TNM086 – VR-technology
2. VR Workbench

November 21, 2016

1 Introduction

The topic of this lab exercise is Virtual Reality rendering, interaction and navigation. The aim is for you to gain insight in how tracking and display hardware interact with VR software to implement an efficient interface to the virtual environment, with particular focus on advanced features such as navigation, interaction and manipulation.

1.1 Examination

The tasks of this exercise must be presented by showing the running program and the code to a supervisor. You must also be able to answer questions regarding the mathematical and Human/Computer interaction principles as well as the tasks and the implementation of these. Complete all tasks before engaging a supervisor for examination.

The exercise will be examinated in the VR laboratory, using the haptic equipment, so when you have finished all tasks, engage a supervisor to request examination. They will then schedule slots for examination in the VR laboratory.

1.2 Equipment

The most important part of this exercise will be performed on the VR workbench in the VR laboratory. Much of the programming, however, can also be done on any computer with the required software installed. Navigation and interaction experiments must be conducted in the lab.

1.3 Software

In this exercise you will be using the Simple Graphics Cluster Toolkit (SGCT) to configure a VR display and handle state synchronization. SGCT includes functionality for cluster data and frame synchronization so you final application will run also in the VR Dome and on any other similar supported platform.

The implementation of 3D graphics will be done using OpenSceneGraph (OSG), so the completion of the associated exercises is encouraged before starting this one. You are also encouraged to read through the instructions before booking the VR equipment and prepare all tasks on other computers, for example in the Linux lab.

There is a code stub available that shows how SGCT is used together with OSG, tracking, wand keys and controls, and keyboard input. Keyboard input can be used in place of the wand during initial development.

SGCT uses GLM for linear algebra while OSG uses its own, so a basic understanding of both and the difference between their respective conventions is of importance.

There are several SGCT configuration files specially designed for this exercise available together with a stub. These are set to run SGCT in a simulated cluster in both the Linux lab and the VR lab. You will need to start two instances of the program to make it run. Use these or equivalent command lines in the Linux lab:

```
./lab2 -config <path-to-config-file> -local 0
./lab2 -config <path-to-config-file> -local 1 --slave
```

and these in the VR lab:

```
lab2.exe -config <path-to-config-file> -local 0
lab2.exe -config <path-to-config-file> -local 1 --slave
```

1.4 Tracking Simulation

It is possible to start a VRPN tracking server in the Linux lab, that reads off mouse movements and use that to simulate tracker movements. These are sent through VRPN as sensor 0 on tracker H3D@localhost.

Start the VRPN server in the Linux lab by the following command:

```
H3DLoad urn:candy:x3d/VRPNServer.x3d
```

Only a single button is supported (triggered by the right mouse button) but more can be simulated by keyboard input in SGCT. Depth movements are performed by dragging with the middle button, up for away and down for closer.

1.5 Documentation

In this exercise you will be using OpenSceneGraph, so the documentation from the exercises you have earlier done on those APIs may be very useful. Here are listed some sources of additional documentation.

- SGCT Wiki
  https://c-student.itn.liu.se/wiki
  - Getting Started Guide
  - Programmer Reference
  - Code samples

- OpenSceneGraph Quick Start Guide (free in PDF format)
  http://www.lulu.com/shop/paul-martz/openscenegraph-quick-start-guide/ebook/product-17451155.html
- OpenSceneGraph Reference Manual
  http://www.openscenegraph.org/projects/osg/wiki/Support/ReferenceGuides
- OpenGL Mathematics (GLM)
  http://glm.g-truc.net
  - Reference manual
  - Code samples

2 Basic VR Programming

In this part of the exercise you will use a scene graph API, preferably OSG, to implement basic graphics and interaction. It is recommended that you record your progess and document how to select between the different functionalities you have implemented.

Task 1 — Understand SGCT + OSG:
Read the code of the stub and try to understand how SGCT interacts with OSG. You should be able to answer the following questions: which states are shared between the nodes, where are the states updated and used, and how is the scene graph structured?

Task 2 — Port Your Application:
Use SGCT to display an OpenSceneGraph application on the VR system, or any other scene graph system of your choice. You may use your code from the OpenSceneGraph exercise. The display should be correcly rendered in stereo and use the head tracking.

Task 3 — Understand the Scenegraph:
Draw the scene graph of your current program. Which transforms do you have and what are their respective purposes.

Explicitly calculating the selection of objects is most simply done by calculating the intersection or closeness between a line pointing from your wand and spherical objects, such as planets. With a scene graph API there are usually methods for automatically checking for intersection with parts of the scene graph, with bounding spaces or with geometries.

Task 4 — Interaction with Objects:
Enable selecting objects on distance in your virtual environment. Indicate the selection in some way, for example by making the bounding box or sphere visible or changing the colour of the object. With OpenSceneGraph you use the `IntersectionVisitor` feature.
Hint: Check the osgintersection.cpp example[a].
[Note: Observe that independent states (e.g. tracker data) should be updated in pre-sync, synchronized with slaves, and then used to update dependent states in post-sync-pre-draw. States that are updated in the draw function may change between drawing for the left and the right eye.]

[a] http://trac.openscenegraph.org/projects/osg//browser/OpenSceneGraph/trunk/examples/osgintersection/osgintersection.cpp

3 Advanced VR Programming

In this part of the exercise you will implement features where the scene graph API can be of much more assistance. Explicit navigation and manipulation can generate quite complex software if care is not taken.

Task 5 — Explicit Navigation:
Use the wand information you can get from VRPN to control the transform of your scene so that you can navigate in your virtual environment. Implement "pointing mode" and "cross hair" mode. Start by navigating with constant speed.
[Note: Since this is VR you have to make sure that your navigation is smooth and will not make the user dizzy or disoriented.]

Task 6 — Control Flying Speed:
Implement hand controlled acceleration and hand controlled speed for the navigation modes from the previous task. It is not necessary, but recommended, to try out a "dead zone".

Task 7 — Alternative Navigation: (optional)
You may also try implementing any other mode of navigation of your choice. Alternatives are, but not limited to, virtual controls, attractors/repellers, dynamic scaling and goal driven navigation. For example, let the user select an object to execute a smooth, continuous navigation that makes the selected object end up in front of the user.

Task 8 — Manipulation of Objects:
Implement so that you can manipulate at least two different objects in your scene. The manipulation should be based on wand movements and include both translation and rotation of the object. How do you move an object at distance? What is the centre of rotation?
Hint: One way to solve this problem is to extract tracker and model matrices and use linear algebra to make the model follow the wand motions.

Task 9 — Object Scaling:
Implement a means for scaling the selected object. You may choose any metaphor for this manipulation.

4 Putting All Together

Task 10 — Implement a Game: (optional)
Select OpenGL or OpenSceneGraph of your own preferences to implement a VR game that makes use of both head tracking to provide natural navigation and the wand for absolutely positioned interaction.