Workshop Developing for Windows Mixed Reality Immersive headsets

Goal:

- build a simple app that allows you place, manipulate and remove trees in 3D space using speech commands and controller actions

Objectives:

- Learn the basics of using Windows Mixed Reality headsets
- Learn to set up a project for Windows Mixed Reality Immersive head sets
- Import external assets
- Basic use of the Mixed Reality Toolkit
- Building a message-based application without relying on hard coded names and asset paths
- Using speech commands and controller actions to manipulate objects
- Deploy a Mixed Reality app to your computer

Part 0: instructor

Demonstrate:

- How to put on a head set
- How to start the Mixed Reality portal
- How to use to controller to move around the Cliff House or the Sky Loft
- How to start apps in the start menu
- How to use Win-Y to change input from the desk top
- Show how you can start apps from the desk top and then move into MR

After the demonstration, allow for like 10 minutes for the attendees to play around with apps.

Part 1: Setting up the environment

In this part you will

- Check your machine's basic configuration
- Install your development environment if you haven't already done so.
- Configure Windows Mixed Reality

Check your machine's basic configuration

Your system needs to run Windows 10 1809 64 bit and have the Mixed Reality Portal installed

Your PC needs to meet hardware requirements described in https://docs.microsoft.com/en-us/windows/mixed-reality/enthusiast-guide/windows-mixed-reality-minimum-pc-hardware-compatibility-guidelines

You can also check your machine's capability to run Windows Mixed Reality apps with this app https://www.microsoft.com/en-us/p/windows-mixed-reality-pc-check/9nzvl19n7cnc

If the PC does not meet the minimum requirements you might be able to *develop* for Windows Mixed Reality, but you can't *run* the app on a headset.

Install development environment

If you have not already done so, install the following prerequisites:

- Visual Studio 2017 Community Edition (not necessary if you have already a Visual Studio 2017 version installed)
- Unity 2017.4.12f1

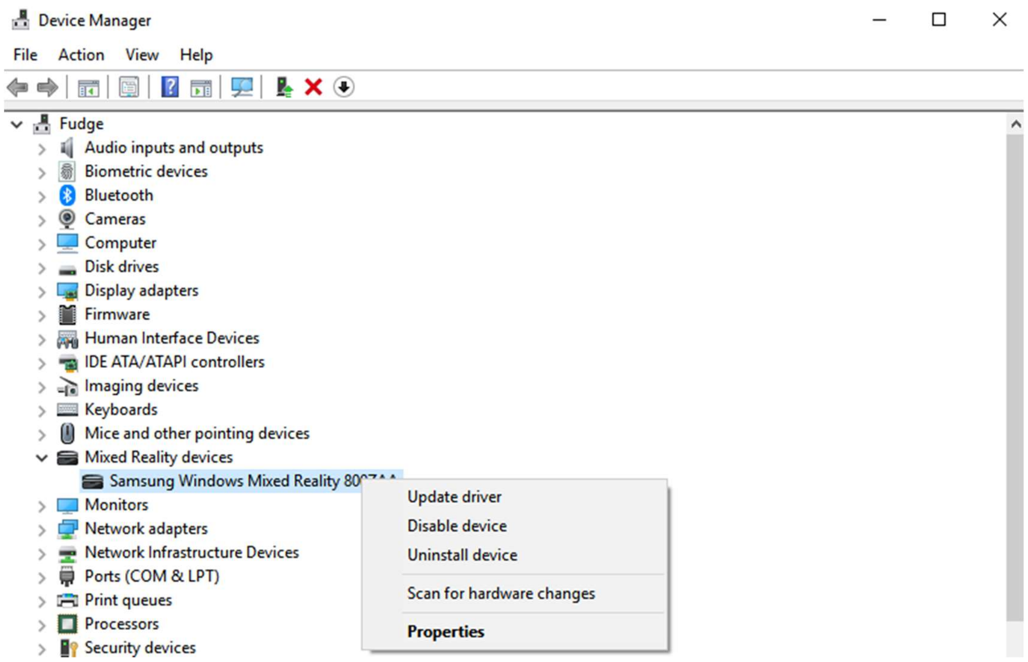
You will find the required files in the "Downloads" folder of the workshop materials.
Steps:

1. First, install Visual Studio.
2. Then, click the "install.bat" files in the Unity folder to install all the required Unity files

Temporarily disable Windows Mixed Reality

If you start to work in Unity, Windows Mixed Reality will pop up, rendering interaction with the Game Window (will be explained later) impossible. To make sure that does not happen prematurely, for now we will disable Mixed Reality devices.

- Start up the device manager
- Find and expand the node "Mixed Reality Devices"
- Disable the device(s) listed there



Part 2: Creating a standard Mixed Reality project

In this part you will:

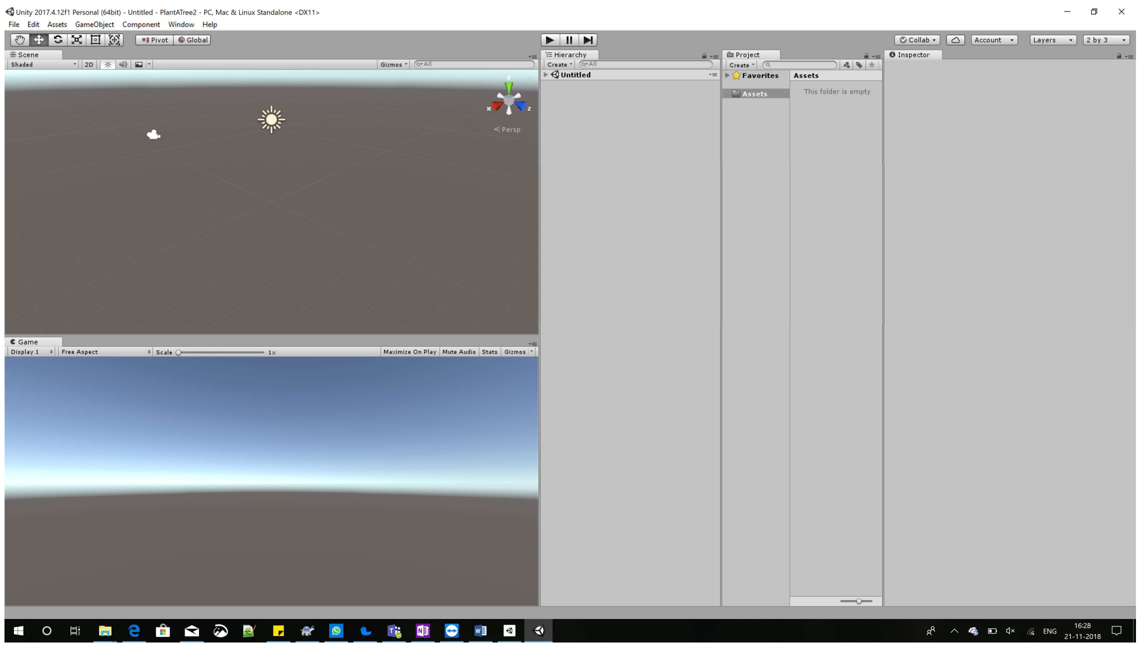
- Create an empty project
- Import some external asset packages
- Configure the Mixed Reality Toolkit settings
- Create and save your first scene
- Do some basic validations
- Select the right audio settings
- (Optionally) do some Unity settings to distinguish between play mode and edit mode.

Creating an empty project:

- Click "new"
- Enter a name and a default location
- Leave template to 3D
- Hit "Create Project"

After the project has been created, select "Window/Layouts/2 by 3"

The environment should now look like this:



We are now going to import a few Unity packages. In normal circumstances, you would download these from the Unity Asset Store, or in the case of the Mixed Reality Toolkit, from GitHub.

Importing the Mixed Reality Toolkit

Click Assets/Import Package/Custom Package.
Navigate to the “Files” Section of the workshop materials

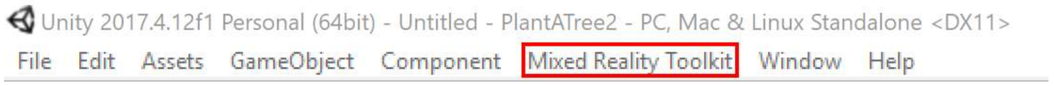
Select the following package: **HoloToolkit-Unity-2017.4.2.0.unitypackage**

After selecting a Unitypackage you can optionally select which parts you want to import. For now, import everything by clicking “Import” bottom right.

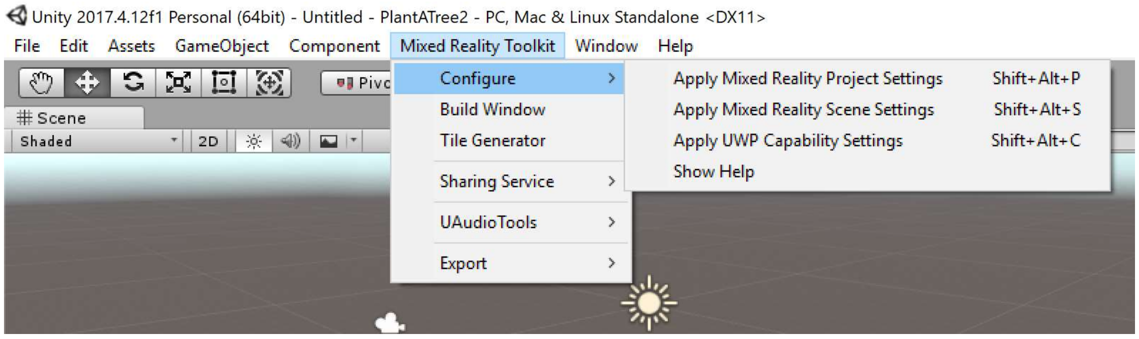
Importing the Mixed Reality Toolkit (MRTK) will take the some time. Please be patient.

Configuring the Mixed Reality Toolkit

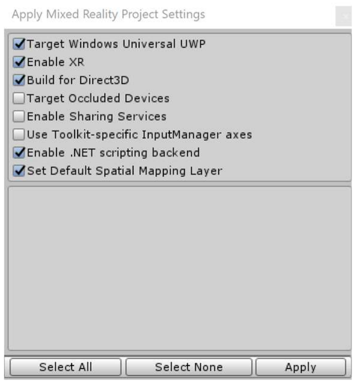
After the MRTK has been imported, you will notice the top of the windows now sports an extra main menu option:


Unity 2017.4.12f1 Personal (64bit) - Untitled - PlantATree2 - PC, Mac & Linux Standalone <DX11>
File Edit Assets GameObject Component Mixed Reality Toolkit Window Help

If you expand that menu and click the ‘configure’ option, you can select three subsections. You will need to use all of them:


Unity 2017.4.12f1 Personal (64bit) - Untitled - PlantATree2 - PC, Mac & Linux Standalone <DX11>
File Edit Assets GameObject Component Mixed Reality Toolkit Window Help
Configure
Build Window
Tile Generator
Sharing Service
UAudioTools
Export
Apply Mixed Reality Project Settings Shift+Alt+P
Apply Mixed Reality Scene Settings Shift+Alt+S
Apply UWP Capability Settings Shift+Alt+C
Show Help
Scene
Shaded
2D

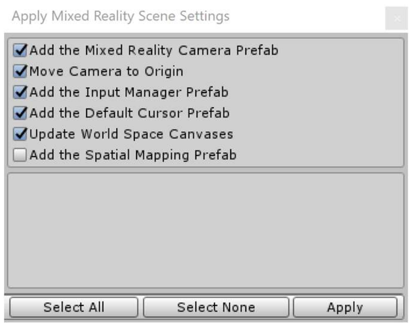
“Apply Mixed Reality Project Settings” yield this dialog.


Apply Mixed Reality Project Settings

- [x] Target Windows Universal UWP
- [x] Enable XR
- [x] Build for Direct3D
- [ ] Target Occluded Devices
- [ ] Enable Sharing Services
- [ ] Use Toolkit-specific InputManager axes
- [x] Enable .NET scripting backend
- [x] Set Default Spatial Mapping Layer

Just hit “Apply”. Unity will shortly show a box “Compiling Scripts….”

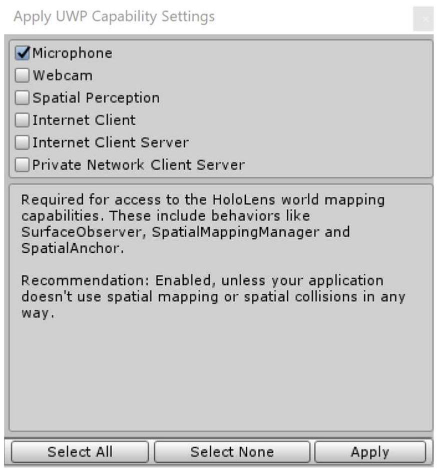
“Apply Mixed Reality Scene Settings” will result in this dialog.


Apply Mixed Reality Scene Settings

- [✓] Add the Mixed Reality Camera Prefab
- [✓] Move Camera to Origin
- [✓] Add the Input Manager Prefab
- [✓] Add the Default Cursor Prefab
- [✓] Update World Space Canvases
- [ ] Add the Spatial Mapping Prefab

If you are planning on using your app on a HoloLens, you might want to select “Add the Spatial Mapping Prefab” as well. If not, just hit “Apply” as well.

Finally, hit “Apply UWP Capabilities Settings”
Since we are using speech commands, we at the very least need the Microphone.


Apply UWP Capability Settings

- [✓] Microphone
- [ ] Webcam
- [ ] Spatial Perception
- [ ] Internet Client
- [ ] Internet Client Server
- [ ] Private Network Client Server

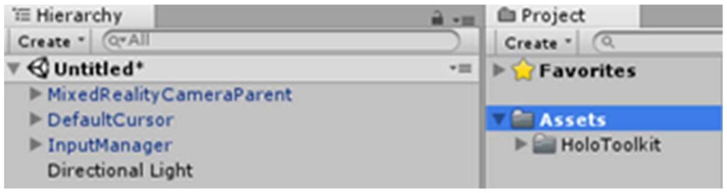
Required for access to the HoloLens world mapping capabilities. These include behaviors like SurfaceObserver, SpatialMappingManager and SpatialAnchor.

Recommendation: Enabled, unless your application doesn't use spatial mapping or spatial collisions in any way.

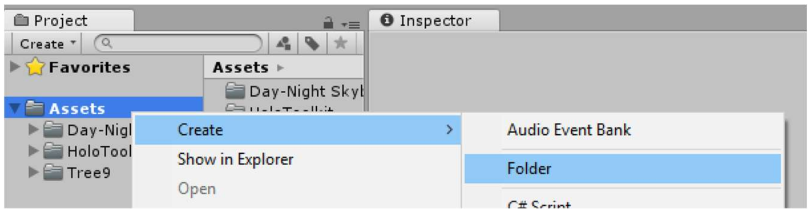
Hit apply.

Saving the first scene

Now over at the right, in the “Hierarchy”, is your scene. The Scene is the place where all your game objects live. Further on the right, you will see the Assets listed. You can see the imported stuff there in folder.

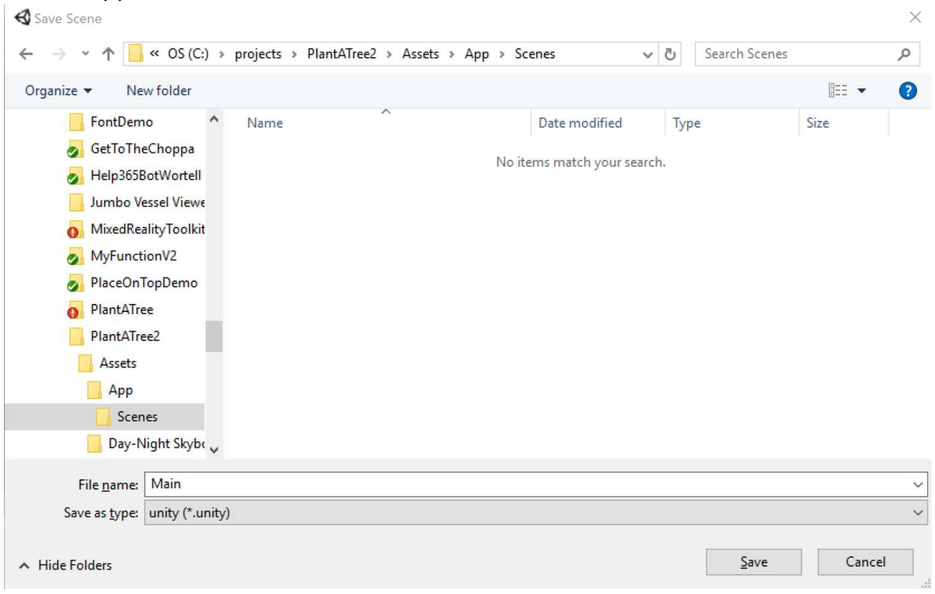


Important now is to save our work, but in a structured way.
Right-click on “Assets”, select “Create”, then “Folder”.



Name your folder App
Then right-click “App”, once again Create”, then “Folder”. Name your folder “Scenes”

Now press CTRL+S (or click File/Save Scene) and save your scene as “Main” (or whatever you like) in Assets/App/Scenes:



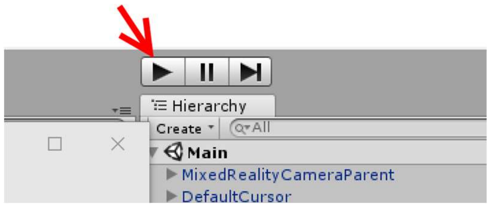
No items match your search.

File name: Main

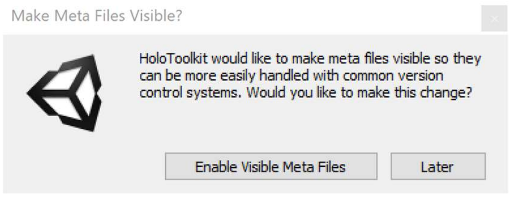
Save as type: unity (*.unity)

<u>Checking if everything is ok</u>

- Hit the play button on top of the screen



- Click "Enable Visible Meta Files



Make Meta Files Visible?

HoloToolkit would like to make meta files visible so they can be more easily handled with common version control systems. Would you like to make this change?

- Allow Windows Defender access if required

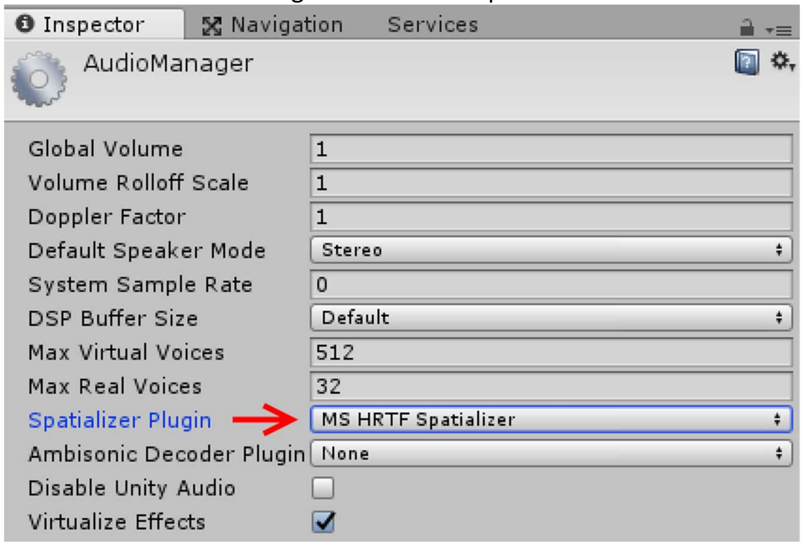
If everything is right you will see a blueish square in the top window (this is the Scene window, where you design the Scene) and the Game window below (that will show a 'runtime' view) will show two hands. If you press the space bar, the right hand will turn blue. If it does not turn blue, try to select the game window first.

That's a general tip: if the Unity Editor does not seem to respond to speech commands or other input in play mode, click the game window first

Exit play mode by pressing the play button again before you continue.

<u>Set the project to use Spatial Audio</u>

If you want to make optimal use of Spatial Audio facilities, click Edit/Project Settings/Audio
And then all over to the right side in the inspector



AudioManager

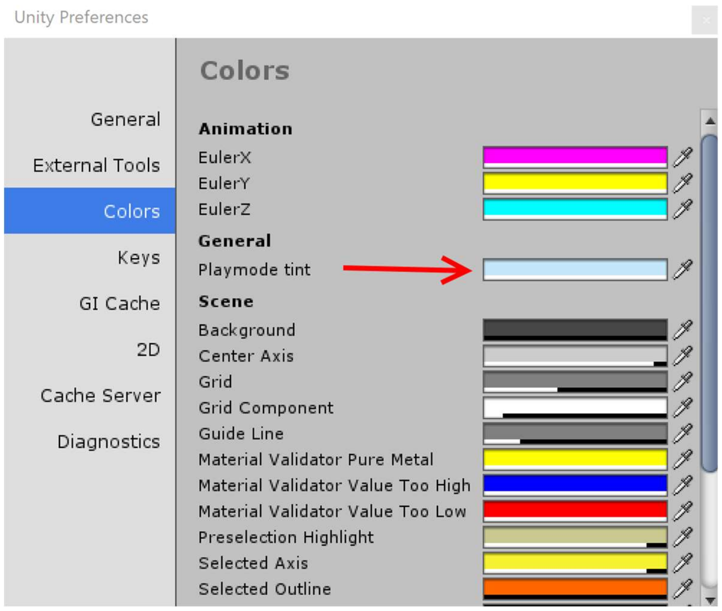
Global Volume	1
Volume Rolloff Scale	1
Doppler Factor	1
Default Speaker Mode	Stereo
System Sample Rate	0
DSP Buffer Size	Default
Max Virtual Voices	512
Max Real Voices	32
Spatializer Plugin	MS HRTF Spatializer
Ambisonic Decoder Plugin	None
Disable Unity Audio	☐
Virtualize Effects	✓

Change "Spatializer Plugin" from "None" to "MS HRTF Spatializer"

<u>(Optional) Play mode color settings</u>

You might have noticed the Unity background in the images in blueish when images are displayed of Unity in play mode (as opposed to design mode where they are gray). The very important reason for this is: you *can* make edits in play mode, but nearly all edits are reverted as soon as you exit play mode. This allows you to tinker with settings (which is good) but also opens an opportunity for hard word to get lost – been there, done that. It's therefore common practice to give 'play mode' a different color:

- Click Edit/Preferences and select tab Colors
- Then select Playmode tint and set a color



Unity Preferences

Colors

Animation
EulerX
EulerY
EulerZ

General
Playmode tint

Scene
Background
Center Axis
Grid
Grid Component
Guide Line
Material Validator Pure Metal
Material Validator Value Too High
Material Validator Value Too Low
Preselection Highlight
Selected Axis
Selected Outline

- Check if the color changes indeed if you click the Play button
- *Exit play mode before you continue*

Part 3: Adding the project specific Assets

In this part you will

- Import some project specific packages
- Copy a few scripts from the resources in the project for some basis functionality

Up until now the project setup has been fairly standard. The nly more or less project specific thing we did was selecting the Microphone capability for this project to support speech commands later on Now we are going to go into project specific things

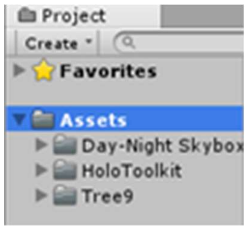
Import project specific packages

When you imported the Mixed Reality Toolkit in the previous part, you might have noticed two other packages

- 10 Skyboxes Pack Day - Night.unitypackage – this contains the 'sky box', which will be explained later
- Realistic Tree 9 Rainbow Tree.unitypackage – these are the trees we are actually going to plant

Unity packages normally come from the Unity Store (https://assetstore.unity.com). This requires you to create an account and sign in to the store. Feel free to do so if you like to explorer the store.

Proceed to import these packages into your Unity project in the same manner. If you have done so, you should see this in your Assets Window:



Copy HoloToolkitExtensions into the project

The HoloToolkitExtensions are some unofficial extensions/helpers and some sounds that can be used in conjunction with the MRTK (but also without). You can find a little excerpt, containing the necessary files for this workshop in Resources\Part3. Copy the folder into the Assets folder of the project. They are basically a script implementing a simple message bus and class utilizing that bus to sound a general confirmation sound – and some sound files
Details of the message bus (Messenger) can be found here
https://dotnetbyexample.blogspot.com/2017/04/using-messenger-to-communicate-between.html

Part 4: Adding speech recognition and feedback sound

In this part you will:

- Create an empty game object
- Search for assets by name
- Learn to intercept speech
- Convert speech commands into messages for the system to interpret
- Debug your app inside the Unity editor
- Learn how to setup keystrokes to test speech control in Unity
- Learn what a prefab is and use it
- Add sound for confirmation

Create the standard object holders

It's a good idea to create a standard format for your scenes, that way people can easily recognize what you are doing and where. Typically, two empty game objects are created in the root:

- HologramCollection – to hold the holograms
- Managers – a 'hat stand' for scripts having a global function in the scene

Creating an empty game object:

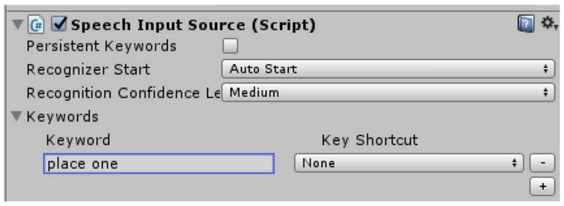
- Right-click on the hierarchy window
- Hit "Create Empty"
- Rename the object (either by clicking it twice, or in the Inspector top right_

Adding the Speech recognition infrastructure

- First, add the messenger. Search for "Messenger" in the assets and drag that on top of the Managers object
- Then, find SpeechInputSource and drag that too on top the Managers object
- Finally, add SpeechInputHandler

Adding keywords

The user should be able to place object 1-4, move, rotate, scale and delete.


Speech Input Source (Script)
Persistent Keywords
Recognizer Start: Auto Start
Recognition Confidence Le: Medium
Keywords

Keyword	Key Shortcut
place one	None

Press + and add the keywords “place one”

<u>Adding some partial scripts for the next step</u>

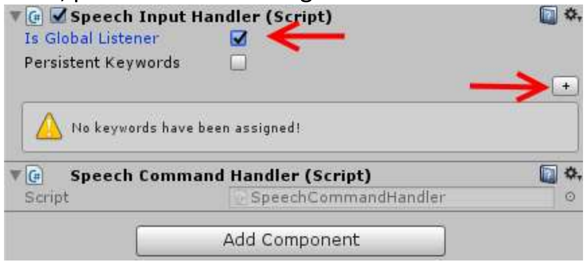
In Resources\Part4 you will find three files:

- CommandMessage
- CommandType
- SpeechCommandHandler

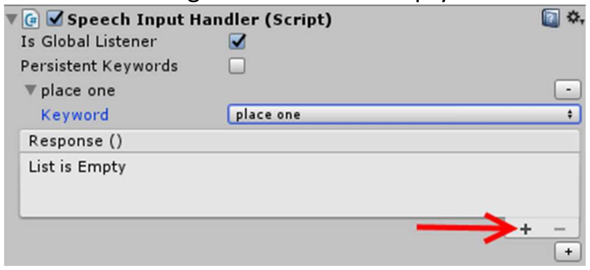
Copy these files into the Assets/App/Scripts folder. The contain a partial implementation of the speech command handling.

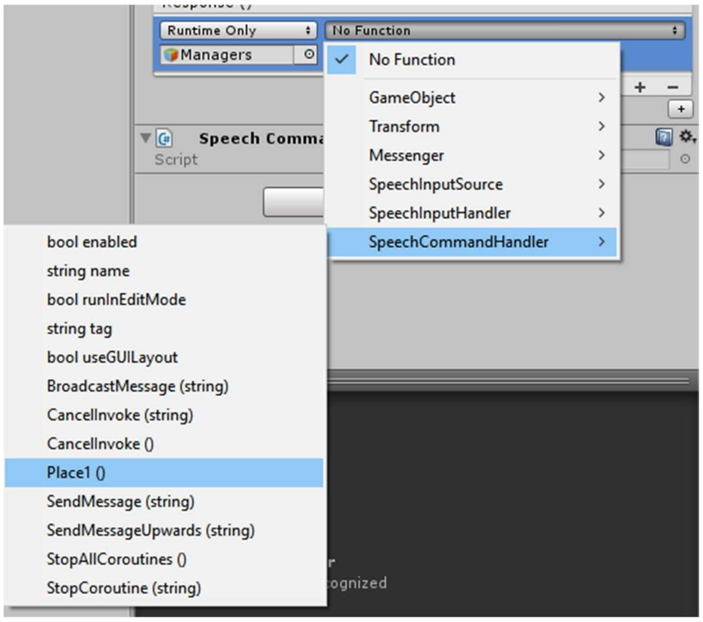
<u>Implementing the first speech recognition</u>

- First, drag SpeechCommandHandler on the “Managers” object below “Speech Input Handler”
- Then, on “Speech Input Handler”, check the “Is Global Listener” check box
- Then, press the little “+” sign


Speech Input Handler (Script)
Is Global Listener ✓
Persistent Keywords
No keywords have been assigned!
Speech Command Handler (Script)
Script SpeechCommandHandler

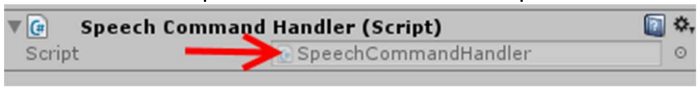
- Expand the “Element 0” that appears
- Click the drop down, select “Place One”
- The click the + sign on the “List is empty”


Speech Input Handler (Script)
Is Global Listener ✓
Persistent Keywords
place one
Keyword place one
Response ()
List is Empty

- In the UI that appears now:
    - Select “Editor and Runtime”
    - Drag “Managers” on top of “None (Object)
    - Select the method “Place1” like this
- 

Testing and debugging your app inside the Unity Editor

You can run and debug your app *in Unity*. To debug, take the following steps:

- Double click the SpeechCommandHandler script box


- This will open Visual Studio in the right file.
- Place a breakpoint the “Place1” method on top
- Run the Visual Studio solution
- Start “Play Mode” in Unity
- Say loud and clear “Play one” into your computer’s microphone
- Now, if the breakpoint is hit, you have successfully configured speech recognition
- Hit continue (or f5) in Visual Studio to unblock Unity

Adding key strokes to simulate speech recognition

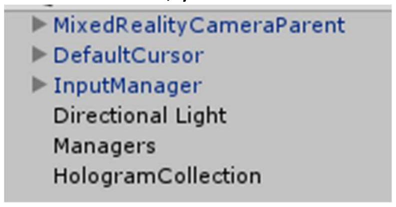
As you may have noticed, testing speech recognition this way can be a bit a awkward for you and even more so for office mates. So, Unity offers a way to simulate that by setting keystrokes to simulate that.

- *Exit play mode*
- Go back to the Speech Input Source
- Behind the “place one” keyword theirs is a “Key Shortcut” drop down that’s set to “None”
- Click the drop down, then select “Alpha 1”
- Start Play Mode again
- Verify the break point is hit again
- Exit play mode

Add confirmation sound

Audio feedback is super important when using a Mixed Reality app. If 'nothing happens' (for instance, because of network delay) people get confused. So, giving audio feedback, indicating a command has understood and action has been initiated, helps giving confirming user things are happening. The Holotoolkit Extensions contain a little helper *prefab* for doing that.

A prefab is like a pre-packaged component containing game objects, attached scripts and assets that you can use in one go. You can recognize them in the Scene's hierarchy by their names being listed as blue. In fact, you have used them already:



```
▶ MixedRealityCameraParent
▶ DefaultCursor
▶ InputManager
  Directional Light
  Managers
  HologramCollection
```

Add confirmation sound goes as follows:

- Dragging the prefab "ConfirmationSound" from HolotoolkitExtensions to the Managers object
- Open the code of SpeechCommandHandler in Visual Studio and adding the following line to the empty method "PlaySound"
  `Messenger.Instance.Broadcast(new ConfirmSoundMessage());`
- Now, if you start Play mode and press 1 (or say "play one") you should hear a clear ringing sound
- If you expand the ConfirmationSound node in the hierarchy, you can see the Prefab consists of a script and an "AudioSource" containing the Audio Clip "Ready".
- Exit play mode before you continue

Challenge

Complete speech recognition

- Complete the enum "CommandType" to include entries for place two, place three, place four, move, rotate, scale and delete.
- Add methods to SpeechCommandHandler for these entries
- Make the "place" commands methods play sound, the other four not
- Have them call the SendMessage method
- Add speech command keywords to the Speech Input Source. Also add key strokes to make testing easier (hint: *never* use the following keys: A, D E, Q, S, W - as these are used to control the camera position in play mode)
- Connect the speech keyword commands to the methods in SpeechCommandHandler as shown on the sample
- Test all commands by debugging and see if they are all properly understood

Part 5 – actually (and finally) showing some stuff

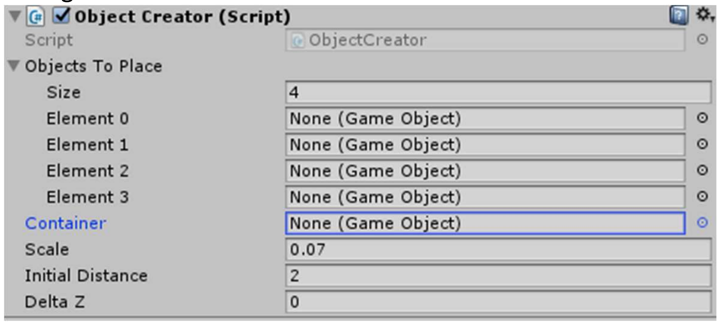
In this part you will:

- Create your first Mixed Reality Holograms
- Generate your first Mixed Reality app
- Deploy the app to your PC
- View your app in a Mixed Reality head set

Basic object creator

In Resources\Part 5 you will find the script "ObjectCreator".

- Put this script it Assets/App/Scripts
- Then, in the Unity editor, drag this script on top of the Manager in the hierarchy.
- Expand the node "ObjectsToPlace".
- Change the value behind Size to 4. You should see now this:

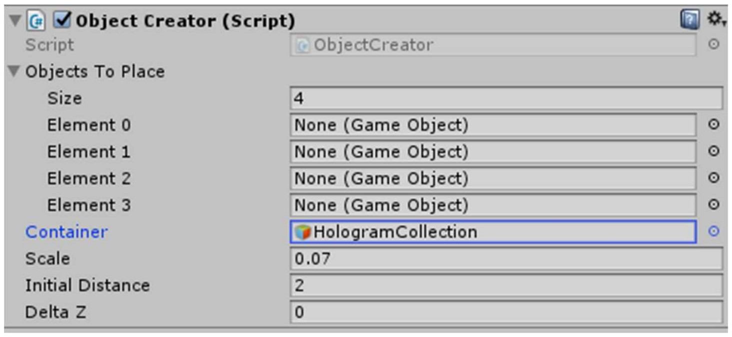


Object Creator (Script)

Script	ObjectCreator
Objects To Place	
Size	4
Element 0	None (Game Object)
Element 1	None (Game Object)
Element 2	None (Game Object)
Element 3	None (Game Object)
Container	None (Game Object)
Scale	0.07
Initial Distance	2
Delta Z	0

Element 0-3 are the trees we are going to place. Container is the game object that will be the 'parent' of the trees, the 'thing' in which they are created.

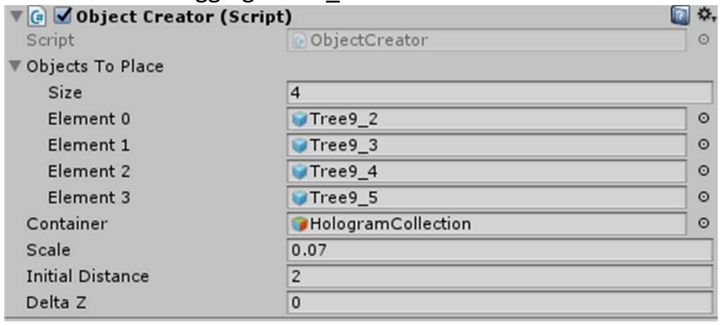
- To set the container, simply drag "HologramCollection" from the Hierarchy into the "Container" field. Net result:



Object Creator (Script)

Script	ObjectCreator
Objects To Place	
Size	4
Element 0	None (Game Object)
Element 1	None (Game Object)
Element 2	None (Game Object)
Element 3	None (Game Object)
Container	HologramCollection
Scale	0.07
Initial Distance	2
Delta Z	0

- Then, look into your Assets folder, open folder Tree9 (all the way to the bottom) and drag the prefab (there it is again) Tree9_2 on “Element 0”
- Proceed with dragging Tree9_3 on “Element 1” etc till all the 4 fields are filled. Net result:

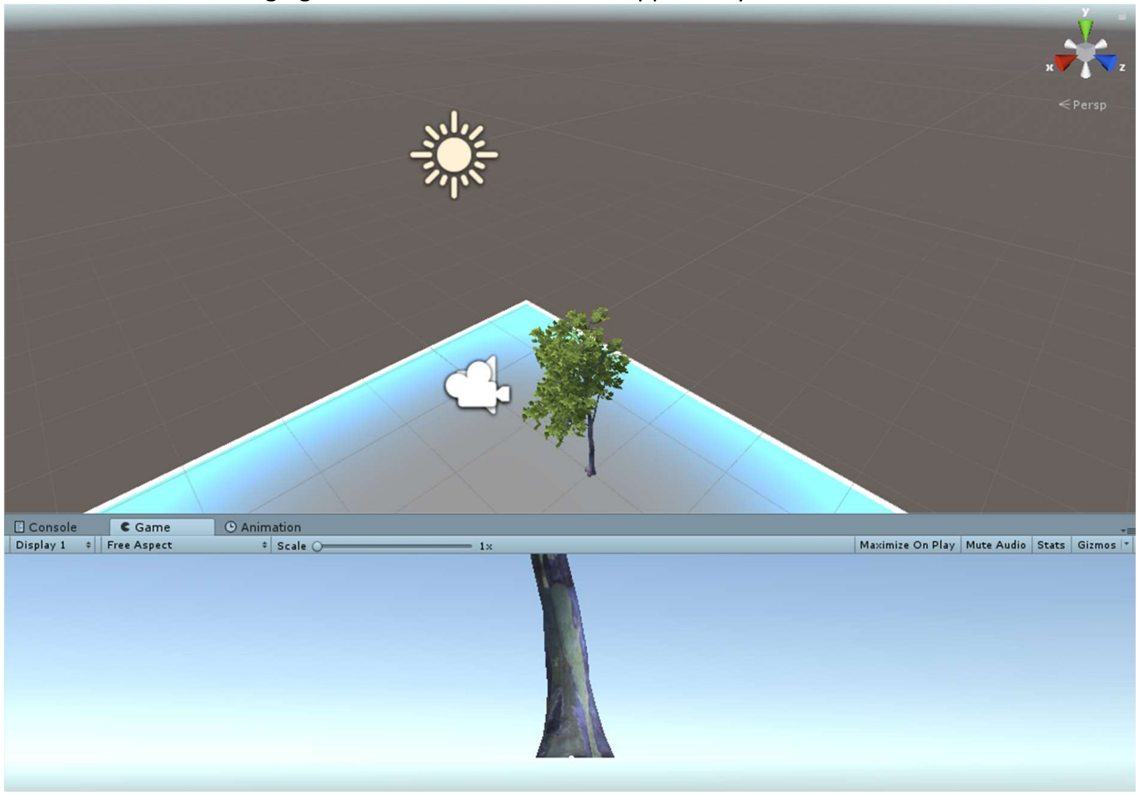


Object Creator (Script)

Script	ObjectCreator
Objects To Place	
Size	4
Element 0	Tree9_2
Element 1	Tree9_3
Element 2	Tree9_4
Element 3	Tree9_5
Container	HologramCollection
Scale	0.07
Initial Distance	2
Delta Z	0

Now the script is armed to create an instance of the trees, 2 meters right before your eyes, at 0 meters height, at a scale of 0.07. Let’s have a look in Unity how that looks.

- Start Playmode
- Press “1”
- You should hear the ringing sound and see a tree trunk appear in you view:

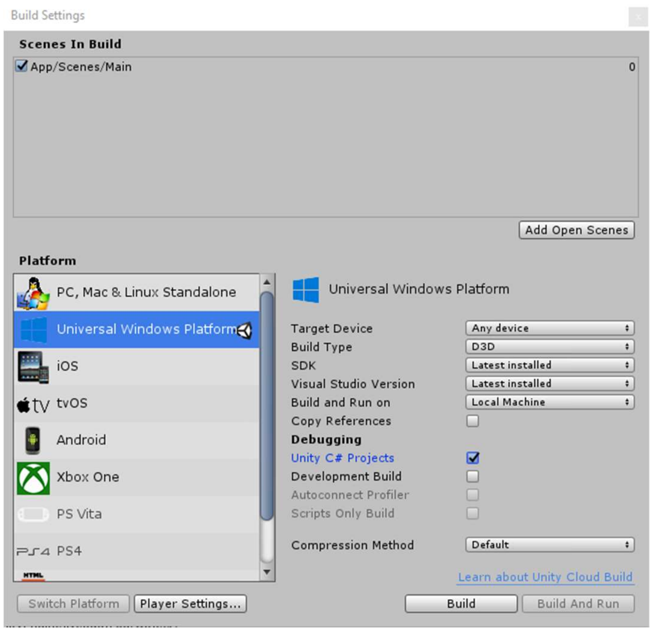


- Click the game window
- Use the S key to move backwards (W to move forward again)
- Drag and click with the right mouse key to rotate the camera
- You can also use the Q and E key to move up and down, and D and A to move from left to right (in gamer land this is called WASD (and QE extra)

Exit play mode before you continue

Generate the UWP app

- Click File/Build Settings (or hit CTRL+SHIFT+B). The following screen popup up
- In the screen that pops up:
  - Click "Add open scenes"
  - Make sure Universal Windows Platform" is selected"
  - Select "Any Device" for Target Device
  - Select "Unity C# Projects"
- Net result should be this:



Build Settings

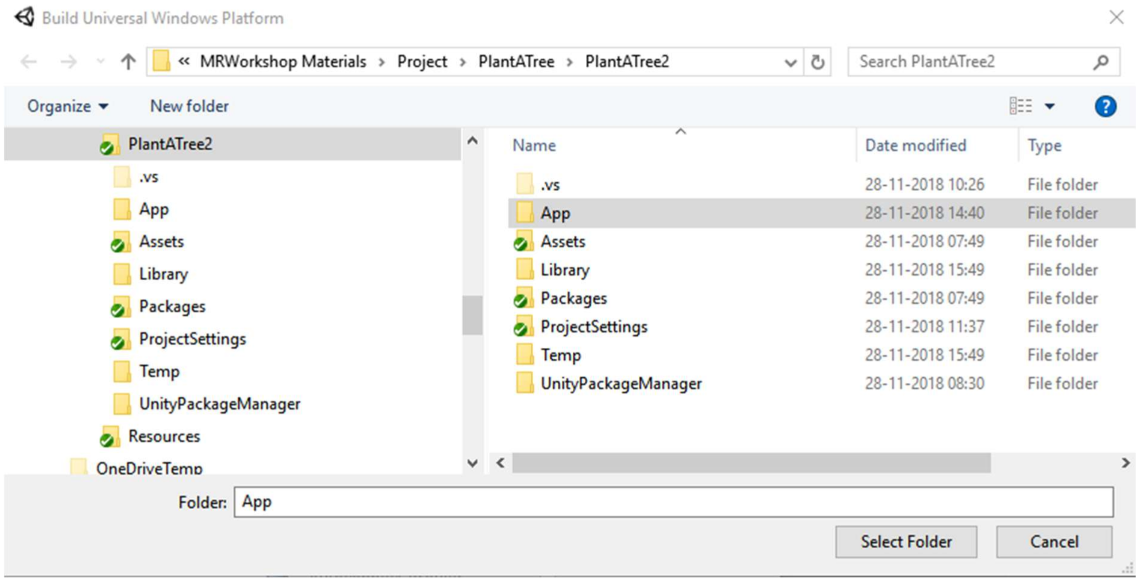
Scenes In Build

☑ App/Scenes/Main 0

Platform: Universal Windows Platform

Target Device: Any device
Build Type: D3D
SDK: Latest installed
Visual Studio Version: Latest installed
Build and Run on: Local Machine
Copy References: ☐
Debugging
Unity C# Projects: ☑
Development Build: ☐
Autoconnect Profiler: ☐
Scripts Only Build: ☐
Compression Method: Default

- Hit the Build button.
- Create a new folder "App" in the project folder and hit select



Build Universal Windows Platform

« MRWorkshop Materials › Project › PlantATree › PlantATree2

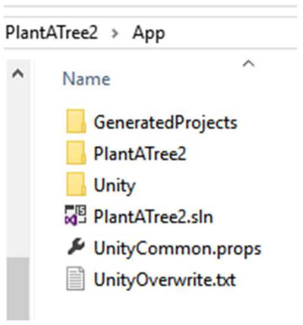
Name	Date modified	Type
.vs	28-11-2018 10:26	File folder
App	28-11-2018 14:40	File folder
Assets	28-11-2018 07:49	File folder
Library	28-11-2018 15:49	File folder
Packages	28-11-2018 07:49	File folder
ProjectSettings	28-11-2018 11:37	File folder
Temp	28-11-2018 15:49	File folder
UnityPackageManager	28-11-2018 08:30	File folder

Folder: App

Be careful – don't select the "Assets" folder inadvertently.

- It's possible the build fails the first time. If so, try again. If it still fails, click the console window check what's wrong – or ask the instructor

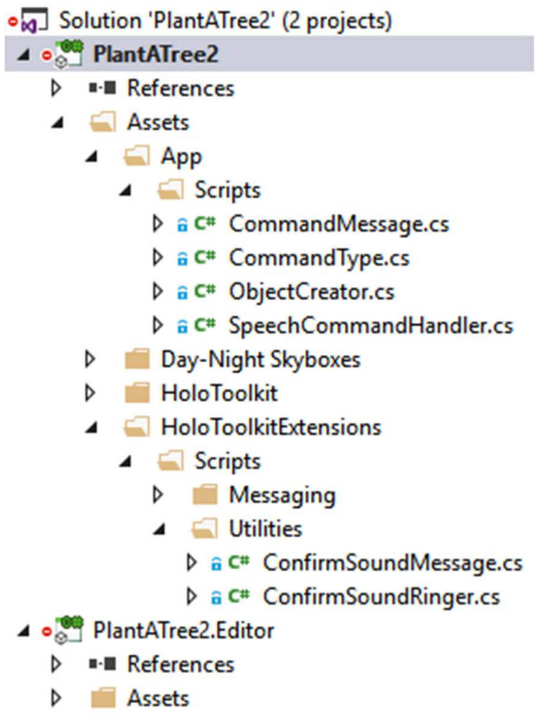
Inside the App folder, there's now a second solution. Previously you have been working in the Unity solution, that's basically a bunch of C# files. You can run it, but only to debug Unity projects, and you will notice there's for instance no properties you can inspect.

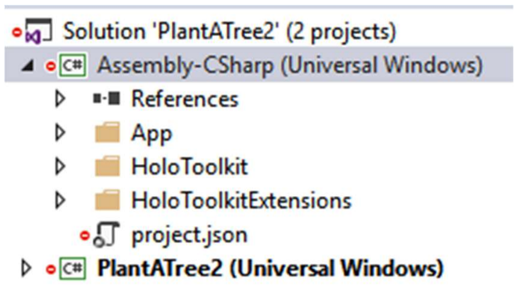

PlantATree2 > App

Name

GeneratedProjects
PlantATree2
Unity
PlantATree2.sln
UnityCommon.props
UnityOverwrite.txt

The other solution, in the app folder is a full-fledged UWP solution. The difference is easy to see: left the Unity solution, right the UWP solution


```
Solution 'PlantATree2' (2 projects)
  PlantATree2
    References
    Assets
      App
        Scripts
          CommandMessage.cs
          CommandType.cs
          ObjectCreator.cs
          SpeechCommandHandler.cs
      Day-Night Skyboxes
      HoloToolkit
      HoloToolkitExtensions
        Scripts
          Messaging
          Utilities
            ConfirmSoundMessage.cs
            ConfirmSoundRinger.cs
  PlantATree2.Editor
    References
    Assets
```

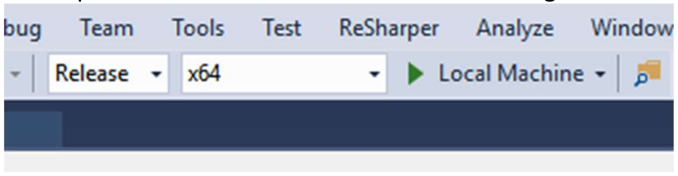

```
Solution 'PlantATree2' (2 projects)
  Assembly-CSharp (Universal Windows)
    References
    App
    HoloToolkit
    HoloToolkitExtensions
    project.json
  PlantATree2 (Universal Windows)
```

Unfortunately, they have the same name. Important things to know:

- You *can* change code in both solutions
- *Adding* or *deleting* files and folders only works in the Unity solution (left). The UWP solution has just links to the files.

Running the UWP App

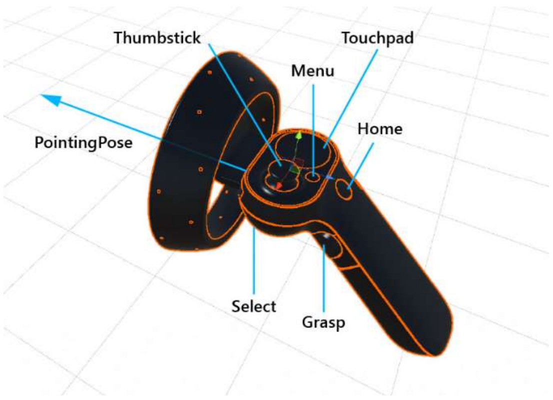
Now open the UWP solution. Select the following build configuration



You can choose either Release or Debug. Don't select Master – that's only for generating an app that can be posted to the Microsoft Store. That's super optimized and fast, but it takes forever to build. Debug is slow but allows you to set breakpoint, Release is a good middle ground if you just want to run it.

Now first enable your Mixed Reality device using the Device Manager again. Then run the solution, press F5, whatever you fancy and put on the head set. If everything works, you should what you see in Unity, only in 3D

- The first time, or every time the app is redeployed from scratch, the app will ask for permission to use for instance your microphone. Point your controller to the OK button and press the Select trigger
- Use the thumb stick to teleport and rotate.
- Say "place one", "place two", "place 3" or "place four" to see a tree appear



Some little things about the ObjectController script: basically this is what it's all about

```
var newObj = Instantiate(_objectsToPlace[objIndex],
    CalculatePositionDeadAhead(_initialDistance) + Vector3.up * _deltaZ,
    Quaternion.identity, _container.transform);
newObj.transform.localScale *= _scale;
```

Basically:

- Create a new object 2 meters before the use, optionally move it a bit up (we have set that to 0)
- No rotation
- Create it inside the container object

- Scale it down to 0.07th of the original scale (if you like giant trees or rather bonsai trees, feel free to fiddle with it.

Some other things to notice.

- ObjectController is a MonoBehaviour child. This means it's a component that can be attached to a game object and be dragged on top of it in the Inspector.
- On top of ObjectController there's this:

```
[SerializeField]
private float _initialDistance = 2;

       [SerializeField]
       private float _deltaZ = 0f;

       private void Start()
       {
          // Listen to command messages
           Messenger.Instance.AddListener<CommandMessage>(
           CommandMessageHandler);
       }
```

Unity has a bit odd approach to what's private and not. The [SerializedField] attribute for instance makes a private field accessible to the editor, so although the values are safe runtime, you can actually provide and change values in the editor. There are also a couple of methods that although the are private, the game engine can actually call them – from outside. They go by name convention. "Start" is one, called when the object is created. If you define a method void Update(), it's called 60 times a second. There are a couple of more of these methods. See https://docs.unity3d.com/ScriptReference/MonoBehaviour.html for an overview.

Part 6 – selecting and moving

In this part you will:

- Learn how to select an object
- How to move an object

Adding some basic scripts

In Resources\Part 6 you will find the scripts ObjectManipulator.cs and SelectedMessage.cs. Put these scripts in Assets/App/Scripts.

- ObjectManipulator is a script that will be added to each tree, so it can be selected (and moved, etc.)
- SelectedMessage is the message that the selected object will passed around to all other trees – so they know they are *not* selected.

Enabling selection

Revisit the ObjectCreator script. Upon creation, the tree must be outfitted with at least two components:

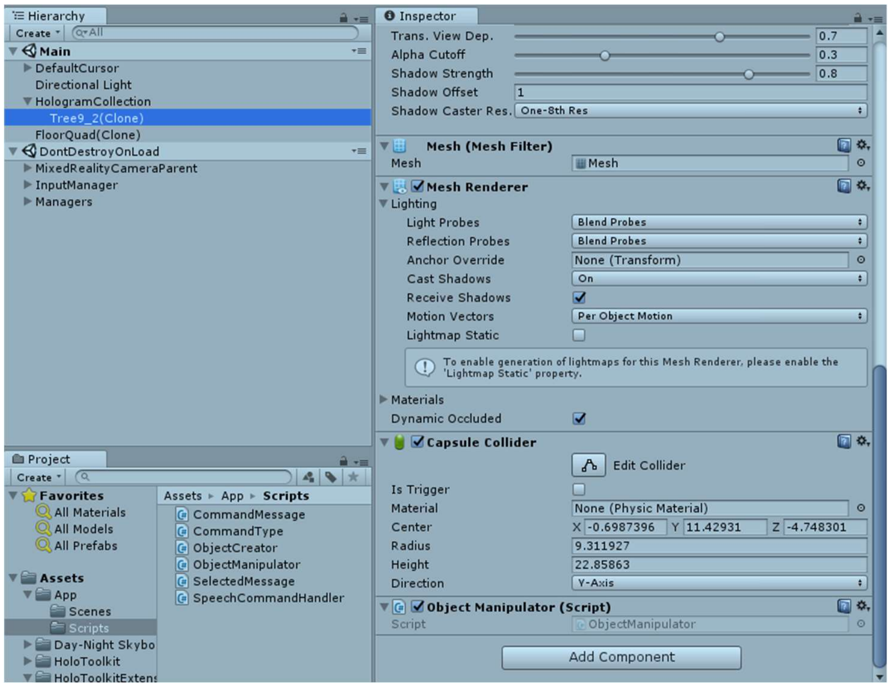
- An ObjectCreator
- A so-called Collider. A Collider is a component that makes it hittable other object – case in point, the gaze cursor that follows your gaze.

Add the following two lines to the ObjectCreator's PlaceObject method

```
newObj.AddComponent<CapsuleCollider>();
newObj.AddComponent<ObjectManipulator>();
```

There are several types of colliders. This one roughly follows a capsule-like boundary of the object. You can also choose a BoxCollider - that is basically a block- or a MeshCollider, which follows the outline of the way the object is drawn most precisely. But it also uses the most processing power, so the Capsule Collider is a good compromise in this case.

Run the project in Play Mode, place a tree (by pressing 1, 2, 3 or 4) and check if the tree indeed gets a capsule collider and an Object Manipulator script attached (scroll all the way down in the Inspector



```
Main
  DefaultCursor
  Directional Light
  HologramCollection
    Tree9_2(Clone)
  FloorQuad(Clone)
DontDestroyOnLoad
  MixedRealityCameraParent
  InputManager
  Managers

Favorites
  All Materials
  All Models
  All Prefabs
Assets
  App
    Scenes
    Scripts
  Day-Night Skybo
  HoloToolkit
  HoloToolkitExtens

Assets ▸ App ▸ Scripts
  CommandMessage
  CommandType
  ObjectCreator
  ObjectManipulator
  SelectedMessage
  SpeechCommandHandler
```

```
Trans. View Dep.       0.7
Alpha Cutoff           0.3
Shadow Strength        0.8
Shadow Offset          1
Shadow Caster Res.     One-8th Res

Mesh (Mesh Filter)
Mesh                   Mesh

Mesh Renderer
Lighting
  Light Probes         Blend Probes
  Reflection Probes    Blend Probes
  Anchor Override      None (Transform)
  Cast Shadows         On
  Receive Shadows      ✓
  Motion Vectors       Per Object Motion
  Lightmap Static      ☐
  To enable generation of lightmaps for this Mesh Renderer, please enable the
  'Lightmap Static' property.
Materials
Dynamic Occluded       ✓

Capsule Collider
Is Trigger             ☐
Material               None (Physic Material)
Center                 X -0.6987396   Y 11.42931   Z -4.748301
Radius                 9.311927
Height                 22.85863
Direction              Y-Axis

Object Manipulator (Script)
Script                 ObjectManipulator
```

Exit play mode.

Now find the "InputClicked" method in ObjectManipulator.

- Put a breakpoint on the Debug.Log statement
- Run the Visual Studio Solution ('attach to Unity')
- Start play mode again
- Place a tree again
- Using the WASD keys and/or dragging the mouse with the right button, place the gaze cursor on the tree
- Press and hold space bar, then left click the mouse – effectively doing an air tap.
- Your break point on Debug.Log should be hit.
- EXIT PLAYMODE

Challenge (part 1)

- In the InputClicked Method,
  - send a SelectedMessage message with the current gameobject as payload.
- In SelectedMessageHandler
  - Set _isSelected to true if the message's game object is equal to the current game object, and false if it is not.
  - Set _activeManipulationCommand to None
- Enter play mode again and
  - Place a tree again
  - Select it
  - Press the key you assigned to "Move" or say "Move"
  - Press and hold spacebar, press and hold right mouse key
  - Move the mouse. If the tree moves with you, you have completed part 1

Now you might have noticed it's a bit hard to determine whether you have selected a tree successfully. It would be nice if you *also* had a selection confirmation sound. And that is part of the next challenge.

Challenge (part 2)

- Exit Play mode if you have not done so already
- In ObjectCreator:
    - Add a serializable private field to ObjectCreator of type AudioClip
    - In PlaceObject, add an AudioSource component to the created tree object
    -
    - Set the clip to the value of your private field
- In ObjectManipulator, make sure the method TryPlaySound is called when an object is selected
- In the editor, drag an audio clip on your newly created field in ObjectCreator. You can find a "Click" sound in HolotoolkitExtension.

If you completed this challenge correctly, you should here a low "thump" sound whenever you 'airtap' a tree. Try this in the Unity editor. If you feel like, you may also deploy this to your computer and try it in a head set.

So, the way this works:

- ObjectManipulator implements IManipulationHandler
- Every time you press and hold the object, OnManipulationUpdated is called. For every tree, by the way
- Only the selected tree will handle the event
- The transform.postion of the three will move 1/30th of the distance the controller (or the gesture) moved.

IManipulationHandler has tree more mandatory methods but we don't use them here.

Part 7 – adding some controller love

In this part you will learn:

- Intercept controller buttons and pass that along to the app

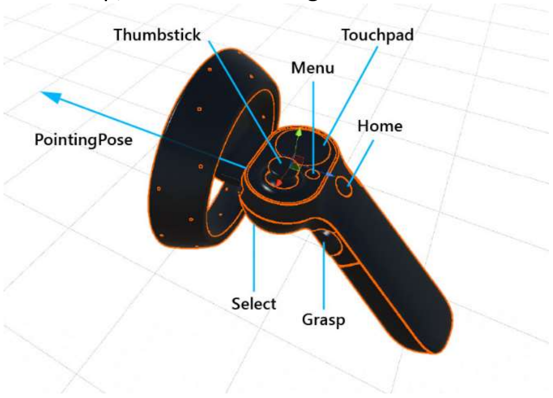
So far, we have done only things that would perfectly well in a HoloLens. That's the beauty Windows Mixed Reality and the current state of the Mixed Reality toolkit. But controllers are an Immersive Headset-only kind of thing. We are now going to 'launch' commands from the controller.

In Resources\Part7 you find ControllerInputHandler. Put it in the App Scripts folder. Then, in Unity, drag it from your assets into the Managers object. And then it already works – for moving. But this you can only test from inside a Mixed Reality app, not in Unity.

So

- Build the UWP app
- Deploy that to your computer
- Enable the headset in the device manager
- Run the app.

Now, you should be able to place the tree. Select it, and then you can move it by holding both Select and Grasp, and then moving the controller around. Make sure the cursor stays on the object.



This works as follows:

- The ControllerInputHandler implements IInputHandler
- Whenever you press a controller button, it's OnInputDown method is called
- And then you can move stuff.

So now you are basically doing moving without having to say "move" first.

Notice that don't get any "pringg" sounds when you select for Move this way. This has the following reason: in SpeechCommandHandler you are sending a message CommandMessage with a constructor that has the command and and "isSilent" parameter. Is SpeechCommandHandler that is set to false. But in the ControllerInputHandler it is not, and thus true. The PlayCormationSound in ObjectManipulator then checks if the command is 'silent' and only if it is not, it will play the sound

Basically, we now have the complete app.

Part 8 – Adding rotation, scale, and delete

Basically, this whole part is a challenge. You will

- Implement rotate, scale and delete to the ObjectManipulator
- Add code to the ControllerInputHandler to rotate and scale the object using controller buttons.

ObjectManipulator

- Add code to ObjectManipulator, in the case statement in OnManipulationUpdated. For Rotate you will need to work with transform.RotateAround and rotate around the 'up' direction.
- For scale, you will have to change transform.localScale

In both cases, experiment how much you will have to multiply or divide the value of the movement of the controller to get a workable gesture

ControllerInputHandler

This is simpler:

- Make sure Rotate is sent when the Menu button in held
- Make sure Scale is sent when the Touchpad is pressed
- Also make sure to send None is sent when either of these three buttons is *released*.

Part 9 – some fit & finish

In this part you will learn

- How to set a sky box
- What a rigid body is and what it's useful for

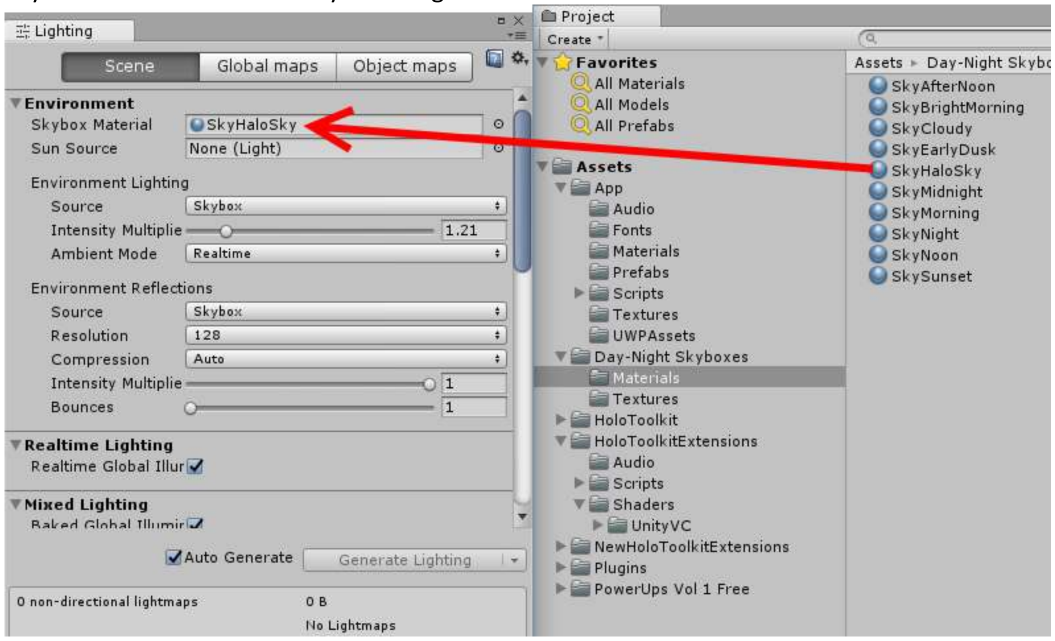
Sky Box

A Mixed Reality app for Immersive headsets is, in effect, a virtual reality app. To make the user feel comfortable it should have a kind of floor and some kind surrounding that gives visual clues when you turn – so, not uniformly colors. For the first type, the MRTK delivers a fixed blue 'floor' plane of about the size you can walk without yanking the headset's cable out of your computer. The surroundings are defined by a Sky Box, usually some blueish air with wisps of cloud in it – not a uniform blue color as this is disorientating to the users.

One of the things that apparently says "amateur stuff" in an app is using the default Unity Skybox.

Changing the default Sky Box:

- Select Windows/Lighting/Settings
- Select a sky box material from Day-Night-SkyBoxes/Materials and drag that on top of the SkyBox Material. Pick one to your liking



Environment
Skybox Material: SkyHaloSky
Sun Source: None (Light)
Environment Lighting
Source: Skybox
Intensity Multiplie: 1.21
Ambient Mode: Realtime
Environment Reflections
Source: Skybox
Resolution: 128
Compression: Auto
Intensity Multiplie: 1
Bounces: 1
Realtime Lighting
Realtime Global Illur ✓
Mixed Lighting
Baked Global Illumir ✓
✓ Auto Generate
0 non-directional lightmaps 0 B
No Lightmaps

Favorites
All Materials
All Models
All Prefabs
Assets
App
Audio
Fonts
Materials
Prefabs
Scripts
Textures
UWPAssets
Day-Night Skyboxes
Materials
Textures
HoloToolkit
HoloToolkitExtensions
Audio
Scripts
Shaders
UnityVC
NewHoloToolkitExtensions
Plugins
PowerUps Vol 1 Free

Assets ▸ Day-Night Skybo
SkyAfterNoon
SkyBrightMorning
SkyCloudy
SkyEarlyDusk
SkyHaloSky
SkyMidnight
SkyMorning
SkyNight
SkyNoon
SkySunset

Rigid Bodies

Rigid bodies are components that give game objects 'physical' properties. You might have noticed that you can easily create multiple trees in one spot. We can fix that.

- Revisit ObjectCreator
- Add this line:
  `var rigidBody = newObj.GetComponent<Rigidbody>();`
- Then place a tree, preferably a bit above the ground.
- What do you notice?
- Add another line
- `rigidBody.useGravity = false;`
- Try to place two trees in the same spot.

The End