A Tree Glyph for the Fresco TM. Toolkit

David M Sternlicht
MIT X Consortium
545 Technology Square
Cambridge, MA 02139

February 23, 1993

Abstract

This paper discuses the design and implementation of a tree glyph in which any Fresco subclass of Glyph can be imbedded.

Electronic Address: dave@expo.lcs.mit.edu

1 Introduction

1.1 Motivation

Trees play a fundamental role in the art of computer science. They form the basis of a number of sorting algorithms [1]. They are a basic part of the data structures' world largely because of their amenability to processing by recursive techniques. [2]. Finally, since they are a pleasing part of the natural world, mathematics have been used to simulate and display them in all their natural wonder [3]. The author felt that a class which managed tree's was an important missing part of the Fresco library and that writing such a class would provide a large enough programming project to further his working knowledge of object oriented design and implementation in the C++/GUI environment.

1.2 Implementation

The class interface to Tree is as follows:

```
typedef enum { TopDown, LeftToRight } Orientation;

class Tree : public Glyph, public Adjustable, public Observer {
public:
  Tree(Session*);
  ~Tree();

  // Tree API Functions
  void insert_node(Glyph *parent, Glyph* child);
  void force_layout(Orientation);

  // Glyph, Adjustable, and Observer delegation member functions.
  // ...

private:
  TreeImpl* timpl_; // you can't touch this :-)
};
```

This class is similar to Doug Young's Xt Tree Widgit [5]. Besides the constructor and destructor, there are only two unique public methods. Tree::insert_node() embeds a glyph in the tree by taking a pointer to it as well as its parent. Tree::force_layout() forces the layout process in the specified orientation.

One major goal in designing this class was to hide as much of the implementation as possible. Unlike the OMG's IDL [1], one is forced to specify implementation in a C++

[1]The Object Management Groups's Interface Description Language.

class declaration. There is no motivation for the programmer to expose implementation details such as how the tree performs layout. He may want to change it in the future and not force the user of his class to recompile [2]. This was done by creating a class called TreeImpl which only is only declared as a forward reference in the header file `tree.h`. Unfortunately it proved too difficult to hide certain methods of the Glyph, Adjustable, and Observer class from the API. `class Tree` must inherit from Glyph so that it can participate in the Fresco layout protocol. It must be an Adjustable and an Observer so that it can be adjusted by panners and scrollbars. Tree's Glyph, Adjustable and Observer methods simply call the TreeImpl's methods. One could envision the removal of these parent methods from tree.h by overloading the dereference operator, and using a scheme like Coplien's En Masse Delegation with Envelope classes [4]. However this proved unachievable.

The TreeImpl class will of course change without warning. The code has been based on the ruiScene class implemented by Denis Bohm and revised by Joerg Ringel. It maintains a list of pointers to TreeInfo objects that maintain information on a given Glyph in the tree. Panning is supported by the maintenence of a transformation matrix object. Every TreeInfo object maintains a particular glyphs Allocation and Requisition on the canvas. Also included are links to the TreeInfo objects corresponding to parent and children. Tree.c is a little less than 1300 lines of code. The TreeImpl class declares about 75 methods. By contrast the tree class only defines 2 unique public methods.

The tree class code can be ftp'ed from export.lcs.mit.edu as contrib/ftree.tar.Z. This paper is in the file contrib/ftree.ps.Z The code comes with a demonstration program, `sort`, also take from Doug Youngs work. This program displays a binary tree of sorted numbers taken from stdin. The tree class has also been successfully integrated into a Fresco Style browser tool that communicates with Fresco clients using X selections and the editres protocol developed by Chris Peterson.

1.3 Further Work

There are a number of areas that might be worth additional work. The tree class could support zooming through the Adjustable class API. The class's geometry management could also be extended to cause a tree glyph to visually streatch or shrink according to its requisition and the canvases resulting allocation. Lastly, for this class to be useful for class browsers, directed acyclic graphs should be implemented to graphically display multiple inheritance.

[2]Of course he would have to relink.

References

[1] Knuth, D., Sorting And Searching. Addison-Wesley, 1973.

[2] Horowitz, E. and Sahni S., Fundamentals of Data Structures. Computer Science Press Inc. 1976.

[3] Foley,J.D., and vanDam,A., and Feiner,S.K., and Hughes,J.F., Computer Graphics Principles and Practice, Second Edition, Addison-Wesley, 1990

[4] Coplien, J. O., Advanced C++ Progarmming Styles And Idioms. Addison-Wesley, pp 160-163, 1992

[5] Young, D., X Window Systems Programming And Applications With Xt, Prentice Hall, pp 359-387, 1989.