How to Use Gestures or Create Your Own Gestures

There are two ways to add gesture detection to your Unity-project. The first one is easier and utilizes the KinectManager – a component of the KinectController in the example scenes. There are two list-settings, called “Player1 Gestures” for the gestures of Player #1 and “Player2 Gestures” for the gestures of Player #2. The gestures in these lists will be tracked during the entire game. You can use the Player1-Gestures list to test quickly the gestures you need.

The second way is to specify user or scene-specific gestures programmatically. You need to create a script that implements KinectGestures.GestureListenerInterface and use its methods to start the gesture tracking and for gesture processing. As an example, look at the KinectScripts/Samples/SimpleGestureListener.cs-script. Here is a short description of the GestureListenerInterface’s methods:

- UserDetected() – invoked when a new user is detected. Can be used to start the gesture tracking.
- UserLost() – invoked when a user is lost. It can be used to clean up or free the allocated resources. You don’t need to stop explicitly the detection of the gestures added in UserDetected().
- GestureInProgress() - invoked when a gesture start is detected, but the gesture is not yet completed or cancelled. Can be used to report the gesture progress or to process continuous gestures, like walking or running.
- GestureCompleted() - invoked when the gesture is completed. You can process the gesture detection here, and decide whether to reset the gesture (i.e. restart its detection) or not.
- GestureCancelled() - invoked, if the gesture is cancelled. The gesture is considered cancelled by the gesture detection routine, when it is not completed within the allowed time frame. You can decide, whether to reset the gesture (i.e. restart its detection) or not.

Currently Recognized Gestures

The following gestures are currently recognized:

- *RaiseRightHand / RaiseLeftHand* – left or right hand is raised above the shoulder and the user stays in this pose for 1.0 seconds.
- *Psi* – both hands are raised above 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 the left or right hand
- *Click* – left or right hand stays in place for at least 2.5 s. Useful in combination with cursor control.
- *RightHandCursor / LeftHandCursor* – continuous gesture, used to move the hand cursor with the right or left hand.
- *ZoomOut* – continuous gesture, left and right hands are to the front and put together at the beginning, then the hands move apart in different directions.

- *ZoomIn* - continuous gesture, left and right hands are to the front and at least 0.7 meter apart at the beginning, then the hands move closer to each other.
- *Wheel* - continuous gesture, left and right hands are to the front and shoulder size apart at the beginning, then the hands keep the distance and move as if they turn an imaginary wheel counter clockwise (which returns positive angle) or clockwise (which returns negative angle).
- *Jump* – the hip center gets at least 15 cm above its last position within 1.5 seconds.
- *Squat* - the hip center gets at least 15 cm below its last position within 1.5 seconds
- *Push* – push forward with the left or right hand within 1.5 seconds
- *Pull* - pull backward with the 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, basic understanding on how the sensor works and what joint positions mean.

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 example on how to do that, look at the processing of some simple poses and 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 tracked 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, for instance 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 simple gestures, like hand raises or swipes, work. Then find a gesture similar to the one you need. Copy and modify its code, as to your needs.

Support, Examples and Feedback

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