Virtual reality application development in Linux

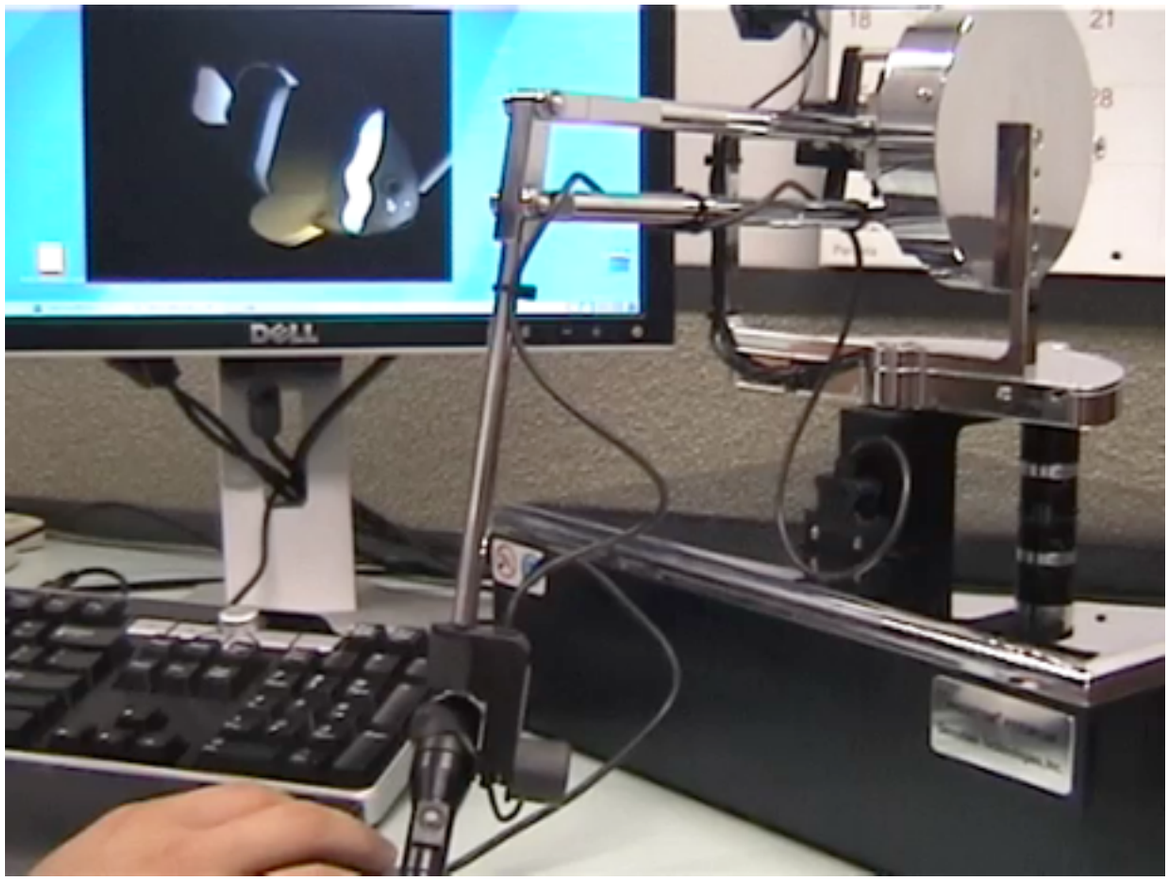
by Ioanna Ioannou
ioannoui@unimelb.edu.au

Outline

- PART I: The hardware
- PART II: The software
- PART III: H3D API
- PART VI: Code example
- PART V: Problems and challenges

PART I : The hardware

Force feedback devices



Force feedback devices

- Force Dimension haptic devices
  $41,000-$115,700
- Sensable haptic devices
  $3,790-$115,845
  - PHANToM OMNI - $3800
    - Cheapest Sensable, 6DOF movement, 3DOF force feedback, max force 3.3N
- Novint Falcon - $300
  - Most affordable by far
  - Designed for gaming
  - 3DOF movement & force feedback, 8N max force
  - No official linux support

Force feedback devices - Novint Falcon

3D/stereo vision

- 3D vision goggles
  - eg. Vuzix iWear - $470
    - compatible with linux and a range of existing games
- CRT monitor + Shutter goggles - $1400
  - Any >100Hz large CRT monitor
    eg. NEC AccuSync 120-BK (~$300)
  - Crystal eyes goggles ($900) + emitter ($220)
- Stand alone 3D monitors and projectors ($3000+)
  - Very cool but expensive and typically don't support linux

PART II : The software

VR software

- Visuo-haptic software typically runs two threads
  - Graphics loop (30Hz) - generates/updates graphics
  - Haptics loop (1500Hz) - generates/updates haptics
- Some of the responsibilities of VR software:
  - Communication with haptic device
  - Tread handling
  - Collision detection
  - Haptic rendering
  - Displaying and updating graphics
- Typically we use an API or toolkit that handles most of the above tasks

Open source VR tools for Linux

- Several tools are in very early stages of development, for example:
  - OpenSceneGraph Haptic Library (http://www.vrlab.umu.se/research/osgHaptics/)
  - Panthera (http://panthera.sourceforge.net/)
  - Novint Falcon Open Source Library (http://sourceforge.net/projects/libnifalcon)
- Two mature toolkits are available on Linux:
  - Chai3D (http://www.chai3d.org/)
  - H3D API (http://www.h3dapi.org/)

PART III : H3D API

Focus on H3D API

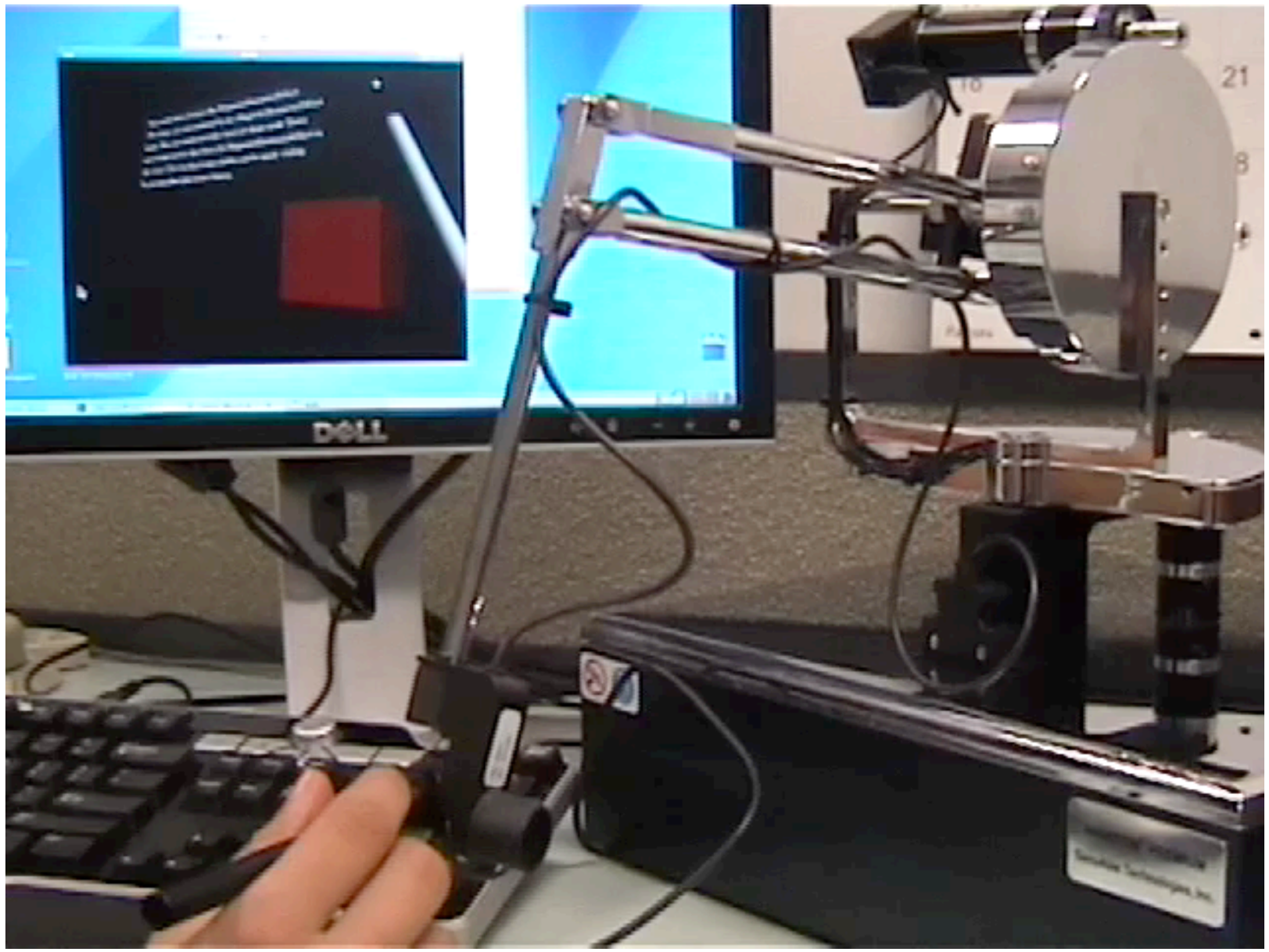
- Dual licensing model
- Cross-platform
- Supports all haptic devices available for linux
- Supports stereo projection and several specialised immersive displays

- One unified scenegraph for both graphics and haptics
- Uses open standards such as X3D and OpenGL
- Written in C++ for efficiency
- Three ways of programming: X3D, Python, C++
- Designed for extensibility and rapid development

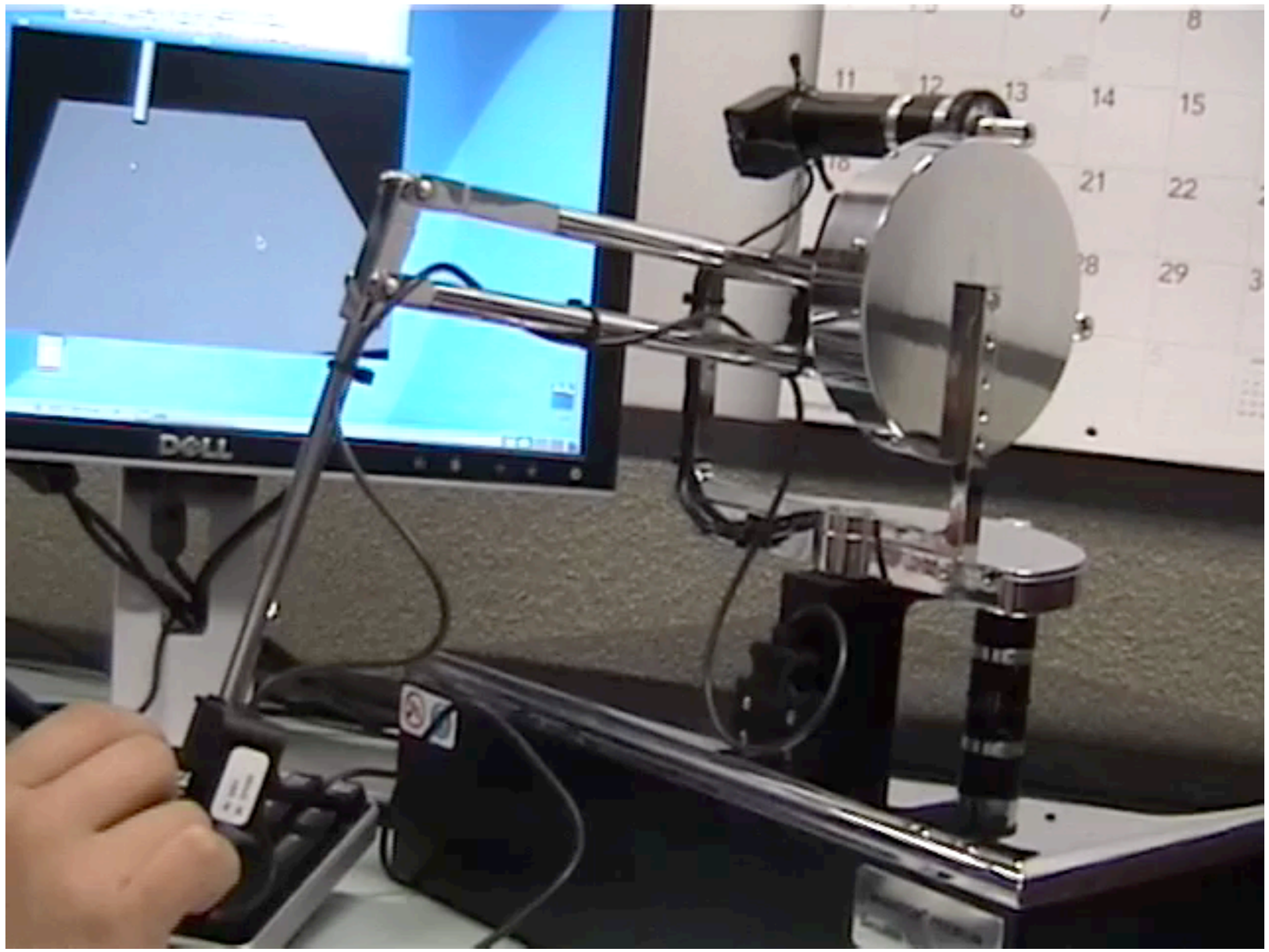
Fields, nodes and routing

- Fields are
  - data containers that store and manipulate data properties
  - arranged into a directed graph where events are passed from field to field according to the field network connections
  - able to perform operations on data as it passes through the network
- Nodes are containers and managers of fields and field networks
- Routes are field connections
  - one-to-one, many-to-one or many-to-many
  - connected fields must be of the same type
  - facilitate event handling

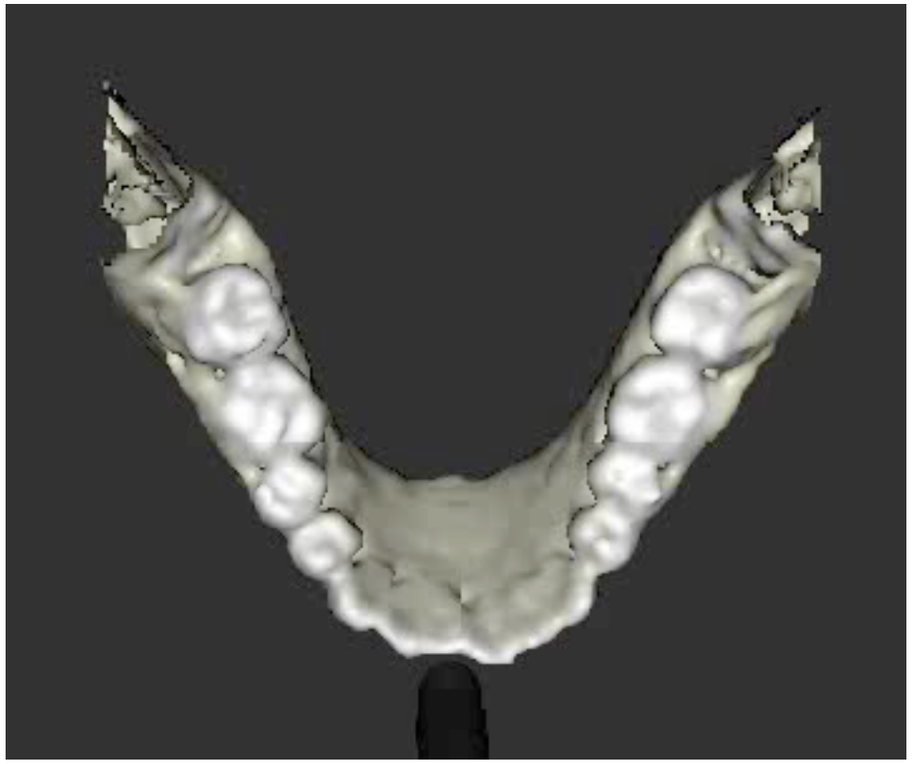
Magnetic surface example



Paintable texture example

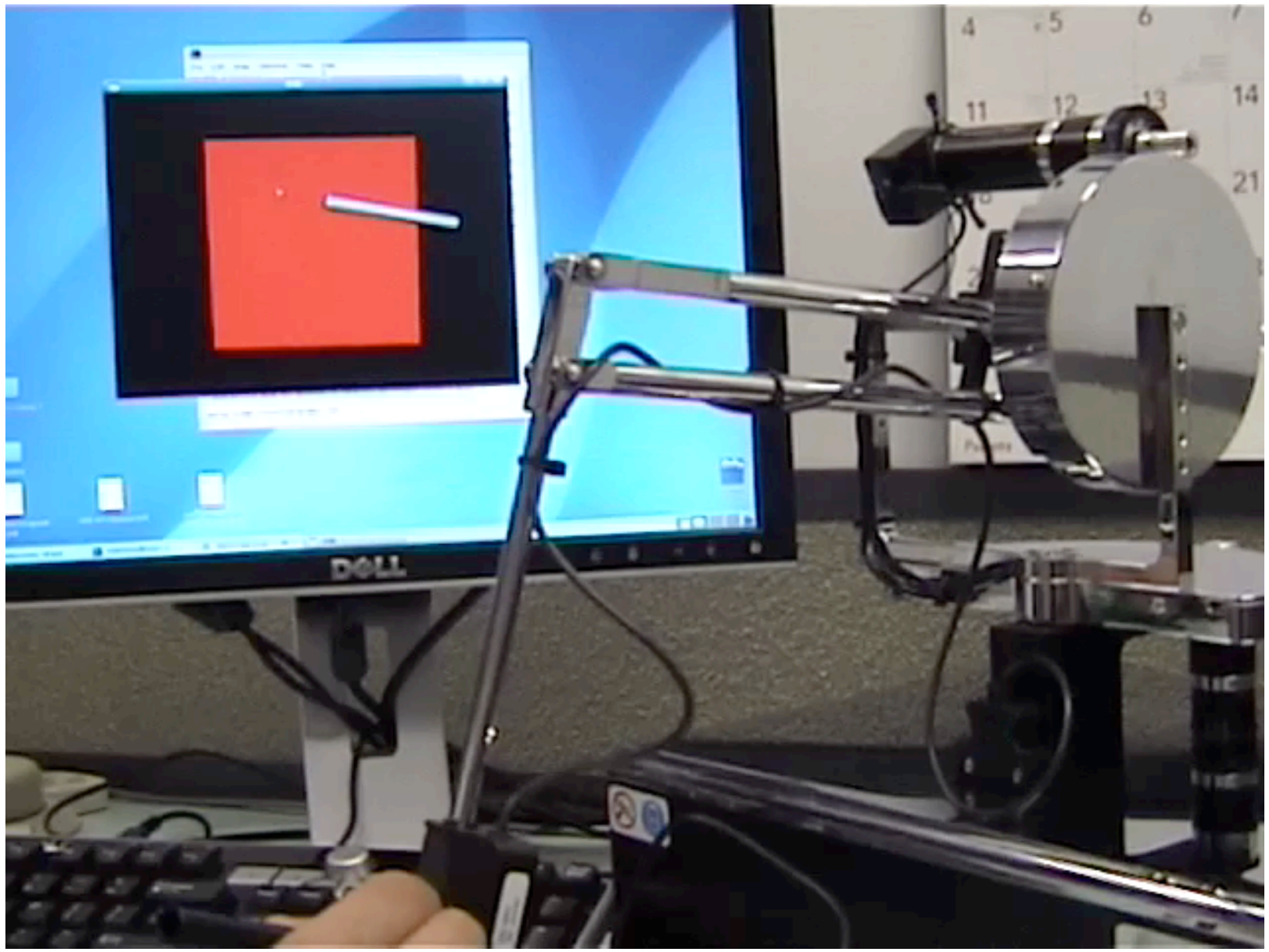


Dental “sculpting” prototype



Flexible surface example

Dynamic transform example



PART IV : Code example

A simple scene

- This code draws a cube at the default position on the screen
- Graphics-wise, H3D supports everything you can do with X3D
- Haptics are added by adding a Surface node to the Appearance of a Shape
- <X3D> and <Scene> tags are not enforced by H3D

```
<X3D version='3.0'>
  <Scene>
    <Group>
      <Shape>
        <Appearance>
          <Material diffuseColor="1 0 0"/>
          <SmoothSurface stiffness="1.0"/>
        </Appearance>
        <Box DEF=”BOX1” size="0.2 0.2 0.2"/>
      </Shape>
    </Group>
  </Scene>
</X3D>
```

Routing using X3D and Python

- X3D nodes can be accessed through Python and C++ and vice versa
- Each field has an update() function that updates the field when an event is received
- An event is by default the same type as the field itself
- The event structure contains the new value for the field and the time that the event occured
- The TypedField modifier allows routing between fields of different types

```
from H3DInterface import *

class ChangeColor( TypedField( SFColor, MFBool ) ):
  def update( self, event ):
    # Check that MFBool field is not empty and check
    # whether first value is true (i.e. box is touched)
    if len(event.getValue()) > 0 and event.getValue()[0]:
      # If touched return green color
      return RGB(0, 1, 0)
    else:
      # If not touched return red color
      return RGB(1, 0, 0)

# Create an instance of the field
changeColor = ChangeColor()
```

Adding the routes

```
<X3D version='3.0'>
  <Scene>
    <Group>
      <Shape>
        <Appearance>
          <Material DEF="FunkyMaterial" diffuseColor="1 0 0"/>
          <SmoothSurface stiffness="1.0"/>
        </Appearance>
        <Sphere DEF="BOX1" size="0.2 0.2 0.2"/>
      </Shape>
    </Group>

  <PythonScript DEF="PS" url="change_colour.py" />

  <ROUTE fromNode="BOX1" fromField="isTouched"
         toNode="PS" toField="changeColor" />
  <ROUTE fromNode="PS" fromField="changeColor"
         toNode="FunkyMaterial" toField="diffuseColor" />
  </Scene>
</X3D>
```

PART V : Problems and challenges

Problems and challenges

- Linux-specific:
  - “Linux support” = it runs after a few hours of installation/compilation hassles
  - Device drivers are not open-source
  - Open source Falcon driver not integrated with H3D or Chai3D
- H3D-specific:
  - Difficult to debug large programs across X3D/Python/C++
- General VR challenges:
  - Simple real life things can be VERY complicated to represent in VR! Need to simplify to appropriate level of realism.

Visit MUVES at Melbourne University to find out about our exciting VR projects and see this technology in action! For more info go to:

http://www.muves.unimelb.edu.au

Questions