**Kinect for Windows: Find user height accurately**

[VangosVangos](http://studentguru.gr/members/Vangos/default.aspx)

Mon, May 7 2012 1:20

* [5](http://studentguru.gr/b/vangos/archive/2012/05/07/kinect-for-windows-find-user-height-accurately.aspx#comments)

Some people ask me if there is a way to determine the height of a user standing in front of a Kinect device. The answer is defintely positive: We can detect the user's height regardless of her distance from the sensor! We only need some very basic Maths knowledge. Let's find out more...

[Download the sample demo application](http://studentguru.gr/cfs-file.ashx/__key/communityserver-components-postattachments/00-00-13-54-17/KinectUserHeight.zip).

**Prerequisites**

* [Kinect for Windows sensor](http://www.amazon.com/Microsoft-L6M-00001-Kinect-Sensor-Windows/dp/B006UIS53K) ([Kinect for XBOX](http://www.amazon.com/Microsoft-XBOX-360-Kinect-Sensor/dp/B005UFNNWC/) will do the job as well)
* [Microsoft Kinect SDK](http://www.microsoft.com/en-us/kinectforwindows/develop/)
* Windows 7
* Basic knowledge of C#, WPF and Visual Studio

**The algorithm of finding the user's height**

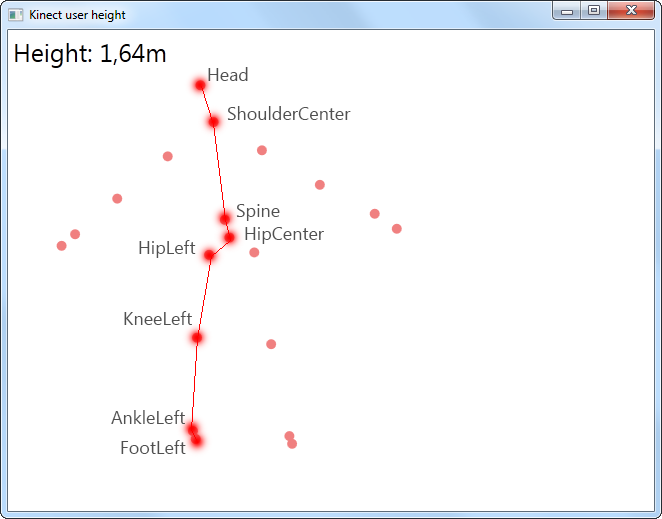
Kinect provides you with the coordinates (X, Y and Z) of 20 skeleton joints. Someone might think that a person's height is the distance from the head joint to a foot joint, right? Wrong! Your users might stand in bended knees, they might lean a bit to the left or to the right. If you try to calculate the distance defined from the head joint to one of the foot joints, you'll get a far from accurate result.

Another point of interest is that Kinect gives you the **center** of the head joint. This means that you'll need to add 9 - 12 centimetres to the calculated height. You won't be able to calculate the total height with 100% accuracy. If you need more accuracy, you'll need to detect the end of the head using the depth stream. Not that difficult, but let's focus on the skeleton stream right now.

If you examine the human skeleton joints a little more carefully, you'll notice that the height is the sum of the lengths of the following line segments:

* Head - ShoulderCenter
* ShoulderCenter - Spine
* Spine - HipCenter
* HipCenter - KneeLeft or KneeRight
* KneeLeft / KneeRight - AnkleLeft / AnkleRight
* AnkleLeft / AnkleRight - FootLeft / FootRight

Here is a comprehensive picture illustrating the above segments:



First of all, we need to calculate the length of the line defined by two joints in the 3D space. This is quite easy Mathematics. Simply find the square root of the sum of squared differences of the coordinates:

public static double Length(Joint p1, Joint p2)  
{  
    return Math.Sqrt(  
        Math.Pow(p1.Position.X - p2.Position.X, 2) +  
        Math.Pow(p1.Position.Y - p2.Position.Y, 2) +  
        Math.Pow(p1.Position.Z - p2.Position.Z, 2));  
}

And here is a way to find the length of more than two joints:

public static double Length(params Joint[] joints)  
{  
    double length = 0;

    for (int index = 0; index < joints.Length - 1; index++)  
    {  
        length += Length(joints[index], joints[index + 1]);  
    }

    return length;  
}

Pretty straightforward till here. But, wait a second! How will you determine which leg corresponds to the most accurate user height? Since we suppose that both of the user's legs have the same length, we need to choose the one that is tracked better! This means that no joint position is hypothesized. Here is how to find out the number of tracked joints within a joint collection:

public static int NumberOfTrackedJoints(params Joint[] joints)  
{  
    int trackedJoints = 0;

    foreach (var joint in joints)  
    {  
        if (joint.TrackingState == JointTrackingState.Tracked)  
        {  
            trackedJoints++;  
        }  
    }

    return trackedJoints;  
}

We can now safely decide on which leg to use:

int legLeftTrackedJoints = NumberOfTrackedJoints(hipLeft, kneeLeft, ankleLeft, footLeft);  
int legRightTrackedJoints = NumberOfTrackedJoints(hipRight, kneeRight, ankleRight, footRight);

double legLength = legLeftTrackedJoints > legRightTrackedJoints ? Length(hipLeft, kneeLeft, ankleLeft, footLeft) : Length(hipRight, kneeRight, ankleRight, footRight);

And here is the final method:

public static double Height(this Skeleton skeleton)  
{  
    const double HEAD\_DIVERGENCE = 0.1;

    var head = skeleton.Joints[JointType.Head];  
    var neck = skeleton.Joints[JointType.ShoulderCenter];  
    var spine = skeleton.Joints[JointType.Spine];  
    var waist = skeleton.Joints[JointType.HipCenter];  
    var hipLeft = skeleton.Joints[JointType.HipLeft];  
    var hipRight = skeleton.Joints[JointType.HipRight];  
    var kneeLeft = skeleton.Joints[JointType.KneeLeft];  
    var kneeRight = skeleton.Joints[JointType.KneeRight];  
    var ankleLeft = skeleton.Joints[JointType.AnkleLeft];  
    var ankleRight = skeleton.Joints[JointType.AnkleRight];  
    var footLeft = skeleton.Joints[JointType.FootLeft];  
    var footRight = skeleton.Joints[JointType.FootRight];

    // Find which leg is tracked more accurately.  
    int legLeftTrackedJoints = NumberOfTrackedJoints(hipLeft, kneeLeft, ankleLeft, footLeft);  
    int legRightTrackedJoints = NumberOfTrackedJoints(hipRight, kneeRight, ankleRight, footRight);

    double legLength = legLeftTrackedJoints > legRightTrackedJoints ? Length(hipLeft, kneeLeft, ankleLeft, footLeft) : Length(hipRight, kneeRight, ankleRight, footRight);

    return Length(head, neck, spine, waist) + legLength + HEAD\_DIVERGENCE;  
}

Finally, you can use this extension method within a custom application easily:

void Sensor\_SkeletonFrameReady(object sender, SkeletonFrameReadyEventArgs e)  
{  
    using (var frame = e.OpenSkeletonFrame())  
    {  
        if (frame != null)  
        {  
            Skeleton[] skeletons = new Skeleton[frame.SkeletonArrayLength];  
            frame.CopySkeletonDataTo(skeletons);

            var skeleton = skeletons.Where(s => s.TrackingState == SkeletonTrackingState.Tracked).FirstOrDefault();

            if (skeleton != null)  
            {  
                double height = skeleton.Height();  
            }  
        }  
    }  
}

You can now integrate this code to your own applications and find out the user's height! [Download the complete WPF application](http://studentguru.gr/cfs-file.ashx/__key/communityserver-components-postattachments/00-00-13-54-17/KinectUserHeight.zip) in order to see it in action right now!

Enjoy kinecting!