Interactivity with 3D Models through Body Gestures

September 1st, 2011

Matthew Religioso

Introduction

For my PRIME Project I was tasked in a collaborative effort between Osaka University and the National institute of Information and Communications (NICT) to develop a Kinect interface to interact with a 3D model of Osaka's Dontonburi rendered in NICT's NexCAVE. It was required to be easy to use with effortless gestures, and perform well from a distance of about 2.5 meters with the Kinect having an un-tilted sight of the user. While working on the Kinect interface I collaborated with Sarah Larson on developing the Opencover plugin called Navi running on the NexCAVE to display the model.

Design

Since the user was required to be distant from the Kinect, I opted for body gestures over hand gestures. Hands have many minute details that at a distance are lost in the Kinect's depth sensor, making them difficult to track. The body however is very large and will still have a sizable surface area of data for the Kinect to track.

For traversing a 3D model with minimal effort, I split the tasks of rotation and translation to the right hand and body positions respectively. For these two I used reference points to determine the user's intended direction to translate and rotate the world. To start tracking the user, they must perform a specific body pose to start, and another to end. To begin tracking, we used the

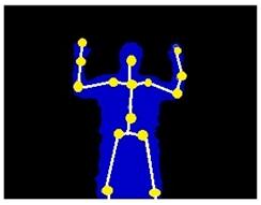

Figure 1: Start Pose

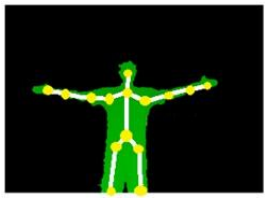

Figure 2: Stop Pose

Libfreenect's starting pose, as it is a very unique and unlikely body pose for the user to be in unless intentional. To stop tracking, we used an airplane arm position with the arms at their sides.

To translate the world, users need to set their body's reference point first via the starting pose. This point is used to determine the user's desired translation movement through the world with respect to the current view/rotation. Every frame we calculate the difference in the XY of the body's center and its reference point. If it exceeds a threshold we then apply translations using the XY difference as the speed. If the user moves left beyond the threshold for example, the world will be translated right with respect to their current view to simulate walking left. To stop, they simply return back to their reference point. This method of translation makes it easy and intuitive to move. To go forward, step forward, etc.

Figure 3: Users move their body to translate within 3D space

Implementation of rotations is very similar to the translation method. First the user must extend their right hand forward towards the Kinect, passing a depth threshold for 30 frames. After the Kinect will set the hand's center reference point to be at around their shoulder, and begin tracking their hand with along an XY coordinate system. At every frame the gesture is tracked it calculates the XY difference between the reference point and the hand, and these values

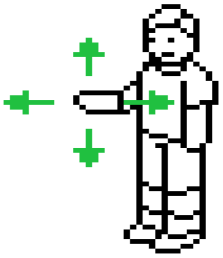

Figure 4:Users move their right hand to rotate the world

become the angle of the yaw and pitch world rotations if the distance passes a certain threshold. Should the user's hand depth leave an allowed range of tracking activity, or they touch their right shoulder it will end the tracking of the hand.

Like the body tracking, the reference point is used interpret the user's desired rotation. I delayed the initiation of the hand tracking to prevent false-positives, such as when the user places his in front of him for any reason other than to rotate the world. Ending the tracking is a little tricky. If they simply drop their hands at their side it will rotate down. To avoid this, we end tracking when the user places their hand near their shoulder away from the Kinect to prevent any unwanted rotations, at the cost of being slightly unintuitive. An issue with concatenating pitch and yaw rotations upon the world matrix will indirectly modify the roll angle.

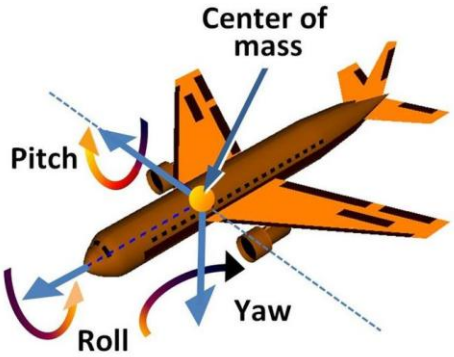

Figure 5: Roll, Pitch, and Yaw angles.

NASA. (2010). Roll Pitch Yaw. *Wikpedia.* Retrieved August 29th 2011, http://en.wikipedia.org/wiki/File:Roll_Pitch_Yaw.JPG

Humans typically do not experience this under normal conditions, so I compensated this angle by extracting the roll from the world matrix and rotating it in the opposite direction, effectively canceling it out.

Intentionally I did not allow the user to translate directly up and down in the world. Originally I wanted to use a superman-esque pose with their arms held straight up to go up and ducking to go down but this goes against our intent to be an effortless gesture. Ducking/squatting require much energy, and the up pose would be difficult to parameterize their speed for it may be confused with the start pose.

With the body and right hand handling user movement through the world, we free the left hand to handle potential menu items or manipulations of objects. Later during the project, NICT wanted to include pre-defined viewpoints in the world that the user can cycle through. By pushing forward passed a depth threshold, Navi will cycle through the viewpoints defined in a separate file specifically for that model.

In case multiple people appear in the Kinect's sight range, I implemented a skeletal tracking lock on the current skeleton. Any additional skeletons appearing after tracking has been started will not interfere with gestures . This will prove especially useful at the exhibition site where space will be limited, and people will inevitably appear somewhere behind the user.

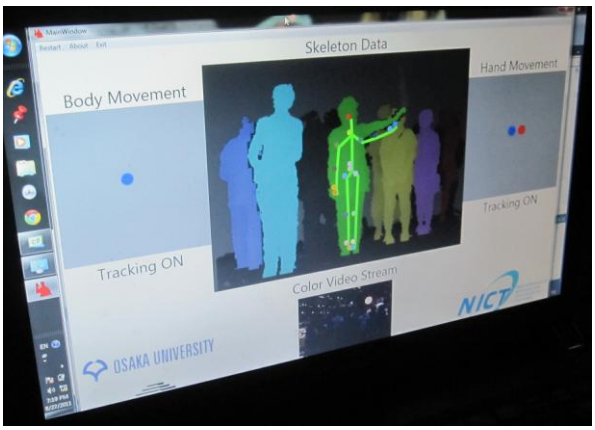

Figure 6: Interface of my Kinect Application. Note the person tracked is the user with the skeleton displayed, ignoring all others tracked.

To add additional user feedback to keep them informed of their gestures aside from visually seeing the transformations take effect world, I added audio and visual cues to each gesture. Each gesture a unique sound to signal and confirm the gesture has been received and processed. For visual feedback I included two boxes, each displaying the direction and magnitude of the body and hand offsets for their respective reference points.

In the event the Kinect is placed in an awkward angle, I added the ability to alter its tilt with a button. Though this is not recommended, as the Kinect will skew the data should it's angle be too obtuse with respect to the ground.

Implementation

I used the Microsoft Kinect SDK Beta to recognize gestures via the included skeletal tracking feature. The skeleton tracking recognizes the person and returns a set of 3D points of the user‘s joints which are utilized for gestures. C#/.NET framework was selected to write the application due to the SDK being easily integrated with it and its ease of use when writing GUI interfaces for applications. I based off the GUI implementation for the Kinect from a Skeletal Tracking tutorial from Microsoft, as it showed how to utilize and display the depth and video streams to the user. Navi was written as an Opencover plugin on Covise with c++ and OSG to utilize NexCAVE’s 3D environment.

Since the applications were running on different systems, I created a UDP connection to quickly send data in this real-time application. Later during development, I realized the packets were being heavily queued and delayed due to Navi running at a significantly lower FPS than the Kinect application. So I implemented a separate thread on Navi to constantly read in packets and update a shared buffer that Navi’s rendering pipeline will use when applying world transformations at its own pace.

Figure 7: Small children were difficult to track with their smaller stature and shorter arms.

Results and Analysis

My project was tested from August 26th - 28th at the Knowledge Capital Trial 2011. It was able to track adults with relative ease, however it had great difficulty with children. Children’s smaller size yielded sparse data to the Kinect, making

the recognition very sporadic and unstable. Also my program did not take a body scale into consideration, thus sometimes their arms were too short to trigger the minimum threshold for the hand gestures. The arm threshold issue could easily be fixed by implementing some sort of scale calculation to determine the depth threshold dynamically for each skeleton. As for the small skeleton size of kids, moving them closer to the Kinect partially solved this problem with the caveat they may become untrackable should they got beyond the minimum depth range.

A few times people were confused with the idea of a reference point, and continuously turned around in circles. Those that understood the reference point concept well quickly navigated with ease. Another issue was that my system gave the user too much control over their movement that they did not need for the short demo. People did not need to go sideways very often, and this could have been removed as it caused more headaches and confusion than benefits. The "Flying" aspect through the world made it difficult for some to cope with, as they constantly fought the pitch angle going up and down, sometimes getting lost beneath the model. A simpler beginner's control scheme that mapped the movement all to one hand would have been preferable in these cases, before easing them into the more advanced scheme. This could have been implemented by using the right hand control scheme for the viewing direction, and using its depth towards the Kinect as the forward movement speed. In addition, possibly locking the Z position so that they will appear walking instead of flying through the scene could also help.

Figure 8: Pitch control disoriented a few users

Some people had a tendency to drop their right hand when they want to stop its tracking. This seems to be the most intuitive and natural way to stop tracking, but this commands the Kinect to tilt the world down. My tap the shoulder approach to ending hand tracking prevents this downward tilt. Due to the nature of the right hand's reference point, this is an innate setback to its design and needs an alternative control scheme to circumvent this issue. Proper instruction to the user makes this a small concern, though it is not very intuitive and unwieldy to a new user.

Next Steps

During the 8th week of my project I was informed of the original intent to have the user be in close proximity to the Kinect for them to immerse themselves inside the NexCAVE. Due to some miscommunication between NICT and Osaka University I unknowingly built the program to be optimized at a distance from the Kinect in mind with body gestures. The next logical step is to utilize hand gestures instead, tracking only the upper body and hands which the Microsoft Kinect SDK cannot do. This allows the user to be closer to the Kinect without any problems of tracking a full body from afar. Thankfully Libfreenect provides the basic implementation for hand tracking. We can track the hand's 3D location and shape to determine different gestures, such as a drag, rotate, translate, holding or menu selection. Using different hand signals we can distinguish between gestures and their beginning and end, making the overall experience easier, more precise and intuitive to a new user.

Navi must be updated to handle these new manipulations. My c# application would no longer be needed as Libfreenect runs on c++ and is crossplatform, but we can integrate a new

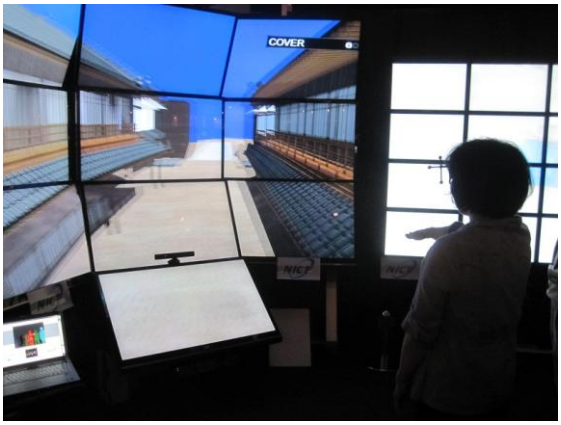

Figure 9: With Libfreenect, separate programs and a UDP connection will no longer be needed.

Kinect interface directly into Navi to handle gesture recognition. With both combined we would no longer need a UDP connection as it will all be contained in a single Opencover plugin.

Bibliography

Libfreenect (2011). OpenKinect. *OpenKinect Wiki*.Retrieved August 29th, 2011, from openkinect.org/wiki/Main_Page

Microsoft. (2011). Kinect for Windows SDK Beta, *Microsoft* Research, Retrieved August 29th, 2011, Microsoft Kinect SDK Beta, from research.microsoft.com/en-us/um/Redmond/projects/kinectsdk

HLRS. COVISE. *HLRS*. Retrieved August 30th, 2011, from www.hlrs.de/organization/av/vis/covises

Fox, T. (Aug. 14, 2009) Calit2 Visualization Team Develops 3-D Technology from Modified HDTV LCD Screens. *Calit2*. Retrieved August 30th, 2011, from www.calit2.net/newsroom/article.php?id=1584