Scenegraphs

- Scenegraph nodes:
  - Compound node
  - Leaf node

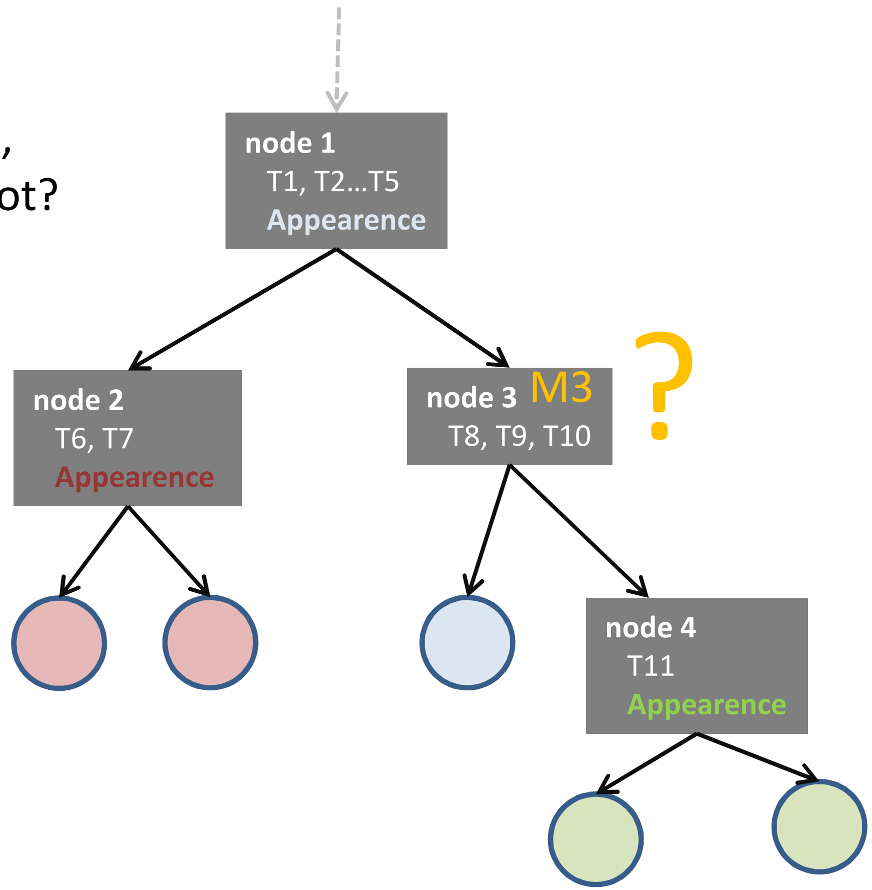
- Node properties
  - Geometric transformations and propagation

Compound nodes and leaves

Questions

What is the transformation matrix M3,
to apply to sub-tree with node 3 as root?

How to store M3?



Compound nodes and leaves

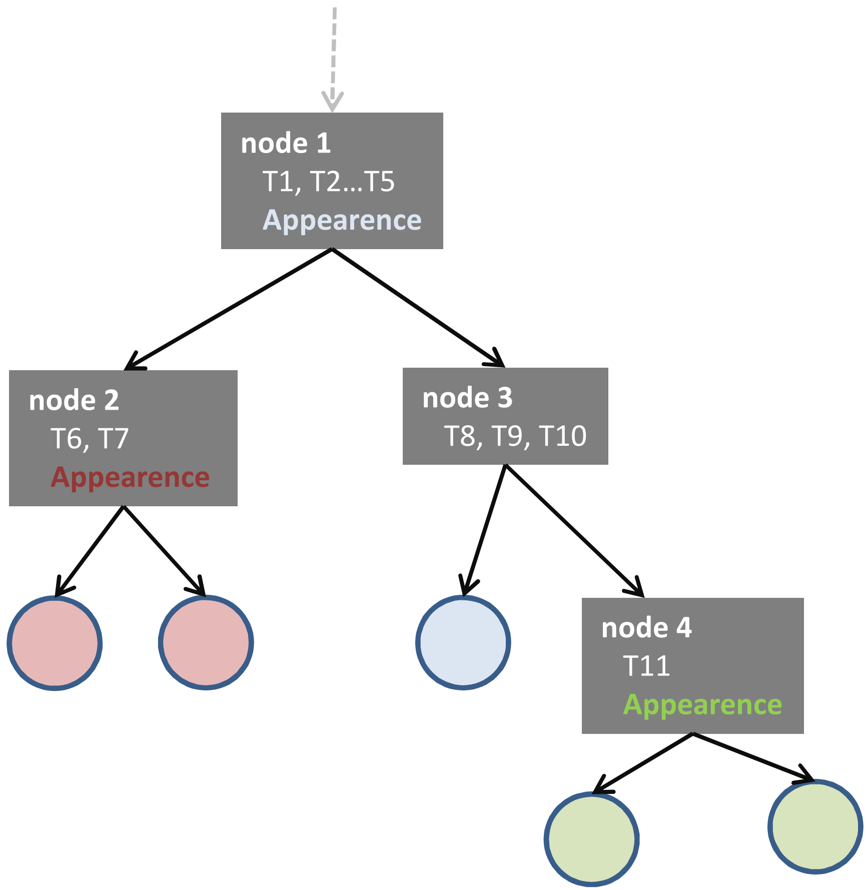
Solution 1: transformation list

Keep a transformation list, accumulating transformations for parent nodes.

For node 3: T1, T2,…,T5, T8, T9, T10

In rendering time traverse the list and apply transformations.

Drawback: **not efficient!**



Compound nodes and leaves

Solution 2: calculate and keep a transformation matrix for each node

Use OpenGL to perform the calculation of a transformation matrix for each compound node in the scenegraph!

Requires a function to retrieve the current MODEL_VIEW matrix. Ex 1:

```
float m1[4][4];
glGetFloatv(GL_MODELVIEW_MATRIX, &m1[0][0]);
```

Ex 2:

```
float m1[4][4];
float m2[4][4];

glLoadIdentity();                                   // M = I
glRotated(20.0,1,0,0);                              // M = M.Rx(20)
glRotated(-45,0,1,0);                               // M = M.Ry(-45)
glTranslated(0.0,0.0,-25);                          // M = M.Tz(-25)
glGetFloatv(GL_MODELVIEW_MATRIX, &m1[0][0]);        // m1 = M

glTranslated(0.0,15.0,0.0);                         // M = M.Ty(15)
glGetFloatv(GL_MODELVIEW_MATRIX, &m2[0][0]);        // m2 = M
```

Compound nodes and leaves

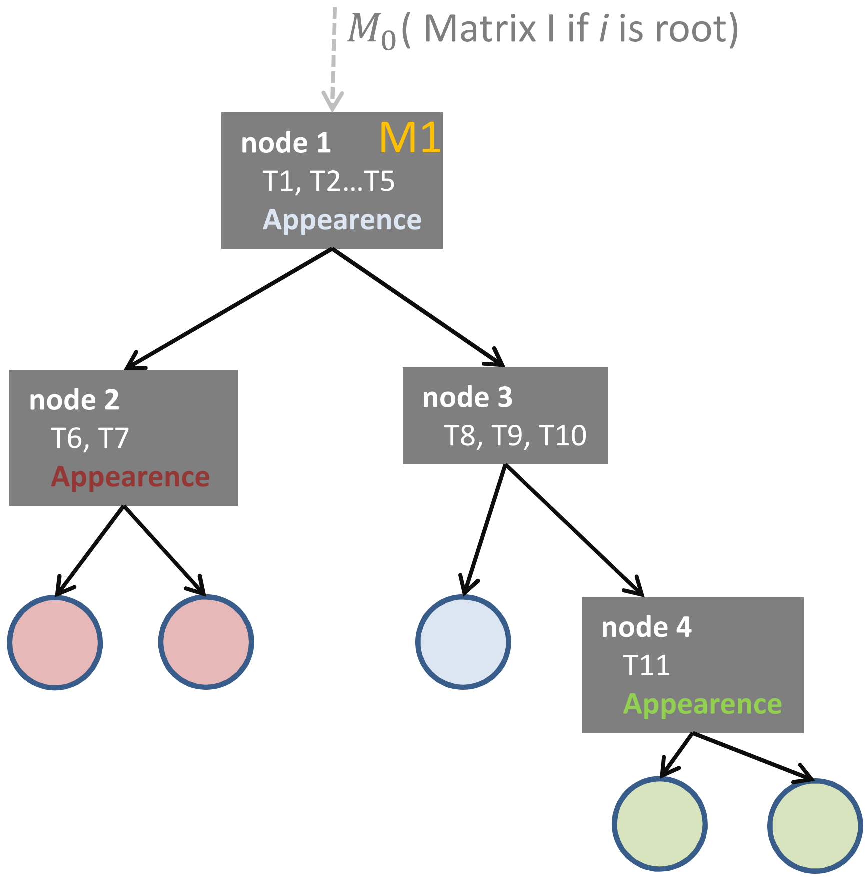
Solution 2: calculate transformation matrix for node 1

Calculate:

$M_1 = I \; x \; M_0 x \; (I \, .T_1.T_2. \ldots T_5)$

Store M1 in node 1

Important: *I* is the identity matrix



Compound nodes and leaves

Transformation Matrix for nodes 2 and 3

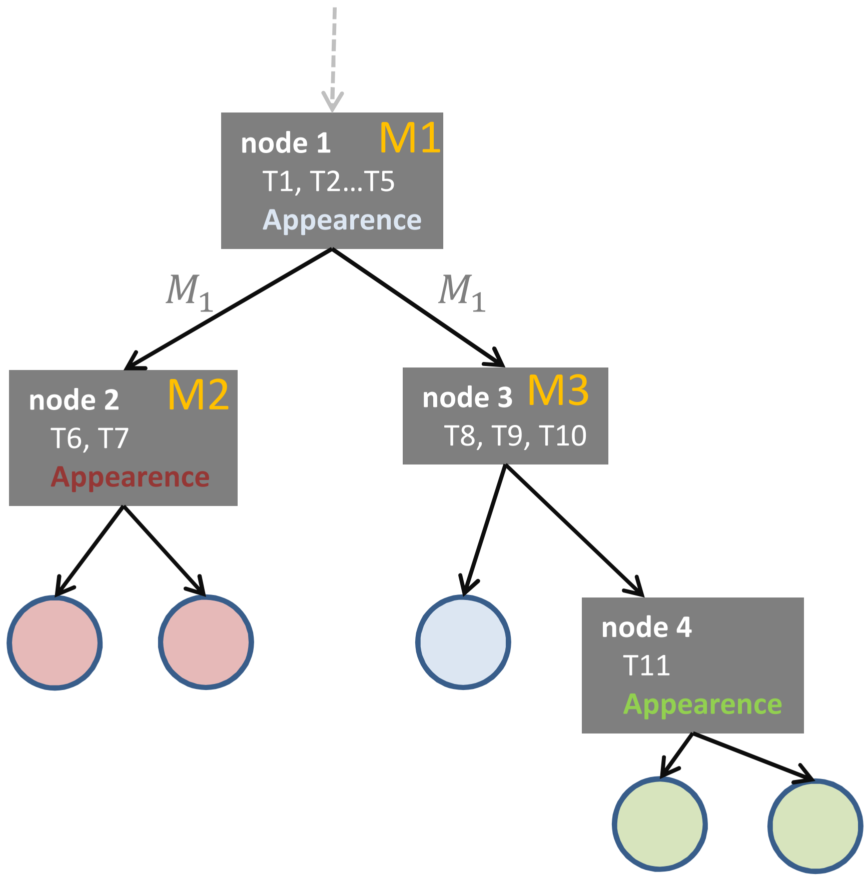
Calculate M_2

$$M_2 = I\ x\ M_1 x\ (I\ .T_1.T_2)$$

Calculate M_3

$$M_3 = I\ x\ M_1 x\ (I\ .T_1.T_2.T_3)$$

Store M2 in node 2 and M3 in node 3



Compound nodes and leaves

OpenGL transformation Matrix for node 3

$$M_3 = I\ x\ M_1 x\ (I\ .T_1.T_2.T_3)$$

```
// calculate m = (I.T1.T2.T3). In this example assume:
// T1 = Rx(50)
// T2 = Sy(1.5)
// T3 = Tz(10)

float m[4][4];
glLoadIdentity();                                        // M = I
glRotated(50.0,1,0,0);                                   // M = M.Rx(50.0)
glScaled(1.5,0,1,0);                                     // M = M.Sy(1.5)
glTranslated(0,0,10.0);                                  // M = M.Tz(10.0)
glGetFloatv(GL_MODELVIEW_MATRIX, &m[0][0]);              // m = M

// now calculate I x M1 x m
// assumed M1 is passed from parent node
float M3[4][4];
glLoadIdentity();                                        // M = I
glMultMatrixf(&M1);                                      // M = M x M1
glMultMatrixf(&m);                                       // M = M x m
glGetFloatv(GL_MODELVIEW_MATRIX, &M3[0][0]);             // M3 = M

// we now got M3 calculated!
```

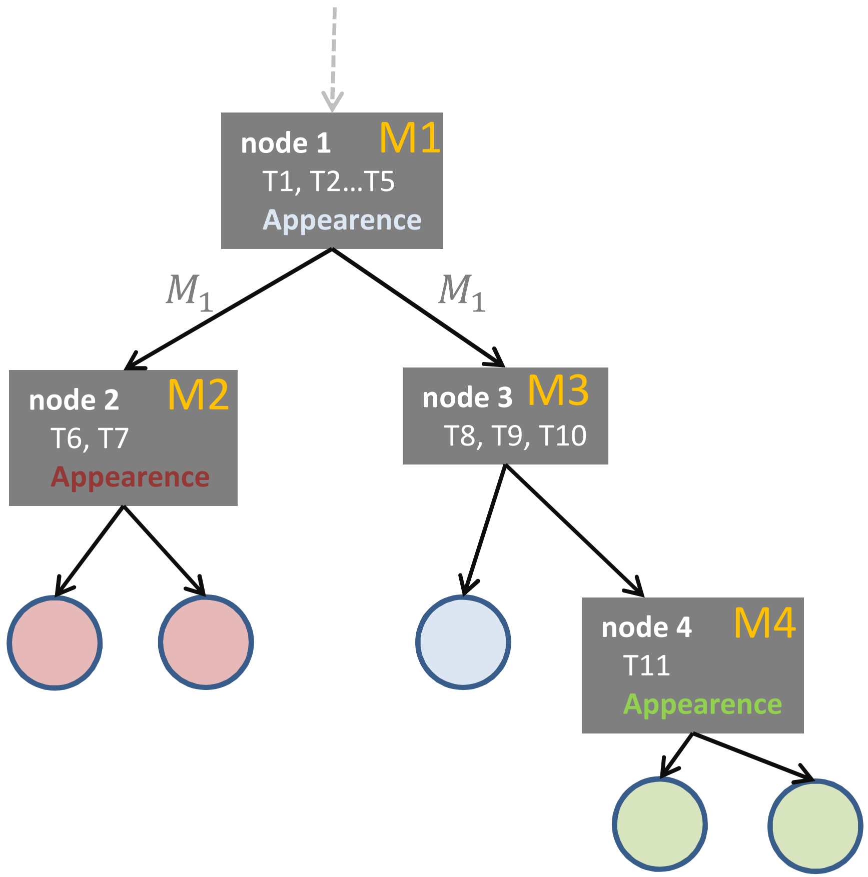
Compound nodes and leaves

Transformation Matrix for node 4

Calculate M_4

$$M_4 = I\ x\ M_3 x\ (I\ .T_1)$$

Store M4 in node 4



Geometric transformations

From parser to a transformation matrix **m**

```
<transformations>
        <rotate axis=“x” angle=“40” />
        <rotate axis=“y” angle=“20” />
        <scale factor=“1.0 5.0 2.0” />
        <translate to=“10.0 5.0 3.0” />
</transformations>
```

Rx(40.0); Ry(20.0); Sy(5.0); Sy(2.0); T(10,5,3)

```
// now calculate I x M1 x m
// assumed M1 is passed from parent node
float m[4][4];
glLoadIdentity();
glRotated(40.0,1,0,0);
glRotated(20.0,0,1,0);
glScaled(5,0,1,0);
glScaled(2,0,0,1);
glTranslated(10.0,5.0.3.0);
glGetFloatv(GL_MODELVIEW_MATRIX, &m[0][0]);
```

→ **m**

Whatever the number of geometric transformations defined for a node, a single matrix **m** is required to represent the set.