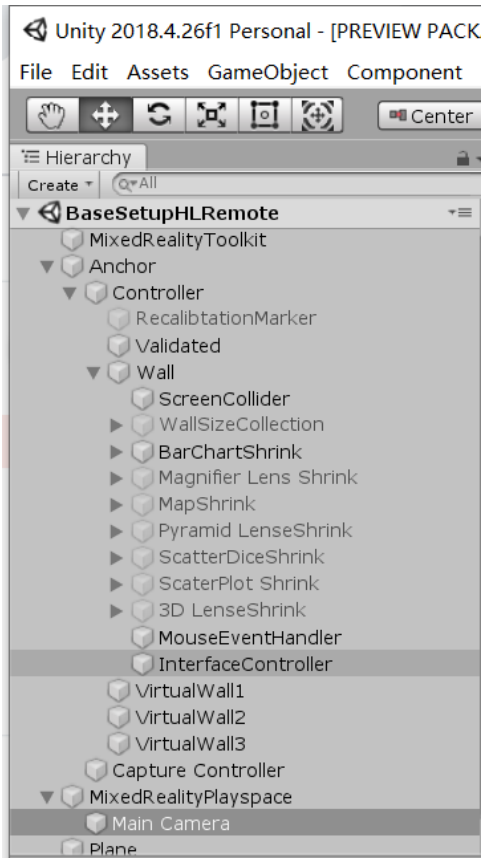
This project is done in 2020 from March to September. The project includes two unity projects. One for the monitor application and remote room(With the name desktop). The other is for the HoloLens application(With the name HoloLens project).

HoloLens application

Architecture:

In this architecture, the *Virtual Walls* represent the size of the room of wilder(the room for the wall display.) The *Wall* object contains everything related to the virtual visualization that is synchronized with the HoloLens and monitor display. Thus, use transform.localPosition as much as possible. The content inside the *Wall* object is really similar to the one in Remote room. The *MouseEventHandler* object manages mouse event sending from the monitor application. The *InterfaceController* object controls the showing and hiding of visualizations.



The *Capture controller* controls the script that supports video recording from the camera.

The *WallSizeCollection (obsolete)* contains all the wall size visualizations developed previously.

Important GameOjects

1.Anchor>Controller>Wall>InterfaceController

- PhotonView.cs : sending message in each “Photon Frame”. (From photon Network)
- PhotonviewController.cs :
  1. Controll the visualization examples in the application. The public variable “visualizationName” controls the first visualization shown when the application starts. The value of this variable should be exactly the same as the components you added into the visualization List.
  2. Add PhotonSynChroManager class attached to each visualization to the PhotonView

Observed components. Each visualization contains a PhotonSynChroManager class component. This component is responsible for synchronizing the position of visualization elements with the ones in the HoloLens application.

- Eventhandler.cs: for receiving the size data sending from the monitor application.

2.Anchor>Controller>Wall> MouseEventHandler

- MouseEventCoreService.cs: this is a Singleton, It responsible for receiving interaction events (Mouse click events, UI button events, Keyboard events) from the monitor application via Network.

3.Mouse event related

All to mouse interaction related script is inside the folder : Assets> HoloLens> Scripts> HoloLensSynchronize> MouseInteractionToolkits. All the scripts handling UI events are subscribed to different unity <Action> . Once the event id received from Network matches the id in the scripts, it will trigger the interaction event. To use the scripts, simple attach the script to the object if manipulation or an empty object if UI.

One thing that needs to be noticed is that when using the *MouseManipulationhandler.cs* script, you have to attach a *MouseInteractable.cs* to the same object as well.

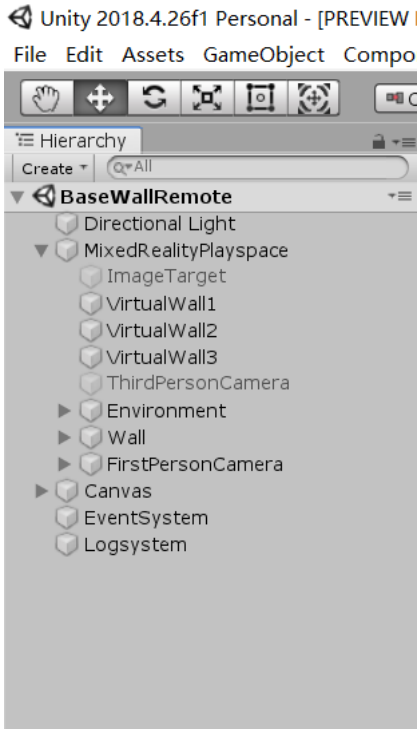
4.First Person View Camera

The first person view camera is synchronized with the camera inside HoloLens. For details, please check "*TransformSynchronizeCameraContoller.cs*". When debugging inside unity, try to disable the script, otherwise it will try to synchronize itself to location (0,0,0).

Remote room

Architecture:

The architecture of this application is similar to the HoloLens one. The *Enviornment* object contains a virtual construture of the monitor and the table, all the office related prefabs are inside the *Background prototyping models folder.* The *Canvas* object contains all the UI element of the scene, and the *logsystem* is responsible for log event. The Wall object should contain the same visualizations example as the HoloLens application, with the same order.



Important GameOjects

1.MixedRealityPlayspace>Wall>InterfaceController

- **There is a equivalent object in the HoloLens application as well.**
- PhotonView.cs : sending message in each “Photon Frame”. (From photon Network)
- PhotonviewController.cs :
  3. Controll the visualization examples in the application. The public variable “visualizationName” controls the first visualization shown when the application starts. The value of this variable should be exactly the same as the components you added into the visualization List.
  4. Add PhotonSynChroManager class attached to each visualization to the PhotonView Observed components. Each visualization contains a PhotonSynChroManager class component. This component is responsible for synchronizing the position of visualization elements with the ones in the HoloLens application.
- Generaleventhandler.cs: for receiving the size data sending from the monitor application.

2.MixedRealityPlayspace>Wall>visualizations(BarChartShrink, Magnifier Lens Shrink, MapShrink, Pyramid

LenseShrink, ScatterDiceShrink, ScaterPlot Shrink, 3D LenseShrink)

Each of the visualization has a BehaviorController GameObject which controls Network logic including position synchronization and sending/receiving the event message to other applications.

The BehaviorController object contains a *PhotonSynChroManager.cs* script responses for Synchronize the gameobject. To use it, simply grab the gameobject you want to synchronize and refer it to the *Synchronize Objs* list. You will need to do the same thing in the HoloLens applications, and the order of the gameobjects should be mapping each other.
You can also use script to add synchronizable gameobject at run time. One example can be found at "*MixedRealityPlayspace >Wall > BarChartShrink >BehaviorCotroller > SynChronizeAssignment.cs*".

Most of the visualizations contains a component called Event handler.cs or end with the name EventHandler. This is responsible receiving and sending events using PhotonNetwork class.(Notice that the Event handlers are using different namespace, they are not the same script.)

3.UI Logging system

A scroll view UI is implemented in the bottom left corner in order to display events from the HoloLens application or the monitor application. Current example only appears in "Pyramid LenseShrink" visualization. You can find a example at "*MixedRealityPlayspace >Wall > Pyramid LenseShrink >BehaviorCotroller > EventHandler.cs*".

4.First Person View Camera

The first person view camera is synchronized with the camera inside HoloLens. For details, please check "*TransformSynchronizeCameraContoller.cs*"

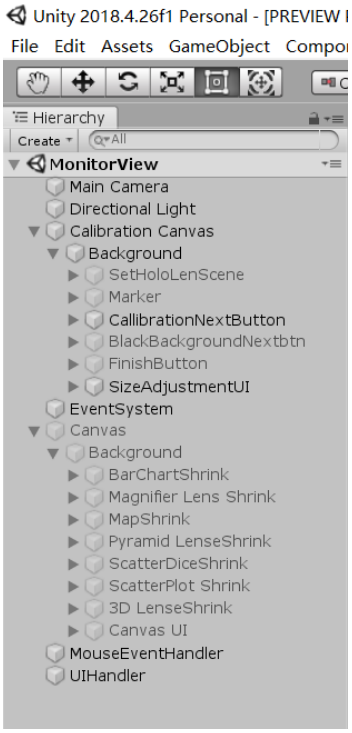
Monitor application

Architecture:

The Monitor application is mainly consist of two canvas, one is the calibration Canvas, another one is the visualization canvas.

The MouseEventHandler handles both the mouse interaction network and also the UI network interaction.

The UIHandler handles switch between different viualizations.



Important GameOjects

1.MouseEvent Handler

- MouseEventHandler.cs : this script responsible for handling the mouse double click, mouse dragging and mouse right click function, and send the event via network to the HoloLens application.
- UIEventHandler.cs : this script responsible for handling UI remote event sending to the HoloLens application. For each time the visualization changes, the `RegisterEvent()` function needs to be called to register the `uIEventInteractable` component to the uIEventInteractable list.
- For each of the UI elements that needs to be connected to the HoloLens network, a uIEventInteractable.cs component is required. In this component, user can set up the UI type, the UIid(which needs to be the same as the one in HoloLens).

2.UIEvent Handler

- MonitorUIHandler.cs : this script controls the rescale of the Marker and store the scale value into a singleton.
- VisualizationController.cs : this script responsible for managing the switch of different visualizations.
- EventHandler.cs: this script is responsible for handling event for sending to the HoloLens and remote room.