Getting started with the "Be the Controller tool kit" with Kinect for Windows SDK Beta

About Be the Controller Project:

Be the controller is a developer's tool kit made by Nicolas Hadjimina as part of his MSc in Computer Games Technology dissertation at the City University London. This tool aims to help programmers easily recognize user's pose and gestures, using the Microsoft kinect controller and perform the appropriate actions. The project is divided into two parts:

A) A windows application (PoseAndGestureGenerator.exe), which allows a developer to create a list of poses and gestures that he want to recognize in his application, using an easy to use graphic interface.

B) A library (Bethecontroller.dll) which provides the infrastructure to the developer of an application to recognize the set of poses and gestures he created with the PoseAndGestureGenerator project.

The tool calculates all the relations relative to the user's body so the position of the user in the environment is not important for the recognition of a pose as far as the kinect controller can track the user's skeleton.

System Requirements

Supported Operating Systems and Architectures

- Windows 7 (x86 or x64)

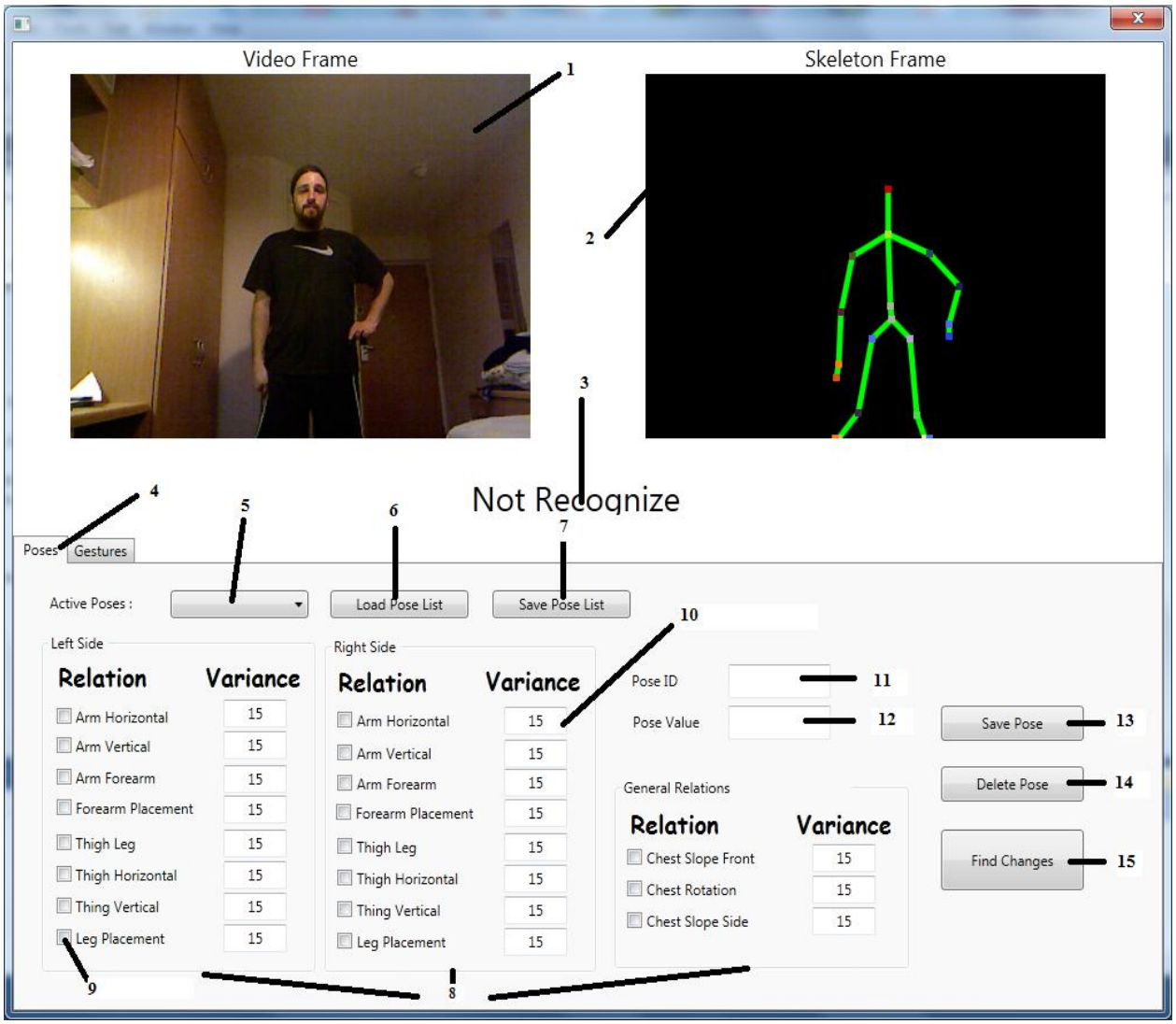
Hardware Requirements

- Computer with a dual-core, 2.66-GHz or faster processor
- Windows 7–compatible graphics card that supports Microsoft® DirectX® 9.0c capabilities
- 2 GB of RAM
- Kinect for Xbox 360® sensor—retail edition, which includes special USB/power cabling

Software Requirements

- Microsoft .NET Framework 4.0
- Microsoft XNA Game Studio 4.0
- Microsoft Speech Platform - Server Runtime, version 10.2 (x86 edition)
- Microsoft Speech Platform - Software Development Kit, version 10.2 (x86 edition)
- Kinect for Windows Runtime Language Pack, version 0.9 (acoustic model from Microsoft Speech Platform for the Kinect for Windows SDK Beta)

Pose and Gesture Generator Quick Guide:



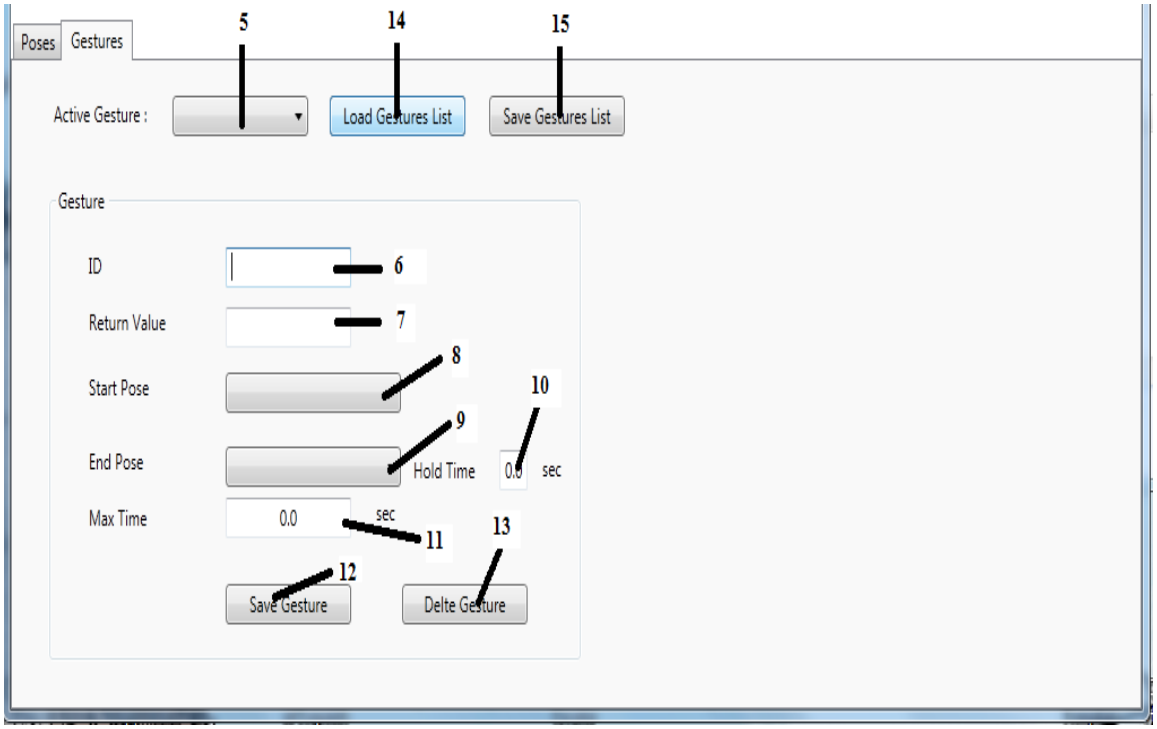
1. Video Frame : Displays the stream of images captured by kinect device
2. Skeleton Frame: Outline the user's skeleton which is draw using the body points received from kinect device
3. Status Area : Shows the current reorganisation status, if the current user's pose is recognised or not
4. Pose and Gesture Tap control: allows the selection of pose/gesture processing.

POSE TAB

5. Active Pose Combo box: Shows the current selected pose.
6. Load Pose List Button: Activates the procedure to load a pose list from a file.
7. Save Pose List Button: Activates the procedure to save the current pose list to a file.
8. Relations Sets: Sets of available relations the tool can track.
9. Relation Check Box: Enables/Disables the tracking of a relation.

10. Variance Value: Shows how much we can vary from this relation. It is given in present (0-100)
11. ID: The unique identification of the pose
12. Return Value : The value which will be return when the pose is recognized
13. Save Pose Button: Saves the current pose to the poses list.
14. Delete Pose Button: Deletes the selected pose form the pose list.
15. Find Changes Button: Activates the Find Changes mode.

GESTURE TAB



5. Active Gesture Combo box: Shows the current selected gesture.
6. ID: The unique identification of the gesture
7. Return Value : The value which will be return when the gesture is recognized
8. Start Pose Combo box: Allows the selection, from the available poses, the start pose of this gesture.
9. End Pose Combo Box: Allows the selection, from the available poses, the end pose of this gesture.
10. Hold Time value: Defines how much time (in seconds) the user should hold this pose in order the gesture to be recognize
11. Max Time value: Defines the maximum time the user has to take the end pose from the moment he loses the start pose.
12. Save Gesture Button: Saves the current gesture in the gestures list.
13. Delete Gesture Button: Deletes the selected gesture form the gestures list.
14. Load Gesture List Button: Activates the procedure to load a gesture list from a file.
15. Save Gesture List Button: Activates the procedure to save the current gesture list to a file.

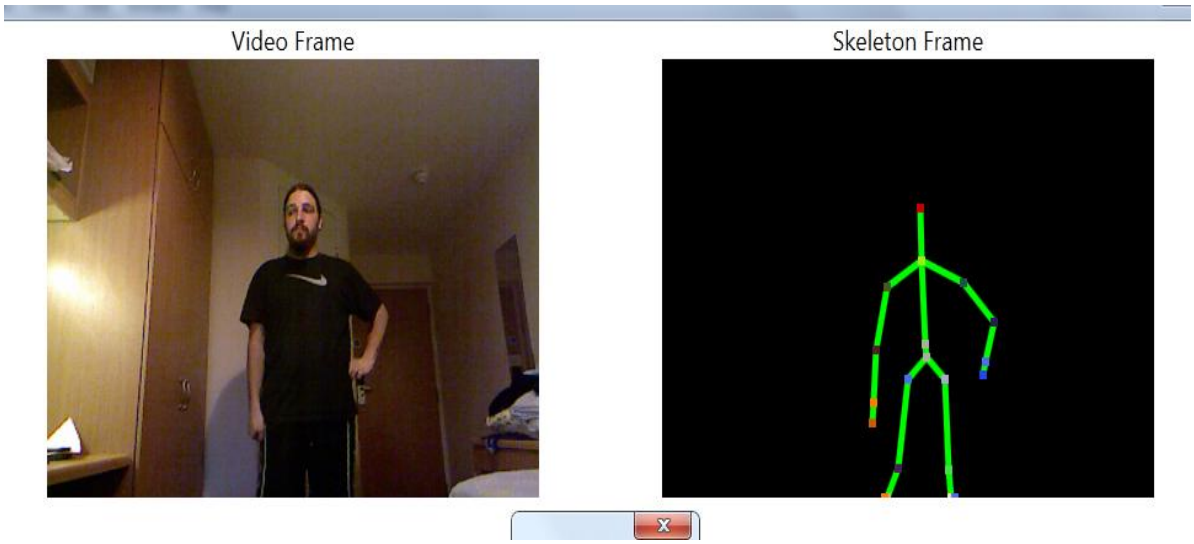
Pose and Gesture Generator Step By Step

START THE POSE AND GESTURE GENERATOR:

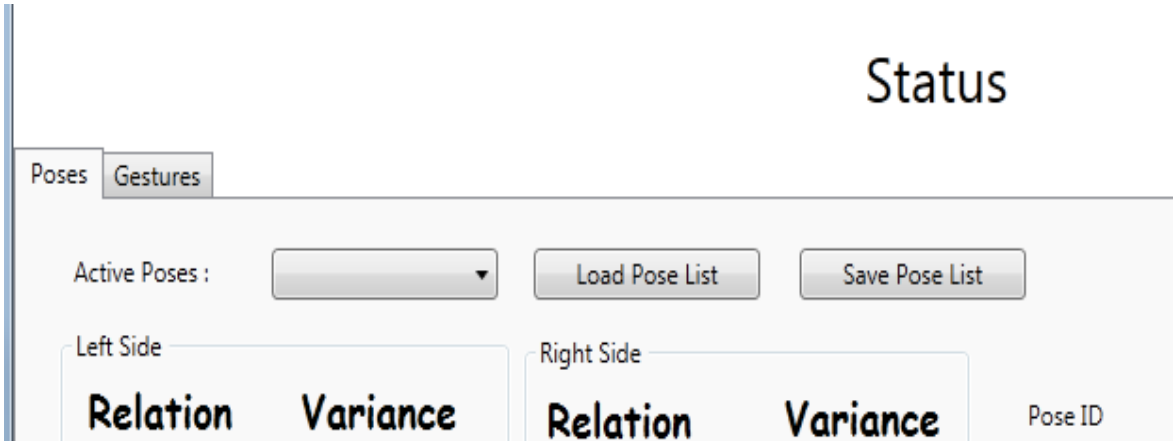
1. Connect Kinect to your Pc
2. Run the PoseAndGestureGenerator.exe

CREATE A POSE

1. Make sure that the kinect device is connected correctly: verify that Video frame area is displaying video stream and a skeleton outline is shown in the Skeleton frame area.

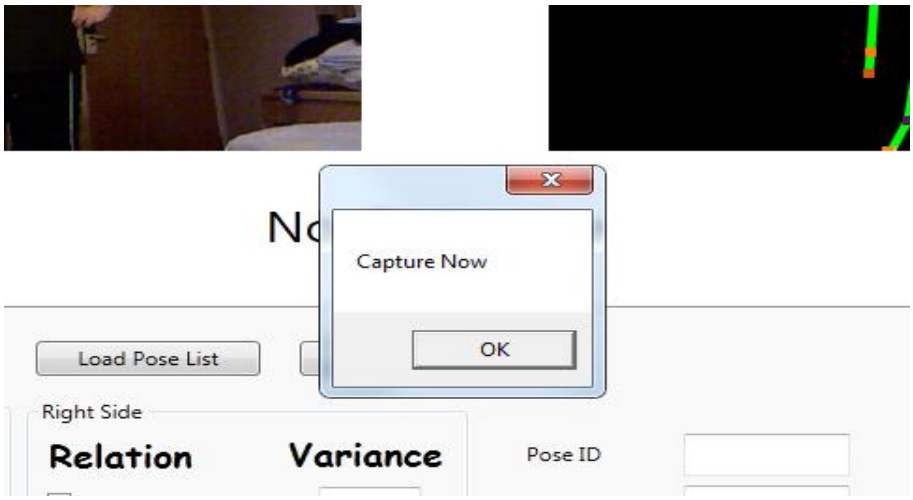


2. Select Poses Tab :

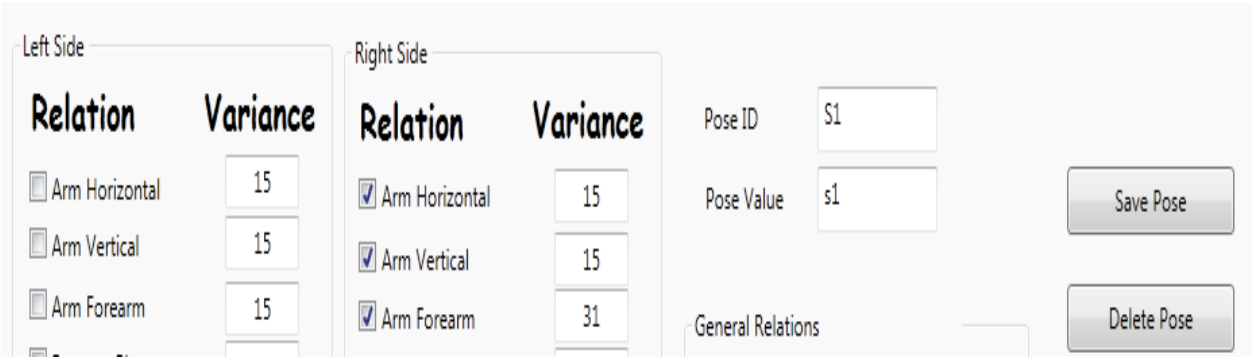


3. Stand in an appropriate distance (1.2–3.5 m) in front of the kinect controller and take the pose that you want to capture.

4. Say the Voice command "Capture Now" and wait for a verification message, which indicates that the pose was capture. Note: if the verification message does not appear, repeat again "Capture Now".



5. Select the relations you want to track and set the Variance for each selected relation. Note: The Find Changes mode can be used to help find the appropriate relations to track (see next section)
6. Give an ID (unique) for the pose
7. Set the return value (not unique) for the pose (his value will be returned by the reorganization procedure when the pose is recognized)

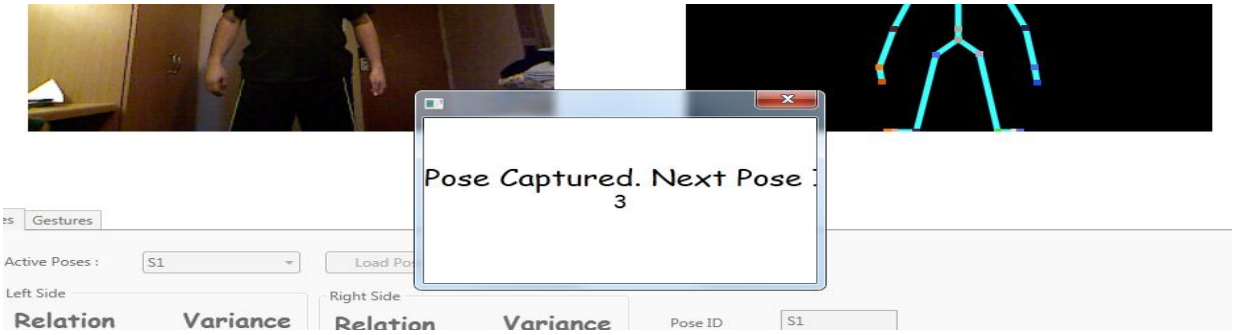


8. Press the Save Pose button. Note: If a pose with the same ID exists, a dialog box will ask you to verify that you want to replace the existing pose with the new one.
9. Say the Voice command, "Play" to reactivate the kinect.
10. Stand in an appropriate distance (1.2–3.5 m) in front of the kinect controller and re-take the pose in order to verify that the pose is captured correctly

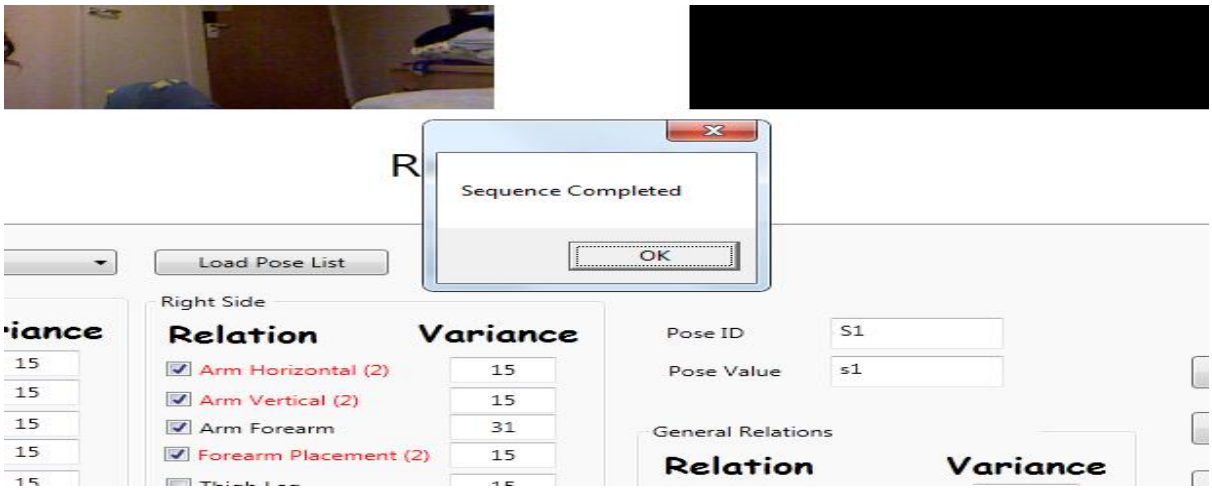
FIND CHANGES MODE

1. Select from the Active Pose Compo Box the pose that you want to use as the base for the comparison.

2. Press the button Find Changes.
3. Stand in an appropriate distance (1.2–3.5 m) in front of the kinect controller and take the pose you want to compare with the base pose.
4. Say the Voice command "Capture Now" and wait for a verification message, which indicates that the pose was captured (the message window will close automatically after 5 seconds). Note: if the verification message does not appear, repeat again "Capture Now"..

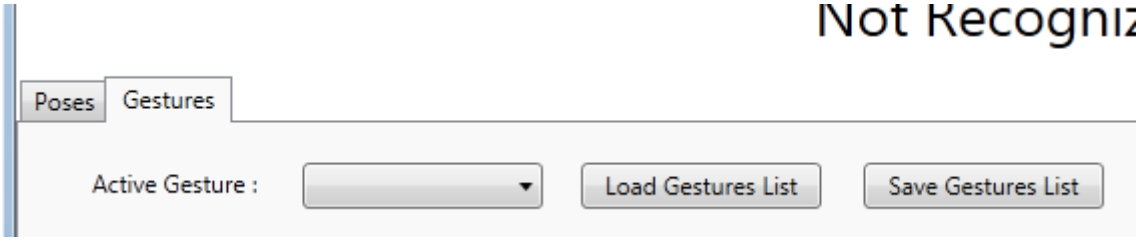


5. Repeat step 3 and 4 to add more poses to the comparison.
6. Say the Voice command "Stop" to stop the comparison procedure and display the result. The relations indicated in red are those that did not match in all the poses and the number in the brackets shows how many poses the relation did not match.

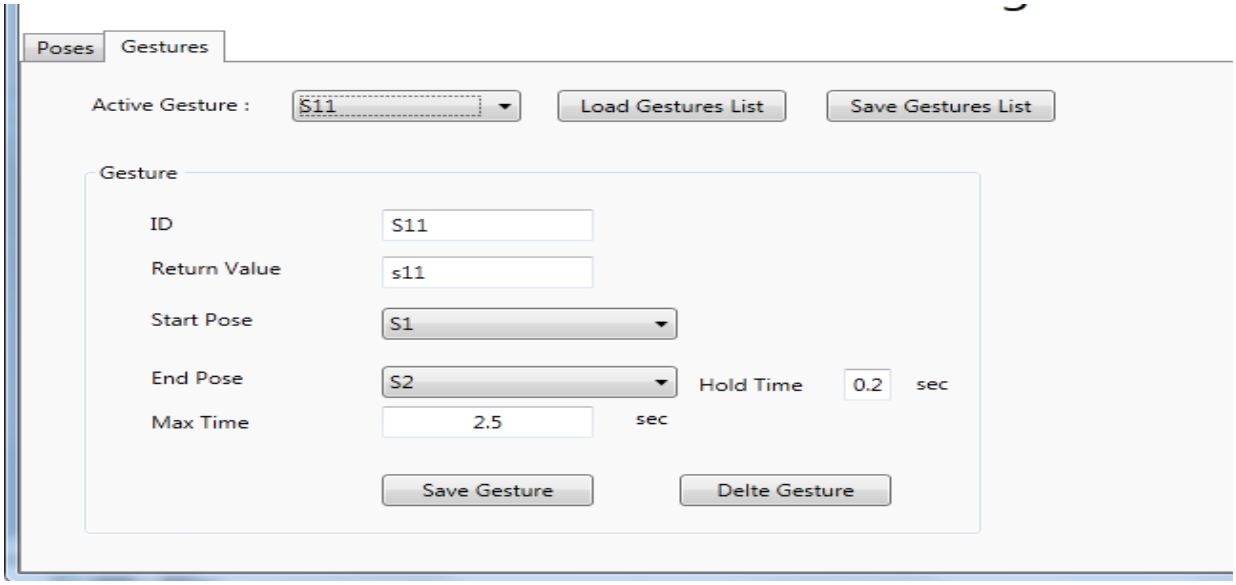


CREATE A GESTURE

1. Select Gesture Tab :



2. Give an ID(unique) for this gesture
11. Set the return value (not unique) for the gesture (this value will be returned by the reorganization procedure when the gesture is recognized)
3. Select the pose that will start the gesture
4. Select the pose that will be the end of this gesture and define how much time (seconds) the user should hold this pose in order for the gesture to be recognize.
5. Set the life of the gesture. The maximum time the user has to take the end pose from the moment he loses the start pose. If the user fails to take the end pose in this period the gesture will expire.
6. Press the Save Gesture button. Note: If a gesture with the same ID exists, a dialog box will ask you to verify that you want to replace the existing gesture with the new one.



Be The Controller library in C# - Step By Step

1. Add the bethecontroller.dll file to your project's references.
2. Create a new instance of the library :

```
BTCRunTime BTC = new BTCRunTime();
```

3. Load Pose an Gesture Files

```
BTC.LoadPoses("c:\\poses.pbtc");
BTC.LoadGestures("c:\\gestures.gbtc");
```

4. Initialize kinect device

```
try
   {
    BTC.Start();
   }
catch (InvalidOperationException)
   {
   System.Console.Write("Runtime initialization failed. Please
make sure Kinect device is plugged in.");
   }
```

5. Get the kinect run times instance to make sure that the kinect is initialized (if returned null then the kinect wasn't initialise correctly) :

```
KinectNui = BTC.GetKinectNui();
```

6. Set the function which will be called when an event (Pose,Gesture) will be recognized.

```
BTC.OnEventRecognized += new
BTCRunTime.EventRecognizedEventHandler(Game_OnEventRecognized);
```

7. Set the function, which will be called, when the user stops holding a pose or an active gesture expires (fails to be performed in the max gesture's lifetime).

```
BTC.OnEventStopRecognized += new
BTCRunTime.EventStopRecognizedEventHandler(Game_OnEventStopRecognized);
```

Examples of the two types of call functions:

```
void Game_OnEventRecognized(EventRecognizedEventArgs gca)
        {
            foreach (KinectEvent Event in gca.Events) {
                if (Event.Type == KEvents.Gesture) {
                    if (Event.Value == "s11") swort = true;

                }
                if (Event.Type == KEvents.Pose)
                {
                    if (Event.Value == "s1")
                    {
                        startcasting = true;
```

```
                        swort = false;
                    }
                }
            }
        }
```

```
void Game_OnEventStopRecognized(EventStopRecognizedEventArgs gca)
        {
            foreach (KinectEvent Event in gca.Events)
            {
                if (Event.Type == KEvents.Pose)
                {
                    if ((Event.Value == "s1"))
                    {
                        startcasting = false;
                    }
                }

            }
        }
```

Note: We use the return values to choose which actions to perform because the values are not unique compared to the ID and we may need to perform the same actions for two different events.

BE THE CONTROLLER HELPER

The library also provides some helper functions under the BTCHelper class. Currently it only provides a function that takes an input: the hand (Left/Right), a vector that represents any transformation performed to the skeleton points (eg. reverse the X coordinate), the active skeleton instance and returns a normalized vector that represents the direction of the hand.