Notes on tutorial #7 (for week 10 - March 12, 2004)

April 8, 2004

Q1: Write a small forms/web-based application which asks people to enter in some personal details (name, identification number), and then puts up a new form which allows them to select a day-of-the-week, and an hour-of-the-day timeslot for a meeting with the head of the department. The result of this will be either:

success - the timeslot is allocated to that person, with a web page clearly stating that, or
failure - a web page is displayed which allows the user to try again, and which shows the remaining free time slots

Answer: This will be up to the student...

Q2: Research the use of animated 3D. Write or demonstrate a technique that gives an animated (i.e. changing) display of a group of cubes that are rotating, and moving relative to each other. Describe your technique.

Answer: I used animated VRML, something like this:

```
CODE LISTING s1.wrl Page 1/1
VRML V2.0 utf8
Viewpoint { position 0 5 325 orientation 1 0 0 0 }
DEF transform1 Transform { translation 4 0.5 0 children [ Shape { appearance Appearance { material Material { emissiveColor .8 0 0 } } geometry Box { size 15 15 15 } } ] }
DEF transform2 Transform { translation 4 0.5 0 children [ Shape { appearance Appearance { material Material { emissiveColor 0 .8 0 } } geometry Box { size 15 15 15 } } ] }
DEF transform3 Transform { translation 4 0.5 0 rotation 0.0 1.0 0.0 0.0 children [ Shape { appearance Appearance { material Material { emissiveColor 0 0 .8 } } geometry Box { size 25 25 25 } } ] }
DEF time TimeSensor { cycleInterval 5 loop TRUE enabled TRUE startTime 1 }
DEF position1 PositionInterpolator { key [ 0 0.5 1 ] keyValue [ 95 -37.5 0, -39.0522 41.0994 17.8899, 95 -37.5 0 ] }
DEF position2 PositionInterpolator { key [ 0 0.5 1 ] keyValue [ 6.0123 19.1845 −12.0646, −5 −37.5 0 ] }
DEF position3 PositionInterpolator { key [ 0 0.5 1 ] keyValue [ −105 −37.5 0, 19.7851 21.74 8.0744, −105 −37.5 0 ] }
DEF ThingSpinner OrientationInterpolator { key [ 0.0, 0.5, 1.0 ] keyValue [ 0.0 1.0 0.0 0.0, 0.0 1.0 0.0 3.14, 0.0 1.0 0.0 6.28 ] }
ROUTE time.fraction_changed TO position1.set_fraction
ROUTE time.fraction_changed TO position2.set_fraction
ROUTE time.fraction_changed TO position3.set_fraction
ROUTE time.fraction_changed TO ThingSpinner.set_fraction
ROUTE position1.value_changed TO transform1.translation
ROUTE position2.value_changed TO transform2.translation
ROUTE position3.value_changed TO transform3.translation
ROUTE ThingSpinner.value_changed TO transform3.set_rotation
```
Q3: Find/install a programming system which gives a 3D interface. Write a small program which spins a cube in a 3D canvas, with the speed of rotation changed by a scale.

Answer: The TkOGL package seemed fairly simple to set up to add 3D to Tk. It creates a 3D canvas - OGLwin, and the following code (extracted from the TkOGL distribution) shows how to use it. I have omitted the code used to create the display list, but it is on the web site and in the distribution:

```
package require Tkogl

# code to create display list
init_polybuild
pack [OGLwin .gl -aspect 1 -stencil 8] -side left -fill both -expand yes
set dlist [.gl newlist]
set mode fill
pack [frame .cmd] -side left -fill y
pack [frame .cmd.main] -side top -fill x
polybuild .gl hexahedron regular $dlist $mode
.gl eval \
-‐matrixmode projection \n-‐loadidentity \n-‐perspective 20 1 0.5 20\n-‐matrixmode modelview \
-‐loadidentity \n-‐lookat 0 0 0 0 0 0 1 0 \n-‐pushmatrix \n-‐material front ambient 0.3 0.3 0.3 \n-‐material front diffuse 0.5 0.5 0.5 \n-‐material front specular 0.3 0.3 0.3 \n-‐material front shininess 40 \n-‐light light0 position 1 1 1 0 \n-‐light light0 ambient 1 1 1 \n-‐light light0 diffuse 1 1 1 \n-‐light light0 specular 1 1 1 \n-‐clearcolor 0 0 1 \n-‐enable lighting -‐enable light0 -‐enable light1 \n-‐enable depthtest
.gl main -‐clear colorbuffer depthbuffer stencilbuffer\n-‐call $dlist
scale .speed -‐label Speed -‐from 0 -‐to 20 -‐length 10c
pack .speed
set xrot 1
set yrot 1
rotate .gl
```