## Development Environment

### Repository

<https://github.com/WearableComputerLab/HololensSatelliteProject>

### Required tools

#### Git

#### Unity 2022.3.21f1

* + If building to HoloLens, install with the ‘*Universal Windows Platform Build Support*’ module
  + If building to Quest, install with the ‘*Android Build Support*’ module and its submodules (‘*OpenJDK*’ and ‘*Android SDK & NDK Tools*’)

This information was sourced from the below MRTK3 wiki article and is relevant as of 19/11/2024:

<https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk3-overview/getting-started/setting-up/setup-dev-env>

#### Visual Studio 2022 with the following workloads

* + ‘.*NET desktop development*’
  + ‘*Desktop development with C++*’
  + ‘*Game development with Unity*’
  + ‘*Windows application development*’ with the following optional features
    - ‘*C++ (v143) Universal Windows Platform tools*’
    - ‘*Windows 10/11 SDK*’ (choose Windows 10 or 11 depending on your OS. Each has multiple versions though specific version appears to be irrelevant)
    - ‘*USB Device Connectivity*’ (if deploying/debugging over USB, we didn’t install or use this)

This information was adapted from the MRTK3 wiki source below and is relevant as of 19/11/2024. NOTE: The article mentions to install the Universal Windows Platform (UWP) development workload, however this has been phased out in VS 2022 and replaced with the Windows application development workload.

<https://learn.microsoft.com/en-us/windows/mixed-reality/develop/install-the-tools?tabs=unity>

#### Blender (required if using .blend models else Unity cannot read .blend files)

## High level Overview

### Project files

The two main folders in the Unity Assets directory are:

* Assets/Resources, which contains:
  + /Models, where model files are placed to be imported into StellAR
  + /Prefabs, where the automatically generated model prefabs are stored and used to populate the Models hand menu list
  + /VuforiaConfiguration.asset, which contains the settings relevant to Vuforia (the App Licence Key is currently set to one generated in Marcello Morena’s personal Vuforia account, but this can be safely changed)
* Assets/ProjectFiles, which contains:
  + /Materials, where custom materials used by StellAR are stored
    - /OutlineMaterials, which contains the materials used for outlining selected objects (different colours distinguish users in a networked session)
    - /UIButtonMaterials, which are the materials used for the different coloured highlighting buttons on the Annotation pane
  + /Scenes, which contains the scenes. This currently only contains the StellAR scene
  + /Scripts, which contains all the custom scripts created for StellAR
  + /SystemPrefabs, which contains various prefabs used in the StellAR scene

### Model importing

Model files (either .blend or .fbx) are placed in Assets/Resources/Models. ModelPrefabGenerator (Assets/ProjectFiles/Scripts/Importing) is an editor script that runs each time Unity reloads (e.g. on startup, when files change, etc.). This script checks if models in the Models folder have associated prefabs in Assets/Resources/Prefabs, and if not it creates a prefab from the model file. Models are associated with prefabs via name therefore each model must have a unique file name.

#### Importing a new model

**Drag and drop** the model file into Assets/Resources/Models in the Unity editor. If you place the file in the directory on your OS file system, it won’t generate a prefab until you right click the model file in Unity and **reimport it**.

**Ensure that you tick Read/Write** permissions on the model in the Unity inspector (under Meshes) or the model won’t be interactable.

ModelPrefabGenerator will create the required prefab in Assets/Resources/Prefabs. Models are linked to prefabs via name, so **ensure that all models have a unique name** (i.e. if you have two different file types with the same name only one prefab will be generated).

#### Updating a model

ModelPrefabGenerator only creates prefabs if they don’t exist in the Prefabs folder. If you update a model file in the Models directory, the object in StellAR won’t update until you **delete the prefab** in the Prefabs directory so it can be regenerated to include the new model changes. Once the prefab is deleted, right click on the model you have changed and **reimport it** to generate a new prefab.

#### Deleting a model

When removing a model from StellAR, just delete the prefab and model files from the Models and Prefabs directories.

## Explanation of Managers and GameObjects in StellAR Scene

* \_\_/MRTK contains MRTK prefabs
  + MRTKInputSimulator contains MRTK’s code for simulating AR hand input on PC (in the editor play mode or in the Windows build of StellAR)
* \_\_/Network contains managers that specifically relate to running networked sessions
  + \_MBInstanceManager manages finding message based interactables and synchronising their transforms
  + \_ExplodableManager handles exploding and collapsing message based interactables over the network
  + \_InteractableLockHandler handles the locking and unlocking of message based interactables so users can’t interact with them during explosions
  + \_ConnectionManager facilitates hosting, joining, and disconnecting online sessions
  + \_ClientManager manages user selection outlines and each user’s distinct identifying colour
* \_\_/System contains managers that are used in both local and networked sessions but primarily local
  + \_ApplicationManager manages the order that startup processes occur, either first in startupProcess or second in postStartupProcess. Additional events can be to this class added to add further granularity to the startup order of events
  + \_PrefabManager handles spawning interactables in both offline and online sessions
  + \_SelectionManager manages the currently selected interactable and alerts subscribers when a new selection is made
  + \_InteractableFactory attaches the required scripts when a model is spawned
  + \_MetadataManager loads metadata from disk and attaches it to models. If a metadata file doesn’t exist for a model, it creates a blank json file on disk
  + \_AnnotationManager loads annotation data from disk, creates new annotations during runtime, and writes new annotations to disk
  + \_HighlightManager is used to set the highlight colour of interactables
  + \_AudioLoader loads audio files from disk into the runtime
  + \_VuforiaManager handles centring the world space in networked sessions. It spawns a NetworkOriginObject that tracks the position of the Vuforia image target, and all spawned network models are children of this to stay centred in the shared network world space
* \_NGO\_NW\_Manager is the gameobject that contains the Unity NGO Network Manager script. This should be a child of \_\_/Network but NGO required Network Manager to be attached to a root gameobject
* \_\_/UI contains UI elements
  + \_HandManager
    - The hand menu that comes up when a user turns their palm up.
    - This is how the user interacts with StellAR to spawn models, explode a model, etc.
    - The Animations and Global View buttons do not have implemented functionality
  + \_DebugConsole
    - A window that shows debug messages and errors in the game world. This is used during development to see errors in deployed versions of StellAR
    - When developing, instead of using the traditional Unity Debug.LogDebug() use DebugConsole.Instance.LogDebug() to show messages on the debug console. These messages are then forwarded to Debug.LogDebug() anyway
    - This can be enabled/disabled with the Enable Console bool in the editor. When disabled,
    - This panel follow’s the user’s view but can be locked in place with the pin icon at the top right
  + \_DataPanelRoot
    - The UI for showing and interacting with Annotations and Metadata
    - This panel follows the user’s view but can be locked in place with the pin button at the top right
  + \_AnnotationUIGenerator
    - Handles creating and storing annotation UI elements that are shown in \_DataPanelRoot
* \_ImageTarget is the Vuforia image that is used by \_VuforiaManager to centre the world space in networked sessions.

### Scripts attached to models

When a model is spawned at runtime from the Import Model hand menu list, multiple components are added to it and its children via the InteractableFactory (which adds components depending on if the spawned model needs to be:

* Metadata Component
  + This component is the only component not added at runtime since it is already a part of the prefab generated by ModelPrefabGenetaror when a model file is placed in the Models directory. However, it is modified at runtime to populate the Metadata string with information from the metadata file
* Interactable (either LocalBasedInteractable or MessageBasedInteractable)
  + A façade encapsulating the behaviours of the Explodable and Selectable scripts
* Explodable (either LocalBasedExplodable or MessageBasedExplodable)
  + This component handles the explosion and collapsing functionalities.
* Selectable (either LocalBasedSelectable or MessageBasedSelectable)
  + Enables an interactable to be marked as the current selection
* Object Manipulator (either ExtendableObjectManipulator or MessageBasedManipulator)
  + An MRTK3 script that handles the implantation of grabbing objects in AR
  + This also adds the ConstraintManager component
* Mesh Collider
  + This creates the collision bounds on an object which is used by Object Manipulator to know when a user has touched the object

## Project Notes

* If you need more information on the code process of how some features work look at the sequence and class diagrams in the Documentation folder of the repository
* Take a look at the **GlobalConstants** class in Assets/ProjectFiles/Scripts/Utility to see common variables that are used throughout the project
* Take a look at the **interfaces** in Assets/ProjectFiles/Scripts/Utility for performing common functionalities, such as listening for when a new object is selected by the user
* StellAR uses coroutines for some asynchronous tasks, this is because Unity objects cannot be used outside of Unity’s main thread
* For networking communication Unity’s Netcode for GameObjects (NGO) is used, specifically their Remote Procedure Calls (RPCs) and their named custom messaging system (see <https://docs-multiplayer.unity3d.com/netcode/1.10.0/advanced-topics/message-system/rpc/> and <https://docs-multiplayer.unity3d.com/netcode/1.10.0/advanced-topics/message-system/custom-messages/#named-messages>)
* Models in networked sessions are instantiated on each local device using custom messaging, and each frame when a user translates an interactable the new Transform information is sent to all clients to update on their end. This is all done using custom messaging, and NGO’s network object and network transform scripts are not used
* Vuforia Engine is used in StellAR networked sessions as a means to centralise the spawn locations of models across devices. An empty root game object is instantiated on the Vuforia image target’s location on each user’s local device, and whenever a model is spawned it is spawned at that position
* For information on how to deploy StellAR to HoloLens, and where metadata and annotation data is stored, see the repository’s README
* Microsoft’s MRTK3 is used in the project, for more information on MRTK3 please see <https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk3-overview/>

## Glossary

* **Message based**: Synonymous in our code to refer to something that is networked. This is because we implemented a lot of networked features ourselves using network messages. So, when we refer to a ‘message based selectable’ script as opposed to a ‘local selectable’, we mean that the script is used for message based objects i.e. networked objects.
* **Ownership**: The authority for a user to interact with an interactable. This is given to users when they select an object and is held while they use it, then relinquished when they let go of the object.
* **Lock**: A server-wide restriction preventing anyone from gaining ownership while the lock is in place. The server locks an object during the process of an explosion/collapse and removes the lock when the process is complete.
* **Interactable**: An object that can be exploded and selected.