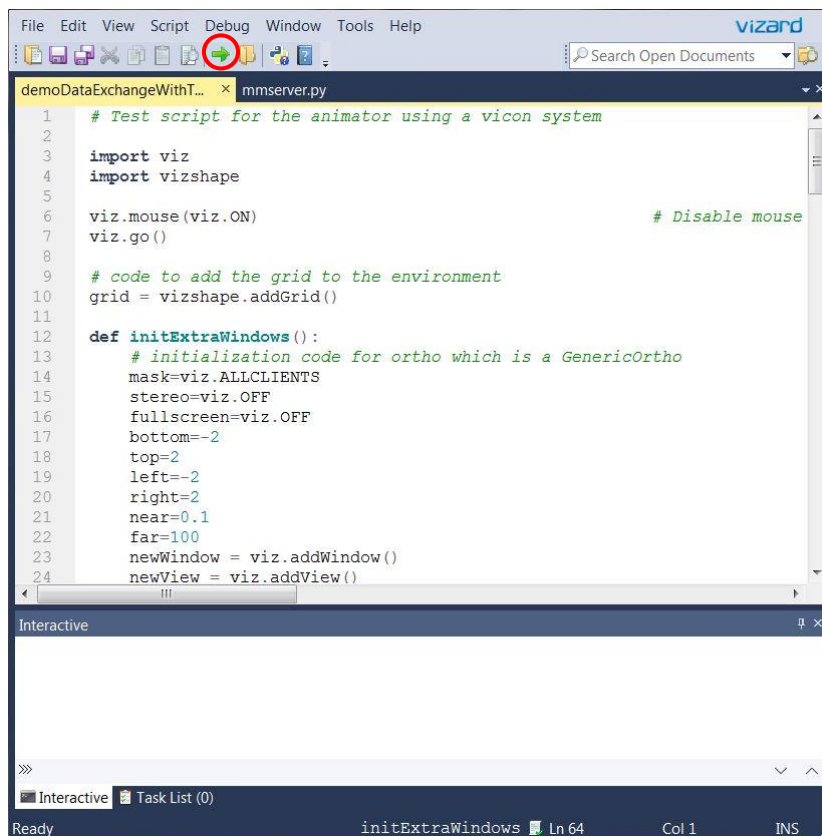


The MotionMonitor xGen Hardware Guide: Virtual Reality with Vizard

This document describes the files that will demonstrate how to exchange data between The MotionMonitor xGen and Vizard virtual worlds.

You should have the following files: VizardTMMxGenDataExchange.iws, and the VizardXGenDataExchangeVizardData.zip folder containing demoDataExchangeWithTMM.py, mmserver.py, and object files.

Please contact your Client Support Engineer or support@TheMotionMonitor.com for any of these files.



The screenshot shows the Vizard software interface. The main window displays a Python script named `mmserver.py` with the following code:

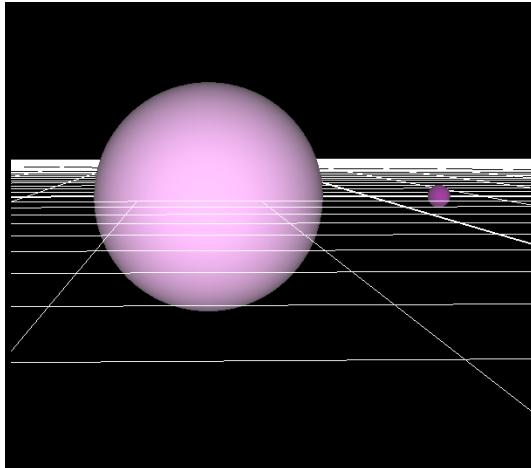
```
1 # Test script for the animator using a vicon system
2
3 import viz
4 import vizshape
5
6 viz.mouse(viz.ON) # Disable mouse
7 viz.go()
8
9 # code to add the grid to the environment
10 grid = vizshape.addGrid()
11
12 def initExtraWindows():
13     # initialization code for ortho which is a GenericOrtho
14     mask=viz.ALLCLIENTS
15     stereo=viz.OFF
16     fullscreen=viz.OFF
17     bottom=-2
18     top=2
19     left=-2
20     right=2
21     near=0.1
22     far=100
23     newWindow = viz.addWindow()
24     newView = viz.addView()
```

Below the script editor is an "Interactive" console window, which is currently empty. The status bar at the bottom indicates the current position is at line 64, column 1.

First, to demonstrate the data exchange, run Vizard and open the `demoDataExchangeWithTMM.py` and `mmserver.py` files. If prompted by your Firewall, allow access to any networks. Run the scripts using the green arrow in the toolbar.

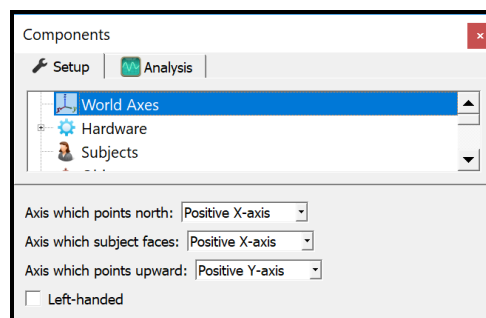
Launch The MotionMonitor xGen software, select an existing user or create a new one and Open the Workspace, `VizardTMMxGenDataExchange.iws`. This workspace can be used to demonstrate the communication between The MotionMonitor xGen and Vizard without any hardware. Click on the Activate/Deactivate Hardware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

At this point you should see two objects, a large pink sphere and a smaller purple sphere, moving together horizontally within your Vizard animation window. Click the Activate/Deactivate Hardware icon again to terminate the connection with Vizard.

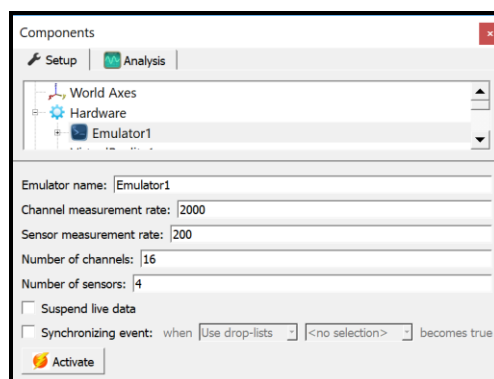


Within The MotionMonitor xGen Setup Components window the following settings have been configured:

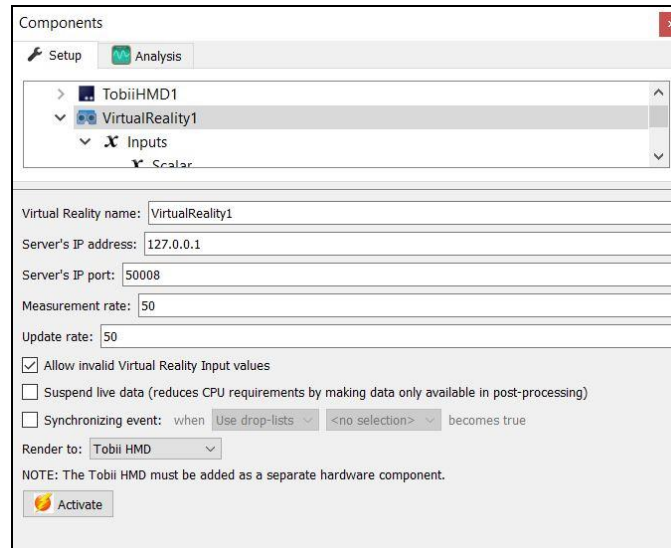
World Axes - Configuration of the World Axes layout for The MotionMonitor xGen



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.

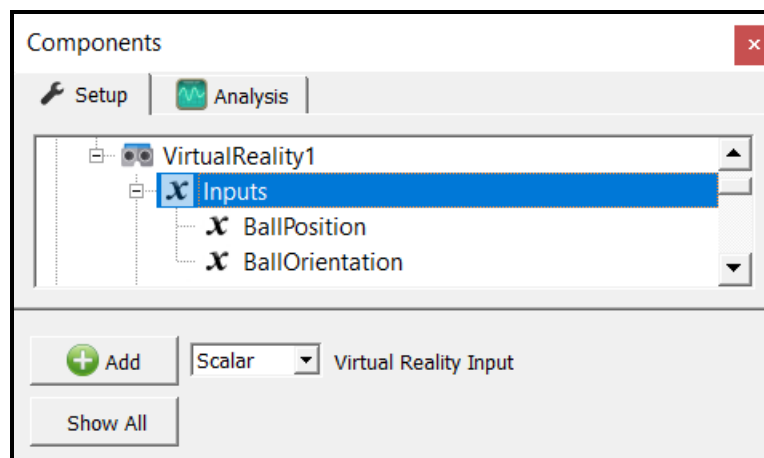


VirtualReality - Virtual Reality is the hardware component that controls the bi-directional communication between The MotionMonitor xGen and Unity or WorldViz virtual environments. The Server's IP address is either that of the local computer or of the server computer running the Unity application. The 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 a rate can cause data to become unsynchronized. Discuss this with your Client Support Engineer for more details.



Enable the input of invalid values from virtual reality, suspend live data to reduce processing demands of the software and add a synchronizing event using the check boxes. Finally select the medium for which the VR will be rendered to, either immersive display, Tobii HMD or standard HMD.

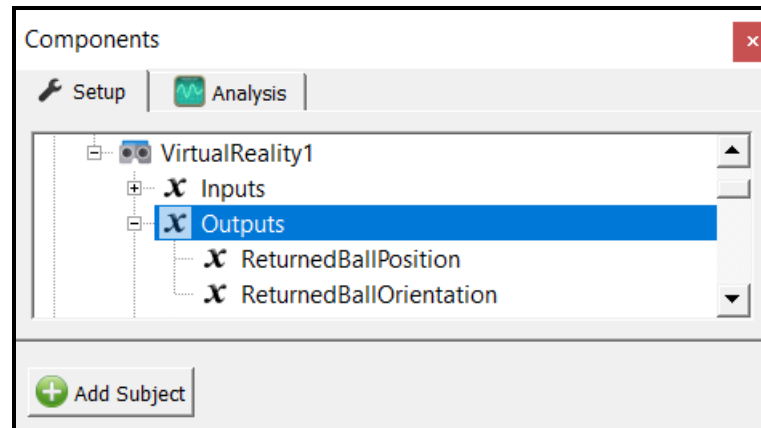
Inputs - Inputs are the variables that are sent to Vizard. Their names need to match that of the variables defined within the demoDataExchangeWithTMM.py.



BallPosition - Is a vector variable sent to Vizard whose value is used to control the position of a sphere.

BallOrientation- Is a rotation variable sent to Vizard whose value is used to control the orientation of a sphere.

Outputs - Outputs are variables that are received from Vizard. Their name needs to match that of the variable defined within the demoDataExchangeWithTMM.py.

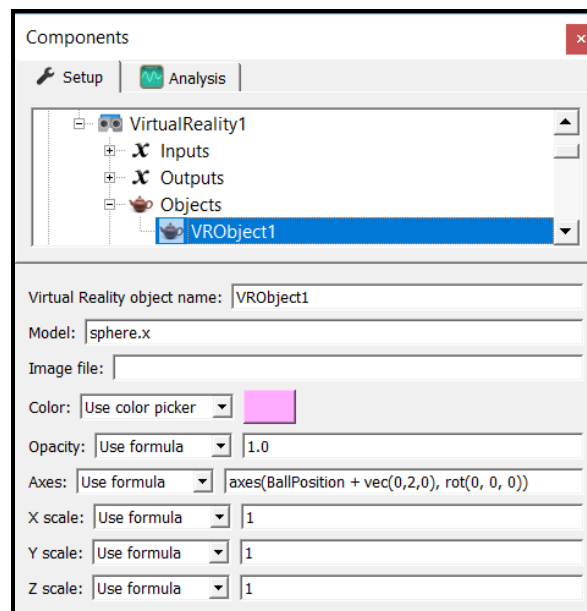


ReturnedBallPosition – Is a vector variable received from Vizard.

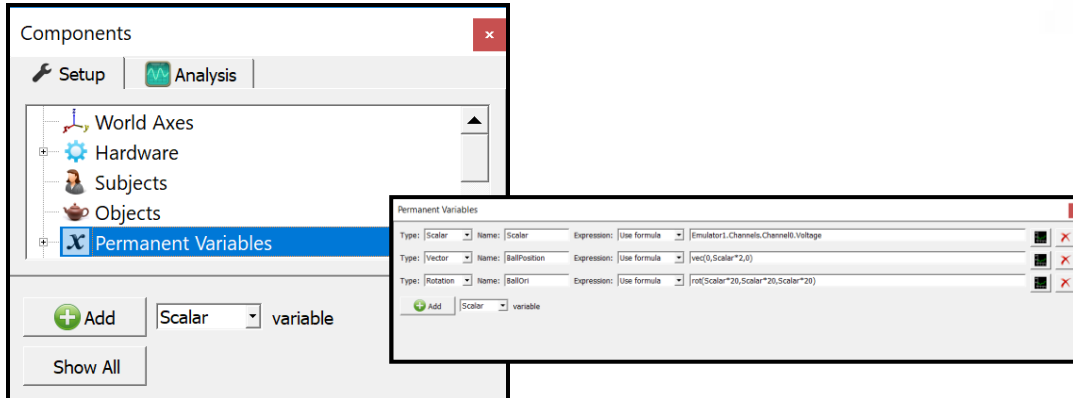
ReturnedBallOrientation – Is a rotation variable received from Vizard.

Objects - Any objects defined here will automatically be populated within the Vizard environment based on their defined parameters. This automatic population into the Vizard scene is handled by the MMServer script. The Model file specified here, typically an .obj file, needs to also be located within the Vizard directory. The same goes for if an image file is being used to display on the surface of an Object.

VRObject1 - Is an object whose axes (position and rotation) are used to control an object within Vizard. The model, in this case a Marker.obj file, also resides within the Vizard 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 files 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.



Scalar - This is the scalar variable that's being used for the "BallPosition" and "BallOrientation" permanent variable definitions.

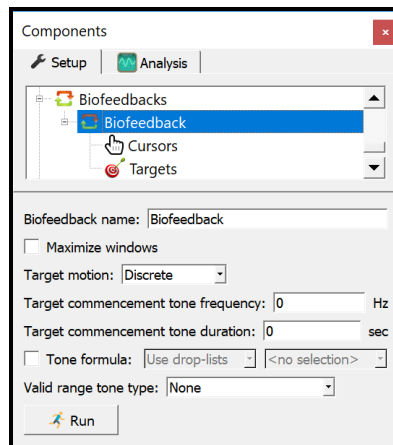
BallPosition - This is the vector variable that's being used for the "BallPosition" Virtual Reality Input.

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

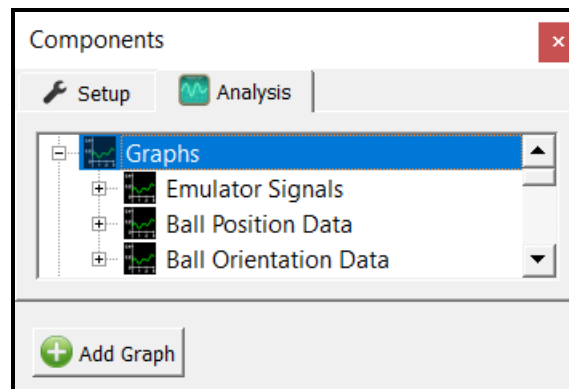
Biofeedbacks - Provides the ability to perform cursor-target paradigms. They provide data similar to VR Objects in that they are automatically populated in the Vizard environment, if enabled under the Virtual Reality Hardware node. However, there are many ways in which they differ including that running 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.

Ball Position Data – Includes plots for data from the BallPosition and ReturnedBallPosition vector Virtual Reality Inputs and Outputs

Ball Orientation Data – Includes plots for data from the BallOrientation and ReturnedBallOrientation Virtual Reality Inputs and Outputs.

There are some things to keep in mind.

1. The demo python script must be run from the VR folder containing mmserver.py file using Vizard 5 or above.
2. The demo python script must be restarted, each time TMM reconnects. If doing repeated collections within biofeedback, it is not necessary to restart the script.
3. Setting the measurement rate in the VirtualReality dialog at too high a rate can cause data to become unsynchronized. Discuss with your Client Support Engineer for more details.

Future Developments:

- Ability to stream scalar variables into Vizard.
- Ability to optimize Vizard clock for extended recordings.