CSE 167:
Introduction to Computer Graphics
Lecture 10: Scene Graph

Jürgen P. Schulze, Ph.D.
University of California, San Diego
Fall Quarter 2013

Midterm

- Midterm has been graded
  - A score of 90 in the exam will count as a grade of 100
- Please return midterm after review if you want to discuss with me later
  - Otherwise feel free to keep it

# Submissions	112
Average score	60.0
Median score	61.0
Highest score	89.5
Lowest score	9.5

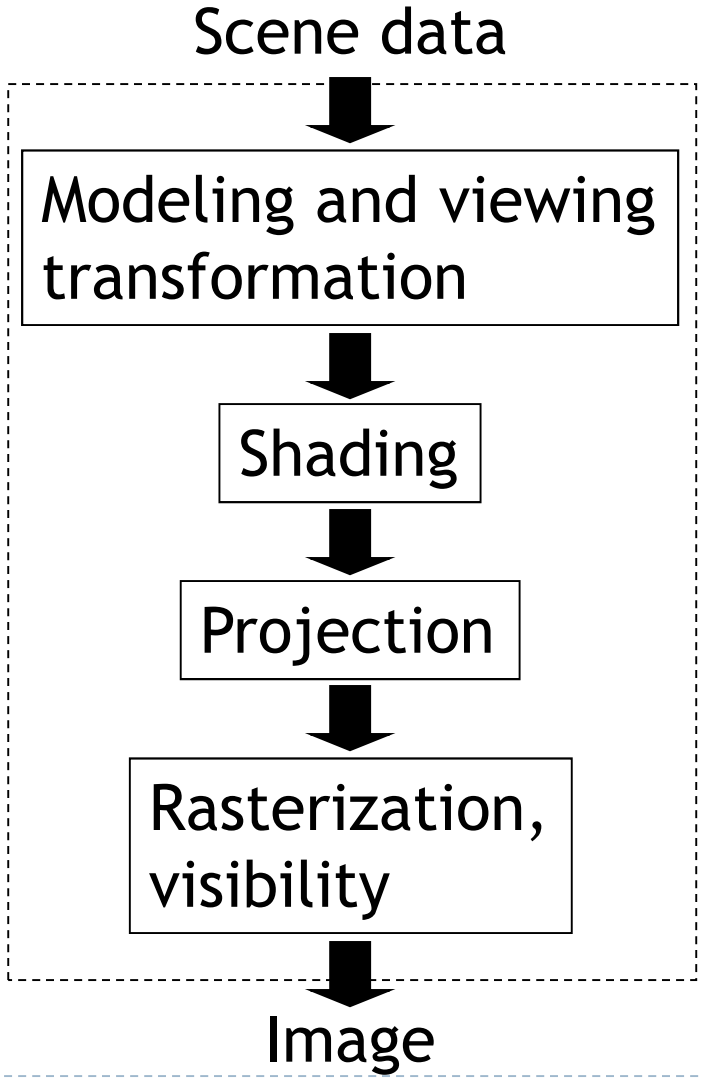
Announcements

- Homework #4:
  Glee web site has been down:
  Matteo put files on Dropbox link: see course forums
- Homework #5 discussion on Monday, Nov 4

Lecture Overview

- Scene Graphs & Hierarchies
  - Introduction
  - Data structures
- Performance Optimization
  - Level-of-detail techniques
  - View Frustum Culling

Rendering Pipeline



Graphics System Architecture

Interactive Applications

- Games, scientific visualization, virtual reality

Rendering Engine, Scene Graph API

- Implement functionality commonly required in applications
- Back-ends for different low-level APIs
- No broadly accepted standards
- Examples: OpenSceneGraph, NVSG, Java3D, Ogre

Low-level graphics API

- Interface to graphics hardware
- Highly standardized: OpenGL, Direct3D

Scene Graph APIs

- APIs focus on different types of applications
- OpenSceneGraph (www.openscenegraph.org)
  - Scientific visualization, virtual reality, GIS (geographic information systems)
- NVIDIA SceniX (https://developer.nvidia.com/scenix)
  - Optimized for shader support
  - Support for interactive ray tracing
- Java3D (http://java3d.java.net)
  - Simple, easy to use, web-based applications
- Ogre3D (http://www.ogre3d.org/)
  - Games, high-performance rendering

Commonly Offered Functionality

- Resource management
  - Content I/O (geometry, textures, materials, animation sequences)
  - Memory management
- High-level scene representation
  - Graph data structure
- Rendering
  - Optimized for efficiency (e.g., minimize OpenGL state changes)

Lecture Overview

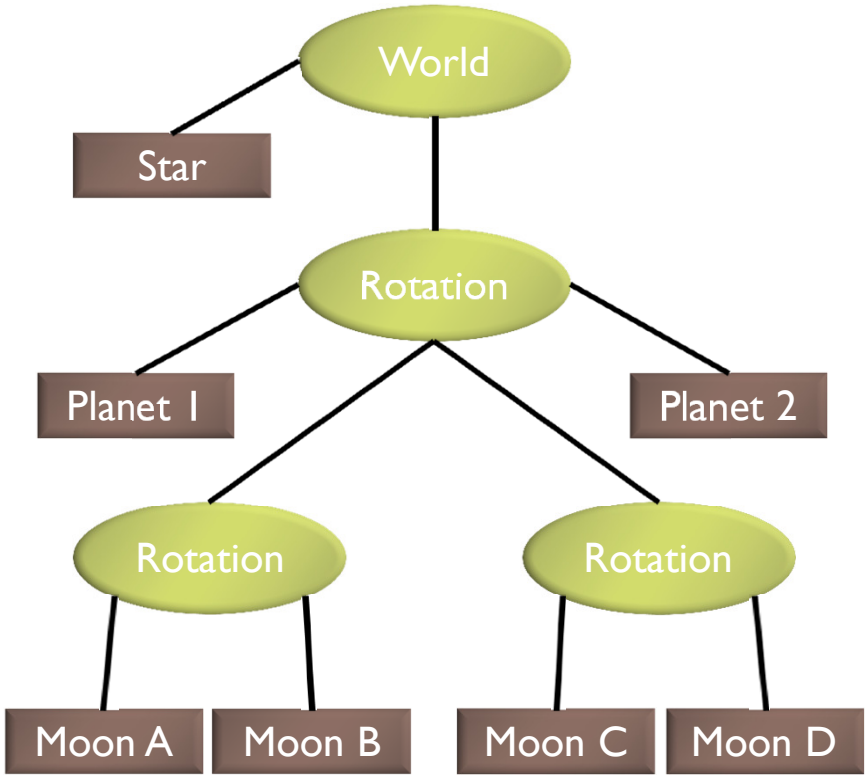
- Scene Graphs & Hierarchies
  - Introduction
  - Data structures
- Performance Optimization
  - Level-of-detail techniques
  - View Frustum Culling

Scene Graphs

- Data structure for intuitive construction of 3D scenes
- So far, our GLUT-based projects store a linear list of objects
- This approach does not scale to large numbers of objects in complex, dynamic scenes

  → Homework Assignment #1 – Animated Objects

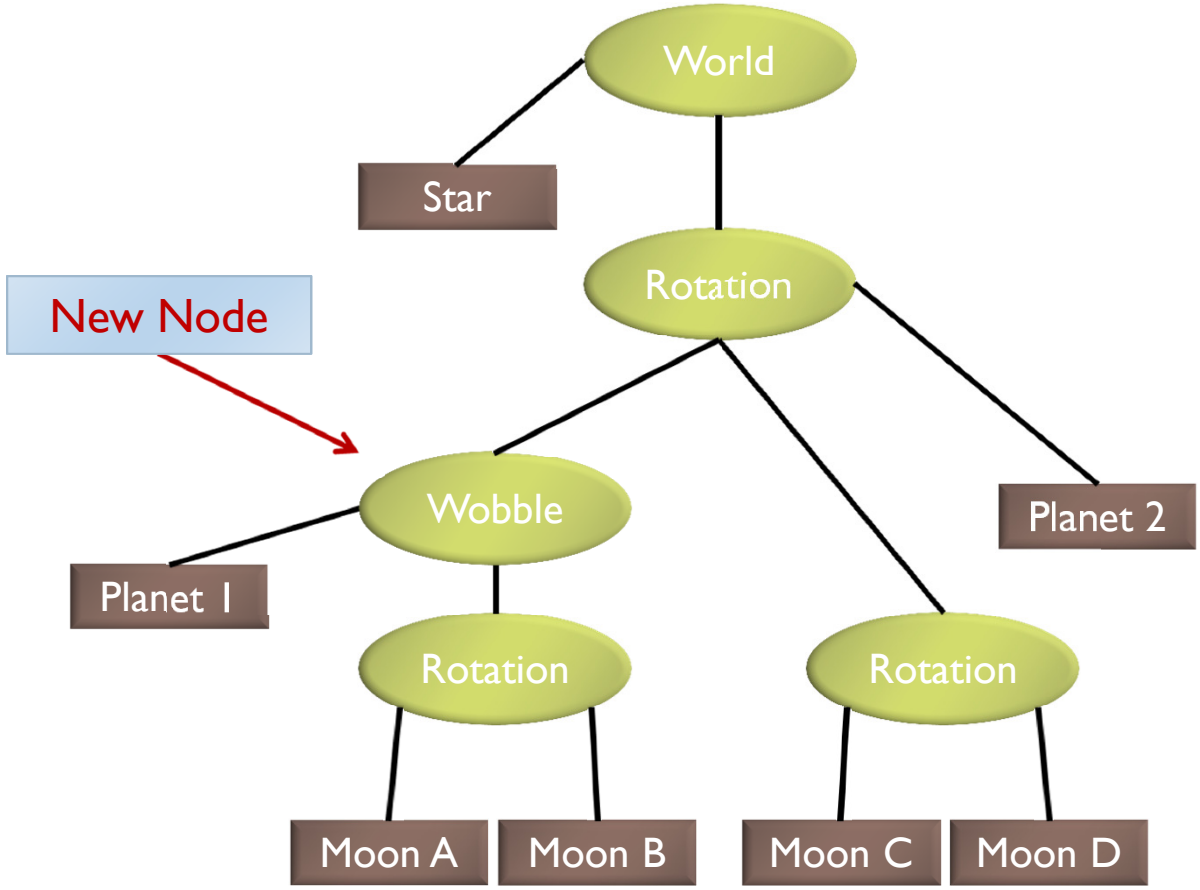
Solar System



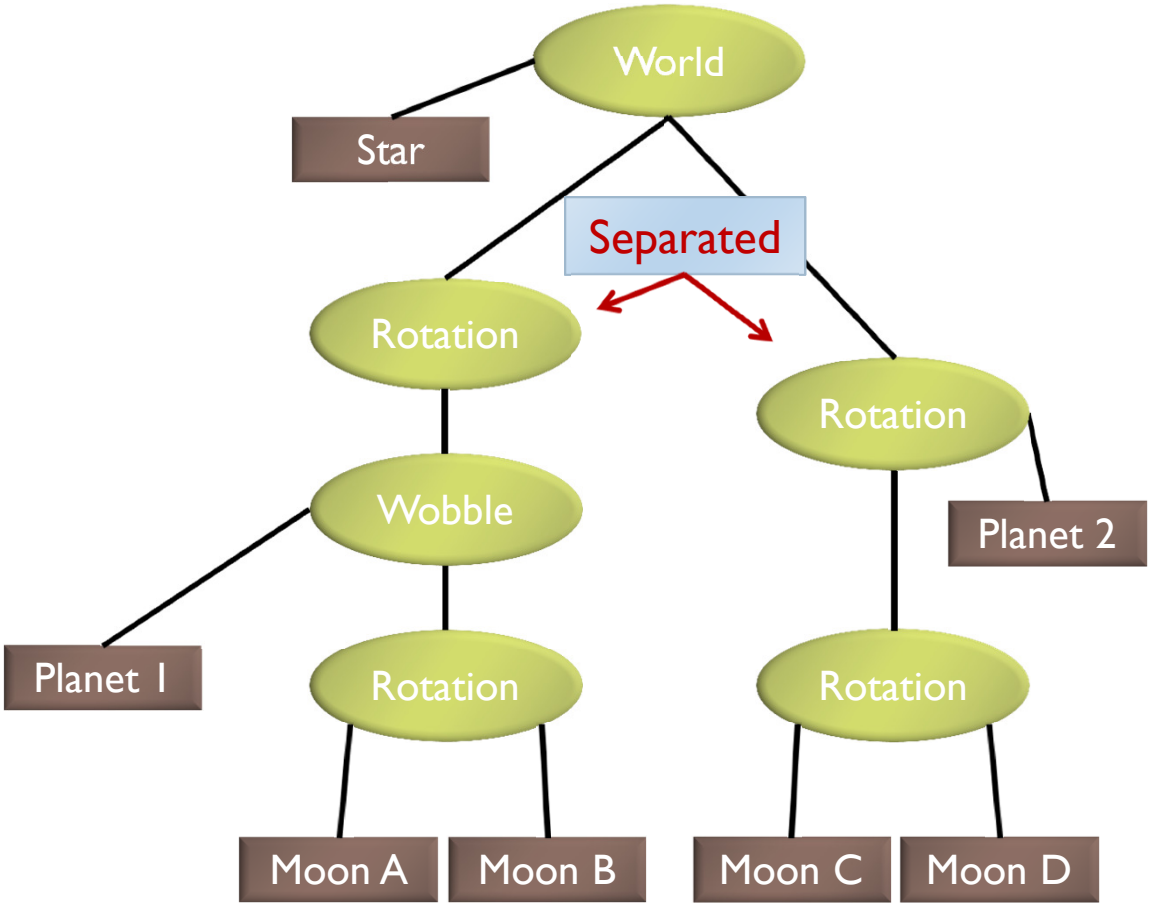
- Draw the star
- Save the current matrix
-
  - Apply a rotation
  - Draw Planet One
  - Save the current matrix
  -
    - Apply a second rotation
    - Draw Moon A
    - Draw Moon B
  - Reset the matrix we saved
  - Draw Planet two
  - Save the current matrix
  -
    - Apply a rotation
    - Draw Moon C
    - Draw Moon D
  - Reset the matrix we saved
- Reset the matrix we saved

Example from http://www.gamedev.net

Solar System with Wobble



Planets rotating at different speeds



- Draw the Star
- Save the current matrix
- - Apply a rotation
  - - Save the current matrix
    - - Apply a wobble
      - Draw Planet 1
      - - Save the current matrix
        - - Apply a rotation
          - - Draw Moon A
            - Draw Moon B
        - Reset the Matrix
    - Reset the matrix
  - Reset the matrix
- Reset the matrix
- Save the current matrix
- - Apply a rotation
  - - Draw Planet 2
    - Save the current matrix
    - - Apply a rotation
      - Draw Moon C
      - Draw Moon D
    - Reset the current matrix
  - Reset the current matrix
- Reset the current matrix

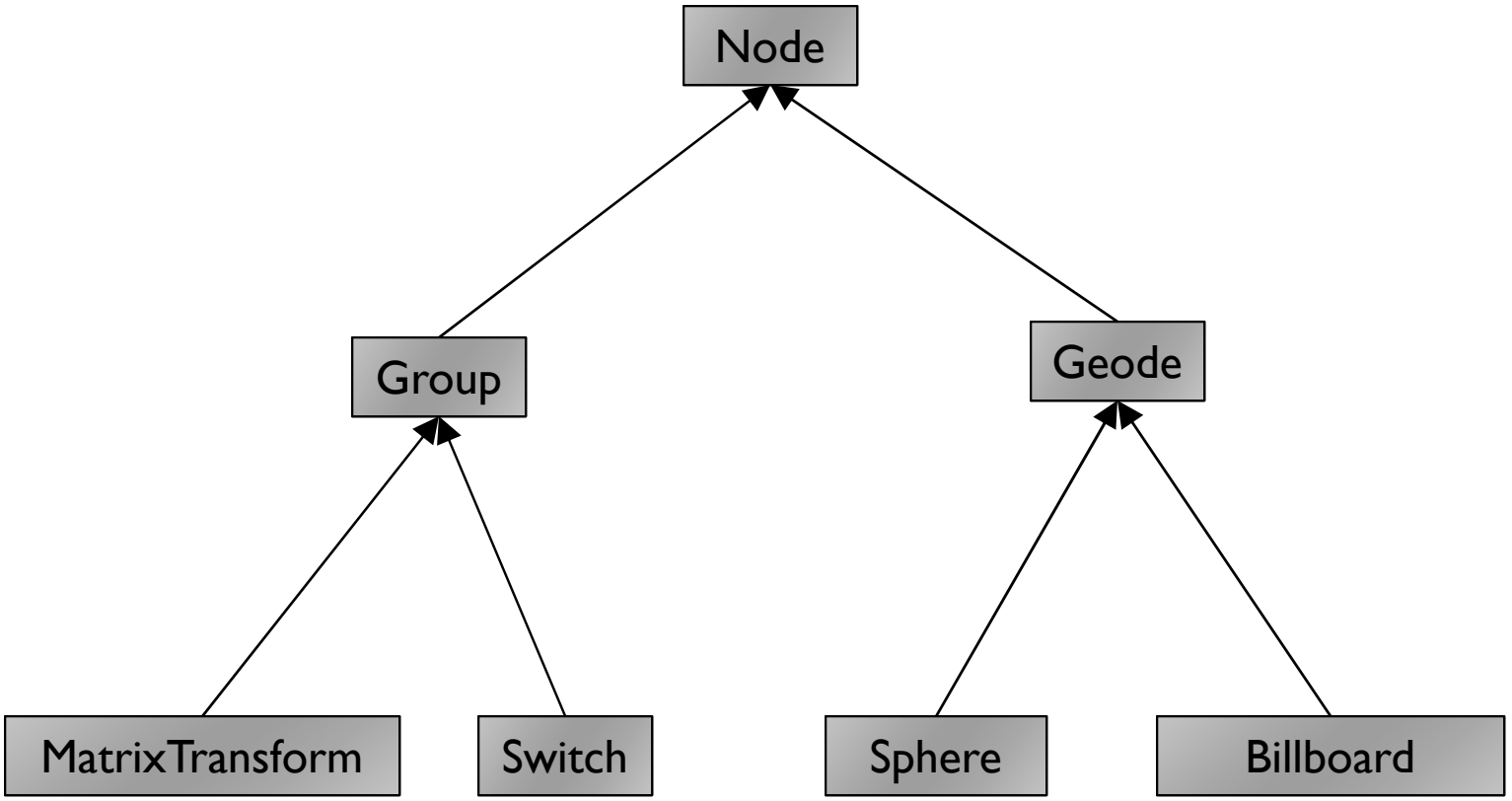
Data Structure

- Requirements
  - Collection of separable geometry models
  - Organized in groups
  - Related via hierarchical transformations
- Use a tree structure
- Nodes have associated local coordinates
- Different types of nodes
  - Geometry
  - Transformations
  - Lights
  - Many more

Class Hierarchy

- Many designs possible
- Design driven by intended application
  - Games
    - Optimized for speed
  - Large-scale visualization
    - Optimized for memory requirements
  - Modeling system
    - Optimized for editing flexibility

Sample Class Hierarchy



Inspired by OpenSceneGraph

Class Hierarchy

`Node`

- Common base class for all node types
- Stores node name, pointer to parent, bounding box

`Group`

- Stores list of children

`Geode`

- Geometry Node
- Knows how to render a specific piece of geometry

Class Hierarchy

`MatrixTransform`

- Derived from Group
- Stores additional transformation **M**
- Transformation applies to sub-tree below node
- Monitor-to-world transformation $\mathbf{M_0M_1}$