Java 3D's Scene Graphs

Basic elements of Graph Theory

Examples of where graphs are used in graphics

And then, Java 3D's Scene Graphs

The basics of Graph Theory

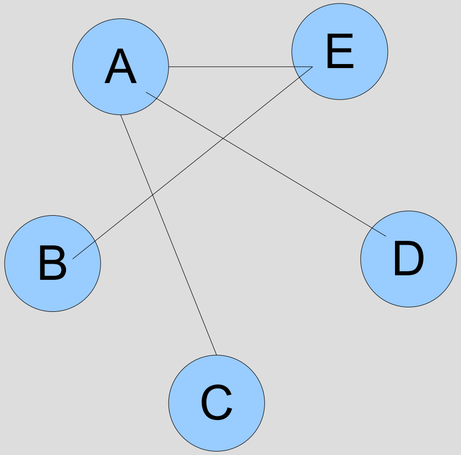
- Graph Theory:
  - What it is...
  - Some of its applications...
  - Why we care about graphs in a course on computer graphics
- Some computer graphics applications of graph theory
- Java 3D's Scene Graphs

Graph Theory

- Graph Theory:
  - Mathematical field which uses a structure called a graph to study the interrelationship of entities
- Graph:
  - A mathematical structure consisting of:
    - Nodes (also known as vertices and sometimes points)
    - Edges (sometimes less commonly called arcs or lines)
  - A Graph is formally defined with the notation:
    - G = (V, E)
  - Where, V is a set of nodes (or vertices), e.g.,
    - V = { a, b, c, d, e }
  - And E is a set of edges where an edge is a pair of vertices
    - E = { (a,c), (b,e) (a,d), (e,a) }

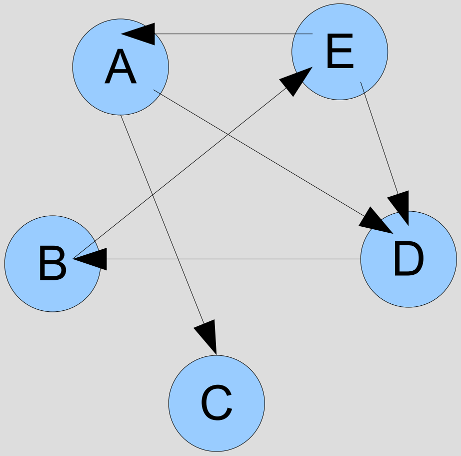
Examples of Graphs

- Graphs can be visualized with circles or some other shape for nodes or vertices and lines for edges
- E.g.,
  - V = { a, b, c, d, e }
  - E = { (a,c), (b,e) (a,d), (e,a) }



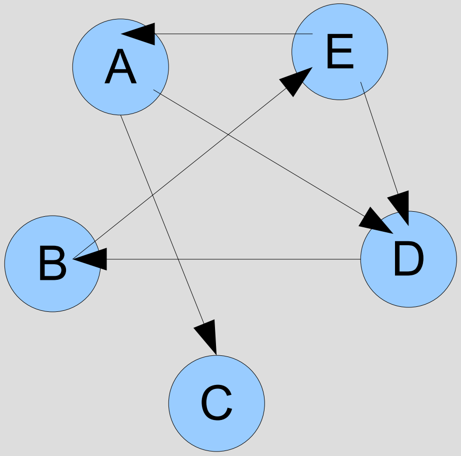
Directed Graphs

- A directed graph (or digraph) is a graph but with directions on the edges
- E.g.,
  - V = { a, b, c, d, e }
  - E = { (a,c), (b,e), (a,d), (e,a) , (e,d) , (d,b)}



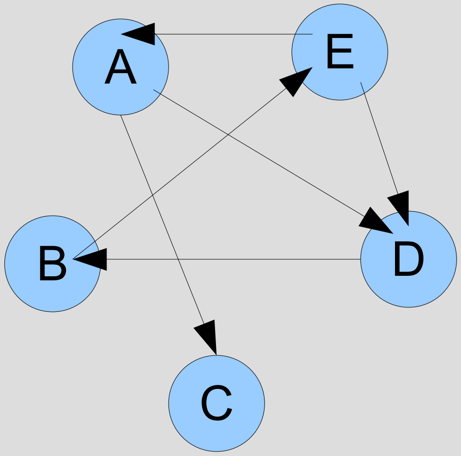
Paths

- A Path in a graph is a sequence of nodes that follow the edges of the graph
- Example path:
  - A path from A to B
    - A,D,B



Cycles

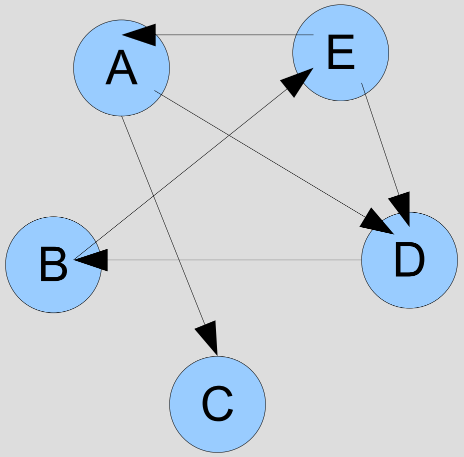
- A cycle in a graph is a path that includes the same node more than once
- Example of a cycle
  - B,E,D,B
- Another example
  - A,D,B,E,A



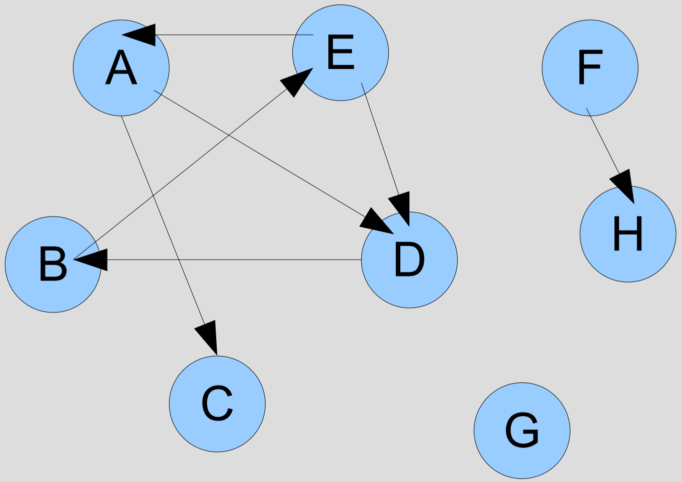
A Connected Graph

- Graphs can be connected or disconnected
- A graph is connected if for every pair of nodes in the graph there exists a path if you ignore directions on the edges

Connected

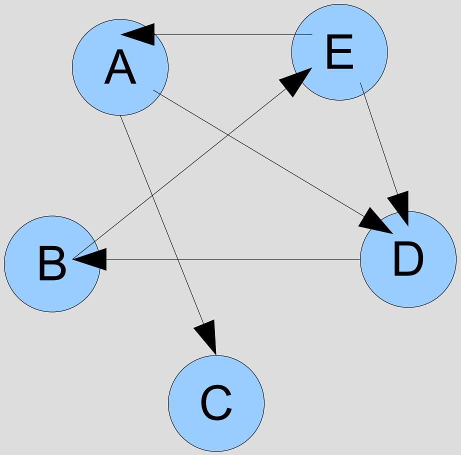


Not Connected



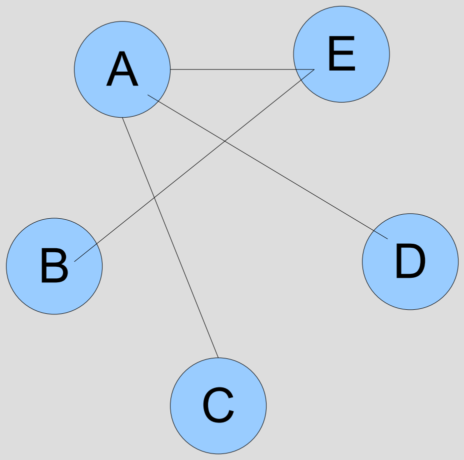
Degree of a Node

- The degree of a node is the number of edges connected to it
- The in-degree of a node in a digraph is the number of edges that end at that node
- The out-degree of a node in a digraph is the number of edges that begin at that node
  - Degree(E) = 3
  - InDegree(E) = 1
  - OutDegree(E) = 2



An Acyclic Graph

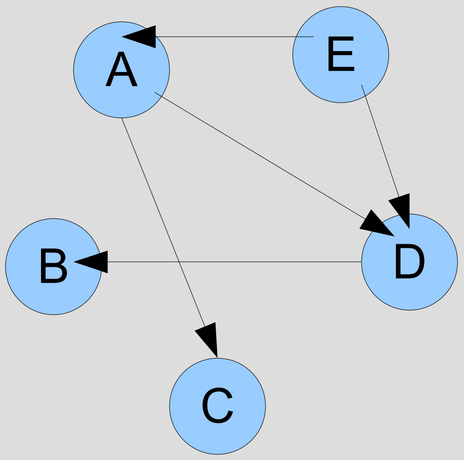
- A graph without cycles is called an acyclic graph



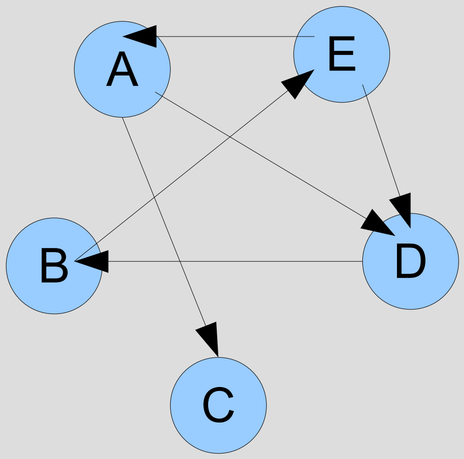
A DAG

- A DAG (directed acyclic graph) is a digraph that contains no cycles

A DAG

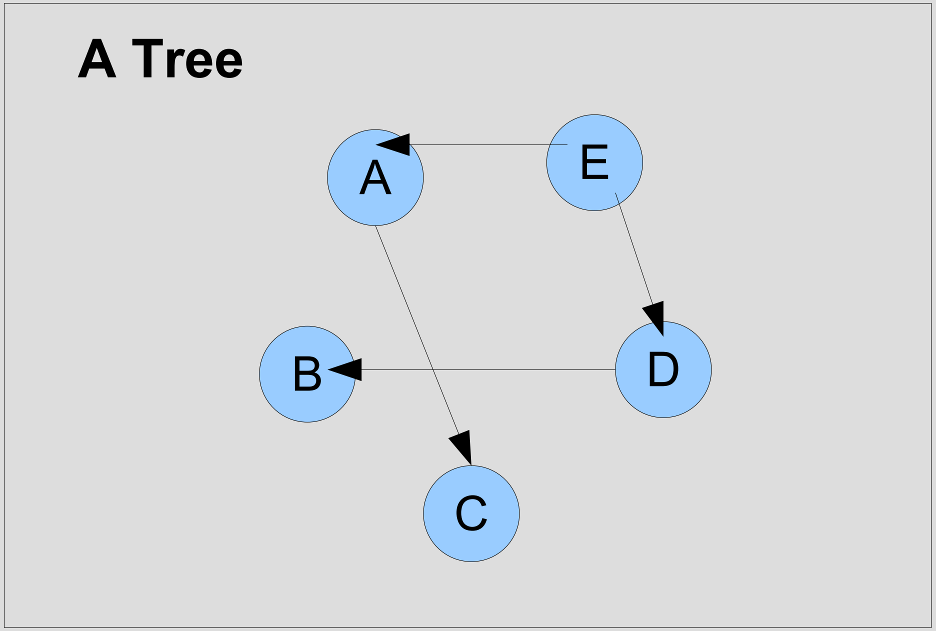


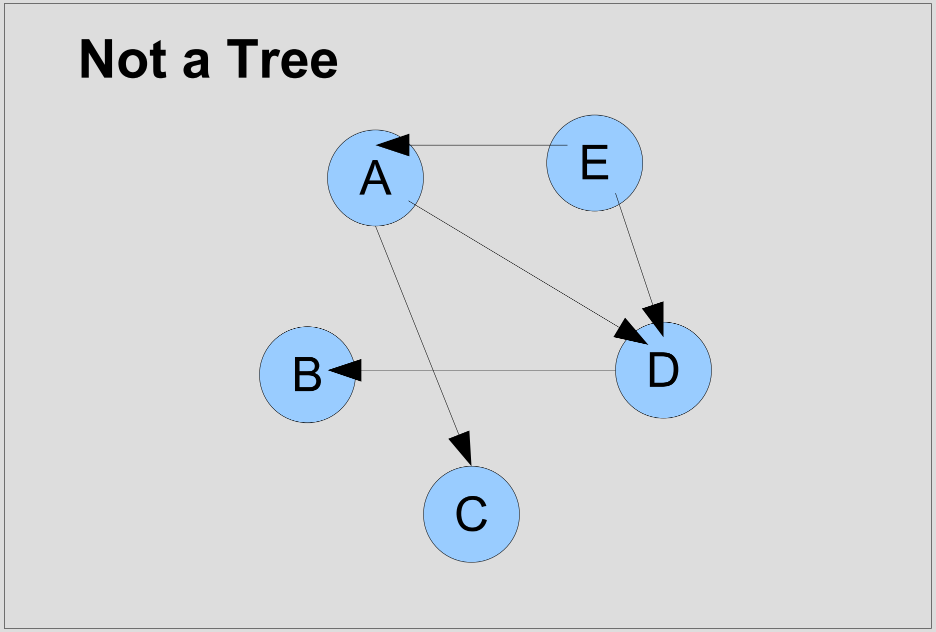
Not a DAG



A Tree

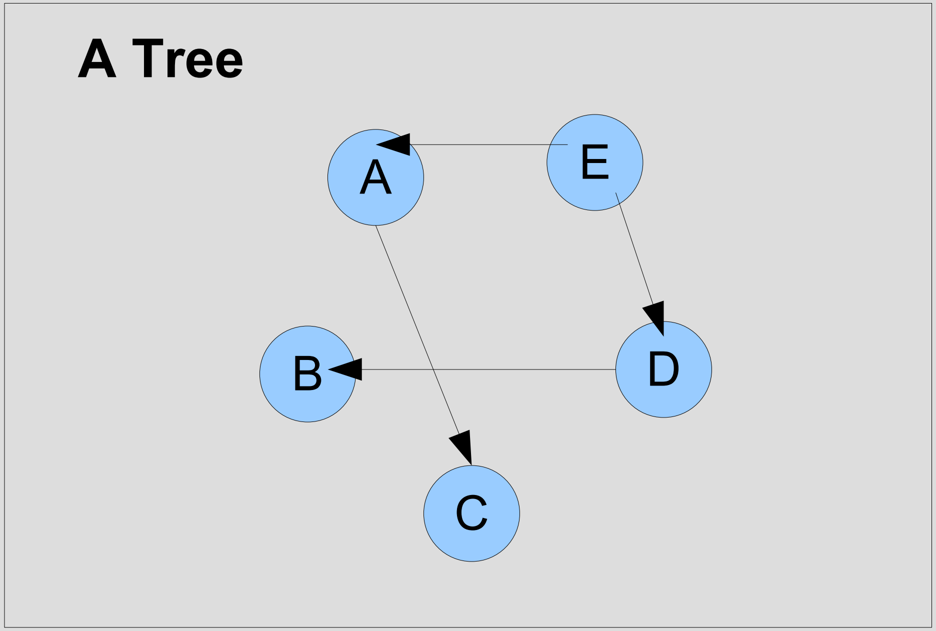
- A Tree is:
  - a DAG...
  - that is connected...
  - And for which no node has an in-degree greater than 1





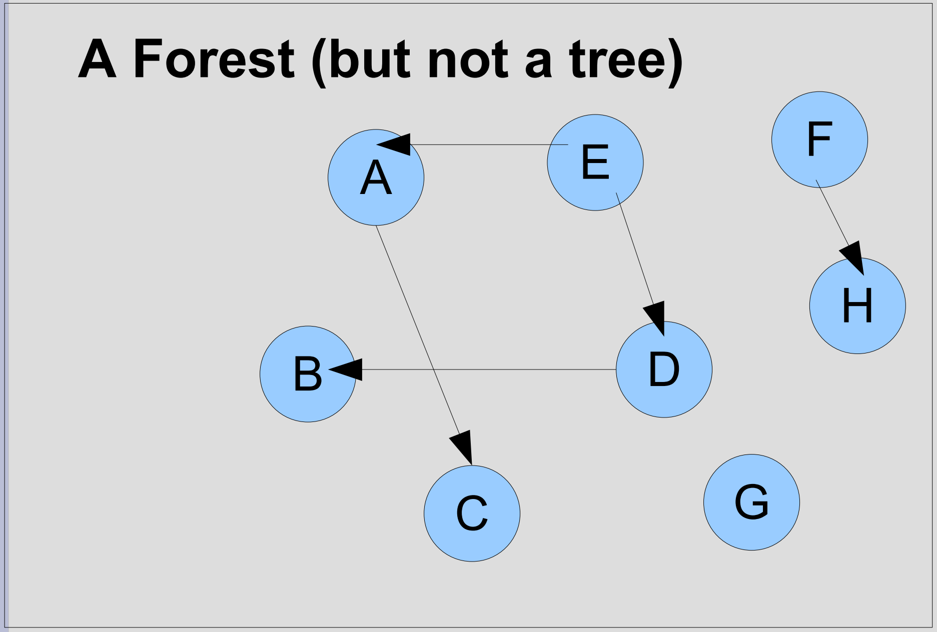
Leafs and Interior Nodes

- A leaf in a tree is a node whose outdegree is 0
- An Interior node is a node in a tree whose outdegree is at least 1
  - E.g., B and C are leafs
  - A,D,and E are interior

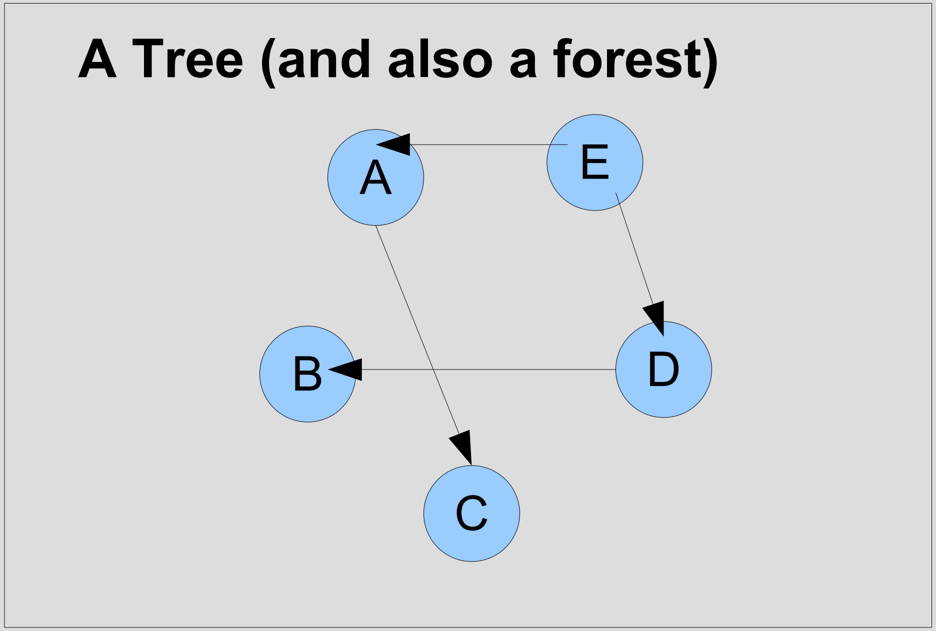


A Forest

- A Forest is a:
  - DAG...
  - for which no node has an in-degree greater than 1...
  - But is not necessarily connected
- Note any graph that is a tree is also a forest



A Forest (but not a tree)



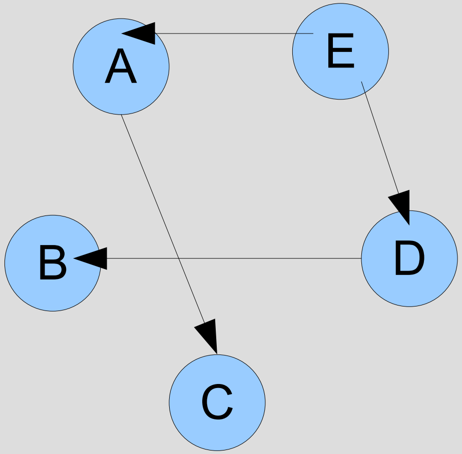
A Tree (and also a forest)

Visualizing a Tree

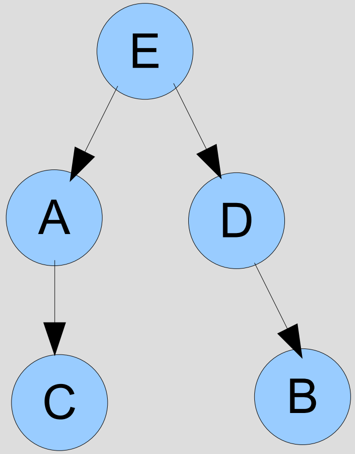
- The root of a tree is the one (and only one node) of the tree whose indegree is 0.
- The children of a node in a tree are the nodes at the ends of the outgoing edges of the node
- Trees are usually visualized with the root at the top, its children lined up below, and so forth

Visualizing a Tree

This Tree



Would be visualized like:

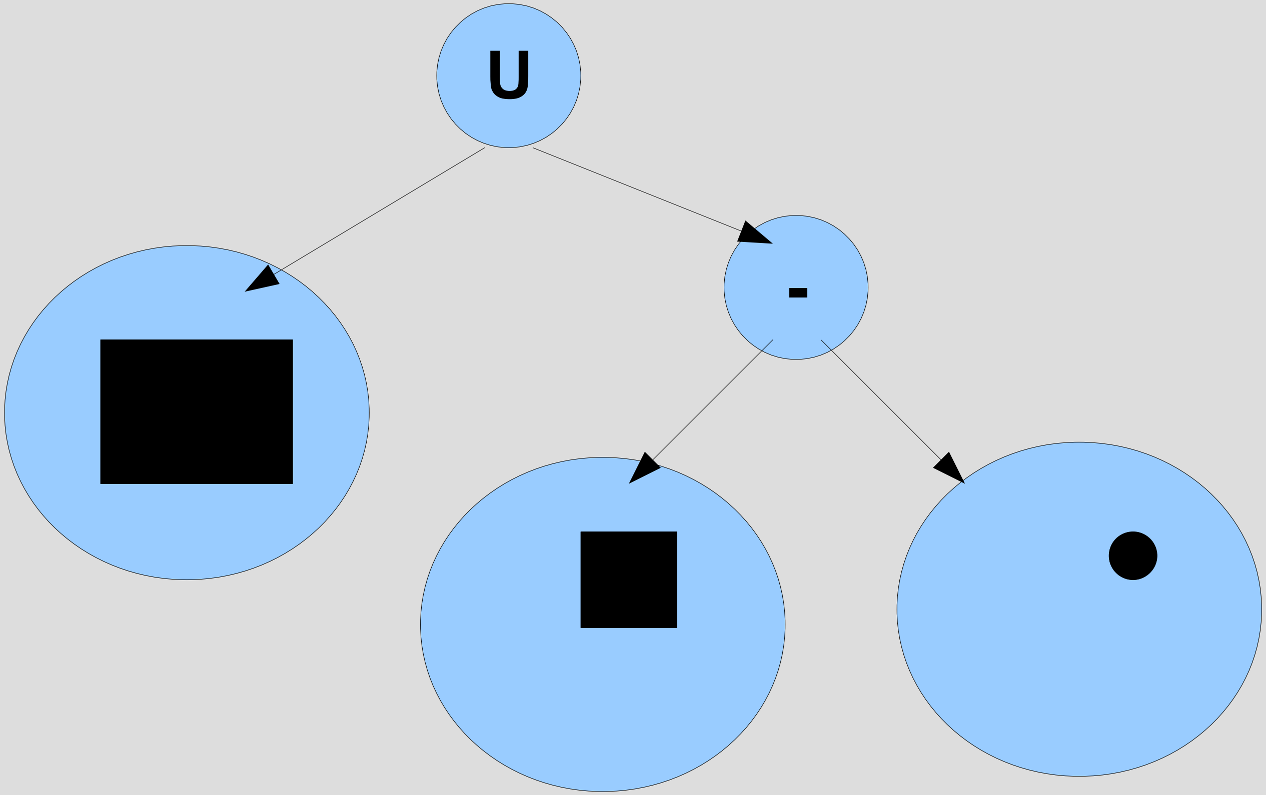


What are Graphs Used For

- Representing a network
  - Nodes: computers, routers, etc
  - Edges: direct connections
- Cluster Computing
  - Representing the interconnectedness of the nodes (computers) of the cluster
- Social Network Analysis
  - Nodes: people or other entities
  - Edges: represent various types of relationships
- Software Engineering
  - e.g., UML diagrams
- Chemistry:
  - Representing molecular structures

Graphs in Computer Graphics

- Constructive Area Geometry: some implementations use graphs to represent the operations and primitive shapes, etc



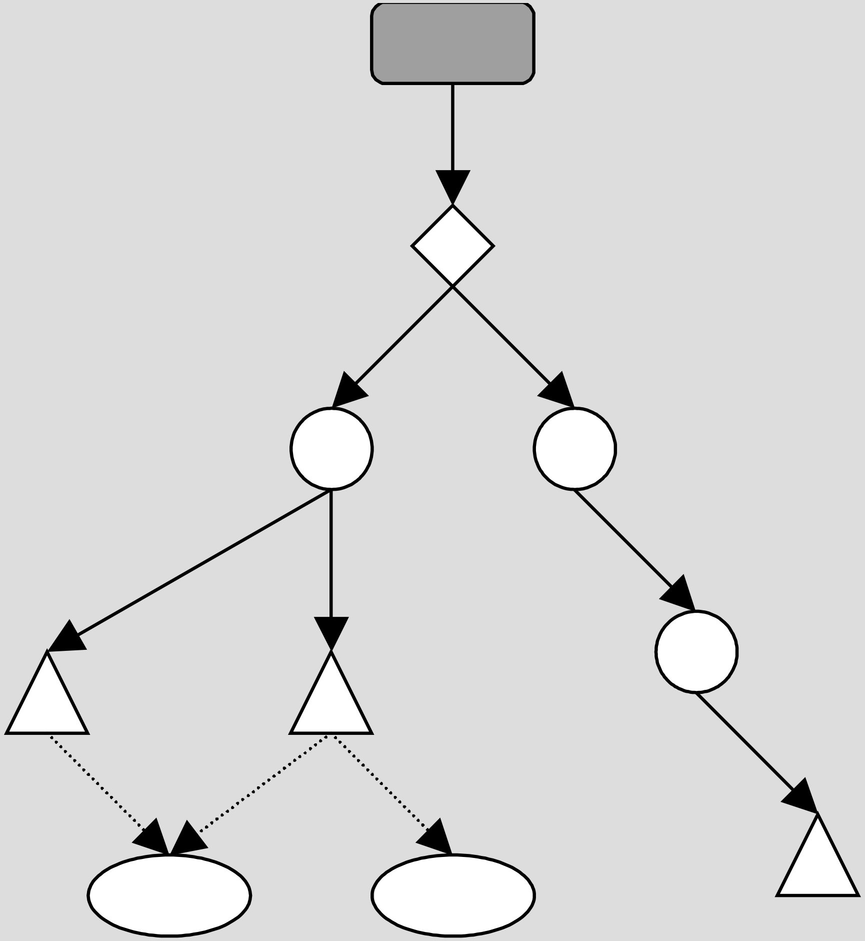
Graphs in Solid Modeling

- Constructive Solid Geometry:
  - The 3D equivalent of constructive area geometry
  - Almost always implemented as a graph structure

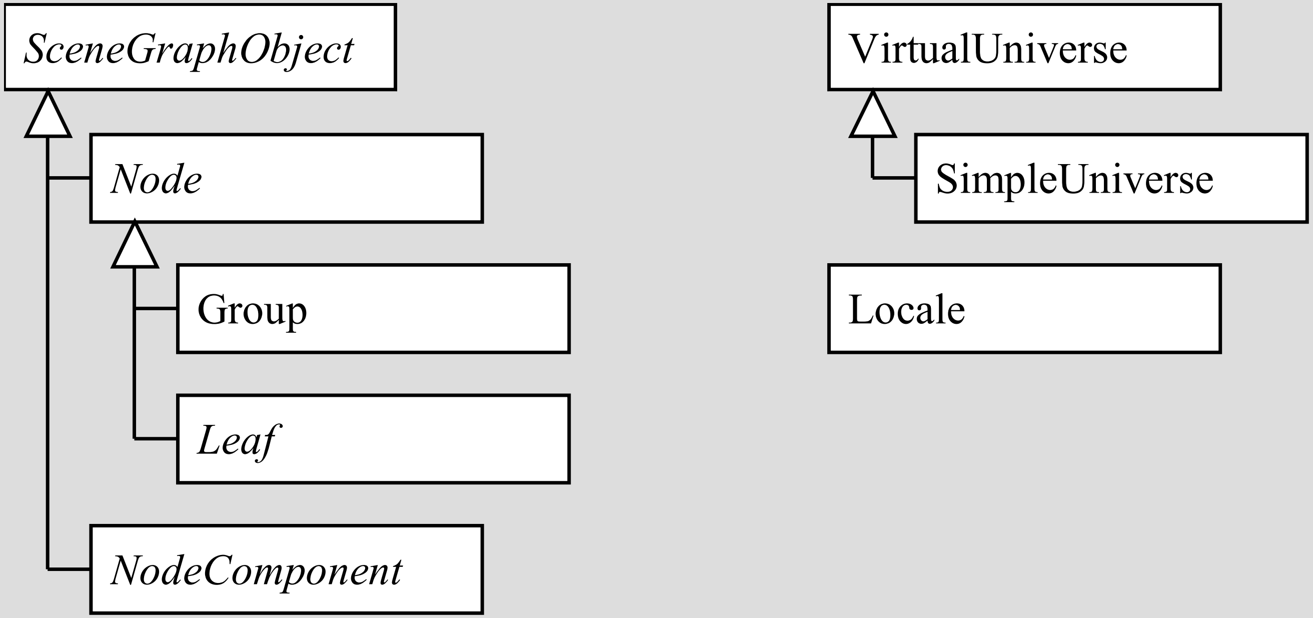
Scene Graph: Legend

▬	Virtual Universe
◇	Locale
○	Group Node
△	Leaf Node
⬭	Node Component
▭	Other Object
→	Parent-Child Link
⇢	Reference

Scene Graph

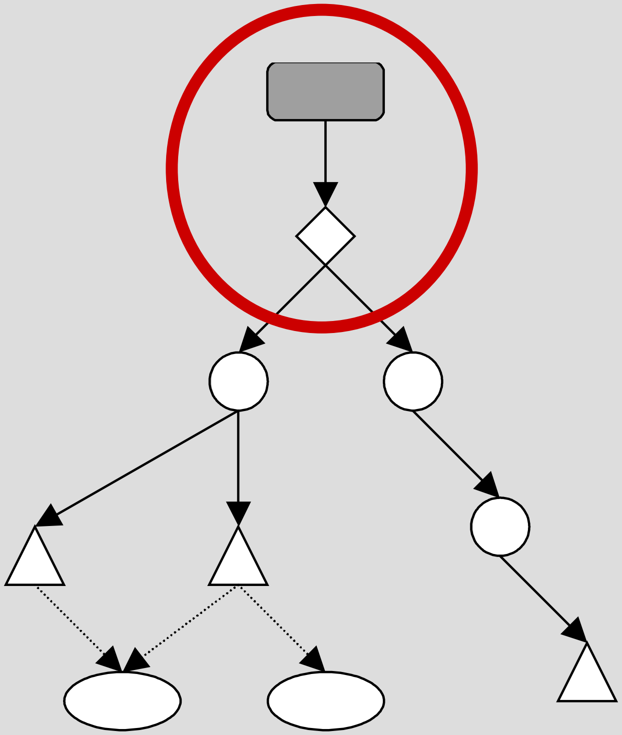


Scene Graph: Classes



Superstructure

- VirtualUniverse – the universe
- HiResCoord – 256bit fixed point numbers
- Locale – a “smaller” space



Superstructure

VirtualUniverse – the universe

HiResCoord – 256bit fixed point numbers

- Why 256bit fixed point?
- Fixed point in the middle
  - 128 bits for integer part
  - 128 bits for the fractional part
- Allows for distances of up to 2^{127} meters if units are in meters
- Resolution can be as fine as 2^{-128} meters
- Can model anything in the universe
  - E.g., distance from Earth to Sun is about 2^{37} meters
  - E.g., radius of a proton is about 2^{-50} meters

Superstructure

- It would be inefficient to use 256bit numbers to represent all coordinates
- The Locale class is used to represent a local space within the Universe
- A Virtual Universe can contain one or more Locale objects
- A Locale object
  - Has a specific location in the VirtualUniverse with a HiResCoord
  - Uses normal floating point numbers within the Locale

Superstructure

- A Java 3D Program has one VirtualUniverse
  - VirtualUniverse universe = new VirtualUniverse();
- A Locale object is attached to the VirtualUniverse
  - A VirtualUniverse can potentially have multiple Locale objects
- Multiple ways of constructing Locales
  - Locale locale = new Locale(universe);
    - Default position in the universe of (0,0,0)
  - Or you can specify where in the universe with the constructors
    - Locale(VirtualUniverse vu, HiResCoord location)
    - Locale(VirtualUniverse vu, int[] x,int[] y,int[] z)

The Locale Object

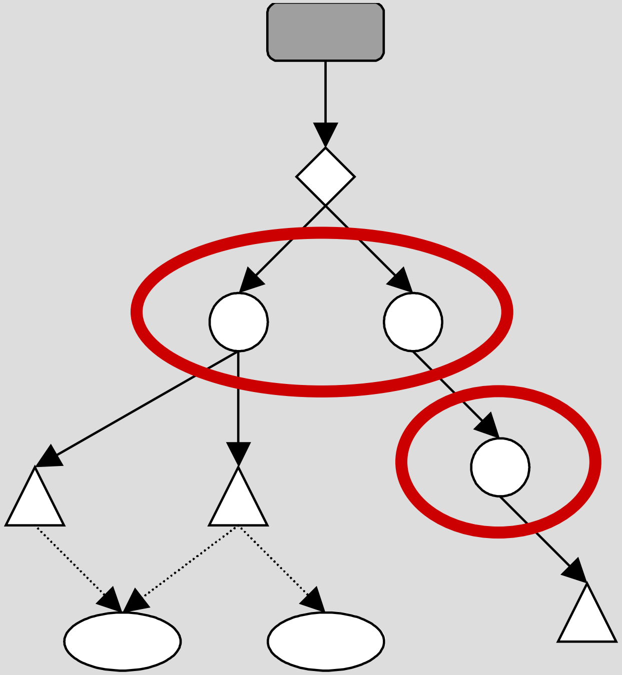
- A Locale object can have one or more BranchGroup objects attached
- A BranchGroup is a Scene Graph
- Once a BranchGroup is attached to a Locale, Java 3D begins rendering
- Can add a Scene Graph (BranchGroup object) with:
  - void addBranchGraph(BranchGroup branch)

SimpleUniverse

- SimpleUniverse is a highly useful utility class that extends VirtualUniverse
- SimpleUniverse includes
  - A Locale object
  - Set of objects to define a standard view
- Can quickly form a complete Scene Graph by adding a content branch
- Default view position (from viewers perspective)
  - The x-axis points to the right
  - The y-axis points up
  - The viewer is on the z axis looking towards -z

Group Nodes

- BranchGroup
- OrderedGroup
- Primitive
- SharedGroup
- Switch
- TransformGroup



BranchGroup

- The only type of Node that can be added to a Locale
- No special operations beyond this
- Children can either be leafs or other Group nodes
- Add children with
  - void addChild(Node child)
  - void insertChild(Node child, int index)
- Can also get the number of children, etc
  - int numChildren()
  - Enumeration getAllChildren()

OrderedGroup

- Java3D does not specify order that children of a node are visited during rendering
  - Children of a node can be rendered in any order
- OrderedGroup
  - Allows programmer to specify order that children are rendered
  - Rendered in order of indices
  -

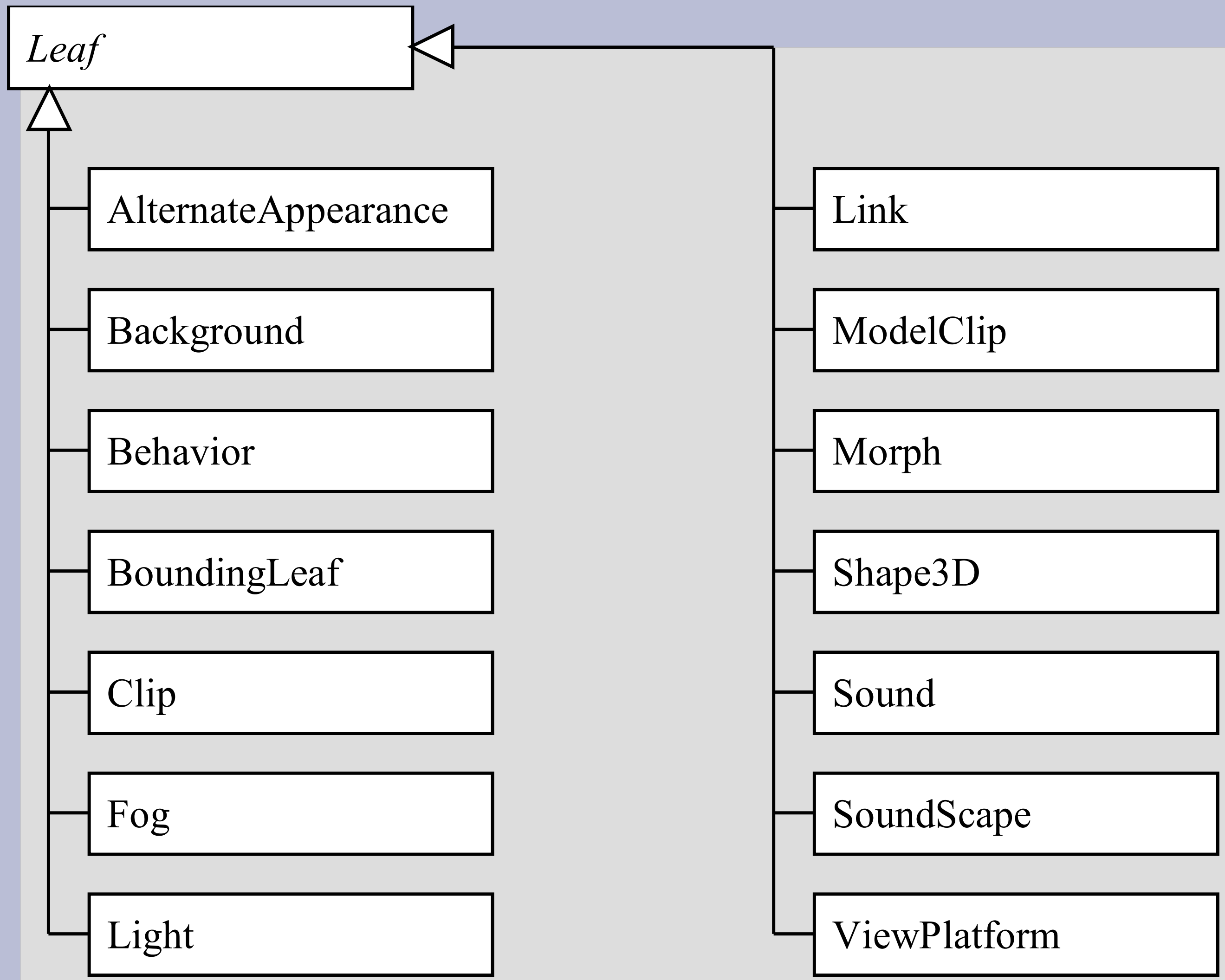
Some other group nodes

- Primitive node
  - A geometric primitive such as a sphere
  - We'll look at these later
- A SharedGroup
  - Branch graphs must be trees
  - If 2 branches are identical:
    - We can use a special type of node called a link which is a leaf
    - The Link nodes can each have a reference to a single SharedGroup
    - We'll see this in more detail later

Some other group nodes

- Switch
  - Used to select a particular set of the children for rendering
  - We'll see these later on
- TransformGroup
  - A geometric transform that is applied to all of its children

Leaf Nodes



Leaf

AlternateAppearance

Background

Behavior

BoundingLeaf

Clip

Fog

Light

Link

ModelClip

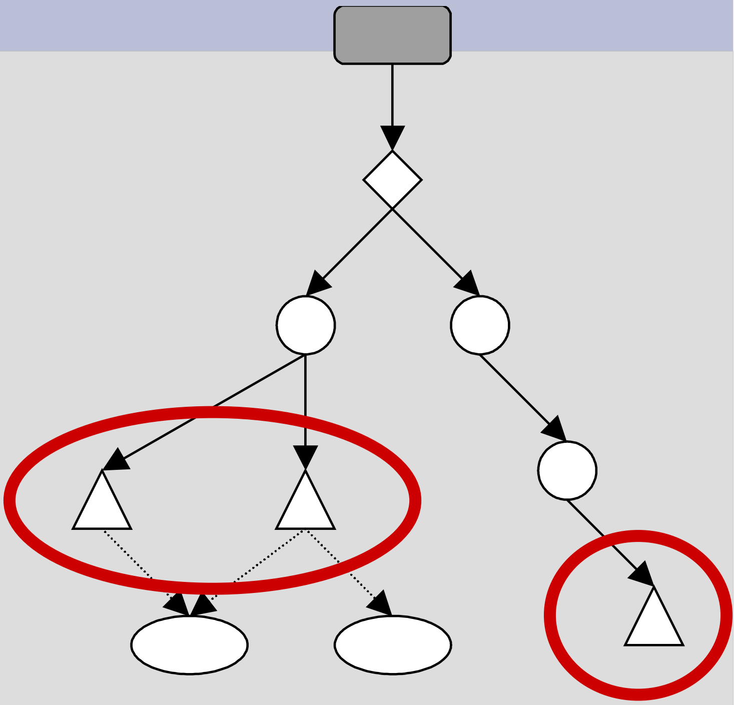
Morph

Shape3D

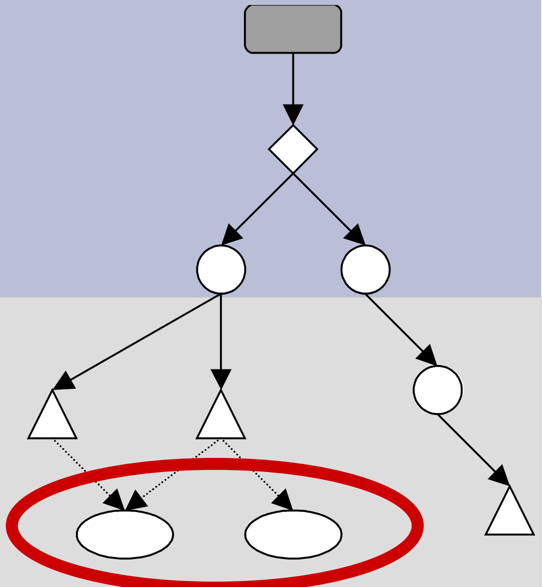
Sound

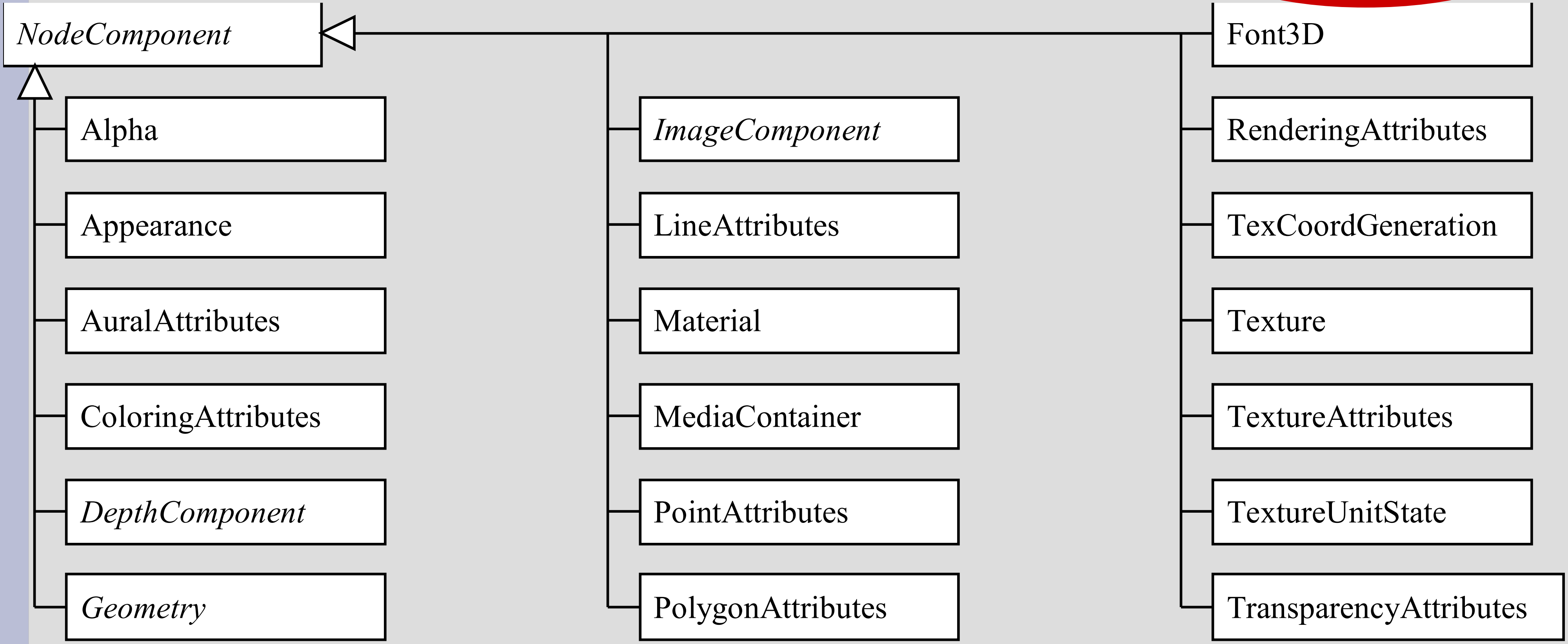
SoundScape

ViewPlatform



Node Components





NodeComponent

Alpha

Appearance

AuralAttributes

ColoringAttributes

DepthComponent

Geometry

ImageComponent

LineAttributes

Material

MediaContainer

PointAttributes

PolygonAttributes

Font3D

RenderingAttributes

TexCoordGeneration

Texture

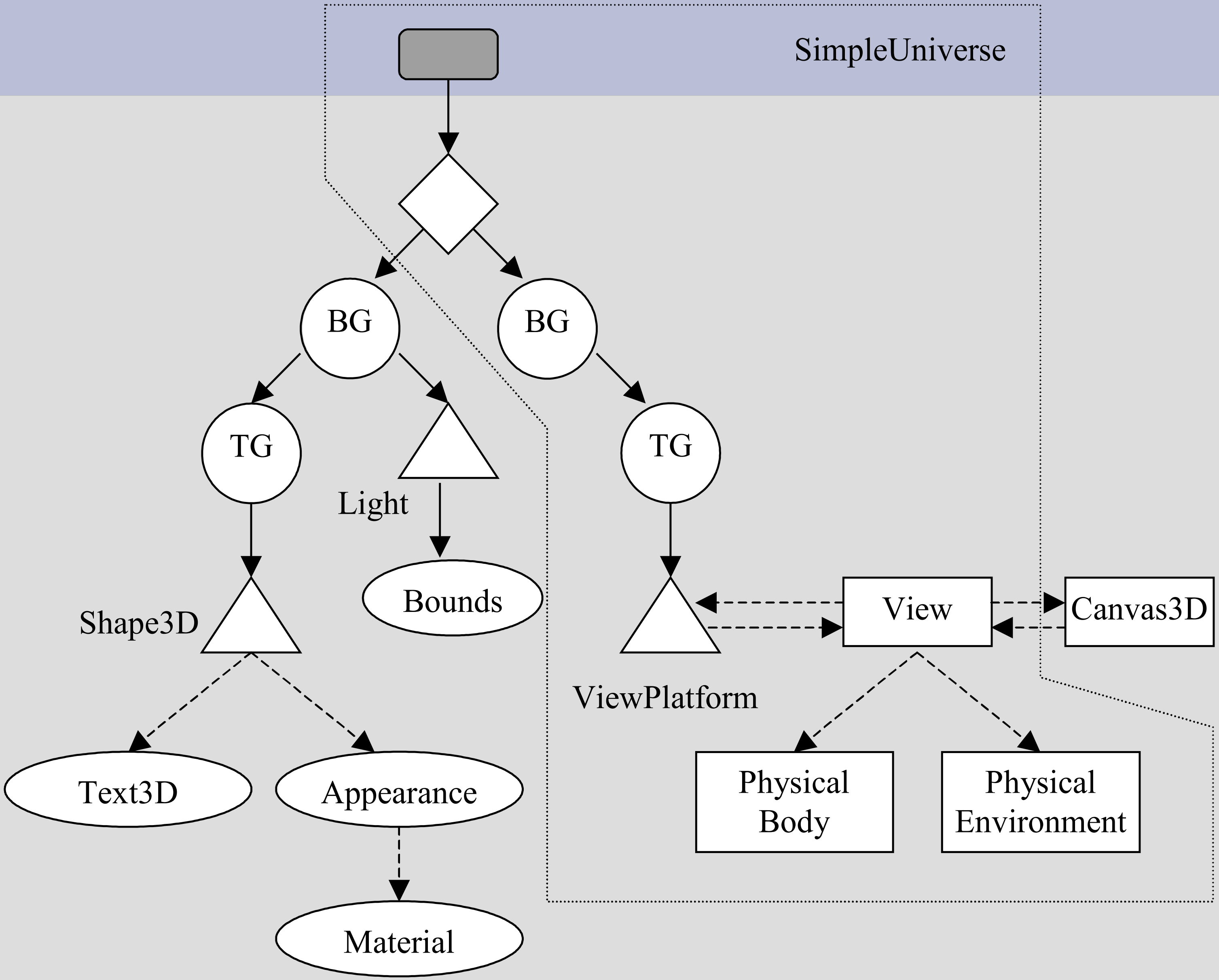
TextureAttributes

TextureUnitState

TransparencyAttributes

Scene Graph

The Java 3D "Hello" program



Use SimpleUniverse

- Create a Canvas3D object
- Create a SimpleUniverse object
- Add content branch

SimpleUniverse's View

- The View of SimpleUniverse
  - Projection: Perspective Projection
  - Field of View: $\pi/4$
- Also passes through the origin by default
  - Use: su.getViewingPlatform().setNominalViewingTransform(); to move view back along the Z axis if some objects near the origin