Iconic Drawing of Scenegraph Structure

Christoph Fünfzig

Computer Graphics Group, Braunschweig University of Technology,
Mühlenpfordtstr. 23, 38106 Braunschweig, Germany,
c.fuenfzig@tu-bs.de,
Webpage: http://www.cg.cs.tu-bs.de

Abstract. In this paper we consider graph drawing for the application of scenegraphs as encoded by VRML97/X3D. As these fileformats store the structure and attributes of scenes in a directed acyclic graph, the techniques for drawing rooted trees are suitable but require much drawing space. Therefore we devised a smart, interactive drawing, which uses a detailed view of the current point of interest and an iconic view of subgraphs besides the current point of interest. This interactive method balances detail and overview much better than classic techniques to cope with limited space like fisheye views. Concerning implementation it is easy to implement with a small lines-of-code count (in Java).

1 Introduction

In computer graphics the scenegraph is an established concept to define and represent scene content [8]. Somewhat simplified a three-dimensional scene consists of geometric primitives (connected triangles, quadrangles and planar polygons, freeform surfaces, etc), which are assigned appearances (coefficients for the Phong illumination model, texture maps, etc). For logical grouping there are also possibilities to group and to transform existing geometric primitives in the scene. Now, in the scenegraph the nodes are organized in this object-oriented way, which can also be used for rendering with low-level libraries like OpenGL [9]. The scene fileformats VRML97/X3D also use the concept of the scenegraph to store the scene. With a modelling tool the user can edit the scene content by a tool specific interface and store it in a proprietary fileformat. Finally the user can export and distribute it in the fileformat VRML97/X3D for rendering. Usually always there are still small things to change even in the VRML97/X3D file, which is a large text file containing the scenegraph structure in serial form. In practice it is tedious to explain the scene structure just with the VRML97/X3D file. For the dynamic behaviour a step-by-step simulator application would be very helpful, which also requires a mechanism to draw the scene nodes involved.

In this paper we consider a graph drawing application, which reads in VRML97 /X3D files and draws the scenegraph structure with the node attributes (section 2). To cope with the size of the scenegraphs it employs detailed drawing at the current point of interest and an iconic drawing (section 4) of subgraphs besides the current point of interest in a new form.

SOFSEM 2005 Communications, 31st Annual Conference on Current Trends in Theory and Practice of Informatics, pp. 31-40
ISBN 80-969255-4-7

2 Scenegraph Structure as encoded by VRML97/X3D

A scene described by VRML97[10] is a text file, which consists of nodes of roughly 60 different types. There are inner nodes like group nodes, e.g., Group, Transform, LOD, Switch. Leaf nodes describe scene content like graphical leaf nodes, e.g., Box, Cylinder, IndexedFaceSet, Appearance, as well as non-graphical leaf nodes, e.g., Sensor, Interpolator, Sound.

A node can be assigned a name with the DEF statement, so that it can be reused or referenced with the USE statement at a different location. With this mechanism arbitrary graphs can be constructed, which should be directed acyclic (a node is not its own ancestor) in order to be displayable. A second mechanism used for defining dynamic behaviour is called ROUTE. A field of a named node declared as eventOut can be connected to an eventIn field (see Figure 1). A subset of fields declared as exposedField can be used as input or output of ROUTEs.

Fig. 1. Node with data fields, declared as eventIn, eventOut or exposedField for ROUTE connections.

The semantics of event routing is precisely defined by the standard. In every timestep the renderer checks for events generated by nodes and propagates them along ROUTEs in an event cascade. To prevent inconsistencies the fan-in of ROUTEs is restricted to one, whereas the fan-out is unrestricted.

For VRML97 there exists an XML-encoding X3D[11] which is extensible by profiles and easy to process due to the XML format.

```
#VRML V2.0 utf8
WorldInfo {
title "Mobile"
info [ "This VRML World was created with Spazz3D" ]
}
DEF Background1 Background {
skyAngle [ .31416 1.36136 1.6057 1.88496 2.96706 ]
skyColor [ 1 1 0  0 0 1  .80784 .80784 1  1 1 1  .80784 .80784 1  .42745 .42745 .42745 ]
groundAngle [  ]
groundColor [ 0 0 0 ]
}
DEF dad_GROUND Transform {
children [ DEF GROUND Group {
  children [
...
  ] }
]
}
DEF Sensor2 TouchSensor {
}
DEF dad_Sensor3 Transform {
translation -.1341 2.1289 0
children [ DEF Sensor3 ProximitySensor {
size 20 20 20
} ] } ] } ]
}
DEF Animation2 TimeSensor {
cycleInterval 10
loop FALSE
startTime  -1
}
DEF Animation2_rot0 OrientationInterpolator {
key [ 0 .25 .5 .75 1 ]
keyValue [ 0 1 0 0  0 1 0 1.571  0 1 0 3.142  0 1 0 4.712  0 1 0 6.283 ]
}
DEF Animation2_rot1 OrientationInterpolator {
key [ 0 .25 .5 .75 1 ]
keyValue [ 0 1 0 0  0 1 0 1.571  0 1 0 3.142  0 1 0 4.712  0 1 0 6.283 ]
}
DEF Animation2_rot2 OrientationInterpolator {
key [ 0 .25 .5 .75 1 ]
keyValue [ 0 1 0 0  0 1 0 1.571  0 1 0 3.142  0 1 0 4.712  0 1 0 6.283 ]
}
DEF Animation2_rot3 OrientationInterpolator {
key [ 0 .25 .5 .75 1 ]
keyValue [ 0 1 0 0  0 1 0 1.571  0 1 0 3.142  0 1 0 4.712  0 1 0 6.283 ]
}
DEF Script_Animation2 Script {
field    SFNode tproc     USE Animation2
field    SFBool state     FALSE
field    SFTime pause     0
eventIn  SFTime startAt0
eventIn  SFTime stop
eventOut SFTime triggerStart
eventOut SFTime triggerStop
eventOut SFTime triggerStartInt
eventIn  SFBool activated
eventIn  SFTime cycle
directOutput TRUE
url "vrmlscript:
function activated( active, curtime ) {
  if (active) { triggerStart = curtime; state = TRUE; }
  else { triggerStop = curtime; state = FALSE; }
}
function cycle( curtime ) {
  triggerStartInt = curtime;
}
function startAt0( stime ) {
  state = TRUE; tproc.set_startTime = stime; tproc.enabled = TRUE; pause = 0;
}
function stop( stime ) {
  if (state) {
    state = FALSE; pause = stime - tproc.startTime_changed; tproc.enabled = FALSE;
  }
}
"
}
ROUTE Sensor2.touchTime TO Script_Animation2.startAt0
ROUTE Sensor3.enterTime TO Script_Animation2.startAt0
ROUTE Sensor3.exitTime TO Script_Animation2.stop
ROUTE Animation2.fraction_changed TO Animation2_rot0.set_fraction
ROUTE Animation2_rot0.value_changed TO dad_Group1.set_rotation
ROUTE Animation2.fraction_changed TO Animation2_rot1.set_fraction
ROUTE Animation2_rot1.value_changed TO dad_Group2.set_rotation
ROUTE Animation2.fraction_changed TO Animation2_rot2.set_fraction
ROUTE Animation2_rot2.value_changed TO dad_Group3.set_rotation
ROUTE Animation2.fraction_changed TO Animation2_rot3.set_fraction
ROUTE Animation2_rot3.value_changed TO dad_Group4.set_rotation
ROUTE Animation2.isActive TO Script_Animation2.activated
ROUTE Animation2.cycleTime TO Script_Animation2.cycle
```

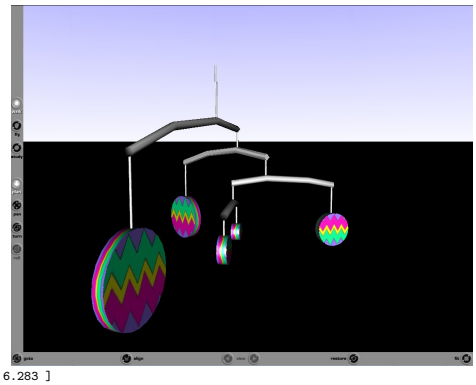

Fig. 2. VRML97 scene of a mobile, consisting of 3 sensors, 4 interpolators and 1 script.

3 Previous Work

The scenegraph is a directed acyclic graph with a single root node. If we draw a copy of the node, which references a node (named with DEF) somewhere else, then we can employ any drawing algorithm for trees. The drawing of trees is a well-known field [4]. In their seminal work [4] Tilford and Reingold state a set of requirements for tidy layered drawings:

- Nodes at the same depth should lie on parallel straight lines.
- The parent node should be centered above the drawings of its children.
- A tree and its mirrored tree should have mirrored drawings. A subtree should have the same drawing independently of its position in the complete tree.

The drawing algorithm of Tilford and Reingold produces tidy drawings, but it cannot be used on restricted space. The radial drawing of trees accounts to some extend for the fact that more space is required in deeper levels of the tree and better fills quadrangular drawing regions. But in principle it suffers from the same problems like vertically layered drawings. Space transformations, usually called fisheye drawings, have been proposed [5], [6] for large labeled graphs. An abstract illustration of this technique on a regular grid is shown in Figure 3. By computing the magnification parameter d appropriately we could achieve a detailed drawing of the current point of interest and a condensed drawing of the remaining. But additionally to the drawing of the tree we have to draw ROUTE arrows between widely separated nodes. Although [6] gives a generalization of interesting regions to convex polygons, the use of this approach here is infeasible because of a difficult choice of the interesting region.

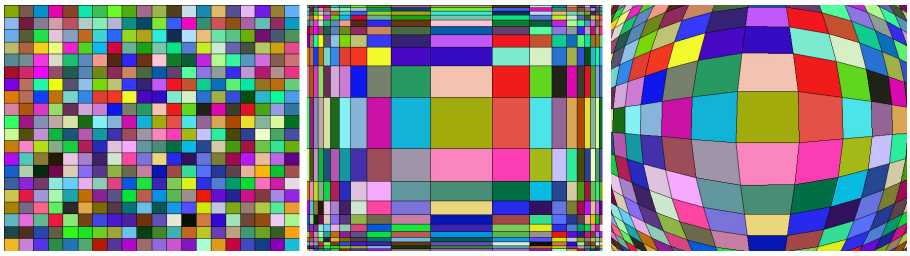

Fig. 3. 20×20 grid (left) shown in two fisheye views with parameter $d = 4$: cartesian transform (middle) and polar transform (right). These images were generated by a Java applet of Christopher C. Yang, The Chinese University of Hong Kong [3].

Recently several interactive techniques [7], [2] for drawing large hierarchies have been proposed. These techniques adapt to a user selected focus point but ideally without loosing the larger context, like with simple Zoom and Pan approaches. Here the categories which are connected with our approach are Semantic Zooming (by drawing the current point of interest and the DEF/USE and ROUTE connections with full detail) and Clustering (by drawing all remaining nodes as subtree icons).

4 Iconic Drawing

For drawing the scenegraph we have devised an interactive technique, which uses a detailed region around the currently selected node and triangle icons for subgraphs. Such icons have also been used in the textbook [1, Chapter 20. Binomial Heaps] to illustrate inductive graph definitions. Inside the triangle icons just the subtrees induced by DEF/USE and ROUTE connections are drawn.

The method draws the path to the currently selected node as a vertical node chain, which ends in the selected node with its children (Figure 4). All the chain nodes and the current node's children are active and can be used for navigation up–down and left–right, respectively. The subgraphs below the children are sketched as triangles with fixed width and a height proportional to the number of contained nodes. At each level of the node chain the left siblings and the right siblings are outlined as a box with a triangle to the left and to the right of the chain, respectively. The box contains the number of sibling nodes and the triangle contains the number of nodes in the subtree as a label. The triangle size is computed as follows

$$\begin{aligned}\text{width} &= \max(\text{MaxWidth} \cdot \frac{\text{Number of nodes in subtree}}{\text{Max Number of nodes in complete tree}}, \text{MinWidth})\\ \text{height} &= \text{Height of Root Quadrangle} + \text{Height for Label} +\\ &\quad \text{Max Length of Path to referenced nodes} \cdot \text{Height for node}\end{aligned}$$

The layout algorithm for the new method is especially simple, as there are three horizontally disjoint regions: Region with left subtree icons, vertical node chain to the currently selected node, and region with right subtree icons (see Figure 5). For the regions with subtree icons in our current implementation the icons are placed horizontally non-overlapping from left to right, where the vertical positions are given by a constant level width.

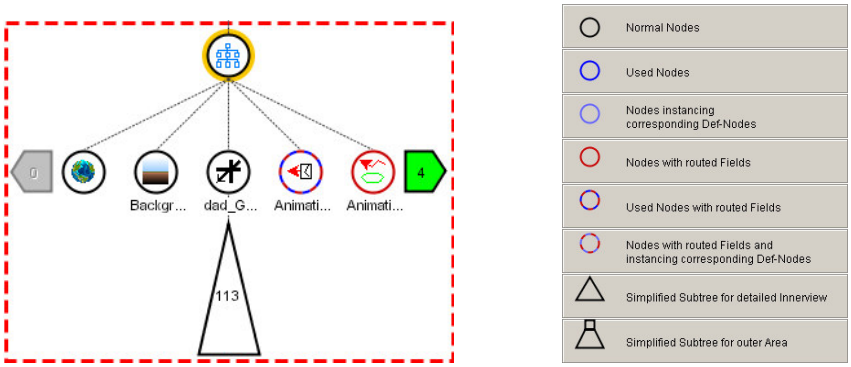

Fig. 4. Elements used for the iconic drawing of the scenegraph.

The DEF/USE and ROUTE relationships for the current node are drawn as arrows in an overlay. Usually the number of arrows is bounded by a small constant. Inside the triangle icons the subtree induced by referenced nodes is drawn with a modified Tilford and Reingold algorithm (see Figure 7).

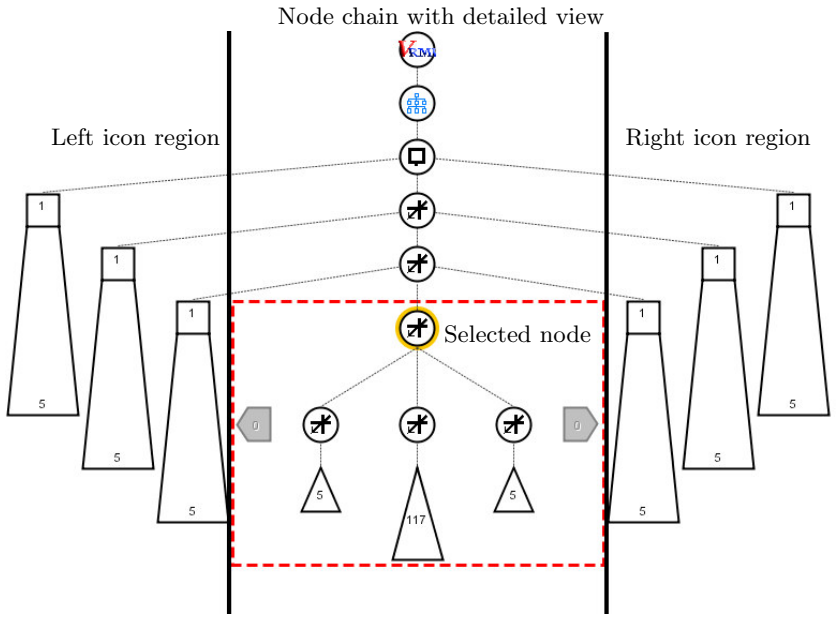

Fig. 5. Three vertical regions: Region with left subtree icons, vertical node chain to the selected node, and region with right subtree icons.

5 Conclusion and Results

We have implemented the iconic drawing technique in Java for scenegraphs as encoded by VRML97/X3D. The scenegraph is stored as a JTree after parsing the VRML97/X3D file. The drawing kernel consists of a group of classes for tree drawing and another group for the DEF/USE and ROUTE overlay in addition to the Java base (see Figure 6). In total this kernel consists of 2745 lines of code.

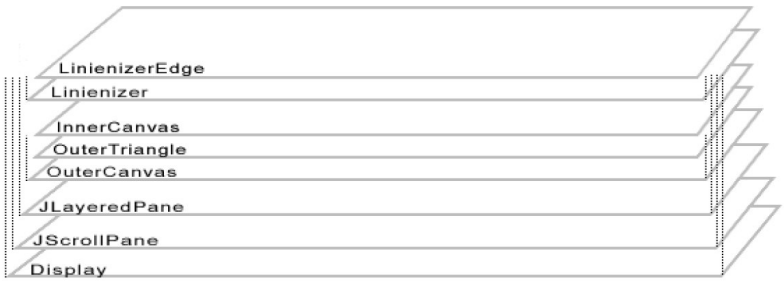

Fig. 6. Java classes used in the implementation of the layout algorithm. OuterCanvas and related classes contain the tree drawing. Linienizer and LinienizerEdge take care for the DEF/USE and ROUTE overlay.

In Figure 9 we show an image sequence generated for the mobile scene of Figure 2. There you can see the subtree defining the moving geometry, which consists of the suspensions made of cylinders, a freeform branch, and the textured cylinder decoration. As the model geometry consists of four levels moving

indepently, there are four groups with quite the same contents, referenced by ROUTEs from the orientation interpolators. The script node in the scene handles the starting of the time sensor for the animation sequence due to mouse clicks or viewer proximity.

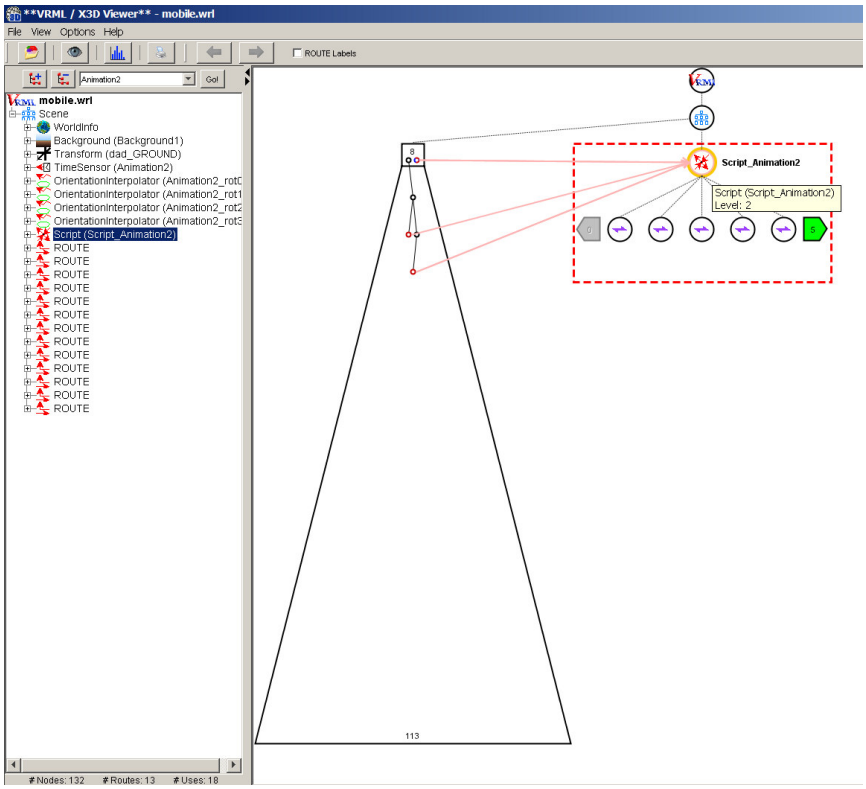

Fig. 7. Application for drawing the scenegraph structure as encoded in a VRML97/X3D file. The right side shows the script node (in the mobile scene), which has 3 ROUTE connections to other nodes.

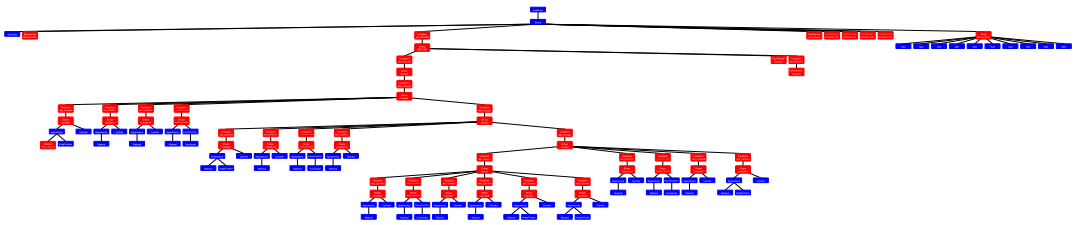

Fig. 8. Layered graph drawing of the mobile scene generated by Tilford and Reingold algorithm.

The mobile scene is rather small and consists of 132 nodes in 18 tree levels. For comparison Figure 8 shows the result of the Tilford and Reingold algorithm on the complete mobile scene. Here the overlay showing the DEF/USE and ROUTE connections is omitted.

In conclusion we devised a smart, interactive drawing, which uses a detailed view of the current point of interest and an iconic view of subgraphs besides the current point of interest. This interactive method makes it possible to draw large scenegraphs in terms of node count on twodimensional drawing space. The method provides just the interesting details like DEF/USE and ROUTE connections, of which there are usually only a small number. We think that the smart drawing is superior to the full drawing of the scenegraph in terms of the time a user needs to perceive and understand the scene structure. Nevertheless our implementation can be improved further in some details like smarter drawing of multiple ROUTE connections to/from the same node, and inclusion of script field names. Currently interaction is possible only with nodes on the path to the currently selected node and its children nodes. In our experience it would be beneficial to be able to expand subtrees in the detailed region, and to have some navigation shortcuts to subtrees represented as subtree icons. Further work should incorporate a global, smooth up–down and left–right navigation as with condensed full tree drawings.

Acknowledgements. We would like to thank Benjamin Budgenhagen, Patrick Helmholz, Philipp Kluge, Sven Nemeth, Henrik Peters, Frederik Suhr and Timo Winkelvos for implementing the application **VRML97/X3D Viewer** in Java. The nice node icons were taken from X3DEdit of Don Brutzman. The mobile scene, we use for demonstration purposes, was done by Phillip Hansel and generously donated to the world wide web.

References

1. Cormen T., Leiserson C.E., Rivest R.L.: Introduction to Algorithms. MIT Press, (1990).
2. Herman I., Melancon G., Marshall M.S.: Graph Visualization and Navigation in Information Visualization: A Survey. **IEEE Transactions on Visualization and Computer Graphics 6(1)** (2000). 24–43.
3. Yang C.C., Chen H., Hong K.: Visualization Tools for Self-Organizing Maps. **Proceedings of the ACM International Conference on Digital Libraries, Berkeley** (August 1999).
4. Reingold Edward S., Tilford, John S.: Tidier drawings of trees. **IEEE Transactions on Software Engineering SE-7(2)** (March 1981). 223-228
5. Kaugars K., Reinfelds J., Brazma A.: A Simple Algorithm for Drawing Large Graphs on Small Screens. Springer Verlag **LNCS 1027 (Proceedings of Graph Drawing, Passau)** (1995). 278–281.
6. Formella A., Keller, J.: Generalized Fisheye Views of Graphs. Springer Verlag **LNCS 1027 (Proceedings of Graph Drawing, Passau)** (1995). 242–253.
7. Card S.K., Nation D.: Degree-of-Interest Trees: A Component of an Attention-Reactive User Interface. **Advanced Visual Interfaces** (2002), Trento, Italy.

8. Reiners, D.: Scene Graph Rendering **Proceedings of IEEE Virtual Environments** (2002).
9. Segal M., Akeley K.: The OpenGL Graphics System: A Specification SGI Inc. (1997).
10. Web3D: VRML97 Specification, Part 1 - Language Specification, Part 2 - External Authoring Interface. ISO Standard **ISO/IEC 14772** (1997).
11. Web3D: X3D Specification. ISO Standard **ISO/IEC 14775:200x-14777:200x** (2003).

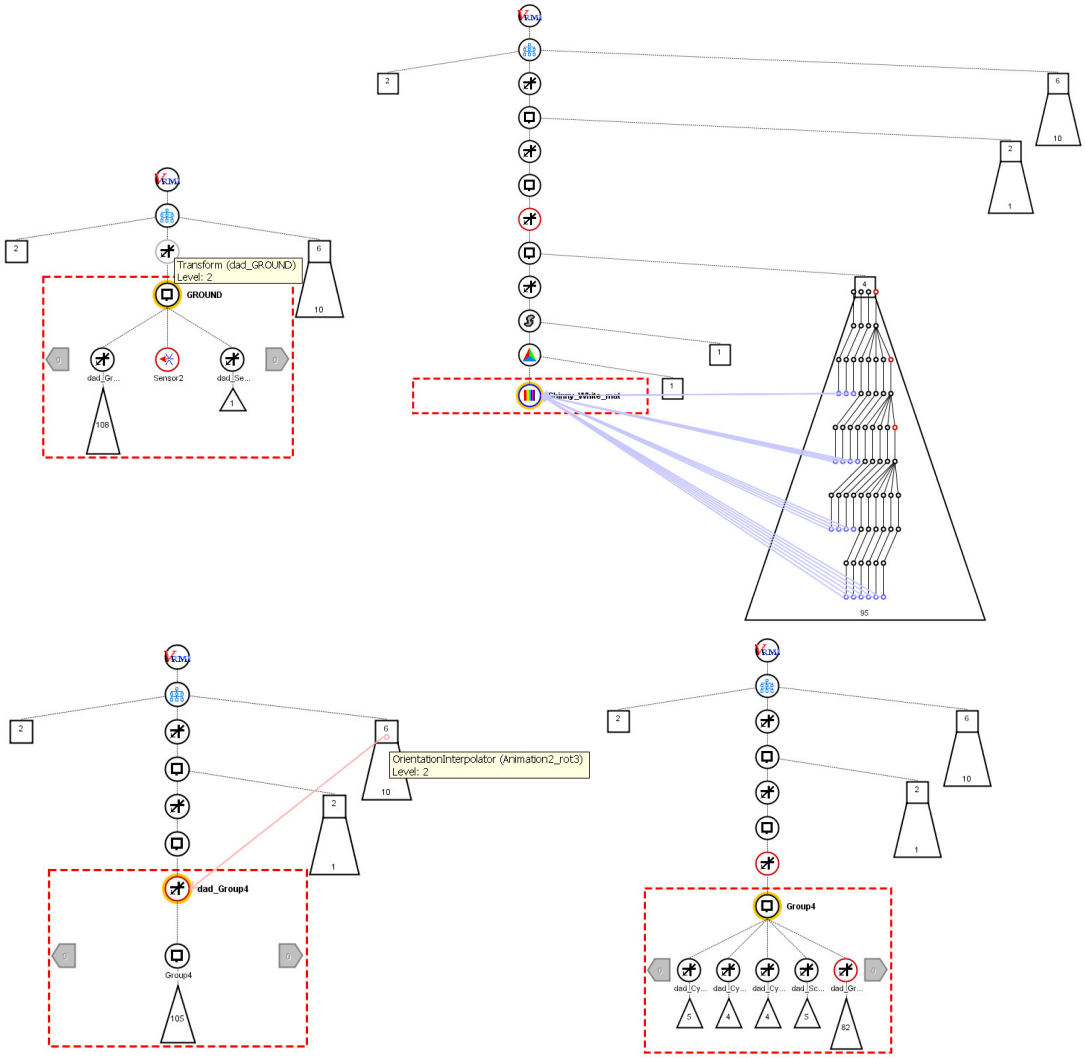

Fig. 9. Iconic drawing of the subtree defining the geometry of the mobile scene.

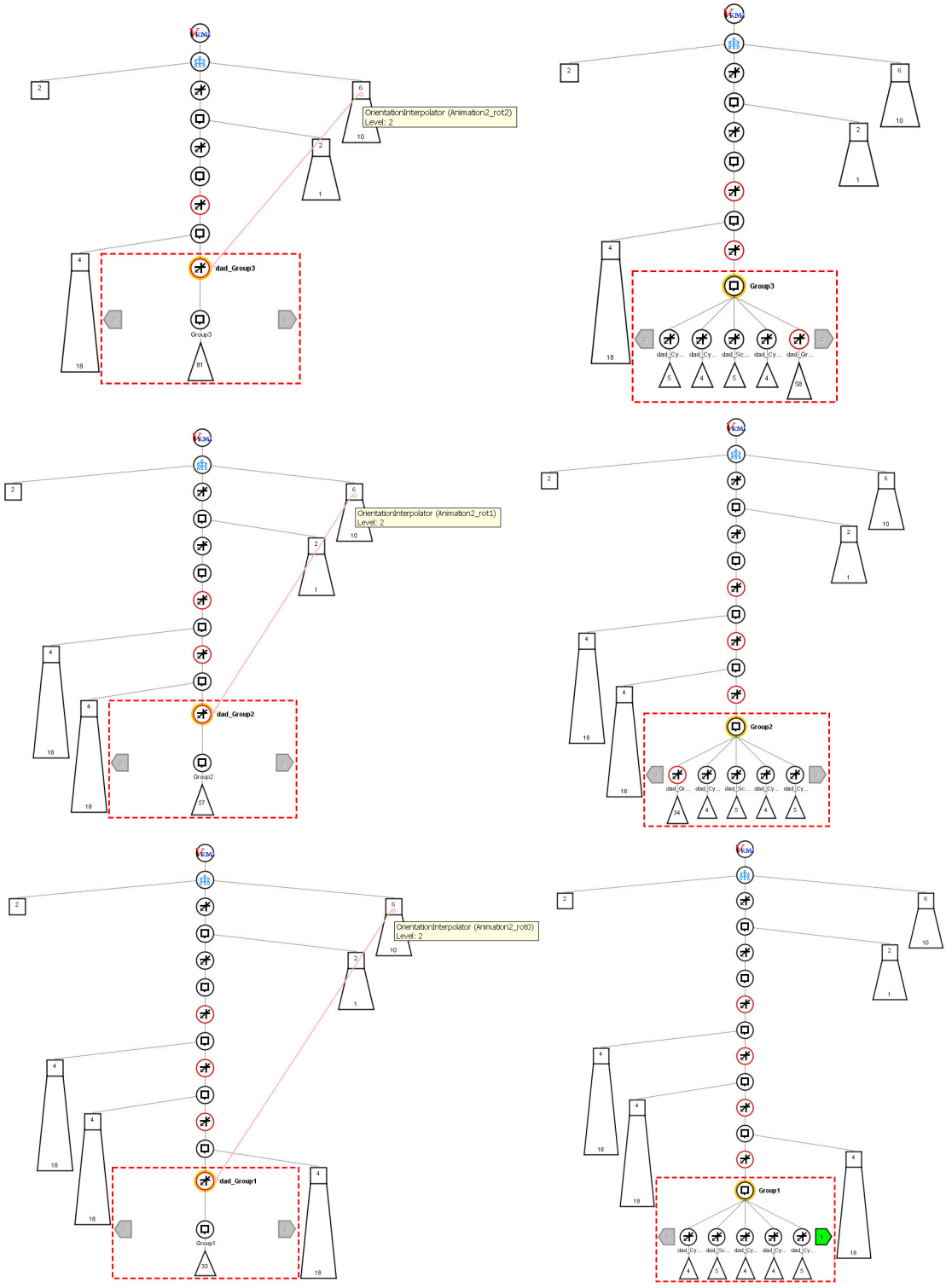

Fig. 9. Iconic drawing of the subtree defining the geometry of the mobile scene (continued).