Microsoft Hololens и Unity

Краткий обзор устройства

В визор встроены два прозрачных экрана – по одному перед каждым глазом. Вперед и по сторонам смотрят несколько камер, а над ушами расположены два небольших динамика, звук из которых накладывается на звук окружающей среды.

Картинка стереоскопическая. Поэтому нужно думать про фокусное расстояние. Например, мы в самом начале выводили отладочный текст, используя фокусное расстояние 30 сантиметров. Глазам было больно. Клавиатуры и мыши нет, хотя их и можно подключить через Bluetooth. Основной способ управления устройством – курсор, который двигается вместе с поворотом головы (“gaze”), и жест, похожий на щипок, заменяющий обычный тап (“air tap”). Неплохо работает управление голосом. Хотя, пока доступен только английский язык.

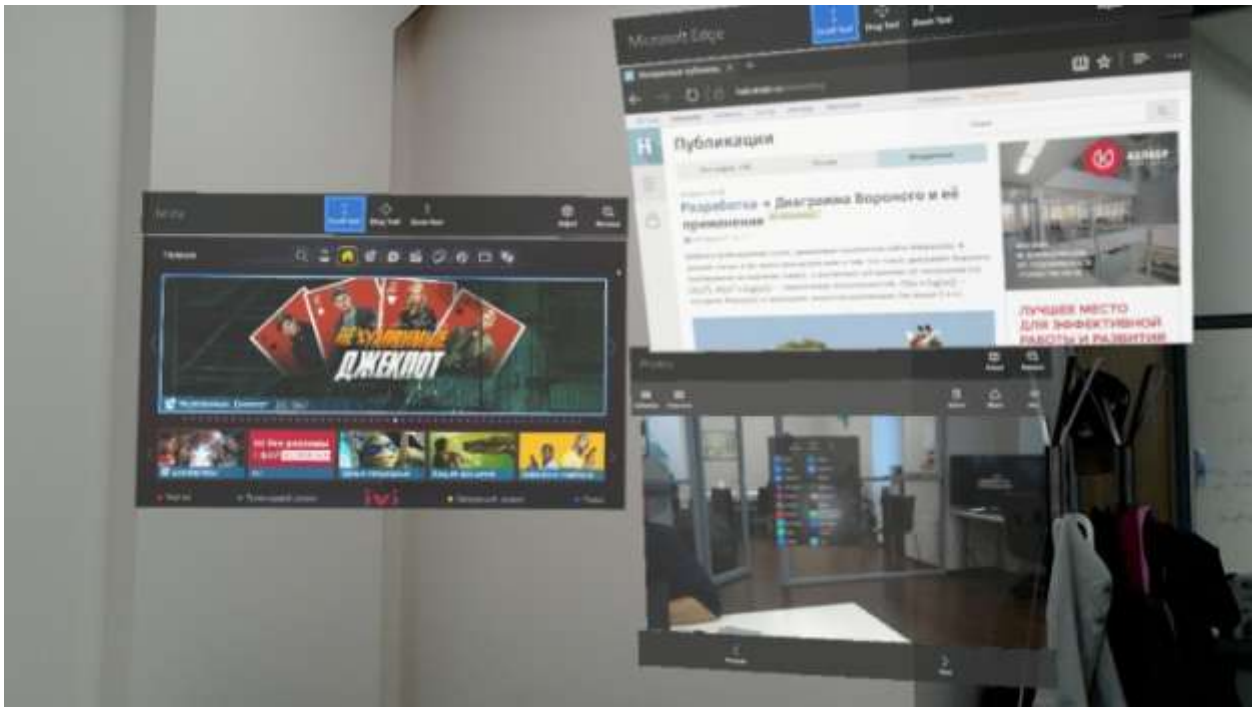
При помощи камер устройство сканирует окружающее пространство и превращает его в 3D модель, доступ к которой можно получить прямо в Unity (для разработки под устройство используется специальная версия) или через Windows API. Одно из самых частых применений этой возможности – поиск пола и установка на него голограмм.

В будущем подобные устройства заменят все, у чего есть экраны и звук. Представьте, что вы можете редактировать в реальном времени то, как вы видите и слышите реальный мир. Например, у соседей играет музыка, которая вам не нравится. Вы можете от нее избавиться или заменить на ту, которая нравится вам. Без помощи дрели можете повесить куда угодно сколько угодно постеров или маркерных досок. Одним жестом переставить (виртуальный) телевизор. Смартфоны, телевизоры, вывески, дорожные знаки, лампочки и дисплеи на бытовой технике – все это уйдет в прошлое, как только устройства, подобные HoloLens стану стоить столько, сколько стоит сейчас смартфон, а выглядеть будут как обычные очки. Это только самые очевидные примеры, мы уверены, что изменения будут более глобальными.

То есть эта технология, в отличие от чистого VR – следующая платформа для всего.

Разрабатывать под HoloLens имеет смысл начинать прямо сейчас, потому что уже есть платежеспособный спрос со стороны банков, нефтянки, образования, машиностроения, архитекторов. И у вас есть шанс стать первыми. Устройства становятся доступными гораздо быстрее, чем мы ожидали. Еще пару месяцев назад они были редкостью, а сейчас каждый может заказать пять штук в американском Microsoft Store.

Поскольку HoloLens работает на Windows 10, приложения для него разрабатываются на платформе Universal Windows Platform (UWP). Это означает что приложения, разработанные ранее для мобильных устройств, десктопа и Xbox на базе UWP можно запустить и на HoloLens.



Типы HoloLens приложений

Набор инструментов разработчика сильно зависит от типа приложения которые вы хотите сделать. На данный момент приложения для HoloLens поддерживает 2 вида представлений (views): 2D views и Holographic views.

2D view – это отображение привычного нам 2D контента в виде плашки внутри shell HoloLens. (см. скриншот выше) Одновременно множество приложений могут рендериться в 2D view, поэтому организовать рабочее пространство вокруг себя с одновременно запущенной почтой, бразуером и skype не составит труда.

Holographic View — в этом режиме приложение может создавать вокруг вас голограммы. При этом голограммы других приложений и shell ОС не видны, а любые системные нотификации будут произноситься голосом с помощью Cortana.

Между этими двумя типами можно переключаться. Один из частых сценариев совмещения нескольких view в одном приложений – переход из Holographic View в 2D view для ввода текста в поле ввода с помощью экранной клавиатуры.

Для разработки 2D views основным инструментом будет Visual Studio и UWP API с XAML и/или DirectX. Для разработки Holographic Views предлагается использовать 2 подхода – разработка с использованием Visual Studio и DirectX или Unity3D.
В рамках данной статьи мы рассмотрим создание приложения с одним Holographic View в Unity3D.

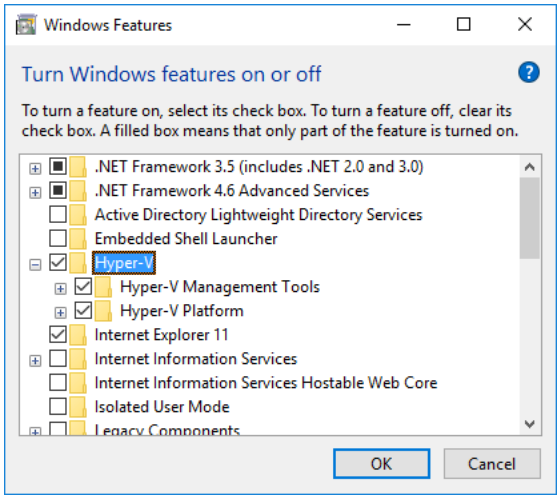
Шаг 0: Устанавливаем инструменты

Системные требования

- Если у вас еще нет устройства, то для запуска эмулятора потребуется Windows 10 64-bit с поддержкой Hyper-V: Windows 10 Pro, Enterprise или Education. Иначе хватит и Win 10 Home
- 64-bit CPU с 4 и более ядрами и поддержкой виртуализации
- 8 Gb RAM или более (Если вы планируете разрабатывать сетевые приложения то 12Gb и более, т.к. для запуска нескольких эмуляторов HoloLens потребуется порядка 10-11Gb RAM
- Поддержка в BIOS следующих функций:
  1. Hardware-assisted virtualization
  2. Second Level Address Translation (SLAT)
  3. Hardware-based Data Execution Prevention (DEP)

- GPU (Эмулятор может работать с неподдерживаемым GPU, но будет работать значительно медленнее)

Если ваш ПК соответствует этим требованиям и вы планируете для тестирования использовать HoloLens Emulator, вам потребуется активировать в BIOS вышеуказанные настройки.
После этого в панели управления Windows необходимо включить поддержку Hyper-V. Для этого перейдите в Control Panel -> Programs -> Programs and Features -> Turn Windows Features on or off. Найдите и выберите секцию Hyper-V. После завершения установки перезагрузитесь.



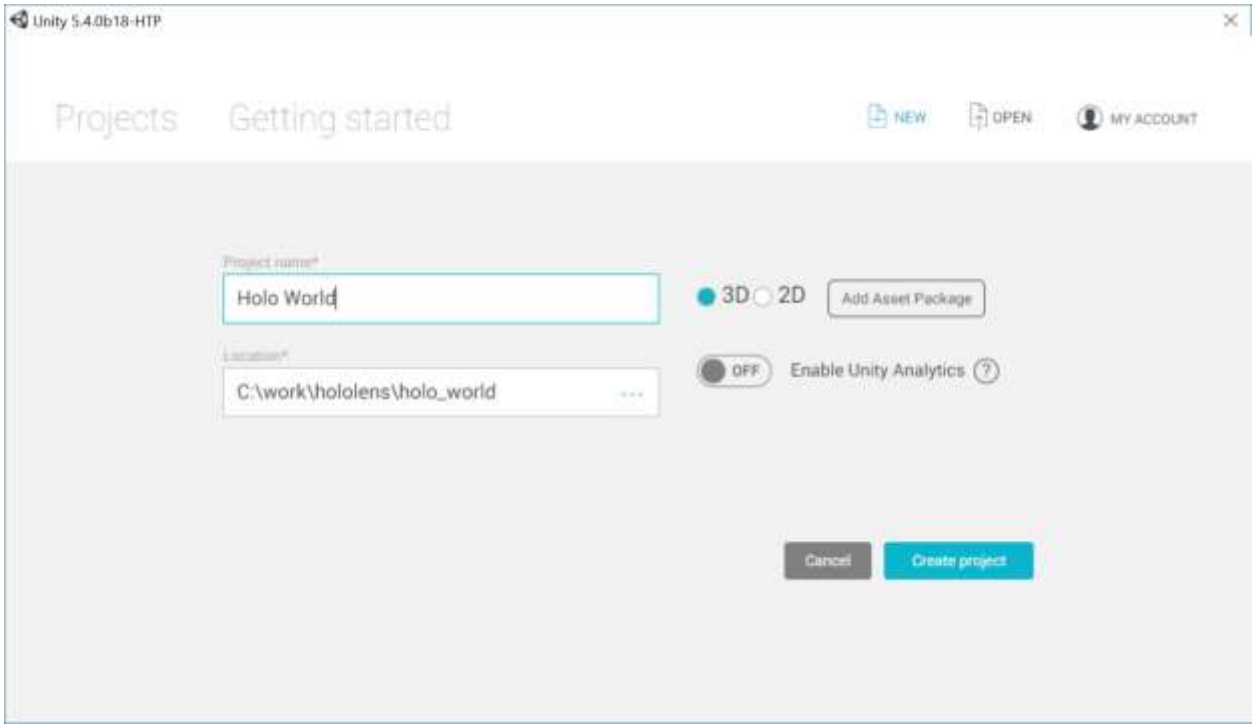
Инструменты

- Windows 10
- Visual Studio 2015 Update 3 (Community Edition)
- Windows 10 SDK v. 1511 или выше(Если вы его не установили вместе с Visual Studio 2015)
- HoloLens Emulator или HoloLens Development Edition
- Unity 3D HoloLens Technical Preview

Во избежание ошибок во время сборки приложения, инструменты рекомендуется устанавливать последовательно, в указанном порядке.

Шаг 1: Создание проекта

Теперь у нас есть все необходимое для работы, давайте начнем делать наше hello world приложение. Запустите Unity Hololens Technical Preview (HTP) и создайте пустой проект. Выберите имя и расположение на диске. Для типа проекта выберите 3D и нажмите *Create project*.

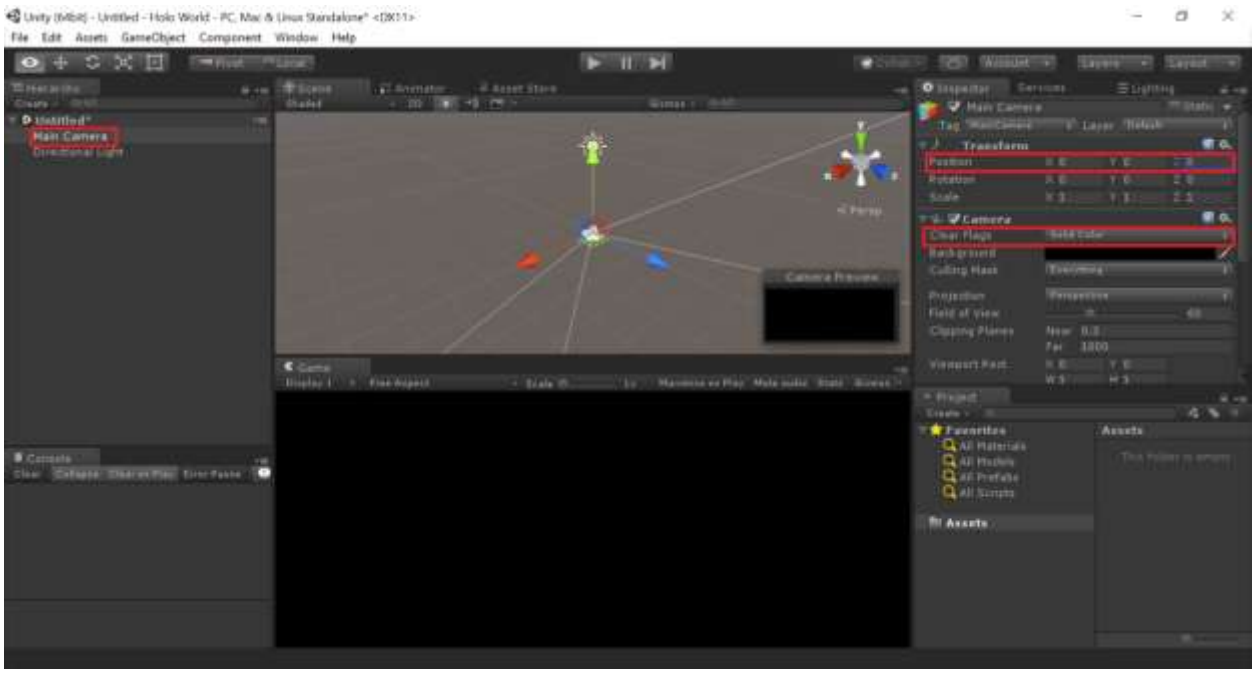


Настройка главной камеры

После создания проекта Unity создаст и откроет автоматически пустую сцену с камерой и источником освещения. Обратите внимание на следующие панели:

1. *Scene* – элементы, которые мы помещаем в 3D мир. Сейчас это только источник света и камера.
2. *Game* просмотр приложения в режиме игры
3. *B Hierarchy* находятся все элементы вашей сцены
4. *Inspector* показывает свойства выбранного элемента
5. *Project* все файлы рерурсов, добавленные для текущего проекта.

Давайте начнем с настройки камеры. В режиме Holographic View главная камера – это позиция между глаз человека. Поэтому сбрасываем смещение по умолчанию (0,0,0). Позиция и поворот камеры во время работы приложения в эмуляторе или на устройстве будут меняться вместе с положением и наклоном головы пользователя.



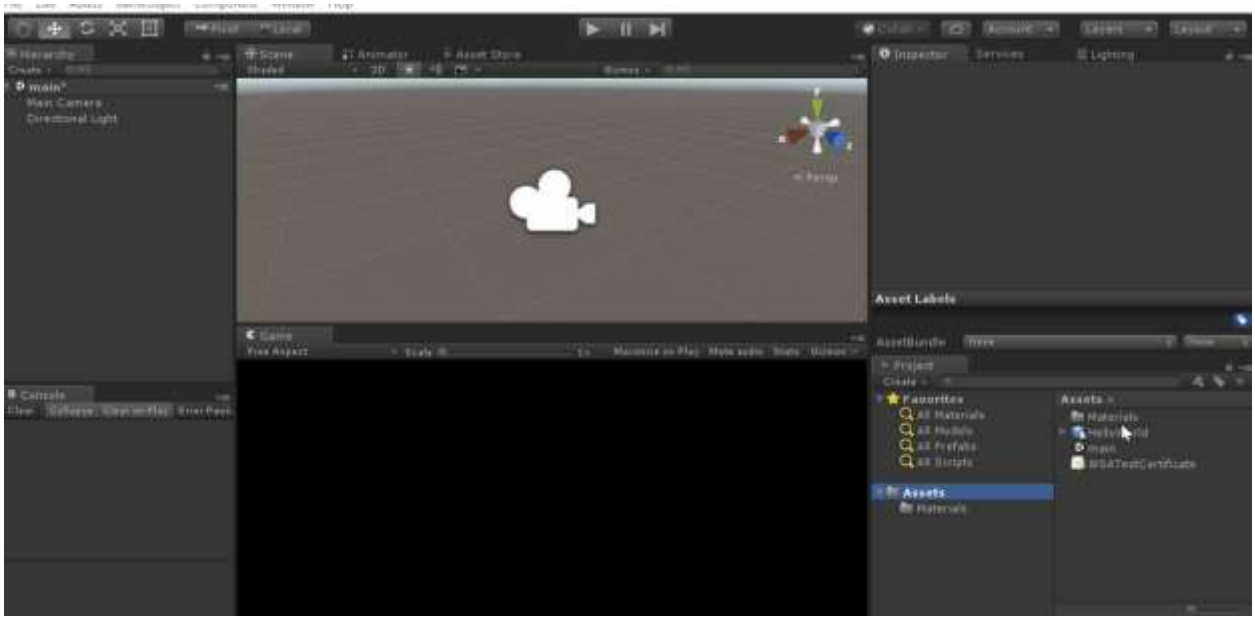
Вы можете настроить расположение вкладок как вам удобно.

HoloLens позволяет проецировать 3D объекты в реальном мире. Для этого у камеры должен быть прозрачный фон. Чтобы его задать установите значение *Clear Flags* камеры в *Solid Color*. После установите *Background* в [0,0,0,0] или #00000000. Для более точной имитации устройства в Unity установите *Field of View* камеры в 16-20.

Шаг 2: Добавим в сцену 3d объект

Мы подготовили 3D модель Hello world. Загрузите и добавьте .fbx файл модели в папку *Assets* проекта. Или перетащите файл в область *Assets* панели *Project*. После успешного импорта модели добавьте ее в сцену.

Для этого перетащите модель HelloWorld из панели Project в окно Hierarchy. Установите позицию по Z=3 и разверните надпись на 180, чтобы она смотрела в камеру по умолчанию. Исходная модель довольно большая, давайте ее немного уменьшим установив *Scale* в 0.25 и сохраните сцену с любым именем.



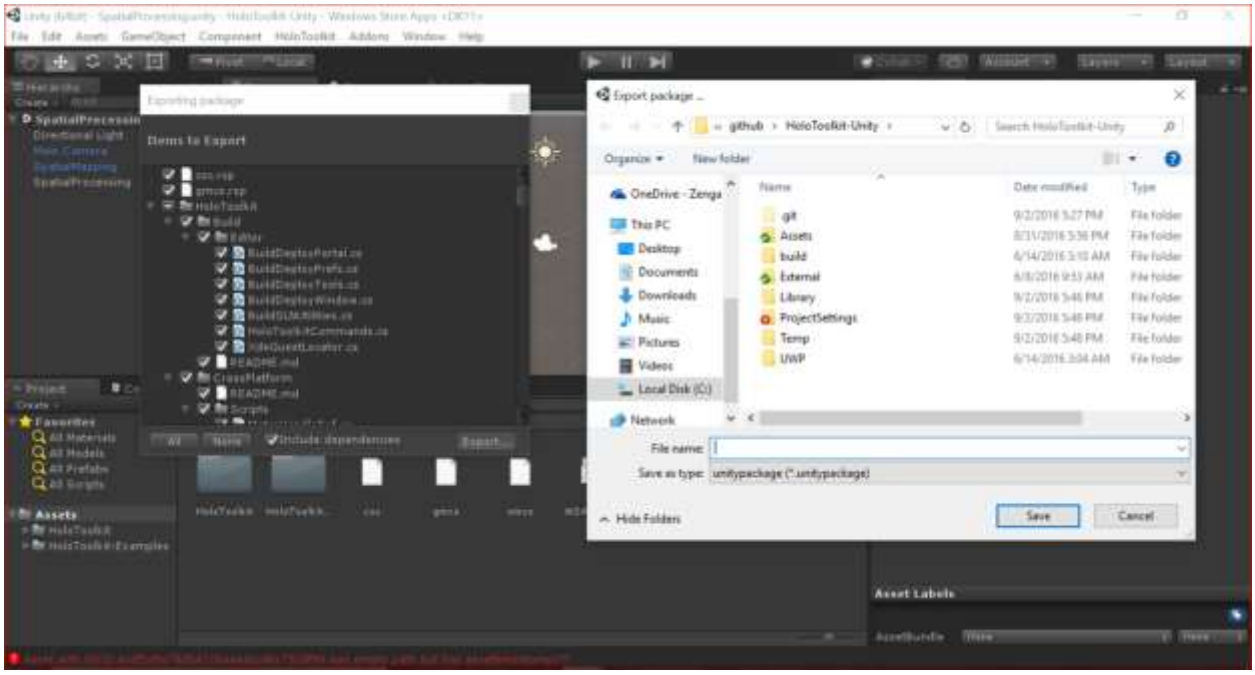
Знакомство с HoloToolkit

Компания Microsoft вместе с сообществом разработали набор компонентов для ускорения разработки приложений на HoloLens.

На текущий момент в репозитории на github содержатся компоненты и примеры работы с API:

- Ввод с помощью жестов, генерация речи
- Shared experience – инструменты для реализации совместного опыта в приложении
- Определение и обработка поверхностей в пространстве
- Пространственное звучание
- Ускорение процесса сборки приложения
- Готовые компоненты и скрипты для курсора, управления камерой и другие

Давайте импортируем HoloToolkit в наш проект и рассмотрим некоторые из компонентов. Для этого склонируйте репозиторий HoloToolkit к себе на диск и откройте его в Unity. В панели *Project* в контекстном меню корневой папки *Assets* выберите *Export Package*. Дождитесь загрузки списка файлов для экспорта, нажмите *Export* и укажите путь для сохранения пакета.



После окончания экспорта импортируйте этот пакет в вашей проект, вызвав контекстное меню на корневой папке *Assets*. Выберите *Import Package -> Custom Package*.

После окончания в проект добавятся ассеты из HoloToolkit. В меню Unity появится новая группа меню, позволяющая быстро применять общие настройки для проекта и делать быстрый деплой. В рамках статьи мы не будем пользоваться этими опциями с целью обучения.

Подробную настройку по импорту HoloToolkit в проект можно найти в вики проекта на github: HoloToolkit Getting Started

Добавляем компоненты для отладки в Unity

Для начала давайте добавим управление камерой в редакторе Unity. Выберите в панели *Hierarchy* объект камеры. В панели *Inspector* нажмите *Add Component* и в открывшемся окне введите *Manual*. Выберите в списке компонент *Manual Camera Control*. Теперь нажмите Ctrl+P или кнопку Play на панели вверху. Приложение запустится и по нему можно будет перемещаться с помощью WASD и смотреть вокруг с помощью Shift.

В настройках скрипта *Manual Camera Control* можно настроить клавиши управления, чувствительность мыши, оси перемещение и пр. на свой вкус.

Отображение комнаты в Unity и эмуляторе

Если вы разрабатываете приложение под HoloLens вы обязательно столкнетесь с необходимостью размещать ваши голограммы с привязкой к окружающему пространству.

Примеры привести легко:

- Практически любое приложение под HoloLens требует понимания уровня пола в помещении где оно запущено для корректного размещения голограмм.
- Возможно вы пишете action/strategy игру возможно вы захотите чтобы монстры/роботы/корабли появлялись из стен вам нужно понимать где находятся стены.

Для этого у HoloLens есть специальное API для получения поверхностей в реальном времени. В Unity и HoloToolkit для работы с ним есть несколько готовых компонентов, которых достаточно для реализации большинства таких кейсов.

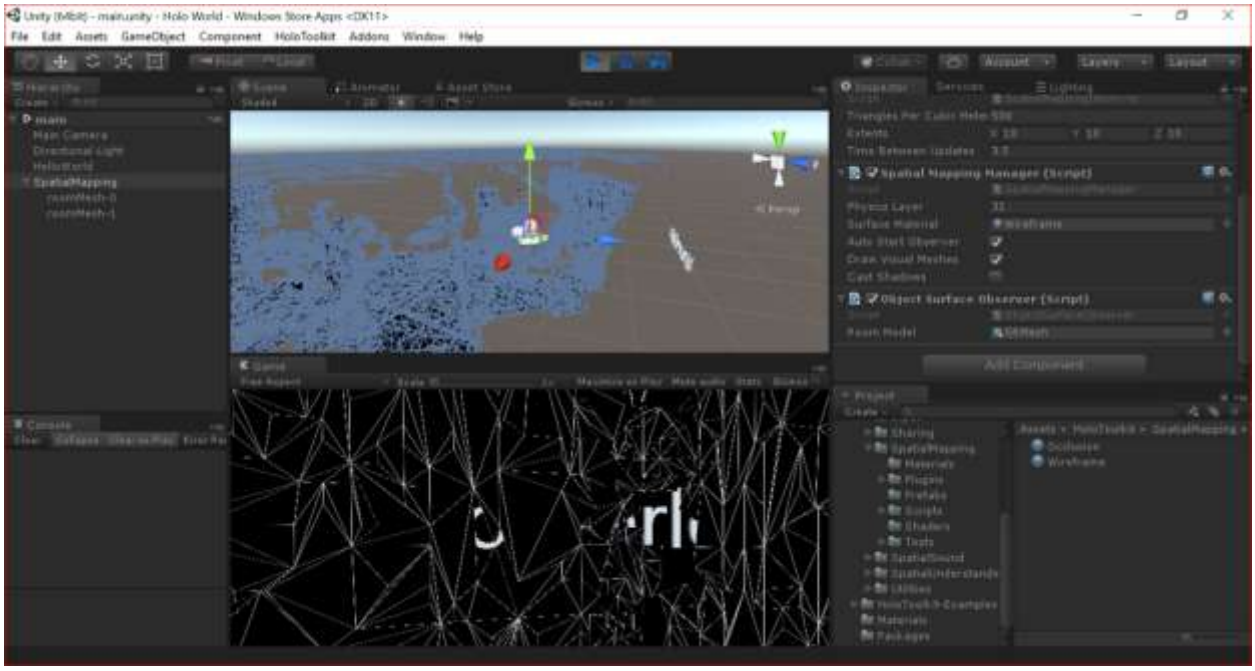
Для начала давайте загрузим готовую модель комнаты и отобразим ее в Unity. Для этого найдите в ресурсах проекта во вкладке Project готовый компонент *SpatialMapping* и добавьте его в сцену. Данный компонент состоит из 3х частей:

- *SpatialMappingObserver* – скрипт использующий API Unity *SurfaceObserver* для получения поверхностей от HoloLens. В нем можно задать интересующий вас объем в пространстве, в рамках которого вы хотите получать поверхности, количество полигонов на кубометр пространства и частоту обновлений. Анализ поверхностей – трудоемкий процесс, поэтому точная настройка параметров крайне важна для быстрой работы приложения.

- *SpatialMappingManager* – управление источником поверхностей, как правило это скрипт SpatialMappingObserver в эмуляторе или ObjectSurfaceObserver в редакторе Unity.

- *ObjectSurfaceObserver* – загрузка готовой модели комнаты из устройства или эмулятора.

В импортированном пакете HoloToolkit уже есть готовая модель комнаты. Давайте загрузим ее в *ObjectSurfaceObserver*. Выберите в панели *Inspector* поле *Room Model*, найдите в поиске объект SRMesh и выберите его двойным кликом. Запустите приложение и у вас должна появиться модель комнаты, отрисованная белой сеткой. При этом объекты поверхностей будут динамически добавлены при запуске в объект *SpatialMapping*.



Как видно на скриншоте, добавленная ранее надпись практически полностью скрывается моделью помещения. Это также одна из приятных возможностей интеграции с пространством в HoloLens — можно добавить тени от голограмм или перекрытие голограмм окружающим миром.

Взгляд, жесты и привязка к поверхностям

Решить данную проблему можно разными способами. В рамках данной статьи мы позволим пользователю устанавливать данную надпись на окружающих поверхностях используя взгляд и жесты.

Для этого добавим 2 скрипта отвечающих за управление взглядом и жестами к главной камере – *GazeManager* и *GestureManager*. К объекту надписи добавим скрипт *Tap To Place* и *Box Collider*. Отключим отображение комнаты по умолчанию через свойство *Draw Visual Meshes* у компонента *SpatialMappingManager* и запустим приложение.

В режиме игры посмотрите на надпись и нажмите пробел. Теперь надпись будет перемещаться вместе с нами и если посмотреть на поверхность – будет автоматически к ней «прилипать».

Каждый кадр *GazeManager* проверяет на какой объект смотрит пользователь и запоминает его используя стандартное API Unity Physics.Raycast. Помните мы добавили *Box Collider* для нашей надписи? Этот объект описывает форму объекта для просчета столкновений физическим движком, что позволяет нам легко проверить смотрит ли человек на объект.

Далее по тапу (клавиша пробел в Unity) проверяется смотрит ли человек на какой-то объект, и если да — объекту посылается сообщение OnSelect по которому для нашей надписи в скрипте *TapToPlace* вызывается одноименный метод, который начинает или завершает процесс ее перемещения.

При этом скрипт *TapToPlace* по окончанию процесса перемещения сохраняет позицию объекта в пространстве, привязывая его к так называемому пространственному якорю (World Anchor). При этом сам якорь сохраняется в память устройства и при следующем запуске приложения скрипт его загрузит и надпись появляться там же, где ее оставил пользователь.

У стандартной реализации скрипта *TapToPlace* есть 2 проблемы, которые нам нужно решить:

- при установке на полу надпись будет наполовину погружена в пол.
- надпись будет повернута от пользователя, ее вектор forward будет совпадать с forward камеры. Неплохо бы ее развернуть к пользователю.

Откроем скрипт *TapToPlace* в VisualStudio/MonoDevelop/Notepad/etc. и поменяем код, отвечающий за установку объекта на поверхности.

Установка объекта в TapToPlace

В этом кусочке мы берем позицию пересечения луча взгляда пользователя с окружающей поверхностью:

```
Vector3 surfacePoint
```

и поднимаем ее по вертикальной оси на половину размера нашей надписи. Так, чтобы она оказалась над уровнем пола.

```
surfacePoint.y += GetComponent<Renderer>().bounds.extents.y/2;
```

```
this.transform.position = surfacePoint;
```

Далее мы разворачиваем надпись на 180 градусов вокруг вертикальной оси с учетом поворота камеры. Получается при привязке надпись всегда будет обращена по одной оси к пользователю.
На этом все, наша tap to place логика готова.

Голосовые команды

В API Windows 10 есть крайне мощное API для работы с распознаванием и генерации речи. HoloLens имеет на борту 4 микрофона и прекрасно распознает речь. Поэтому голосовое управление в приложении крайне удачно дополняет работу с жестами.
Как пример реализации подобной логики мы напишем с вами простой скрипт, который будет реагировать на фразу *Change color* и случайно менять цвет надписи. Прим. автора: код скрипта оставлен максимально простым и приведен в

качестве примера.
Для этого добавим в проект *C# Script* и назовем его Change Color Command. Для этого в меню Unity выберите *Assets-> Create -> C# Script*. Дважды кликните по нему в Unity чтобы открыть редактор.

Шаблон скрипта

Добавим в секцию *using* следующую строчку:

```
using UnityEngine.Windows.Speech;
```

Это позволит нам использовать встроенную в Unity обертку для работы с голосовыми командами. Добавим поле-ссылку на объект типа KeywordRecognizer и ссылку на компонент Renderer у которого мы будем менять цвет.

```
private KeywordRecognizer keywordRecognizer;
public Renderer target;
```

Выполним начальную инициализацию данного компонента внутри метода Start. Метод Start будет вызван единожды после добавления этого скрипта в сцену.

```
void Start()
{
    var commands = new[] { "Change color" };
    keywordRecognizer = new KeywordRecognizer(commands);
}
```

В методе Start мы задаем массив команд, которые система будет слушать. При успешном распознавании у keywordRecognizer будет вызвано событие OnPhraseRecognized, в котором мы будем выполнять смену цвета. Добавим в конец метода подписку на событие и запуск распознавания голоса.

```
keywordRecognizer.OnPhraseRecognized += KeywordRecognizerOnOnPhraseRecognized;
keywordRecognizer.Start();
```

и простую функцию-обработчик:

```
private void KeywordRecognizerOnOnPhraseRecognized(PhraseRecognizedEventArgs args)
{
    if (args.text == "Change color")
        GetComponent<Renderer>().material.color = new Color(Random.value, Random.value,
Random.value);
}
```

Внутри функции мы проверяем что распозналась фраза «Change color» и меняем цвет материала объекта, к которому будет привязан этот скрипт, на случайный.

ChangeColorCommand.cs

Добавим скрипт ChangeColorCommand на объект надписи и запустим приложение в Unity, чтобы проверить результат. Для проверки произнесите «Change color» и посмотрите как изменится цвет надписи.

Оптимизация графики

В данный момент наша надпись использует медленный стандартный шейдер Unity что больше подходит для высокопроизводительных платформ, чем для носимых устройств и HoloLens.

Кроме того для HoloLens есть ряд специфичных оптимизаций с точки зрения графики, например использование min16float при реализации шейдеров.

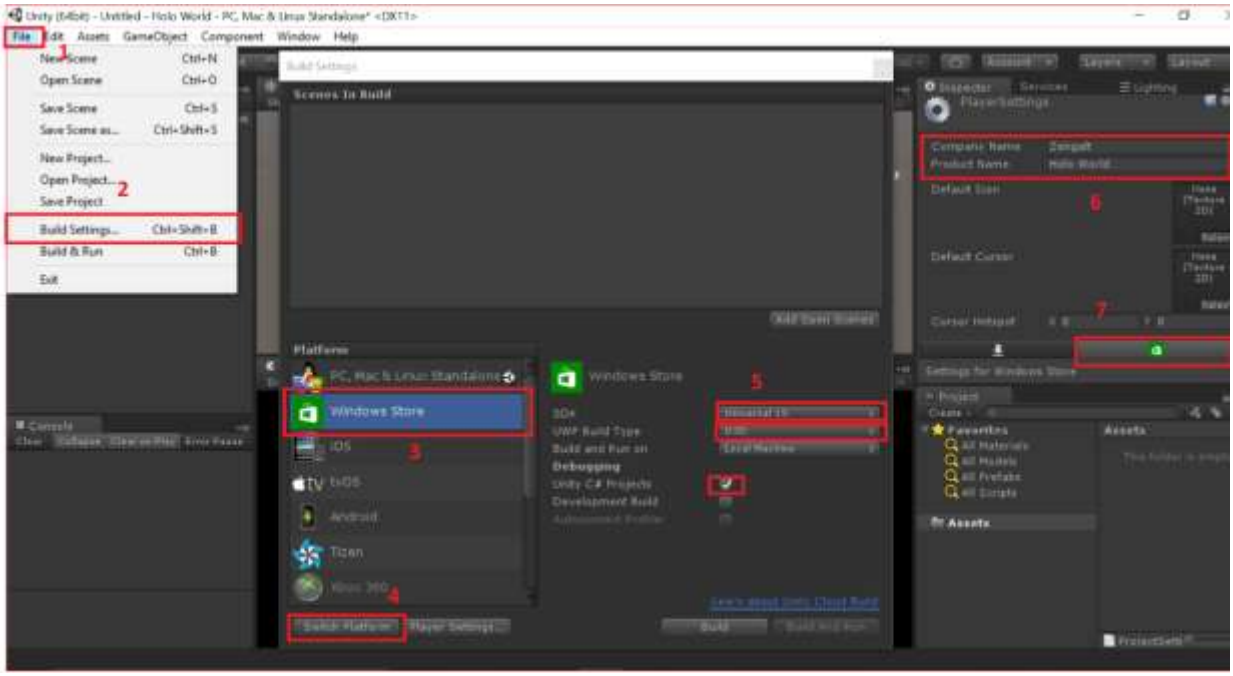
В рамках статьи мы не будем детально рассматривать написание шейдеров под HoloLens. Да и к тому же в HoloToolkit есть набор настраиваемых оптимизированных стандартных шейдеров, которые покроют большинство нужд.

Давайте заменим шейдер нашей надписи на StandardFast. Выберите надпись в иерархии сцены, и найдите в панели *Inspector* секцию с *Material*. Внутри в выпадающем списке выберите HoloToolkit -> StandardFast.

Даже только с одной надписью наше приложение могло работать с FPS ниже 60, что негативно сказывается на стабильности голограмм.

Шаг 3: Настройка параметров сборки

Для запуска нашего приложения на HoloLens нам нужно настроить соответствующие параметры сборки в Unity – тип Windows проекта, название и Capabilites пакета с приложением.



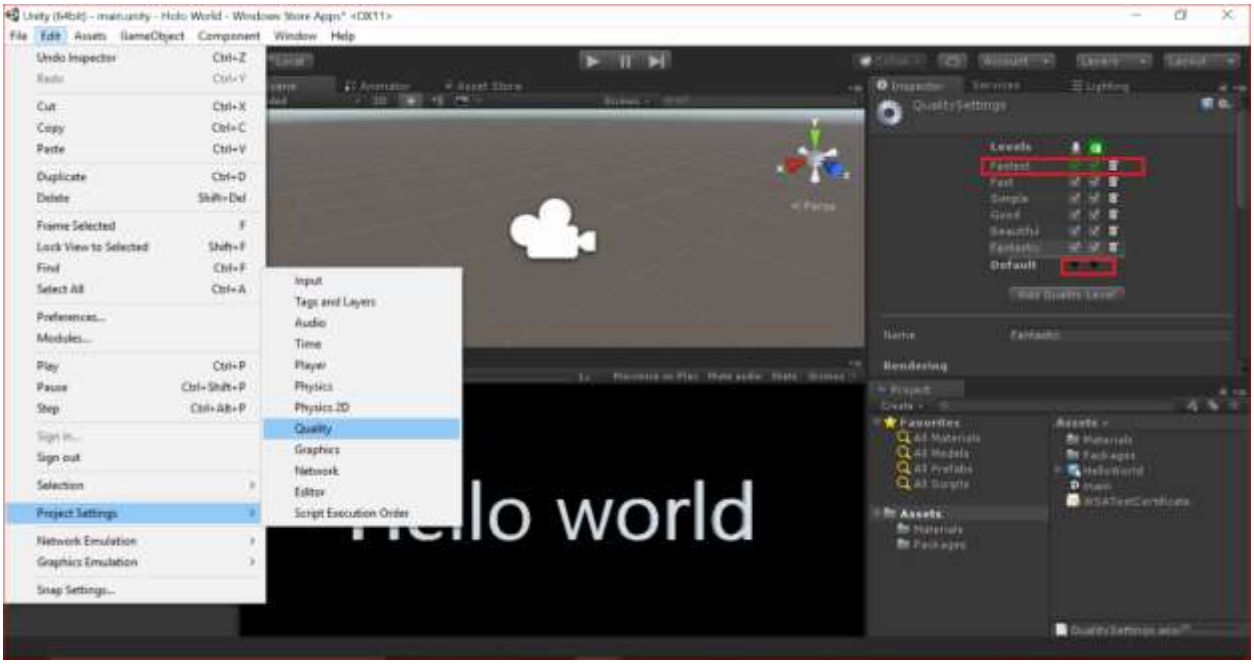
1. Откройте меню *File -> Build Settings*
2. Выберите в списке окна Build Settings Windows Store и нажмите кнопку *Switch Platform*
3. В секции *Windows Store* выберите SDK – Universal 10, UWP Build Type – D3D и поставьте галочку Unity C# Projects для отладки
4. Нажмите кнопку *Players Settings…* в окне Build Settings. В панели *Inspector* заполните название компании и имя приложения. Тут же выберите секцию настроек для *Windows Store*(п.7 на скриншоте)
5. Перейдите к секции *Other Settings* в панели инспектора и проставьте галочку у *Virtual Reality Supported*, у вас должен появиться в списке SDK ниже “Windows Holographic”.
6. Далее прокрутите панель инспектора до конца и в секции *Publishing Settings -> Capabilities*отметьте следующие пункты

    - Internet Client
    - Microphone
    - SpatialPerception

Последним шагом в процессе настройки – оптимизация и качество ресурсов.

Откройте в меню *Edit -> Quality*. В окне *Inspector* нажмите на треугольник в каждом столбце и выберите **Fastest**. Важно выбрать эти настройки и для Unity и для

HoloLens чтобы видеть более-менее актуальную картинку и там и там. В этом профайле отключено сглаживание, импортированные текстуры сжимаются до половины разрешения, отключаются тени и др. Все эти профайлы можно настроить под свои нужды, на данном этапе достаточно настроек по умолчанию для этого профиля производительности.



Шаг 4: Сборка проекта в Visual Studio и запуск в HoloLens Emulator

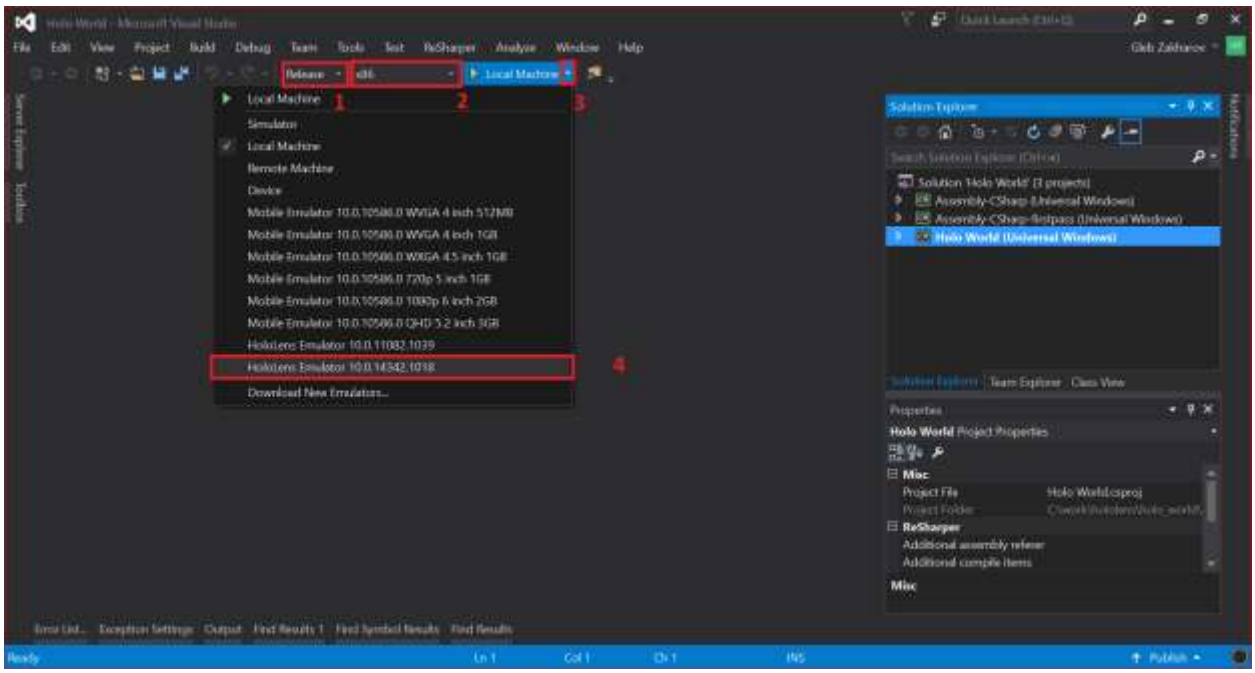
Итак, у нас есть наш первый готовый проект для HoloLens. Чтобы запустить его на устройстве или эмуляторе нам нужно сгенерировать проект для Visual Studio, построить и залить сборку в эмулятор или устройство.

Построение проекта

Тут все мало отличается от стандартного процесса сборки из Unity.

Откройте окно *Build Settings*, нажмите *Add Open Scenes* и далее *Build*. Unity предложит вам выбрать папку где сгенерировать решение для Visual Studio. Мы в наших проектах используем для этого папку build в корне проекта. После завершения процесса сборки откройте сгенерированное решение в Visual Studio.

Для запуска на эмуляторе установите в панели отладки следущие параметры: Release, x86, HoloLens Emulator 10.0.14342.1018

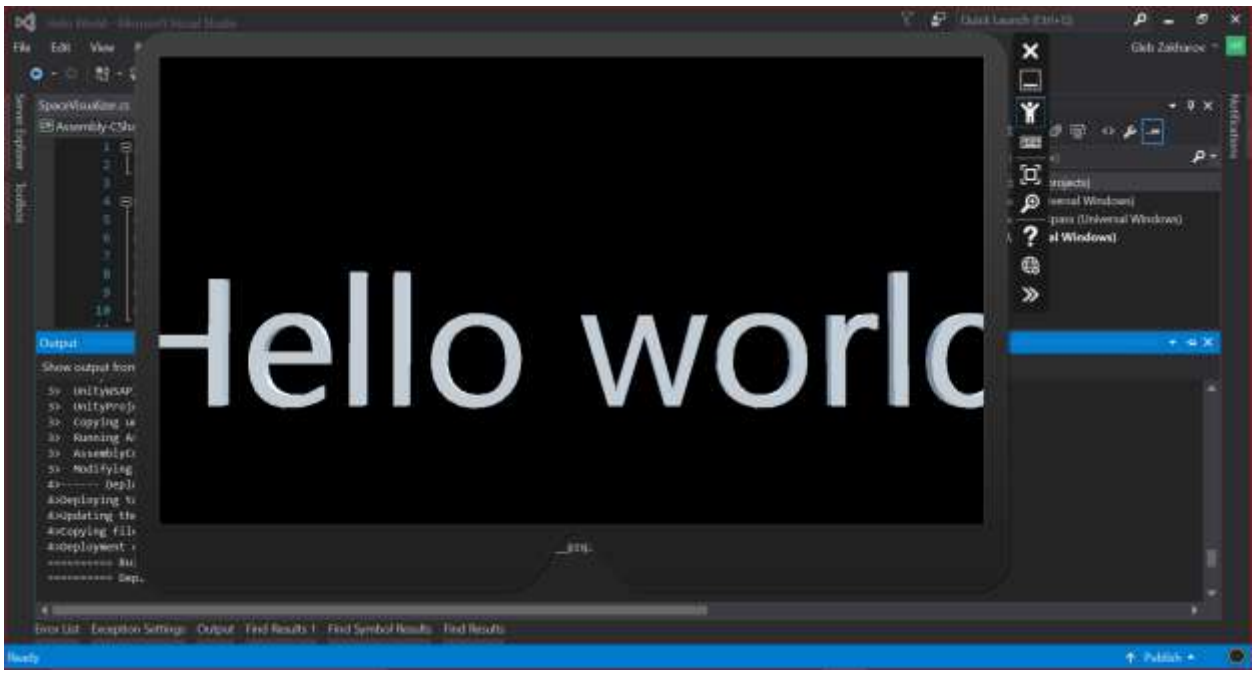


Нажимаем Ctrl+F5 или в меню Debug -> Start Without Debugging.

Во время этого процесса Visual Studio:

- соберет проект
- запустит эмулятор
- дождется загрузки ОС в эмуляторе
- задеплоит и запустит наше приложение

После загрузки приложения в эмулятор мы увидим примерно следующее:

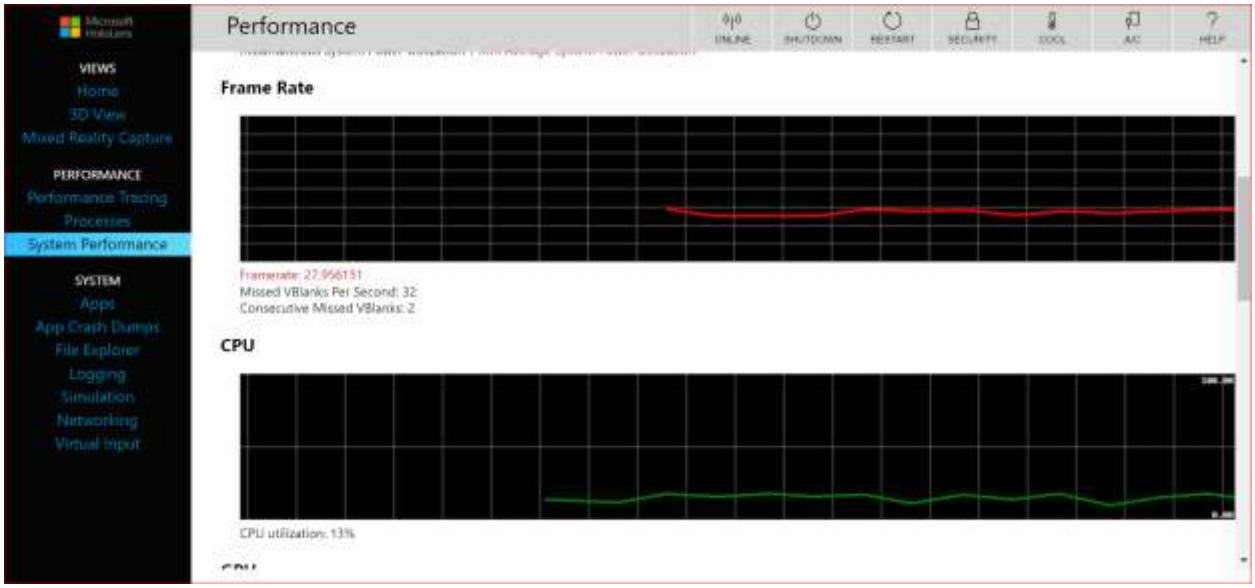


Для перемещения в эмуляторе можно использовать WASD, управление взглядом – ЛКМ + Мышь, Тап – ПКМ, Bloom (выход в shell) – Esc.

Все возможности по управлению вводом смотрите в документации к эмулятору. Есть большой бонус что эту технологию выпустил Microsoft – документация к устройству, его использованию и API крайне насыщенная и полная.

Один из ключевых инструментов используемых вместе с эмулятором или самим HoloLens – Device Portal. В нем легко и удобно можно посмотреть:

- Статус устройства: FPS, Power, CPU, GPU, I/O
- Текущие запущенные и установленные приложения
- Сделать видео или скриншот
- Посмотреть поверхности помещения, определенные девайсом
- Доступ к файловой системе
- Настройки логирования, подключения к сети

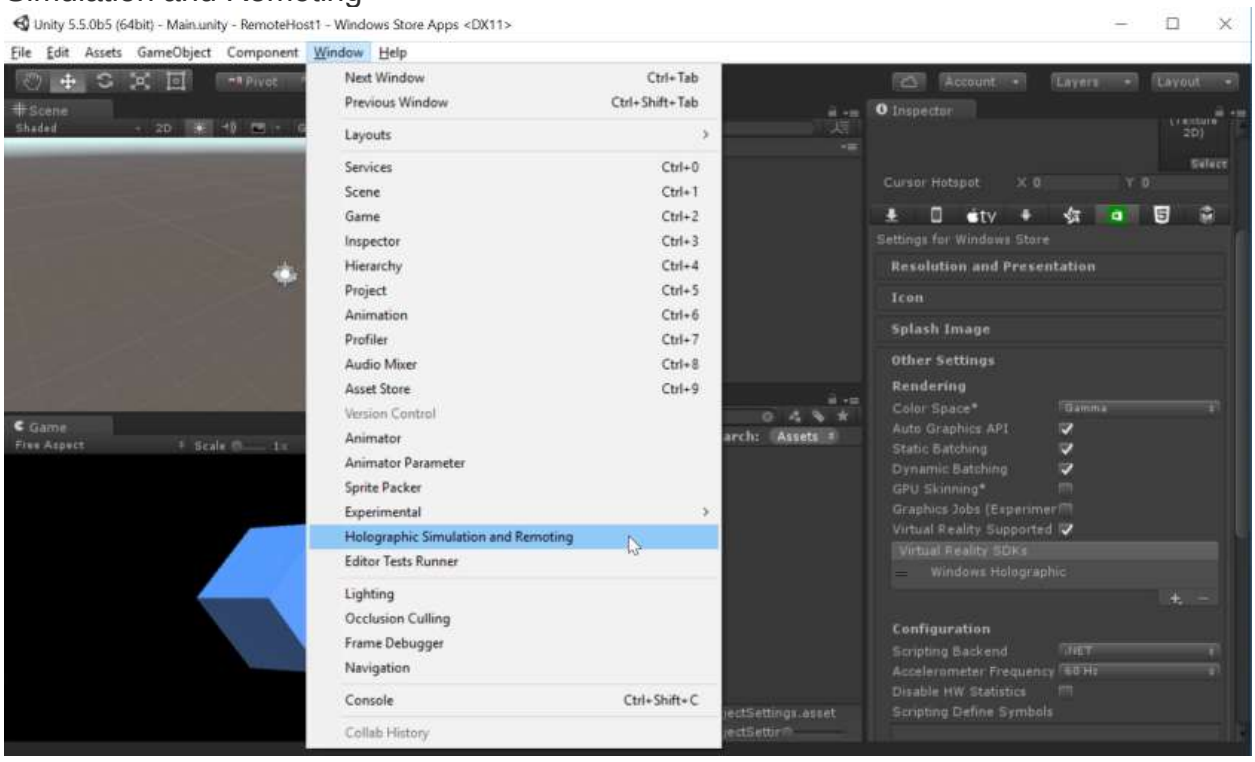


Мой встроенный GPU (Intel 5500) выдает довольно скромный framerate в эмуляторе

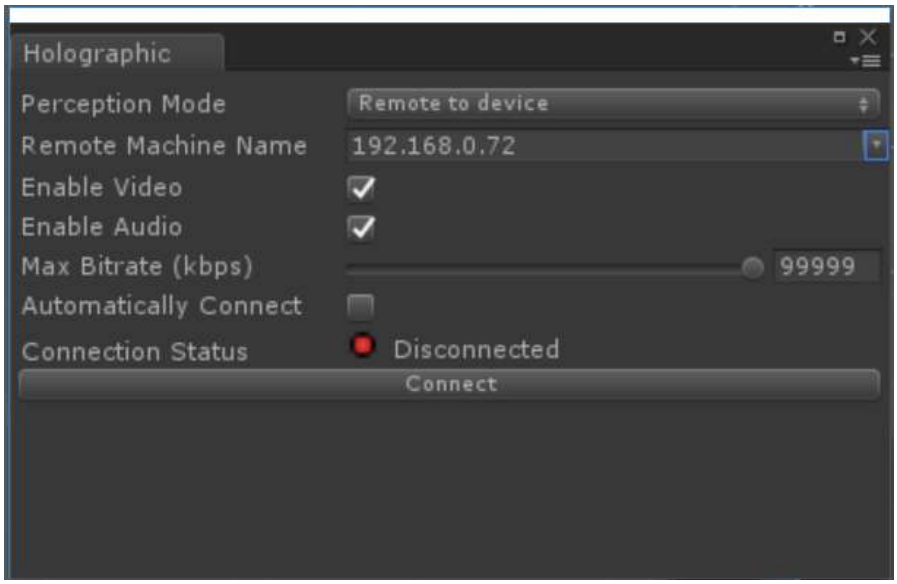
В режиме записи видео на устройстве все выглядит хуже чем на самом деле, т.к. видео пишется в 30 fps. Видны несглаженные линии и есть ощущение небольшого лага. При нормальном запуске этого нет, поэтому важно в итоге посмотреть ваше приложение на реальном устройстве.

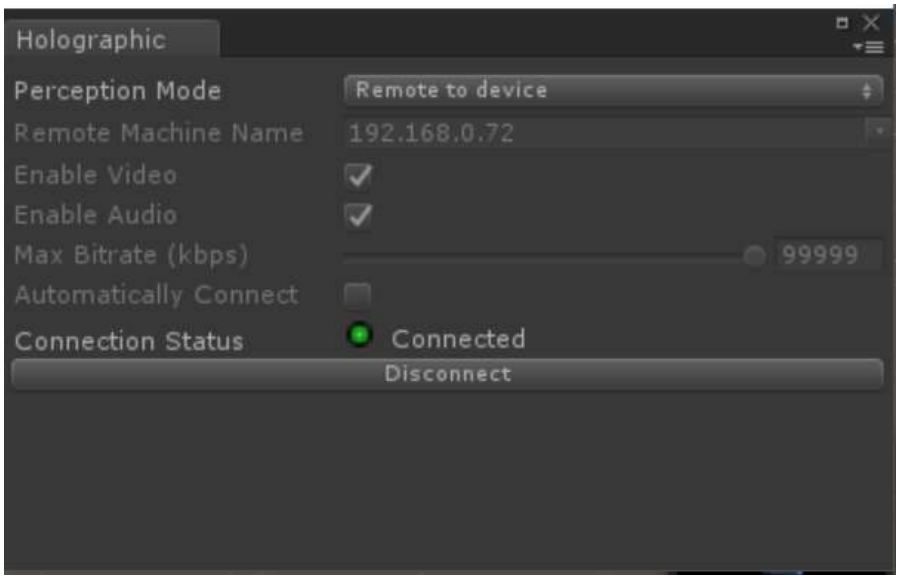
Для работы с устройством Hololens потребуется скачать на самом устройстве в Microsoft Store приложение Hololens Remoting Player

После скачивания, в основном окне Unity выбрать “Windows” / “Holographic Simulation and Remoting”



Указать IP адрес Hololens устройства (его можно найти в настройках Hololens “Wifi/Advance settings”)





Теперь демонстрация будет происходить напрямую в Unity Editor после нажатия Play кнопки

Ресурсы для изучения

- Официальная документация – гайдлайны и спецификации по разработке для DirectX и Unity
- Holographic Academy – серия обучающих видео по разработке под Hololens
- HoloToolkit – набор компонентов для ускорения разработки приложений под HL
- GalaxyExplorer – opensource приложение для HoloLens, опубликованное в Store для HoloLens
- HoloLens App Development Forum – официальные форумы по разработке
- Официальный канал HoloLens на Youtube