Visualization

✓ In visualization, we are concerned with *exploration*

✓ In computer-graphics, we are concerned with *rendering*
The use of 3D

✔ Analog with real-world physics.

✔ 10-fold improvement in item density with 3D.

✔ Familiarity with spatial location helps reduce visual clutter.

✔ Need sufficient visual cues.
OpenGL

✔ SGI in-house graphics system

✔ Now a widely accepted graphics standard

✔ Standard on UNIX and Windows

✔ API supports rendering, buffering, anti-aliasing, shading, colouring, texture-mapping, a display list, Z-buffering...

OpenGL Application
OpenGL source

```c
#include <GL/glut.h>

void
Teapot (long grid)
{
    /* ... code to construct drawlist of teapot here. */
}

static void
Init (void)
{
    glEnable (GL_DEPTH_TEST);
    glLightModeli (GL_LIGHT_MODEL_LOCAL_VIEWER, local_view);
    /* Lighting model, materials... */
}

static void
SpecialKey (int key, int x, int y)
{
    switch (key) {
    case GLUT_KEY_UP:
        rotX -= 20.0;
        glutPostRedisplay ();
        break;
    /* Move in other directions */
    }
}

static void
Draw (void)
{
    glClear (GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glPushMatrix ();
    /* ... translations ... */
    glCallList (teaList);
    glPopMatrix ();
    glutSwapBuffers ();
}

int
main (int argc, char **argv)
{
    glutInit (&argc, argv);
    type = GLUT_RGB | GLUT_DEPTH;
    type |= (doubleBuffer) ? GLUT_DOUBLE : GLUT_SINGLE;
    glutInitDisplayMode (type);
    glutInitWindowSize (300, 300);
    glutCreateWindow ("Teapot");
    Init ();
    glutReshapeFunc (Reshape);
    glutKeyboardFunc (Key);
    glutSpecialFunc (SpecialKey);
    glutDisplayFunc (Draw);
    glutMainLoop ();
    return 0;
}
```

Java3D & VTK

- 3D OO toolkits
- VTK is open source
  - C++ class library, and
  - interface layers for Tcl/Tk, Java, and Python.

---

CS3283 - Hugh Anderson's notes.  Page number: 338

---

CS3283 - Hugh Anderson's notes.  Page number: 339
Network traffic application

To help answer questions such as the following:

- Which segments carry the most traffic?
- Which sections of the network are down?
- At what times, and where do traffic bottlenecks occur?
- ...

Application elements

Following elements are represented:

- Background: - to convince the viewer that the display is *three dimensional*...
- Nodes: - a computer, a network device...
- Traffic: - the amount of traffic flow...
- Protocol: - the *type* of traffic...
- ...

## Node representation

![Node representation](image)

## Rendering speed

<table>
<thead>
<tr>
<th>Machine</th>
<th>Rendering speed</th>
<th>Computer (a)</th>
<th>Computer (b)</th>
<th>Computer (c)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workstation</td>
<td>485,000 Δ/sec</td>
<td>0.485 frames/sec</td>
<td>11.5 frames/sec</td>
<td>69 frames/sec</td>
</tr>
<tr>
<td>PC1</td>
<td>30,000 Δ/sec</td>
<td>0.03 frames/sec</td>
<td>0.71 frames/sec</td>
<td>4.3 frames/sec</td>
</tr>
<tr>
<td>PC2</td>
<td>11,000 Δ/sec</td>
<td>0.011 frames/sec</td>
<td>0.26 frames/sec</td>
<td>1.6 frames/sec</td>
</tr>
</tbody>
</table>
Some representation methods allow different *levels of detail*.

In VRML an object may be represented in different ways depending on how large it is.

If the object is near you, it could be represented in detail, but if it is a long way away, the representation could be as simple as a coloured square.

```vbnet
LOD {
  range [20]
  level [
    Shape{ #full detail 16 sided cone
      appearance Appearance { material Material { ... } }
      geometry Extrusion{ ... }.
    }
    Shape{ #low detail 4 sided cone
      appearance Appearance { material Material { ... } }
      geometry Extrusion{ ... }
    }
  ]
}
```
LOD

✔ If the distance from the user to the object is smaller than the first range value specified, then the first version is drawn.

✔ If the distance is greater than the last range specified, the last version is drawn.

Traffic and protocols

Draw a line between nodes.
A line indicates source and destination, but not the amount of traffic:

1. Colour coding (black through red to white for maximum traffic),

2. Line width, and

3. The length of partial lines, as discussed in Eick’s papers.
Partial lengths

Trend representation

✔ Graphing

✔ 4D visualization methods

✔ Encode previous *on-top-of* the current - *visual* echoes.
Systems

✔ CosmoPlayer VRML viewer,

✔ geomview.

The visualization is not dependant on the navigation or implementation method.
Aggregation

Implementation #1

✔ A data collector

✔ A web page with... a
  ✔ Java program loaded as an applet, and a
  ✔ VRML view of the network.
Web page

<html><head> <title>Sample 3DVNT Page</title> </head>
<center><H1>Sample 3DVNT Page</H1></center>
<center> <embed src="root.wrl" height="600" width="700"> </center>
<center> <applet code="View1.class" width="100" height="10" mayscript>
<PARAM name="segment" value="MACS">
<PARAM name="port" value="9876">
<PARAM name="host" value="opo.usp.ac.fj"> </applet> </center>
OK?</html>
VRML

PROTO CLUSTER [] { ... } # Cluster definition
PROTO KEYBOARD [] { ... } # Keyboard definition
PROTO SCREEN [] { ... } # Screen definition
PROTO GLOBE [] { ... } # Traffic sphere definition

# Some setting up declarations
Background { skyColor .4 .66 1 }
NavigationInfo { type [ "EXAMINE", "ANY" ] speed 400 }
Viewpoint { position 0 400 0 orientation 0 1 0 4 description "Camera 1" }

# Lines, floors and roofs
DEF LINES Transform { ... }
DEF FLOORS Transform { ... }
DEF ROOFS Transform { ... }

# and then the nodes
DEF node1 Transform { ... }
DEF node2 Transform { ... }

# and so on ...

---

VRML nodes

DEF node1 Transform {
  translation 4350 150 4365
  rotation 0 1 0 4.71238
  children {
    KEYBOARD {}
    SCREEN {}
    DEF node1box Transform {
      children {
        Shape { ... }
      }
    }
    DEF node1sphere Transform {
      scale 1 1 1
      children {
        Shape { ... }
      }
    }
  }
}

---
Java 1

Java 2
Summary of topics

In this module, we introduced the following topics:

- Visualization versus computer-graphics
- OpenGL
- (Briefly) Java3D, VTK
- VRML/Java/EAI