Graphical Programming (1)

- Programs to process and manipulate graphical objects and data
- Early standards: PHIGS, GKS
- 2D, 3D graphics libraries
- Level of abstraction
  - Low-level API: OpenGL
  - Scene graph API: Java3D

Graphical Programming (2)

- X-Windows
  - X-protocol, Xlib, window manager
- OSF/Motif
  - Widgets, gadgets, window manager
- MS-Windows
- RenderMan
- OpenGL
- Direct3D

Graphical Programming (3)

- OpenInventor, VRML, X3D
  - Scene graph oriented
  - Engines, sensors, manipulators
- IRIS Performer
- Java3D

OpenGL / Windows System

What is OpenGL?

- OpenGL is a hardware-independent interface to the graphics hardware
- OpenGL does not provide rendering context management functions or means to manage user input
  - Interface between OS and OpenGL needed:
    - WGL / Microsoft Windows
    - GLX / UNIX's X-Windows (direct and indirect rendering)
    - GLUT / OS independent

Java3D

<table>
<thead>
<tr>
<th>Java Applet/Application</th>
<th>Java 3D</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>OpenGL</td>
</tr>
</tbody>
</table>

Scenegraph

- Hierarchical structure
- Corresponds to logical structure of the scene
- Easy design and manipulation of complex scenes
**The Renderer**

- Double buffering, RGB(A)
- Infinite loop: Traverse Graph
- Take care of transparency
- Regions of influence of fog, lights, sound
- Optimized for rendering (display list)

**Node States - Capabilities**

- Node created: all operations legal
- **Live**: Node inserted into scene
  - **capabilities** specify permitted operations (r / w / modify)
    - `setCapability(ALLOW_TRANSFORM_WRITE);`
- **Compiled** (optional):
  - Optimized by renderer
  - modifications restricted
- Capabilities fixed if live or compiled

**Java3D Node Types**

- **Group nodes**
  - `addChild(...)`
  - `BranchGroup`
    - only detachable/relocatable node
    - grouping of Shapes
  - `TransformGroup`
    - transform applied to all children

**Hello3D! / Constructor**

```java
...
Canvas3D c = new Canvas3D(null);
aWindow.add(c);
BranchGroup content = createSceneGraph();
SimpleUniverse u = new SimpleUniverse(c);
u.addBranchGraph(content);// now "live"
...```

**Hello3D! / Scenegraph**

![SceneGraph Diagram]

**J3D-Scenegraph**

![J3D-Scenegraph Diagram]
Hello3D! Content

```java
Public BranchGroup createSceneGraph()
{
    BranchGroup objRoot=new BranchGroup();
    Transform3D spin=new Transform3D();
    Transform3D tmpspin=new Transform3D();
    spin.rotX(Math.PI/4.0d);
    tmpspin.rotY(Math.PI/5.0d);
    spin.mul(tmpspin);
    TransformGroup objTrans=new TransformGroup(spin);
    objRoot.addChild(objTrans);
    return objRoot();
}
```

Java3D Node Types

- **Shape3D**
  - Defines object within the scene
  - Contains:
    - **Geometry**
      - Polygon related information
    - **Attributes**
      - Material definition
      - Rendering mode (wireframe, ...)

Geometry Nodes

- Geometry definition:
  - Coordinates
  - Optional: Normals
  - Optional: RGB(A) color
  - Optional: Texture coord. (u/v/w)
  - For indexed types: Indices

Helper Classes

- **com.sun.j3d.util.***
  - SimpleUniverse:
    - Fast viewing model setup
  - Mouse -> Transform mapping for interactions
  - Simple geometrical Objects (cube, ...)

- **javax.vecmath.***
  - 4D matrix + helper functions for CG:
    - Ortho/persp. Projection, scale, rotate, translate, ...
Appearance

Defines attributes for rendering geometry of an object
- polygon attributes: pts/wirefr./fill
- culling
- rendering attrib.: z-buff., α-blend.
- transp./color (if not per vertex)
- material (reflection coefficients)
- texture images

Lighting

Light source nodes require Bounds delimiting the area of influence
- Ambient
- Directional (infinitely distant)
- PointLight (spot light)

Behaviors

“Events” for the scene graph
- Action executed if criteria met:
  - time triggered, kbd & mouse events, picking of objects, collision, frames elapsed
- Boolean combination of criteria
- Bounded area of relevance
- processStimulus()

Behaviors

- Influence transform. / geometry
- Examples:
  - Mouse controlled navigation
  - Interpolator: time->value
  - ProximitySensor
  - Collision reaction
  - Picking: Click -> list of Shape3Ds below mouse

Viewing Model

- Supports multiple Canvas3D -> stereo rendering, (Cave!)
- Support for tracking.
- Detailed description of viewer’s eyes/ears configuration

Java3D - Further Material

Java 3D API specification documentation & tutorial:
https://java3d.dev.java.net/