка будет выключаться.

С помощью клавиши Delete удалим связь между экземплярами.

Нажмем правой кнопкой мышки на зоне редактирования функций и выберем misc/sequence.



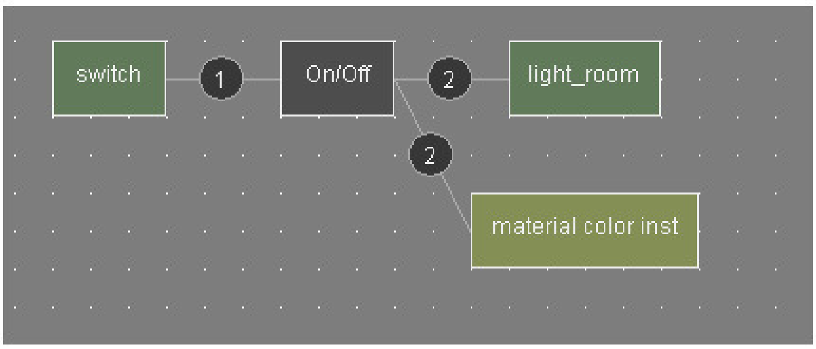
Нажмем правой кнопкой мышки на экземпляре switch и выберем LeftClick, свяжем его с экземпляром On/Off sequence, выбрав Input. Нажмем правой кнопкой мышки на экземпляре On/Off и выберем Out1, свяжем его с экземпляром light_room, выбрав Off. Нажмем правой кнопкой мышки на экземпляре On/Off и выберем Out2, свяжем его с экземпляром light_room, выбрав On.



Теперь при проигрывании сцены, при нажатии левой кнопкой мышки на выключателе у двери, лампочка будет выключаться и включаться.

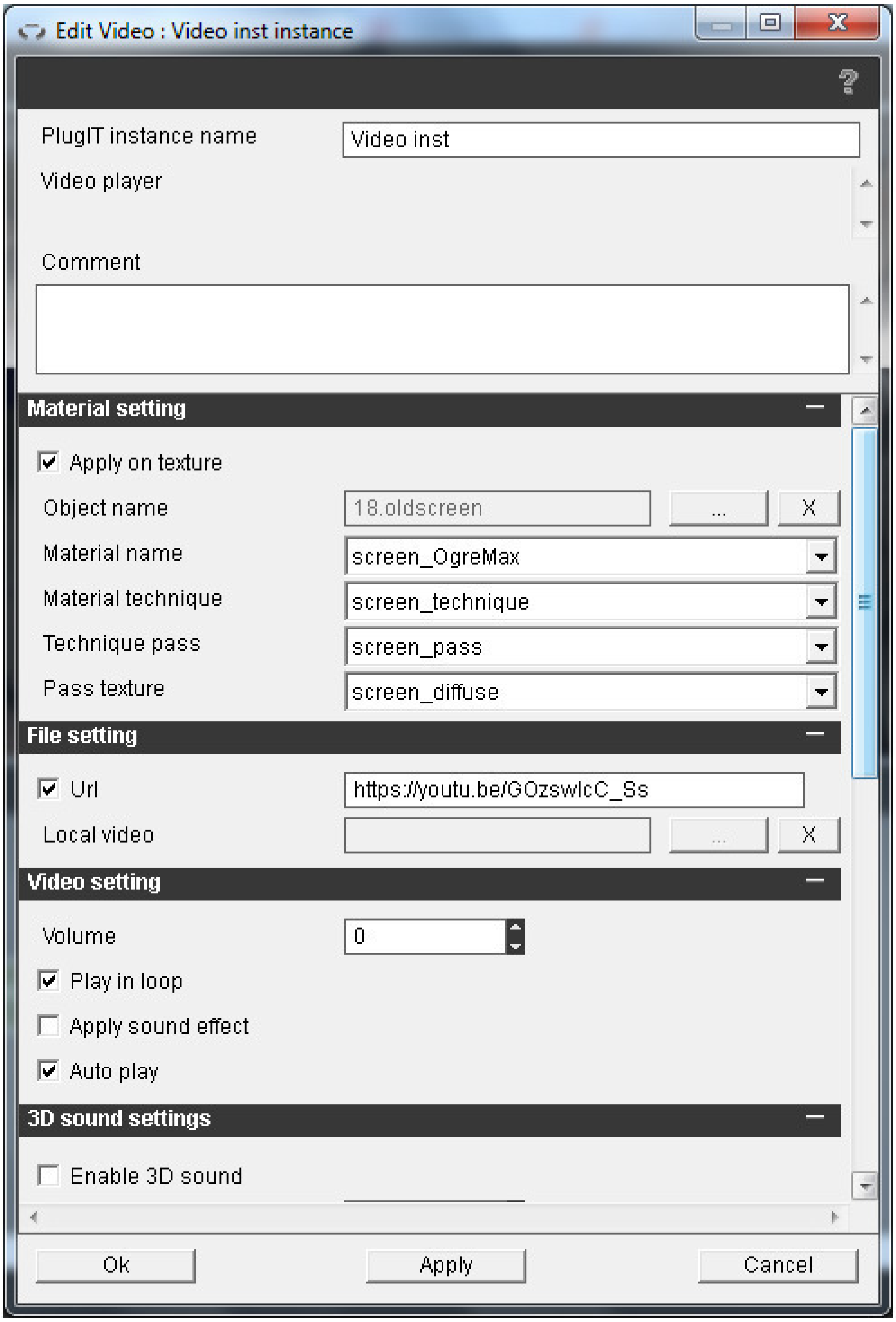
Однако при выключении цвет лампочки остается желтым. Чтобы это исправить нажмем правой кнопкой мышки на зоне редактирования функций и выберем material/material color. Свяжем этот экземпляр с объектом lamp. Установим цвет материала.

Нажмем правой кнопкой мышки на экземпляре On/Off и выберем Out1, свяжем его с экземпляром material color, выбрав Enable. Нажмем правой кнопкой мышки на экземпляре On/Off и выберем Out2, свяжем его с экземпляром material color, выбрав Disable.



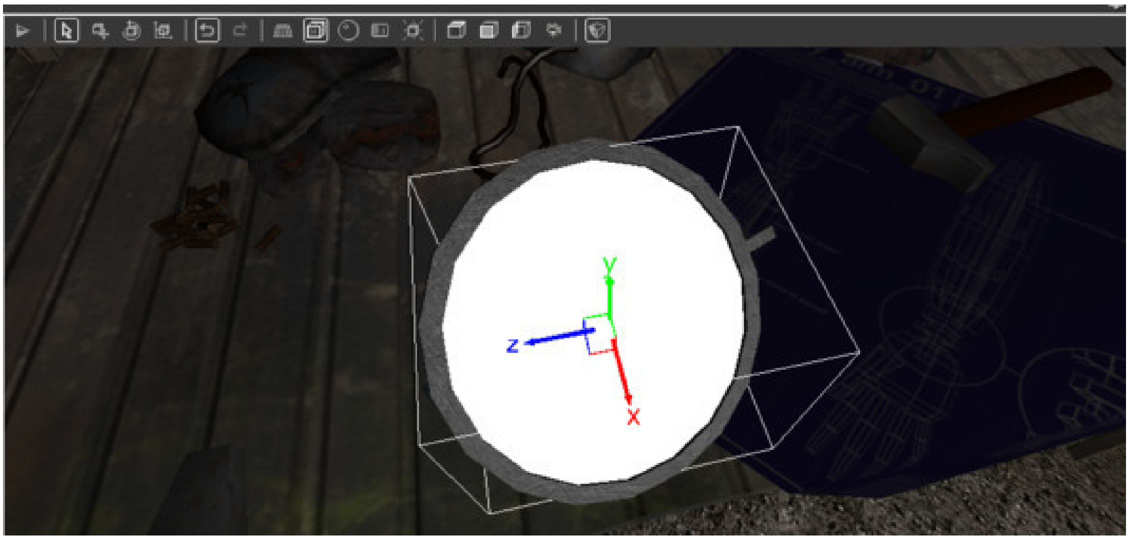
Теперь при проигрывании сцены, при нажатии левой кнопкой мышки на выключателе у двери, лампочка будет выключаться и включаться, а ее цвет будет меняться.

Сделаем так, чтобы на телевизоре сцены проигрывалось видео. Для этого нажмем правой кнопкой мышки на зоне редактирования функций и выберем media/Video. Отметим флажок Apply on texture и свяжем этот экземпляр с объектом oldscreen. В поле Url введем ссылку. Отметим флажок Auto play и Play in loop, в поле Volume поставим 0, уберем флажок Transparency.



Теперь по телевизору можно смотреть видео.

Выберем объект lense и нажмем Zoom on selected object в панели инструментов.



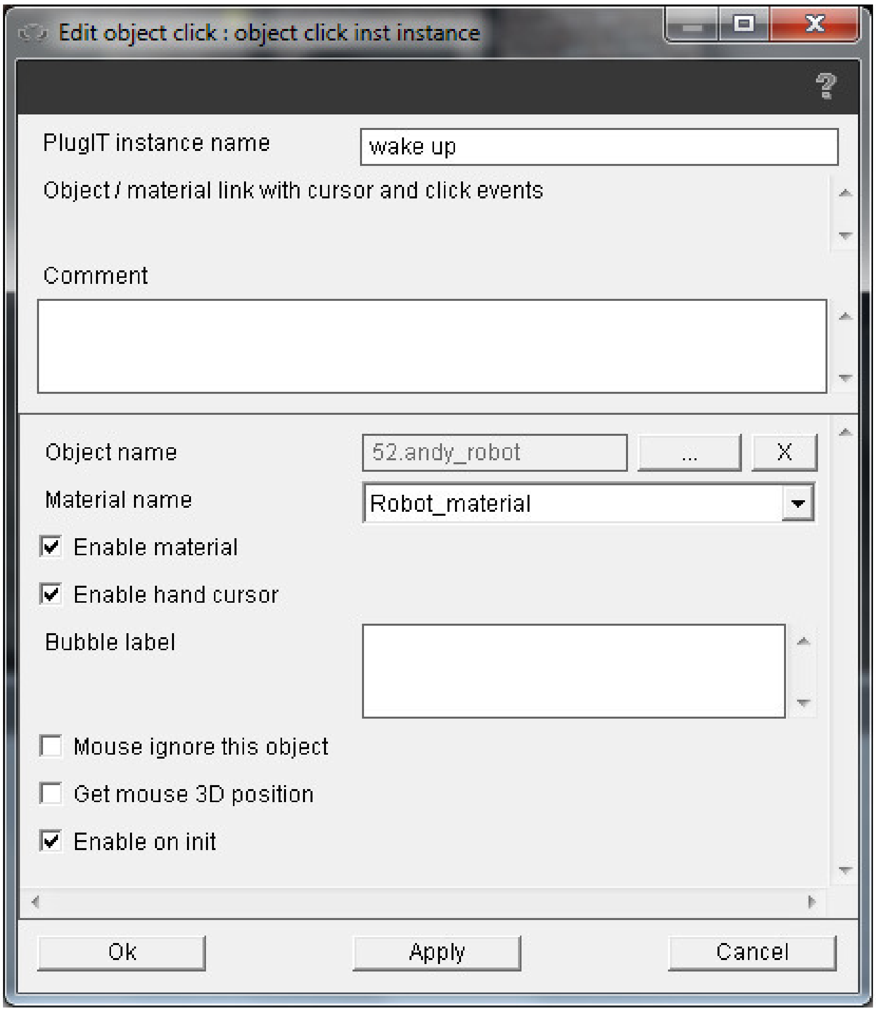
Нажмем правой кнопкой на объекте и выберем Add dynamic reflection map. Выберем материал Lense, текстуру reflection и размер 512, отметим флажок Enable.

Нажмем правой кнопкой на объекте и выберем Add dynamic reflection map. Выберем материал Lense, текстуру refraction и размер 512, отметим флажок Enable и Revert clip plane. В результате получим нормальную линзу.

Импортируем в сцену робота. Для этого выберем меню Import scene/Standard file formats/ OpenSpace3D\demos\tutorial\andy.scene/As a new group/Do you want to import this environment setting/No.

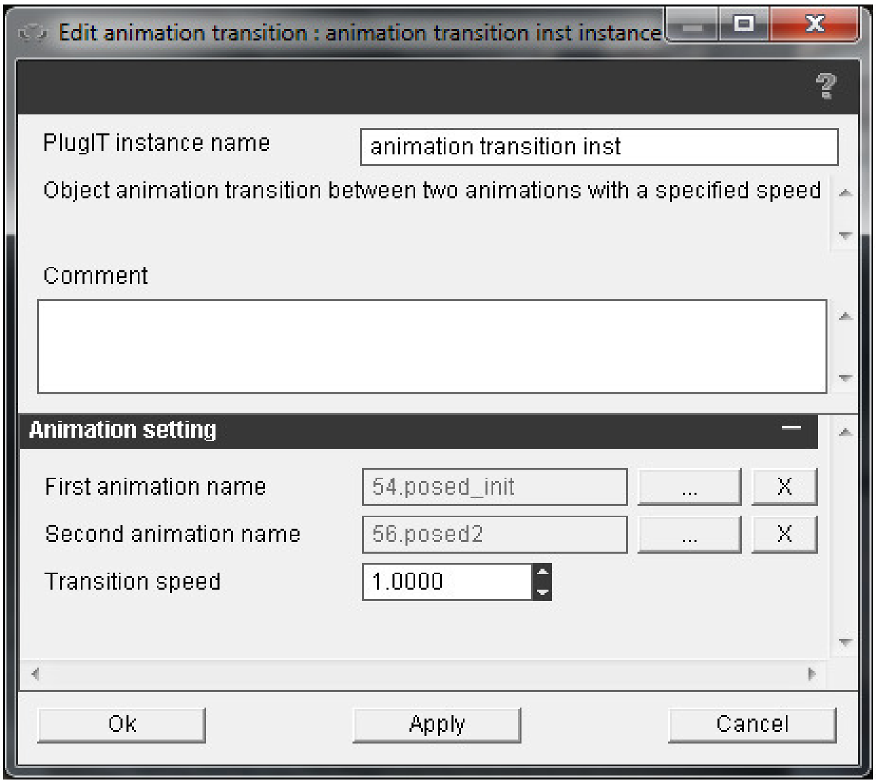
Нажмем правой кнопкой мышки на узле walk_inplace и уберем флажок Enable.

Для этого нажмем правой кнопкой мышки на зоне редактирования функций и выберем object/object click. Свяжем этот экземпляр с объектом andy_robot.

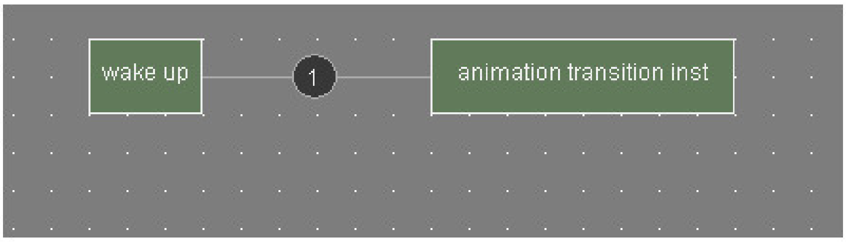


Нажмем правой кнопкой мышки на узле pose_init и отметим флажок Enable.

Для этого нажмем правой кнопкой мышки на зоне редактирования функций и выберем object/animation transition. First animation name свяжем с posed_init, а Second animation name свяжем с posed2.

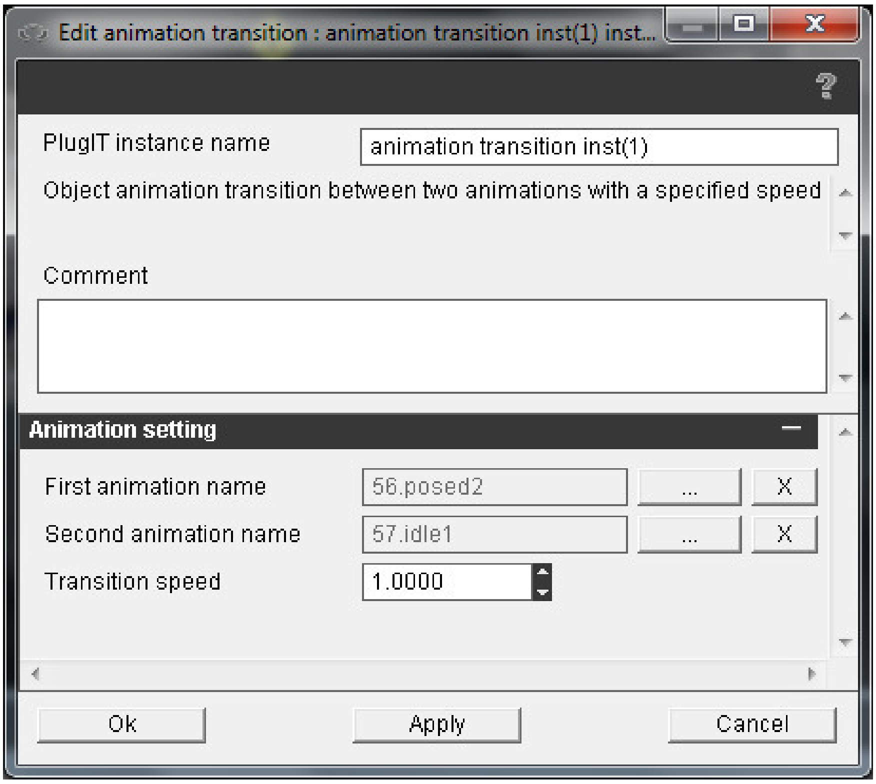


Нажмем правой кнопкой мышки на wake up и выберем LeftClick и свяжем с animation transition, выбрав Fade in.

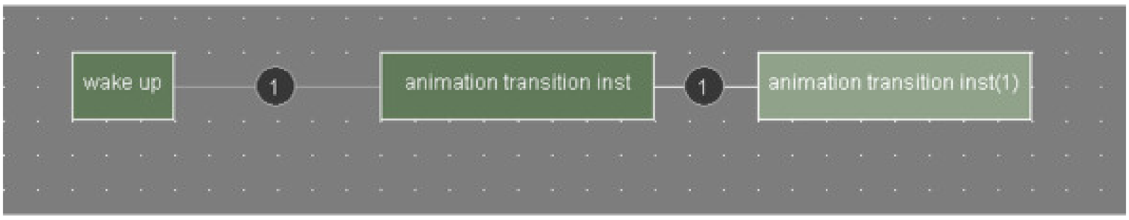


Теперь, при проигрывании сцены, при нажатии на роботе, он будет выпрямляться.

Нажмем правой кнопкой мышки на зоне редактирования функций и выберем object/animation transition. First animation name свяжем с posed2, а Second animation name свяжем с idle1.



Нажмем правой кнопкой мышки на animation transition и выберем Transition ended и свяжем с animation transition1, выбрав Fade in.



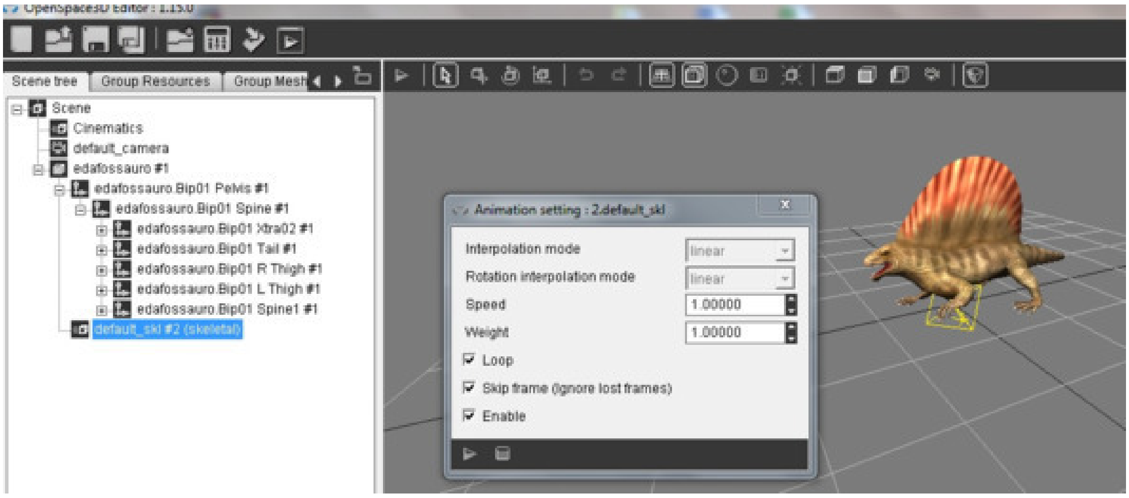
Нажмем правой кнопкой мышки на узле idle и выберем Edit setting. Отметим флажок Loop.

Таким образом, получим двухступенчатую анимацию.

Приступим к созданию дополненной реальности.

Создадим новую сцену, в которую импортируем модель OpenSpace3D\assets\models \library\Primitive_World\Edaphosaurus\Edafossauro.mesh.

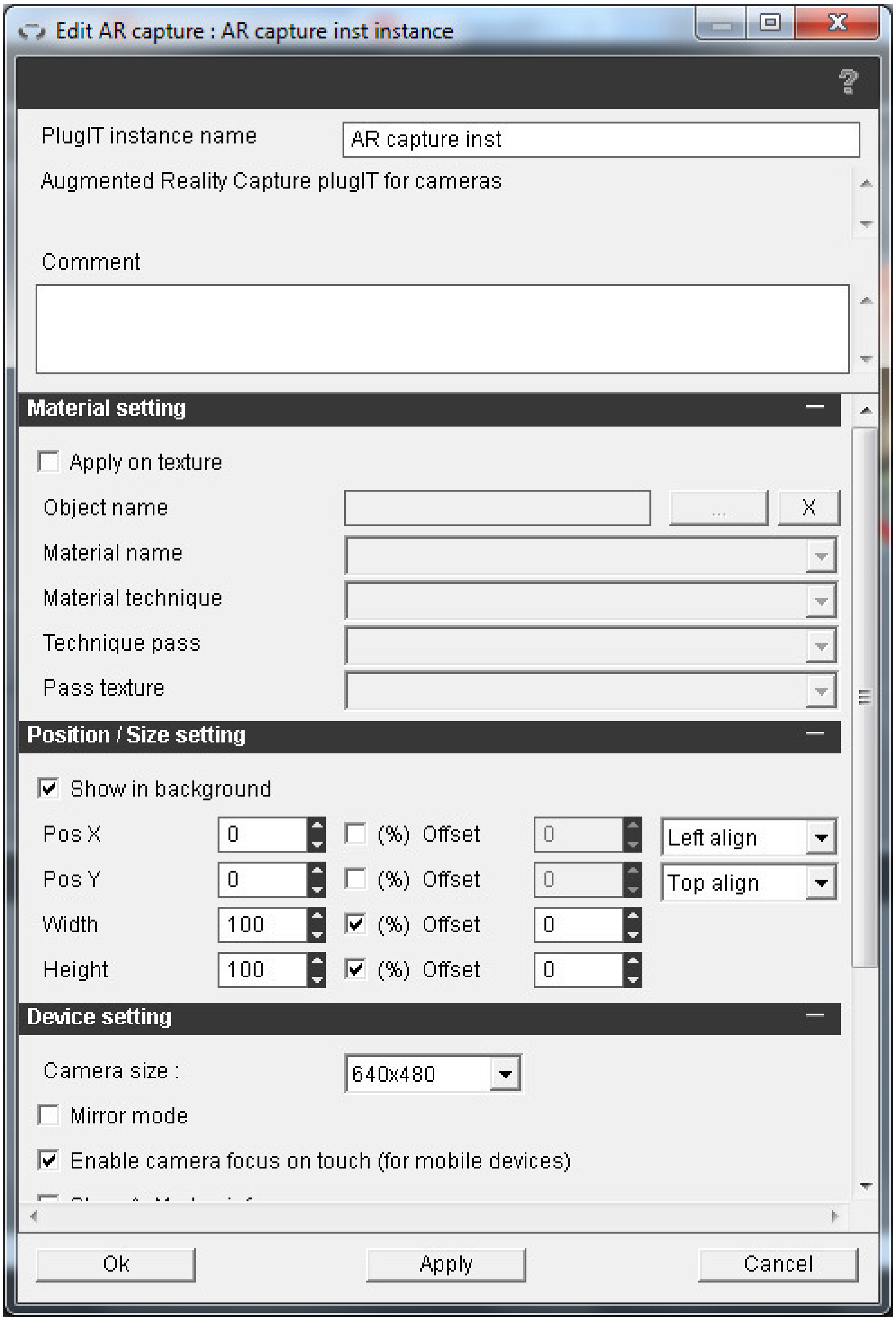
Нажмем правой кнопкой мышки на узле default_skl, выберем Edit setting и отметим флажок Loop.



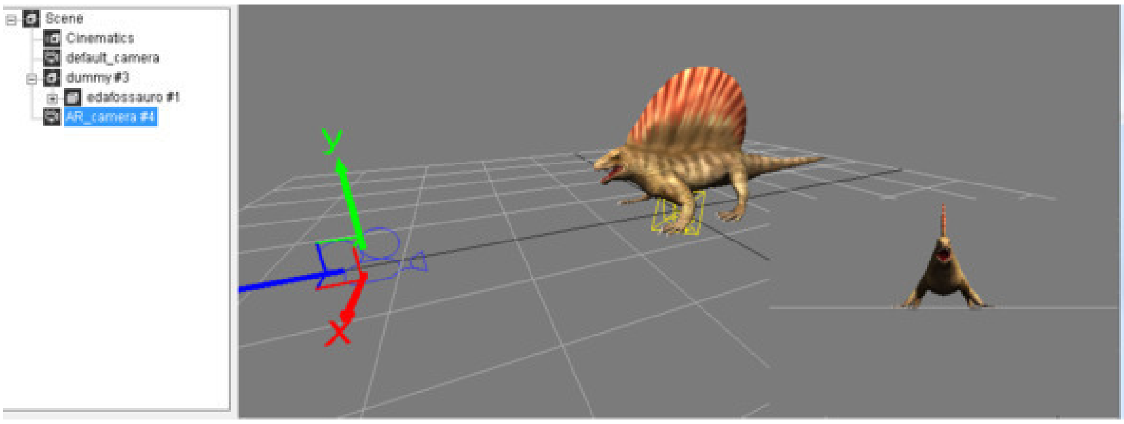
Нажмем правой кнопкой мышки на узле Scene и выберем Add dummy.

Перетащим модель в узел dummy.

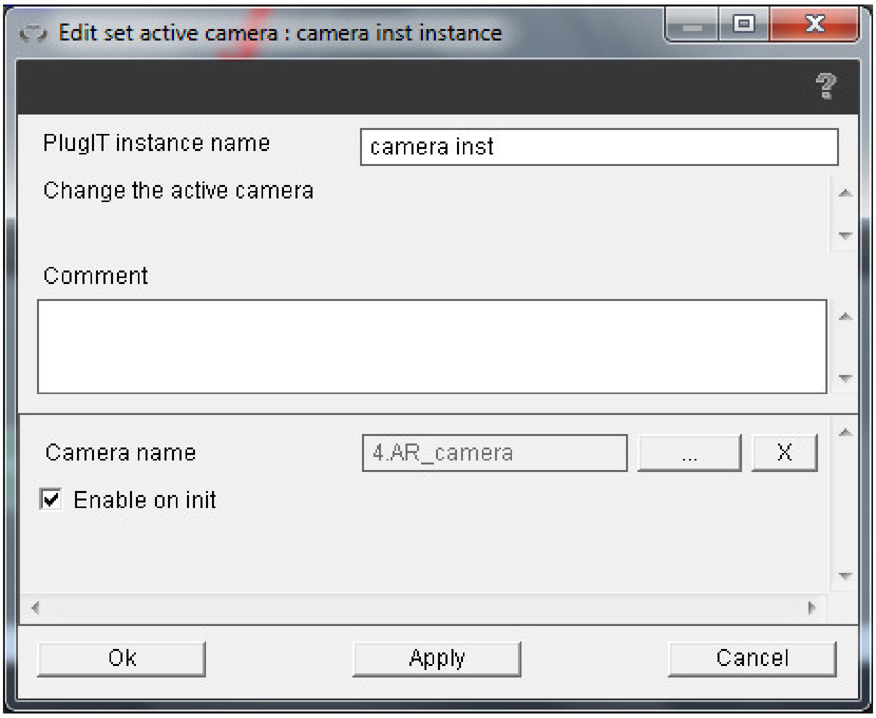
Нажмем правой кнопкой мышки на зоне редактирования функций и выберем input/AR capture.



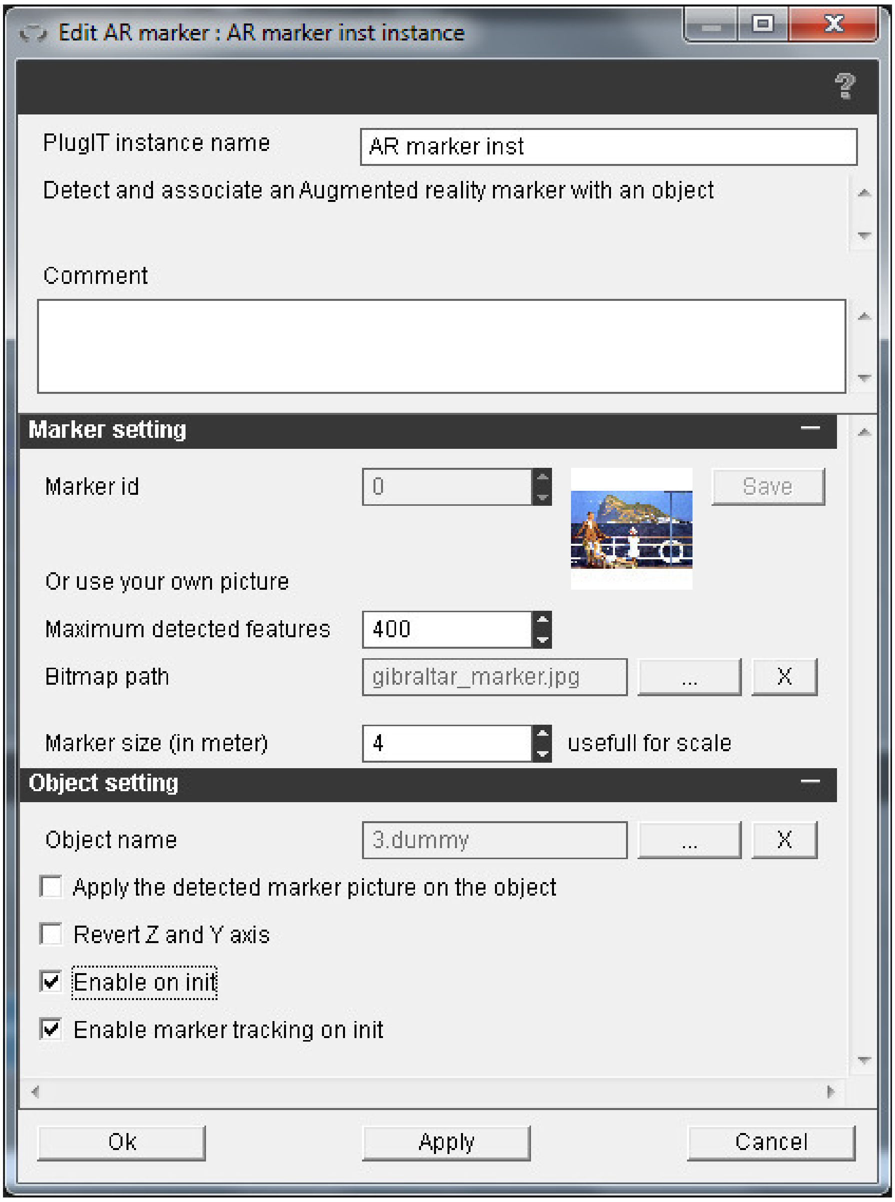
Нажмем правой кнопкой мышки на узле Scene и выберем Add camera.
Нажмем кнопку Move панели инструментов и перетащим камеру по оси Z.



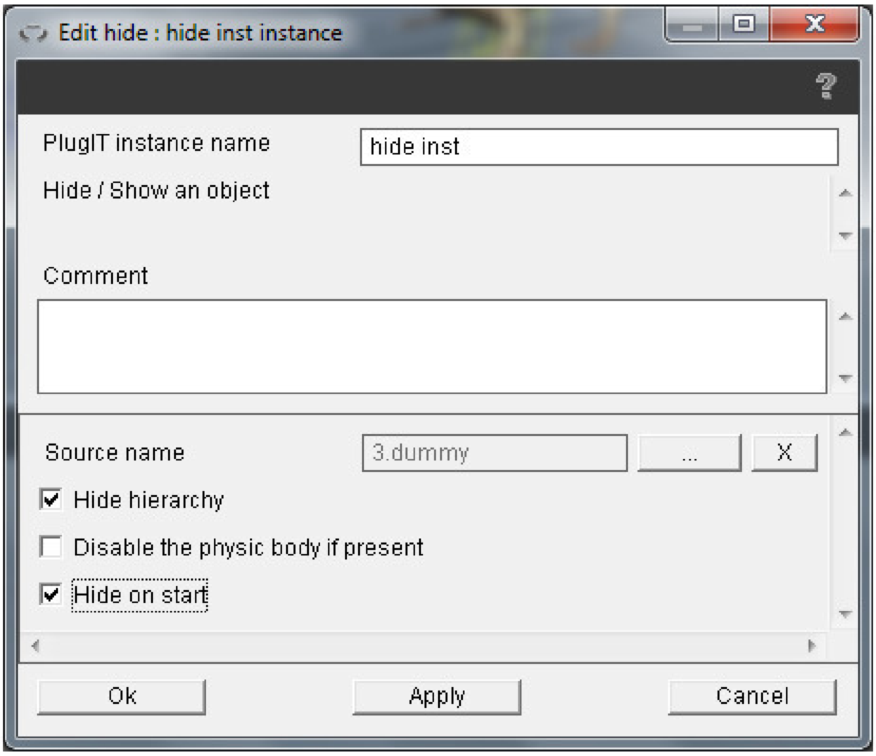
Нажмем правой кнопкой мышки на зоне редактирования функций и выберем object/set active camera. Свяжем этот экземпляр с объектом сцены AR_camera. Отметим флажок Enable on init.



Нажмем правой кнопкой мышки на зоне редактирования функций и выберем input/AR marker. В поле Bitmap path выберем произвольное изображение, которое поместим в каталог OpenSpace3D и распечатаем. Подождем пока изображение не будет обработано в качестве маркера. Изменим размер маркера и свяжем его с объектом dummy.



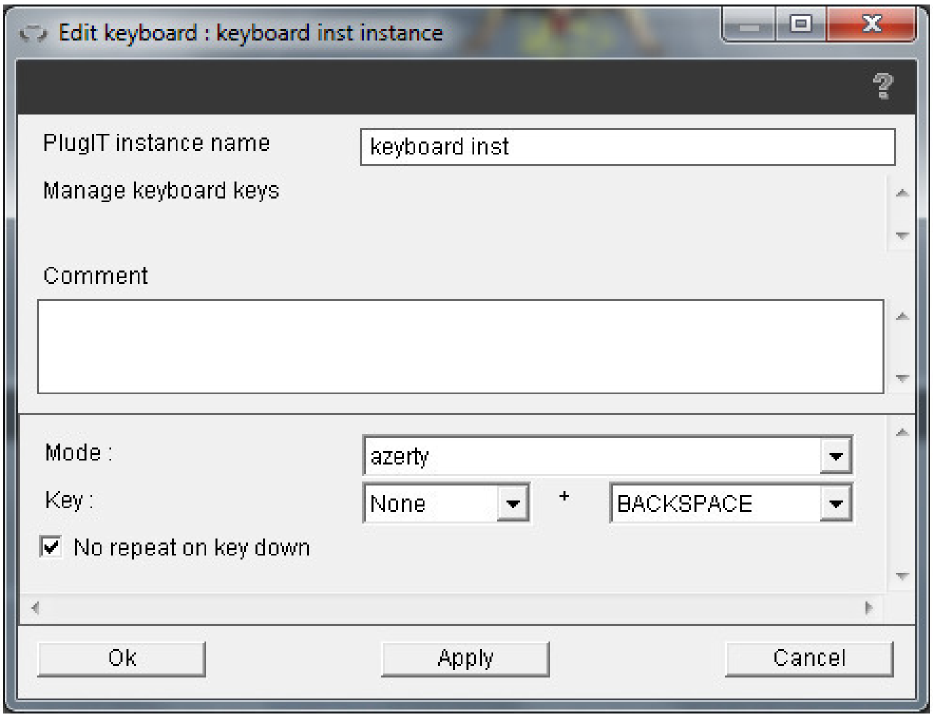
Нажмем правой кнопкой мышки на зоне редактирования функций и выберем oblect/hide. Свяжем этот экземпляр с объектом dummy и отметим флажок Hide on start.



Нажмем правой кнопкой мышки на экземпляре AR marker и выберем Found, соединим его с экземпляром hide и выберем Show.

Нажмем правой кнопкой мышки на экземпляре AR marker и выберем Lost, соединим его с экземпляром hide и выберем Hide.

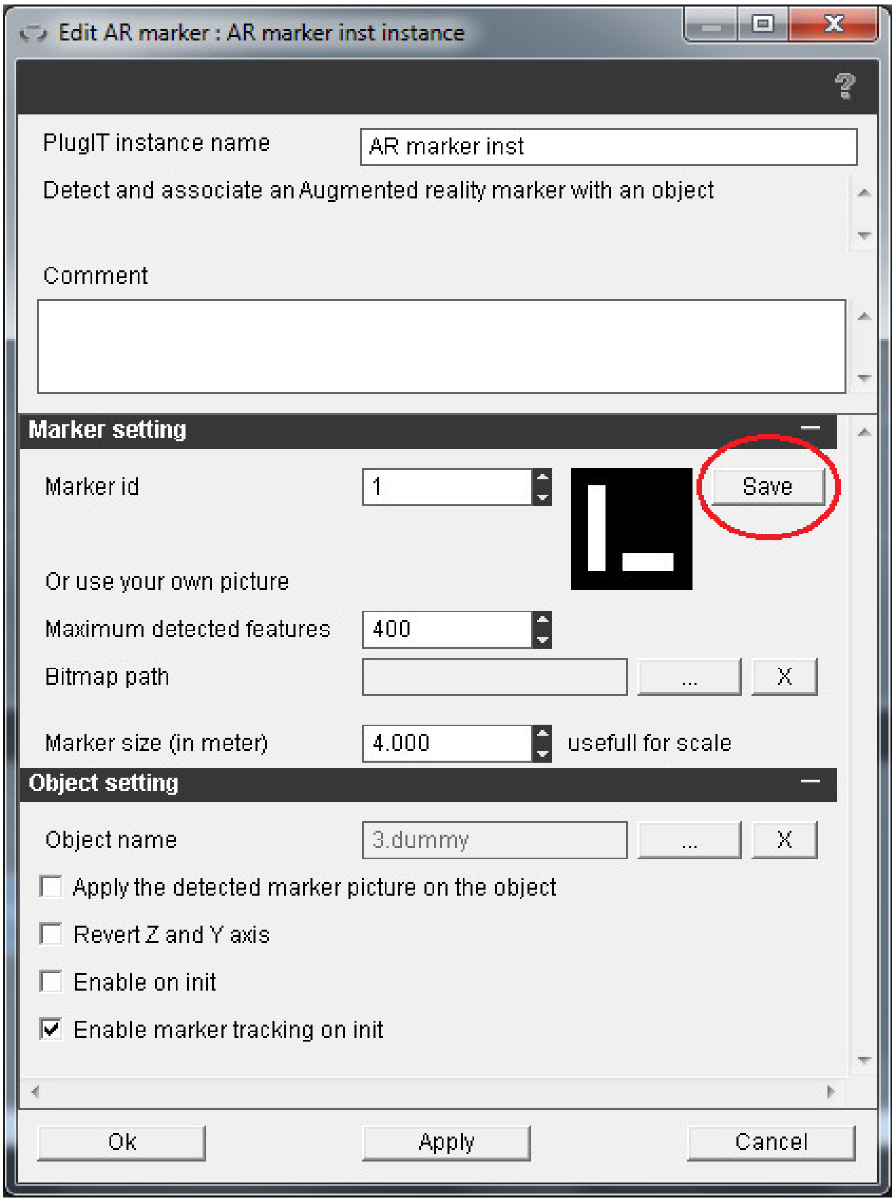
Нажмем правой кнопкой мышки на зоне редактирования функций и выберем input/keyboard. Выберем клавишу.



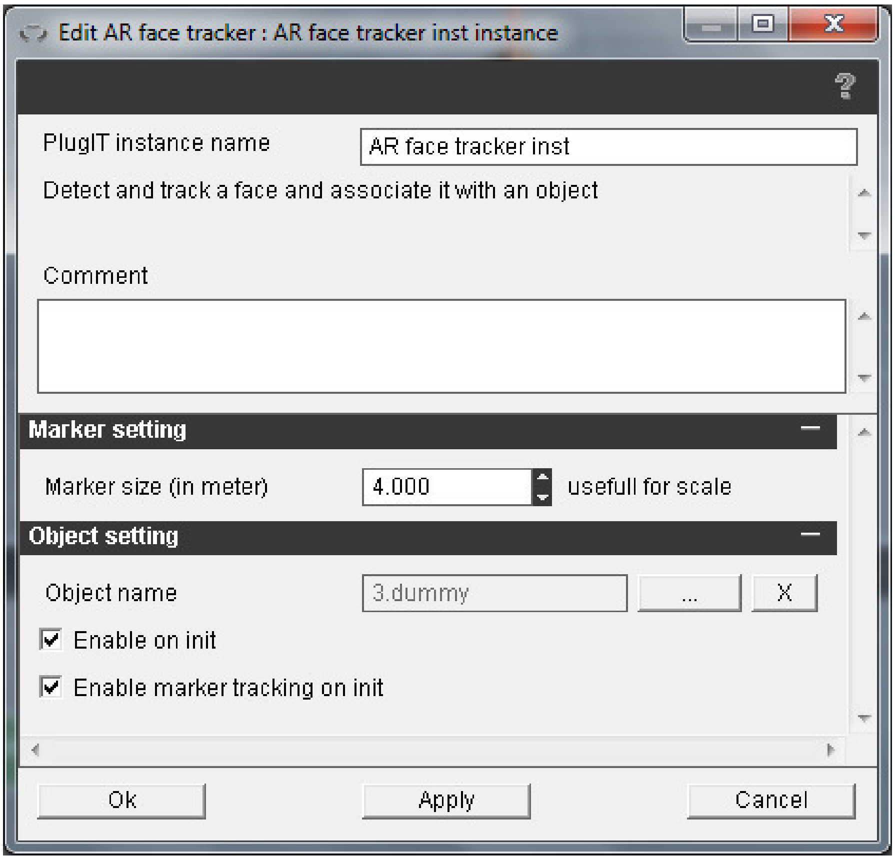
Нажмем правой кнопкой мышки на экземпляре keyboard и выберем Key down, соединим его с экземпляром AR marker, выбрав Register current frame.

Теперь маркер будет регистрироваться динамически, при нажатии соответствующей клавиши – поднесем к камере изображение и нажмем клавишу – должна появиться 3D модель.

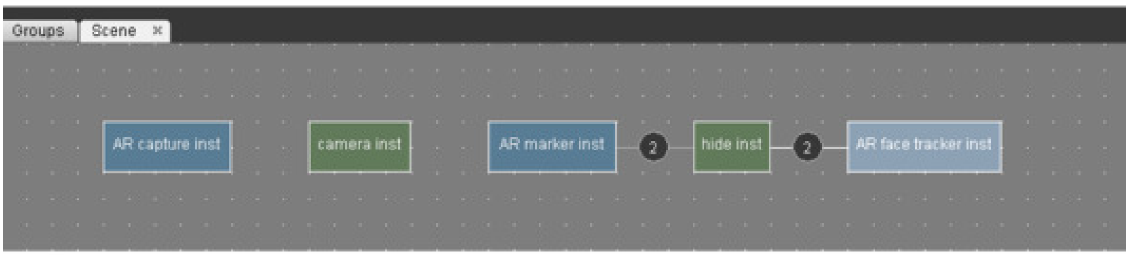
В свойствах экземпляра AR marker, в качестве маркера можно установить реперный маркер, сохранив его для распечатывания.



Нажмем правой кнопкой мышки на зоне редактирования функций и выберем input/AR face tracker. Изменим размер маркера и свяжем его с объектом dummy.



Нажмем правой кнопкой мышки на экземпляре AR face tracker и выберем Found, соединим его с экземпляром hide и выберем Show. Нажмем правой кнопкой мышки на экземпляре AR face tracker и выберем Lost, соединим его с экземпляром hide и выберем Hide.



Теперь, если посмотреть прямо в камеру, появится 3D модель.

Соберем проект в Android приложение. Для этого в меню Export to Openspace3D player выберем As an Android Application.

Добавим или выберем хранилище ключей и в результате получим Android проект с готовым APK файлом в папке bin.

BeyondAR

BeyondAR это фреймворк, обеспечивающий ресурсы для разработки приложений с дополненной реальностью, основанной на георасположении на смартфонах и планшетах.

Для начала работы скачаем BeyondAR фреймворк с Github https://github.com/BeyondAR/beyondar.

Импортируем проект BeyondAR_Examples в среду разработки Android Studio.

Для запуска этого приложения на Android устройстве требуется наличие датчика ориентации.

Для сборки APK файла с большим количеством методов в коде, в Gradle файл добавим:

```
defaultConfig {

multiDexEnabled true
}
dependencies {

compile 'com.android.support: multidex:1.0.0»
}
android {

dexOptions {
javaMaxHeapSize «4g»
}
}
```

В файл манифеста:

```
<application

android:name="android.support.multidex.MultiDexApplication»>
```

Для двух классов, возможно, придется реализовать OnMapReadyCallback.

```
package com.beyondar. example;

import android.content.Context;
import android. location. LocationManager;
import android. os. Bundle;
import android.support.v4.app.FragmentActivity;
import android.view.View;
import android.view.View. OnClickListener;
import android. widget. Button;
import android.widget.Toast;

import com.beyondar.android.plugin. googlemap. GoogleMapWorldPlugin;
import com.beyondar.android.util.location.BeyondarLocationManager;
import com.beyondar.android.world.GeoObject;
import com.beyondar.android. world. World;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps. GoogleMap;
import com.google.android.gms.maps. GoogleMap. OnMarkerClickListener;
import com.google.android.gms.maps. OnMapReadyCallback;
```

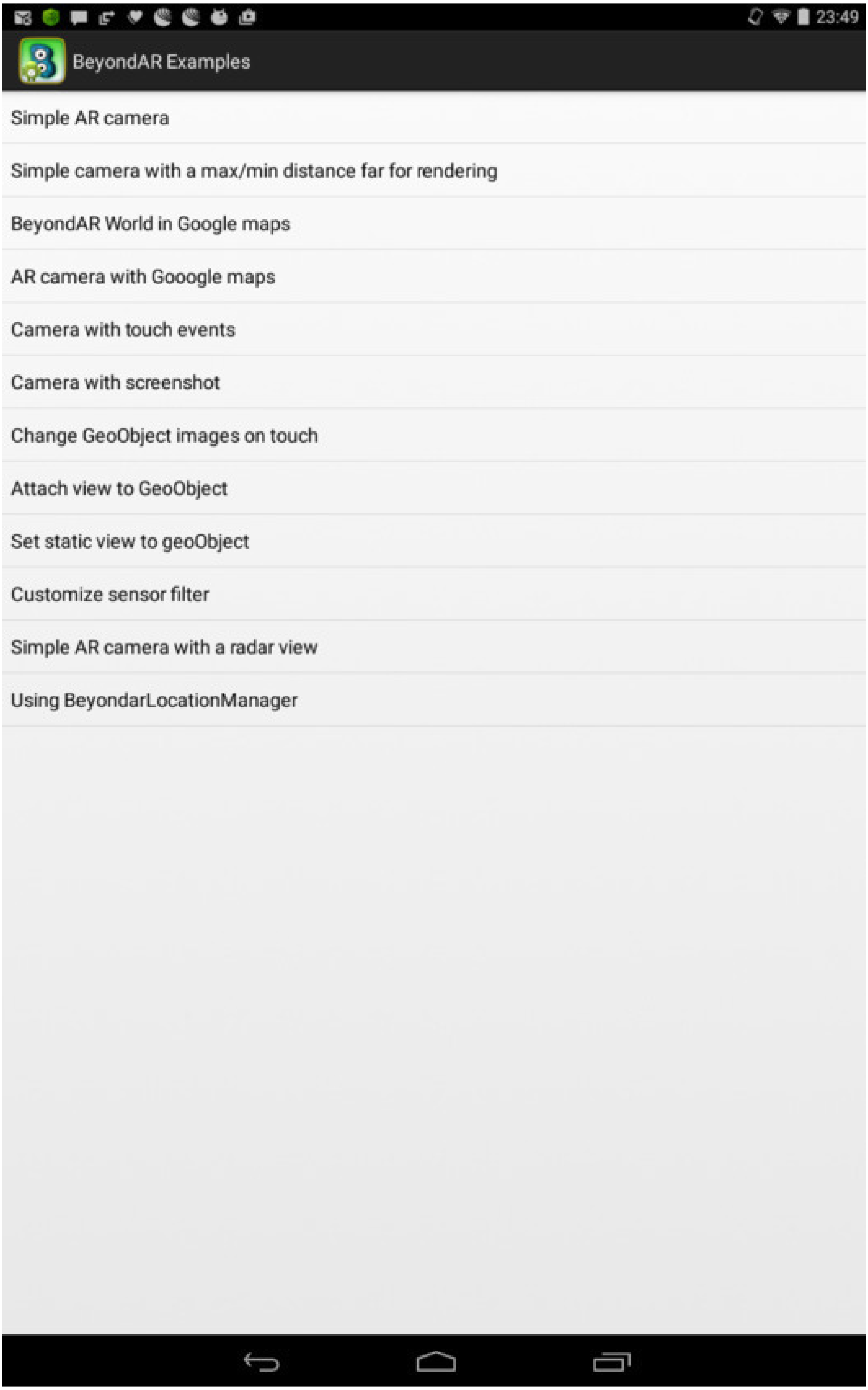
```
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.Marker;

public class BeyondarLocationManagerMapActivity extends FragmentActivity implements OnMarkerClickListener, OnClickListener, OnMapReadyCallback {

private GoogleMap mMap;
private GoogleMapWorldPlugin mGoogleMapPlugin;
private World mWorld;

@Override
protected void onCreate (Bundle savedInstanceState) {
super. onCreate (savedInstanceState);
setContentView(R.layout.map_google);

Button myLocationButton = (Button) findViewById(R.id.myLocationButton);

myLocationButton.setVisibility(View.VISIBLE);
myLocationButton.setOnClickListener (this);

((SupportMapFragment) getSupportFragmentManager
() .findFragmentById(R.id.map)).getMapAsync (this);
}

@Override
public boolean onMarkerClick (Marker marker) {
// To get the GeoObject that owns the marker we use the following
// method:
GeoObject geoObject = mGoogleMapPlugin.getGeoObjectOwner (marker);
if (geoObject!= null) {
Toast.makeText (this, «Click on a marker owned by a GeoOject with the name: "
+ geoObject.getName (),
Toast.LENGTH_SHORT).show ();
}
return false;
}

@Override
protected void onResume () {
super. onResume ();
// When the activity is resumed it is time to enable the
// BeyondarLocationManager
BeyondarLocationManager. enable ();
}

@Override
protected void onPause () {
super. onPause ();
```

```
// To avoid unnecessary battery usage disable BeyondarLocationManager
// when the activity goes on pause.
BeyondarLocationManager. disable ();
}

@Override
public void onClick (View v) {
// When the user clicks on the button we animate the map to the user
// location
LatLng userLocation = new LatLng(mWorld.getLatitude (), mWorld.getLongitude ());
mMap.moveCamera(CameraUpdateFactory.newLatLngZoom (userLocation, 15));
mMap.animateCamera (CameraUpdateFactory. zoomTo (19), 2000, null);
}

@Override
public void onMapReady (GoogleMap googleMap) {
mMap=googleMap;
// We create the world and fill the world
mWorld = CustomWorldHelper.generateObjects (this);

// As we want to use GoogleMaps, we are going to create the plugin and
// attach it to the World
mGoogleMapPlugin = new GoogleMapWorldPlugin (this);
// Then we need to set the map in to the GoogleMapPlugin
mGoogleMapPlugin.setGoogleMap (mMap);
// Now that we have the plugin created let's add it to our world.
// NOTE: It is better to load the plugins before start adding object in
// to the world.
mWorld.addPlugin (mGoogleMapPlugin);
mMap.setOnMarkerClickListener (this);

mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(mGoogleMapPlugin.getLatLng
(), 15));
mMap.animateCamera (CameraUpdateFactory. zoomTo (19), 2000, null);

// Lets add the user position to the map
GeoObject user = new GeoObject (1000l);
user.setGeoPosition(mWorld.getLatitude (), mWorld.getLongitude ());
user.setImageResource (R. drawable. flag);
user.setName («User position»);
mWorld.addBeyondarObject (user);

BeyondarLocationManager.addWorldLocationUpdate (mWorld);
BeyondarLocationManager.addGeoObjectLocationUpdate (user);

// We need to set the LocationManager to the BeyondarLocationManager.
BeyondarLocationManager
.setLocationManager ((LocationManager) getSystemService (Context.
LOCATION_SERVICE));
```

```
}
}
package com.beyondar. example;

import android. os. Bundle;
import android.support.v4.app.FragmentActivity;
import android.widget.Toast;

import com.beyondar.android.plugin. googlemap. GoogleMapWorldPlugin;
import com.beyondar.android.world.GeoObject;
import com.beyondar.android. world. World;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps. GoogleMap;
import com.google.android.gms.maps. GoogleMap. OnMarkerClickListener;
import com.google.android.gms.maps. OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.Marker;

public class GoogleMapActivity extends FragmentActivity implements
OnMarkerClickListener, OnMapReadyCallback {

private GoogleMap mMap;
private GoogleMapWorldPlugin mGoogleMapPlugin;
private World mWorld;

@Override
protected void onCreate (Bundle savedInstanceState) {
super. onCreate (savedInstanceState);
setContentView(R.layout.map_google);

((SupportMapFragment) getSupportFragmentManager
() .findFragmentById(R.id.map)).getMapAsync (this);

}

@Override
public boolean onMarkerClick (Marker marker) {
// To get the GeoObject that owns the marker we use the following
// method:
GeoObject geoObject = mGoogleMapPlugin.getGeoObjectOwner (marker);
if (geoObject!= null) {
Toast.makeText (this, «Click on a marker owned by a GeoOject with the name: "
+ geoObject.getName (),
Toast.LENGTH_SHORT).show ();
}
return false;
}

@Override
```

```
public void onMapReady (GoogleMap googleMap) {
mMap=googleMap;
// We create the world and fill the world
mWorld = CustomWorldHelper.generateObjects (this);

// As we want to use GoogleMaps, we are going to create the plugin and
// attach it to the World
mGoogleMapPlugin = new GoogleMapWorldPlugin (this);
// Then we need to set the map in to the GoogleMapPlugin
mGoogleMapPlugin.setGoogleMap (mMap);
// Now that we have the plugin created let's add it to our world.
// NOTE: It is better to load the plugins before start adding object in to the world.
mWorld.addPlugin (mGoogleMapPlugin);

mMap.setOnMarkerClickListener (this);

mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(mGoogleMapPlugin.getLatLng
(), 15));
mMap.animateCamera (CameraUpdateFactory. zoomTo (19), 2000, null);

// Lets add the user position
GeoObject user = new GeoObject (1000l);
user.setGeoPosition(mWorld.getLatitude (), mWorld.getLongitude ());
user.setImageResource (R. drawable. flag);
user.setName («User position»);
mWorld.addBeyondarObject (user);
}
}
```

После запуска приложения на Android устройстве появится список с примерами.


23:49
BeyondAR Examples
Simple AR camera
Simple camera with a max/min distance far for rendering
BeyondAR World in Google maps
AR camera with Gooogle maps
Camera with touch events
Camera with screenshot
Change GeoObject images on touch
Attach view to GeoObject
Set static view to geoObject
Customize sensor filter
Simple AR camera with a radar view
Using BeyondarLocationManager

Simple AR camera – показывает набор изображений на фоне камеры. При этом изображения расположены в пространстве вокруг устройства.

Simple camera with a max/min distance far for rendering – показывает набор изображений на фоне камеры с возможностью регулировки расстояния до изображений.

BeyondAR World in Google maps – показывает набор изображений на карте.

AR camera with Google maps – показывает набор изображений на фоне камеры с кнопкой переключения на карту.

Camera with touch events – показывает набор изображений на фоне камеры, а также сообщение при нажатии на одном из изображений.

Camera with screenshot – показывает набор изображений на фоне камеры с кнопкой скриншота.

Change GeoObject images on touch – показывает набор изображений на фоне камеры, которые заменяются на другие изображения при нажатии.

Attach view to GeoObject – показывает набор изображений на фоне камеры с добавлением вида к изображению при нажатии.

Set static view to geoObject – вместо изображений показывает виды на фоне камеры, а также сообщение при нажатии на одном из видов.

Customize sensor filter – показывает набор изображений на фоне камеры с возможностью регулировки чувствительности датчика ориентации.

Simple AR camera with a radar view – показывает набор изображений на фоне камеры, а также расположение изображений вокруг устройства.

Using BeyondarLocationManager – показывает набор изображений на карте с кнопкой обновления местоположения.

Для работы BeyondAR фреймворка в файле манифеста приложения декларируются необходимые разрешения и наличие сенсоров устройства.

```
<! – Minimum permissions for Beyondar – >
<uses-permission android:name="android.permission.CAMERA» />

<! – For Beyondar this is not mandatory unless you want to load something from Internet (for instance images) – >
<uses-permission android:name="android.permission.INTERNET» />

<! – BeyondAR needs the following features – >
<uses-feature android:name="android.hardware.camera» />
<uses-feature android:name="android.hardware.sensor.accelerometer» />
<uses-feature android:name="android.hardware.sensor.compass» />
```

Активность SimpleCameraActivity, отображающая набор изображений на фоне камеры, имеет достаточно простой код.

```
package com.beyondar. example;

import android. os. Bundle;
import android.support.v4.app.FragmentActivity;
import android.view. Window;

import com.beyondar.android.fragment.BeyondarFragmentSupport;
import com.beyondar.android. world. World;

public class SimpleCameraActivity extends FragmentActivity {
```

```
private BeyondarFragmentSupport mBeyondarFragment;
private World mWorld;

/** Called when the activity is first created. */
@Override
public void onCreate (Bundle savedInstanceState) {
super. onCreate (savedInstanceState);

// Hide the window title.
requestWindowFeature (Window. FEATURE_NO_TITLE);

setContentView(R.layout.simple_camera);

mBeyondarFragment = (BeyondarFragmentSupport)
getSupportFragmentManager().findFragmentById(R.id.beyondarFragment);

// We create the world and fill it…
mWorld = CustomWorldHelper.generateObjects (this);
// … and send it to the fragment
mBeyondarFragment.setWorld (mWorld);

// We also can see the Frames per seconds
mBeyondarFragment.showFPS (true);
}

}
```

В методе onCreate создается фрагмент BeyondarFragmentSupport, отвечающий за отображение вида камеры и вида BeyondarGLSurfaceView, рисующего дополненную реальность.

Для этого используется файл компоновки.

```
<?xml version=«1.0» encoding=«utf-8»? >
<FrameLayout xmlns: android="http://schemas.android.com/apk/res/android"
android: layout_width=«match_parent»
android: layout_height=«match_parent»
android: id="@+id/parentFrameLayout»>

<fragment
android: id="@+id/beyondarFragment»
android:name="com.beyondar.android.fragment.BeyondarFragmentSupport»
android: layout_width=«match_parent»
android: layout_height=«match_parent» />

</FrameLayout>
```

Далее создается объект World – контейнер объектов дополненной реальности, который затем добавляется во фрагмент BeyondarFragmentSupport.

Метод mBeyondarFragment.showFPS (true) показывает количество кадров в секунду в левом верхнем углу экрана.

Вся магия по созданию объектов дополненной реальности осуществляется в классе CustomWorldHelper.

Здесь создается новый контейнер World, устанавливается его местоположение в реальном мире, а также на основе изображений создаются объекты GeoObject, которые добавляются в контейнер World.

```
public static World sharedWorld;

sharedWorld = new World (context);
sharedWorld.setGeoPosition (41.90533734214473d, 2.565848038959814d);

GeoObject go4 = new GeoObject (4l);
go4.setGeoPosition (41.90518862002349d, 2.565662767707665d);
go4.setImageUri("assets://creature_7.png»);
go4.setName («Image from assets»);

sharedWorld.addBeyondarObject (go4);
```

По умолчанию для контейнера World и для его объектов, в классе CustomWorldHelper, задаются фиксированные координаты в реальном мире. Исправим это, привязав координаты контейнера World к местоположению устройства.

Для определения местоположения устройства используем Fused location provider API (Android API Level> v9, Android Build Tools> v21).

Изменим код классов CustomWorldHelper, GoogleMapActivity и SimpleCameraActivity.

```
import android.annotation.SuppressLint;
import android.content.Context;
import android. location. Location;
import android.widget.Toast;

import com.beyondar.android.world.GeoObject;
import com.beyondar.android. world. World;

@SuppressLint («SdCardPath»)
public class CustomWorldHelper {
public static final int LIST_TYPE_EXAMPLE_1 = 1;

public static World sharedWorld;

public static World generateObjects (Context context, Location mCurrentLocation) {
sharedWorld = new World (context);
// The user can set the default bitmap. This is useful if you are
// loading images form Internet and the connection get lost

sharedWorld.setDefaultImage(R.drawable.beyondar_default_unknow_icon);
// User position (you can change it using the GPS listeners form Android
// API)
if (mCurrentLocation== null) {
mCurrentLocation=new Location (»»);
mCurrentLocation.setLatitude (41.90533734214473d);
mCurrentLocation.setLongitude (2.565848038959814d);
}
```

```
sharedWorld.setGeoPosition(mCurrentLocation.getLatitude(),mCurrentLocation.getLongitude ());

// Create an object with an image in the app resources.
// And the same goes for the app assets
GeoObject go = new GeoObject (1l);
go.setGeoPosition(mCurrentLocation.getLatitude()+0.00005,mCurrentLocation.getLongitude
() -0.0001);
go.setImageUri("assets://creature_7.png»);
go.setName («Image from assets»);

// Add the GeoObjects to the world
sharedWorld.addBeyondarObject (go);

return sharedWorld;
}

}
import android.content.pm.PackageManager;
import android. location. Location;
import android. os. Bundle;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.v4.app.ActivityCompat;
import android.support.v4.app.FragmentActivity;
import android.widget.Toast;

import com.beyondar.android.plugin. googlemap. GoogleMapWorldPlugin;
import com.beyondar.android.world.GeoObject;
import com.beyondar.android. world. World;
import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common. api. GoogleApiClient;
import com.google.android.gms. location. LocationListener;
import com.google.android.gms. location. LocationRequest;
import com.google.android.gms. location. LocationServices;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps. GoogleMap;
import com.google.android.gms.maps. GoogleMap. OnMarkerClickListener;
import com.google.android.gms.maps. OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.Marker;

public class GoogleMapActivity extends FragmentActivity implements
OnMarkerClickListener, OnMapReadyCallback, LocationListener,
GoogleApiClient.ConnectionCallbacks, GoogleApiClient. OnConnectionFailedListener {

private GoogleMap mMap;
private GoogleMapWorldPlugin mGoogleMapPlugin;
private World mWorld;
GoogleApiClient mGoogleApiClient;
```

```
Location mCurrentLocation;
LocationRequest mLocationRequest;

@Override
protected void onCreate (Bundle savedInstanceState) {
super. onCreate (savedInstanceState);
setContentView(R.layout.map_google);

((SupportMapFragment) getSupportFragmentManager
() .findFragmentById(R.id.map)).getMapAsync (this);

buildGoogleApiClient ();
}

/**
* Builds a GoogleApiClient. Uses the {@code #addApi} method to request the
* LocationServices API.
*/
protected synchronized void buildGoogleApiClient () {
mGoogleApiClient = new GoogleApiClient. Builder (this)
.addConnectionCallbacks (this)
.addOnConnectionFailedListener (this)
.addApi (LocationServices. API)
.build ();
createLocationRequest ();
}

protected void createLocationRequest () {
mLocationRequest = LocationRequest.create ();

// Sets the desired interval for active location updates. This interval is
// inexact. You may not receive updates at all if no location sources are available, or
// you may receive them slower than requested. You may also receive updates faster than
// requested if other applications are requesting location at a faster interval.
mLocationRequest.setInterval (10000);

// Sets the fastest rate for active location updates. This interval is exact, and your
// application will never receive updates faster than this value.
mLocationRequest.setFastestInterval (5000);

mLocationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
}

@Override
public void onStart () {
super. onStart ();
mGoogleApiClient.connect ();
}
```

```
@Override
public void onStop () {
super. onStop ();
mGoogleApiClient. disconnect ();
}

@Override
public void onResume () {
super. onResume ();

// Within {@code onPause ()}, we pause location updates, but leave the
// connection to GoogleApiClient intact. Here, we resume receiving
// location updates if the user has requested them.

if (mGoogleApiClient.isConnected ()) {
startLocationUpdates ();
}
}

@Override
protected void onPause () {
super. onPause ();
// Stop location updates to save battery, but don't disconnect the GoogleApiClient object.
if (mGoogleApiClient.isConnected ()) {
stopLocationUpdates ();
}
}

protected void startLocationUpdates () {

if (ActivityCompat.checkSelfPermission (this,
android.Manifest.permission.ACCESS_FINE_LOCATION)!=
PackageManager.PERMISSION_GRANTED && ActivityCompat.checkSelfPermission (this,
android.Manifest.permission.ACCESS_COARSE_LOCATION)!=
PackageManager.PERMISSION_GRANTED) {

return;
}
LocationServices.FusedLocationApi.requestLocationUpdates (
mGoogleApiClient, mLocationRequest, this);
}

/**
* Removes location updates from the FusedLocationApi.
*/
protected void stopLocationUpdates () {
// It is a good practice to remove location requests when the activity is in a paused or
// stopped state. Doing so helps battery performance and is especially
// recommended in applications that request frequent location updates.
```

```
// The final argument to {@code requestLocationUpdates ()} is a LocationListener
// (http://developer.android.com/reference/com/google/android/gms/location/
LocationListener.html).

LocationServices.FusedLocationApi.removeLocationUpdates (mGoogleApiClient, this);
}

@Override
public boolean onMarkerClick (Marker marker) {
// To get the GeoObject that owns the marker we use the following
// method:
GeoObject geoObject = mGoogleMapPlugin.getGeoObjectOwner (marker);
if (geoObject!= null) {
Toast.makeText (this, «Click on a marker owned by a GeoOject with the name: "
+ geoObject.getName (),
Toast.LENGTH_SHORT).show ();
}
return false;
}

@Override
public void onMapReady (GoogleMap googleMap) {
mMap=googleMap;
// We create the world and fill the world
mWorld = CustomWorldHelper.generateObjects (this, mCurrentLocation);

// As we want to use GoogleMaps, we are going to create the plugin and
// attach it to the World
mGoogleMapPlugin = new GoogleMapWorldPlugin (this);
// Then we need to set the map in to the GoogleMapPlugin
mGoogleMapPlugin.setGoogleMap (mMap);
// Now that we have the plugin created let's add it to our world.
// NOTE: It is better to load the plugins before start adding object in to the world.
mWorld.addPlugin (mGoogleMapPlugin);

mMap.setOnMarkerClickListener (this);

mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(mGoogleMapPlugin.getLatLng
(), 15));
mMap.animateCamera (CameraUpdateFactory. zoomTo (19), 2000, null);

// Lets add the user position
GeoObject user = new GeoObject (1000l);
user.setGeoPosition(mWorld.getLatitude (), mWorld.getLongitude ());
user.setImageResource (R. drawable. flag);
user.setName («User position»);
mWorld.addBeyondarObject (user);
}
```

```
@Override
public void onConnected (@Nullable Bundle bundle) {

if (ActivityCompat.checkSelfPermission (this,
android.Manifest.permission.ACCESS_FINE_LOCATION)!=
PackageManager.PERMISSION_GRANTED && ActivityCompat.checkSelfPermission (this,
android.Manifest.permission.ACCESS_COARSE_LOCATION)!=
PackageManager.PERMISSION_GRANTED) {

return;
}
Location mLastLocation = LocationServices.FusedLocationApi.getLastLocation
(mGoogleApiClient);
if (mLastLocation!= null) {
mCurrentLocation = mLastLocation;
String lat = String.valueOf(mCurrentLocation.getLatitude ());
String lon = String.valueOf(mCurrentLocation.getLongitude ());
Toast toast = Toast.makeText (this, «Last location» + lat + " " + lon, Toast. LENGTH_LONG);
toast.show ();

mWorld.clearWorld ();
mMap.clear ();
mWorld = CustomWorldHelper.generateObjects (this, mCurrentLocation);
mGoogleMapPlugin = new GoogleMapWorldPlugin (this);
mGoogleMapPlugin.setGoogleMap (mMap);
mWorld.addPlugin (mGoogleMapPlugin);
mMap.setOnMarkerClickListener (this);

mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(mGoogleMapPlugin.getLatLng
(), 15));
mMap.animateCamera (CameraUpdateFactory. zoomTo (19), 2000, null);
GeoObject user = new GeoObject (1000l);
user.setGeoPosition(mWorld.getLatitude (), mWorld.getLongitude ());
user.setImageResource (R. drawable. flag);
user.setName («User position»);
mWorld.addBeyondarObject (user);
} else {
startLocationUpdates ();
}
}

@Override
public void onConnectionSuspended (int i) {

}

@Override
public void onConnectionFailed (@NonNull ConnectionResult connectionResult) {
```

```
}

@Override
public void onLocationChanged (Location location) {
mCurrentLocation = location;
String lat = String.valueOf(mCurrentLocation.getLatitude ());
String lon = String.valueOf(mCurrentLocation.getLongitude ());
Toast toast = Toast.makeText (this,«Current location " + lat+" "+lon, Toast.
LENGTH_LONG);
toast.show ();
mWorld.clearWorld ();
mMap.clear ();
mWorld = CustomWorldHelper.generateObjects (this, mCurrentLocation);
mGoogleMapPlugin = new GoogleMapWorldPlugin (this);
mGoogleMapPlugin.setGoogleMap (mMap);
mWorld.addPlugin (mGoogleMapPlugin);
mMap.setOnMarkerClickListener (this);

mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(mGoogleMapPlugin.getLatLng
(), 15));
mMap.animateCamera (CameraUpdateFactory. zoomTo (19), 2000, null);
GeoObject user = new GeoObject (1000l);
user.setGeoPosition(mWorld.getLatitude (), mWorld.getLongitude ());
user.setImageResource (R. drawable. flag);
user.setName («User position»);
mWorld.addBeyondarObject (user);
}
}
import android.content.pm.PackageManager;
import android. location. Location;
import android. os. Bundle;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.v4.app.ActivityCompat;
import android.support.v4.app.FragmentActivity;
import android.view. Window;
import android.widget.Toast;

import com.beyondar.android.fragment.BeyondarFragmentSupport;
import com.beyondar.android. opengl. util. LowPassFilter;
import com.beyondar.android. world. World;
import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common. api. GoogleApiClient;
import com.google.android.gms. location. LocationListener;
import com.google.android.gms. location. LocationRequest;
import com.google.android.gms. location. LocationServices;
```

```
public class SimpleCameraActivity extends FragmentActivity implements LocationListener, GoogleApiClient.ConnectionCallbacks, GoogleApiClient. OnConnectionFailedListener {

private BeyondarFragmentSupport mBeyondarFragment;
private World mWorld;
GoogleApiClient mGoogleApiClient;
Location mCurrentLocation;
LocationRequest mLocationRequest;

/** Called when the activity is first created. */
@Override
public void onCreate (Bundle savedInstanceState) {
super. onCreate (savedInstanceState);
// Hide the window title.
requestWindowFeature (Window. FEATURE_NO_TITLE);

setContentView(R.layout.simple_camera);

mBeyondarFragment = (BeyondarFragmentSupport) getSupportFragmentManager () .findFragmentById(R.id.beyondarFragment);

// We also can see the Frames per seconds
mBeyondarFragment.showFPS (false);

// We create the world and fill it…
mWorld = CustomWorldHelper.generateObjects (this, mCurrentLocation);
// … and send it to the fragment
mBeyondarFragment.setWorld (mWorld);
LowPassFilter.ALPHA = 0.003f;

buildGoogleApiClient ();
}

/**
* Builds a GoogleApiClient. Uses the {@code #addApi} method to request the
* LocationServices API.
*/
protected synchronized void buildGoogleApiClient () {
mGoogleApiClient = new GoogleApiClient. Builder (this)
.addConnectionCallbacks (this)
.addOnConnectionFailedListener (this)
.addApi (LocationServices. API)
.build ();
createLocationRequest ();
}

protected void createLocationRequest () {
mLocationRequest = LocationRequest.create ();
```

```
// Sets the desired interval for active location updates. This interval is
// inexact. You may not receive updates at all if no location sources are available, or
// you may receive them slower than requested. You may also receive updates faster than
// requested if other applications are requesting location at a faster interval.
mLocationRequest.setInterval (10000);

// Sets the fastest rate for active location updates. This interval is exact, and your
// application will never receive updates faster than this value.
mLocationRequest.setFastestInterval (5000);

mLocationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
}

@Override
public void onStart () {
super. onStart ();
mGoogleApiClient.connect ();
}

@Override
public void onStop () {
super. onStop ();
mGoogleApiClient. disconnect ();
}

@Override
public void onResume () {
super. onResume ();
// Within {@code onPause ()}, we pause location updates, but leave the
// connection to GoogleApiClient intact. Here, we resume receiving
// location updates if the user has requested them.

if (mGoogleApiClient.isConnected ()) {
startLocationUpdates ();
}
}

@Override
protected void onPause () {
super. onPause ();
// Stop location updates to save battery, but don't disconnect the GoogleApiClient object.
if (mGoogleApiClient.isConnected ()) {
stopLocationUpdates ();
}
}

protected void startLocationUpdates () {
```

```
if (ActivityCompat.checkSelfPermission (this, android.Manifest.permission.ACCESS_FINE_LOCATION)!= PackageManager.PERMISSION_GRANTED && ActivityCompat.checkSelfPermission (this, android.Manifest.permission.ACCESS_COARSE_LOCATION)!= PackageManager.PERMISSION_GRANTED) {

return;
}
LocationServices.FusedLocationApi.requestLocationUpdates (
mGoogleApiClient, mLocationRequest, this);
}

/**
* Removes location updates from the FusedLocationApi.
*/
protected void stopLocationUpdates () {
// It is a good practice to remove location requests when the activity is in a paused or
// stopped state. Doing so helps battery performance and is especially
// recommended in applications that request frequent location updates.

// The final argument to {@code requestLocationUpdates ()} is a LocationListener
// (http://developer.android.com/reference/com/google/android/gms/location/LocationListener.html).
LocationServices.FusedLocationApi.removeLocationUpdates (mGoogleApiClient, this);
}

@Override
public void onConnected (@Nullable Bundle bundle) {

if (ActivityCompat.checkSelfPermission (this, android.Manifest.permission.ACCESS_FINE_LOCATION)!= PackageManager.PERMISSION_GRANTED && ActivityCompat.checkSelfPermission (this, android.Manifest.permission.ACCESS_COARSE_LOCATION)!= PackageManager.PERMISSION_GRANTED) {
// TODO: Consider calling
// ActivityCompat#requestPermissions
// here to request the missing permissions, and then overriding
// public void onRequestPermissionsResult (int requestCode, String [] permissions,
// int [] grantResults)
// to handle the case where the user grants the permission. See the documentation
// for ActivityCompat#requestPermissions for more details.
return;
}
Location mLastLocation = LocationServices.FusedLocationApi.getLastLocation (mGoogleApiClient);
if (mLastLocation!= null) {
mCurrentLocation = mLastLocation;
String lat = String.valueOf(mCurrentLocation.getLatitude ());
String lon = String.valueOf(mCurrentLocation.getLongitude ());
```

```
Toast toast = Toast.makeText (this, «Last location» + lat + " " + lon, Toast. LENGTH_LONG);
toast.show ();
mWorld.clearWorld ();
mWorld = CustomWorldHelper.generateObjects (this, mCurrentLocation);
mBeyondarFragment.setWorld (mWorld);
} else {
startLocationUpdates ();
}
}

@Override
public void onConnectionSuspended (int i) {
mGoogleApiClient.connect ();
}

@Override
public void onConnectionFailed (@NonNull ConnectionResult connectionResult) {

}

@Override
public void onLocationChanged (Location location) {
mCurrentLocation = location;
String lat = String.valueOf(mCurrentLocation.getLatitude ());
String lon = String.valueOf(mCurrentLocation.getLongitude ());
Toast toast = Toast.makeText (this,«Current location " + lat+" "+lon, Toast.
LENGTH_LONG);
toast.show ();
mWorld.clearWorld ();
mWorld = CustomWorldHelper.generateObjects (this, mCurrentLocation);
mBeyondarFragment.setWorld (mWorld);
}
}
```

Теперь дополненная реальность будет привязана к текущему местоположению пользователя.

В качестве примера использования фреймворка BeyondAR создадим игровое приложение Creatures in Camera, в котором пользователь сможет расставлять 2D объекты в реальном мире, а потом наблюдать их через камеру.

Создадим новый проект в Android Studio, используя шаблон Navigation Drawer Activity.

Для сборки APK файла с большим количеством методов в коде, в Gradle файл добавим:

```
defaultConfig {

multiDexEnabled true
}
dependencies {

compile 'com.android.support: multidex:1.0.0»
}
android {
```

dexOptions {
javaMaxHeapSize «4g»
}
}

В файл манифеста добавим:

<application

android:name="android.support.multidex.MultiDexApplication»>

Добавим зависимость от библиотек beyondar-googlemap-plugin-v0.9.0.jar, beyondar-radar-plugin-v0.9.1.jar и beyondar-v0.9.3.jar, скопировав соответствующие файлы в папку libs проекта.

Добавим зависимость от библиотеки Google Play Services.

compile 'com.google.android.gms: play-services:9.6.1»

Добавим необходимые разрешения в файл манифеста.

<! – Google maps stuff – >
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE» />
<uses-permission android:name="android.permission. WRITE_EXTERNAL_STORAGE» /
>
<uses-permission
android:name="com.google.android.providers.gsf.permission.READ_GSERVICES» />

<! – Minimum permissions for BeyondAR – >
<uses-permission android:name="android.permission.CAMERA» />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION» />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION» />

<! – For BeyondAR this is not mandatory unless you want to load something from the network – >
<uses-permission android:name="android.permission.INTERNET» />

<! – BeyondAR needs the following features – >
<uses-feature android:name="android.hardware.camera» />
<uses-feature android:name="android.hardware.sensor.accelerometer» />
<uses-feature android:name="android.hardware.sensor.compass» />

Для использования Google Map добавим Google API Key в файл манифеста. Для того получим ключ в Google Developers Console и добавим в тег <application> файла манифеста.

<meta-data
android:name="com.google.android.geo. API_KEY»
android: value=«AIzaSyBcRu9Vvb7…» />

Изменим файл компоновки content_main. xml.

<?xml version=«1.0» encoding=«utf-8»? >
<android.support.v4.widget.NestedScrollView
xmlns: android="http://schemas.android.com/apk/res/android"
xmlns: app="http://schemas.android.com/apk/res-auto"
xmlns: tools="http://schemas.android.com/tools"
android: layout_width=«match_parent»
android: layout_height=«match_parent»
android: paddingLeft="@dimen/activity_horizontal_margin»

```
android: paddingRight="@dimen/activity_horizontal_margin»
android: paddingTop="@dimen/activity_vertical_margin»
android: paddingBottom="@dimen/activity_vertical_margin»
android: fillViewport=«true»
android: layout_gravity=«fill_vertical»
app: layout_behavior="@string/appbar_scrolling_view_behavior»
tools:context=".MainActivity»
tools: showIn="@layout/app_bar_main»
android: id="@+id/content_main»
>

<RelativeLayout
android: layout_width=«match_parent»
android: layout_height=«match_parent»>

<fragment
android: id="@+id/beyondarFragment»
android:name="com.beyondar.android.fragment.BeyondarFragmentSupport»
android: layout_width=«match_parent»
android: layout_height=«match_parent» />

</RelativeLayout>

</android.support.v4.widget.NestedScrollView>
```

Изменим код класса главной активности.

```
package com.tmsoftstudio.aryourworld;

import android. app. Dialog;
import android.content. DialogInterface;
import android.content.Intent;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android. os. Bundle;;
import android.support.design. widget. FloatingActionButton;
import android.support. v4.app. DialogFragment;
import android.support.v4.widget.NestedScrollView;
import android.support.v7.app.AlertDialog;
import android.view.LayoutInflater;
import android.view.View;
import android.support.design.widget.NavigationView;
import android.support.v4.view.GravityCompat;
import android.support. v4.widget. DrawerLayout;
import android.support.v7.app.ActionBarDrawerToggle;
import android.support. v7.app. AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.MenuItem;
import android.Manifest;
import android.content.Context;
import android.content.SharedPreferences;
```

```
import android.content.pm.PackageManager;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorManager;
import android. location. Location;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.v4.app.ActivityCompat;
import android.view.ViewGroup;
import android.view.ViewTreeObserver;
import android.widget.ProgressBar;
import android. widget. RadioButton;
import android. widget. RadioGroup;
import android.widget.Toast;

import com.beyondar.android.fragment.BeyondarFragmentSupport;
import com.beyondar.android.plugin. radar. RadarView;
import com.beyondar.android.plugin. radar. RadarWorldPlugin;
import com.beyondar.android.sensor.BeyondarSensorListener;
import com.beyondar.android.sensor.BeyondarSensorManager;
import com.beyondar.android. world. World;
import com.beyondar.android. opengl. util. LowPassFilter;
import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common. api. GoogleApiClient;
import com.google.android.gms. location. LocationListener;
import com.google.android.gms. location. LocationRequest;
import com.google.android.gms. location. LocationServices;

import org. json. JSONArray;
import org. json. JSONObject;

import java.util.Iterator;
import java.util.LinkedHashSet;
import java.util.Set;

public class MainActivity extends AppCompatActivity
implements NavigationView. OnNavigationItemSelectedListener, BeyondarSensorListener,
LocationListener, GoogleApiClient.ConnectionCallbacks, GoogleApiClient.
OnConnectionFailedListener {

private BeyondarFragmentSupport mBeyondarFragment;
private World mWorld;
private RadarView mRadarView;
private RadarWorldPlugin mRadarPlugin;

private Location mCurrentLocation;
private Context context;

GoogleApiClient mGoogleApiClient;
```

```
LocationRequest mLocationRequest;

private float [] mLastAccelerometer = new float [3];
private float [] mLastMagnetometer = new float [3];
private float [] mR = new float [9];
private float [] mOrientation = new float [3];

private static boolean flagLocationUpdate=true;
private static SharedPreferences mSettings;
private Set <String> boLat=new LinkedHashSet ();
private Set <String> boLon=new LinkedHashSet ();

private static ProgressBar spinner;

@Override
protected void onCreate (Bundle savedInstanceState) {
super. onCreate (savedInstanceState);
setContentView(R.layout.activity_main);
Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
setSupportActionBar (toolbar);

spinner = (ProgressBar)findViewById(R.id.progressBar);
spinner.setVisibility (View. GONE);

DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
ActionBarDrawerToggle toggle = new ActionBarDrawerToggle (
this, drawer, toolbar, R.string.navigation_drawer_open, R.string.navigation_drawer_close);
drawer.addDrawerListener (toggle);
toggle.syncState ();

NavigationView navigationView = (NavigationView) findViewById(R.id.nav_view);
navigationView.setNavigationItemSelectedListener (this);

final NestedScrollView nestedScrollView =
(NestedScrollView)findViewById(R.id.content_main);
nestedScrollView.getViewTreeObserver().addOnGlobalLayoutListener (
new ViewTreeObserver. OnGlobalLayoutListener () {

@Override
public void onGlobalLayout () {
int height = nestedScrollView.getHeight ();
int width = nestedScrollView.getWidth ();
if (height> width) height=width;
if (width> height) width=height;
ViewGroup.LayoutParams params = nestedScrollView.getLayoutParams ();
params. width=width;
params. height=height;
nestedScrollView.setLayoutParams (params);
nestedScrollView.getViewTreeObserver().removeGlobalOnLayoutListener (this);
```

```
}
});

context = this;
mSettings = getSharedPreferences («APP_PREFERENCES», Context.MODE_PRIVATE);

if (!mSettings.contains («BOLAT»)) {
SharedPreferences. Editor editor = mSettings. edit ();
editor. putStringSet («BOLAT», boLat);
editor.commit ();
}

if(!mSettings.contains («BOLON»)) {
SharedPreferences. Editor editor = mSettings. edit ();
editor. putStringSet («BOLON», boLon);
editor.commit ();
}

if(!mSettings.contains («CREATURES»)) {
JSONArray creatures = new JSONArray ();
SharedPreferences. Editor editor = mSettings. edit ();
editor.putString("CREATURES",creatures.toString ());
editor.commit ();
}

if(!mSettings.contains («USERLON»)) {
SharedPreferences. Editor editor = mSettings. edit ();
editor. putString («USERLON», «82.9346»);
editor.commit ();
}
if(!mSettings.contains («USERLAT»)) {
SharedPreferences. Editor editor = mSettings. edit ();
editor. putString («USERLAT», «55.0415»);
editor.commit ();
}

checkPermissions ();

mBeyondarFragment = (BeyondarFragmentSupport)
getSupportFragmentManager().findFragmentById(R.id.beyondarFragment);

mRadarView = (RadarView) findViewById(R.id.radarView);
mRadarPlugin = new RadarWorldPlugin (this);
mRadarPlugin.setRadarView (mRadarView);
mRadarPlugin.setMaxDistance (100);

CustomWorldHelper.setActivity (this);
mWorld = CustomWorldHelper.generateObjects (this);
mWorld.addPlugin (mRadarPlugin);
```

```
mBeyondarFragment.setWorld (mWorld);

LowPassFilter.ALPHA = 0.001f;

BeyondarSensorManager.registerSensorListener (this);
mBeyondarFragment.setMaxDistanceToRender (10);

FloatingActionButton fabAdd = (FloatingActionButton) findViewById(R.id.fabAdd);
fabAdd.setOnClickListener (new View. OnClickListener () {
@Override
public void onClick (View view) {
SelectCreatureDialogFragment dialog = new SelectCreatureDialogFragment ();
dialog.show (getSupportFragmentManager (), «SelectCreatureDialogFragment»);
```

Конец ознакомительного фрагмента.