How to Run the Example

1. Download and install Kinect SDK as described in the next section, if you haven't done it already.
2. Open scene 'KinectAvatarsDemo', located in Assets/AvatarsDemo-folder.
3. Run the scene. Both avatars are connected to the 1st Kinect player.
4. Try to change some parameters of KinectManager-script, attached to 'MainCamera' or AvatarController, attached to 'U_Character_REF' avatars, and then to re-run the scene.
5. Open and run 'KinectGesturesDemo'-scene, located in Assets/GesturesDemo-folder

Installation of Kinect Sensor with MS SDK

1. Download the Kinect SDK or Kinect Windows Runtime. Here is the download page: http://www.microsoft.com/en-us/kinectforwindowsdev/Downloads.aspx
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.

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. Wait until Unity detects and compiles the new Kinect scripts.
3. Add 'AvatarController'-script to each avatar (humanoid character) in your game 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'-script in the Inspector.
5. Uncheck 'Mirrored Movement', if the avatar should move in the same direction as the user. Check it, if the avatar should mirror user movements
6. Add 'KinectManager'-script to the MainCamera. If you use multiple cameras, create an empty GameObject and add the script to it. KinectManager's Start()-method initializes Kinect sensor, Update()-method updates positions of all Kinect-controlled avatars.
7. Optional: Drag and drop the avatars from Hierarchy to the 'Player 1 Avatars' list.
8. If you need a 2nd Kinect-user to control avatars, check 'Two Users' in the parameters of 'KinectManager'-Script in the Inspector. If this is the case, repeat steps 4-5 for each avatar, controlled by the 2nd user. Then repeat step 7, but this time for 'Player 2 Avatars' list.
9. Check 'Compute User Map' and 'Display User Map'-checkboxes, if you want to see the User-depth Map after the user calibration completes.
10. Use the public functions of 'KinectManager'-script in your scripts. As an example, take a look at 'AvatarController.cs' or at 'GesturesDemoScript.cs', used by KinectGesturesDemo-scene.

Gestures

The following gestures are currently recognized:

- *RaiseRightHand / RaiseLeftHand* – left or right hand is raised over the shoulder and stays so for at least 1.0 second.
- *Psi* – both hands are raised over the shoulder and the user stays in this pose for 1.0 seconds.
- *Stop* – both hands are below the waist.
- *Wave* – right hand is waved left and then back right, or left hand is waved right and then back left.
- *SwipeLeft* – right hand swipes left.
- *SwipeRight* – left hand swipes right.
- *SwipeUp / SwipeDown* – swipe up or down with left or right hand
- *Click* – left or right hand stays in place for at least 2.5s. Useful in combination with cursor control.
- *RightHandCursor / LeftHandCursor* – pseudo gesture, used to provide cursor movement with the right or left hand.
- *ZoomOut* – left and right hands are together and above the elbows at the beginning, then the hands move in different directions.
- *ZoomIn* - left and right hands are at least 0.7 meter apart and above the elbows at the beginning, then the hands get closer to each other.
- *Wheel* - left and right hands are less than 0.7 meter apart and above the elbows at the beginning, then the hands start to turn an imaginary wheel left (positive) or right (negative).
- *Jump* – the hip center gets at least 10cm above its last position within 1.5 seconds.
- *Squat* - the hip center gets at least 10cm below its last position within 1.5 seconds
- *Push* – push/punch forward with left or right hand within 1.5 seconds
- *Pull* - pull backward with left or right hand within 1.5 seconds

How to Add Gesture Detection

To add gestures to be detected, first look at the KinectManager-component of MainCamera. There are two collections - “Player1 Gestures” and “Player2 Gestures”. For a start use “Player1 Gestures”. Set the list size to the number of gestures you want to detect, and then select the individual gestures. As you can see, there are already some sample gestures defined in the example.

If you want to stop the cursor control and Click detection, remove the “RightHandCursor”, “LeftHandCursor” and “Click”-gestures in “Player1 Gestures”-collection.

To handle a detected gesture, you need to implement KinectGestures.GestureListenerInterface. For an example look at the KinectScripts/Extras/SimpleGestureListener.cs-script. GestureInProgress()-function is invoked when a gesture is in progress, but not yet completed or cancelled. GestureCompleted() is invoked when the gesture is completed. You can add your own code there to handle the completed gestures. GestureCancelled() is invoked, if the gesture in progress was cancelled. UserDetected()-function can be used to start gesture detection programmatically, for instance, if a gesture needs to be only temporary detected. UserLost()-function can be used to clear variables or free allocated resources. You don't need to remove the programmatically added gestures explicitly, as they are removed before the invocation of UserLost().

References

This example is based on the following two examples from CMU.edu. A big "Thank you" to their authors:

- http://wiki.etc.cmu.edu/unity3d/index.php/Microsoft_Kinect_-_Open_NI
- http://wiki.etc.cmu.edu/unity3d/index.php/Microsoft_Kinect_-_Microsoft_SDK

Support and Feedback

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