How to Use Gestures or Create Your Own Gestures

There are two ways to add gesture detection and gesture recognition to your Unity-project. For the first one look at KinectManager – a component of MainCamera in the example scene. There are two lists - "Player1 Gestures" (these are the gestures expected from player 1) and "Player2 Gestures" (expected from player 2). The gestures in these lists will be detected during the entire game.

The second way is to specify user specific gestures programmatically. To add such gestures or to handle any of the specified gestures (in 1. or 2. way), you need to implement KinectGestures.GestureListenerInterface. For an example look at the KinectScripts/Extras/SimpleGestureListener.cs-script. Here is a short description of its methods:

UserDetected() can be used to start gesture detection programmatically. UserLost() can be used to clear variables or to free the allocated resources. You don't need to remove the gestures added by UserDetected()-method explicitly. They are removed automatically, before the invocation of UserLost().

GestureInProgress()-method is invoked when a gesture is started, 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 is cancelled.

Currently Recognized 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.
- *Tpose* – the hands are to the sides, perpendicular to the body (T-pose), for 1.0 seconds.
- *Stop* – right hand is down and left hand is slightly to the side, but below the waist, or left hand is down and right hand is slightly to the side, but 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 to the front and put together at the beginning, then the hands move in different directions.
- *ZoomIn* - left and right hands are at least 0.7 meter apart and to the front at the beginning, then the hands get closer to each other.
- *Wheel* - left and right hands are shoulder size apart and to the front at the beginning, then the hands start to turn an imaginary wheel left (positive angle) or right (negative angle).
- *Jump* – the hip center gets at least 15cm above its last position within 1.5 seconds.

- *Squat* - the hip center gets at least 15cm below its last position within 1.5 seconds
- *Push* – push 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 Your Own Gestures

Here are some hints on how to add your own gestures to the Kinect gesture-detection procedure. You need some C# coding skills and a bit of basic understanding on how the sensor works. It reports the 3d-coordnates of the tracked body parts in the Kinect coordinate system, in meters.

To add detection of custom gesture, open Assets/KinectScripts/KinectGestures.cs. Then:

1. Find the Gestures-enum. First you need to add the name of your gesture at the end of this enum.
2. Find the CheckForGesture()-function. There is a long switch() there, and its cases process the detection of each gesture, defined in the Gestures-enum. You need to add a case for your gesture at the end of this switch(), near the end of the script. There you will implement the gesture detection.
3. For an example on how to do that, look at the processing of some simple gestures, like RaiseLeftHand, RaiseRightHand, SwipeLeft or SwipeRight.
4. As you see, each gesture has its own internal switch() to check and change the gesture's current state. Each gesture is like a state machine with numerical states (0, 1, 2, 3…). Its current state along with other data, is stored in an internal structure of type GestureData. This data-structure is created for each gesture that needs to be detected in the scene.
5. The initial state of each gesture is 0. At this state, the code needs to detect if the gesture is starting or not. To do this, it checks and stores the position of a joint, usually the left or right hand. If the joint position is suitable for a gesture start, it increments the state. At the next state, it checks if the joint has reached the needed position (or distance from the previous position), usually within a time interval, let's say within 1.0 - 1.5 seconds.
6. If the joint has reached its target position (or distance) within the time interval, the gesture is considered completed. Otherwise - it is considered cancelled. Then, the gesture state may be reset back to 0 and the gesture-detection procedure will start again.

To add detection of your own gestures, first try to understand how relatively simple gestures, like RaiseHand or Swipes, work. Then find a gesture similar to the one you need. Copy and modify its code to fit your needs. Hope this helps for a start ;)

Support, Examples and Feedback

E-mail: rumen.filkov@gmail.com;
Web: http://rfilkov.com
Skype: roumenf
Twitter: roumenf