OpenSceneGraph

2008년 여름
박경신

Setup OSG Libraries

- Download OSG 2.4.0
  - ftp dis.dankook.ac.kr/OSG
  - login: dis
  - passwd: mm12345
- Unzip at C drive
- Set environment variables
  - Go to Control panel -> System icon -> Advanced tab -> Environment variables
  - Add "C:\OSG-2.4.0\bin;C:\OSG-2.4.0\bin\osgplugins-2.4.0" for System's "Path" variable
  - Create a new variable called "OSG_FILE_PATH" and set the value "C:\OSG-2.4.0\OpenSceneGraph-Data" for the variable
- Open a command window and run "osgviewer cow.osg"

Runing osgviewer

- osgviewer cow.osg

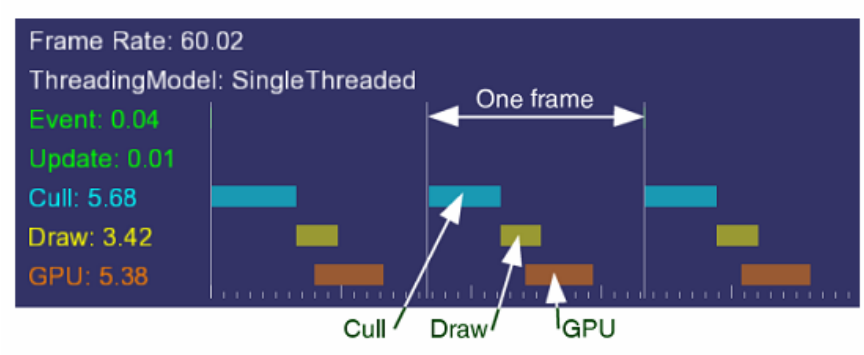
- osgviewer --help
- osgviewer --clear-color 1.0,1.0,1.0,1.0 cow.osg
- osgviewer --image osg256.png

Environment Variables

- Two environment variables often used for OSG applications including **osgviewer**
- File Search Path
  - OSG_FILE_PATH
    - Specifies the search path OSG uses when loading image and model files
    - If a data file is not in the current directory, OSG finds and loads it from the directory path specified in OSG_FILE_PATH
- Debug Message Display
  - OSG_NOTIFY_LEVEL
    - Can show large amount of debugging information to std::cout
    - OSG_NOTIFY_LEVEL controls how much debugging information OSG displays
    - The values can be one of the ALWAYS (least verbose), FATAL, WARN, NOTICE, INFO, DEBUG_INFO, DEBUG_FP (most verbose)

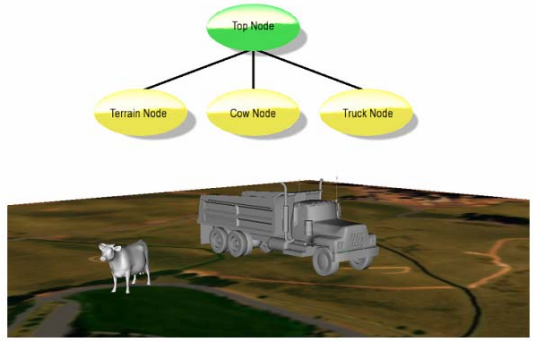
Playing with osgviewer

- Statistics Display – 's' key in osgviewer



Scene Graphs

- A hierarchical tree data structure
- Organizes spatial data for efficient rendering
- Following picture shows an abstract scene graph consisting of terrain, a cow, and a truck

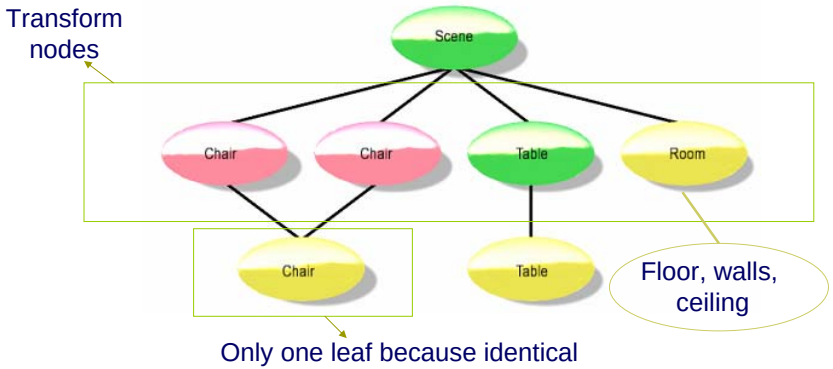


Scene Graphs

- Scene graph tree is head by a top-level root node
- Right under the root node, group nodes organize geometry and the rendering state that controls their appearance
- Root and group nodes can have zero or more children
- Leaf nodes contain the actual geometry

Scene Graphs

- Think of a 3D scene containing a room with a table and two identical chairs
- Following picture shows a way of organizing this (one of the many ways)



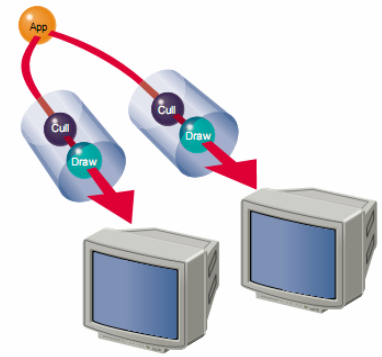
Scene Graph Nodes

- Scene Graphs usually a variety of different node types
  - Switch
    - Enable or disable their children
  - Level of detail (LOD)
    - Select children based on distance from the viewer
  - Transform
    - Modify transformation state of child geometry

Scene Graph Features

- Scene Graphs provide additional features and capabilities
  - Spatial organization
    - Scene graph tree structure lends to intuitive spatial organization
  - Culling
    - View frustum and occlusion culling reduces overall system overloads
  - LOD
    - Efficient rendering at varying levels of detail
  - Translucency
    - Correct and efficient translucent (non-opaque) geometry sorted by depth and rendered in back-to-front order
  - State change minimization
    - Scene graphs commonly sort geometry by state to minimize the state changes
  - File I/O
    - Reading and writing 3D data
  - And more

Rendering Scene Graphs



- Scene graphs often supports multiple stages while rendering
- Update (App) traversal
  - allows the application to modify the scene graph
- Cull traversal
  - tests the bounding volumes of all nodes for inclusion in the scene
- Draw traversal
  - traverses the list of geometry created during the cull traversal and issues low-level graphics API (e.g., OpenGL) to render that geometry

Rendering Scene Graphs

- OSG includes a fourth traversal, Event traversal
  - processes input and other events each frame just before the update traversal
- OSG scene graph traversals

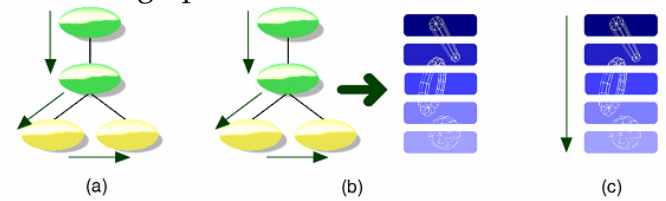


Figure 1-8
Scene graph traversals
Rendering a scene graph typically requires three traversals. In (a), the update traversal modifies geometry, rendering state, or node parameters to ensure the scene graph is up-to-date for the current frame. In (b), the cull traversal checks for visibility, and places geometry and state references in a new structure (called the *render graph* in OSG). In (c), the draw traversal traverses the render graph and issues drawing commands to the graphics hardware.

Rendering Scene Graphs

- These traversals are executed once for each rendered frame
- For stereo rendering and multiple display systems
  - Update traversal is executed once per frame
  - Cull and draw traversals execute once per view per frame

OpenSceneGraph

- Open Source High Performance Scene Graph Toolkit
  - Written in ANSI C++, Standard Template Library (STL), and OpenGL low-level graphics API
  - Supports view frustum culling, occlusion culling, small feature culling
  - Level of Detail (LOD)
  - Vertex arrays, vertex buffer objects
- Supports Multi-platform
  - Windows, Mac OSX, Linux, and others
- Support multiple file formats
  - COLLADA, MAX (.3ds), Performer (.pfb), LightWave (.lwo), Alias Wavefront (.obj), OpenFlight (.flt)
- Node Kits
  - Particle system, high quality anti-aliased text, special effects framework, interactive controls

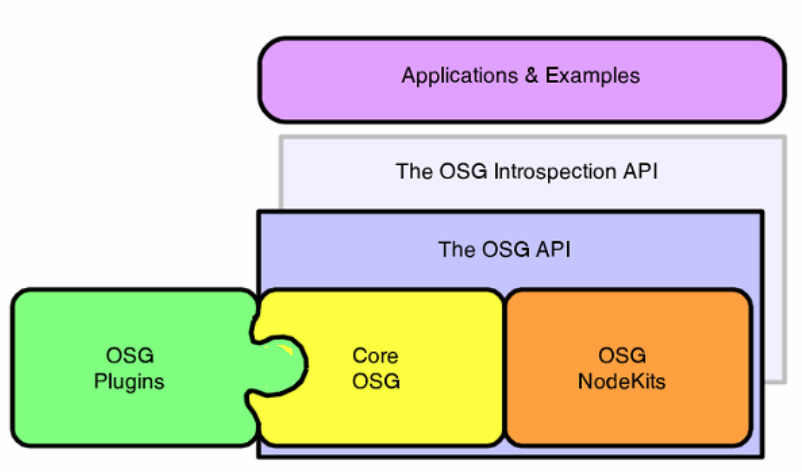
OSG Naming Conventions

- Namespaces
  - osg, osgSim, osgFX
- Classes
  - MatrixTransform, NodeVisitor, Optimizer
- Class methods
  - addDrawable(), getNumChildren(), setAttributeAndModes()
- Templates
  - ref_ptr<>, graph_array<>, observer_ptr<>
- Statics variables and functions
  - s_applicationUsage, s_ArrayNames()
- Globals
  - g_NotifyLevel, g_readerWriter_BMP_Proxy

OSG Components

- Core OSG
  - provides essential scene graph and rendering capability
  - And additional functionality that 3D graphics applications typically require
- OSG NodeKits
  - extend the functionality of core OSG scene graph node classes to provide higher-level node types and special effects
- OSG Plugins
  - reads and writes 2D image and 3D model files
- Interoperability libraries
  - allow OSG to be integrated with other programming languages, such as Python and Lua
- Extensive collection of applications and examples

OSG Components



OSG Libraries

- osg library
  - Contains the scene graph node classes
  - Vector, matrix math, geometry, rendering specification and management
  - Other classes required to build 3D applications, such as argument parsing, animation path management, and error and warning communication
- osgUtil library
  - Contains classes and functions for operating on a scene graph and its contents, gathering statistics and optimizing a scene graph, and creating the render graph
- osgDB library
  - Contains classes and functions for creating and rendering 3D databases

OSG Libraries

- osgViewer library
  - Contains classes that manage views into the scene
  - Integrates OSG with a wide variety of windowing systems

OpenSceneGraph Classes

- osg Library
  - Namespace: osg
  - Header files <OSG_DIR>/include/osg
  - Windows library files: osg.dll and osg.lib

  - Node – the base class for all nodes in the scene graph, contains methods to facilitate scene graph traversals, culling, application callbacks, state management
  - Group – the base class for any node that can have children
  - Geode (Geometry Node) – corresponds to the leaf node in OSG; has no children, but contains osg::Drawable objects that contain geometry for rendering
  - LOD – the LOD class displays its children based on their distance to the view point

OpenSceneGraph Classes

- MatrixTransform – the class contains a matrix that transforms the geometry of its children
- Switch – contains a Boolean mask to enable or disable processing of its children

- Geometry classes
  - Drawable – the base class for storing geometric data
  - Geometry – act as high-level wrappers around the OpenGL vertex array functionality
  - Primitive Set – the class provides high-level support for the OpenGL vertex array drawing commands
  - Vector classes (Vec2, Vec3, etc.)
  - Array classes (Vec2Array, Vec3Array, etc)

OpenSceneGraph Classes

- State Management Classes
  - StateSet – OSG stores a collection of state values (called modes and attributes) in the StateSet class; any osg::Node in the scene graph can have StateSet associated with it
  - Modes – analogous to the OpenGL calls glEnable() and glDisable(); modes allow us to turn on and off features in the OpenGL fixed-function rendering pipeline, such as lighting, blending, and fog; use osg::StateSet::setMode()
  - Attributes – store state parameters; use osg::StateSet::setAttribute()
  - Texture attributes and modes – use osg::StateSet::setTextureMode() and osg::StateSet::setTextureAttribute()

- And many more

OpenSceneGraph Libraries

- osgUtil Library
  - Namespace: osgUtil
  - Header files <OSG_DIR>/include/osgUtil
  - Windows library files: osgUtil.dll and osgUtil.lib

  - Intersection
    - Intersector, IntersectionVisitor, LineSegmentIntersector, PolytopeIntersector, PlaneIntersector
  - Optimization
    - Optimizer, Statistics and StatesVisitor
  - Geometry Manipulation
    - Simplifier, Tessellator, DelaunaryTrianglator, TriStripVisitor, SmoothingVisitor, Texture map generation

OpenSceneGraph Libraries

- osgDB Library
  - Namespace: osgDB
  - Header files <OSG_DIR>/include/osgDB
  - Windows library files: osgDB.dll and osgDB.lib

- osgViewer Library
  - Namespace: osgViewer
  - Header files <OSG_DIR>/include/osgViewer
  - Windows library files: osgViewer.dll and osgViewer.lib

Compiling OSG Applications

- Use Visual Studio 2005 (install service pack 1)
- Add the following path to the Additional Include Directories
  - C:\OSG-2.4.0\include

  Similarly, set the linker directory to the following
  - C:\OSG-2.4.0\lib
- And add the libraries to the input option of the linker setting
  - Release Mode: osgViewer.lib osgDB.lib osgUtil.lib osg.lib OpenThreads.lib opengl32.lib glu32.lib
  - Debug Mode: osgViewerd.lib osgDBd.lib osgUtild.lib osgd.lib OpenThreadsd.lib opengl32.lib glu32.lib