Fullbody Interaction

Software environment: Unity 5

Hardware environment: Kinect 2

Practice objectives:

1. Go through the relevant source code and test the game
2. Improve skeleton animation by smoothing the rotation of the joints
3. Implement the translation of the character
4. Play the multiplayer game!

Working directory:

D:\TP\group_1xh\KinectTP (you may find a final version of the project at *D:\TP\complete*)

Step 1: Run Unity, the project you will work on should either open automatically or be the first on the Unity Wizard list.

Goal: Real time character animation is not a trivial subject. It contains several steps (modeling, rigging and skinning) and may follow different animation approaches (forward/inverse kinematics). It also involves a tradeoff between latency and quality. In this TP we are going to use a low cost approach, with a Kinect 2 and forward kinematics. We will improve the quality of the animation with smoothening and implement the character position control.

Objective 1: Go through the relevant source code and test the game.

KinectModelControllerV2.cs

Avatar animation: It uses the joint position information provided by Kinect to set the avatar pose (joints orientation). Have a look on the RotateJoint function, called for each joint (used in Objective 2).

KinectModelPosition.cs

Set the avatar position. It will be written by the students in Objective 3, currently it only contains the variables the students might need.

Tasks:

a. Run the program, and observe the degree of control you have over the avatar (i.e. only the noisy control of the avatar joints)
b. Observe the debugging information drawn in the scene, they inform the RGB image obtained by Kinect, the depth + RGB drawn as a mesh, and the position of the joints of the tracked player. (disable the KinectDebug game object afterwards as it consumes resourses)

Objective 2: Improve skeleton animation by smoothing the rotation of the joints

As you might have noticed, the avatar pose is very unstable. This is more evident when arms are in front of the performer body, or when the performer raises his legs or try to kick the air.

This happens because the tracking data provided by the Kinect SDK is raw and unreliable. What we can do to address this issue is to implement a **smoothing algorithm** (i.e. low pass / filter out the high frequencies). However, this brings up a **tradeoff** between **tracking data quality** and **latency** (we will increase the time it takes for the input data to be presented in the screen).

a. Using the Quaternion.Lerp function, you should interpolate the tracking data with an exponential smoothing: $s_t = \alpha \cdot x_t + (1-\alpha) \cdot s_{t-1}$.
b. As the frame rate may fluctuate, the α parameter should be scaled in order to keep the smoothing constant in time. How can we do that? You can use the value in **Time.deltaTime** to know how much time has elapsed between two frames.
c. Optimize the **α** parameter in order to balance the **latency/pose stability** tradeoff for the ball catching game. You can do that by trial and error, what was the optimal parameter for you?

Objective 3: Implement the translation of the character

Now that you have stable control of the avatar pose, wouldn’t it be much easier to grab those balls if you could get closer to the goal and move around?

To do so, fill in the **Update** function of the script **KinectModelPosition.cs**.

a. Implement the position smoothing with Vector3.Lerp (similar to Objective 2)
b. Add a **position offset** parameter so that the avatar feet don`t go through the ground if the player is too short, and don`t float in the air if the player is too tall.

Objective 4: Play the multiplayer game!

Tired of playing alone? Deactivate the GameObject “SinglePlayer” and activate the GameObject “Multiplayer”

a. You will have to add an **offset** along the X axis so that the players don`t hit each other while sharing the Kinect tracking space
b. Make it a fair game by repositioning the canon so that the balls are thrown from above

Known issue: sometimes the same user gets 2 player IDs and end up controlling both avatars. It this happens, restart the game.