The MotionMonitor xGen Hardware Guide: Virtual Reality with Unity

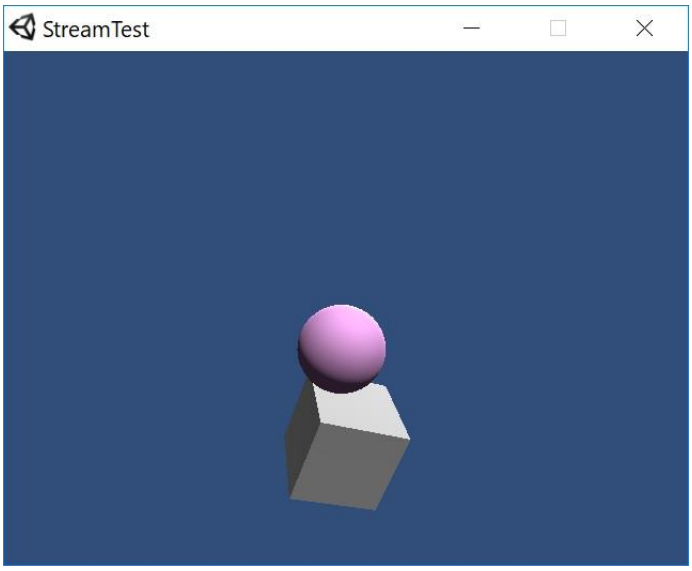
This document describes the files that will demonstrate how to exchange data between The MotionMonitor xGen and Unity worlds. You should have the following files: UnityTMMxGenDataExchange.iws, Stand Alone Projects with UnityTMMxGenDataExchange executables and UnityXGenDataExchange.zip. Please contact your Client Support Engineer or support@TheMotionMonitor.com for any of these files.

Note: This document is not intended to be a manual or to be used in place of manuals or training materials for Unity and The MotionMonitor xGen.

First, to demonstrate the data exchange, run the UnityTMMxGenDataExchange.exe. If prompted by your Firewall, allow access to any networks.

Launch The MotionMonitor xGen software, select an existing user or create a new one and Open the Workspace, UnityTMMxGenDataExchange.iws. This workspace can be used to demonstrate the communication between The MotionMonitor xGen and Unity without any hardware. Click on the Activate/Deactivate Hardware icon to initiate the communication between The MotionMonitor xGen and the Unity standalone scene. Click OK on the “Activation Successful!” message.

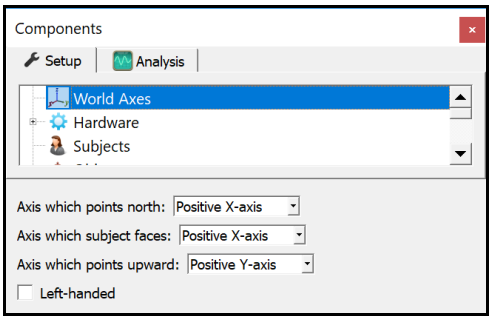
At this point you should see two objects, a gray cube and a pink sphere, moving together vertically within your Unity standalone scene. The cube should also be rotating. Click the Activate/Deactivate Hardware icon again to terminate the connection with Unity and close the Unity application.



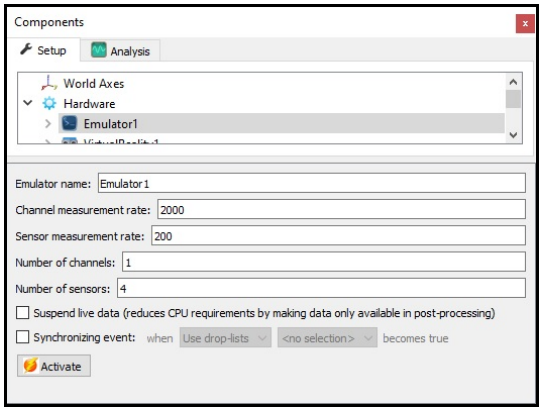
The following steps will describe how The MotionMonitor xGen Workspace and Unity Project were configured to create this scene.

Within The MotionMonitor xGen Setup Components window the following settings have been configured:

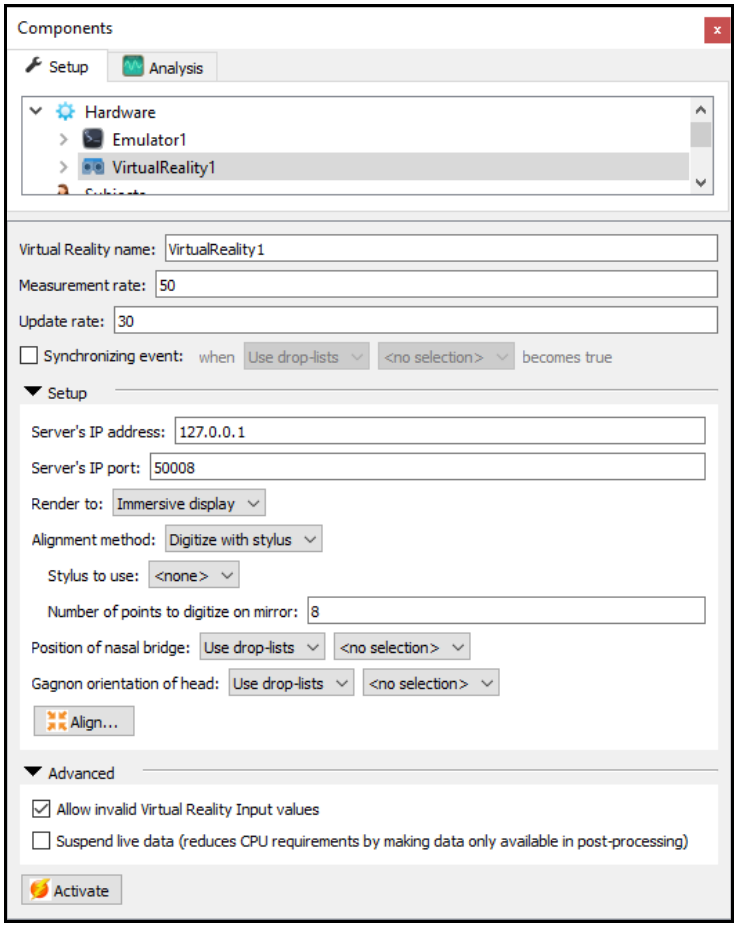
World Axes - Configuration of the World Axes layout for The MotionMonitor xGen



Hardware Emulator - The emulator is a virtual hardware component that generates data without any physical hardware. This device can be removed in place of actual hardware connected to the system.

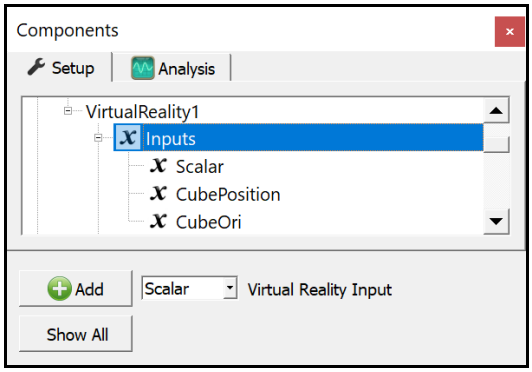


VirtualReality - Virtual Reality is the hardware component that controls the bi-directional communication between The MotionMonitor xGen and Unity or WorldViz virtual environments. The Server's IP address is either that of the local computer (127.0.0.1) or of the server computer running the Unity application. The IP port should not change. The Measurement rate is the rate at which Unity Sends data to The MotionMonitor xGen and the Update rate is the rate at which The MotionMonitor xGen outputs data to Unity. Setting the measurement rate too high can cause data to become unsynchronized or result in communication errors. This value may need to be lowered if encountering communication errors.



Enable the input of invalid values using the check box. Finally, select the medium for which the VR will be rendered to, either immersive display, Tobii HMD or standard HMD.

Inputs - Inputs are the variables that are sent to Unity. Their names need to match that of the variables defined within the Unity script, MainCode for our demo, located within the Unity Assets directory. These Inputs can be graphed and used in analyses within The MotionMonitor xGen. Caution should be taken to use variables to control the position and orientation of items in the Unity environment because The MotionMonitor xGen is not automatically accounting for the Unity world axes layout, as it does with VR Objects.

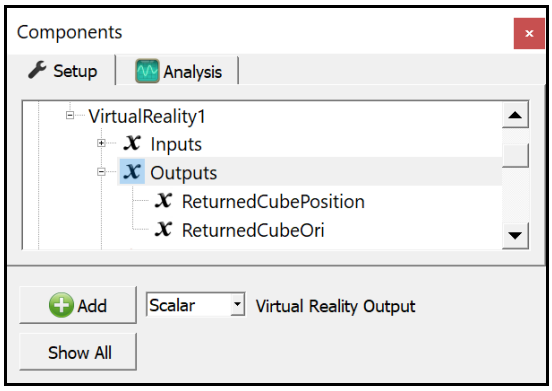


Scalar - Is a scalar variable sent to Unity.

CubePosition - Is a vector variable sent to Unity whose value is used to control the position of a cube.

CubeOri - Is a rotation variable sent to Unity whose value is used to control the orientation of a cube.

Outputs - Outputs are variables that are received from Unity. Their name needs to match that of the variable defined within the Unity script, MainCode, located within the Unity Assets directory. These outputs can be graphed and used in analyses within The MotionMonitor xGen.

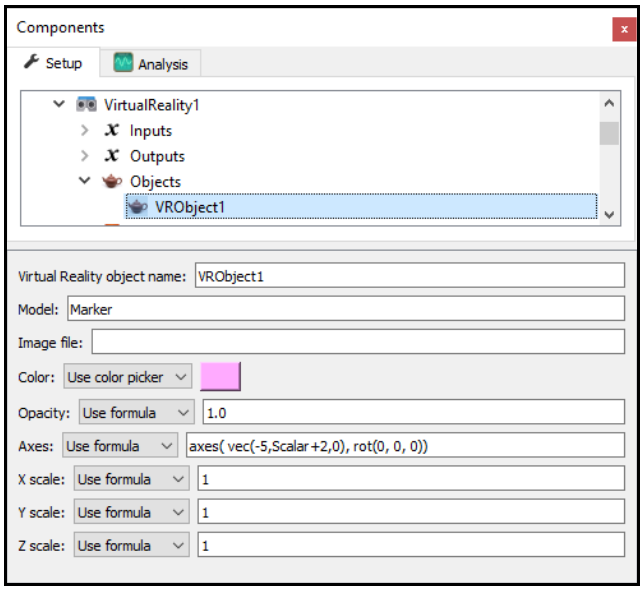


ReturnedCubePosition – Is a vector variable received from Unity.

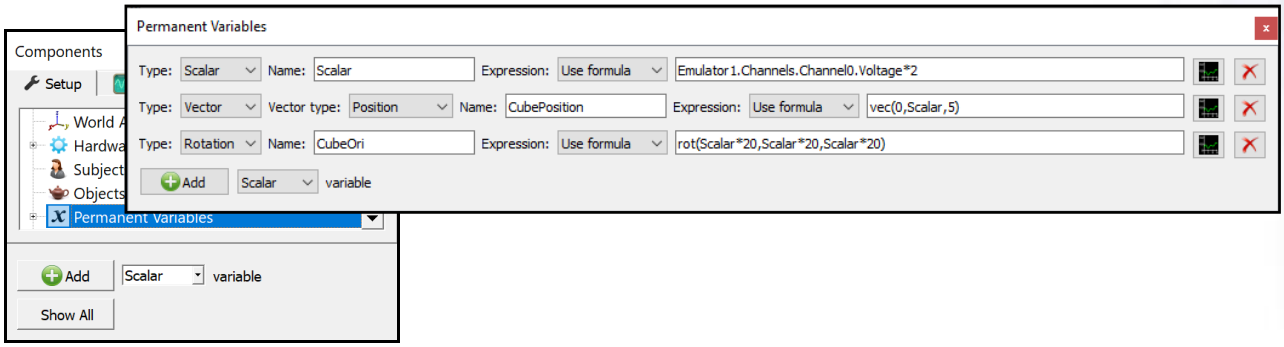
ReturnedCubeOri – Is a rotation variable received from Unity.

Objects - Any objects defined here will automatically be populated within the Unity environment based on their defined parameters. This automatic population into the Unity scene is handled by the MMServer script located within the Unity Assets directory. The Model file specified here, typically an .obj file, needs to also be located within the Unity Assets\Resources directory. The same goes for if an image file is being used to display on the surface of an Object. Unity has a native axis system that is left-handed and where +Z is the forward-pointing axis and +Y is the upward-pointing axis. In order to keep the VR objects from appearing sideways or upside down, xGen automatically revises the position and orientation data for objects before sending, so that the upward axis in our world maps to the Y axis in the virtual world and the forward axis in our world maps to the Z axis in the virtual world.

VRObject1 - Is an object whose axes (position and rotation) are used to control an object within Unity. The model, in this case a Marker.obj file, also resides within the Unity Assets\Resources directory. Available settings that can be configured for the object include applying image files to the object's surface, specifying the color, opacity and scaling of the object. The scaling is applied to the file's native dimensions.



Permanent Variables - Permanent variables are variables that are typically used within Live experimental computations and cannot be modified after data are recorded. Each of these variables are currently receiving data from the Emulator hardware device, but could be modified to use data from any hardware device. These variables could have also been defined directly through the scalar, vector or rotation Virtual Reality Input definitions, bypassing the process of creating Permanent Varialbes. Caution should be taken to use variables to control the position and orientation of items in the Unity environment because The MotionMonitor xGen is not automatically accounting for the Unity world axes layout, as it does with VR Objects.



Scalar - This is the scalar variable that's being used for the "Scalar" Virtual Reality Input.

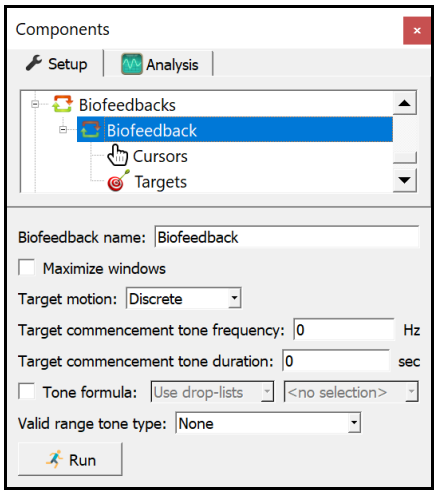
CubePosition - This is the vector variable that's being used for the "CubePosition" Virtual Reality Input.

CubeOri - This is the rotation variable that's being used for the CubeOri Virtual Reality Input.

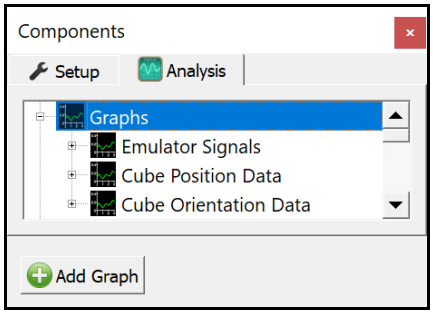
Biofeedbacks - Provides the ability to perform cursor-target paradigms. They provide data similar to VR Objects in that they are automatically populated in the Unity environment, if enabled under the Virtual Reality Hardware node. However, there are many ways in which they differ including that running the biofeedback initiates its own recording, they can be displayed in The MotionMonitor xGen animation window, have many more parameter settings, can provide auditory feedback, and their data can be accessed for analyses. Please refer to the Knowledge Base article for The MotionMonitor xGen Biofeedback to learn more about configuring and performing Biofeedback recordings.

Cursors - Are the chasing object.

Targets - Can be configured as Discrete or Randomized.



Graphs



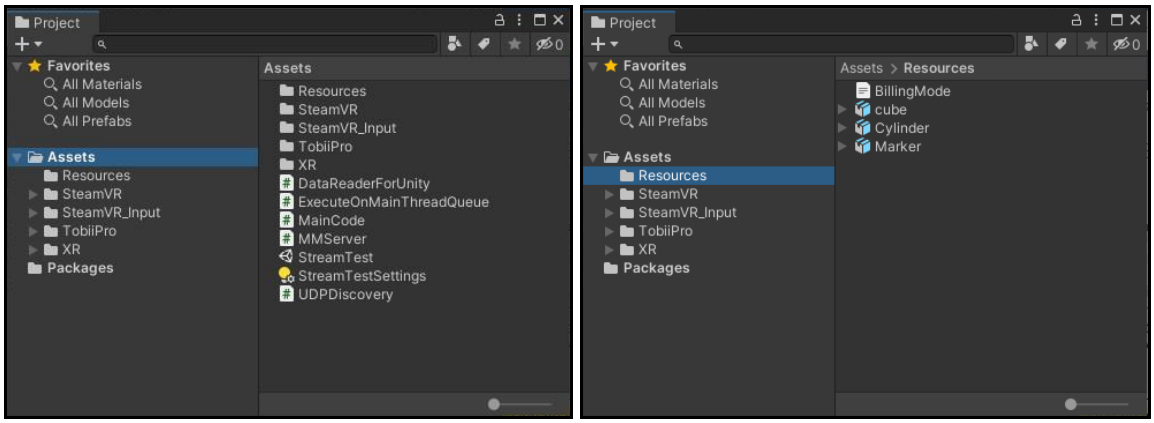
Emulator Signals - Includes graphs for the raw data being generated from the Emulator hardware device.

Cube Position Data – Includes plots for data from the CubePosition and ReturnedCubePosition vector Virtual Reality Inputs and Outputs

Cube Orientation Data – Includes plots for data from the CubeOri and ReturnedCubeOri rotation Virtual Reality Inputs and Outputs.

Launch the Unity application and create a new Project. Close the application and unzip the demo Unity Project directory, UnityXGenDataExchange.zip, into your new Unity Project directory. After initially creating the project, it can be quickly loaded through the Unity Hub application. The versatility of Unity provides the ability to customize your virtual worlds in many ways, but for the purpose of this demo, Unity can be broken up into the following windows:

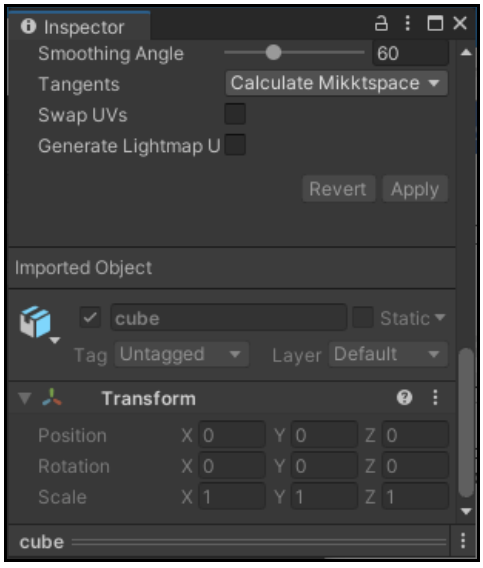
Project Window - Is where you can view and manage all of the assets for your Unity project.



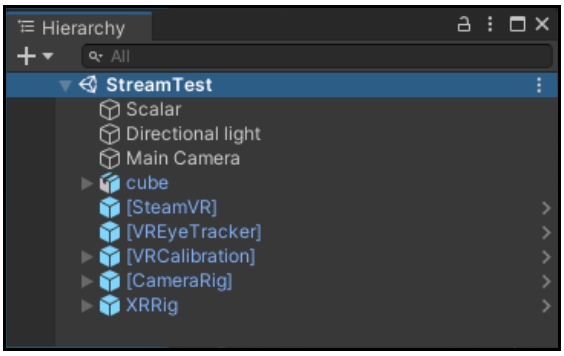
The Assets folder is where the 2 main scripts are located. The MainCode script is the base code for your Unity Project. This is where the Virtual Reality Inputs and Outputs that were discussed previously would be specified. The MMServer script is the script that communicates with The MotionMonitor xGen and where the Virtual Reality Objects and Biofeedback Cursors and Targets get automatically added to the Unity Project.

The Assets\Resources folder is where any model files (.obj files) for Virtual Reality Objects and Biofeedback Cursors and Targets need to be located, as well as any image files assigned to Virtual Reality Objects.

Inspector Window - Displays the parameters for any asset highlighted in the Project or Hierarchy Windows.

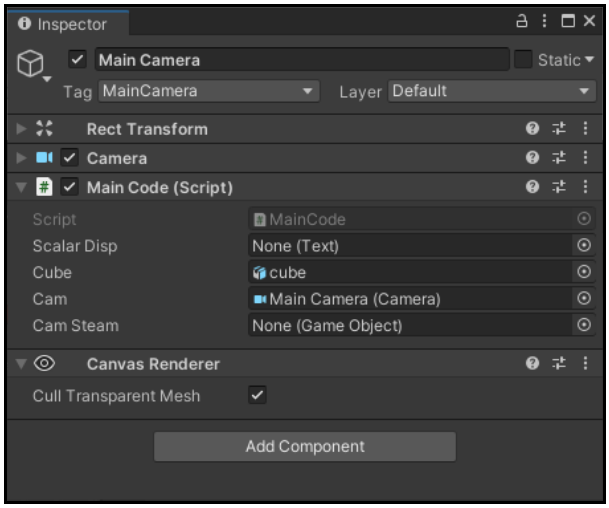


Hierarchy Window - List of all Objects in the scene. Virtual Reality Objects and Biofeedback Cursors and Targets will get populated once the Virtual Reality Hardware Component has been activated within The MotionMonitor xGen. Other objects such as Scalar and cube were dragged into the Hierarchy Window from the Assets Window or from right clicking within the Hierarchy Window.

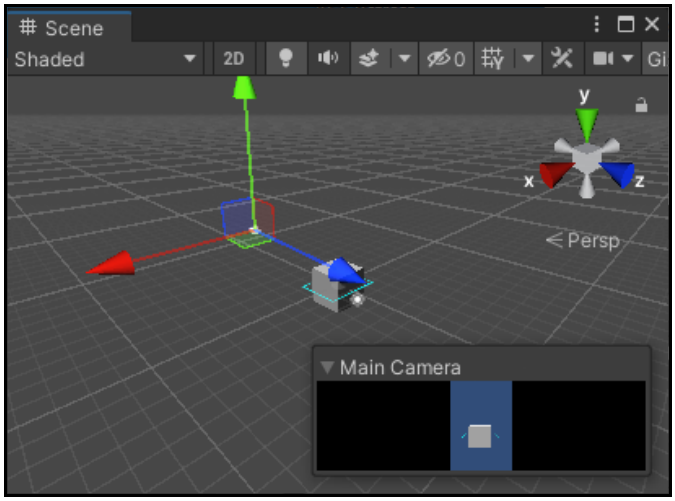


There is an item from the hierarchy that should be removed based on whether an HMD is used and it is desired to use the head compensation form the HMD or not. If the head compensation is desired, the [CameraRig] should be deleted. If no head compensation is desired, the XRRig should be deleted.

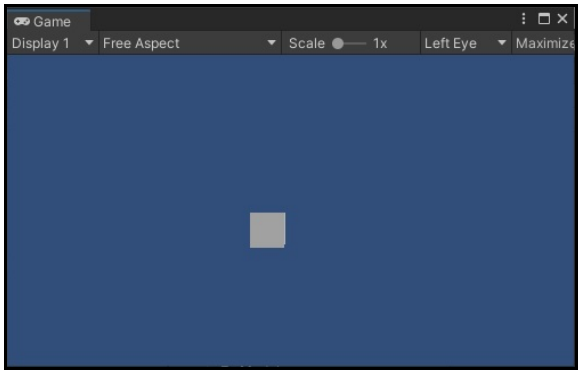
The cube Object will remain static unless linked to data from the MainCode script. This linking is accomplished by clicking on the Main Camera in the Hierarchy Window and going to the MainCode(Script) section within the Inspector Window. From here, we can click on the link to the right of the variables from our script and link them to items from our Hierarchy Window, the Cube data to the cube object. In our Demo, the Virtual Reality Input “CubePosition” and “CubeOri” are being used to control the position and rotation of the cube object in the Unity Game Window.



Scene Window - Contains a visualization for anything within your Hierarchy Window. When creating a new scene or project, the Scene is empty except for the Main Camera and Directional Light.



Game Window – Shows the Scene as the subject would see it. This is your unity "environment". The perspective for the Game Window is controlled by the Main Camera. The Main camera perspective can be modified by selecting Main Camera in the Hierarchy Window and physically moving it around in the Scene Window, where it will become highlighted, or through the Inspector Window in the Transform Component section. Alternatively, the same way that the cube Object from the Hierarchy Window was linked to data from the MainScript through the Main Camera Inspector Window, The Main Camera could receive data from the MainScript to control its position and orientation.



To test a Unity Project and its Scene, click the Play button at the top of the Unity application. The Scene and Game Windows should now be Live. From The MotionMontior xGen activate the Virtual Reality Hardware component by either clicking on the Activate/Deactivate Hardware icon, as before, or by clicking the Activate button from the Virtual Reality Hardware Components parameters pane, assuming that the emulator or other hardware are already activated.
When making modifications, first deactivate the Virtual Reality Hardware device within The MotionMonitor xGen and then click the Play button again within the Unity Project to stop the Scene from Playing.

Settings can be saved within The MotionMonitor xGen by clicking on the Save Workspace As icon or through the File|Save Workspace As menu item.

Settings can be saved within Unity by going to File|Save and File|Save Project.

When a Unity Project and Scene are ready, they should be Built to run standalone, similar to how we first tested the UnityTMMxGenDataExchange.exe application. Go to File|Build Settings to congigure the Player Settings and to perform a build for the first time or File|Build and Run.

The Unity application can now be run by launching the Unity.exe file that was just generated. The MotionMonitor xGen should then be ready to connect as we did initially with the UnityTMMxGenDataExchange.exe application.

General Notes:

- There are many resources available for purchasing or generating .obj files. Basic geometric shapes may already be available through Unity or can be created using 3rd party applications such as Ultimate Unwrap 3D Pro. Alternatively, files can also be purchased through vendors such as TurboSquid.

- This demo was developed using Unity Version 2020.3.30f1 (64-bit) and The MotionMonitor xGen version 3.55.07(3.55g). Please make sure that Unity Version 2020.3.30f1 (64-bit), Unity Hub, and Steam are all installed on your platform. Do not upgrade

Future Developments:

- Ability to stream scalar variables into Unity.

- Ability to optimize Unity clock for extended recordings.