DTrack Plugin for Unreal Engine 4/5

GitHub - ar-tracking/UnrealDTrackPlugin: ART DTrack interface for Unreal Engine

This is a plug-in for the Unreal Engine with the purpose of native integration of the Advanded Realtime Tracking *DTrack* tracking solutions. It injects data into the engine through LiveLink. Data can be accessed through Blueprint or C++. The plugin currently supports the DTrack body `6d` and flystick `6df2` as well as the finger tracking `gl` data format.

Prerequisites

- Unreal Engine 4 (4.23 or later), Unreal Engine 5 (5.0 or later)
- Windows
- Microsoft Visual Studio. See the Unreal documentation for selecting and installing the correct version.

Installation

Preparation

- The plugin is prepared to run with Unreal Engine 5. If you want to use it with Unreal Engine 4, you have to manually remove the entry "*LiveLinkAnimationCore*", from the file *Source\DTrackPlugin\DTrackPlugin.Build.cs*.

Install into the global Engine plugin folder

- Compile the plugin manually:
  *<UEDir>\Engine\Build\BatchFiles***RunUAT.bat BuildPlugin -Plugin=***\Path\to\DTrackPlugin.uplugin* **-TargetPlatforms=***Win64* **-Package=***<OutDir>* **-Rocket -VS20*XY***
  Here *-VS20XY* designates the Visual Studio version chosen above (e.g. -VS2022).
- Copy the folder *<OutDir>* to *<UE4Dir>\Engine\Plugins\DTrackPlugin*

Alternatively install into your local project plugin folder

- Open the *Unreal Editor* and create an Unreal C++ project
- Copy the plugin to *<project>\Plugins\DTrackPlugin*
- Compilation then takes place automatically when starting your Unreal project

DTrack Configuration

Room Calibration

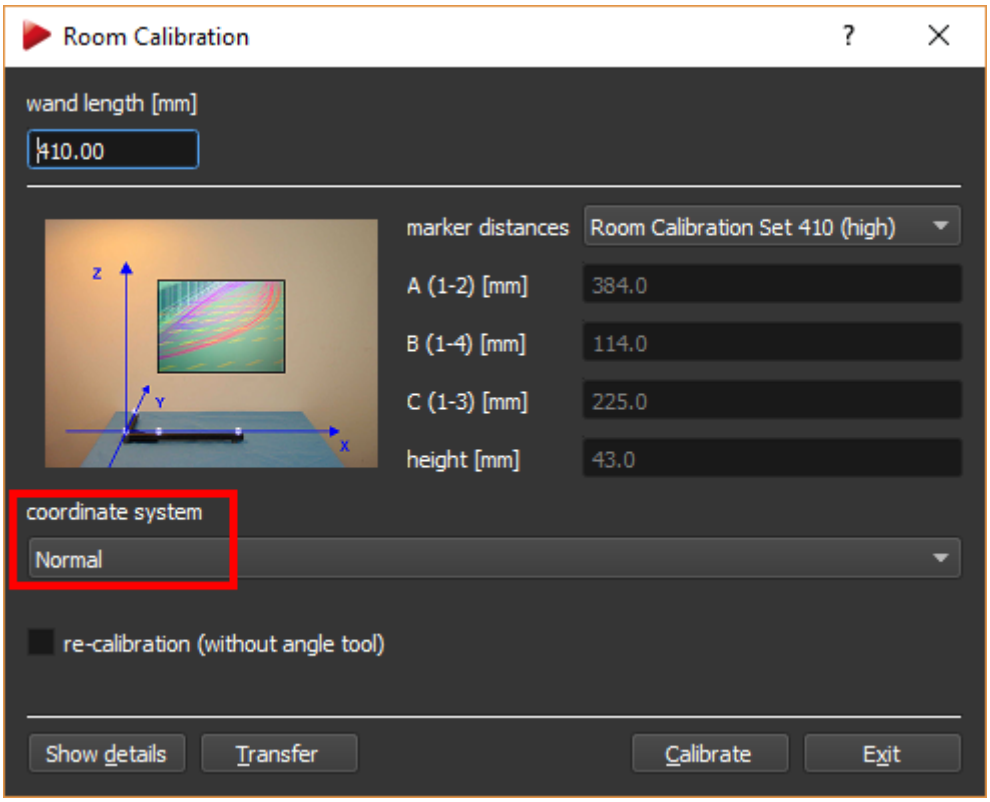
For general information about the DTrack room calibration and room adjustment see the DTrack User Manual. Here we discuss details relevant for use with the Unreal Engine.

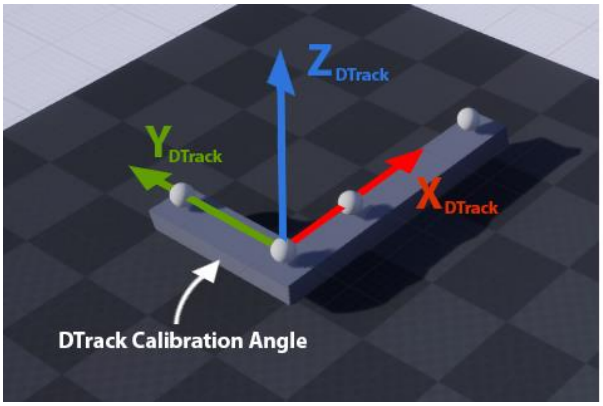
The calibration angle which comes with your ART tracking system defines the coordinate system layout in your tracking area. It consists of four retroreflective or active markers mounted onto a L-shaped frame.

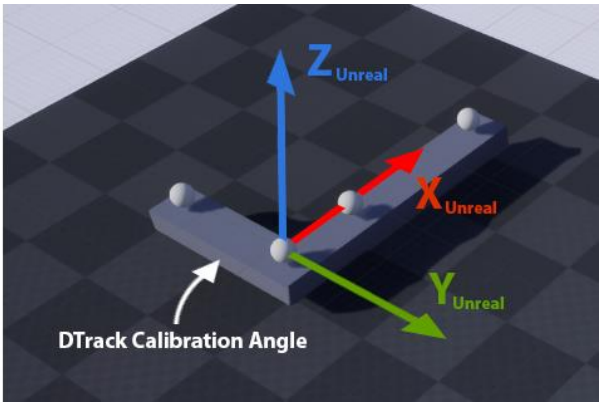
The marker in the edge of this L-shape by default designates the origin of the DTrack coordinate system. When using the *Normal* calibration mode, the long arm of this L-shape corresponds to the X axis, the short arm to the Y axis. DTrack coordinates refer to a right-handed coordinate system, so when the angle is placed flat on the ground with the markers on top the Z axis points upwards.

You can change orientation and position of the DTrack coordinate system with respect to the calibration angle via *Tracking* > *Room adjustment* in the DTrack UI.

The plugin transforms a right-handed position of a DTrack 6DoF measurement to a left-handed Unreal position by inverting the Y axis: $(X_{Unreal} , Y_{Unreal} , Z_{Unreal}) = (X_{DTrack} , -Y_{DTrack} , Z_{DTrack})$.

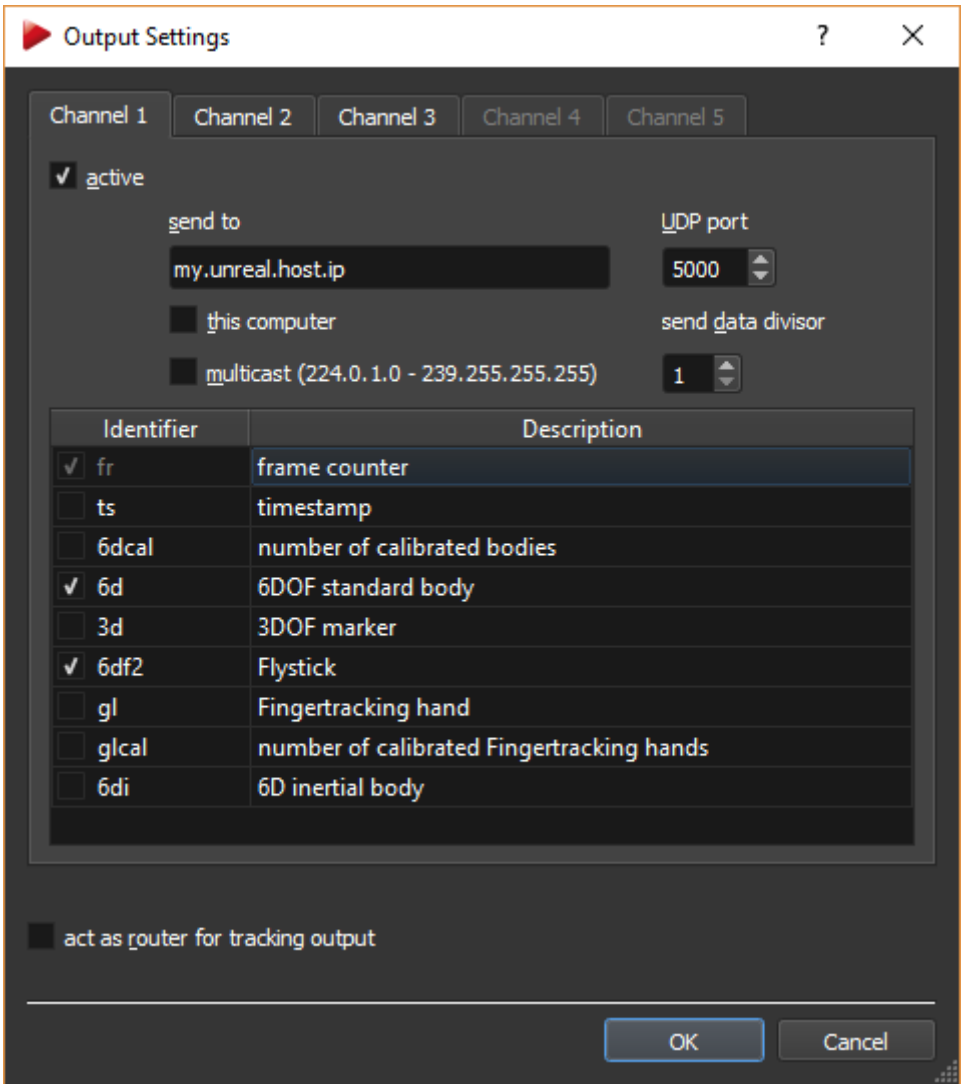






DTrack Output Configuration

Via *Tracking* > *Output* in the DTrack UI you can set up IP and port of the host of your *Unreal Editor* or application. In the corresponding dialog, you can also enable the DTrack output types `6d`, `6df2` and `gl`.



[1]

Plugin Usage

See UnrealDTrackSample for a detailed example.

The mapping of Flystick buttons and joystick is listed in *DTrackFlystickInputDevice.cpp* within the *DTrackPlugin\Source\DTrackInput\Private* directory.

DTrack Plugin for Unreal Engine 4/5 - Sample Scene

An *Unreal Editor* project to demonstrate *UnrealDTrackPlugin* features.

Installation

Install the Plugin

- Download *UnrealDTrackPlugin* from *Github* and follow the installation instructions there
  (Download via *Unreal Engine Marketplace* is planned for a later point in time)

Configure DTrack

- Calibrate or re-adjust your DTrack room coordinate system so that
  - the origin is close to the area where you later want to track your ART targets
  - the Z axis points upwards
- See the documentation of *UnrealDTrackPlugin* and the *DTrack User Manual* for more details on room calibration and adjustment.
- Open the DTrack2 UI and configure your output data stream (*Tracking* > *Output*):
  Set the *UDP port* to *5000*, and enable output of `6d`, `6df2` or `gl` depending on your available hardware.
- Start the DTrack measurement of your bodies and/or flysticks.

Configure the sample project

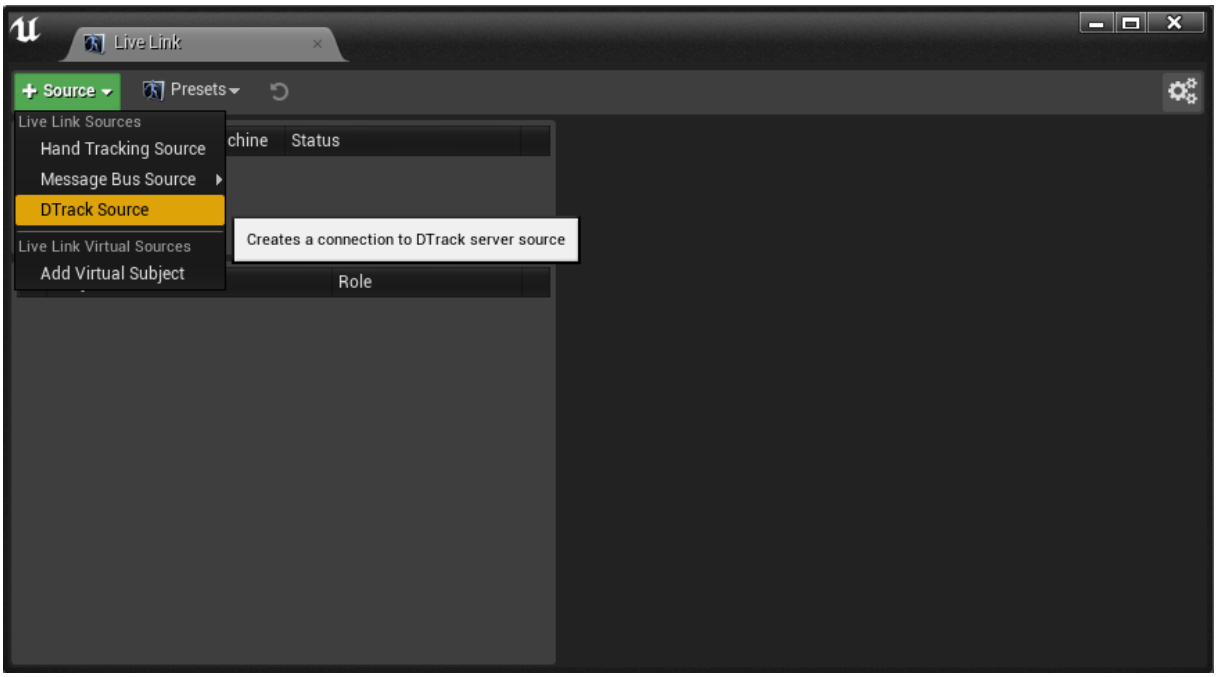
- Download and copy *UnrealDTrackSample* into your *Unreal Editor* project directory
- Adopt the *.uproject* file to your Unreal Engine version (e.g. set *"EngineAssociation": "4.27"*)
- If using Unreal Engine 4 (instead of 5) prepare *UnrealDTrackPlugin* as described in *UnrealDTrackPlugin*
- Open the *.uproject in *Unreal Editor* and agree to rebuild the project
- Ignore any error complaining that *TestMap_BuildData* is missing - this file was removed to reduce project size.
  In *Unreal Editor* click the *Build* button in the toolbar to rebuild the map *TestMap*, then save the map

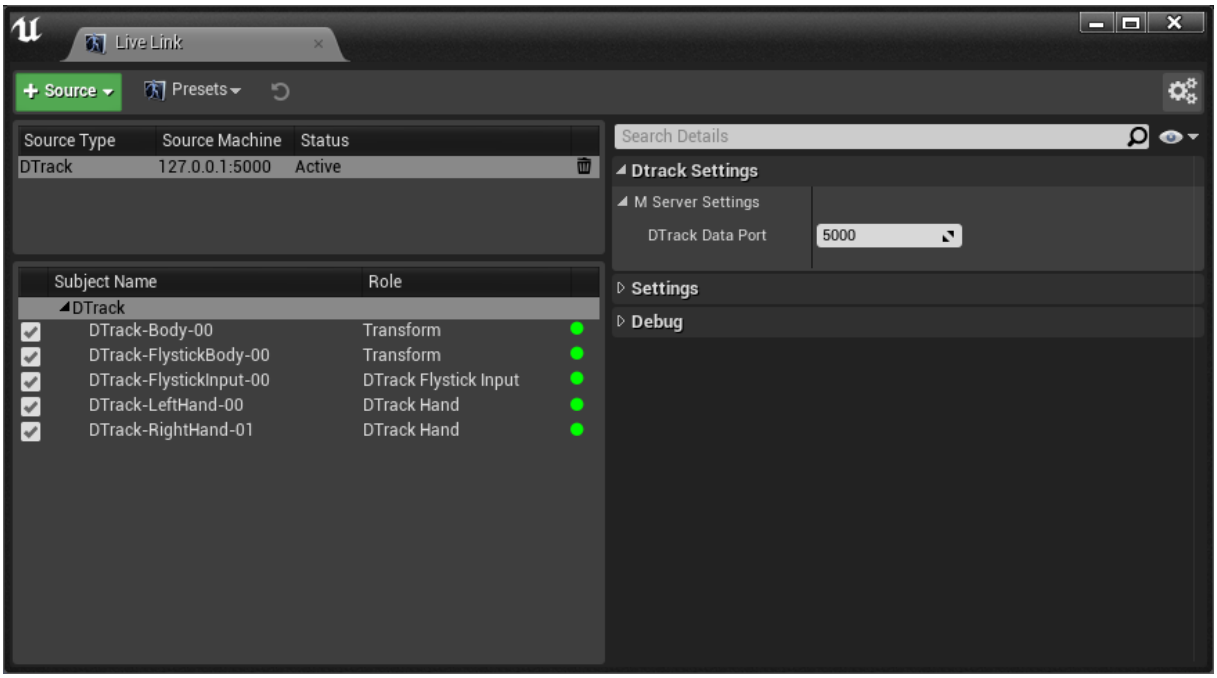
- In *Edit* > *Plugins* search for *DTrackPlugin* and enable it
- In the *Window* > *Live Link* (UE4) or *Window* > *Virtual Production* > *Live Link* (UE5) dialog add the Source *DTrack* and configure *DTrack Settings* > *Server Settings*

DTrack Live Link Source Configuration

The following screenshots show an example Live Link configuration for DTrack. Note that the DTrack data stream is split into three *Roles*:

- a *Transform Role* for 6DoF data (of standard bodies as well as Flystick bodies),
- a *DTrackFlystickInput Role* for Flystick button and joystick data,
- a *DTrackHand Role* for Fingertracking data



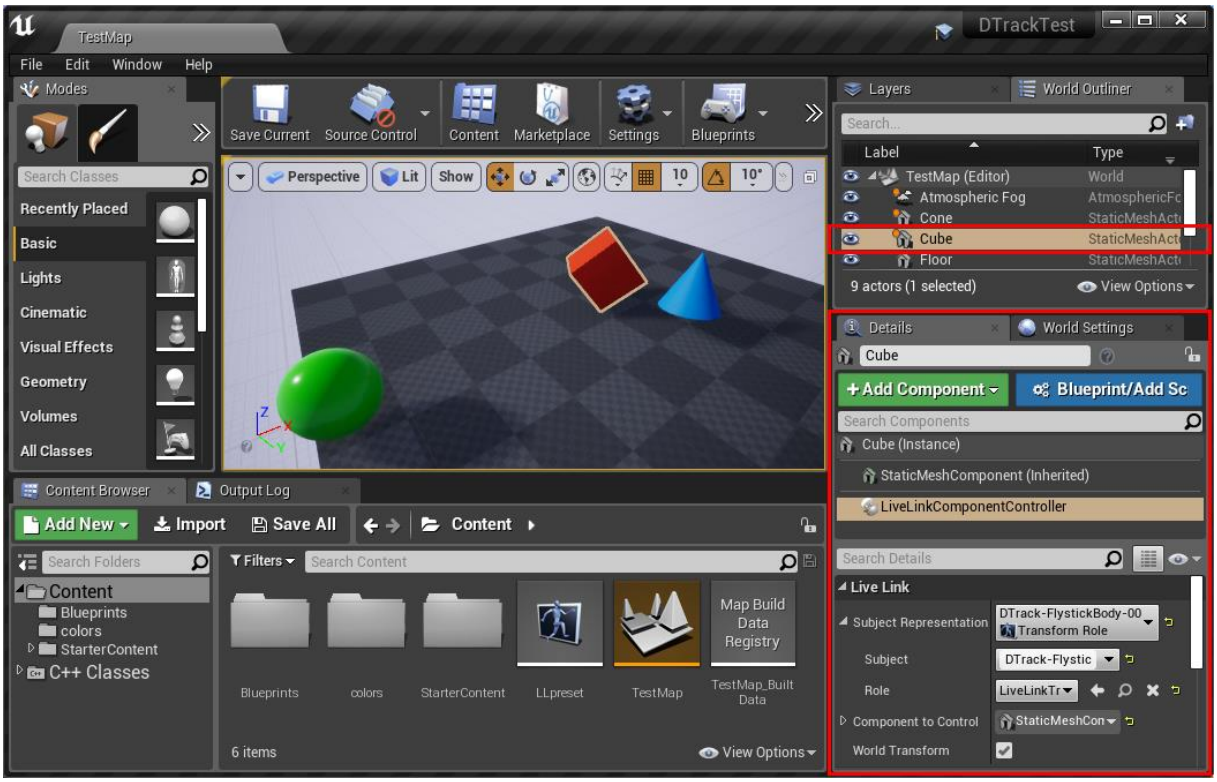


Transform Role Configuration

In the Editor window, select either the *Cone* or the *Cube*. In the *Details* tab of this actor, select the component *LiveLinkComponentController*. In the *Live Link* section of this component you will find Role and Live Link Subject this actor is associated with.

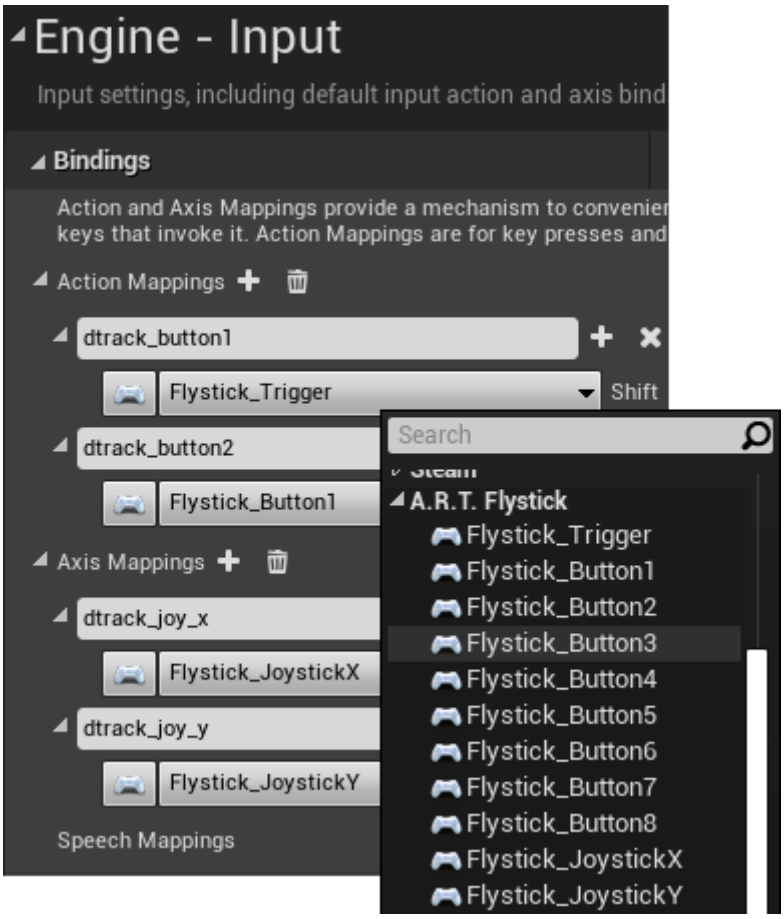
For 6D transformations this is always a *Transform Role* (*DTrackFlystickInput Role* is discussed below). Here, the *Cone* is configured to be controlled by a standard 6DoF body with *DTrack ID 1* ('*DTrack-Body-00*'), the *Cube* is controlled by a Flystick 6DoF body with *DTrack ID F1* ('*DTrack-FlystickBody-00*').

At this point, when you look to the Editor viewport, either *Cone* or *Cube* should already move in sync with corresponding targets tracked by DTrack (if the Edtior's *Viewport Options* are set to *Realtime*).

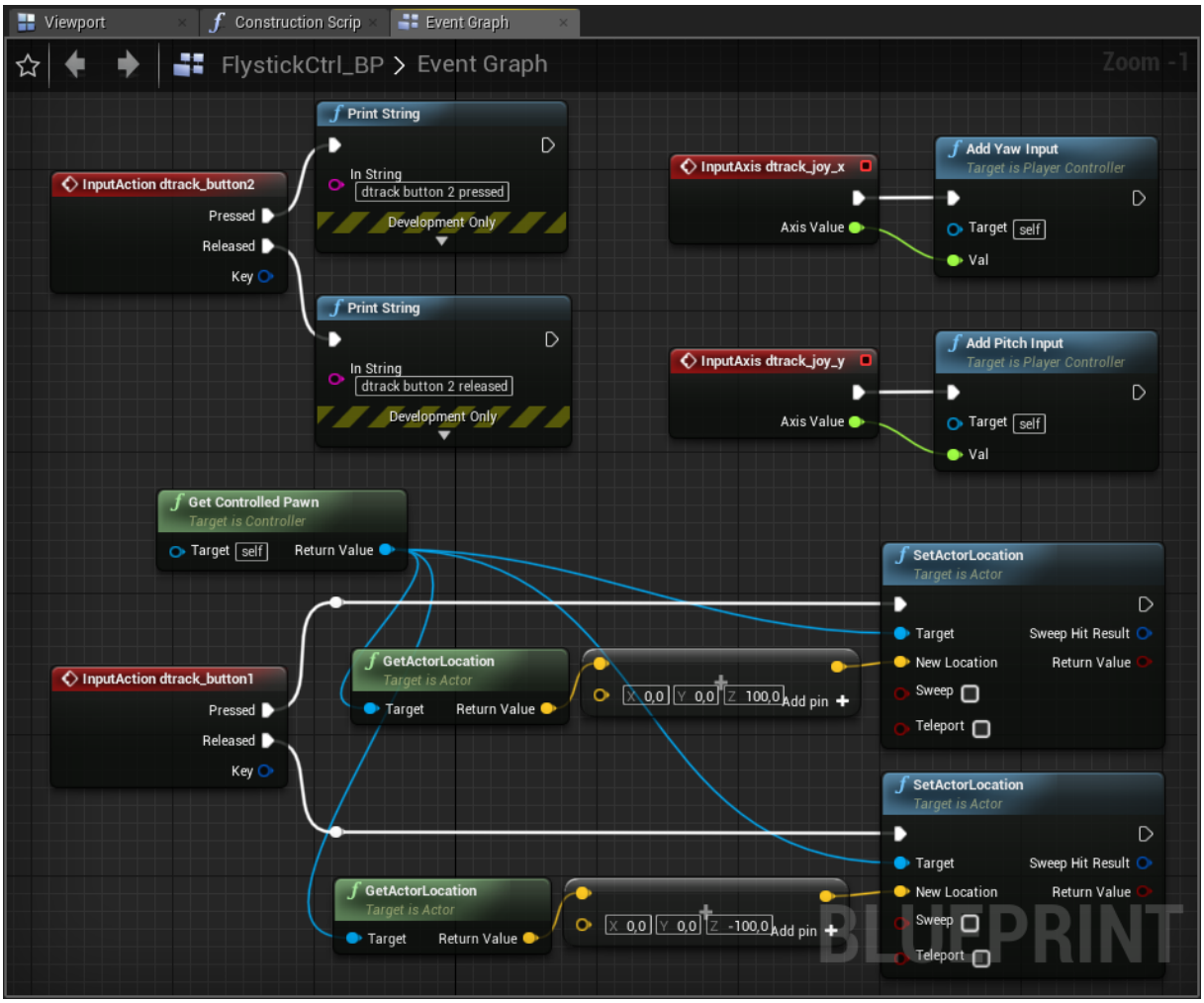


DTrackFlystickInput Configuration

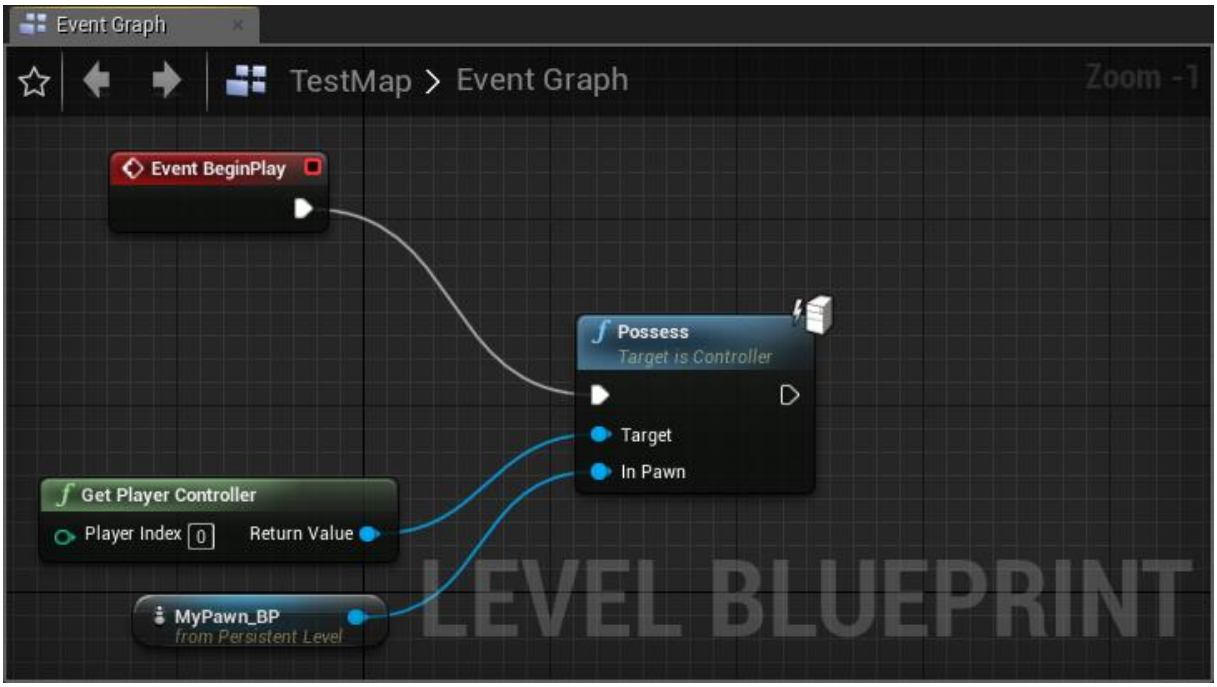
UnrealDTrackPlugin maps Flystick buttons and joystick via a custom *A.R.T. Flystick* device. Note that *UnrealDTrackPlugin* can be used with the new **Enhanced Input** system (since UE4.26) as well as the 'legacy' input system (in *Edit* > *Project Settings* > *Engine* > *Input*, marked as deprecated in UE5.1).



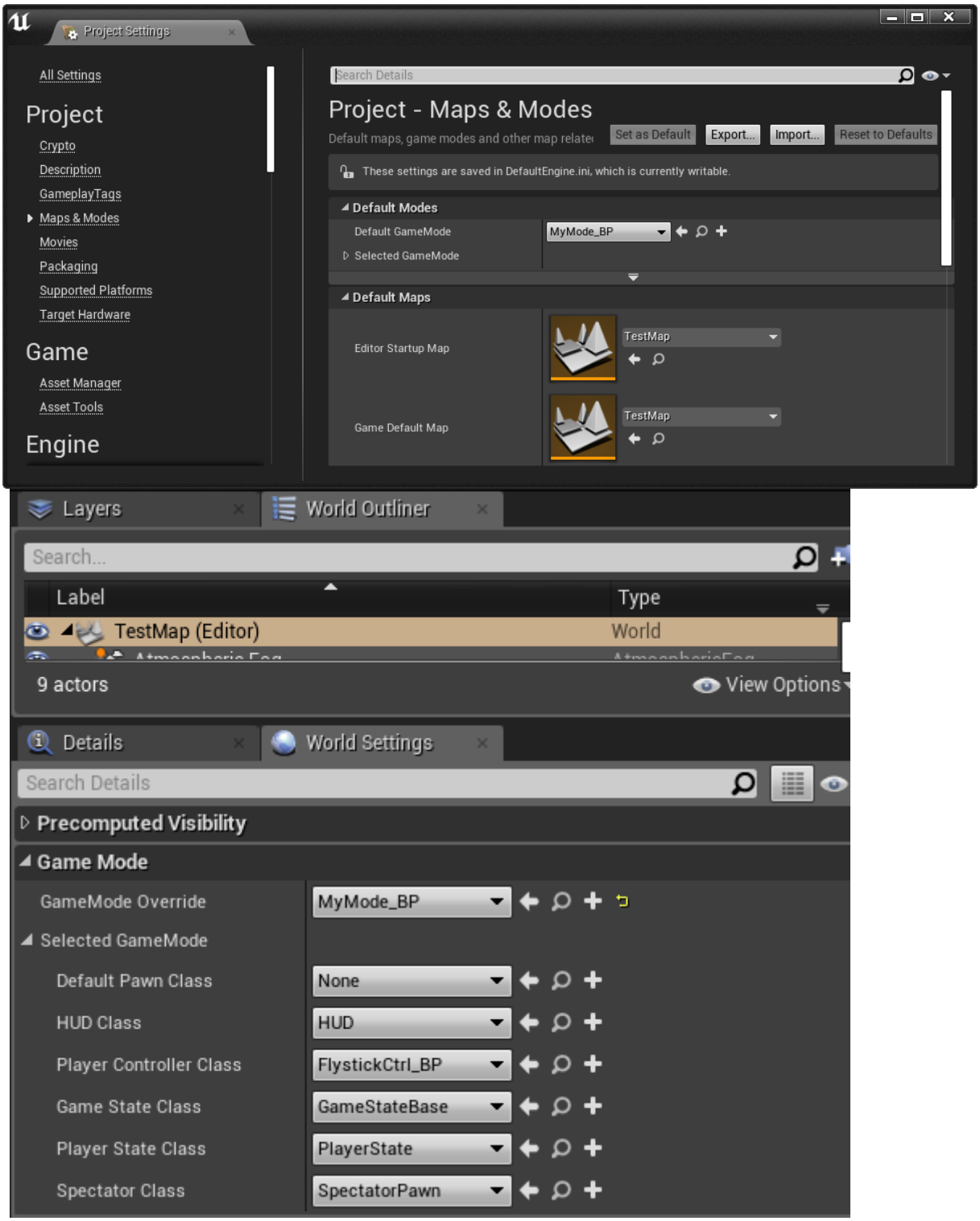
Actions and axis values triggered by the Flystick are then handled by the Playercontroller *FlystickCtrl_BP* (in the Content Browser under *Content* > *Blueprints*). This contains a simple script which demonstrates that Flystick data are actually passed-through by the plugin: The joystick rotates the player camera, and button presses lead to either jumps in camera location or a message printed to the screen.



The *Level Blueprint* of *TestMap* then associates the PlayerController with the Pawn *MyPawn_BP* in the scene.



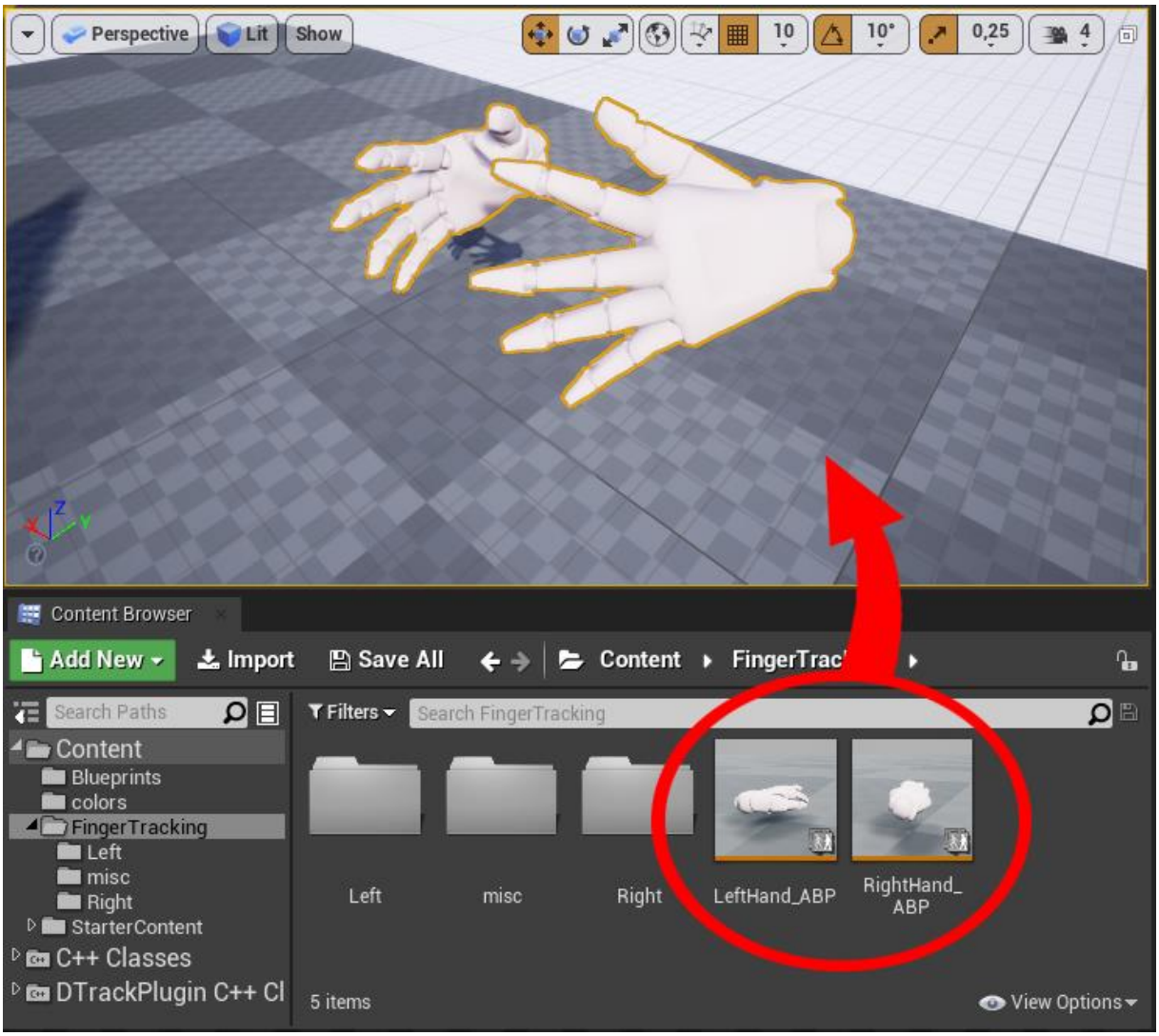
The game mode *MyMode_BP* uses *FlystickCtrl_BP* as the Playercontroller class, and is itself set up as the default game mode used by *TestMap*.



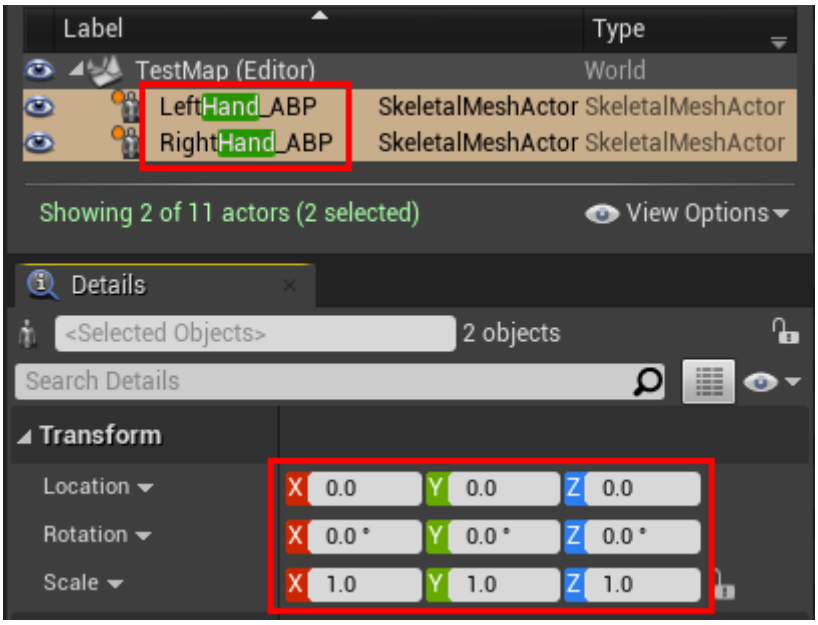
Finally you can test this configuration in *Play In Editor* (PIE) mode.

Fingertracking

In case you have a *Fingertracking* set connected to your *A.R.T. tracking system*, you can send the corresponding data to your *Unreal* application just as you did with standard bodies and Flystick. To use such data in this demo project go to *Content* > *FingerTracking*, select the assets *LeftHand_ABP/RightHand_ABP* and drop them into the scene:

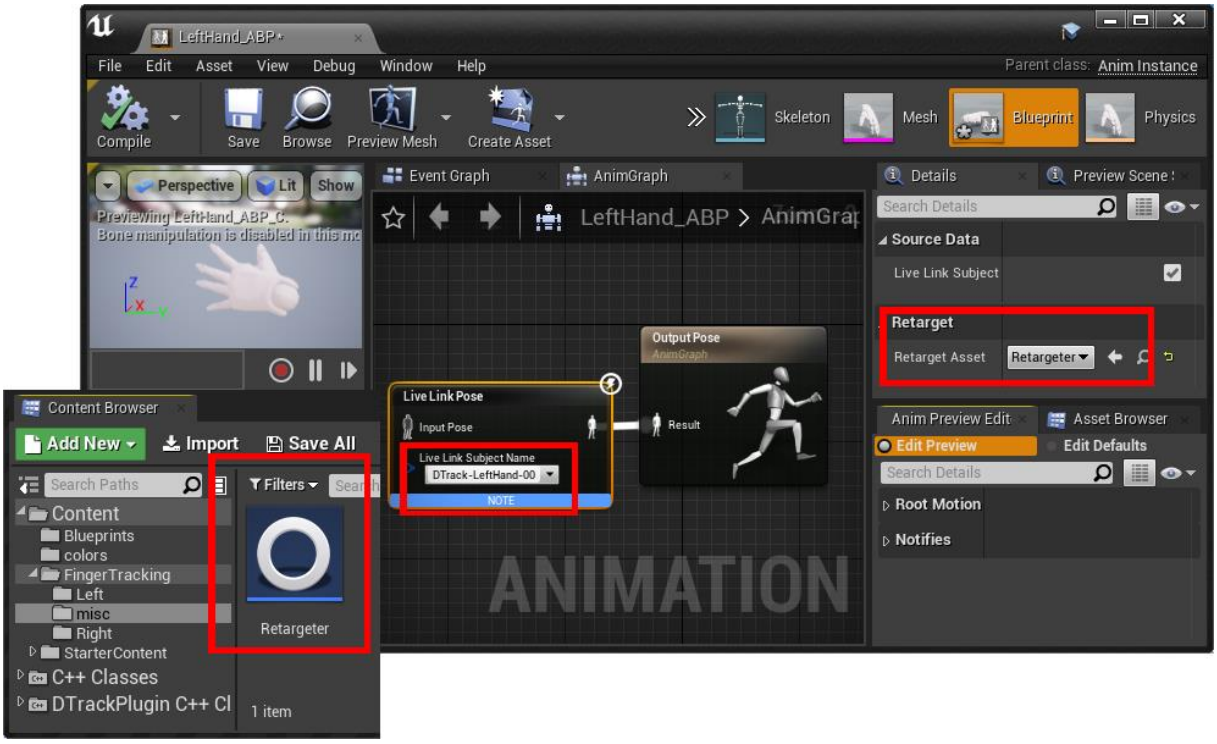


Note that the Unreal hand locations and orientations only correspond to the DTrack room coordinate system if for both hands in *Unreal Editor* > *Details* > *Transform* all values for rotation and location are set to `0`, and for scale to `1`.



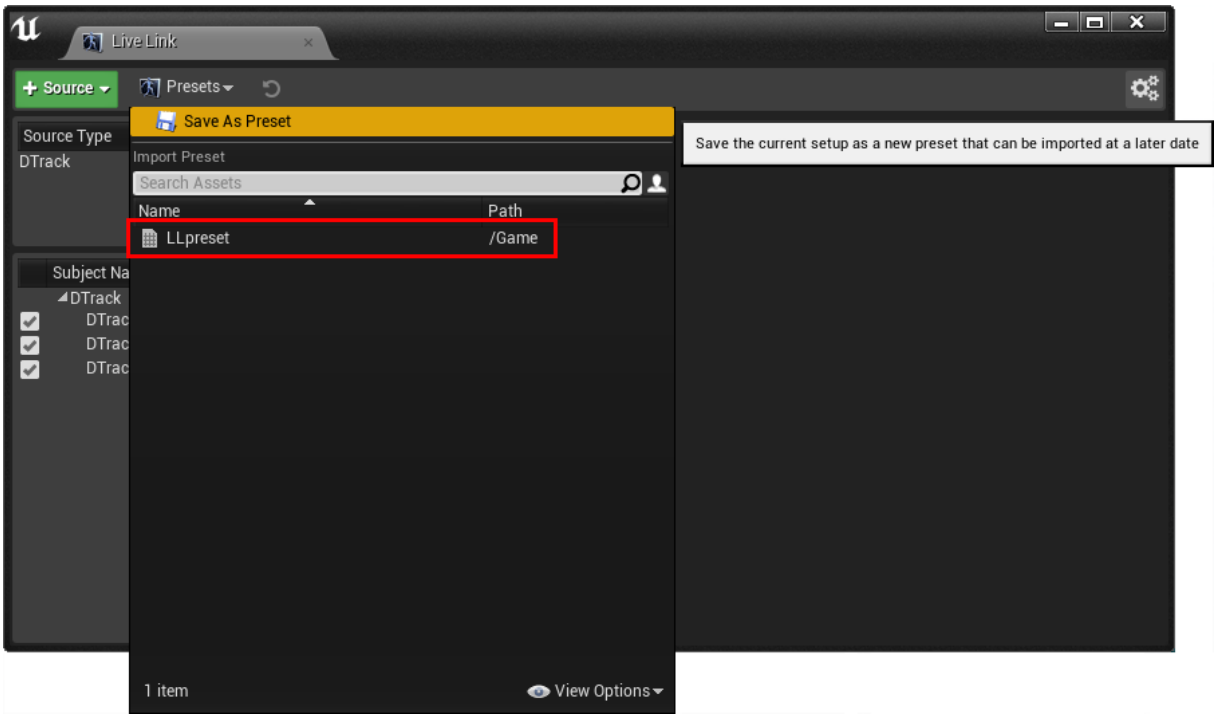
To view movement of hands and fingers directly within the scene, enable *Play* > *Simulate*. Alternatively, you can also switch to *Play In Editor* (PIE) mode.

The asset *misc* > *Retargeter* is a Blueprint derived from *DTrackLiveLinkRetargetAsset*, which is part of *DTrackPlugin*. The folders *Left* and *Right* contain skeletons and meshes which are edited versions of the SK_Mannequin. The assets *LeftHand_ABP* and *RightHand_ABP* are AnimationBlueprints associated with the target skeletons *LeftHand_Skeleton* and *RightHand_Skeleton*, respectively. Here, the *Live Link Pose* node is associated with *Retargeter*, and its subject is set to one of the *DTrackHand Role* instances configured earlier:

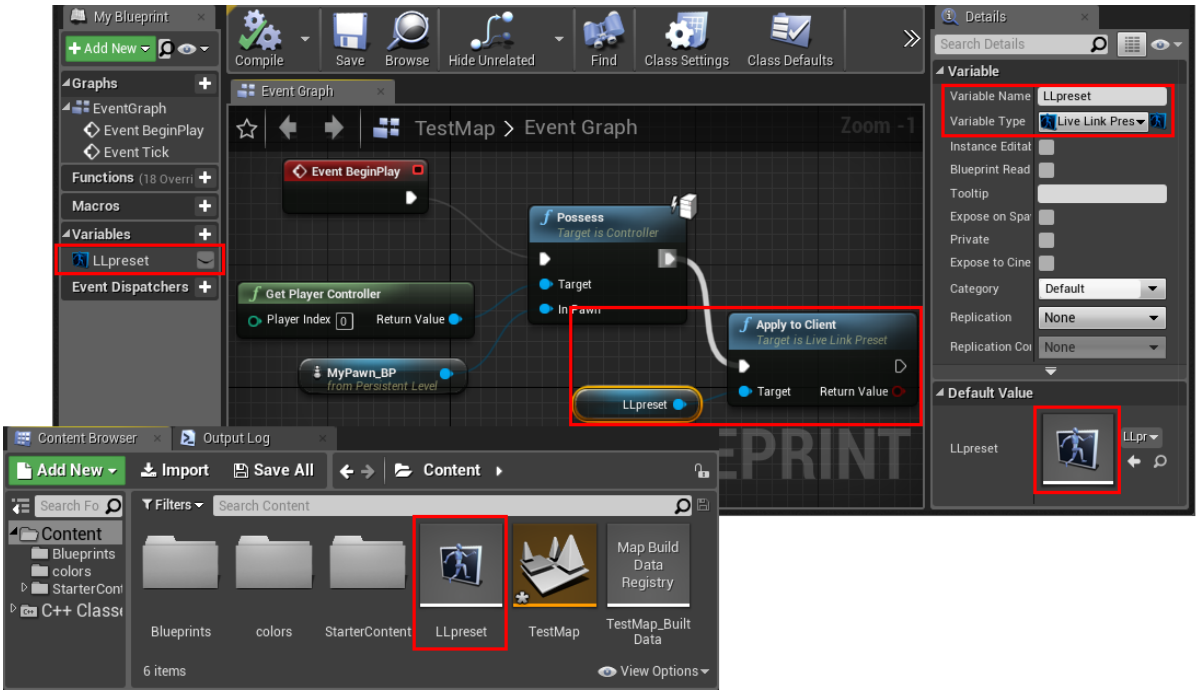


Preparation of Shipping and Development Builds

The procedure above assumes you manually configure DTrack as a Live Link source each time you start *Unreal Editor*. For shipping or developement builds you can automate this step by first creating a *Live Link Preset* in the *Window* > *Live Link* dialog:



Next you apply this preset in the *Level Blueprint*: Create a variable of type *Live Link Preset* and compile the Blueprint. The *Details* tab then offers you a slot for the default value, which you set to the saved preset. You then connect an *Apply to Client* node to the execution path of the *Begin Play* event, with the preset as target.



Note: Currently the Flystick (buttons and joystick) to work correctly with packaged builds requires to first start the game, and then the DTrack measurement. Make sure that DTrack measurements are stopped before starting the game.

Note: If the project cannot be opened via the Launcher, but only via the Editors file menu, you presumably have to adjust the *.uproject* file to the version of your *Unreal Editor*. Follow the corresponding installation step in *UnrealDTrackPlugin*.