TNM086 – VR-technology
6. Haptics

October 11, 2012

1 Introduction

The subject of this lab exercise is haptics. The aim is to get an understanding of how haptic equipment and software works and some properties of our sense of touch.

In the exercise you have to program in X3D to build your 3D world, and in Python (a scripting language) to implement dynamic behaviour. You are encouraged to read through the instructions before booking the haptic equipment and prepare all tasks on other computers, for example in the Linux lab.

1.1 Examination

The tasks of this exercise must be presented by showing the running program and the code to a lab assistant. You must also be able to answer questions regarding the physical principles, the tasks and the implementation of these. Complete all tasks before engaging a lab assistant for examination.

1.2 Equipment

The exercise will be performed on the haptic workstations in the VR laboratory. Much of the programming can also be done on any computer with H3D API installed, but haptic experiments must be conducted in the lab. Also, images for texturing can be created on another computer where appropriate software is available. Note that the size of textures is still preferably a power of two.

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 pen 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.

1.3 Software

The software you will use in the development is known as H3D API. It is an open source scene-graph API, based on the X3D standard, used to build a scene-graph that both the visual and the haptic rendering algorithms will work with. The programming interface consists of

- An extended X3D parser for haptic feedback,
- a Python interpreter, and
- a C++ class library

You will work with X3D and Python in this lab and both these file formats can be edited in Emacs or Notepad++ that are installed on the haptic workstations. You define your 3D scene in X3D and when needed this scene will include script nodes that loads Python scripts to control effects in your scene. To load your scene you first open a shell with H3D set up as needed. In the linux lab this is any shell. On the haptics workstations in the VR laboratory you start the H3D bat file on the desktop. Then load your file by executing

```
H3DLoad scene.x3d
```

1.4 Documentation

To pass this exercise you will have to gain experience in several different fields. You will have to learn X3D and Python, and learn how to setup a routing network in a scene-graph system. Important information can be found in several locations, such as:

- Tutorials
  - Python at `http://docs.python.org/release/2.6.7/`
  - H3D at `http://www.h3dapi.org/modules/mediawiki/index.php/H3DAPI_Tutorials`
- Documentation at `http://www.h3dapi.org/modules/judoc`
  - H3D API Manual
    - H3DLoad (1.5.0), pp 16
    - Designing for H3D, pp 19–21
    - Fields (4.1), pp 22–27
    - Nodes (4.2.0), pp 27
    - Python (4.3), pp 31–39
    - Surface Properties (4.4.5), pp 43
  - Reference Manual
    - Nodes

  * Properties
  * Fields

2 Haptic Degrees-of-Freedom

This part of the exercise is about different degrees-of-freedom for the haptic interface. Different devices will of course have different such properties, but our devices have a common combination of input and output degrees-of-freedom.

Task 1:
Create a simple object, for example a box, at the centre of the 3D environment. Assign a haptic surface with default properties.

Tip: H3D uses metric units so you may use a ruler to determine approriate size and translation before specifying them in your X3D file.

The default navigation system in X3D is not suitable for our semi-immersive workstations. It is based on moving the viewpoint and in a VR environment the viewpoint is defined by the location of the viewer, not by the navigation. Therefore, you need to turn off the navigation by adding a `NavigationInfo` node with the `type` field set to "None". If you want to be able to rotate your objects, take a look at the `Rotate.py` script in `c:/Haptics/H3D/H3D_2.1/Candy/python`.

Task 2:
Load your X3D file and explore the object with the haptic pen. What effects do the input degrees-of-freedom have on the exploration? How many input degrees-of-freedom does this haptic device have? What effects do the output (feedback) degrees-of-freedom have on the exploration? How many output degrees-of-freedom are there?

3 Haptic Rendering and Stability

This part of the exercise is about properties of the haptic rendering and stability issues. Most haptic rendering algorithms are based on a PD regulator (why not PID?) and you will explore the stability of this regulator for different parameters.

Tip: Make a new copy of your X3D file from the previous part and continue from there.

Task 3:
Use the smooth surface (`SmoothSurface`) for your object. Set `useRelativeValues` to false — that way we can specify parameters in metric units. Try different levels of stiffness (no more than 1500 N/m). At what point do you experience instabilities? Does it make a difference where in the workspace you touch a stiff surface? What difference does the firmness of your grip do?

Tip: You may use several objects with different properties and different colours to distinguish between them.

Task 4:
Try 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?

4 Material Properties and Perception

This part of the exercise is about exploring the specification and perception of material properties. You will explore the parameter space of haptic materials and test your perceptual limits.

Task 5:
Create multiple objects at different positions, all with frictional surfaces but with different friction. Initially set stiffness to about 800 N/m. Try various combinations of dynamic and static friction. What is the least noticeable difference you can detect? Can you use friction in combination with stiffness to simulate real materials, such as rubber?

Task 6:
Try different combinations of friction and stiffness. Can you simulate real materials, such as rubber or glass?

Task 7:
Create an object with `DepthMapSurface` and assign a grey scale texture, for example with Perlin noise. Try different values of `maxDepth`. What does that parameter do? What is the smallest noticeable value of that parameter?

Tip: If you use a depth map surface, apply also a similar colour texture to create a natural mapping between visual and haptic impressions.

Task 8:
Create geometrical bumps on a flat surface by placing one or more small objects under a larger `Rect-angle2D` so that their surfaces stick out of the rectangle. What is the smallest bump height you can detect? Compare with moving your finger over bumps on a real surface, for example over the edge of a paper lying flat on a table. What is causing the differences?

5 Dynamic Behaviour

In this part of the exercise you will do some basic scripting and set up routing to induce dynamic behaviour in the haptic environment. Some or all of these examples may induce unstable behaviour so it is of utmost importance that you keep a firm grip on the haptic pen whenever these programs are running.

Task 9:
Create several objects with haptic surfaces. Create a Python script that takes a boolean as input to generate a colour value. Connect the `isTouched` field of the object's geometries to the engine field of this script, and route the result to some material property of the objects so that touching the objects affects their appearance.

Tip: Create one field in the Python script for each object, for example by instantiating the same class several times.

Task 10:
Select and complete at least one of the following optional tasks involving Python scripts and force feedback. You will have to route the probe position to your script and use this in the script to calculate the feedback. The Python script is running in the graphics loop (∼100 Hz) and not the haptics loop (∼1000 Hz). What implications does this have?

Tip: You can get the current haptic device into your X3D file by writing
`<IMPORT inlineDEF="H3D_EXPORTS" exportedDEF="HDEV" AS="HDEV"/>`

Task 11: (optional)
Create a sphere *without* haptic surface. Use the `ForceField` node together with a Python script to create haptic rendering of the sphere surface. Use Hooke's law to calculate the force from penetration depth (penalty algorithm). What stiffness can you use? Start with low stiffness ($\leq$ 100 N/m) and gradually increase the value.

Task 12: (optional)
Use the `LineSet` geometry to draw a line. Use the `ForceField` node together with a Python script to create a haptic pull towards the line. Use Hooke's law to calculate the force from the distance from the line (penalty algorithm). Implement an "escape distance", so that there is no feedback if the haptic probe is further away from the line than a specified distance. What stiffness can you use? Start with low stiffness ($\leq$ 100 N/m) and gradually increase the value.

Task 13:
Modify your script from the task above so that instead of using the probe position to calculate the force it uses a point a few centimetres in front of the probe (a virtual probe). The orientation of the haptic instrument, which has no feedback, is thus used as an input parameter to the feedback. What effect does this have on the feedback? Does it affect the stability? What if you quickly change the pen orientation?