TNM093 — Practical Data Visualization and Virtual Reality
Tri-modal Interaction With Scene Graphs

August 31, 2017

1 Introduction

This is the exercise on the topic of Virtual Reality for the course TNM093. This exercise is about tri-modal interaction and navigation with scene graphs.

1.1 Purpose

The purpose of this exercise is to let you familiarize yourself with how scene graphs can be used to compose a VR environment with graphical, haptic and sound feedback, also called tri-modal interaction and some basic scene navigation. You will also see the purpose of tracking and how important the viewpoint is for the overall experience.

This is also an exercise in documentation navigation, interpretation and understanding so the laboratory assistants will also be helpful in that regard.

1.2 Examination

After finishing all tasks in this exercise, show you results to a supervisor and explain what you have done and what conclusions you have come to.

For the successful examination please make sure that you fully understand what you have done. Also, during the tasks, save intermediate steps so that you can show and explain the result from one individual task.

1.3 Software and Equipment

This exercise can be partly done in the *Linux laboratory* where all the necessary software is installed, however all tasks must also be tested on a haptic workstation in the *VR laboratory* where also the examination will be done.

The VR laboratory is in Kopparhammaren 6, floor 5. To gain access you need to read and accept the conduct rules by signing a form. More information should be available on the course webpage.

The software, H3D API, is open source and can be downloaded from http://www.h3dapi.org. It is installed in both the Linux laboratory and the VR laboratory.

To set up the H3D environment under Windows, first open a command window and run the setup batch script located in the H3D program folder (`C:/VRLaboratory/H3D-2.2-trunk`). This can easily be done by drag-n-dropping the batch file into the command window. Under Linux you may open any command window, for example Konsole. After that your can open your scene graph by executing (for example)

```
H3DLoad scene.x3d
```

There is a stubb available for the exercise, that loads the required libraries and disables the standard X3D navigation.

Important: The haptic devices used in the lab are high-end laboratory equipment and thus both expensive and not very robust. This lab also contains experiments on the limit of what the equipment can handle. You should therefore be careful to follow the instructions and *not* try control parameters outside the prescribed range.

- Keep the pen within its workspace during experiments. If you are unsure of the workspace, try moving the pen around when no haptic program is running.
- *Always* hold the haptic pen when a program is running. Pick up the pen before starting the program and put it down when the program has finished completely.
- Keep an extra firm grip on the pen when experimenting with stability.

1.4 Documentation

This is not a manual so you are expected to search for information in documentation. Please check the documentation before the first session. Useful sources for this exercise are

- H3D API 2.3 Documentation — reference manual for H3D and X3D nodes (http://www.h3dapi.org/uploads/api/H3DAPI_2.3/doc/H3DAPI/html/)
- H3D Wiki — tutorials, Python documentation and other information about H3D (http://www.h3dapi.org/modules/mediawiki/index.php/Main_Page)
- Candy + HVR Toolkit 2.0 Documentation — reference manual over Candy and HVR nodes (http://www.itn.liu.se/∼karlu20/work/Candy)
- Candy Toolkit Wiki page — documentation over Python scripts in Candy (http://www.h3dapi.org/modules/mediawiki/index.php/Candy)
- HVR Toolkit Wiki page — documentation over Python scripts in HVR (http://www.h3dapi.org/modules/mediawiki/index.php/HVR)

2 The Scene as a Graph

Task 1:
Create your own (simple) chair design by composing several geometrical primitives. Use at least one `Sphere` in your design (necessary to see specular highlights). Create several copies at different positions using `DEF` and `USE`, and `Transform` (or `Group`). Check the H3D reference manual for more information about the nodes and their type abstraction. Do not leave the object colours to the default. To make the following tasks more interesting you should distribute the objects so some are in the plane of the screen ($z = 0$) and some are in front of and behind the screen.

Hint: The environment is calibrated for 1-to-1 metric units so you may use a ruler to determine appropriate size and translation before specifying them in your X3D file.

3 The Point of View

The viewpoint specified to the graphics library is having a much larger impact on the experience in VR than in other graphics application, since the perspective and object positions here should be correctly registered with your real world coordinates. This is often done wrong by people beginning to practice VR techniques, such as stereo rendering, however there is an increasing general understanding.

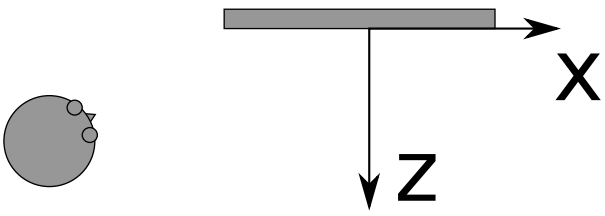

Figure 1: Looking at the screen with an acute angle. The eye's position can be measured with a ruler or estimated.

Perform the following two tasks on viewpoint manipulation in the Linux laboratory. Make sure to comment out the viewpoint definitions for the rest of the tasks, at least when working in the VR laboratory. With no viewpoint specified in the X3D file the loader will apply the calibrated system default instead.

In these tasks you manually set the position of the viewpoint, but in a real VR application this will be controlled with values acquired from head tracking equipment.

Task 2 — Standard Perspective Viewpoint:
First, test the standard perspective viewpoint (`Viewpoint` in H3DAPI). This will create a basic symmetric frustum. 1) Select a location where to position your eyes in the room, 2) measure or estimate to determine this position's coordinates in the 3D scene (origin is the centre of the monitor), and 3) enter the correct numbers into the scene's viewpoint. You may use a view from straight ahead, but you must also try at least one view with an acute angle. How should you orient your head to get the correct view?

Important: If you try this with a monographic monitor, such as in the Linux lab, make sure you close one eye looking at the screen.

Hint: If the 3D monitor is reflected in a mirror, it is the reflection's coordinates you need to measure.

Hint: You can define multiple viewpoints and switch between them using the `pgup`/`pgdown` keys

Task 3 — Skewed Frustum Perspective Viewpoint:
Check the documentation for `SMViewpoint` in Candy/HVR. Use it instead of the viewpoint from the previous example. With the correct numbers entered, this node will provide an off-axis/skewed frustum. Compare with the results from the previous task, at least once with a view that is at an acute angle. How does the frustum affect the perpective (symmetric vs. off-axis)? How can this affect stereoscopic effects? How should head tracking be used?

The view is specified relative the display system, so we need to manipulate the scene graph to enable navigation around the scene (or individual objects). Most scripts in the Candy package are well documented and can be found at `/usr/share/h3d/candy/python/` in the Linux laboratory and at `C:/VRLaboratory/H3D-2.2-trunk/Candy/python` in the VR laboratory, but should be adressed as `urn:candy:python/XXX` in the code (where `XXX` is the file name of the script).

The following tasks should be first set up in the Linux laboratory, where you can get guidance from a supervisor, and subsequently tested in the VR laboratory for full understanding of the VR principles. The mouse will act as wand for testing interaction in the Linux laboratory while you will have interaction through full 3D force feedback stylus devices with both positional and orientational tracking in the VR laboratory.

Task 4 — Navigation:
Use one of the navigation scripts `MultiTouch`, `ManualRotation` or `ManualTranslation` to allow navigation of one or several of your scene objects. While `MultiTouch` is the most powerful, it also requires two wands or haptics devices to be fully functional, which is only available on Haptik-01.

4 Haptic Effects

Now that you have a working scene graph with correct viewpoint settings, it is time to experiment with haptic interaction. This can be prepared in the Linux laboratory since we have a haptics simulation through the mouse device. You can also move the mouse device in depth (z-direction) by pressing down the middle button (or scroll wheel) and moving the mouse up or down. Up and down mouse motion will make the haptic pen move in and out of the screen, respectively.

Task 5 — Basic touch:
Assign haptic surface properties to some objects in your scene. Find a surface model that supports friction. Experiment with touching object with the haptic pen.
Hint: `X3DAppearanceNode` has a field for holding an instance of a subclass of `H3DSurfaceNode`. Check the reference manual for a suitable combination of subclasses.

Task 6 — Varying stiffness:
Set useRelativeValues="false" and experiment with stiffnesses property (never more than 1500 N/m).

Hint: You may use several object with different properties and use different colours to distinguish between them.

The control system that keeps the pen at or outside of surfaces is a PID controller with the haptic pen as process variable, the surface as the desired setpoint and the force feedback as the control signal.

Task 7 — Damping surface:
Experiment with different levels of damping in the range 0–5 Ns/m. What difference does the damping make when touching, pressing or tapping a hard ($\sim$1500 N/m) or a soft ($\sim$200 N/m) surface? Why?

Task 8 — Realistic materials:
Experiment with different dynamic and static friction. Suitable values are in the range 0–2. Combine diffuse and specular colour, stiffness and friction properties to simulate at least three different materials. Would anyone be able to identify them? Even without alternatives?

5 Interactive Sound

The haptic workstations are configured for 3D sound based on Head-related Transfer Functions (HRTF) technology. This can give a better spatial effect of sound in the virtual environment and even let you hear the direction to a sound source.

To add sound to an X3D file you will need to use the `Sound` node, which specifies the sound position, and the `AudioClip` node, which loads the audio file and handles playing the clip. The `spatialize` field should be set to `true` since we want the sound to emerge from a certain position.

The `AudioClip` node is an `X3DTimeDependentNode`, which means that it can automatically activate at a certain time and may deactivate at another time, if set. You can, however, make it start when touching an object. For this you need to use a combination of several functions. Read the documentation of each to understand what actually happens.

`TimeTrigger` is an *engine* (documented in H3D reference manual) that provides the current time as output whenever it gets any single boolean input. This time can be used as start time of any `X3DTimeDependentNode`.

`BooleanFilter` is an *engine* (documented in H3D reference manual) that stops certain boolean events and lets others through. This can be used to stop *release* events so that only *push* events start the sound.

`MFtoSFBool` is an *engine* (available as a script in the Candy package) that extracts a single value from an `MFBool` to provide an `SFBool` as output. This can be used to convert `MFBool` output (an array) from one engine to `SFBool` (single value) to use as input for another engine.

Task 9 — Interactive sound:
The `isTouched` field, available in all geometry nodes (sub classes to `X3DGeometryNode`), provides a list of boolean values (`MFBool`), one for each haptic device on the system. When the geometry is touched with a haptics device its corresponding value becomes `true`. Put at least two sound sources at the position of two different objects and make their respective sounds activate when touching the objects.

[Note: For spatialized sound to work, the sound file has to be mono.]

[Note: Try to find a sound file with as little acoustics as possible. Also, well known sounds with rich frequency contents are easier to localize by hearing.]

Task 10 — From behind:
Put also at least one sound source somewhere behind the user to see if the sound spatialization can make it sound like the sound if coming from behind.

Filtering can be used to create a sense of the environment. The most simple example is the echoes in a large room, but also more complex sound behaviour, such as the sound in a small room connected to a larger room, or a sound source behind a wall, can be perceived by a user and improve their impression of the environment.

The standard X3D Sound node does not support sound effects so you will have to use the VRSound instead, which is an extension of the Sound node. This also accepts subnodes of `HVRSoundEffectNode` and `HVR-SoundFilterNode`.

Task 11 — Environment:
Use the sound nodes available in the Candy/HVR package to create the impression of a more advanced setting, such as a room with an opening to a large cathedral. You may use *presets* in this task. See if it is possible to identify the environment from the sound alone.