## How to Run the Example

1. Download and install Kinect v2 SDK as described in the next section.
2. Open scene 'KinectAvatarsDemo', located in Assets/AvatarsDemo-folder.
3. Run the scene. Move around to see how the avatars and the cube-man reflect your movements.
4. Use your left or right hand to control the hand-cursor on the screen.
5. Try one or more of the suggested gestures and make sure they are detected correctly.
6. Open and run 'KinectGesturesDemo'-scene, located in Assets/GesturesDemo-folder. Use hand swipes – left or right - to turn the presentation cube left or right.
7. Open and run 'KinectInteractionDemo'-scene, located in Assets/InteractionDemo-folder. Use hand grip to grab an object and then drag it around. Open your hand to release the object. Try dragging and dropping objects with your right hand and with your left hand.
8. Open and run 'KinectOverlayDemo'-scene, located in Assets/OverlayDemo-folder. Watch how the green ball follows the position of your right hand on the video screen.
9. Open and run 'KinectFaceTrackingDemo'-scene, located in Assets/FaceTrackingDemo-folder. Look how the head model on screen mirrors your face expression and head position.

## 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/Runtime is simple and straightforward.
3. Connect the Kinect v2 sensor. The needed drivers are installed automatically.

## Why There Are Two Avatars in the Scene

The meaning of the two avatars (3D humanoid characters) in the scene is to demonstrate that you can have both – mirrored and non-mirrored movements.

First, you can have an avatar that mirrors your movement. This is the one facing you in the example scene. As you can see, its transform has Y-rotation (rotation around Y-axis) set to 180 degrees. Also, there is an AvatarController-component, attached to the avatar's game object and its 'Mirrored Movement'-parameter is enabled. Mirrored movement means that when you, for instance, lift your left hand the avatar lifts his right hand and vice versa, like in a mirror.

The second avatar, the one that has his back turned at you, is not mirrored. It reproduces your movements exactly as they are. Your left is his left and your right is his right. See it that way - you're also staying with your back turned to the main camera. Its transform has Y-rotation set to 0 and the 'Mirrored Movement'-parameter of its AvatarController is disabled.

In order to get correct avatar's position and movement, first set the position and rotation of the avatar's game object in the scene, as needed. Then attach AvatarController-component to the avatar's game object and set its 'Mirrored Movement'-parameter accordingly.

## How to Reuse the Kinect-Example in Your Own Unity Project

1. Copy folder 'KinectScripts' from the Assets-folder of the example to the Assets-folder of your project. This folder contains all needed scripts, filters and interfaces.
2. Copy folders 'Resources' and 'Standard Assets' from the Assets-folder of the example to the Assets-folder of your project. These folders contain the needed libraries, wrapper classes and resources.
3. Wait until Unity detects and compiles the newly copied resources and scripts.
4. Add 'AvatarController'-component to each avatar (humanoid character) in the scene that you need to control with the Kinect-sensor.
5. Disable 'Mirrored Movement'-parameter of the AvatarController, if the avatar should move in the same direction as the user. Enable it, if the avatar should mirror user's movements.
6. Add 'KinectManager'-component to the MainCamera. If you use multiple cameras, create an empty game object and add the KinectManager-component to it.
7. (Optional) Drag and drop the avatars' game objects from Hierarchy to the 'Avatar Controllers'-list parameter of KinectManager. Otherwise they will be detected and added automatically to the list at the scene's start-up.
8. Enable 'Compute User Map' and 'Display User Map'-parameters, if you want to see the user-depth map on screen. Enable 'Compute Color Map' and 'Display Color Map'-parameters, if you want to see the color camera image on screen. Enable 'Display Skeleton Lines' parameter, if you want to see how Kinect tracks the skeletons on the user-depth map.
9. You can use the public functions of 'KinectManager' and 'InteractionManager' in your scripts. As examples, see 'GestureListener.cs' and 'PresentationScript.cs' used by KinectGesturesDemo-scene, 'GrabDropScript.cs' used by KinectInteractionDemo-scene, 'KinectOverlayer.cs' used by KinectOverlayDemo-scene or 'ModelFaceController.cs' used by KinectFaceTrackingDemo-scene.

## Additional Reading

The following how-to tutorials are also located in the Assets-folder of the example Unity-package:

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

## Support and Feedback

E-mail: rumen.filkov@gmail.com, Skype, Twitter: roumenf, Whats App: on request