Hierarchical Modeling

- Basic Modelling Concepts
- Building Complex Objects
- Scene Graph
- Scene Graph Traversal

- **Basic Modelling Concepts**
- Building Complex Objects
- Scene Graph
- Scene Graph Traversal

Model

- The creation and manipulation of a system representation is termed modeling
- Any single representation is called a model of the system, which could be defined graphically or purely descriptively
- Graphical models are also called geometric models, because the component parts of a system are represented with geometric entities
- We will use the term model to mean a computer generated geometric representation of a system

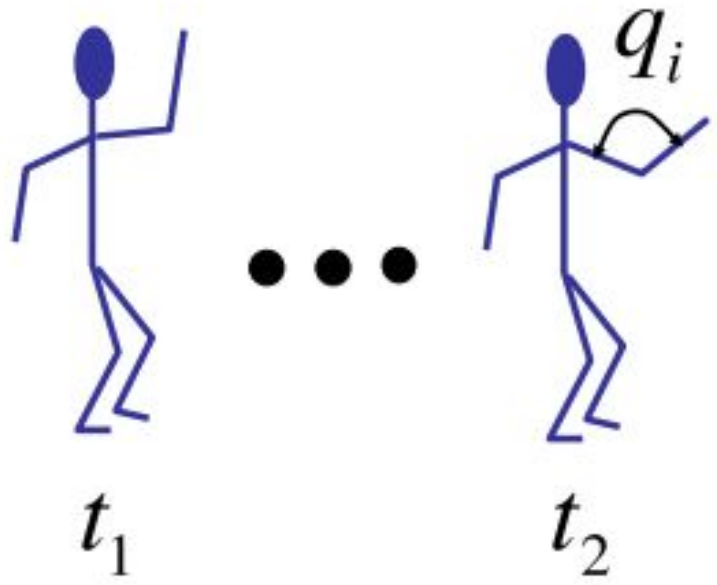
System Representation

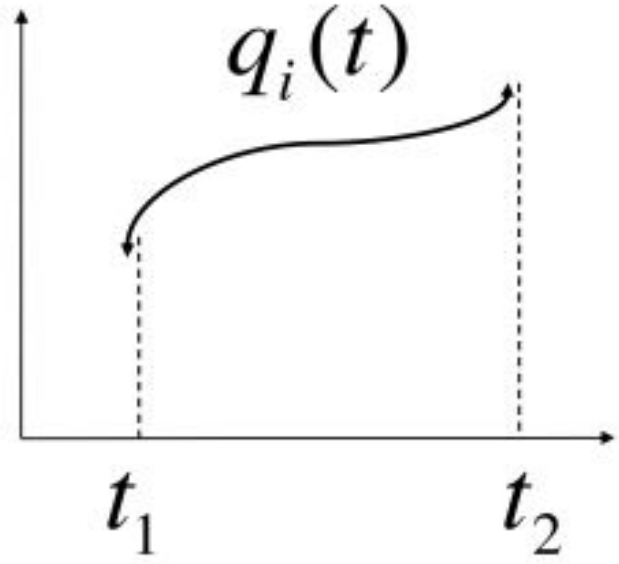
- Component parts of the system are displayed as geometric structures, called symbols
- Each occurrence of a symbol within a model is called an instance of that symbol
- Information describing a model is usually provided as a combination of geometric and non-geometric data
- There are two methods for specifying the information needed to construct and manipulate a model
    - Store the information in a data structure
    - Specify the information in procedures

Articulated Models

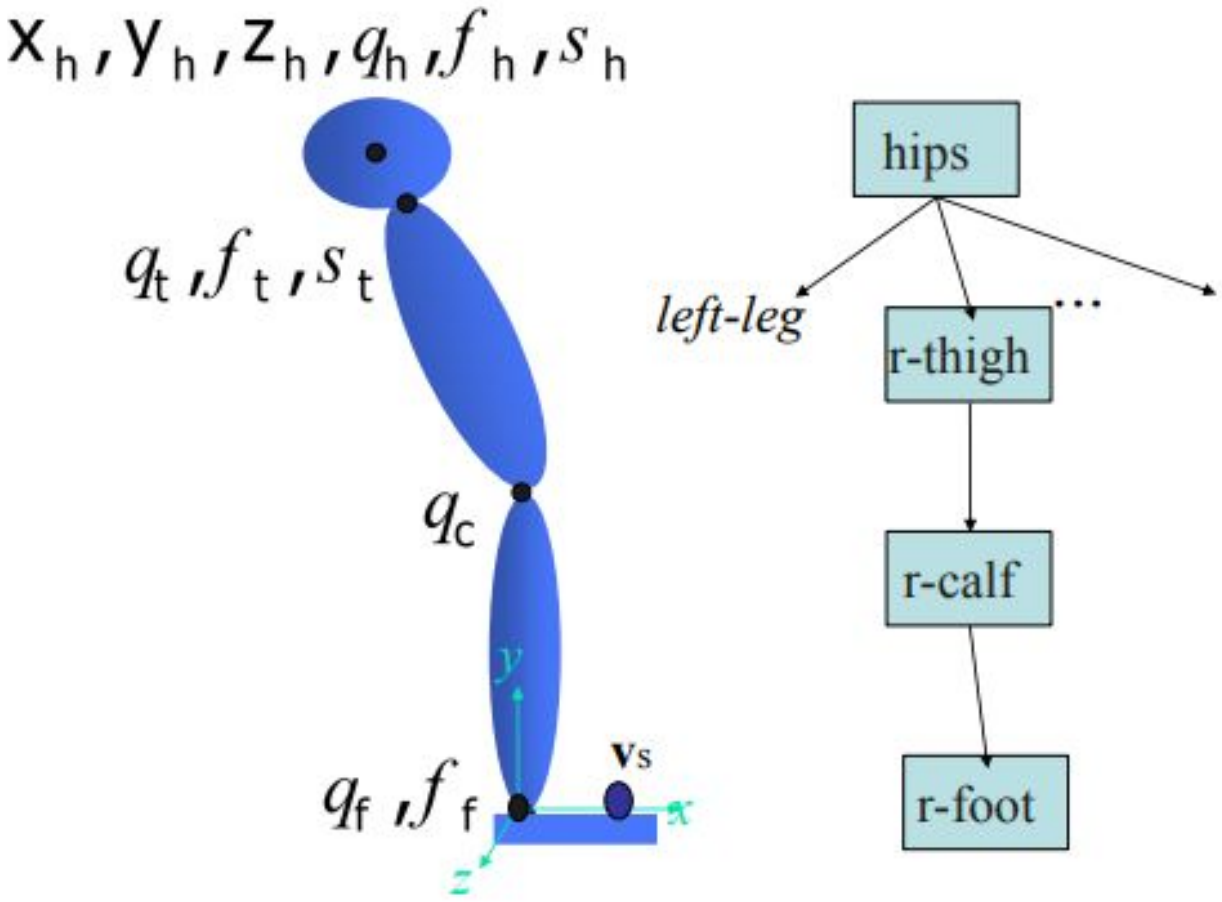
• They are rigid parts connected by joints where each joint has some angular degrees of freedom

• They can be animated by specifying the joint angles as functions of time





Skeleton Hierarchy



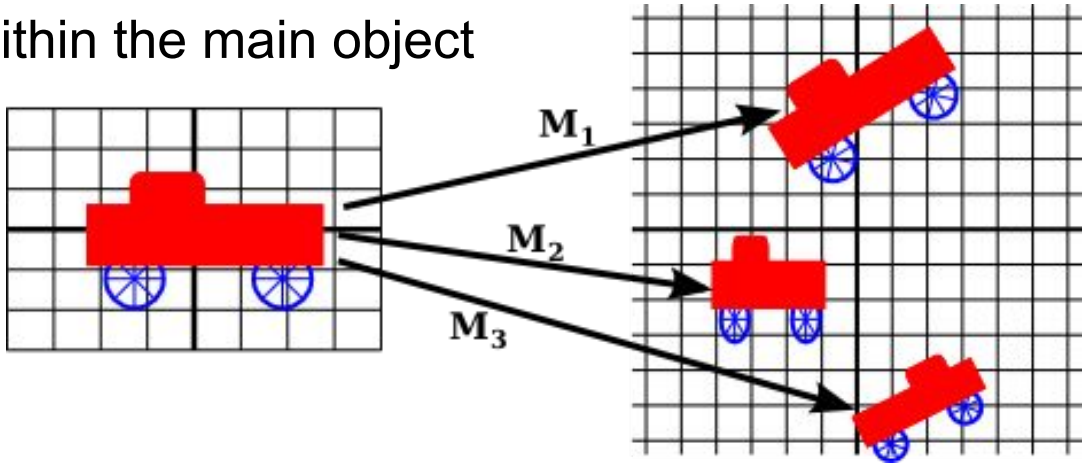
- Basic Modelling Concepts
- **Building Complex Objects**
- Scene Graph
- Scene Graph Traversal

Symbol Hierarchies

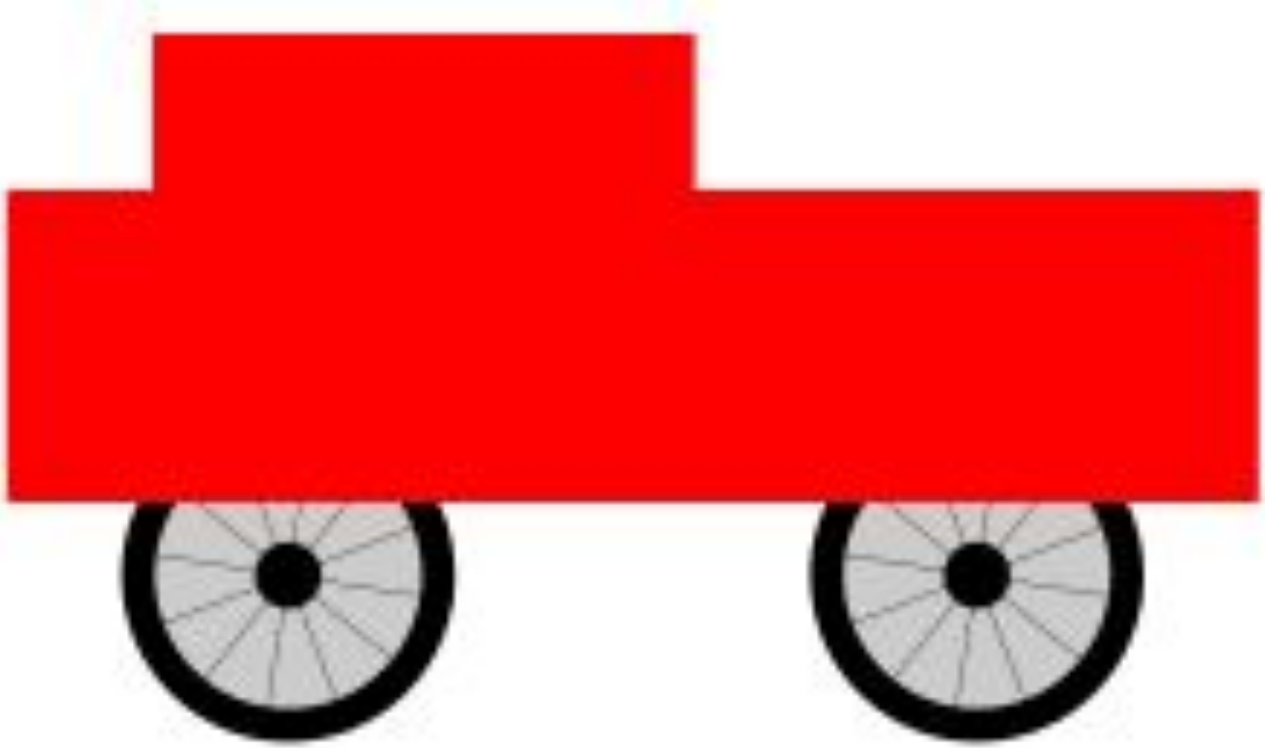
- Modules are the basic elements for the model are defined as simple geometric shapes appropriate to the type of model under consideration
- Modules themselves can be grouped to form higher-level objects, and so on

Building Complex Objects

- A major motivation for introducing a new coordinate system is that it should be possible to use the coordinate system that is most natural to the scene that you want to draw
- We draw each small component object, in its own coordinate system, and use a modeling transformation to move the sub-object into position within the main object



Example



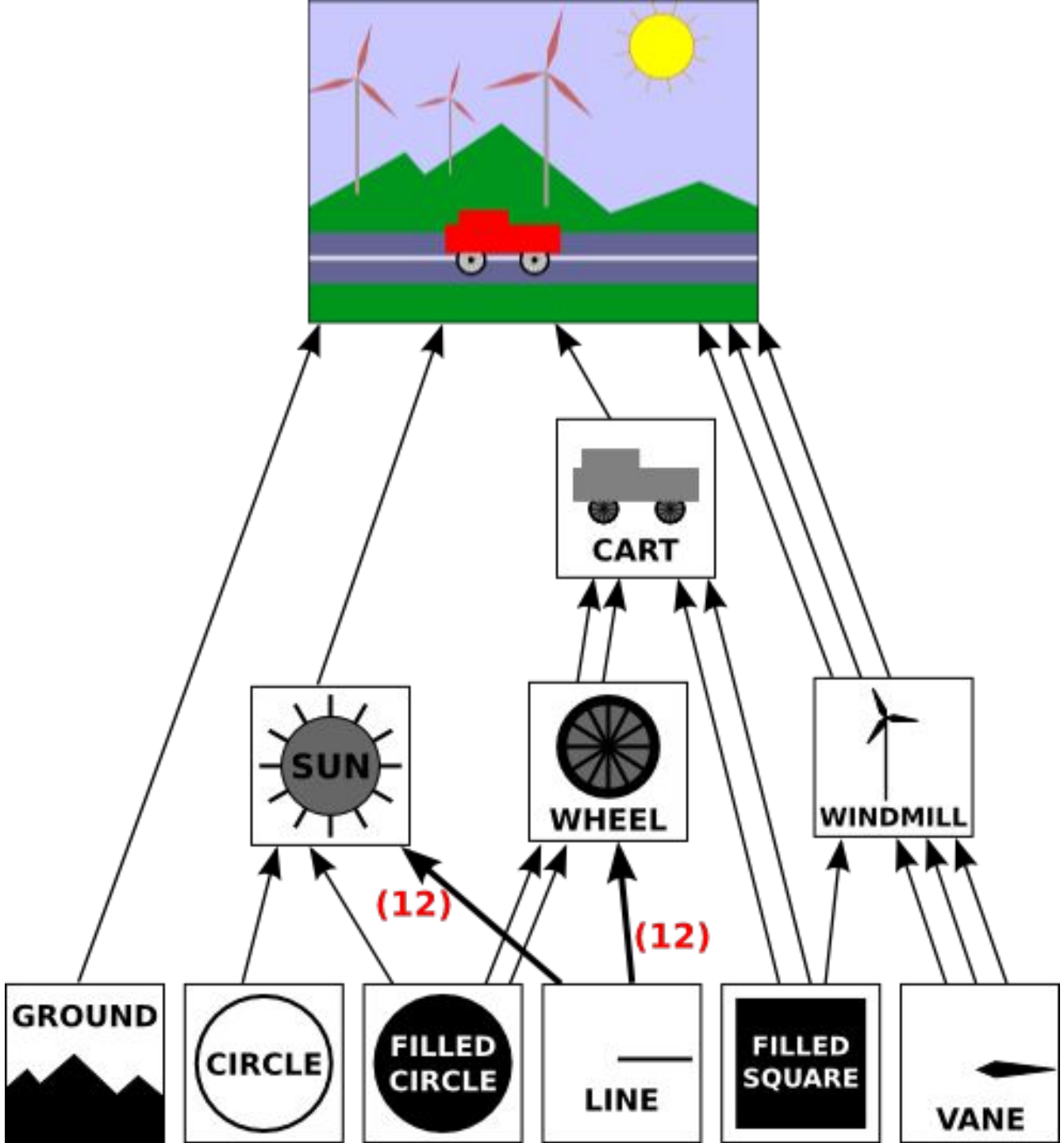
```
subroutine drawCart() :
    saveTransform()         // save the current transform
    translate(-1.65,-0.1) // center of first wheel
    scale(0.8,0.8)          // scale to reduce radius
    drawWheel()             // draw the first wheel
    restoreTransform()      // restore the saved transform
    saveTransform()         // save it again
    translate(1.5,-0.1)     // center of second wheel
    scale(0.8,0.8)          // scale to reduce radius
    drawWheel(g2)           // draw the second wheel
    restoreTransform()      // restore the transform
    setDrawingColor(RED)  // use red color for the rectangles
    fillRectangle(-3, 0, 6, 2)       // draw the body of the cart
    fillRectangle(-2.3, 1, 2.6, 1)  // draw the top of the cart
```

- Basic Modelling Concepts
- Building Complex Objects
- Scene Graph
- Scene Graph Traversal

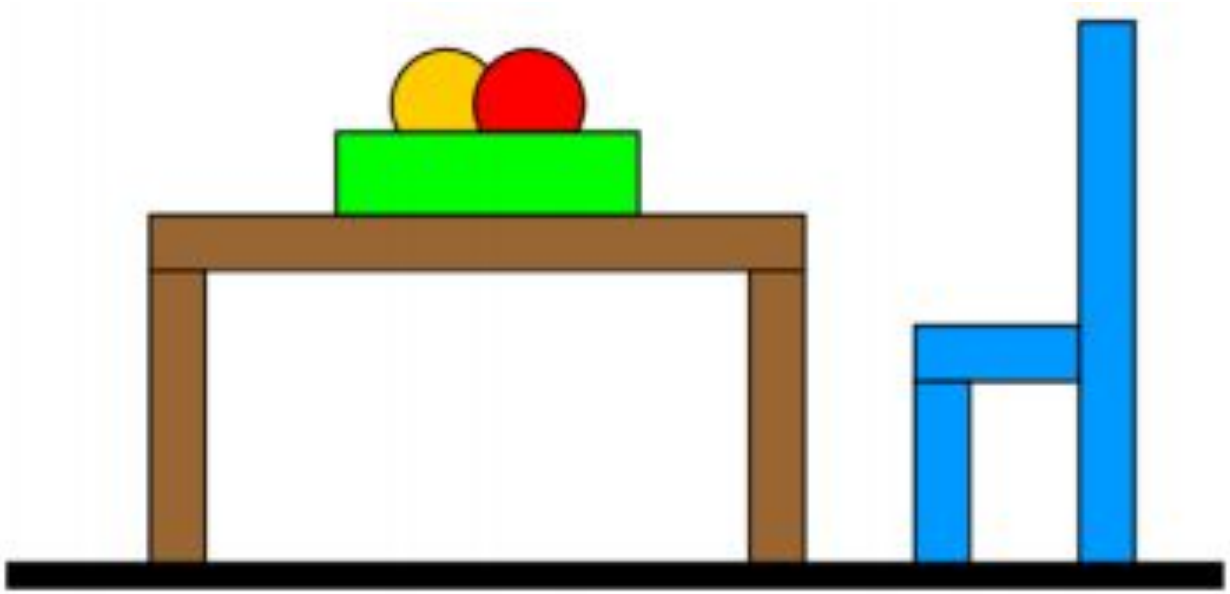
Scene Graph

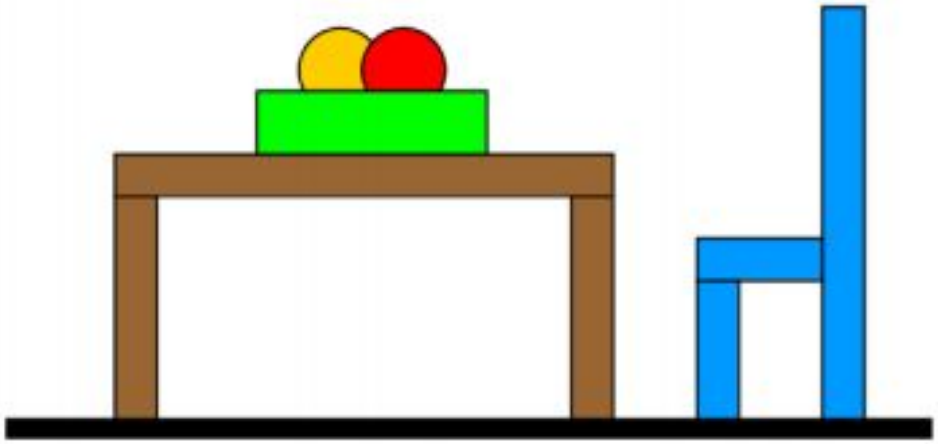
- Logically, the components of a complex scene form a structure. In this structure, each object is associated with the sub-objects that it contains
- A scene graph is a tree-like structure, with the root representing the entire scene, the children of the root representing the top-level objects in the scene, and so on
- A single object can have several connections to one or more parent objects. Each connection represents one occurrence of the object in its parent object and can be associated with a modeling transformation that places the sub-object into its parent object

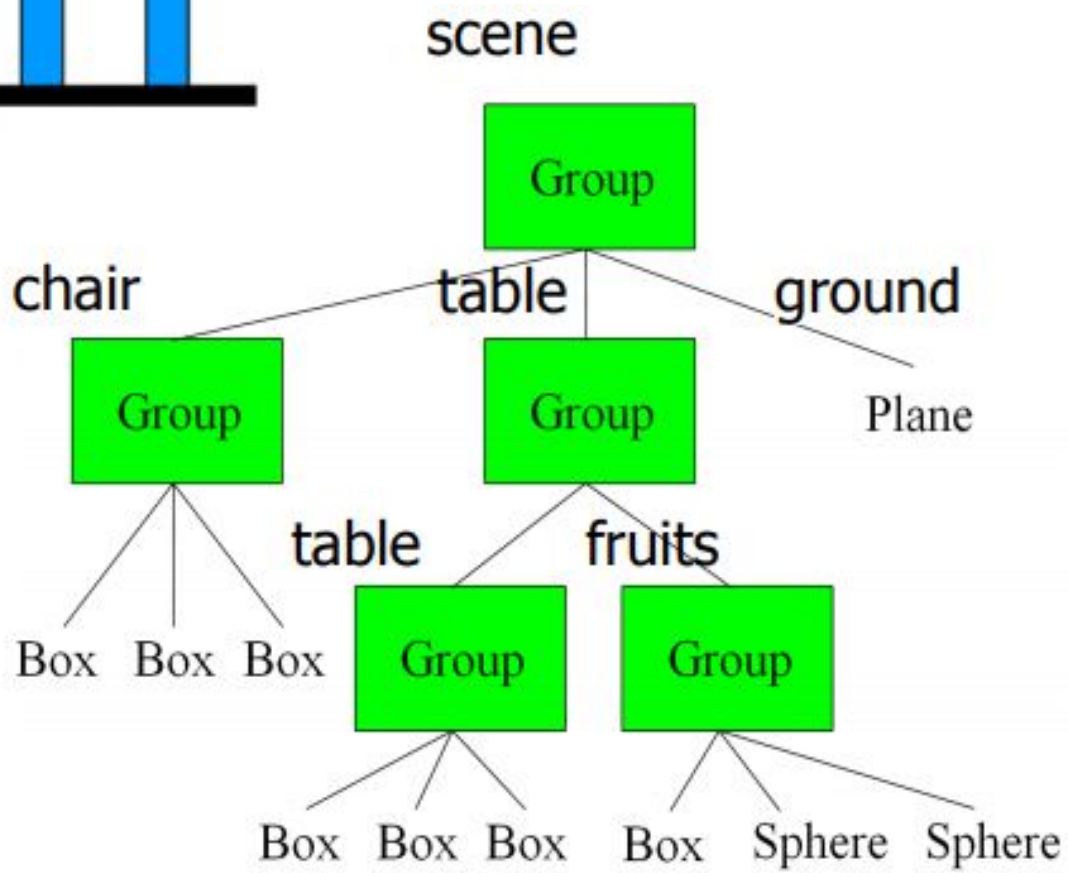
Example


CART
SUN
WHEEL
WINDMILL
(12)
(12)
GROUND
CIRCLE
FILLED CIRCLE
LINE
FILLED SQUARE
VANE

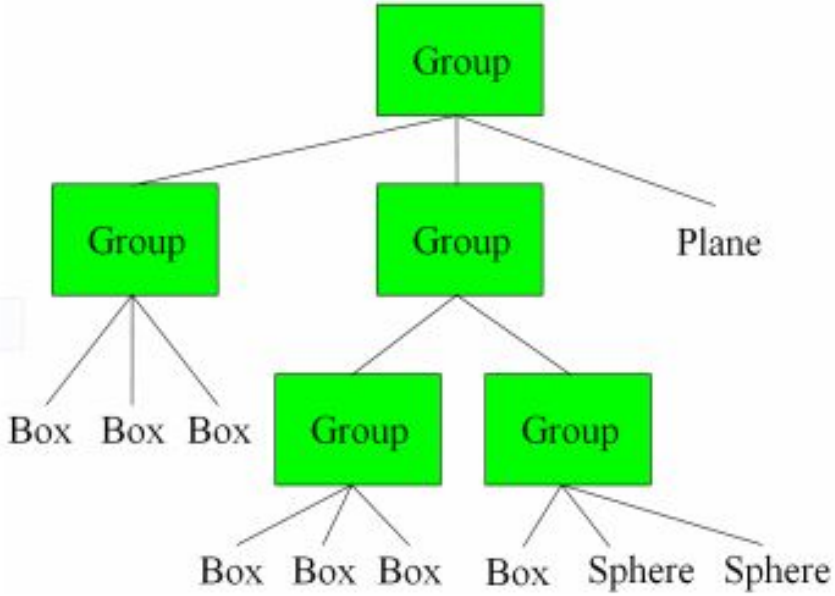
Example

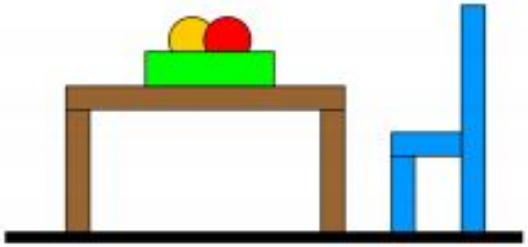





scene
Group
chair
table
ground
Group
Group
Plane
table
fruits
Box
Box
Box
Group
Group
Box
Box
Box
Box
Sphere
Sphere

```
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> } }
```

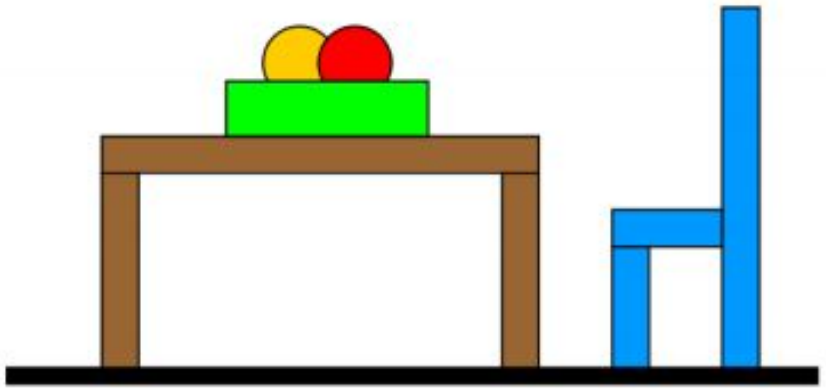




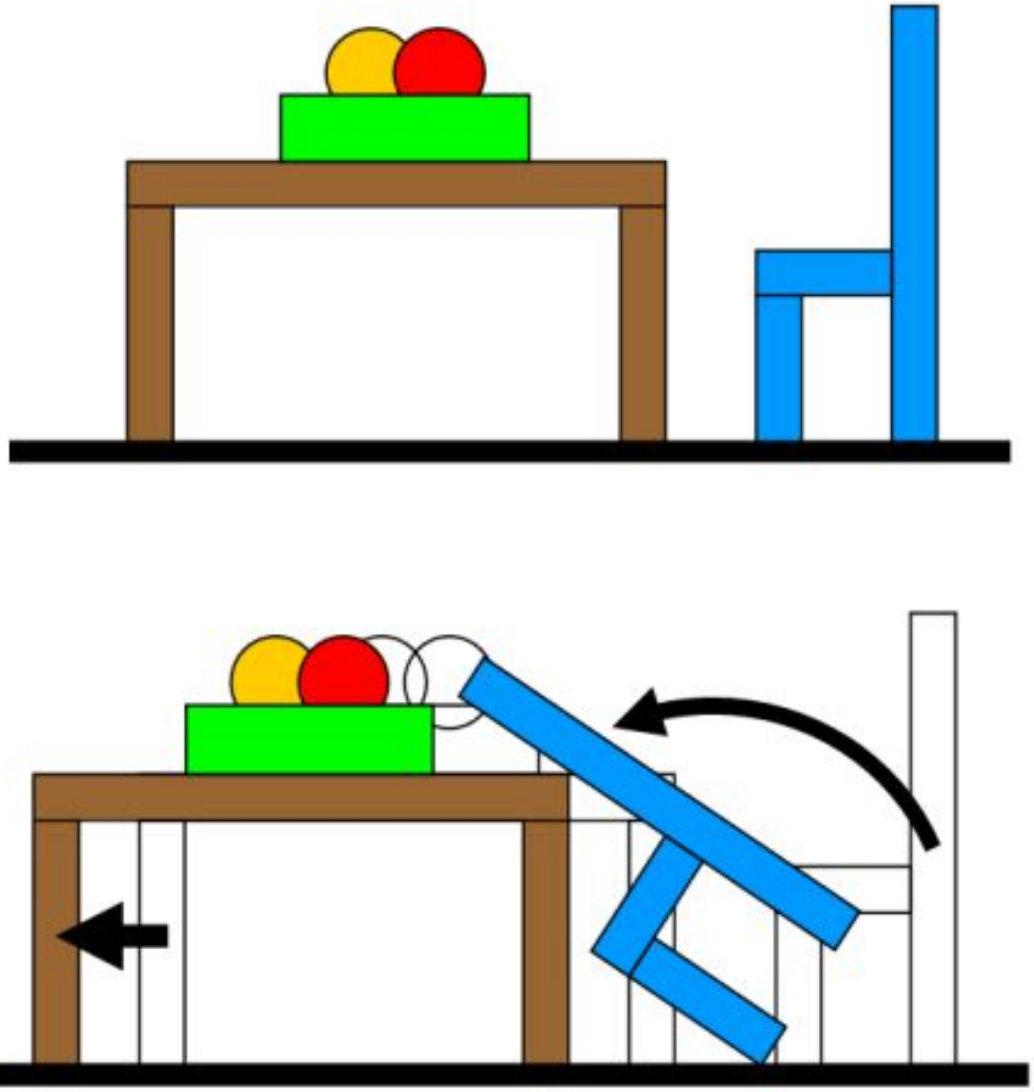
Adding Attributes

```
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> } } }

    Plane { <PLANE PARAMS> } }
```



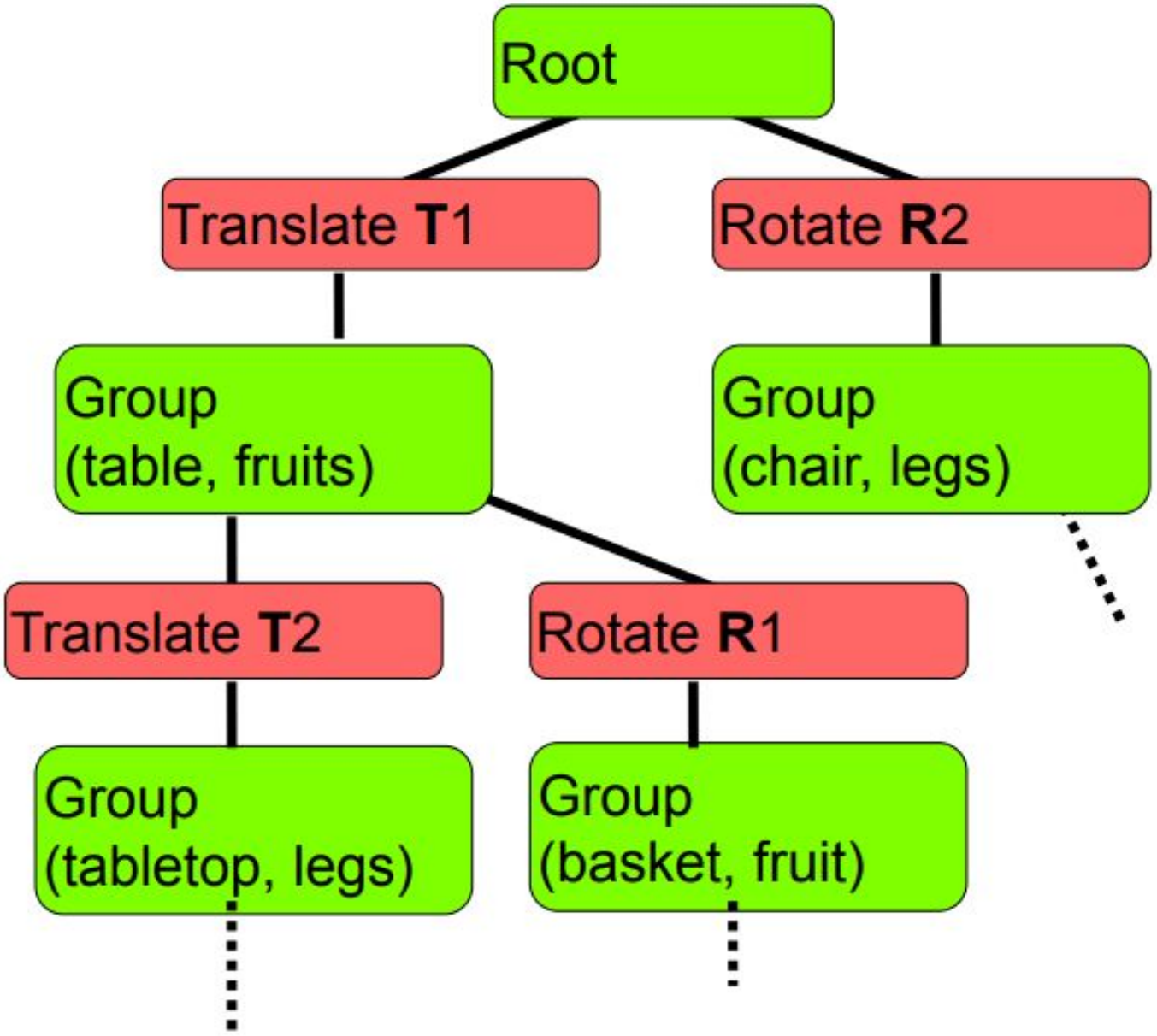
Adding Transformations



Scene Graph Traversal

- Traversal algorithm keeps a **transformation state S** from world coordinates
    - **S** is initialized to identity in the beginning
- Geometry nodes always drawn using current **S**
- When visiting a transformation node **T** multiply current state **S** with **T**, then visit **child nodes**
    - Nodes below will have the **new transformation**
- When all children have been visited, **undo the effect of T**

Example


Root
Translate T1
Rotate R2
Group
(table, fruits)
Group
(chair, legs)
Translate T2
Rotate R1
Group
(tabletop, legs)
Group
(basket, fruit)

References

- Donald D. Hearn, M. Pauline Baker, Warren Carithers. (2016) Computer Graphics with OpenGL
- Wojciech Matusik. MIT OpenCourseWare 6.837 Computer Graphics

Thank You