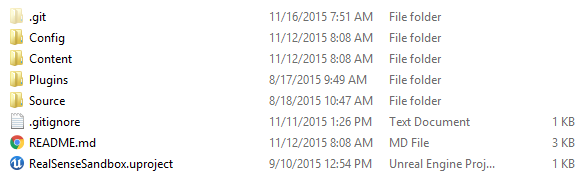
UE4 RealSense Plugin Getting Started Guide

This guide presumes that you have already installed Visual Studio, Unreal Engine 4.8 (or later), the RealSense SDK (R4 or later) and the RealSense DCM(s) for whichever camera(s) you wish to use. If you do not, please install these programs before proceeding.

Step 1: Getting the code

1. You can find the plugin source code and a sample project at  
     
   <http://10.23.235.23/gdouglas/UnrealsensePlugin>  
     
   in the master branch. Other branches represent in-progress, experimental features that may not yet be usable.
2. Clone the git repository and move into the directory where you cloned it. You’ll find a standard UE4 project that looks like this:



1. Right-click the .uproject file and select “Generate Visual Studio project files”. This will generate a Visual Studio .sln solution file.

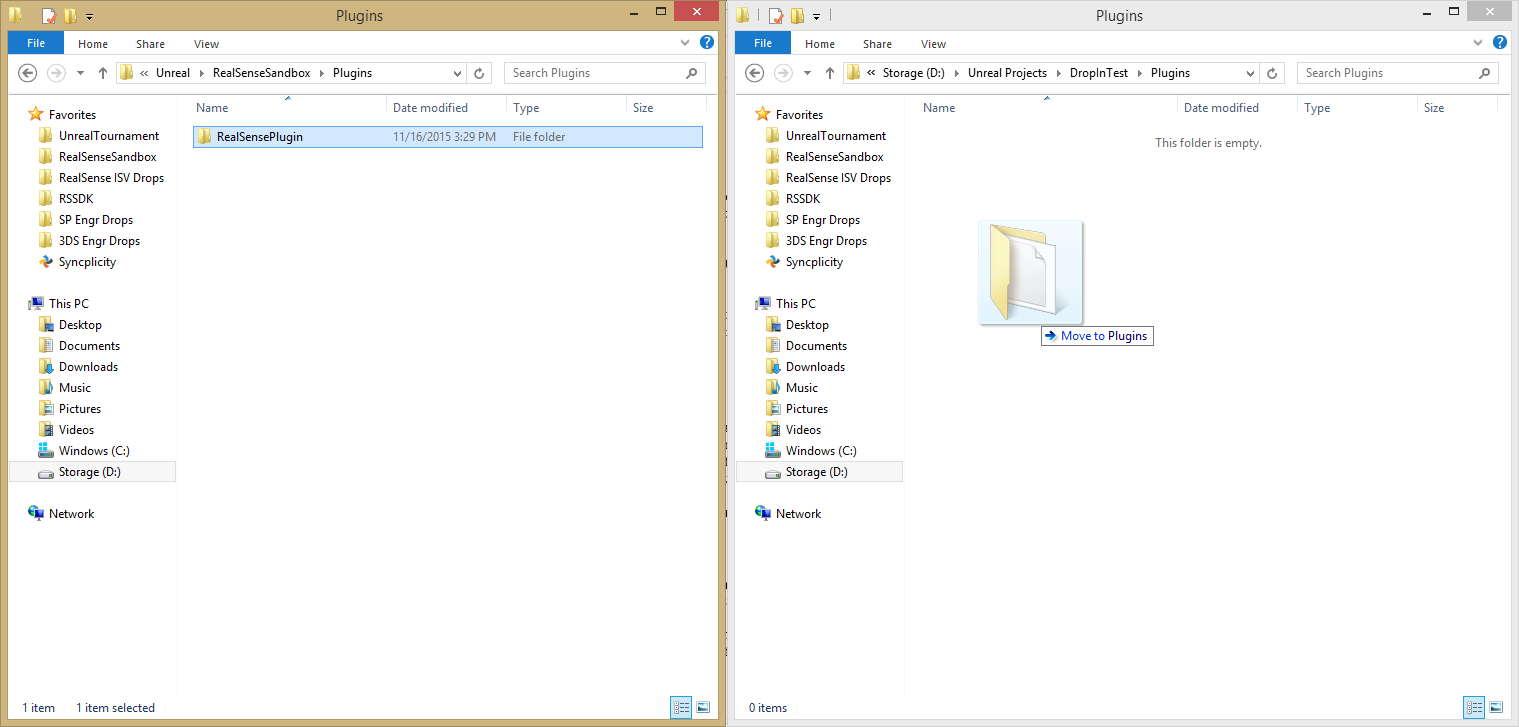
Step 2: Building the sample project

1. Open the project in Visual Studio and build the solution (Ctrl + Shift + B). This should only take a couple of minutes.
2. When it has successfully finished building, you may run the debugger (F5) or double-click the .uproject file in the top-level project directory.

Step 3: Adding the plugin to a new project

For the purposes of this guide, I will walk through the process for adding the plugin to a brand-new project. If you have an existing project you want to add the plugin too, you can skip to #3.

1. Start up the Epic Games Launcher, go to the library tab, and launch the UE4 Editor.
2. Create a new C++ Basic Code project (or use any of the C++ template projects you like). This will open your new project in Visual Studio and the UE4 Editor. For now, close both of these applications.
3. In the file explorer, navigate to your new project directory and add a top-level folder called “Plugins”.
4. Copy and paste the plugin from the sample project into this folder. You can find the plugin in the sample project in the Plugins/ folder, called “RealSensePlugin”.



1. The plugin will not automatically appear in the Visual Studio solution file that was automatically generated for your project. To fix this, right-click the .uproject file for your new project and select “Generate Visual Studio project files”. This will overwrite the existing .sln file, and when you open the new solution, you will find the RealSensePlugin in the solution explorer.
2. Build the solution (as before) and either run the debugger or open up your .uproject file.

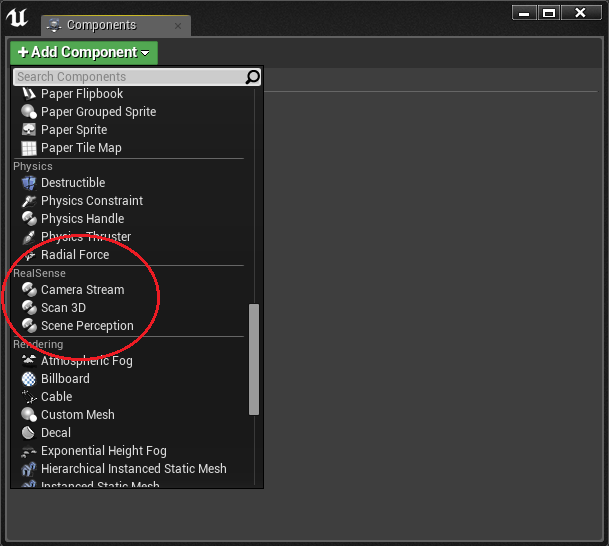
Step 4: Using the plugin

This guide is not meant to enumerate all of the possible ways you could use the RealSensePlugin. Nor is it designed to walk you through a step-by-step tutorial on using the plugin in Blueprints.

I will, however, describe some common steps you will need to repeat to add some basic RealSense functionality. For a more in-depth tutorial, check out the video here:

**COMING SOON!**

1. With your project that you created in the last step open in the UE4 Editor, create a new Blueprint Actor.
2. Open that actor in the editor, and check out the Components windows. If this window is not visible, go to the “Window” tab on the file menu and make sure “Components” is selected.
3. Click the “Add Component” button and scroll down to find the category of components labeled “RealSense”.



**Architecture Description**

The RealSense plugin is architected as a group of components that enable different features of the RealSense SDK. From now on, this guide will use the term “RealSense component” to refer to any component found underneath this category.

When a RealSense component is added to an actor in your level (regardless of its visibility in the scene), a “RealSenseSessionManager” actor is automatically added to the scene. Its purpose is to be the control center for RealSense data so that multiple RealSense components all share the same state.

Note: You should never need to manually add the RealSenseSessionManager actor to your scene. And if you add more than one RealSense component, there will still only be one RealSenseSessionManager created.

**Shared properties of all RealSense Components**

Each RealSense component shares some basic functionality. They all contain the following camera properties:

* Field of view of the color camera
* Field of view of the depth camera
* Model of the currently connected RealSense camera (F200, R200, SR300)
* Firmware version of the currently connected RealSense camera

And the following functions:

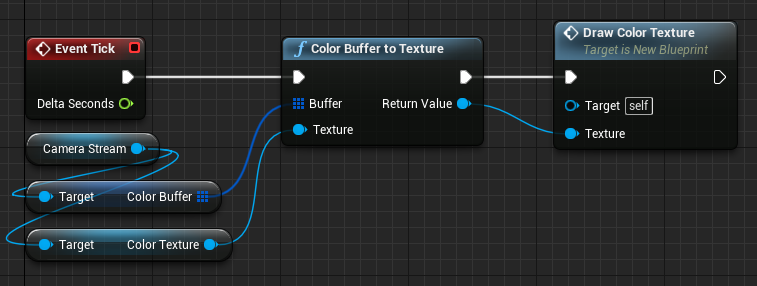
* StartCamera()
  + Turns on the RealSense camera
* StopCamera()
  + Turns off the RealSense camera
* IsCameraRunning()
  + Returns true if the camera is currently running (between calls to StartCamera and StopCamera)
* GetColorCameraResolution() / SetColorCameraResolution()
* GetDepthCameraResolution() / SetDepthCameraResolution()
* IsStreamSetValid()
  + Takes as input a Color and Depth Camera Resolution and returns true if the combination is a valid.
  + The RealSense SDK documentation has more information on valid sets of resolutions.

**Simple Usage**

To use one of the RealSense components, you must set the color and depth camera resolutions you want to use (see below for a chart of some common resolutions) and then call StartCamera().

Note: The current implementation of the plugin does not support calling StartCamera() in the BeginPlay() event callback.

The RealSense components are designed to automatically update their properties on every tick without requiring the developer to call any getters.



For example, as you can see in this Blueprint, you can access the Color Buffer every tick from the Camera Streams component. This buffer will automatically update with new data whenever it is available, without impacting the performance of the main game thread.

The helper function “Color Buffer to Texture” can be found in the “RealSense Utilities” category if you right-click inside the event graph.

For more detailed usage of the CameraStreams Component, check out the Camera Streams map in the sample project.

For more detailed usage of the Scan3D Component, check out the Scan3D map in the sample project.