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 'SensorKinectMsSDK', located in Assets-folder.
3. Run the scene. Both avatars are connected to the 1st Kinect user.
4. Try to change some parameters of the scripts, attached to 'MainCamera' and 'U_Character_REF' avatars, and then re-run the scene.

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/kinectforwindows/develop/developer-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 3 needed scripts – KinectWrapper.cs, KinectManager.cs and AvatarController.cs
2. Wait until Unity detects and compiles the new Kinect scripts.
3. Add script 'AvatarController' 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 a specific GameObject and add the script to it. Script's Start()-method initializes Kinect SDK, Update()-method updates positions of all Kinect-controlled avatars.
7. Set the number of Kinect-controlled avatars as Size of 'Player 1 Avatars'. Drag and drop the avatars from Hierarchy to the Element-slots of 'Player 1 Avatars'.
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 4-7 for each avatar, controlled by the 2nd user. Repeat step 8 too, but this time for 'Player 2 Avatars' collection.
9. Check 'User Map After Calibration', if you want to see the User-depth Map after the calibration has been complete. Check 'Near Mode' if you'd like to ignore the movements of the legs.
10. Save and run your game.

Gestures

The following gestures are currently recognized:

- RaiseHand – left or right hand is raised over the shoulder and stays so for 1.5 seconds.
- Psi – both hands are raised over the shoulder and the user stays in this pose for 1.5 seconds.
- Wave – right hand is waved left and then back right, or left hand is waved right and then back left.
- SweepLeft – right hand sweeps left.
- SweepRight – left hand sweeps right.
- Click – left or right hand moves forward and then back. Useful in combination with cursor control.
- RightHandCursor – pseudo gesture, used to move the cursor with the right hand.
- LeftHandCursor – pseudo gesture, used to move the cursor with the left hand.

How to Add Gesture Detection

Take a look at the AvatarController-script, function "SuccessfulCalibration()". You will see several "KinectManager.Instance.DetectGesture()" invocations. For instance, the line "KinectManager.Instance.DetectGesture(userId, KinectWrapper.Gestures.SweepLeft);" starts gesture detection of SweepLeft-gesture for the currently tracked user with ID=userId. Add similar lines for the gestures you want to recognize in your project.

The callback function "GestureInProgress()" is invoked when a gesture in progress has been detected. This function is useful for cursor or progress display. The function "GestureComplete()" is invoked when the gesture is complete. You can add your code there to handle the detected gestures.

If you want to stop the cursor and click control, comment out the following two lines:

- KinectManager.Instance.DetectGesture(userId, KinectWrapper.Gestures.RightHandCursor);
- KinectManager.Instance.DetectGesture(userId, KinectWrapper.Gestures.Click);

and disable HandCursor-GUITexture object in the Unity editor.

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:

E-mail: firu@fhv.at, Skype: roumenf