Rotation about an axis (Section 10.2)

3D rotation UI (Section 10.3)

Polygons (Chapter 11)

Hidden Surface Removal (Chapter 12)

Hierarchical Modelling (Chapter 13)

Picking (Chapter 14)

3D rotation UI (Section 10.3)

Rotation about an axis (Section 10.2)

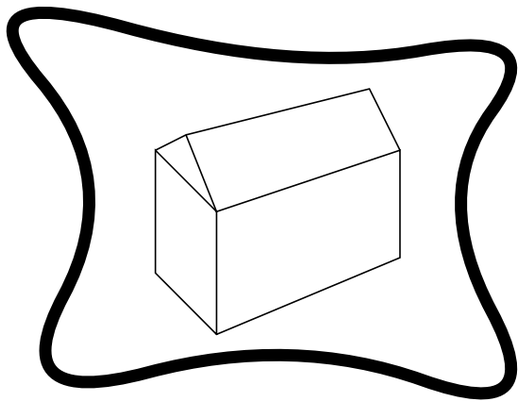
Hierarchical Modelling (Chapter 13)

Picking (Chapter 14)

Polygons (Chapter 11)

Hidden Surface Removal (Chapter 12)

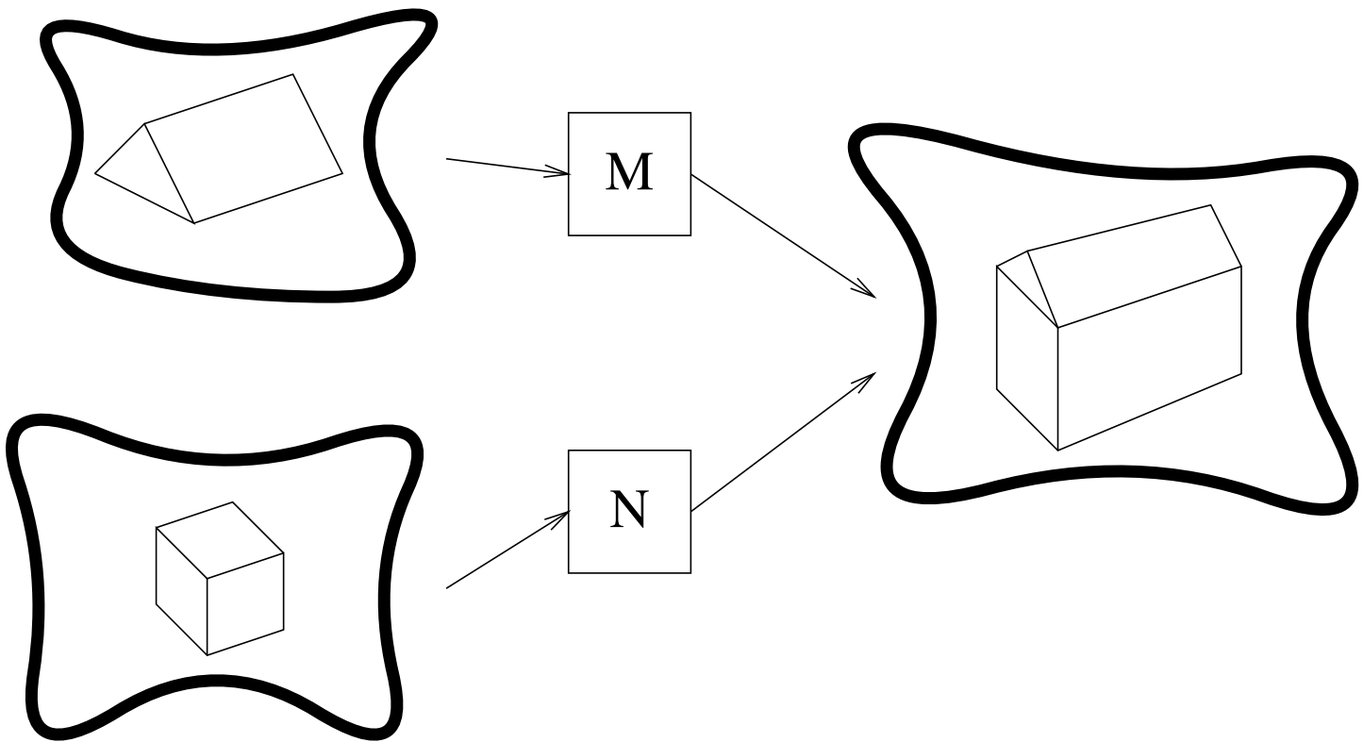
A house is made up of a body (cube)
and a roof (prism).

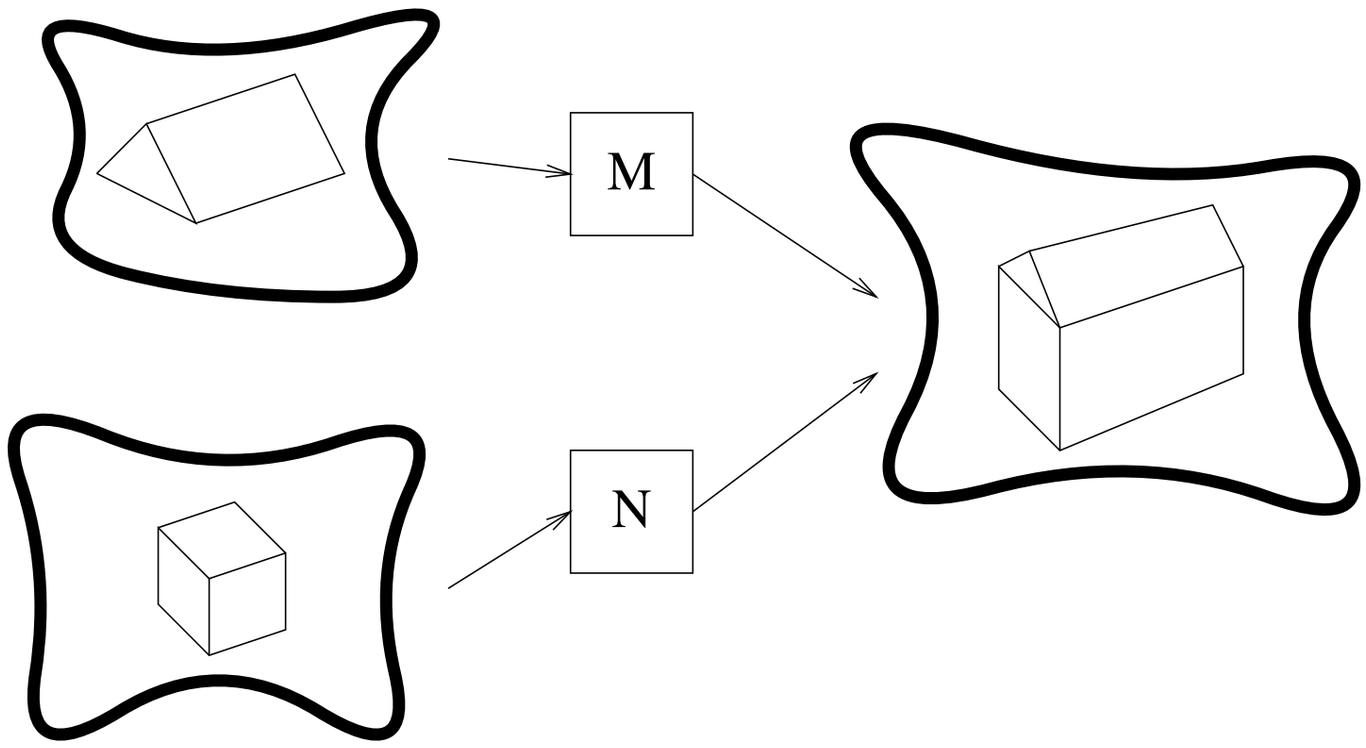


```
procedure House():
  Prism()
  Cube()
```

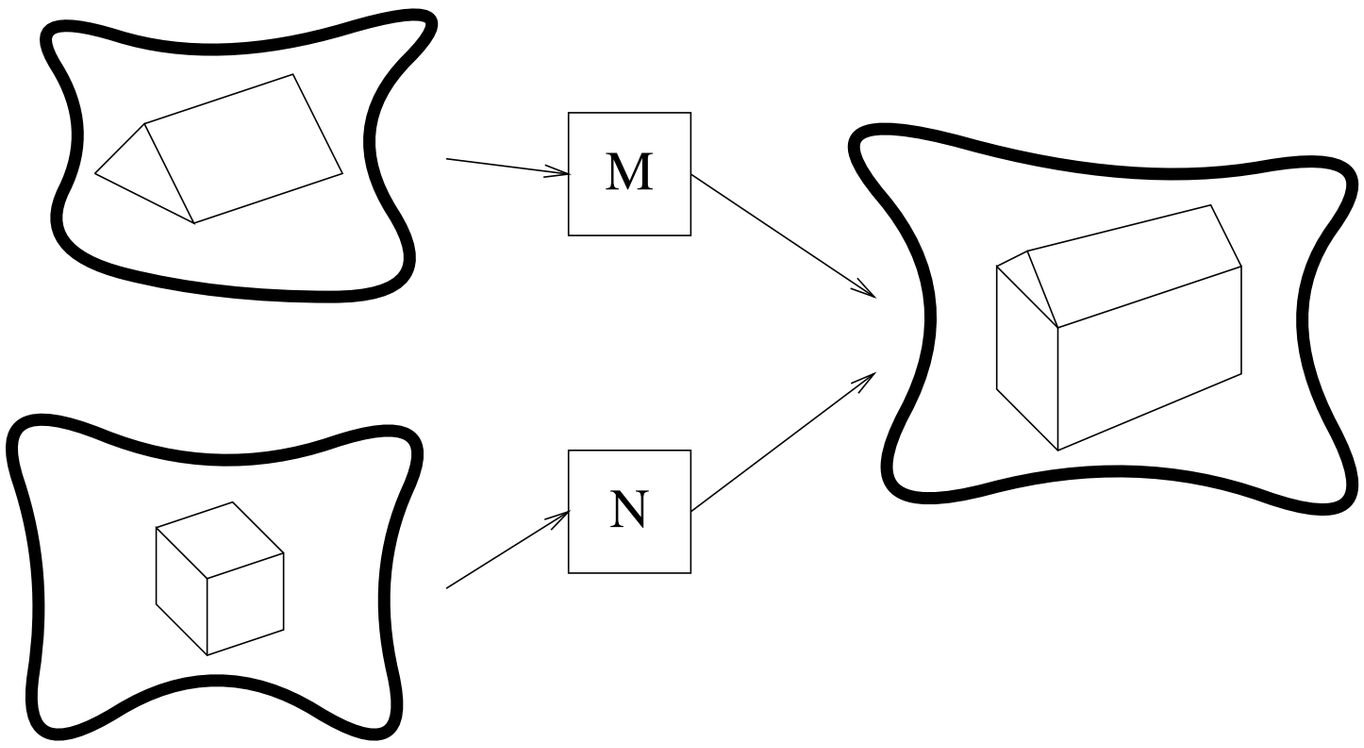
...but it would be nicer to define each in its own modelling coordinate system.

```
procedure Prism(W):
  setUniform(W)
  drawPrismMesh()

procedure Cube(W):
  setUniform(W)
  drawCubeMesh()
```



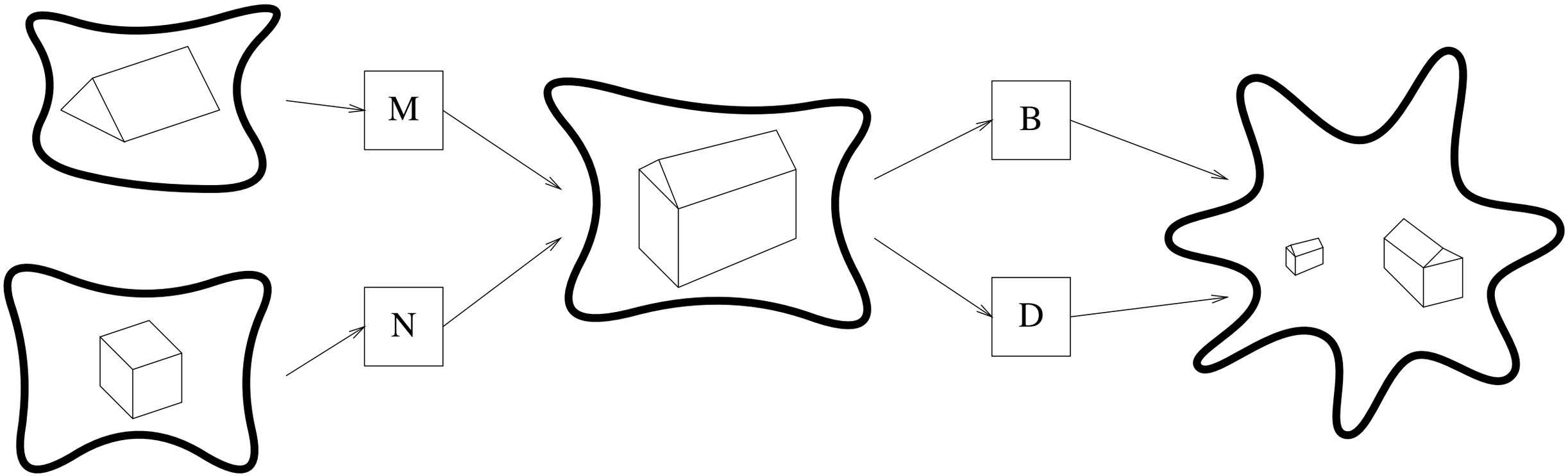



```
procedure Prism(W):
  setUniform(W)
  drawPrismMesh()

procedure Cube(W):
  setUniform(W)
  drawCubeMesh()

procedure House():
  Prism(M)
  Cube(N)
```




```
procedure Prism(W):
  setUniform(W)
  drawPrismMesh()

procedure Cube(W):
  setUniform(W)
  drawCubeMesh()

procedure House(W):
  Prism(W*M)
  Cube(W*N)
```

We can then have multiple instances (copies) of the house model.

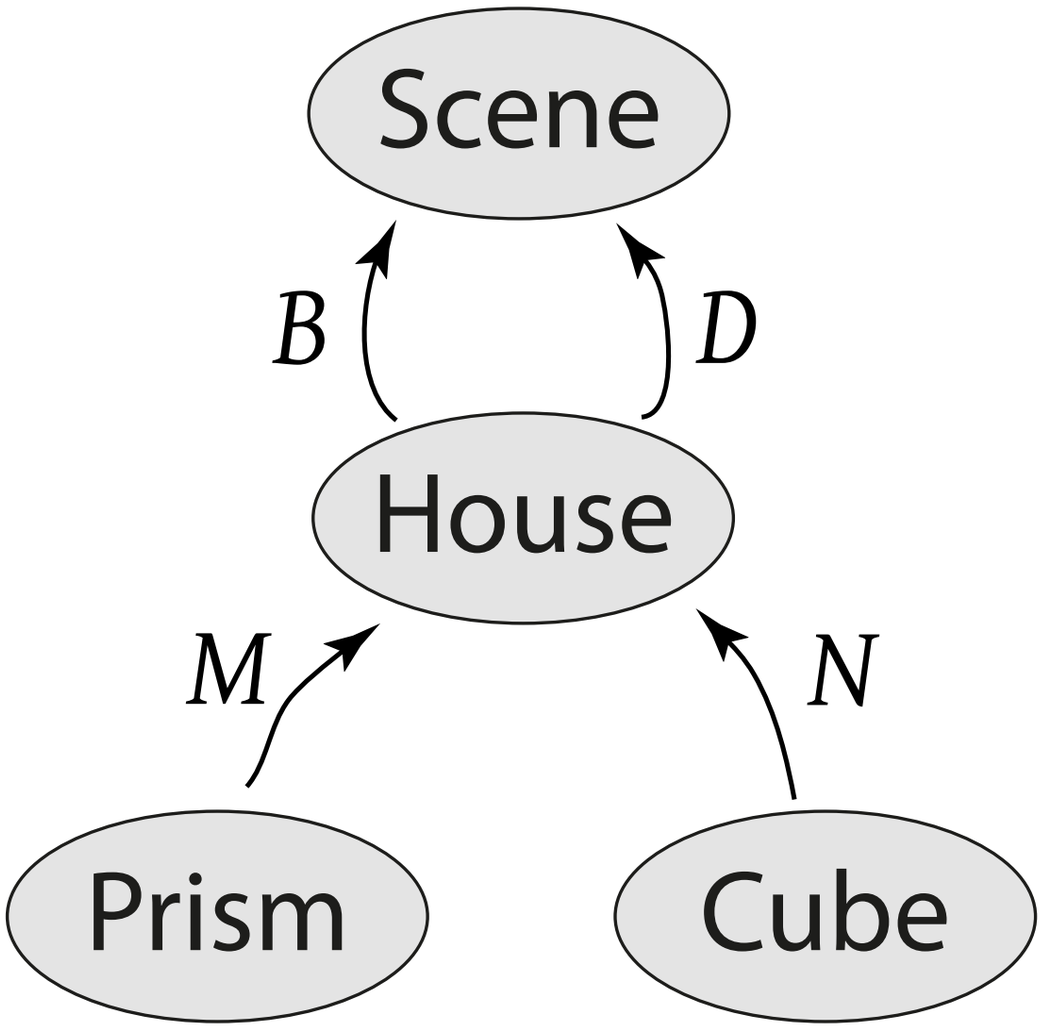



```
procedure Scene():
  House(B)
  House(D)
```

```
procedure House(W):
  Prism(W*M)
  Cube(W*N)
```

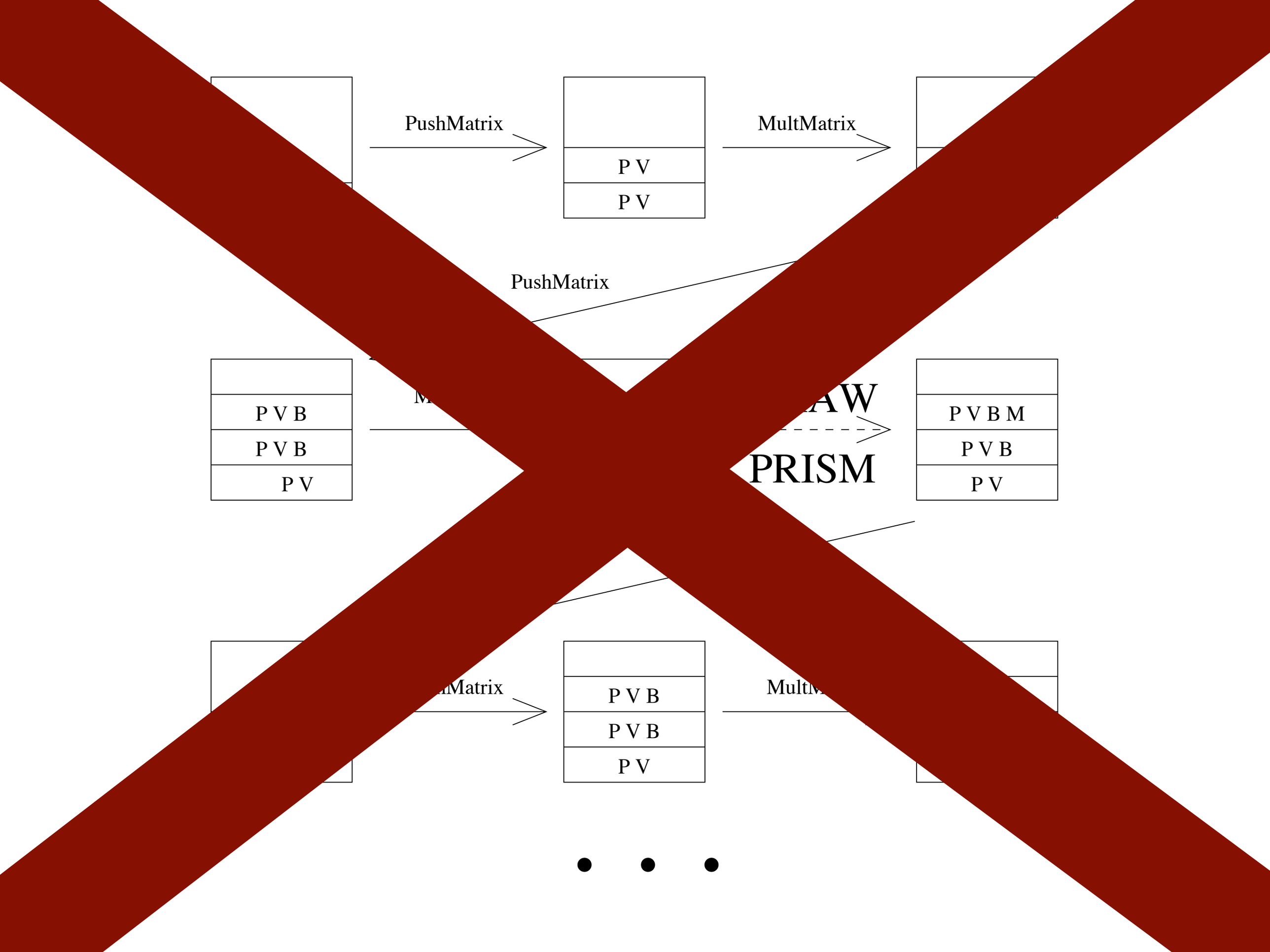
```
procedure Street():
  Matrix T
  for i = 1…10:
    House( T )
    T = Translate(10,0,0) * T
```

```
procedure Scene()
  House(B)
  House(D)

procedure House(W):
  Prism(W*M);
  Cube(W*N);

procedure Prism(W):
  setUniform(W)
  drawPrismMesh()

procedure Cube(W):
  setUniform(W)
  drawCubeMesh()
```



```
dure Scene():
  trix(P);
  (V);
Pushm
  MultMa
  House();
PopMatrix();
PushMatrix();
  MultMatrix(
  House();
PopMat
```

```
procedure H
  PushMat
    Mul          (M);
                 ix();
                 Matrix();
    MultMatrix(N);
    Cube();
                 Matrix();

p                Cube():
  ...

procedure
  ...
```


PushMatrix
MultMatrix
P V
P V
PushMatrix
P V B
P V B
P V
P V B M
P V B
P V
PRISM
P V B
P V B
P V

```
class Node:
procedure traverse(W)
```

```
class GroupNode extends Node:
  procedure traverse(W)

  Matrix T
  vector<Node> children
```

```
procedure GroupNode::traverse(W):
    for child in children:
      child.traverse(W*T)
```

```
class GeometryNode extends Node:
  procedure traverse(W)

  Material mat
  Primitive prim
```

```
procedure GeometryNode::traverse(W):
  setUniform( W )
  setMaterial( mat )
  drawMesh( prim )
```

A3/A4 House Code

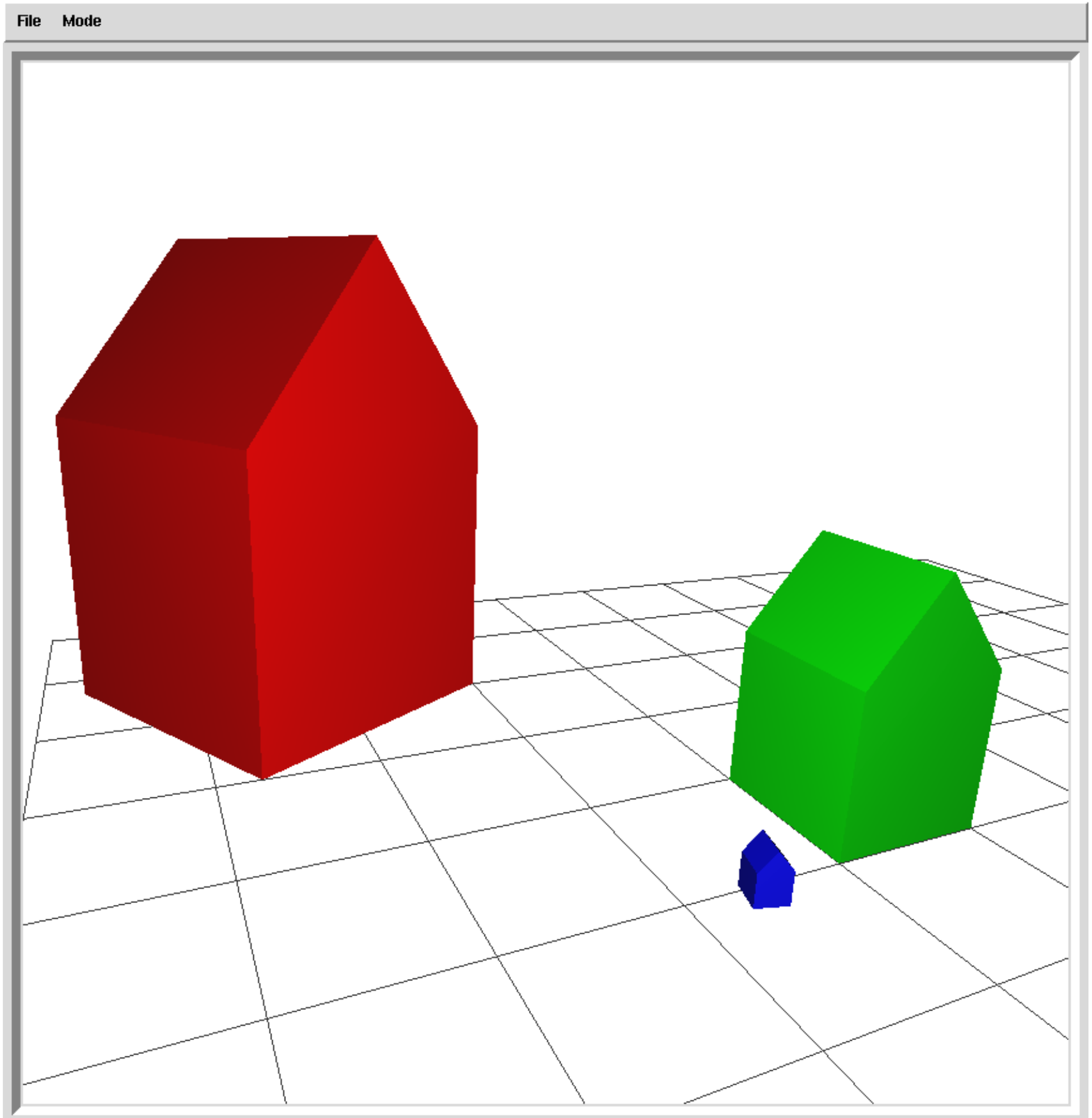
```
scene = gr.transform( 'scene' )

# Assume we have predefined materials 'red', 'green' and 'blue'

house = gr.node( 'house' )
prism = gr.polyhedron( 'prism', ... )

roof = gr.node( 'roof' )
roof:add_child( prism )
roof:translate( 0,1,0 )
house:add_child( roof )
frame = gr.cube( 'frame' )
house:add_child( frame )

farmhouse = gr.node( 'farmhouse' )
farmhouse:add_child( house )
farmhouse:set_material( green )
```

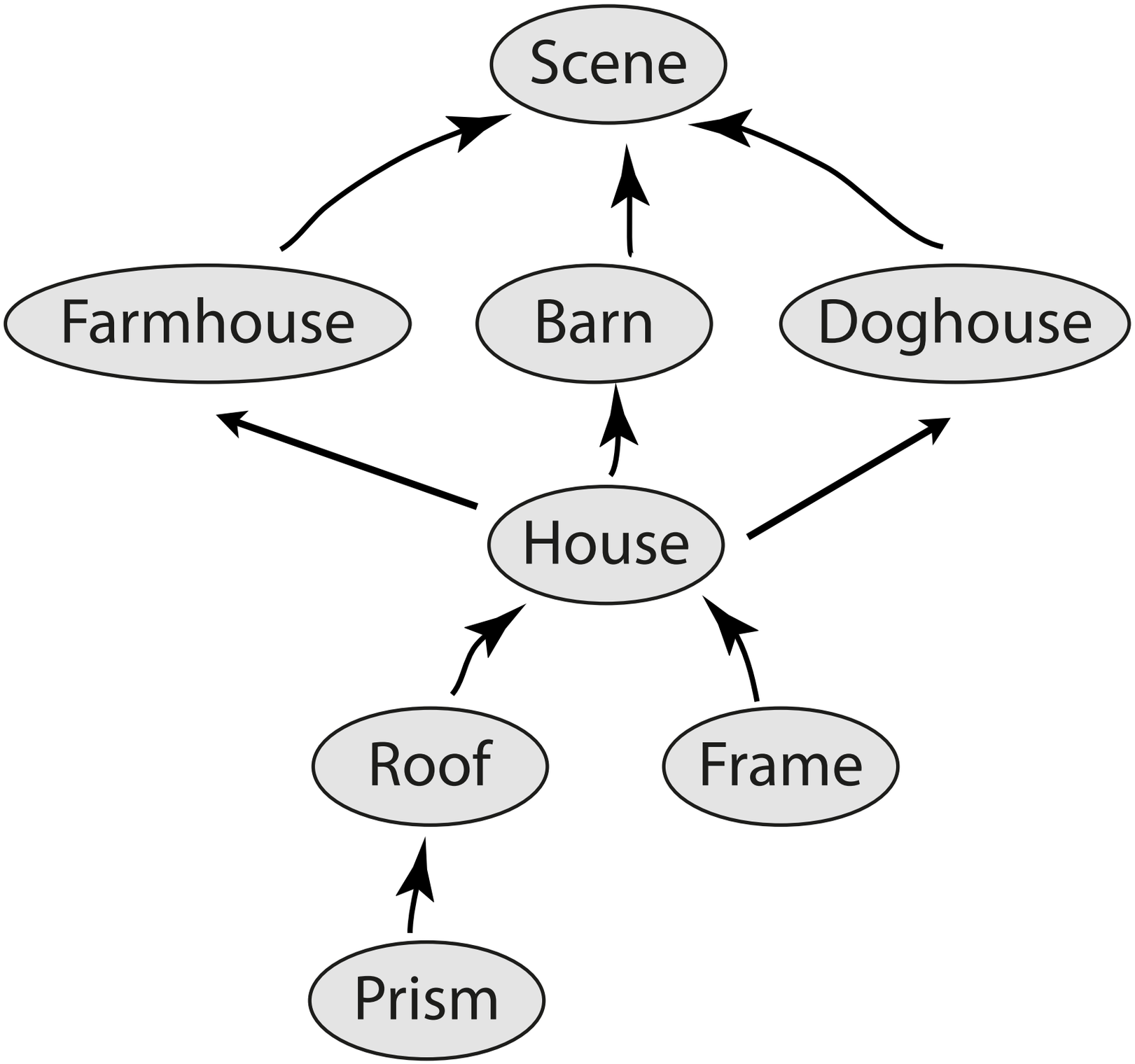
(continued on next slide)

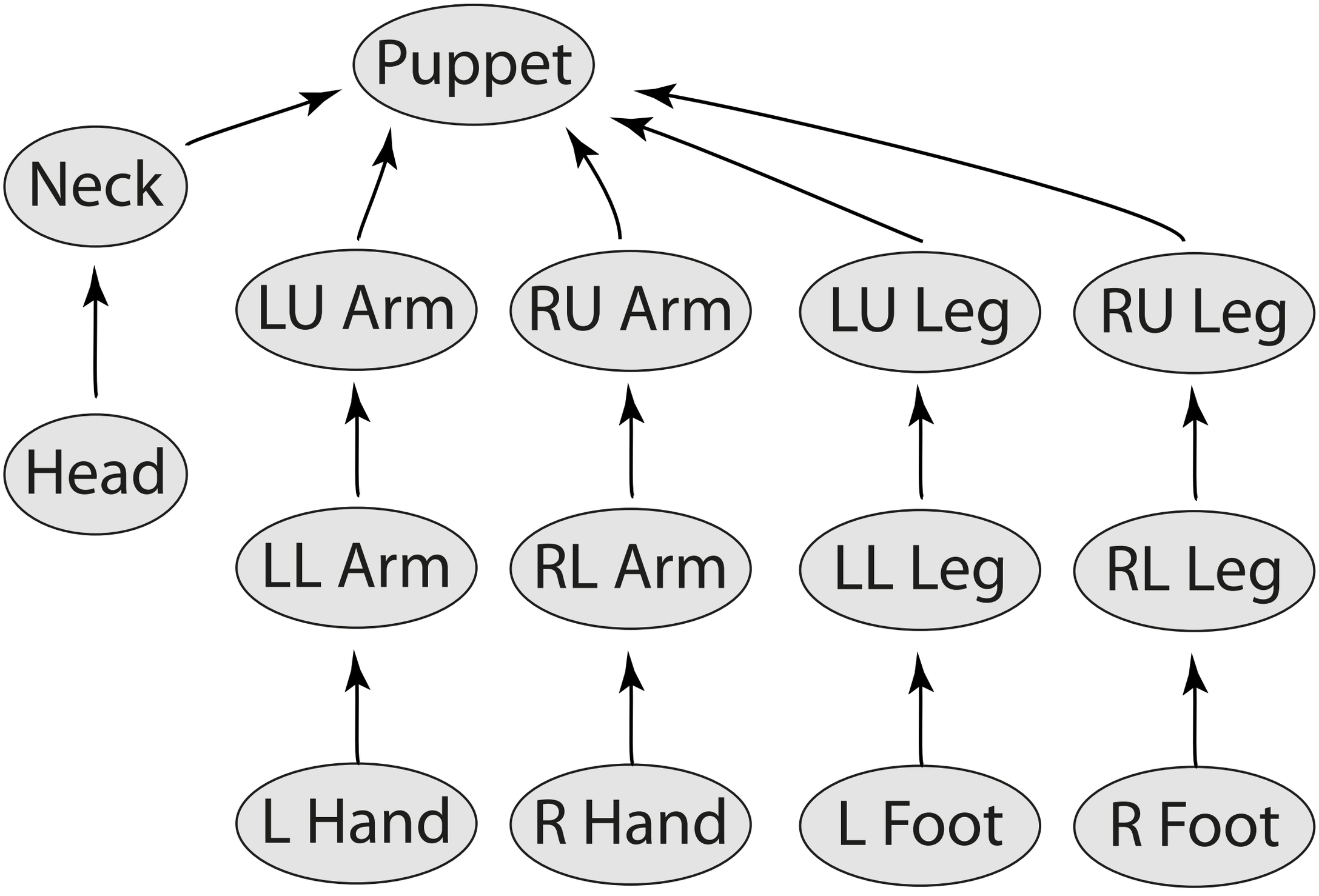
(continuation from previous slide)

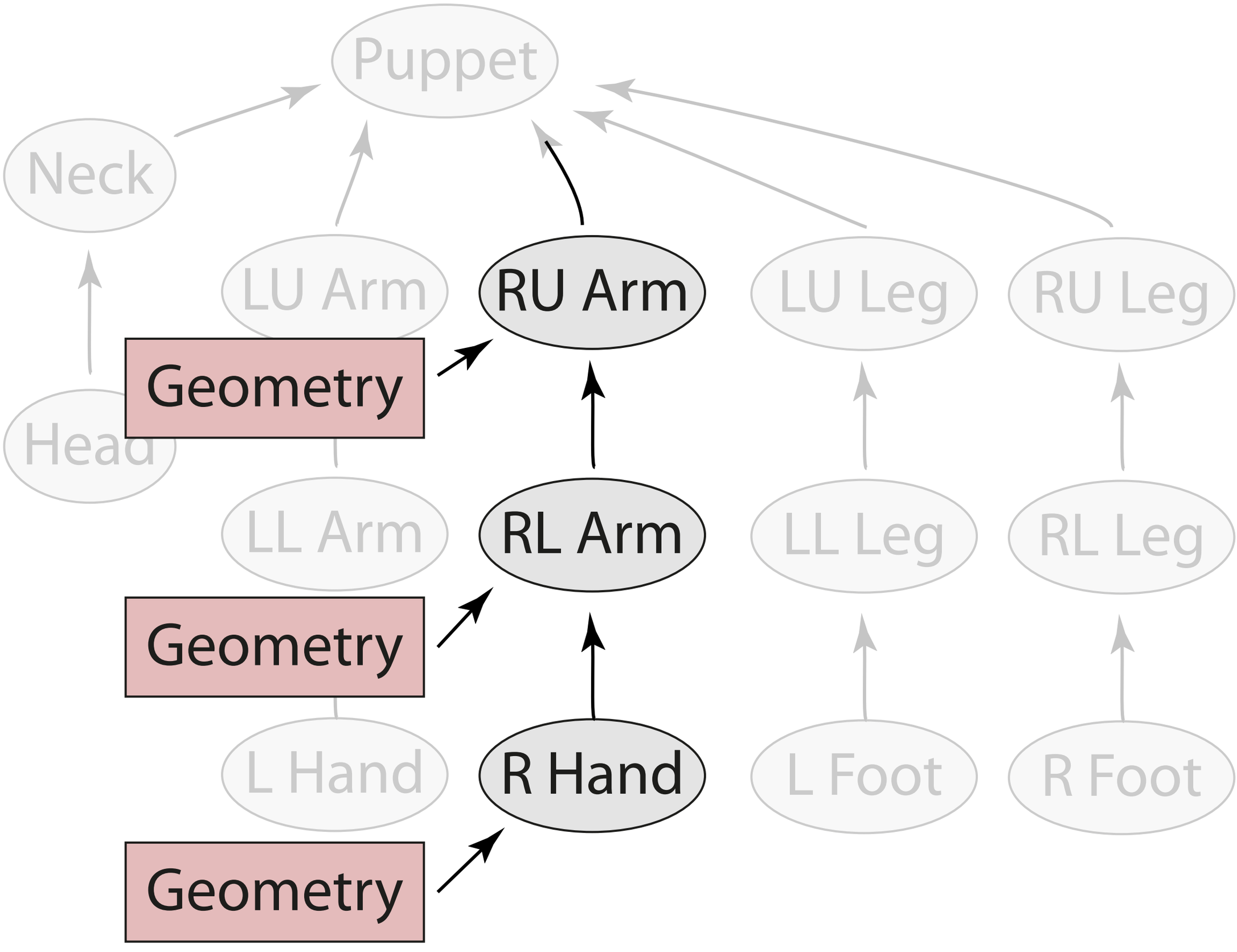
```
barn = gr.node( 'barn' )
barn:add_child( house )
barn:set_material( red )
barn:translate( 2,0,3 ) ; barn:scale( 2,2,2 ) ; barn:rotate( 'Y', 30 )

doghouse = gr.node( 'doghouse' )
doghouse:add_child( house )
doghouse:set_material( blue )
doghouse:translate(1.5, 0, -0.2 ) ; doghouse:scale( 0.2,0.2,0.2 )
    doghouse:rotate( 'Y', -15 ) )

scene:add_child( farmhouse )
scene:add_child( barn )
scene:add_child( doghouse )

return scene
```

File Mode




Scene
Farmhouse
Barn
Doghouse
House
Roof
Frame
Prism


Puppet
Neck
LU Arm
RU Arm
LU Leg
RU Leg
Head
LL Arm
RL Arm
LL Leg
RL Leg
L Hand
R Hand
L Foot
R Foot


Puppet
Neck
LU Arm
RU Arm
LU Leg
RU Leg
Geometry
Head
LL Arm
RL Arm
LL Leg
RL Leg
Geometry
L Hand
R Hand
L Foot
R Foot
Geometry