MOXI SDK Unreal Plugin User Guide

Table of contents

Release Notes

v.0.13 : release date: 03/26, 2025

- Improve the firmware update mechanism.
- Fix the issue where the avatar appears to float during rendering.

v.0.12 : release date: 01/16, 2025

- Support firmware updates for upper or lower body devices separately (via USB connection) for MOXI SDK.
- The issue of incorrect firmware information retrieval when connecting the clothing via USB.

v.0.11 : release date: 06/20, 2024

- Change the avatar in the example scene.
- Enhance network connection stability.

v.0.10 : release date: 06/11, 2024

- Fix the delay in restarting motion capture.
- Provide the multiple floor contact feature.

v.0.9.2 : release date: 05/30, 2024

- Fix the error when packaging the *Unreal* project.
- Rebuild the MOXI SDK library using the MSVC v142 build tools to support *Unreal Engine* versions 5.0 to 5.4.

v.0.9.1 : release date: 05/27, 2024

- release the C++ programming interface

v.0.9 : release date: 05/20, 2024

- Fix an error when the system could not reconnect to the MOXI device via WIFI if the network environment was changed.

v.0.8 : release date: 05/13, 2024

- Improve connection stability.
- Shorten the initialization time.
- Add functions to receive and process compressed motion data for optimizing network performance.
- Add C Function : MxCStartRecordDeviceData
  - Start to record the motion data from MOXI devices.
- Add C Function : MxCFinishRecordDeviceData
  - Finish recording the motion data from MOXI devices.
- Add C Function : MxCIsDeviceConnected
  - Check if an IMU device is connected.
- Add C Function : MxCSetCaptureDataFormat
  - Configure the data format for motion capture.
  - There are three types of formats
    - eMocapDataType.MOXI_MOCAP_TYPE_MOTION
    - eMocapDataType.MOXI_MOCAP_TYPE_RAWMOTION
    - eMocapDataType.MOXI_MOCAP_TYPE_COMPRESSED_MOTION
    - The default type is eMocapDataType.MOXI_MOCAP_TYPE_COMPRESSED_MOTION
- Add C Function : MxCResetCharacterPosition
  - Reset the avatar position to it's base
- Add C Function : MxCStopCapture
  - Stop motion capture

v.0.1 : release date: 04/16, 2024

- Fix the crash problem that occurs when capturing motion using an unavailable bone name of an avatar.
- Clear all the bone names in the Skeleton Mapping property after changing the target skeletal mesh data.

v.0.0 : release date: 04/08, 2024

- Setting up the *MOXI SDK* within the *Unreal Engine Blueprint* interface.
  - Connecting function and it's callback event.
  - Doing Calibration function and it's callback event.
  - Setting up the natural pose function and it's callback event.
  - Animating a target avatar using the motion capture results.
- A sample scene for using the plugin

Introduction

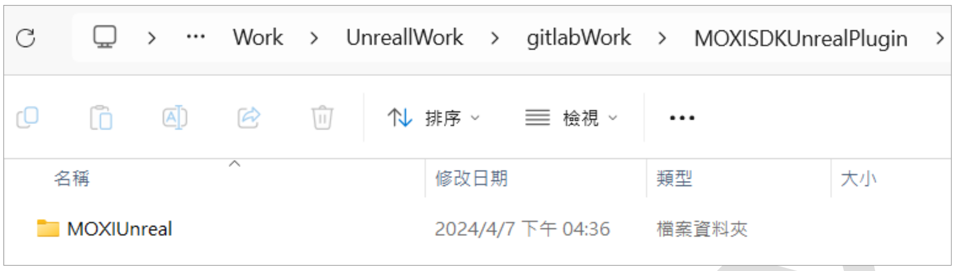
The *MOXI SDK Unreal Plugin* is the plugin for developers to use *j-mex* motion capture device in their applications. The plugin is the extension of the *MOXI SDK* which is a cross-platform API for *j-mex* IMU-based motion capture device, which provides high-performance access to realtime motion control and captures in games, vTubing, and XR applications.

The plugin provides an easy way to use the *MOXI SDK* in *Unreal*, with which developers can connect to the hardware devices and capture the motion data by using the UMOXIManager and UMOXIController, the two major *Unreal Blueprint* classes of the plugin. With the plugin software package, a sample scene and the associated UI components are included for reference

User Guide

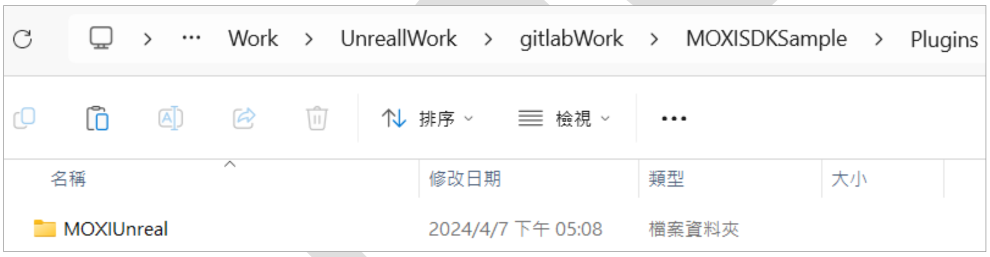
1. Installing the *MOXI SDK Plugin* and support assets

- Unzip the "MOXISDKUnrealPlugin.zip" file to extract the plugin and the support assets as shown below.


Work > UnrealWork > gitlabWork > MOXISDKUnrealPlugin >

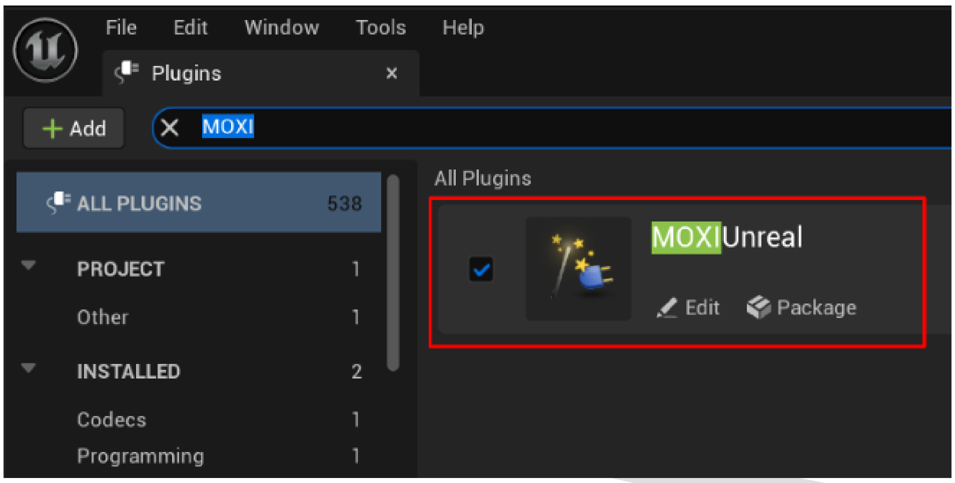
名稱	修改日期	類型	大小
MOXIUnreal	2024/4/7 下午 04:36	檔案資料夾	

- The "MOXIUnreal" Folder contains the *MOXI SDK Unreal Plugin* library built with *Unreal Engine* and a sample scene for using the plugin.
- Copy the extracted "MOXIUnreal" folder to the "Plugins" folder located in the project root directory. The below figure shows an example:

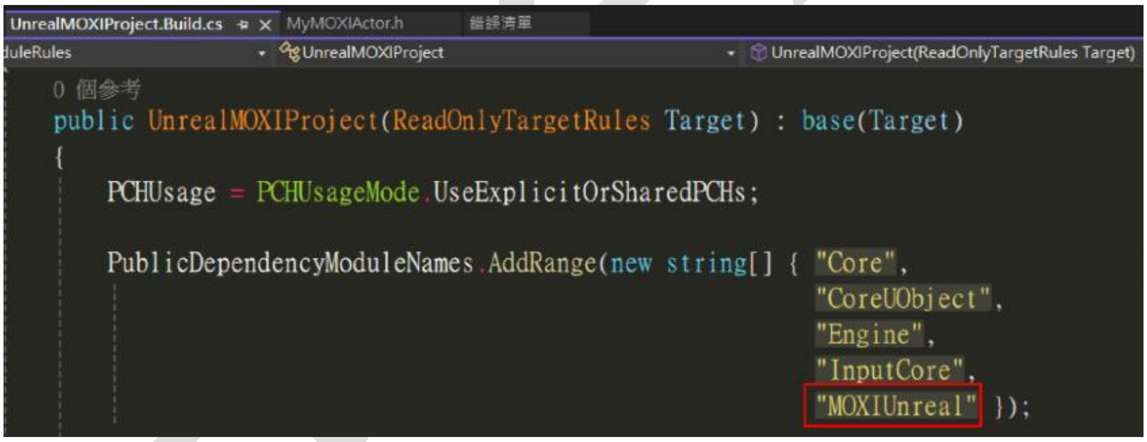

Work > UnrealWork > gitlabWork > MOXISDKSample > Plugins

名稱	修改日期	類型	大小
MOXIUnreal	2024/4/7 下午 05:08	檔案資料夾	

  - The "MOXISDKSample" folder represents the root directory of an *Unreal* project.
  - Place the "MOXIUnreal" folder into the "Plugins" folder. If there is no "Plugins" folder inside the root directory, create one named "Plugins" first.
- After setting up the folders, restart the *Unreal* project.
  - After restarting the *Unreal* project, verify if the plugin has been successfully installed by the following steps:
    - Clicking on the "Edit" menu and select “Plugins” to open the page for plugin management.
    - Type "MOXI" in search bar to show the plugin. If the plugin has been installed successfully, the plugin name should appear in the search results. The following image shows the process:



- Modify the Build.cs file of the project to use the C++ programming interface. An example is shown below :



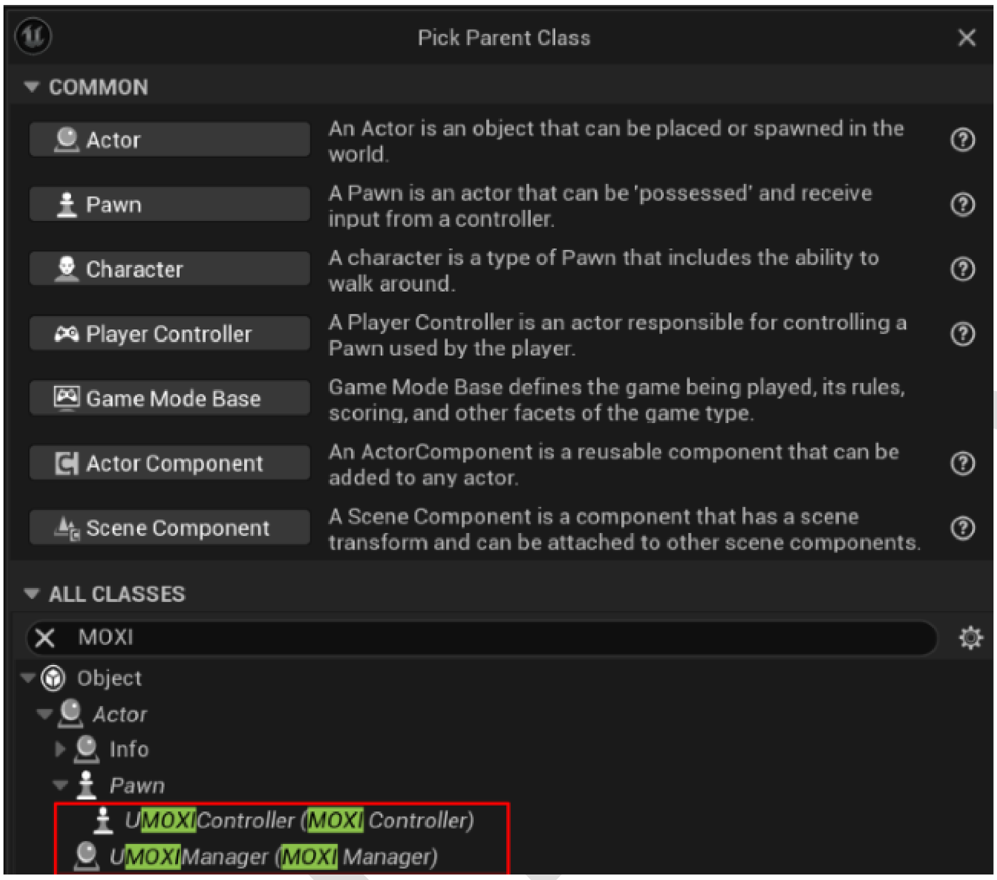
```
public UnrealMOXIProject(ReadOnlyTargetRules Target) : base(Target)
{
    PCHUsage = PCHUsageMode.UseExplicitOrSharedPCHs;

    PublicDependencyModuleNames.AddRange(new string[] { "Core",
                                                        "CoreUObject",
                                                        "Engine",
                                                        "InputCore",
                                                        "MOXIUnreal" });
```

- After setting up the MOXIUnreal module, users can include the header files of the C++ class of the example blueprints in their code, such as '`AUMOXIManager.h`', or directly include '`MoxiC.h`', the header file of the *MOXI* SDK, to communicate with *MOXI* devices.
  - Users can refer to the "MOXI SDK C++ User Guide" for details on using the MOXI SDK C++ interface.

2. *MOXI SDK Unreal Plugin* assets

- The plugin provides two *Unreal Blueprint* assets named "UMOXIController" and "UMOXIManager" as shown below:



Pick Parent Class

COMMON

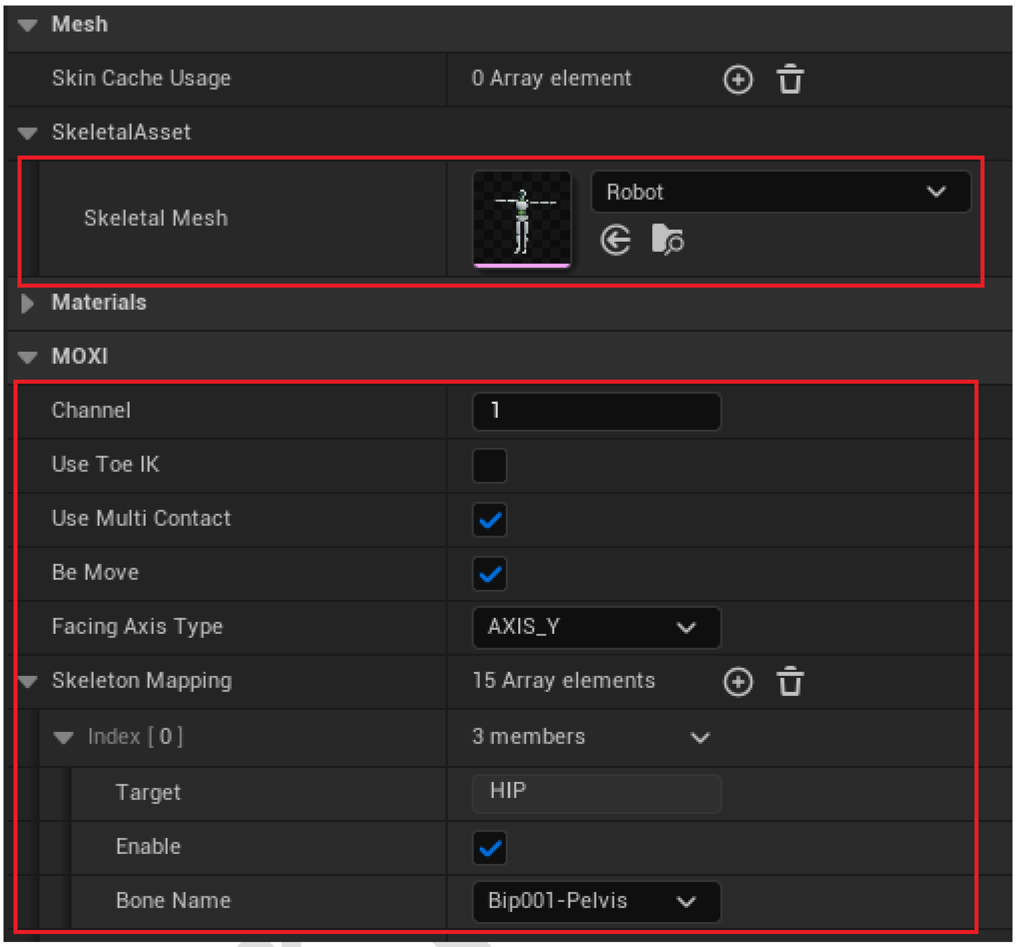
Actor	An Actor is an object that can be placed or spawned in the world.
Pawn	A Pawn is an actor that can be 'possessed' and receive input from a controller.
Character	A character is a type of Pawn that includes the ability to walk around.
Player Controller	A Player Controller is an actor responsible for controlling a Pawn used by the player.
Game Mode Base	Game Mode Base defines the game being played, its rules, scoring, and other facets of the game type.
Actor Component	An ActorComponent is a reusable component that can be added to any actor.
Scene Component	A Scene Component is a component that has a scene transform and can be attached to other scene components.

ALL CLASSES

MOXI

```
Object
  Actor
    Info
    Pawn
      UMOXIController (MOXI Controller)
    UMOXIManager (MOXI Manager)
```

- UMOXIController asset
  - Assigning a skeletal mesh data to be controlled by *MOXI* device.
  - Setting up bone mapping relationship with *MOXI SDK* library.
  - Communicating with *MOXI* device with specific channel id.


Mesh
Skin Cache Usage: 0 Array element
SkeletalAsset
Skeletal Mesh: Robot
Materials
MOXI
Channel: 1
Use Toe IK: ☐
Use Multi Contact: ✓
Be Move: ✓
Facing Axis Type: AXIS_Y
Skeleton Mapping: 15 Array elements
Index [0]: 3 members
Target: HIP
Enable: ✓
Bone Name: Bip001-Pelvis

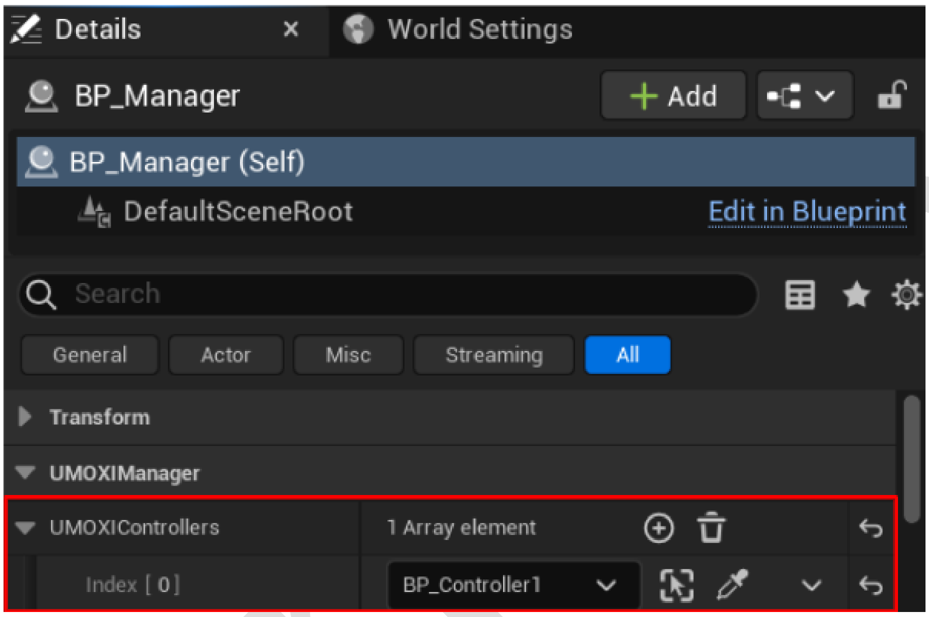
- Skeleton Mesh Data
  - To specify a skeletal mesh data of an avatar for motion control using the channel.

- Channel
  - The channel ID used in *MOXI Connect Mobile App* which is running on the *j-mex* hub to control all IMU-based devices.
  - The channel ID is a positive number which must be unique with the same networking environment. If the avatar is using the device directly connected to the system with USB, the ID is zero.

- Skeleton Mapping
  - To specify the bone joints of the skeletal mesh data for the *MOXI* controllers.
    - Target
      - The official name of the *MOXI* controllers, which are defined in `MOXI.h` in the *MOXI SDK*.
    - Enable
      - To enable or disable control of the bone joint by the *MOXI* controller.
    - Bone Name

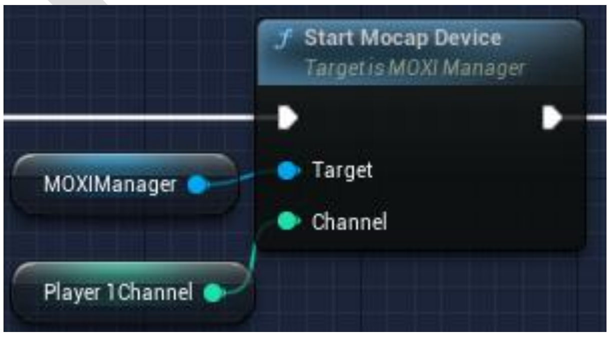
- The value is the assigned bone joint name of the skeletal mesh data.

- UMOXIManager asset
  - It oversees all instances of UMOXIController within the scene, managing their behavior and interactions.
  - The properties need to be set up as shown below:



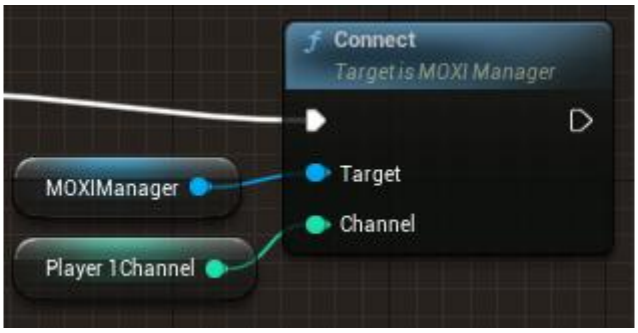
UMOXIControllers | 1 Array element
Index [0] | BP_Controller1

  - UMOXIControllers
    - An array of UMOXIController objects
      - Each UMOXIController object that needs to be used must be added manually.
  - The asset provides some functions to communicate with *j-mex* IMU-based motion capture device, as described below.
    - "StartMocapDevice"



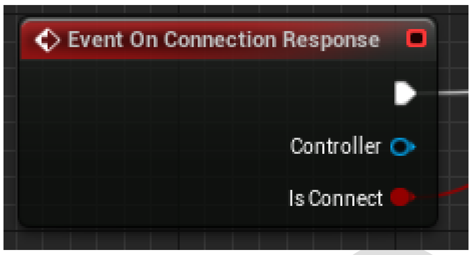
      - Initializing a system for a *MOXI* device.

    - "Connect"



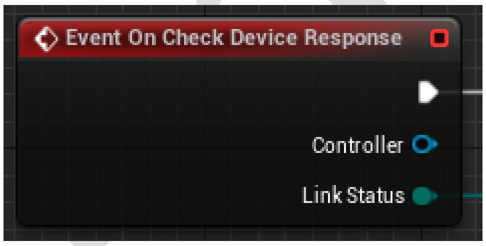
- Connecting to a *MOXI* device.

- "OnConnectionResponse"



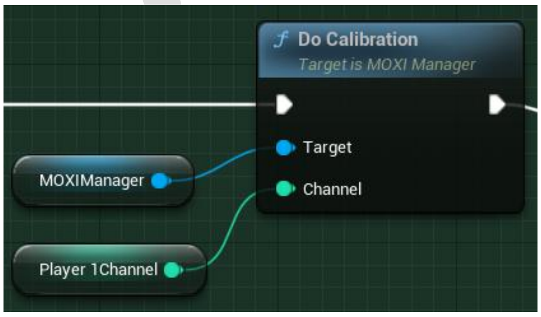
- The callback function to handle the responses from device connection.

- "OnCheckDeviceResponse"



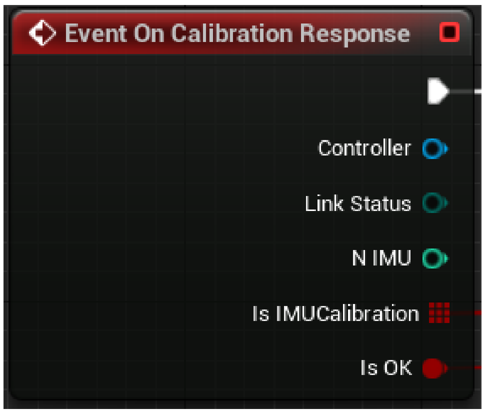
- The callback function to handle the returned connection status of the device hub.
- The system will return the status of upper-body hub and lower-body hub.

- "DoCalibration"



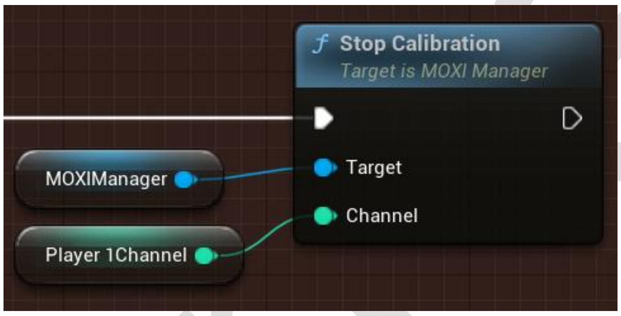
- The function can perform the calibration for a *MOXI* device.

- "OnCalibrationResponse"



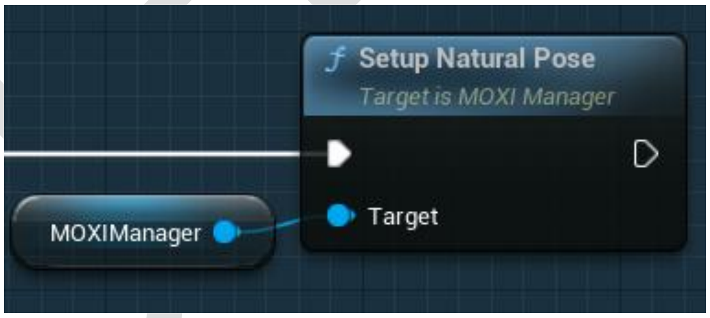
  - The callback function to handle the responses in calibration.

- "StopCalibration"



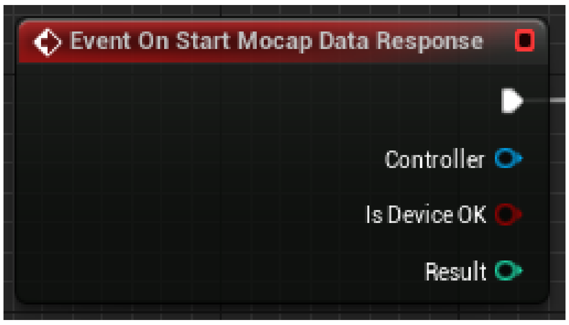
  - The function can stop the calibration process.

- "SetupNaturalPose"



  - The function can start the job of setting up the natural pose.
  - Before processing the function, the user should ask the performer to perform the same pose of the target avatar in natural pose (which can be T-pose or A-pose). The system will record the pose as the rest pose of the motion capture. Be sure to have the same pose as the target avatar to get the better mocap result.
  - The system will automatically move to motion capture mode after setting up the natural pose.
  - For more details, please refer to the provided tutorial of the sample scene.

- "StartMocapDataResponse"

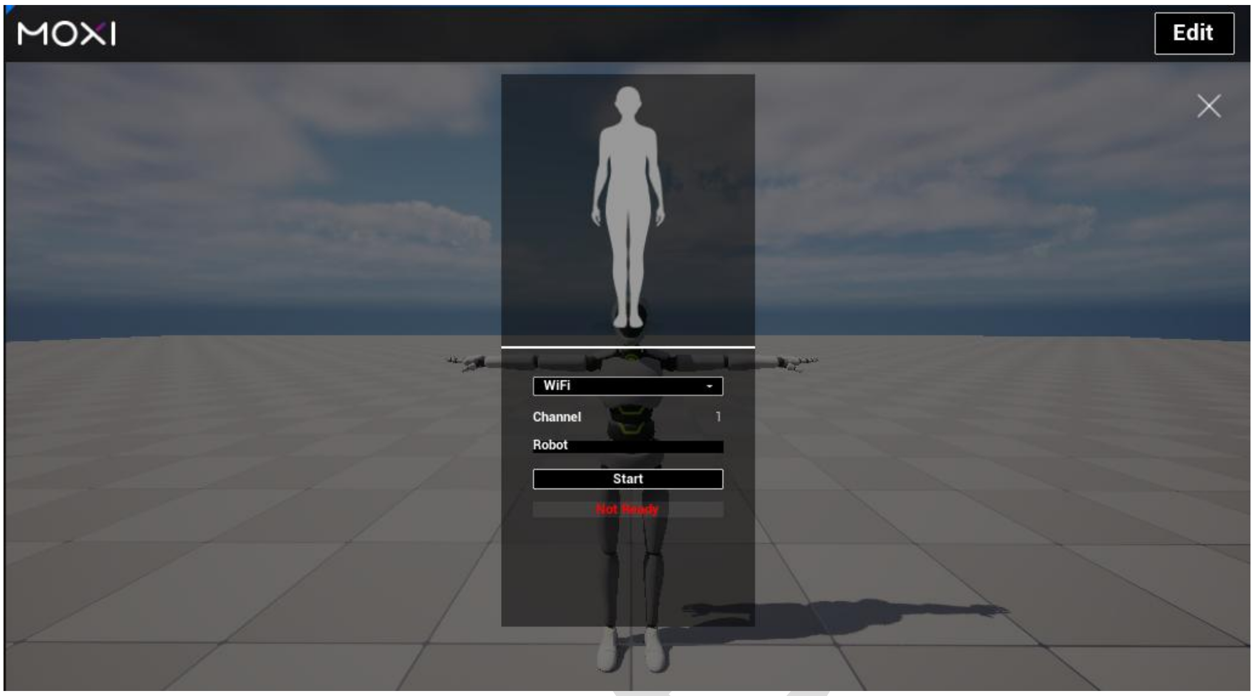


  - The callback function to handle the responses during the setup of natural pose.

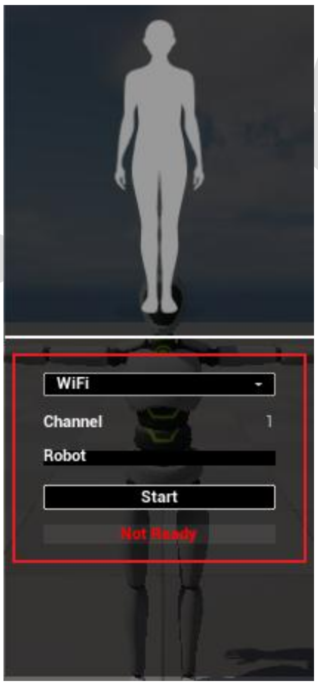
3. ***Using the plugin step by step***

- According to the previous descriptions, users can follow the steps below to set up the scene.
  - Adding <u>UMOXIManager</u> *Blueprint* class asset in the content browser
    - Dragging the asset into the scene
  - Adding <u>UMOXIController</u> *Blueprint* class asset in the content browser
    - Dragging the asset into the scene.
    - Setting up the channel ID, skeletal mesh data, skeleton mapping.
    - Assigning the controller object to <u>UMOXIManager</u>.
    - Repeating the step to create different avatar.
  - Setting up interaction functions and each callback event functions.

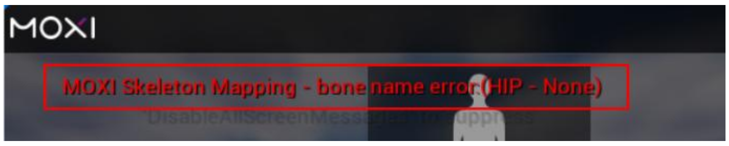
4. A tutorial to use the example scene



- Step 1 : selecting the connection method



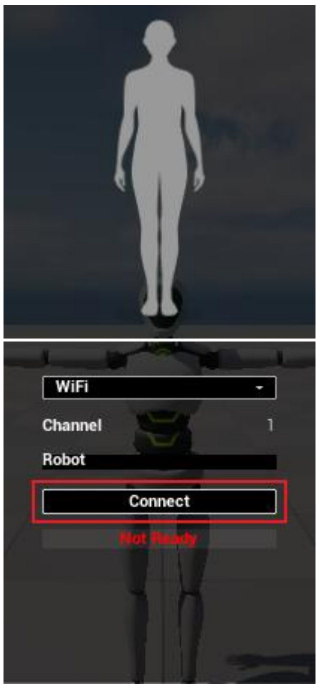
- Selecting the connection protocol : WIFI or USB
- Assigning a channel ID
- Clicking "Start" button to be ready to start the initialization
- If the system is ready to start, the plugin will open the "Connect" button for connection operation.
- If the system cannot find the bone name used in the Skeleton Mapping property, it displays an error message on the screen.



MOXI

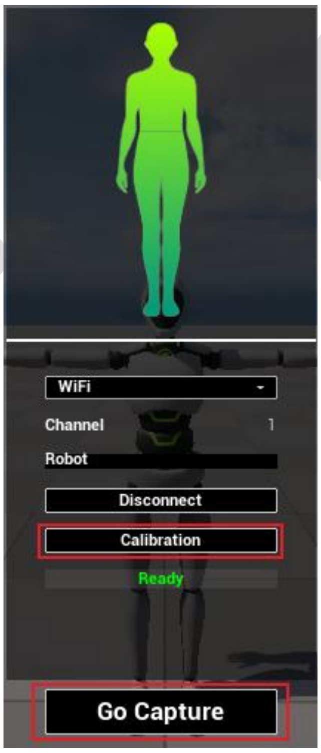
MOXI Skeleton Mapping - bone name error (HIP - None)

- Step 2 : selecting the connection method



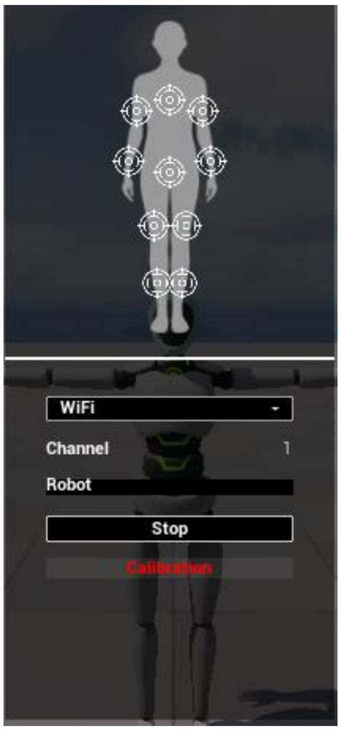
- Clicking "Connect" button to connect to the *MOXI* devices
- If the system is connected successfully, The plugin will open the "Calibration" button for next step.

- Step 3 : perform the calibration process

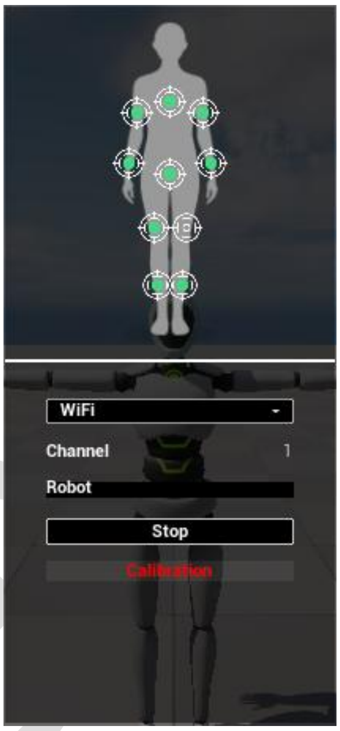


- Click "Calibration" button to do the hardware calibration.
  - Please refer the calibration step for more information.
- If the user has completed the calibration job previously, the user can skip the calibration step and directly click the "Go Capture" button to the final step.

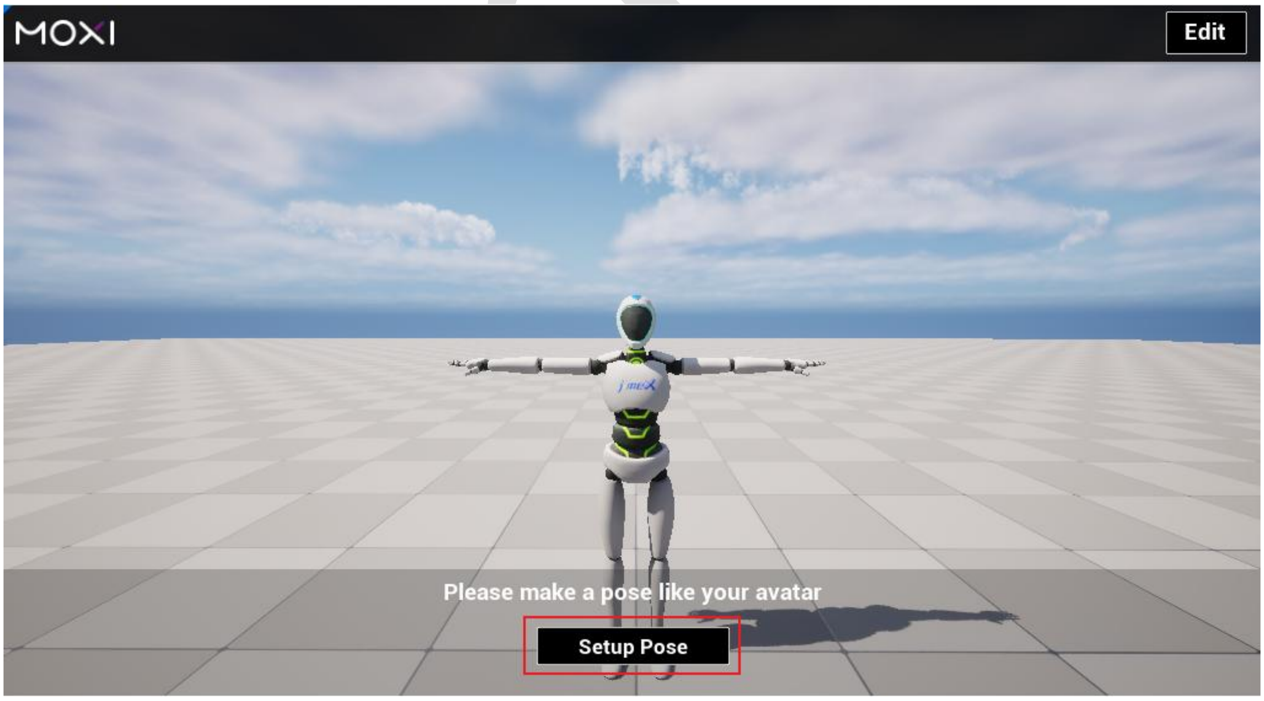
- Calibration Step:



Calibrating devices on your body by moving and rotating the body parts until the target icon becomes green. then can go to the final step.



- Final Step : set up the natural pose of the target avatar



- Clicking the 'Setup Pose' button starts the job of setting up the natural pose.
- Before pressing the button, the user should ask the performer to perform the same pose of the target avatar in natural pose (which can be T-pose or A-pose). The system will record the pose as the rest pose of the motion capture. Be sure to have the same pose as the target avatar to get the better mocap result.
- The system will automatically move to motion capture mode after setting up the natural pose.