Lecture #10

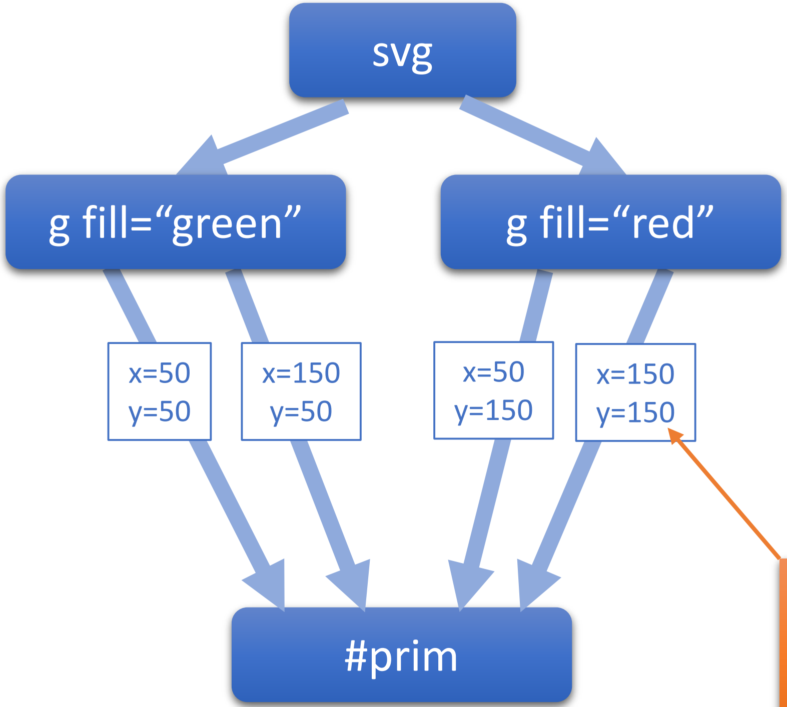
Scene Graphs

Computer Graphics
Winter Term 2016/17

Marc Stamminger / Roberto Grosso

Scene Graphs

- remember from Lecture #1

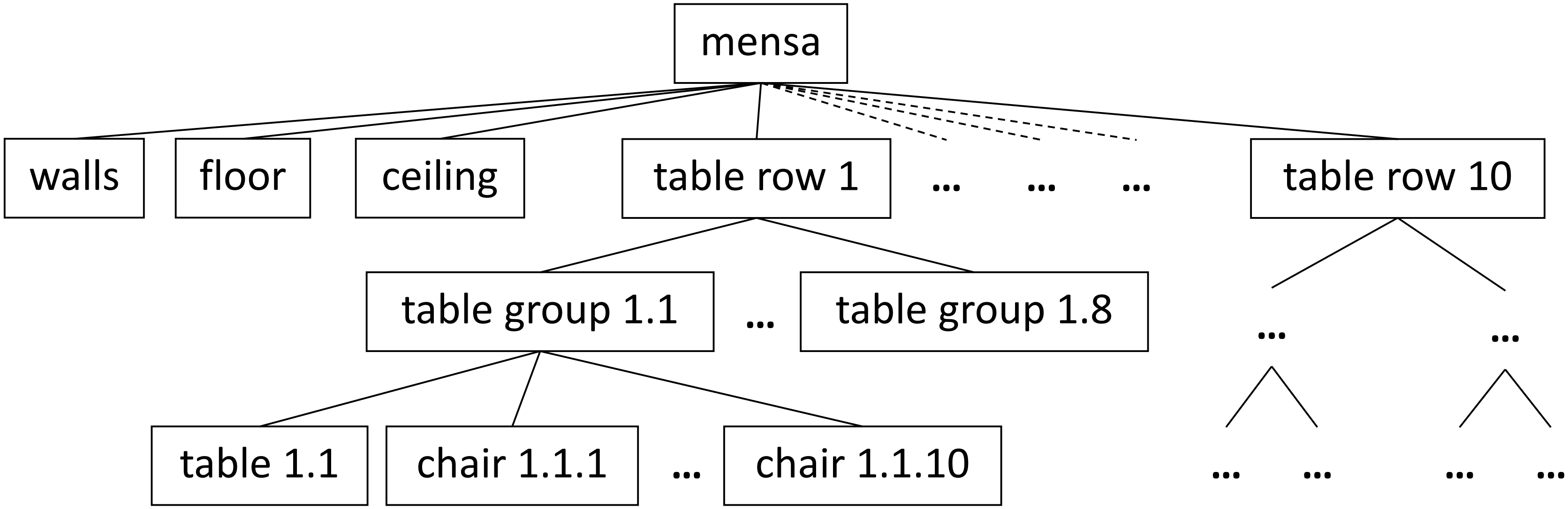


```
<svg id="svg">
    <defs>…</defs>
    <g fill="green">
        <use xlink:href="#prim" x="50
        <use xlink:href="#prim" x="15
    </g>
    <g fill="red">
        <use xlink:href="#prim" x="50
        <use xlink:href="#prim" x="15
    </g>
</svg>
```

Modeling Transformation

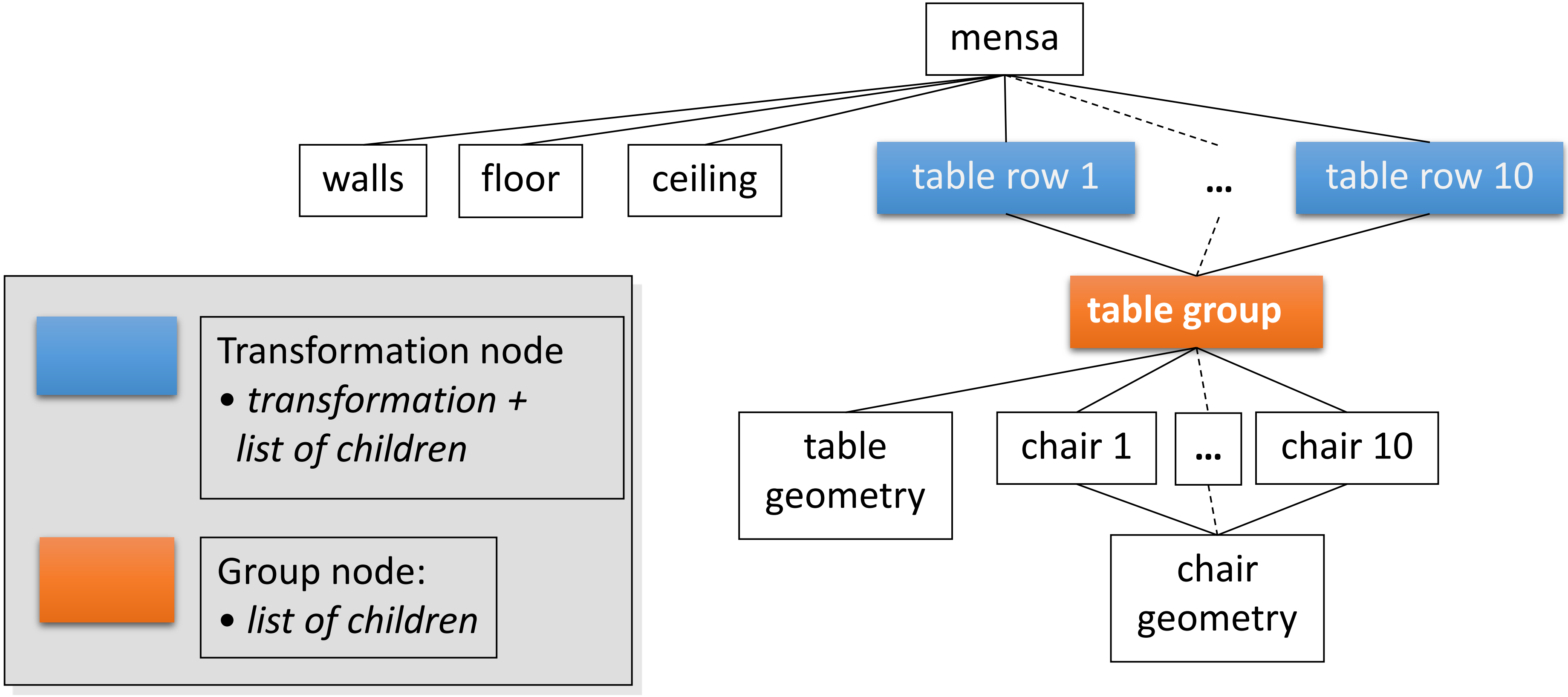
Scene Graphs

- Example: model of mensa scene
  - Walls, floor, ceiling, windows
    - 10 rows of tables
      - Every table row: 8 groups of tables and 10 chairs
  - ⇒ Scene tree



Hierarchical Modeling

- leaf nodes table and chair all have same geometry
- Just transformations of leaf nodes differ ⇒ scene graphs share these common geometries

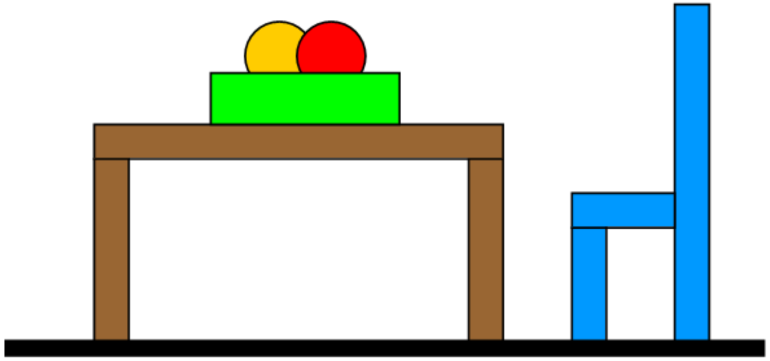


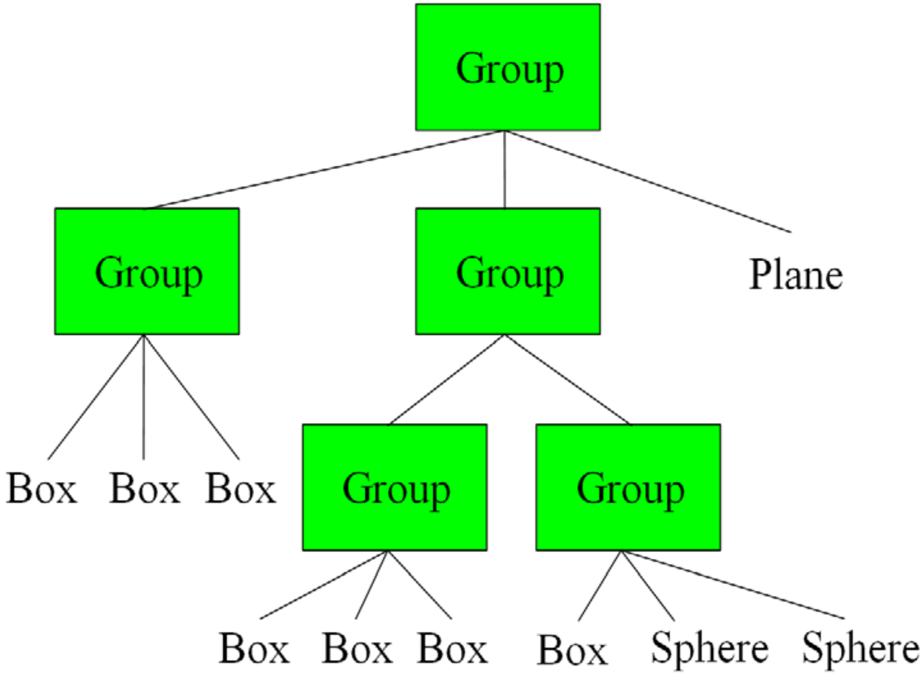
🟦 Transformation node
- *transformation + list of children*

🟧 Group node:
- *list of children*

Hierarchical Modeling

- Organization of scene





Group

Group Group Plane

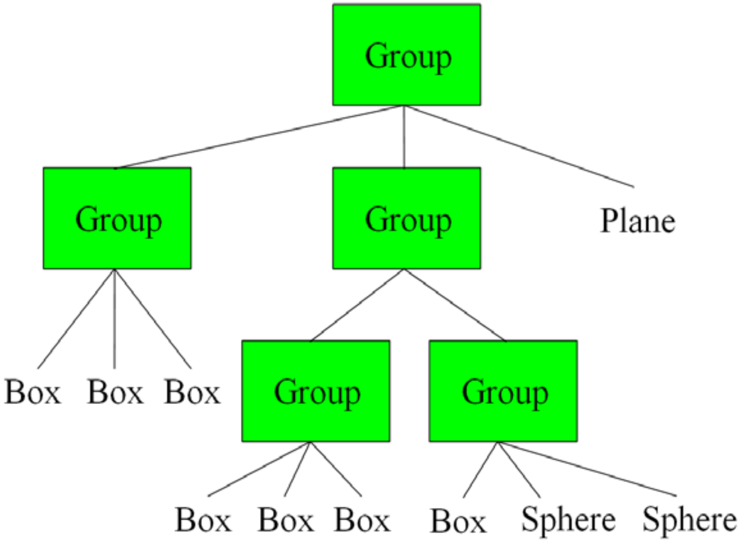
Box Box Box Group Group

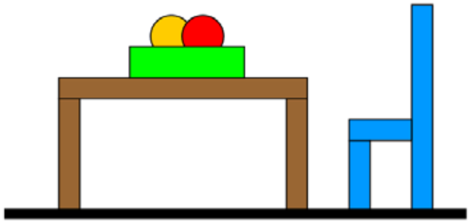
Box Box Box Box Sphere Sphere

Hierarchical Modeling

- Simple Example in scene format VRML with Groups

```
Group {
   numObjects 3
   Group {
     numObjects 3
     Box { <BOX PARAMS> }
     Box { <BOX PARAMS> }
     Box { <BOX PARAMS> }}
   Group {
     numObjects 2
     Group {
         Box { <BOX PARAMS> }
         Box { <BOX PARAMS> }
         Box { <BOX PARAMS> }}
     Group {
         Box { <BOX PARAMS> }
         Sphere { <SPHERE PARAMS> }
         Sphere { <SPHERE PARAMS> }}}
   Plane { <PLANE PARAMS> }}
```

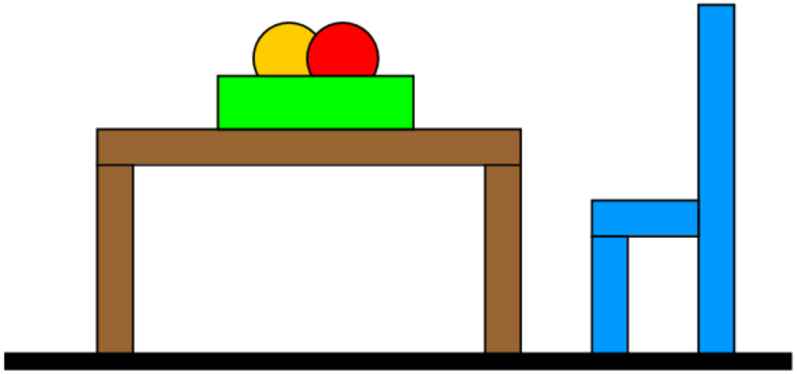




Hierarchical Modeling

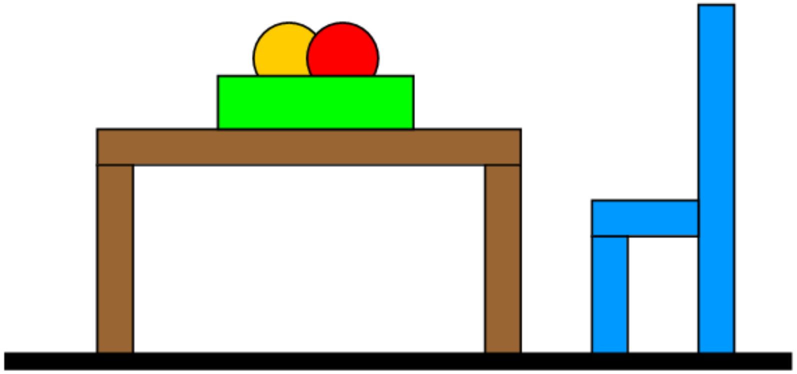
- Adding Materials

```
Group {
    numObjects 3
    Group {
        numObjects 3
        Box { <BOX PARAMS> }
        Box { <BOX PARAMS> }
        Box { <BOX PARAMS> } }
    Group {
        numObjects 2
        Group {
            Box { <BOX PARAMS> }
            Box { <BOX PARAMS> }
            Box { <BOX PARAMS> } }
        Group {
            Box { <BOX PARAMS> }
            Sphere { <SPHERE PARAMS> }
            Sphere { <SPHERE PARAMS> } } }
    Plane { <PLANE PARAMS> } }
```



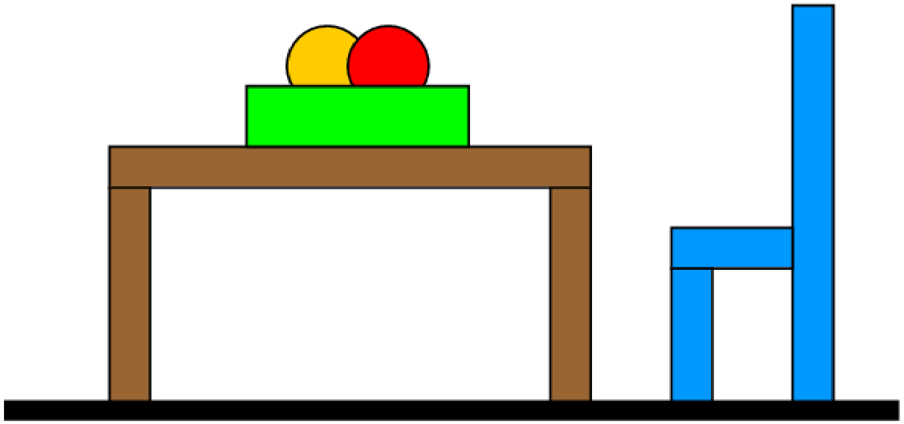
Hierarchical Modeling

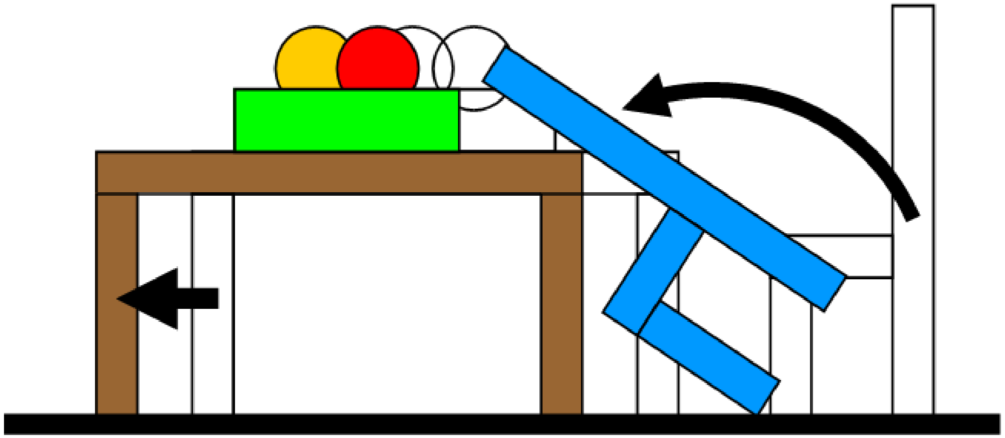
```
Group {
    numObjects 3
    Material { <BLUE> }
    Group {
        numObjects 3
        Box { <BOX PARAMS> }
        Box { <BOX PARAMS> }
        Box { <BOX PARAMS> } }
    Group {
        numObjects 2
        Material { <BROWN> }
        Group {
            Box { <BOX PARAMS> }
            Box { <BOX PARAMS> }
            Box { <BOX PARAMS> } }
        Group {
            Material { <GREEN> }
            Box { <BOX PARAMS> }
            Material { <RED> }
            Sphere { <SPHERE PARAMS> }
            Material { <ORANGE> }
            Sphere { <SPHERE PARAMS> } } }
            Material { <BLACK> }
    Plane { <PLANE PARAMS> } }
```



Hierarchical Modeling

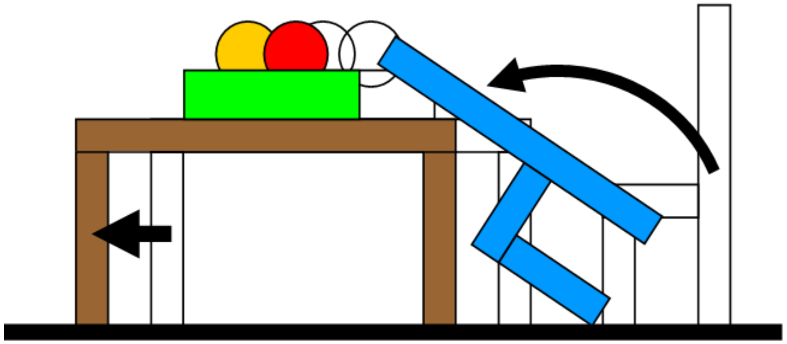
- Adding Transformations

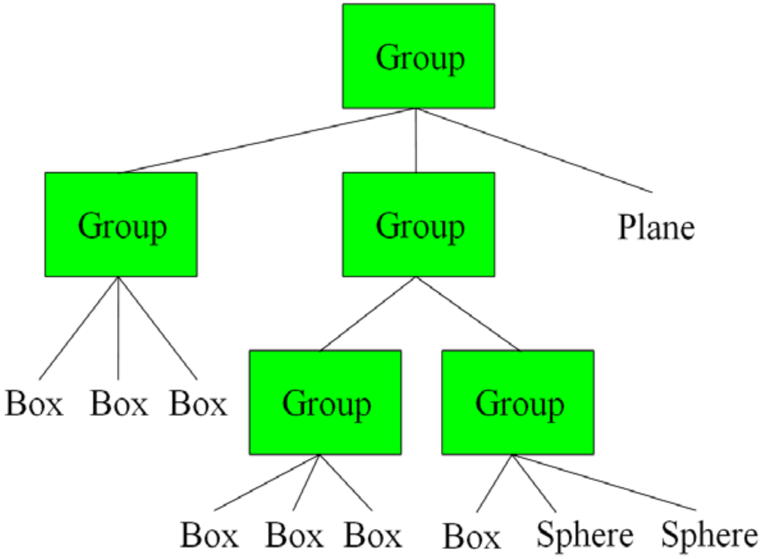




Hierarchical Modeling

- Transform nodes to position the logical groupings of objects within the scene



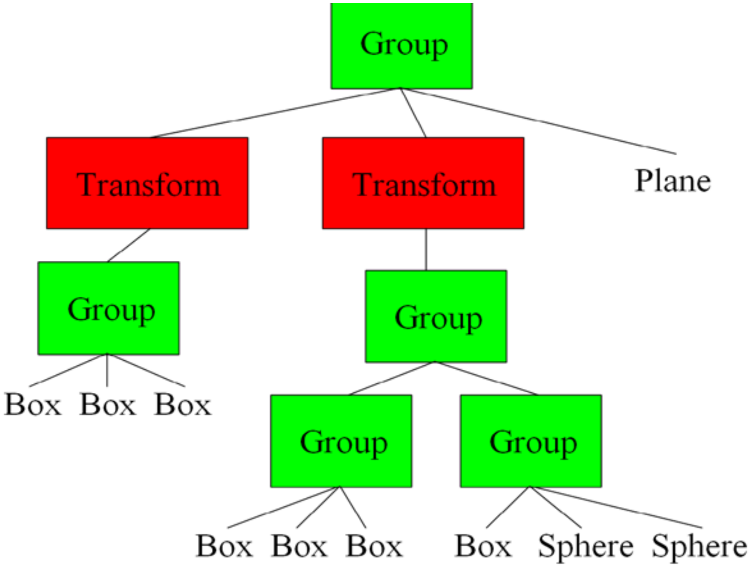


Group

Group Group Plane

Box Box Box Group Group

Box Box Box Box Sphere Sphere



Group

Transform Transform Plane

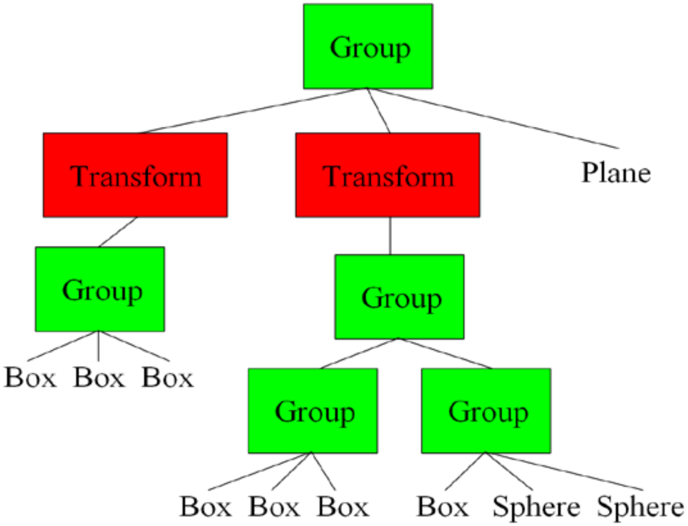
Group Group

Box Box Box Group Group

Box Box Box Box Sphere Sphere

Hierarchical Modeling

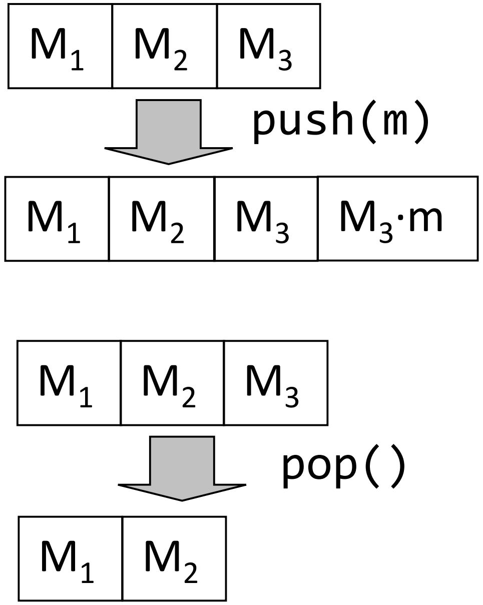
```
Group {
    numObjects 3
    Transform {
        ZRotate { 45 }
        Group {
            numObjects 3
            Box { <BOX PARAMS> }
            Box { <BOX PARAMS> }
            Box { <BOX PARAMS> } } }
    Transform {
        Translate { -2 0 0 }
        Group {
            numObjects 2
            Group {
                Box { <BOX PARAMS> }
                Box { <BOX PARAMS> }
                Box { <BOX PARAMS> } }
            Group {
                Box { <BOX PARAMS> }
                Sphere { <SPHERE PARAMS> }
                Sphere { <SPHERE PARAMS> } } } }
    Plane { <PLANE PARAMS> } }
```



Hierarchical Modeling

- Rendering this graph is done using a **depth traversal of the graph**
- There is always one current material, one current matrix, etc.
- Matrices are efficiently cumulated using a **Matrix Stack**

- Matrix stack
  - stack of 4 x 4 matrices
  - Push (matrix m)
    - Duplicate top matrix
    - Apply m to top matrix
  - Pop ()
    - Remove top matrix



Hierarchical Modeling

- Matrix stacks
  - Top row: current transformation matrix

Load(I)	Push	Mult(T_{V1})	Push	Mult(R_{180})	Push	Mult(T_{S1})	Chair	Pop	...
I	I	I T_{V1}	I T_{V1}	I T_{V1} R	I T_{V1} R	I T_{V1} R T_{S1}		I T_{V1} R	
	I	I	I T_{V1}	I T_{V1}	I T_{V1} R	I T_{V1} R		I T_{V1}	
			I	I	I T_{V1}	I T_{V1}		I	
					I	I			

  - here, push() only duplicates top matrix