How to Run the Example

1. Download and install Kinect SDK 1.8, 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. Try to control the hand-cursor on the screen with your right or left hand.
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 'KinectOverlayDemo'-scene, located in Assets/OverlayDemo-folder. Watch how the green ball follows the position of your right hand on the screen.
8. Open and run 'DepthColliderDemo'-scene, located in Assets/DepthColliderDemo-folder. Try to hit the falling eggs with your hands, arms and other body parts.

Installation of Kinect Sensor with MS SDK 1.8

1. Download the Kinect SDK 1.8 or Kinect Windows Runtime 1.8. Here is the download page: http://www.microsoft.com/en-us/download/details.aspx?id=40278
2. Run the installer. Installation of Kinect SDK/Runtime is simple and straightforward.
3. Connect the Kinect sensor. The needed drivers will be 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 Assets-folder of the example to the Assets-folder of your project. This folder contains the needed scripts and optional filters.
2. Open Unity editor and wait until Unity detects and compiles the new Kinect scripts.
3. Add 'AvatarController'-component to each avatar (humanoid character) in the scene that you need to control with the Kinect-sensor.
4. Drag and drop the appropriate bones of the avatar's skeleton from Hierarchy to the appropriate joint-variables (Transforms) of 'AvatarController'-component in the Inspector.
5. Disable 'Mirrored Movement'-parameter of AvatarController, if the avatar should move in the same direction as the user. Enable 'Mirrored Movement', 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 'Player 1 Avatars' list-parameter of KinectManager. Otherwise they will be detected and added automatically to the list at the scene's start-up.
8. If you need a 2nd Kinect-user, enable 'Two Users'-parameter of 'KinectManager'. In this case, repeat steps 4-5 for each avatar, controlled by the 2nd user. Then go back to step 7, but this time use the 'Player 2 Avatars' list-parameter of KinectManager.
9. 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.
10. You can use the public functions of 'KinectManager'-script in your scripts. Examples: 'GestureListener.cs' and 'PresentationScript.cs' used by the KinectGesturesDemo-scene, 'KinectOverlayer.cs' used by the KinectOverlayDemo-scene or 'DepthImageViewer.cs' used by the DepthColliderDemo-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
Web: http://rfilkov.com
Skype: roumenf
Twitter: roumenf