PRIME 2011

Navi: Covise-Kinect Navigation Interface

Sarah Larsen NiCT Keihanna 2011

Abstract:

In this project, a Navigation plugin was created to interface a Kinect with COVISE to allow users to explore a VRML model through movements and gestures. A flickering and texture exportation problems on the model of old Osaka Doutonbori (道頓堀) also had to be resolved. This plugin was combined with a viewpoint plugin to make it easier to reset and change views, and a texture optimization plugin. This project was done in collaboration with Matthew Religioso, at Osaka University.

Implementation and Description:

Before implementing the Navigation plugin, two problems with the old Osaka Doutonbori model had to be resolved: not all the textures were loading correctly, and the screen flickered between two images, even when the image was static. The OpenSceneGraph (OSG) version of the same model didn't flicker, but also didn't show any of the textures. Professor Schulze suggested we change the NEAR and FAR values in the configuration file, and while that improved the flickering, it didn't completely go away, especially with bigger models, like old Doutonbori. As for the textures, most of them loaded correctly, but the last few could not be fixed without going through each texture by hand. The problem was due to optimizing by reusing textures; some of the textures were mislabeled or optimized incorrectly and didn't display. Manually checking each one would have been difficult due to the size of the model and the number of textures.

At first, Navi used only OSG classes, an input manipulator, to control the viewing angle instead of manipulating OpenCOVER directly, and was controlled through the mouse instead of the keyboard. Because the Viewer might not be compatible with the master-slave screens, it was changed to use OpenCOVER commands only, and to accept keyboard instead of mouse commands along with Kinect instructions.

Navi uses UDP to communicate with the Kinect. At first, it only used a basic connection, checking for new data every 6 milliseconds, but since that dramatically lowered the frame rate with the larger models. Another class called MyThread was created by Matthew Religioso so that Navi could continually take in input from the Kinect, and only use what it needed. The command string it receives specifies the type of action (movement or viewpoint change) and the amount to be translated or rotated, calculated from displacement from the reference point. The more exaggerated the motion, the more it will translate or rotate in a given frame. Navigation with the keyboard uses all hard coded values, and is mainly used for testing, debugging, and selecting viewpoints, since it prints to the command line.

Navi was implemented using full-body tracking. Translation is controlled with body movement, calculated from the center of the body. Rotation is controlled by the right hand, while viewpoints are controlled by the left. The body reference point is set when the user raises both arms at a 90 degree angle, while the rotation reference point is sent whenever the user raises the right hand and holds it in front of them. To turn off all tracking, both arms must be raised straight out, aligned with the users' shoulders. To stop right-arm tracking only, the arm must either be placed on the shoulder or quickly put down on the side.

To move the viewing angle, Navi actually moves the model instead. It takes the Xform matrix of the model, its current position, and rotates and translates it according to the displacement values and the current viewing matrix. It then sets the Xform Matrix of the object as the correctly rotated and translated one, with some correction to the rolling so that it is with respect to the user and so that there is a clean rotation.

The master receives these values, and then sends them along with the scale to all of the slaves before every frame. On a left-hand command, the master accesses the next viewpoint on the list and sends the entire matrix to all of the slaves, along with the other values. This way no data is lost, even during transitions. Originally, the viewpoints were supposed to be taken from the WRL files directly, but there was no API to access the viewpoints easily from Vrml97. Afterwards, a similar Viewpoint class was created, similar to WRL format, but currently Navi uses matrices to store viewpoints. These are still easily convertible, as long as the model can be opened in COVISE. Navi responds to keyboard commands, when ‘e’ is pressed, it will print out the current object matrix onto the command line.

The viewpoint list is created on initialization. Navi reads a file specified in the environment variable FILE, and parses the values for each viewpoint into a matrix. The viewpoint is then stored on a list by the master. The current location of the list is stored as a global variable and is incremented whenever the left hand is pushed forward beyond a certain depth. The depth was small enough that even children could use this features, even if they weren’t able to use the rotation.

Navi can also be scaled according to the model size. The default is for the Doutonbori model, but SCALE can be specified in the command line to make the model more or less sensitive. Some of the OSG optimizing functions were used; however there wasn’t a substantial difference in frame rate.

Testing and Improvement:

At the exposition, most users seemed to have trouble with the flying aspect of Navi, which allows you to rotate in all directions, but only translate in two. An improvement to make a walking-only navigation system might have been easier for users to pick up in a short amount of time, as it's more forgiving. The fact that it uses a reference point was also difficult to convey, though some users picked up on it quickly. Matthew Religioso created an interface to try to show this to people, but it was difficult to express in Japanese, and most people didn't notice the laptop monitor. The right hand posed another difficulty, as some people would go too far down and lose control, while others put their right hand down to rest and accidentally went downwards. Also, children had trouble using the right arm to navigate since their arms were too short for the Kinect to track their joints properly. They were only able to translate using body movement. Distance and arm confusion, resulting in accidentally turning off tracking, were other common problems. The Kinect needs to be able to view the entire body in order to start tracking movement. This was difficult at the convention, where there were many people and projects and little space.

Bibliography:

OpenSceneGraph

http://www.openscenegraph.org/projects/osg

OpenCOVER/COVISE Programming Guide

vis.uni-koeln.de/covise/doc/pdf/programmingguide.pdf

OpenVRUI

http://vis.uni-koeln.de/covise/doc/doxygen/OpenCOVER/index.html

IVL

http://ivl.calit2.net/wiki/index.php/COVISE_and_OpenCOVER_support