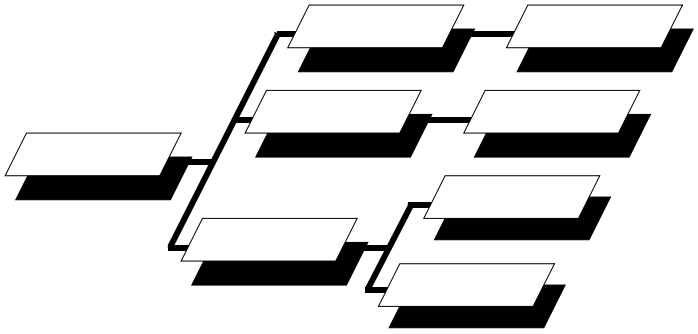
NodeViewer 1.0



Quipus
Mårten Sörliden
Djurgårdsgatan 21A
S-582 29 Linköping
Sweden

Phone +46 13 12 79 79
AppleLink SW1340
Internet SW1340@AppleLink.Apple.com

The Tree Structure Class Library

Introduction

NodeViewer is the first application using the Tree Structure Class Library. The Tree Structure Class Library is an extension to the THINK Class Library. The main idea of the library is to implement a general way to handle information. The information in a document is structured in a tree structure of nodes and each node handle a dynamic number of subnodes.

The main purpose of NodeViewer has been to test and evolve the class library. The second goal has been to make a general browser able to open any document created by applications using the class library. NodeViewer is also a tool which visualize the features of the Tree Structure Class Library. It will give you an idea of how to use the class library in a real application.

The Node

The Node is a piece of information which at least contains a specific header information. The nodes you can create in NodeViewer with the command New Subnode are called dummy nodes. They only contains the header information and have got the type NODE. However, you can import scraps of type TEXT and PICT through the Clipboard, which are converted to nodes of the corresponding type. These nodes contains the information of the text and the picture in addition to the header information.

Unknown types of nodes are treated as virtual nodes. NodeViewer use this feature since it opens all types of documents containing nodes.

The Node Application

The Node Application is a specialized version of the application in the THINK Class Library. It creates the Node Clipboard and handles memory shortage. In a situation where more memory is needed it will close nodes not in use.

The Node Document

The Node Document is used in the same way as a regular document. The difference is that the information in the document is structured in a hierarchy of nodes. The document communicate with the nodes through the top node. The information is saved by the documents standard methods in a node file.

The Node Clipboard

The node clipboard supports a private scrap. The node clipboard handles TEXT, PICT, and SrpN scraps. The SrpN scrap is a new type of scrap. It contains the information of a set of hierarchies of nodes. To deal with other kinds of scraps or to create other types of nodes when loading a scrap, a subclass must be created. The default way of handling unknown types of nodes is to create virtual nodes.

Features

- Persistent objects
- Any subhierarchy may be treated as a cluster of objects
- Ordered set of subnodes
- Identification of individual objects
- Support of document metaphor by giving the user the control over saving information
- Compatibility with future object types through virtual information handling

- Only visible parts of a document needs to be in primary memory
- Dynamic node size and dynamic number of subnodes
- The Clipboard supports Cut, Copy, and Paste of any information structure
- Support of different versions of object types
- Memory management where unused and unaltered parts are automatically closed at need

NodeViewer Menus

The Apple Menu

About NodeViewer…

This command shows you the copyright information of NodeViewer.

The File Menu

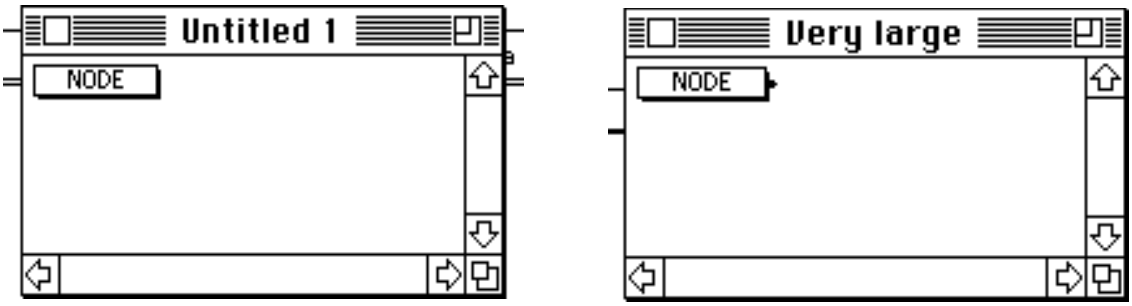
The File menu contains the standard items defined by the resources in the THINK Class Library and proposed by Apple.

New

A new NodeViewer document is created. The document contains one node - the top node. The top node will be the owner of the other nodes in the hierarchy. Each document needs a top node. In NodeViewer the top node is a dummy node.

Open…

NodeViewer will try to open any document. If the information on the file isn't nodes it will tell you that the document is of an unknown type. Only the top node will be loaded into memory when the document is opened. A small arrow on the right side of the node indicates that the node has got subnodes not in primary memory.



A new document and a just opened document.

Close

The Close command lets you close the active window. Clicking in the window's close box does the same thing. If you close a NodeViewer document that has been changed since it was last saved, NodeViewer asks you if you want to save the changes.

Save

This command will save any changed parts of the document. All nodes previously saved on file doesn't have to be in primary memory since the file is modified according to the changes of the document.

Save as…

This command will save the document under another name. In this case all nodes will be loaded into primary memory before they are rewritten on the new file. This will be changed in the future so for instance very large files can be copied as files and the changes will be saved to the new file. The application will then be able to have a small memory partition and still be able to copy large files with this command.

Revert to Saved

This command restores the last-saved version of the document. All nodes in the hierarchy of the document will be disposed and a new top node will be created with information reread from the file.

Page Setup…

This command displays the standard Page Setup dialog.

Print…

This command lets you print the hierarchy of nodes in the current document.

Quit

This command quits NodeViewer.

The Edit Menu

The Edit menu is also a standard menu.

Undo / Redo

NodeViewer use a set of tasks which you are able to Undo and Redo. The tasks are: Cut, Copy, Paste, Clear, Move, and New Subnode.

Cut

This command removes the selected nodes and places them into the Clipboard. All subnodes of the selected nodes are loaded into memory and cut from the document. The new scrap will replace any old scrap. You can not cut the top node since the top node is required by the document.

A special node called the Scrap Node is used as the container of the nodes cut from the document. The node clipboard handle the Scrap Node and converts it to a global scrap of type SrpN when the application is suspended. The clipboard will also convert TEXT nodes and PICT nodes in the scrap node to standard scraps of corresponding types.

Copy

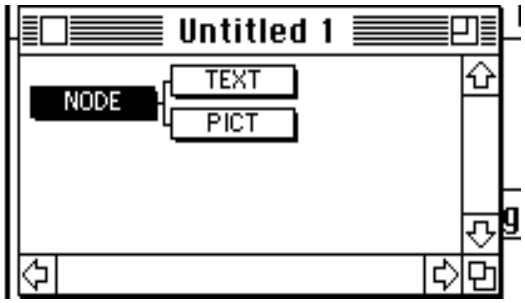
This command copies the selected nodes and places them into the Clipboard. The nodes can be pasted some where else in a node document using the Paste command.

If you make a copy of the top node you will make a copy of the entire document. This would probably not be an appropriate function in a regular application using the Tree Structure Class Library, where most often small pieces of the document are copied and pasted. You should be aware that NodeViewer allows you to create very large scraps. Since nodes only are copied as node objects in primary memory all subnodes of the selected nodes will be loaded into memory.

Paste

This command copies the nodes in the current scrap node of the clipboard and places them as subnodes at the selected node. You are only able to paste if a single node is selected.

You can paste imported information by copying a piece of text or a picture in another application, and select a node in NodeViewer and choose the Paste command.



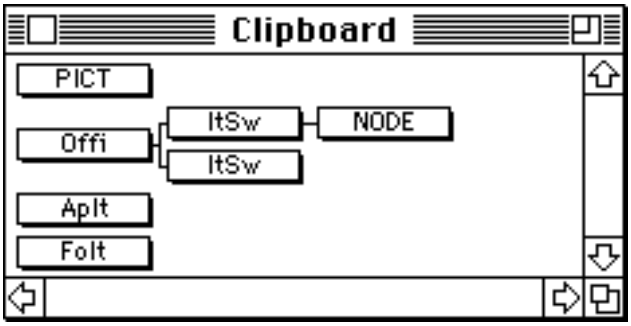
Clear

This command clears the selected nodes without changing the Clipboard. You can not clear the top node since the top node is required by the document.

Show Clipboard / Hide Clipboard

This command shows you the nodes in the Clipboard. All nodes are viewed as connected boxes to show the structure of the information in the clipboard. The scrap node is not visible in the clipboard since the node is not an object that has to be known by the user.

The node clipboard handle global scraps of type TEXT, PICT, and SrpN. The scrap of type SrpN contains the information of all the nodes with the scrap node as the top node. The scrap will create an identical structure of nodes when it becomes converted to a private scrap again.



The Node Menu

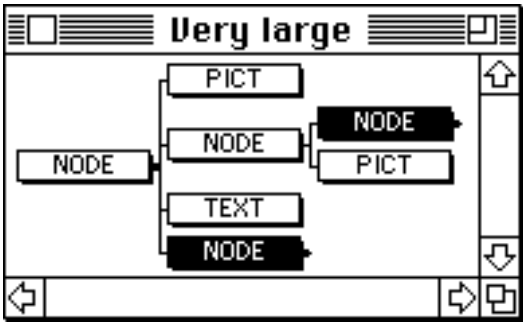
This menu is a unique menu of NodeViewer. It's like a file menu that operates on the nodes in the document instead of documents in the file system.

New Subnode

This command creates a new dummy node as subnode to the selected node. You must only have a single destination node selected. If the selected node doesn't have all nodes in memory they will be loaded before the new subnode is created.

Open Subnodes

This command opens the subnodes just below the selected nodes. The Node Document must have been saved on file and reopened to have any nodes remaining on file. A node with a small arrow on the right side has got more subnodes on file. If the node has got some subnodes in memory but not all, will the line to the right of the node be thicker. You can also double click on a node to open its subnodes.



Close Nodes

This command will close the selected nodes. You can close all nodes except the top node with this command. If any of the selected nodes or any of their siblings has been altered, you are asked if you want to save the document before the nodes are closed. If you don't want to save the document the unsaved and selected nodes will be deleted. Unsaved changes in other parts of the document will still remain.

Get Info

This command will show you the header information of the node. The type of the node is a four letter identifier that should correspond to a subclass of the node class in the application. NodeViewer will make virtual nodes of all types of nodes.

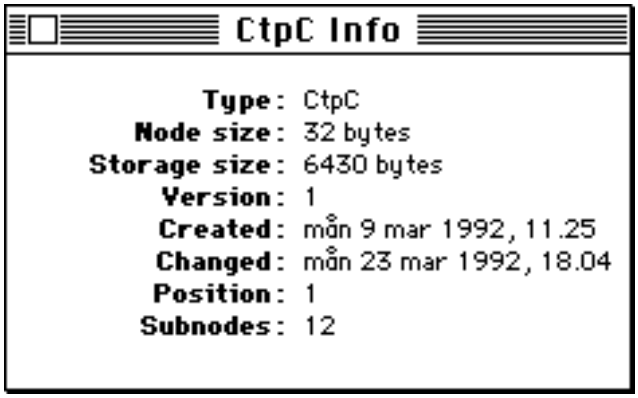
The node size is the size of the node without subnodes when the node is saved on file or in a handle. This size is at least 32 bytes, the size of the header information. The storage size is the size of the node including the subnodes. The storage size of the top node is the size of the file.

The version is the type version of the node. The default size of a node might vary according to the version.

The dates when the node was created and changed are stored. This information makes it possible to treat the node hierarchy as a file-system of its own.

The position of a node is the index of the node among its siblings. A sole node has got position one.

The number of subnodes is also stored in the header information.


CtpC Info
Type: CtpC
Node size: 32 bytes
Storage size: 6430 bytes
Version: 1
Created: mån 9 mar 1992, 11.25
Changed: mån 23 mar 1992, 18.04
Position: 1
Subnodes: 12

Interaction

Select

You can select a node by clicking on it once. A range of nodes can be selected by pressing the mouse button at one corner of the range and without releasing the button, move the pointer to the opposite corner of the range. The window will scroll to allow the selection to be extended out of view. The selection will be extended to include any previous selected nodes if the Shift key is pressed.

One selected node will never be an owner of another selected node. This rule imply that all selected nodes are independent of each other. For instance, when a group of selected nodes are cut out from a document the same node can’t be cut twice. The drawback of this rule is that commands addressed to the nodes on an individual basis wont be able to operate on a hierarchy of nodes. This is true for the Get Info command.

Double click

A double click on a node is a faster way to open its subnodes. If the node belongs to a group of selected nodes the subnodes of all the nodes will be opened.

Move

The selected nodes can be moved to another destination in the hierarchy of nodes. While the nodes are moved the current destination will be highlighted. The selected nodes will be inserted at the specified position in the same order they were selected.

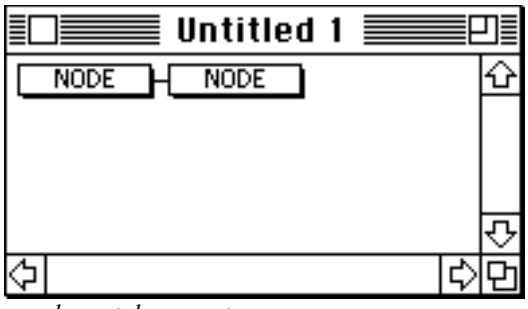
Nodes can’t be moved from one document to another document in this version of NodeViewer.

Examples

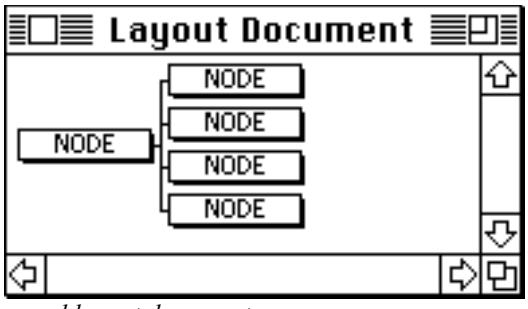
Layout Application

Imagine a simple layout application. A new empty document is open. The top node of the document keeps the information about the pages in the document through a set of subnodes. The top node stores the information about the size and the position of the window and the settings to the associated printer object. A single page in the document is automatically created by the top node.

Lets say we want our document to have four pages. We add three pages by selecting the corresponding command. Since we don't want to loose our work we save the document immediately by the name "Layout Document".



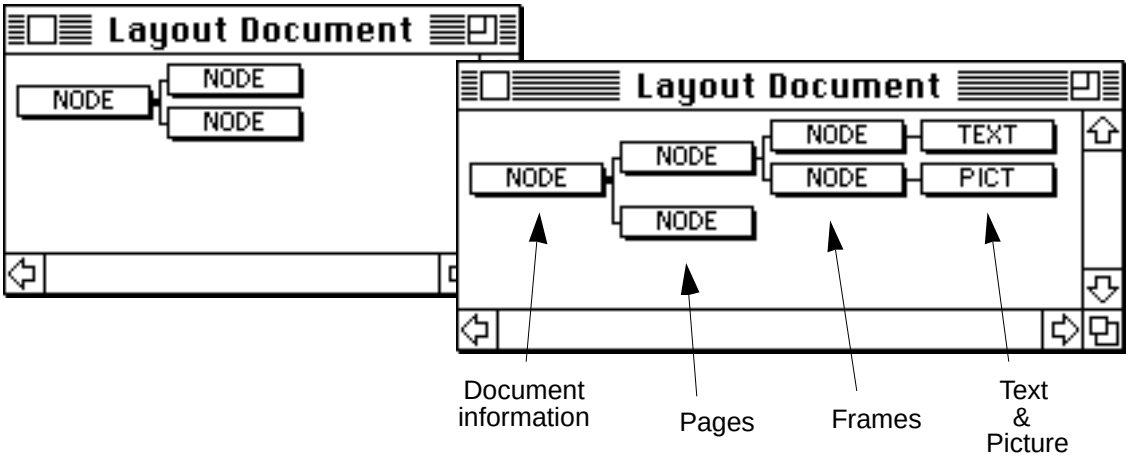
The new layout document



The saved layout document

We start to work with the first page of the document. Since the memory partition is very small the application needs some more memory and decide to close page two and three. That is OK because the pages have been saved and they are not visible at the moment. In NodeViewer we simulate this by selecting subnode two and three and choose the command Close Nodes.

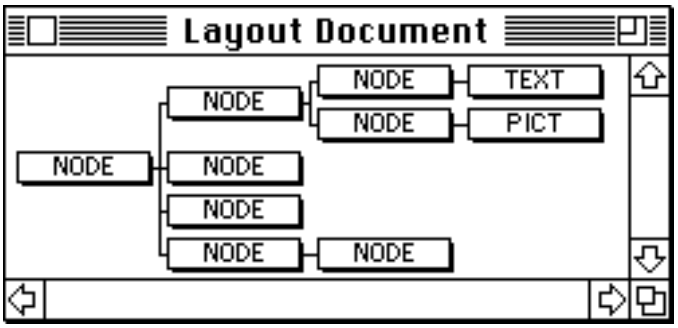
We make a text frame and write a short story about a picture we want to describe. The text frame is stored as a node with the information about position and size in the document. The text is stored as a standard text node. We copy the picture into the page. A default sized frame is created as a surrounding to the picture and a standard picture node is created to contain the information of the picture.



The information in the document is visualized by creating the corresponding hierarchy of visual objects. The document keeps the information of a window, the pages set the dimension of the main pane, the frames are subpanes in the main pane, the text operates through a CEditText, and the picture is visualized by a CPicture.

We decide to go to the last page in the document, where we plan the layout by creating a new frame. Only page two and page three are empty now. Lets leave the scenario and just see what happens when we give the top node the Open Subnodes command.

Page two and three are inserted between page one and four.

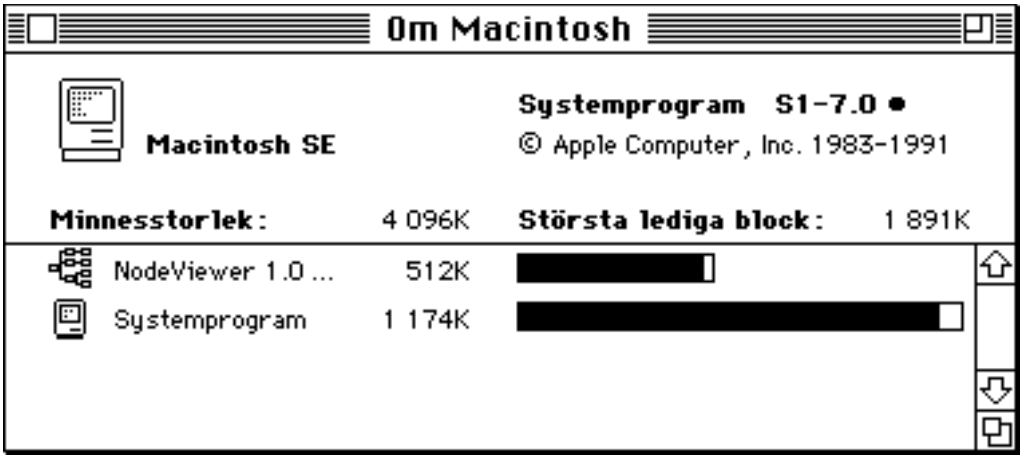


The purpose of this example is to show how the Tree Structure Class Library could be used in a real application. The information structure has been created by letting objects delegate responsibilities to subobjects. The objects are then implemented as different types of nodes.

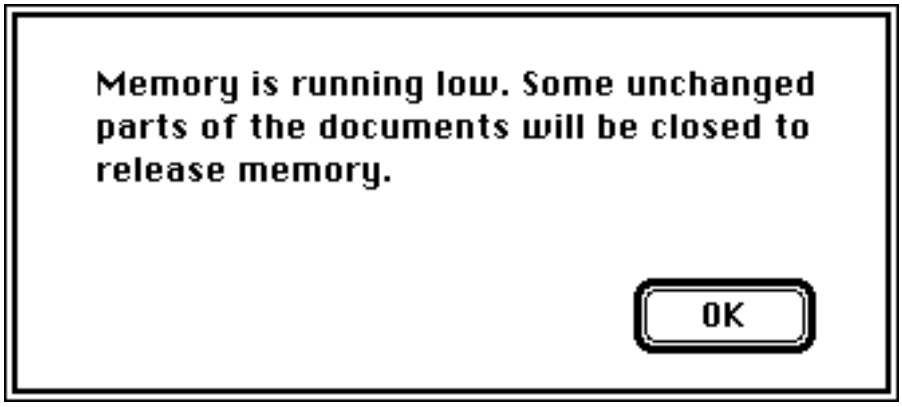
Memory Shortage

This example will show what happens when the application runs out of memory. You should only test this if all documents in other applications are saved or even better, just run NodeViewer as the only active application.

Make a new node document and call it “Very large”. Copy a fairly large picture and paste it as a subnodes several times in the document. Make a deep hierarchy of nodes by pasting more pictures as subnodes to the previous subnodes. Look in the About Macintosh dialog of the Finder and make sure almost all the memory of NodeViewer application is used.



Save the document and make a copy of it. Open the copy of the very large document. You will probably be able to open the document and the top node without memory shortage. But when you start opening the subnodes of the top node you will get the following message:

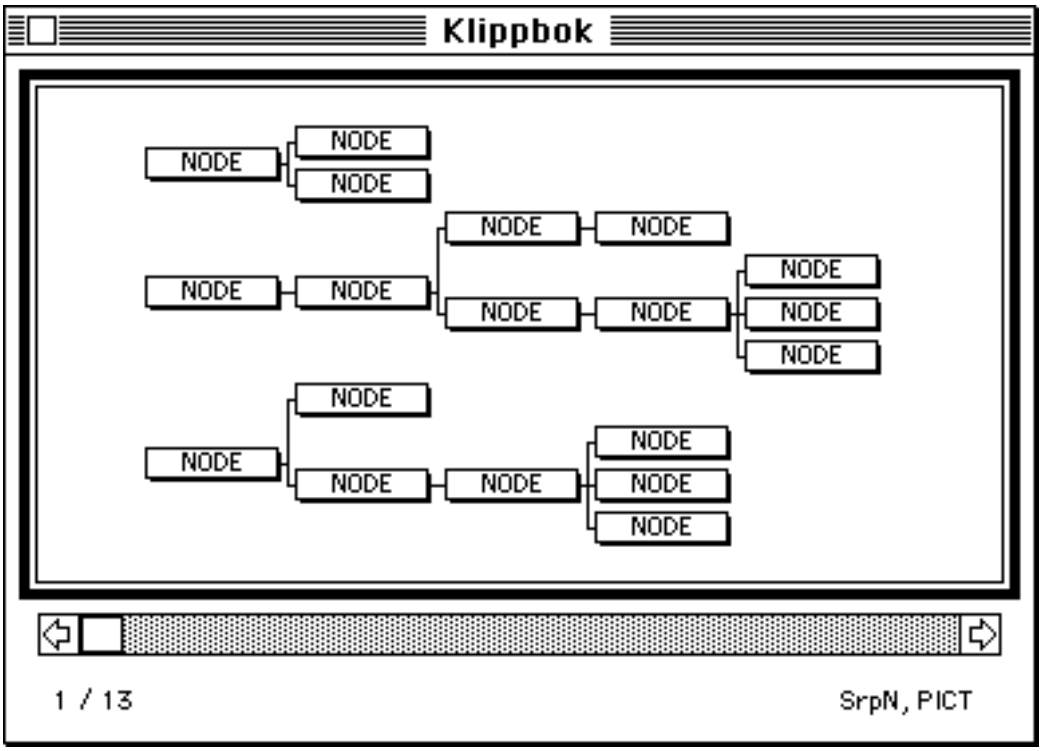


NodeViewer will close some nodes in the first document after showing this dialog. If NodeViewer didn't show all nodes currently in memory, it would close invisible nodes first. A node that has been changed and needs to be saved before it's closed will only be closed after the user is asked to save the document.

Any application using the Tree Structure Class Library can use this facility and specialize it to close the least needed nodes first. The user should of course always be informed of why things are closed.

The Scrapbook

Open for instance the document "Many nodes" and open a few branches of subnodes. Select a few subhierachies of nodes and make a copy of them. When you change to another application the nodes will change to a global scrap of type SrpN.



If none of the nodes just below the scrap node are picture nodes nor text nodes will a picture of the scrap be created. This picture can be used to visualize the hierarchy of nodes in a scrap outside NodeViewer, for instance in the Scrapbook. Applications using the Tree Structure Class Library can create scraps of any format in addition to the scrap node format.

General

NodeViewer

NodeViewer 1.0 is a test application and should be treated as such. Quipus will not take responsibility for information lost in connection with use of NodeViewer.

NodeViewer should work on all Macintosh computers. Macintosh SE, IIci, and IIsi has been tested. It requires at least System 6.0.

NodeViewer 1.0 was developed in THINK C version 5.0.4, THINK Class Library version 1.1.2, Tree Structure Class Library version 1.0 and Intelligent classes 1.0.

NodeViewer is a demonstartion of the Tree Structure Class Library and a tool to be used at development time of applications based on the Tree Structure Class Library. It's free and you are allowed to copy and distribute NodeViewer as long as this documentation and the examples files are kept with it.

The Tree Structure Class Library

The Tree Structure Class Library Version 1.0 is distributed with the following parts:

- Tree Structure Class Library 1.0 Source
- Tree Structure Class Library 1.0 Documentation
- Tree Structure Class Library 1.0 Templates
- NodeViewer 1.0 Source
- Intelligent Classes 1.0 Source

The fee of The Tree Structure Class Library is $20 US or the corresponding amount in SEK, British Pound, German D-Mark, Australian, New Zeeland or Canadian dollar. Please send the cash to the address below. You should specify your address, fax number, and your AppleLink or Internet address. You will get information of new versions of the class library and NodeViewer. Minor revisons will be sent to you at no charge and new versions will be charged a fee of $5 US.

Intelligent classes

The intelligent classes is a small class library adjusting a few of the classes in the Think Class Library 1.1. These classes makes the CDecorator, the CPanorama, the CScrollPane, and the CWindow a little more "intelligent" through a set of subclasses. They cooperate to make the default behaviour of windows containing scroll panes and panoramas nicer. You can use each class on its own but for best result you should use the classes together.

Quipus

Quipu, sometimes spelled QUIPO, an Incan accounting apparatus consisting of a long rope from which hung 48 secondary cords and various tertiary cords attached to the secondary ones. Knots were made in the cords to represent units, tens, and hundreds; and, in imperial accounting, the cords were differently coloured to designate the different concerns of government - such as tribute, lands, economic productivity, ceremonies, and matters relating to war and peace. The quipus were created and maintained as historical records and were kept not only by high officials at the capital of Cuzco - judges, commanders, and important heads of extended families - but also by regional commanders and village headman.

Encyclopœdia Britannica

Address

Please contact me regarding questions and error reports about NodeViewer, The Tree Structure Class Library or the Intelligent classes.

Address
Quipus
Mårten Sörliden
Djurgårdsgatan 21A
S-582 29 Linköping
Sweden

Phone +46 13 12 79 79
AppleLink SW1340
Internet SW1340@AppleLink.Apple.com