How to Use Kinect v2 Examples with MS-SDK

1. (Kinect-v2) Download and install Kinect-v2 SDK as described in the next section.
2. (Kinect-v2) If you want to use the Kinect-v2 speech recognition, download and install the Speech Platform Runtime or SDK, as well as the needed language packs, as described in the next section.
3. (Nuitrack or Orbbec) If you want to work with Nuitrack body tracking SDK, look at this tip. If you want to work with Orbbec Astra or Astra-Pro sensors via OpenNI2, look at this tip.
4. Import this package into new Unity project.
5. Open 'File / Build settings' and switch to 'PC, Mac & Linux Standalone', Target platform: 'Windows'.
6. Make sure that Direct3D11 is the first option in the 'Auto Graphics API for Windows'-list setting, in 'Player Settings / Other Settings / Rendering'.
7. Open and run a demo scene of your choice from a subfolder of the 'K2Examples/KinectDemos'-folder. See the short descriptions of all demo scenes here.

Installation of Kinect v2 SDK

1. Download the Kinect for Windows SDK 2.0. Here is the download page: http://www.microsoft.com/en-us/download/details.aspx?id=44561
2. Run the installer. Installation of Kinect SDK 2.0 is pretty straightforward.
3. Connect the Kinect-v2 sensor. The needed drivers will be installed automatically.
4. If you want to use the Kinect-v2 speech recognition, download and install the MS Speech Platform Runtime v11 (or Speech Platform SDK v11). Install both x86 and x64-packages, to be on the safe side. Here is the download page: http://www.microsoft.com/en-us/download/details.aspx?id=27225
5. For the Kinect-v2 speech recognition, you also need to download and install the respective language pack. Here is the download page: https://www.microsoft.com/en-us/download/details.aspx?id=43662

Short Descriptions of the Available Demo Scenes

The short descriptions of all available demo scenes can be found in the K2-asset online documentation.

How to Reuse the Kinect-related Scripts in Your Own Unity Project (Windows Builds)

1. Copy folder 'KinectScripts' from the K2Examples-folder of this package to your project. This folder contains all needed components, scripts, filters and interfaces.
2. Copy folder 'Resources' from the K2Examples-folder of this package to your project. This folder contains the needed libraries and resources. You may not copy the libraries you don't think you need.
3. Copy folder 'Standard Assets' from the Assets-folder of this package to the Assets-folder of your project. It contains the wrapper classes for Kinect-v2.
4. Wait until Unity detects, imports and compiles the newly detected resources and scripts.

5. In your scene, create a KinectController-game object and add the 'KinectManager'-component to it. This is the most basic, required component in all Kinect-related scenes.
6. Add the 'AvatarController'-component to each avatar model in the scene that should be controlled by the Kinect-sensor. See the next section for more information on the AvatarController-component. If needed, utilize the other script components from KinectScripts-folder, as well.
7. You may use the public API of the Kinect-related components in your scripts. Here is the online API documentation: https://ratemt.com/k2gpapi/annotated.html

Why Are There Two Avatars in KinectAvatarsDemo1-scene

The presence of the two avatars (humanoid characters) in the scene is to demonstrate different options of their AvatarController-components. AvatarController is the component that controls the motion of the avatar model.

Look at the right avatar. It mirrors your movements (your left arm is his right one, etc.), and moves with respect to the main camera, the same way as you move with respect to the sensor. Its transform's Y-rotation is set to 180 degrees, which means the model is turned to you (just as in a mirror). The 'Mirrored Movement'-setting of its AvatarController-component is enabled, too. Its 'Pos relative to camera'-setting references the main camera, to make it the origin of movements' coordinate system.

The left avatar (the one that is turned with his back at you) is not mirrored. It follows your movements as they are. Your left is its left and your right is its right. Its transform Y-rotation is set to 0 (it is turned backwards) and the 'Mirrored Movement'-setting of its AvatarController is disabled. Its 'Pos relative to camera'-setting is None. This makes it move with respect to its initial position.

Pay special attention to the 'Player index'-setting of both avatars. The same setting may be found in many other Kinect-related components. It specifies the index of the user, who controls the component. 0 means the 1st user, 1 – the 2nd one, 2 – the 3rd one, etc. If you want the avatar to be controlled by other than the first detected user, change the 'Player index'-setting of its AvatarController-component accordingly.

Additional Reading

The following how-to tutorials are also located in the K2Examples/_Readme-folder of this Unity-package:

1. Howto-Use-Gestures-or-Create-Your-Own-Ones.pdf
2. Howto-Use-KinectManager-Across-Multiple-Scenes.pdfs

More Information, Support and Feedback

Online Documentation: https://ratemt.com/k2docs/
Tip and Tricks: http://rfilkov.com/2015/01/25/kinect-v2-tips-tricks-examples/
Web: http://rfilkov.com/2014/08/01/kinect-v2-with-ms-sdk/

Contact: http://rfilkov.com/about/#contact (please mention your invoice number from Asset store)
Twitter: https://twitter.com/roumenf