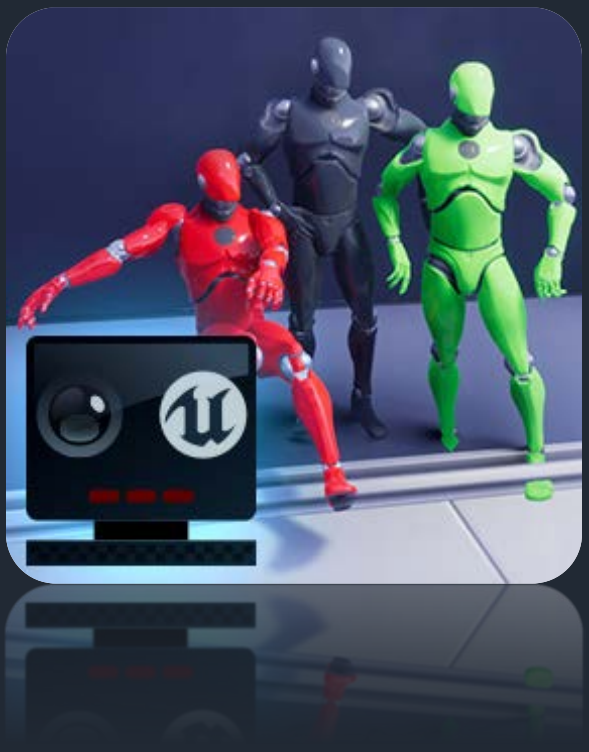


NeoKinect

Quick Start Guide

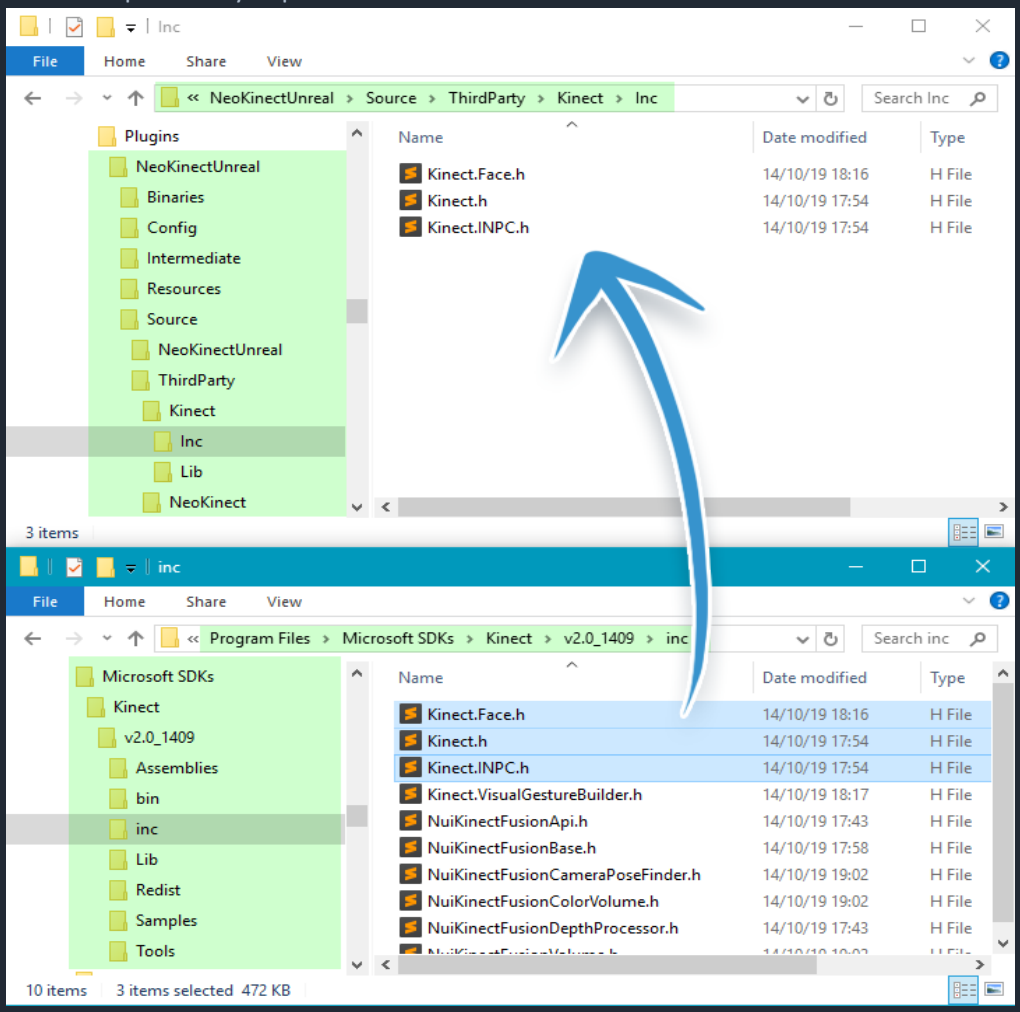
An Unreal Engine Kinect plugin

Developed by Rodrigo Villani

0. Essential MUST DO steps

After installing the plugin from the **Marketplace** there are some needed files that, for license reasons, could not be included in the download. Fortunately, you can get them from the **Kinect SDK** installation folders. Without these files, projects will fail to package. To get those files into the plugin folder structure:

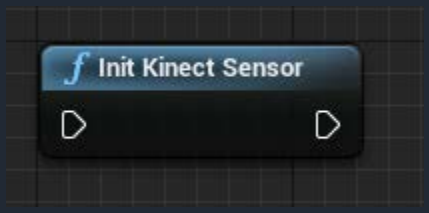
- Make sure you've already installed the **Kinect SDK** from Microsoft
- Navigate to C:\Program Files\Microsoft SDKs\Kinect\v2.xxx\inc
- Select and copy the files **Kinect.Face.h**, **Kinect.h** and **Kinect.INPC.h**
- Navigate to [Unreal installation]\Engine\Plugins\Marketplace\NeoKinectUnreal\Source\ThirdParty\Kinect\Inc
- Paste the previously copied files into that folder



You must do it only once for each Unreal Engine version you add the plugin to. Then, to prevent crashes in the Editor and packaged projects, in every project you enable the **NeoKinect** plugin, remember to:

- Navigate to C:\Program Files\Microsoft SDKs\Kinect\v2.xxx\Redist\Face\x64
- Copy the **NuiDatabase** folder and **Kinect20.Face.dll** file
- In the project folder, go inside Binaries\Win64 (create those folders if not existent) and paste the previously copied folder and file
- In a packaged project, paste those same files into [Package folder]\[Project Name]\Binaries\Win64. It is the same folder where your packaged exe resides.

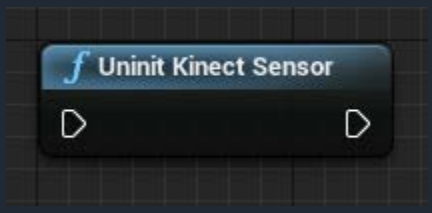
If you fail to follow these instructions, your editor or packaged project will crash as soon as it calls any of the plugin's functions.



1. Starting the Kinect sensor

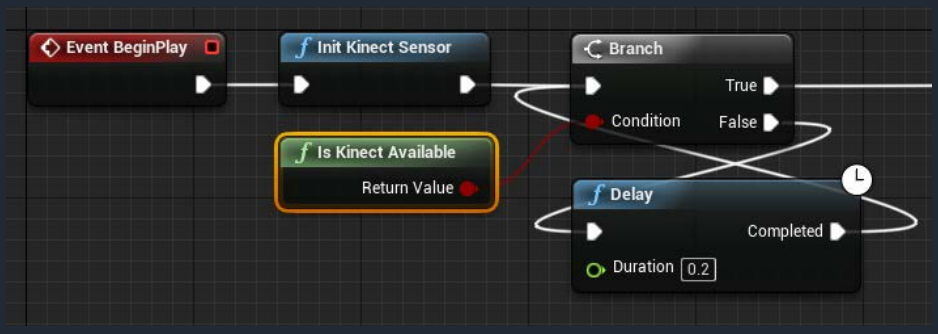
Use the **Init Kinect Sensor** node to start the sensor. It must be called before any calls to Frame functions, body tracking or any of the plugin's functionalities.

After this call, if the sensor is connected and working, it'll be in use until you Uninitialize it or quit the game. Inside the Unreal Editor, if not uninitialized, the sensor will be on until the Editor is closed.



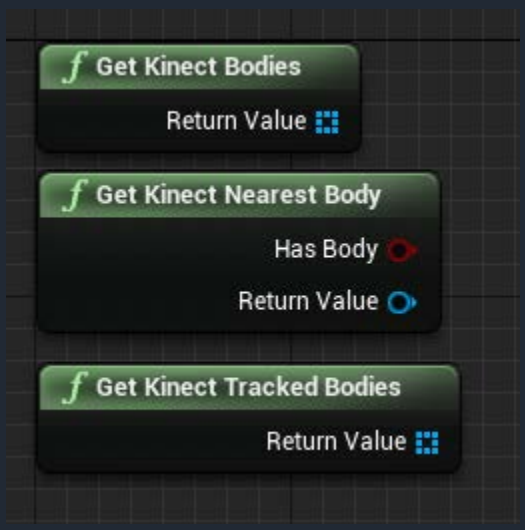
2. Turning the Kinect sensor off

Use the **Uninit Kinect Sensor** to turn it off. This will invalidate all Frame textures and the bodies and faces arrays. Use it when you are sure you will not need the sensor data anymore, like on Quit or an End Play event.



3. Checking sensor status

After calling Init Kinect Sensor you can use the **Is Kinect Available** node to both check when the sensor is really on and, after a while, if it's still returning false, display an error message to the player warning that the sensor is not initializing. If the later happens, it can be because the sensor is disconnected or malfunctioning.

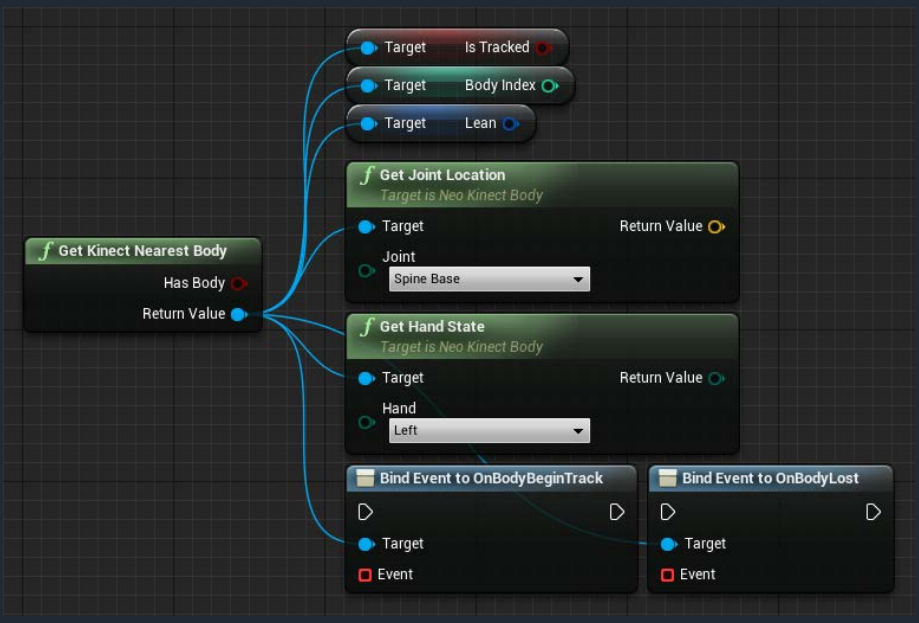


4. Getting tracked bodies data

After initializing the sensor you can store the **Get Kinect Bodies** node returned array to keep track of all 6 trackable bodies. The array is updated automatically, so you only need to save it in a variable and use that variable to check on the bodies' stats. The bodies in this array represent both tracked and untracked bodies (that may be tracked sometime).

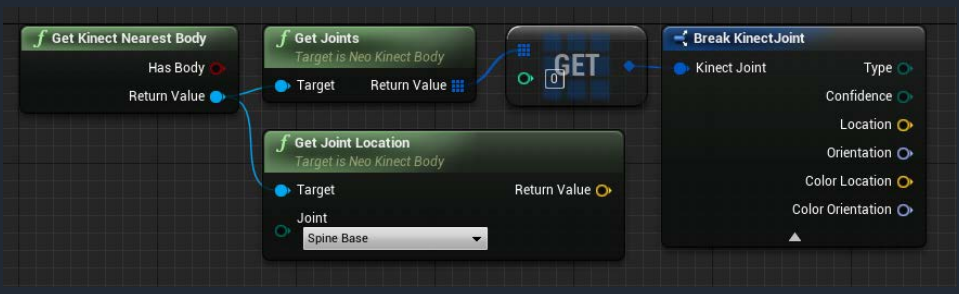
The **Get Kinect Nearest Body** node is an utility to always get the user body data whose hip is closest to the sensor, if any user is being tracked.

The **Get Kinect Tracked Bodies** node will return an array containing only the currently tracked bodies. Unlike the array from Get Kinect Bodies, it's not automatically updated, so there's no use in storing it in a variable for later access. If no bodies are being tracked, it'll be an empty array.



4.1. Body properties, functions and events

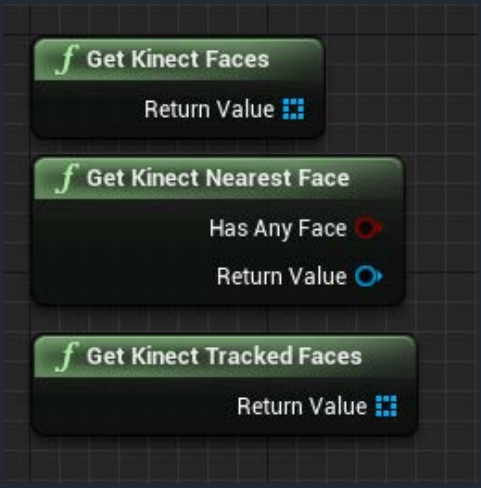
Among several things, you can check if a body is currently tracked, the body index (0~1), its Lean amount, Joints data (location, orientation, distance between joints and more), Hands states/poses and bind to tracking events, so you can execute actions when a body was just found or lost.



4.2. Joints tracking data

From a tracked body you can get all joints data from an array or specific joints status directly.

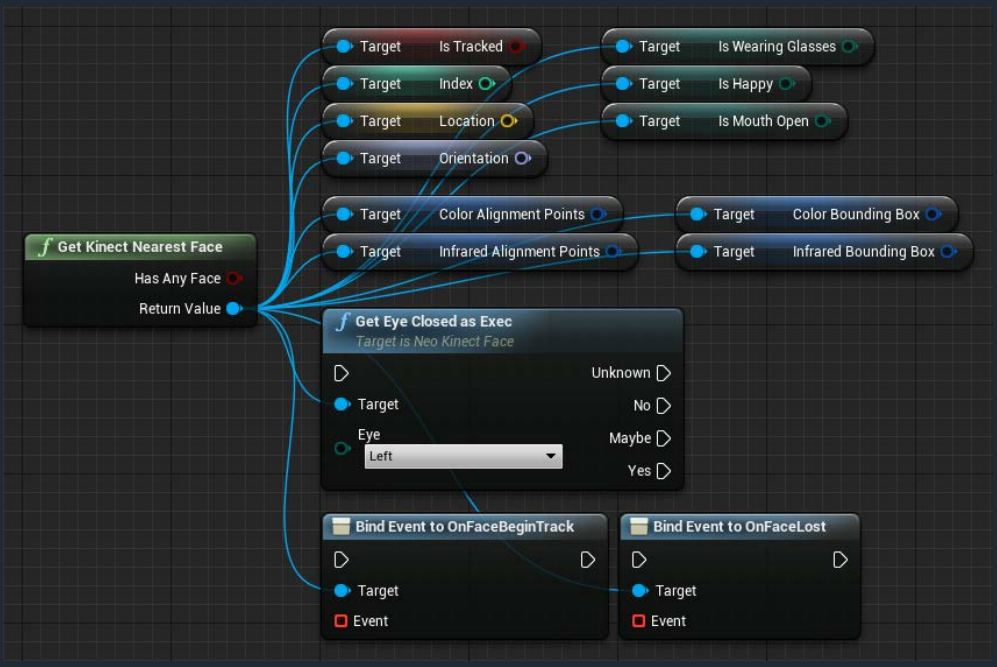
Color Location and **Color Orientation** are special coordinates remapped to fit on top of the Color Frame when viewed from the sensor's point of view with the Color Frame as background. This facilitates AR applications development. The calculations for these re-projected transforms are a bit heavy thus, disabled by default and returning zeroed values. In order to enable them, use the **Set Use Joints Color Space Transforms** node. This also enables Color projected transforms for faces tracking.



5. Tracked faces data

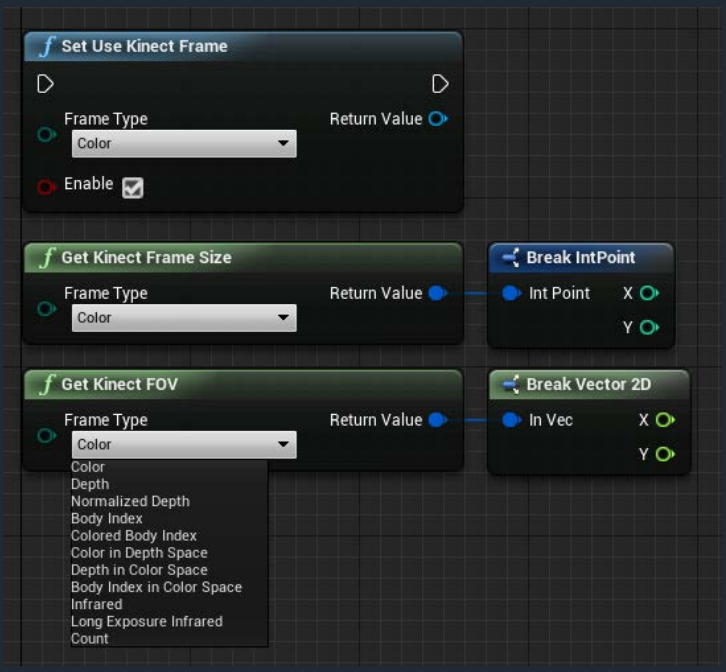
Similar to the body functions, the face functions give you access to the sensor's tracked faces data.

Remember: head orientation is only well tracked when the user is close to the sensor. That's not a plugin flaw, but a Kinect limitation.



5.1. Faces properties, functions and events

Same as with bodies, you have access to all of the Kinect face tracking data through variables, functions and events accessed from a KinectFace object (returned from the Tracked Faces functions mentioned above).

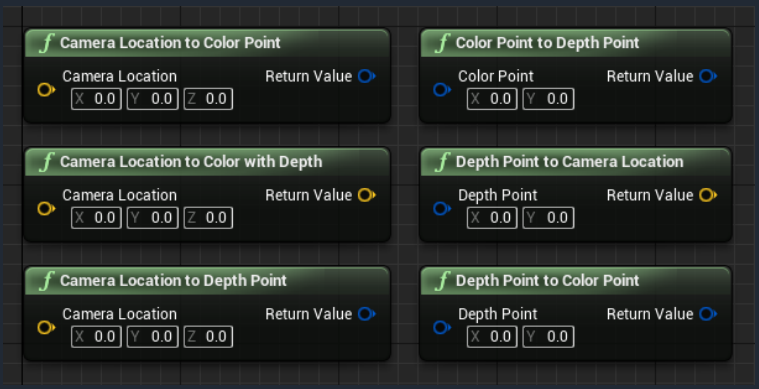


6. Kinect Frames access

Frames are the Kinect sensors' captured images. You can access them in Unreal in the form of Textures and they are updated automatically, so the only thing you need to do is call the **Set Use Kinect Frame** node. It returns the Texture for you to use in materials and widgets.

You can also get Frame size and FOV if needed.

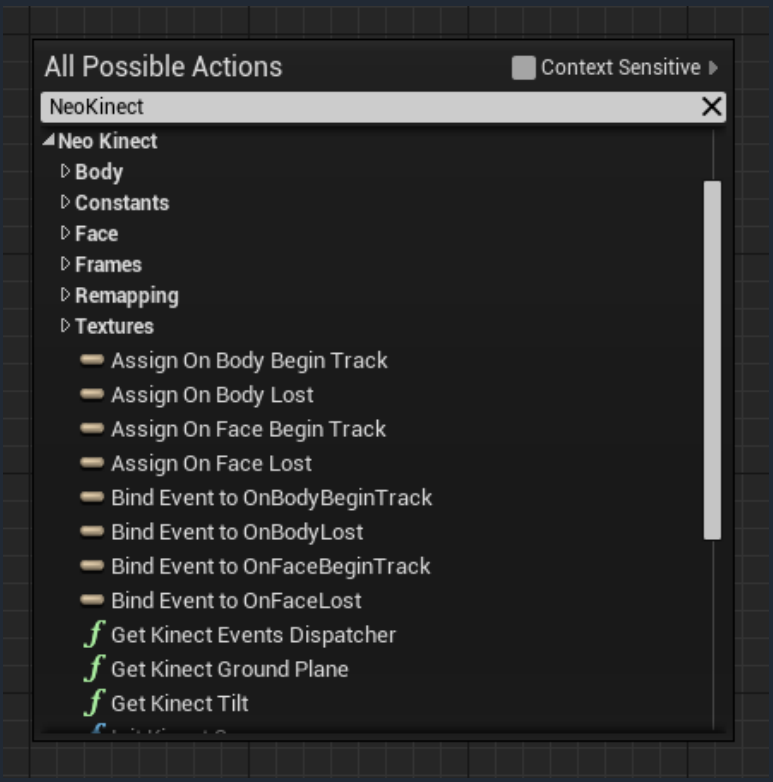
Some frames are remapped versions of others or different representations. Some nice ones are the **Normalized Depth**, where you can see the distance of real world stuff in a white to black range and the **Body Index in Color Space**, which you can overlay on top of the Color frame to highlight detected users.



7. Coordinate remapping

The plugin gives you access to all of the Microsoft's Kinect API remapping methods as well as some more advanced ones, like **Camera Location to Color with Depth**. It remaps a Camera space location (a 3D location relative to the sensor, X being positive in the sensor forward side) into another 3D location that matches the Color frame. For instance, if you get the elbow joint location and pass it through this node, you'll have it remapped to fit on top of the Color Frame texture if that was used as background and the camera was aligned with the sensor. An example of that application can be found in the plugin's content examples.

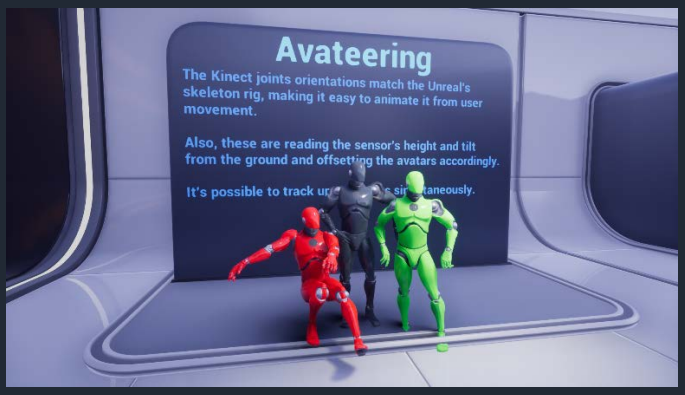
Find and Learn more



You can find many more nodes by searching for **NeoKinect** in the Bluprint Graph context menu.

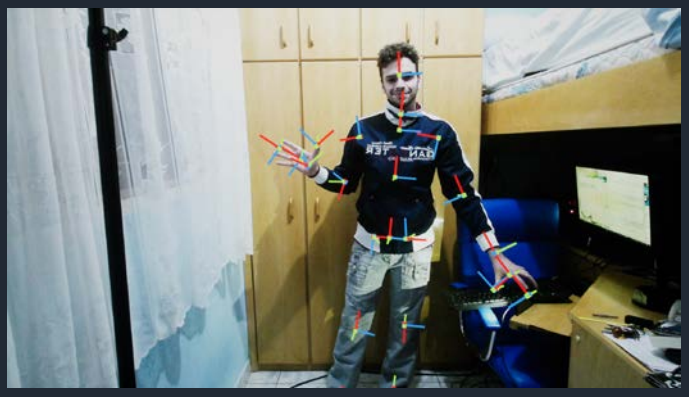
All nodes have explanations of how they work when you hover the mouse over them or any of their parameter properties.

There's also an **example project** you can download here: http://files.rvillani.com/unreal/NeoKinectExamples.zip
The examples are:



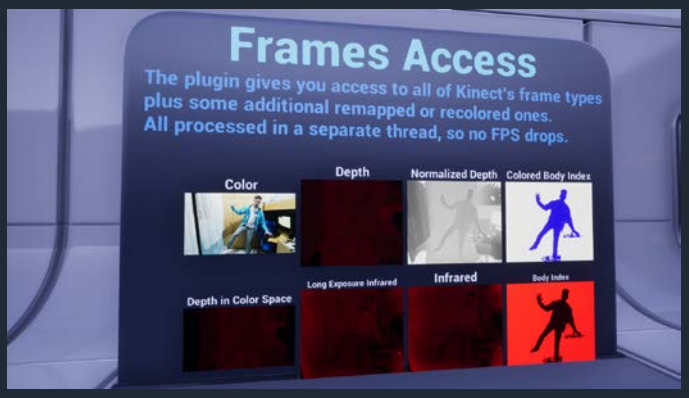
Avateering

Shows how to make a Skeletal Mesh, with the same bone setup as Unreal Engine's Mannequin, follow up to 6 users' movements.



Color Mapping

See how to project 3D objects on top of a tracked user body's joints in the Color Frame.



Frames Access

Check the **BP_KinectFrame** Blueprint to see how live Frame textures can be used in meshes inside the game.

Face Tracking

It is an **AR** example that applies a helmet on up to 6 users' faces. The eyes on the helmet glow only when its user's eyes are open.

Frame Textures on Widgets

Shows how to apply the Kinect frame textures on Widget Images via Blueprint.