Graphical Programming

Eduard Gröller, Johanna Schmidt

Institute of Computer Graphics and Algorithms
Vienna University of Technology
Terms

- Image Processing
- Pattern Recognition
- Computer Graphics
Graphical Data Processing

Modeling
- Modeler
- 3D Data

Rendering
- Transformations
- Clipping
- Shading
- Visibility
- Konversion

Display
- Image Data
- Driver

E.Gröller, H. Doleisch, L. Mroz, S. Grimm
Graphical Data Processing

Modeling

Modeler

3D Data

Transformations

Clipping

Shading

Visibility

Konversion

Driver

3D Data

Image Data

Modeling

Rendering

Display
Graphical Data Processing

Modeling
- Modeler
- 3D Data

Rendering
- Transformations
- Clipping
- Shading
- Visibility
- Konversion

Display
- Image Data
- Driver
Graphical Data Processing

Modeling
- 3D Data
- Modeler

Rendering
- Transformations
- Clipping
- Shading
- Visibility
- Konversion

Display
- Image Data
- Driver
Graphical Programming

- Programs to process and manipulate graphical objects and data
- Several approaches already exists
  - Standards
  - Libraries
Standards

- Early standards: GKS, PHIGS
- X-Windows
  - X-protocol, Xlib, window manager
- OSF/Motif
  - GUI programing for X-Windows
- Microsoft Windows
  - Operating system / GUI interface
Graphic Libraries

- 2D, 3D graphics libraries
- Support for several modern programming languages
- Level of abstraction
  - Low-level APIs (e.g., OpenGL)
  - High-level (scene-graph) APIs (e.g., Java3D)
Graphic Libraries

- OpenGL
  - API for 2D/3D graphics, platform-independent

- Direct3D
  - API for 2D/3D graphics, part of DirectX
Graphic Libraries

- RenderMan
Graphic Libraries

- OpenInventor
  - System to create interactive 3D graphics, based on C++ and OpenGL
- VRML, X3D
  - 3D graphics in web-browser
- OpenGL Performer
  - Based on OpenGL, SGI-hardware
Graphic Libraries

- **Java 3D**
  - Java-based library to create interactive 3D graphics
  - Based on scene graph, structure similar to VRML
  - Supports OpenGL and DirectX
Java 3D

Eduard Gröller, Johanna Schmidt

Institute of Computer Graphics and Algorithms
Vienna University of Technology
Scene Graph

- Hierarchical structure
- Describes structure of the scene
- Easy design and manipulation of complex scenes
Scene Graph in Java 3D

content

viewing

Behavior Node

Shape3D node

Transform Group Nodes

Virtual Universe

Locale

Branch Group Nodes

View Platform

View

Other Objects

Appearance

Geometry
Viewing Model

- Supports multiple Canvas3D -> stereo rendering, CAVE
- Support for tracking
- Detailed description of viewer’s eyes/ears configuration
Java 3D Node Types

- Group nodes
  - BranchGroup (BG): groups child nodes, only node that is detachable
  - TransformGroup (TG): describes transformation
Java 3D Node Types

- **Shape3D**
  
  Defines object in the scene, contains:
  
  - Geometry (polygon-related information)
  
  - Attributes (material-related information, render mode)
Geometry

- Geometry definition
  - Coordinates
  - Normals (optional)
  - RGB(A) color (optional)
  - Texture coordinates (optional)
  - For indexed types: indices
Appearance

- Defines attributes for rendering
  - Polygon attributes: culling, ...
  - Rendering attrib.: z-buff., α-blend., ...
  - Transp./color (if not per vertex)
  - Material (reflection coefficients)
  - Textures
Java 3D Node Types

- Light
  - Defines light source
  - Types:
    - AmbientLight
    - DirectionalLight
    - PointLight
    - SpotLight
Java 3D Node Types

- Behaviour
  - “Events” for the scene graph
  - Examples:
    - Mouse-controlled navigation
    - Picking
    - Collision reaction
Behaviour

- Action executed if criteria met: (-) time triggered, (-) mouse events, (-) picking, (-) collision, (-) frames elapsed
- Boolean combination of criteria
- Bounded area of relevance
- `processStimulus()`
Node States & Capabilities

- **Detached**: node created
- **Live**: nodes inserted into scene
  - *Capabilities* specify permitted operations (r / w / modify)
    - `setCapability(ALLOW_TRANSFORM_WRITE);`
- **Compiled** (optional): nodes optimized by renderer
  - modifications restricted
Java 3D Helper Classes

- `javax.vecmath.*`
- `Tuple/Point/Vector 2/3/4D d/float`
- `Matrix 3/4D d/float`
- `Quaternion classes`
- `javax.media.j3d.Transform3D`
- `orth/persp. Projection`
- `scale, rotate, translate`
Java 3D Helper Classes

- `com.sun.j3d.utils.*`
  - SimpleUniverse (fast setup of viewing branch with default values)
  - Mouse -> Transform mapping
  - Simple geometrical objects
Java 3D Renderer

- Traverse scene graph
- Double buffering, RGB(A)
- Take care of transparency
- Regions of influence of fog, lights, sound
- Optimized for rendering (display list)
Hello3D! / Scenegraph
Hello3D! / Constructor

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

...
Hello3D! / Content

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);
    objTrans.addChild(new ColorCube());
    return objRoot();}
Hello3D!
Further Material

- **CG Reader**
  
  http://www.cg.tuwien.ac.at/courses/CG/VO.html

- **API & Documentation:**
  
  https://java3d.java.net/