GRANT
IN-61-CR
157064
P-47

Old Dominion University Research Foundation

DEPARTMENT OF COMPUTER SCIENCE
COLLEGE OF SCIENCES
OLD DOMINION UNIVERSITY
NORFOLK, VIRGINIA 23529

GEOMETRIC MODELING FOR COMPUTER AIDED DESIGN

By

James L. Schwing, Principal Investigator

Progress Report
For the period ended December 15, 1987

Prepared for the
National Aeronautics and Space Administration
Langley Research Center
Hampton, Virginia 23665

Under
Research Grant NCC1-99
John J. Rehder, Technical Monitor
SSD-Vehicle Analysis Branch

(NASA-CR-183161) GEOMETRIC MODELING FOR COMPUTER AIDED DESIGN Progress Report, period ending 15 Dec. 1987 (Old Dominion Univ.) 47 p CSCL 09B N88-30320
Unclas
G3/61 0157064

August 1988

DEPARTMENT OF COMPUTER SCIENCE
COLLEGE OF SCIENCES
OLD DOMINION UNIVERSITY
NORFOLK, VIRGINIA 23529

GEOMETRIC MODELING FOR COMPUTER AIDED DESIGN

By

James L. Schwing, Principal Investigator

Progress Report
For the period ended December 15, 1987

Prepared for the
National Aeronautics and Space Administration
Langley Research Center
Hampton, Virginia 23665

Under
Research Grant NCC1-99
John J. Rehder, Technical Monitor
SSD-Vehicle Analysis Branch

Submitted by the
Old Dominion University Research Foundation
P.O. Box 6369
Norfolk, Virginia 23508

August 1988

1 Introduction

The following report summarizes the research carried out during the period of June, 1987 through December 1987 for *NASA* grant NCCI-99. The work described here was carried out by the principal investigator, James Schwing, and a graduate research assistant, Jan Spangler, in cooperation with the *SMART* (Solid Modeling Aerospace Research Tool) system design team of the Vehicle Analysis Branch at *NASA* Langley.

Research during this period focused on two major areas. The first effort addressed the design and implementation of a technique that allows for the visualization of the real-time variation of physical properties. The second effort focused on the design and implementation of an on-line help system with components designed for both authors and users of help information.

2 Real-time Visualization Algorithms for the Display of Physical Properties

Calculation of physical properties is an important step in the analysis of aerospace vehicles. Indeed, much of the earlier research under this grant has gone into the development of techniques appropriate to the conceptual design environment. Key aspects of this design environment are summarized below and are taken from [1,2].

Conceptual design requires the evaluation of a multitude of vehicle concepts. To achieve high productivity, a designer must be able to generate complete and accurate three dimensional models of complex vehicle shapes easily, quickly and in a natural way. Completeness of the design process requires, among other things, calculation of physical properties to be carried out as efficiently as possible with in required accuracy constraints. Ease of definition is aided by the use of hierarchical data structures (trees). Trees provide a natural technique for defining and manipulating a configuration.

As mentioned above, previous work on this grant has lead to the devel- opment and implementation of algorithms which improved the efficiency of the calculation of physical properties by a factor of four. However, even with this improvement, calculations for a complex vehicle, such as a concept for a combination booster and orbiter, would require several minutes to complete. Such times are acceptable (and unavoidable given the complexity of vehicle configuration) for a one-time calculation of a given concept.

It should be noted however that the conceptual design process requires the frequent redefinition of major parameters leading to the need for continuing recomputation of physical properties. This raises questions concerning the time required to carry out that process by completely reintegrating the entire vehicle. The sections that follow present conditions under which the amount of computation involved in a revised configuration is trivial. Algorithms are then presented which apply this theory to the visualization of properties such as the centroid. Current implementation has provided the *SMART* designers with a tool that graphically represents the real-time, dynamic changes in the position of the centroid as various sub-assemblies are moved relative to one another. Questions such as how the centroid of a vehicle changes as a function of the position of the wings or fuel tanks are answered graphically in real- time.

2.1 Assumptions and Observations

The basic data structure is hierarchical in nature and can be represented by a tree, see for example *figure 1*. Elements at the leaf positions, such as C_1 and C_2, represent components or the basic building blocks of the design. These nodes also contain a full analytic surface definition for both graphical presentation and analytical computation. The assembly node, A, is the combination of the lower level components via a disjoint union. It is assumed that the general Boolean operations of union, difference and intersection are carried out by *SMART* prior to these computations. Branch labels, T, T_1 and T_2, represent transformation matricies with T_1 and T_2 placing respective components within the assembly and T orienting the total assembly. Such transformations are made up of designer specified rotations, translations and scalings.

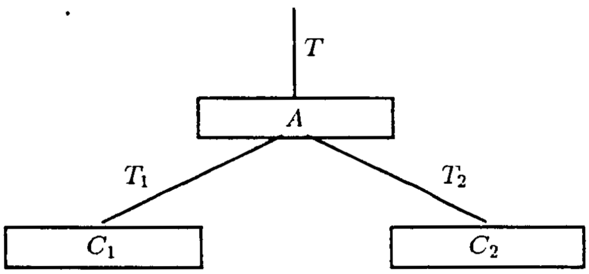


Figure 1.

For initial conceptual design efforts, mass properties are based upon the assumption of uniformly dense objects. For example, a solid is represented by a uniform density times its volume obtained via integration over the analytic surface. Notice that certain objects may be thought of as the combination of more than one type of density. A fuel tank may be represented by both a solid, density per unit volume, for the mass of the fuel and a shell, density per unit area, for the mass of the tank structure.

Under these assumptions, the following observations can be made.

Observation 1:

Mass of an assembly node can be derived from the sum of its component nodes.

Observation 2:

Mass of an assembly node is unaffected by the relative positioning, rotational and translational, of its component nodes.

2.2 Physical Properties and the Effects of Geometric Transformations

The goal of this section is to record the effects of geometric transformations on the physical properties of the assembly described in *figure 1*. General references for this information may be found in [3,4].

2.2.1 Centroids and Translation and Rotation

Let $(\bar{x}_i, \bar{y}_i, \bar{z}_i)$ and M_i be the centroid and mass respectively of the component C_i. Let $(\hat{x}_i, \hat{y}_i, \hat{z}_i)$ be the transformed centroid of the component C_i. It is assumed that the transformation T_i consists only of rotation and/or translation. In which case the transformed component centroid is found simply by applying the indicated transformation to the original centroid. In addition, it is observed that the mass properties are unaffected by the rotation and/or translation. Now the x centroid of the assembly may be calculated by

$$\bar{x}_A = \frac{M_1 x_1 + M_2 x_2}{M_A}.$$

Similar equations hold for the y and z centroids.

2.2.2 Centroids and Uniform Scaling

In order to proceed with the case for uniform scaling, it is necessary to review how a components original centroid was calculated. For example suppose that the component represents a shell, then the x centroid is derived by

$$\begin{aligned} \bar{x}_i &= \frac{\int_{C_i} x\,dm}{\int_{C_i} dm} \\ &= \frac{\int_R x d_i J\,ds\,dt}{\int_R d_i J\,ds\,dt} \end{aligned}$$

where $J = \sqrt{EG - F^2}$, the Jacobian, d_i represents the density factor and R represents the region of parameterization.

If uniform scaling occurs then the same region R may be used. Each coordinate parameterization becomes a scaled version of the original. For example, if k represents the uniform scale factor, then

$$\begin{aligned} X(s,t) &= kx(s,t), \\ \partial X/\partial s &= k\partial x/\partial s, \\ \partial X/\partial t &= k\partial x/\partial t. \end{aligned}$$

Thus

$$\begin{aligned} \hat{E} &= (\partial X/\partial s)^2 + (\partial Y/\partial s)^2 + (\partial Z/\partial s)^2 \\ &= k^2[(\partial x/\partial s)^2 + (\partial y/\partial s)^2 + (\partial z/\partial s)^2] \\ &= k^2 E. \end{aligned}$$

And similarly

$$\begin{aligned} \hat{F} &= k^2 F \\ \hat{G} &= k^2 G. \end{aligned}$$

So that

$$\hat{J} = k^2 J.$$

And finally,

$$\begin{aligned} \hat{x}_i &= \frac{\int_{C_i} X dm}{\int_{C_i} dm} \\ &= \frac{\int_R X d_i J dsdt}{\int_R d_i J dsdt} \\ &= \frac{\int_R kxd_i k^2 J dsdt}{\int_R d_i k^2 J dsdt} \\ &= k\bar{x}_i. \end{aligned}$$

Similarly,

$$\begin{aligned} \hat{y}_i &= k y_i \\ \hat{z}_i &= k z_i. \end{aligned}$$

Analysis for the solid and linear cases lead to the same result. Notice the simplification of the defining integral was dependent upon the scaling being uniform. Non-uniform scaling requires reintegration to determine the transformed centroid.

2.2.3 Moments and Products of Inertia and Translation

Consider first the component itself. Assume that I_x represents the moment of inertia parallel to the x axis. Further let $l_{\bar{x}}$ represent the distance of the original centroid from the x axis, that is $l_{\bar{x}}^2 = \bar{y}^2 + \bar{z}^2$. Similarly let $l_{\hat{x}}$ represent the distance from the transformed centroid to the x axis.

Then

$$\hat{I}_x = I_x + M(l_{\hat{x}}^2 - l_{\bar{x}}^2).$$

Equivalent formulae holding for the other moments and products of inertia. Please refer to [4] for a complete listing of these formulae. From these, the moments and products of inertia for the assembly can be obtained by summing those for each of the components.

2.2.4 Moments and Products of Inertia and Rotation

Again, consider the effect of rotation on a single component. Assume that the given rotation is represented by a post-multiplication matrix

$$R = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{23} & r_{33} \end{pmatrix}.$$

For this case,

$$\begin{aligned} \hat{I}_x &= I_x r_{11}^2 + I_y r_{21}^2 + I_z r_{31}^2 - \\ &\quad 2(P_{xy} r_{11} r_{21} + P_{xz} r_{11} r_{31} + P_{yz} r_{21} r_{31}). \end{aligned}$$

Again a full set of formulae for rotated moments and products of inertia can be found in [4]. Also as is the case for translation, the sum of rotated moments and products of inertia leads to those for the assembly node.

2.2.5 Moments and Products of Inertia and Uniform Scaling

As in the development of section 2.2.2, the section below will detail the effects of uniform scaling on I_x when the given component is assumed to be a shell. Results extend to other coordinates and to solid and linear components as well. Those results will be stated but not looked at in detail.

Recall

$$I_x = \int_{C_i} (y^2 + z^2) dm.$$

Therefore,

$$\begin{aligned} \hat{I}_x &= \int_{C_i} (Y^2 + Z^2) dm \\ &= \int_R (k^2 y^2 + k^2 z^2) k^2 d_i J ds dt \\ &= k^4 I_x. \end{aligned}$$

For shells, all other moments and products of inertia also reflect a scaling of k^4 for a uniform scale factor of k. In the case of solids, the moments and products are scaled by k^5, while linear components are scaled by k^3.

2.3 Calculation of Physical Properties

Previous work on this grant produced a collection of algorithms capable of computing physical properties for components of the type defined by the *SMART* system [5]. These routines can be used to generate an initial collection of physical properties for each leaf node in the hierarchical structure of the tree.

As leaf nodes are defined, *SMART* allows the arbitrary application of geometric transformations including rotation, translation and scaling, both uniform and non-uniform. Later as the designer moves on to the definition of assemblies, all geometric transformations but one remain available for the positioning of assemblies. The exception is non-uniform scaling. This restriction does not limit the scope of *SMART* as all appropriate non-uniform scaling should always be applied to leaf node components.

Section 2.2 provides the foundation which allows the construction of algorithms that transform physical properties throughout the hierarchy from those of the basic leaf components calculated above. This provides access to physical properties at all levels without the need to continually resort to complicated quadrature techniques. In addition, section 2.2 provides a foundation for algorithms which provide a rapid update of physical properties after any geometric transformations except non-uniform scaling.

Before proceeding, it is noted that any node, whether leaf or assembly, is allowed to have constituent portions from each of the three types of densities: solid, shell and linear. Thus records are kept for each node that track the contribution of each type as well as the total. Secondly, it is noted that all geometric transformations in *SMART* are accumulated and applied to objects in the order: scaling, rotating and translating.

2.3.1 Algorithm for Property Calculation at Leaf Nodes

1. Determine if properties are being calculated for the first time or if non-uniform scaling has been applied.

CASE: First-time calculations or non-uniform scaling

2. Apply or reapply the quadrature approximation schemes of [5] to this leaf node.

CASE: All other transforms

3. Determine the new mass, M_s, M_{sh} and M_l, for each type, solid, shell and linear, by scaling previously defined weights by the appropriate factor, k^3, k^2 and k, respectively.

4. Fully scale, rotate and translate the centroid for each type. Refer to sections 2.2.1 and 2.2.2 for the appropriate formulae.
5. Fully scale, rotate and translate the moments and products of inertia for each type. Refer to sections 2.2.3, 2.2.4 and 2.2.5.
6. Form a mass oriented centroid from a mass weighted average of the three types of densities. For example,

$$\bar{x} = \frac{M_l \bar{x}_l + M_{sh} \bar{x}_{sh} + M_s \bar{x}_s}{M_l + M_{sh} + M_s}.$$

7. Form the total moments and products of inertia by accumulating those of each type. For example,

$$I_x = I_{xl} + I_{xsh} + I_{xs}.$$

2.3.2 Algorithm for Propagating Properties Throughout the Hierarchy

Throughout this section, it is assumed that for a given assembly node all components have had their physical properties calculated and transformed with respect to their positioning within the assembly as described in 2.3.1. The algorithm presented below will then combine this information with any geometric transformations of the assembly itself to derive properties in a form ready to pass on to the next level of the hierarchy.

As a final observation of this section, it is noted that all levels of the tree can be generated by a recursive traversal. Therefore combination of the algorithm below with that of 2.3.1 and a traversal of the tree will cause properties for the entire configuration to be generated.

1. Determine the total mass of each type, solid, shell and linear, for the assembly by summing those of each of the corresponding components. Determine the total mass by summing the mass of each type.

$$\begin{aligned} M_{lA} &= \sum_i (k M_{lC_i}), \\ M_A &= M_{lA} + M_{shA} + M_{sA}. \end{aligned}$$

2. Scale each of the type of masses for the assembly, M_{lA}, M_{shA} and M_{sA}, by the appropriate scale factor, k, k^2 and k^3, respectively.

3. For each component, fully scale, rotate and translate the centroid of each type. Refer to sections 2.2.1 and 2.2.2.

4. For each type, form the centroid of the assembly from the mass weighted average of its components. For example, consider the linear centroid:

$$\bar{x}_{lA} = \frac{\sum_i (k M_{lC_i} \bar{x}_{lC_i})}{M_{lA}}.$$

5. Form the mass centroid of the assembly using a mass weighted average of each type. For example,

$$\bar{x}_A = \frac{M_{lA}\bar{x}_{lA} + M_{shA}\bar{x}_{shA} + M_{sA}\bar{x}_{sA}}{M_A}.$$

6. For each component, fully scale, rotate and translate the moments and products of inertia of each type. Refer to sections 2.2.3, 2.2.3 and 2.2.5.

7. For each type, form the moments and products of inertia for the assembly by summing those of the components.

$$I_{xlA} = \sum_i I_{xlC_i}.$$

8. Form mass based moments and products of inertia for the assembly by summing those of each type.

$$I_{xA} = I_{xlA} + I_{xshA} + I_{xsA}.$$

2.3.3 Algorithm for Updating Properties

This section presents an algorithm which allows for the rapid update of properties when geometric transformations are applied to any node within the hierarchical structure of the overall configuration.

Assume a transformation is applied to node C in *figure 2*. Note, no assumption is made concerning whether C is a leaf node or an assembly node. Properties of C must now be updated. As we have seen in the previous section property evaluations of C contribute to the property evaluations of C's parents. Thus the parent must also be updated. However, for the parents, only the portion of the evaluation associated with C must be updated. (*SMART* only allows the geometric transformation of one component or assembly at a time.) This argument applies to all higher level generations up to the root of the tree.

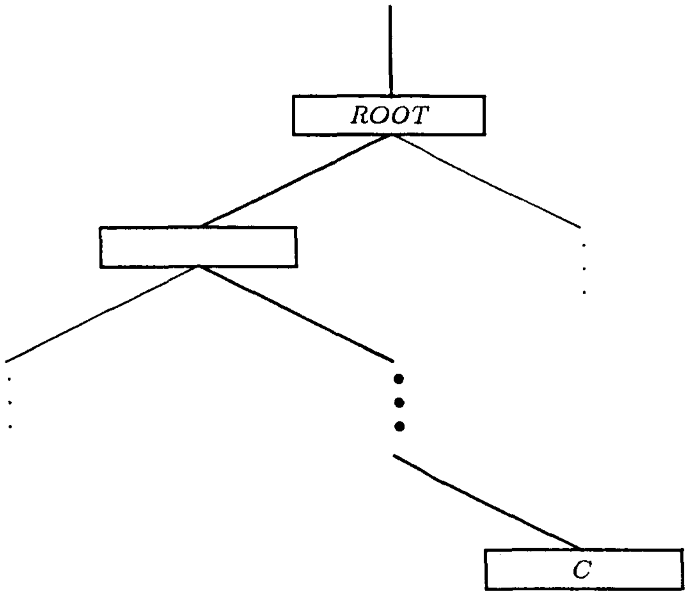


Figure 2.

In summary, the algorithm described here starts with the root and visits nodes along the branch of the tree leading to C. At each node it visits, it subtracts the portion of the physical properties calculation due to C. Once node C is reached, the evaluations of its properties is updated to reflect the new geometric transformations. Finally, the path is reversed and the revised contributions due to C are added back into the upper level nodes.

1. Loop over all nodes along the branch from $ROOT$ to C.

   2. Subtract contributions to total, solid, shell and linear mass components made by the node at the next level down.

   3. Subtract contributions to total, solid, shell and linear moments and products of inertia made by the node at the next level down.

   4. Subtract contributions to total, solid, shell and linear centroids made by the node at the next level down.

5. End loop.

Notes: This is essentially the reverse of the process described for section 2.3.2. With subtraction made only for the component in need of update. Caution is urged in the calculation of the appropriate subtraction factor. Recall,

$$\bar{x}_A = \frac{\sum_i (k\bar{x}_{lC_i} M_{lC_i}) + \sum_i (k^2 \bar{x}_{shC_i} m_{shC_i}) + \sum_i (k^3 \bar{x}_{sC_i} M_{sC_i})}{M_A}.$$

Assume that the jth component needs removal. Then

$$\begin{aligned}
\tilde{M}_{lA} &= M_{lA} - kM_{lC_j}, \\
\tilde{M}_{shA} &= M_{shA} - kM_{shC_j}, \\
\tilde{M}_{sA} &= M_{sA} - kM_{sC_j} \\
\tilde{M}_A &= \tilde{M}_{lA} + \tilde{M}_{shA} + \tilde{M}_{sA}.
\end{aligned}$$

So that,

$$\tilde{\bar{x}}_A = \frac{\bar{x}_A M_A - k\bar{x}_{lC_j} M_{lCj} - k^2 \bar{x}_{shC_j} M_{shC_j} - k^3 \bar{x}_{sC_j} M_{sC_j}}{\tilde{M}_A}.$$

6. Update the properties of C. Use 2.3.1 if C is a leaf node. Use 2.3.2 if C is an assembly node.
7. Loop over nodes along the branch from C to *ROOT*.
   8. Add contribution to total, solid, shell and linear mass by node to the next level up.
   9. Add contribution to total, solid, shell and linear moments and products of inertia by node to the next level up.
   10. Add contribution to total, solid, shell and linear centroids by node to the next level up.
11. End loop.

Note: The same cautions apply when adding back as discussed for subtraction above.

2.3.4 Comparision of Calculations

The purpose of this section is to compare the amount of calculation that occurs in the recalculation of physical properties when computations are carried out by reintegration of the components versus updating properties via the algorithms described in the early portions of this chapter.

For simplicity, refer again to the configuration described by the hierarchy of ***figure 1***. Suppose that a geometric transformation is applied to one of the components, say C_1. There are two possible ways that a system such as $SMART$ could apply the quadrature approximations of [5] to derive updated property calculations of the assembly, A.

First, a new representation for the final world orientation of A could be derived by applying updated transformation T_1 and the transformation T to each of the patches of C_1. The techniques of [5] may then be applied directly to the combination of these newly transformed patches and the previously transformed patches of C_2.

Alternately, the system could take advantage of the algorithm for propagating propery calculations throughout the hierarchy by the techniques described in section 2.3.2. In this case, once a geometric transformation was applied to the patches defining C_1, the system could apply the approximations of [5] and repropagate these results with those already calulated for the node C_2.

In either of these instances the quadrature approximations of [5] are applied at a minimum to each of the patches defining the newly transformed C_1. Inspection of the approximaton algorithm reveals that the quadrature techniques of [5] requires the evaluation of a given surface patch at $(6n+7)^2$ points. Here n is the number of iteration levels required to get convergence for the defining property integral. Such evaluations require the computation of a four dimensional vector matrix product in each of three coordinates. Fortunately, the calculations of [5] reuse the evaluation at these points for each of the physical properties, so that they need only be done once for each patch.

In the alternative proposed above, as long as the tranformation on C_1 does not involve non-uniform scaling, no reintegration is required. Consider the update of the centroid assuming all three geometric transformation have occurred, i.e. uniform scaling, rotation and translation. Begin by updating the mass of C_1. This may require as many as three multiplications if C_1 contains portions defined by each type of density, linear, shell and solid. Next each portion of the centroid of C_1, again linear, shell and solid, must be scaled, rotated and translated. A maximum of ten multiplications and six additions per coordinate and type.

Thus the implementation of the algorithms of this chapter insure that the recomputation of physical properties is a function only of the depth of the tree and not the complexity of the componets that make up the configuration. Neither an increase in the number of patches per component, the number of leaf node components nor the need for increased iterations in the quadrature scheme for improved accuracy will increase the amount of computation required when updating transformed physical properties. This fact combined with the small number of artihmentic operations invlolved in an update allow for the real-time visualization of transformation of these properties.

2.4 An Example

The configuration represented by *figure 3* has been put together to illustrate the capability for using the techniques of the previous sections in a simple verifiable model. It is recognized that the configuration itself is unnecessarily complicated. Extraneous operations and nodes are included to illustrate the capabilities of the algorithms. Indeed, the section will conclude with an equivalent configuration in simplified form.

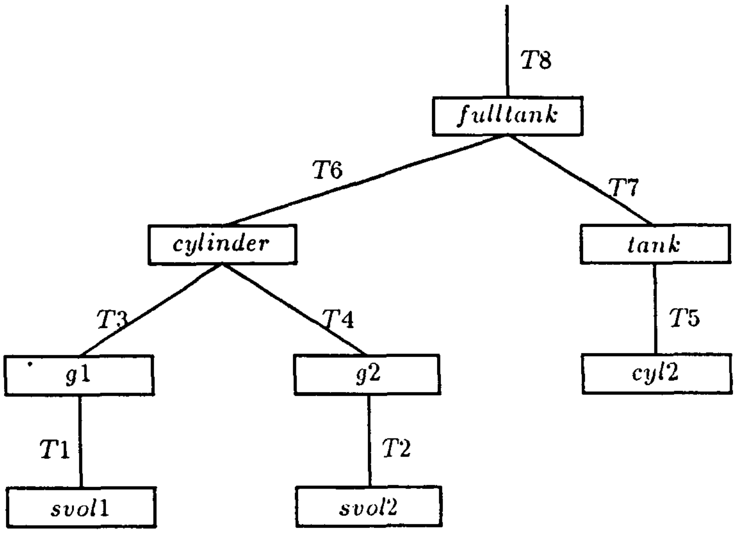


Figure 3.

The configuration *fulltank* represents a cylindrical, fueled tank. The "fuel" for this tank is defined as the union of two separate cylinders $g1$ and $g2$ while *tank* represents the structure of the tank, i.e. a cylindrical shell. A description of the component nodes and geometric transformations follow.

*svol*1, *svol*2 - right circular cylinders, $h = .25$, $r = .5$, axis of rotational symmetry: x, centroid: (0,0,0), solid density: 1

$T1$ - transforms *svol*1, uniformly scales: $k = 1$, translates: $x = -.25$

$T2$ - transforms *svol*2, uniformly scales: $k = 2$, translates: $x = .25$

$g1$ - result of transforming *svol*1 by $T1$

$g2$ - result of transforming *svol*2 by $T2$

$T3$, $T4$ - identity transformations on $g1$ and $g2$

cylinder - the union of $g1$ and $g2$

cyl2 - right circular cylinder, $h = 1$, $r = 1$, axis of rotational symmetry: x, centroid: (0,0,0), shell density: 1

T5 - identity transformation of *cyl2*

tank - same as *cyl2* since no geometric transformation occurs

T6, *T7* - identity transformations on *cylinder* and *tank* respectively

fulltank - union of fuel *cylinder* and *tank* structure combining both shell and solid components

T8 - translation of *fulltank* to its final position: $x = 1$, $y = 1$, $z = 1$

In the tables that follow, quadrature approximation calculations were performed only for the leaf nodes. Properties at the other nodes of the tree are derived via the techniques described earlier. It is also at this point that *SMART* could be used to apply further geometric transformations. Property updates would then be calculated on the fly.

Table 1.

PHYSICAL PROPERTIES OF svoll

THE PROPERTIES OF VOLUME			
total volume:	0.1963	$\pi/16$	= 0.1963
weight per unit volume:	1.0000		
volume-based centroid:	(0.0000, 0.0000, 0.0000)		
volume-based xx moment:	0.0245	$\pi/128$	= 0.0245
volume-based yy moment:	0.0133	$13\pi/3072$	= 0.0133
volume-based zz moment:	0.0133	$13\pi/3072$	= 0.0133
volume-based xy product:	0.0000	0	
volume-based xz product:	0.0000	0	
volume-based yz product:	0.0000	0	
THE PROPERTIES OF MASS			
total mass:	0.1963	$\pi/16$	= 0.1963
mass by area:	0.0000		
mass by volume:	0.1963	$\pi/16$	= 0.1963
mass-based centroid:	(0.0000, 0.0000, 0.0000)		
mass-based xx moment:	0.0245	$\pi/128$	= 0.0245
mass-based yy moment:	0.0133	$13\pi/3072$	= 0.0133
mass-based zz moment:	0.0133	$13\pi/3072$	= 0.0133
mass-based xy moment:	0.0000	0	
mass-based xz moment:	0.0000	0	
mass-based yz moment:	0.0000	0	

Table 1. (cont.)

PHYSICAL PROPERTIES OF g1

THE PROPERTIES OF MASS			
total mass:	1.5707	$\pi/2$	= 1.5708
mass by area:	0.0000		
mass by volume:	1.5707	$\pi/2$	= 1.5708
mass-based centroid:	(-0.2500, 0.0000, 0.0000)		
mass-based xx moment:	0.7855	$\pi/4$	= 0.7854
mass-based yy moment:	0.5238	$\pi/6$	= 0.5236
mass-based zz moment:	0.5238	$\pi/6$	= 0.5236
mass-based xy moment:	0.0000	0	
mass-based xz moment:	0.0000	0	
mass-based yz moment:	0.0000	0	

PHYSICAL PROPERTIES OF svol2

THE PROPERTIES OF VOLUME			
total volume:	0.1963	$\pi/16$	= 0.1963
weight per unit volume:	1.0000		
volume-based centroid:	(0.0000, 0.0000, 0.0000)		
volume-based xx moment:	0.0245	$\pi/128$	= 0.0245
volume-based yy moment:	0.0133	$13\pi/3072$	= 0.0133
volume-based zz moment:	0.0133	$13\pi/3072$	= 0.0133
volume-based xy product:	0.0000	0	
volume-based xz product:	0.0000	0	
volume-based yz product:	0.0000	0	

THE PROPERTIES OF MASS			
total mass:	0.1963	$\pi/16$	= 0.1963
mass by area:	0.0000		
mass by volume:	0.1963	$\pi/16$	= 0.1963
mass-based centroid:	(0.0000, 0.0000, 0.0000)		
mass-based xx moment:	0.0245	$\pi/128$	= 0.0245
mass-based yy moment:	0.0133	$13\pi/3072$	= 0.0133
mass-based zz moment:	0.0133	$13\pi/3072$	= 0.0133
mass-based xy moment:	0.0000	0	
mass-based xz moment:	0.0000	0	
mass-based yz moment:	0.0000	0	

Table 1. (cont.)

PHYSICAL PROPERTIES OF g2

THE PROPERTIES OF MASS

total mass:	1.5707	$\pi/2$	= 1.5708
mass by area:	0.0000		
mass by volume:	1.5707	$\pi/2$	= 1.5708
mass-based centroid:	(0.2500,	0.0000,	0.0000)
mass-based xx moment:	0.7855	$\pi/4$	= 0.7854
mass-based yy moment:	0.5238	$\pi/6$	= 0.5236
mass-based zz moment:	0.5238	$\pi/6$	= 0.5236
mass-based xy moment:	0.0000	0	
mass-based xz moment:	0.0000	0	
mass-based yz moment:	0.0000	0	

PHYSICAL PROPERTIES OF cyl2

THE PROPERTIES OF AREA

total surface area:	12.5650	4π	= 12.5663
weight per unit area:	1.0000		
projected area in xy plane:	1.9994	2	
projected area in xz plane:	1.9994	2	
projected area in yz plane:	3.1415	π	= 3.1416
surface-based centroid:	(0.0000,	0.0000,	0.0000)
area-based xx moment:	9.4257	3π	= 9.4248
area-based yy moment:	6.8069	$13\pi/6$	= 6.8068
area-based zz moment:	6.8069	$13\pi/6$	= 6.8068
area-based xy product:	0.0000	0	
area-based xz product:	0.0000	0	
area-based yz product:	0.0000	0	

THE PROPERTIES OF MASS

total mass:	12.5650	4π	= 12.5663
mass by area:	12.5650	4π	= 12.5663
mass by volume:	0.0000		
mass-based centroid:	(0.0000,	0.0000,	0.0000)
mass-based xx moment:	9.4257	3π	= 9.4248
mass-based yy moment:	6.8069	$13\pi/6$	= 6.8068
mass-based zz moment:	6.8069	$13\pi/6$	= 6.8068
mass-based xy moment:	0.0000	0	
mass-based xz moment:	0.0000	0	
mass-based yz moment:	0.0000	0	

Table 1. (cont.)

PHYSICAL PROPERTIES OF cylinder

THE PROPERTIES OF MASS			
total mass:	3.1415	π	= 3.1416
mass by area:	0.0000		
mass by volume:	3.1415	π	= 3.1416
mass-based centroid:	(0.0000,	0.0000,	0.0000)
mass-based xx moment:	1.5709	$\pi/2$	= 1.5708
mass-based yy moment:	1.0476	$\pi/3$	= 1.0476
mass-based zz moment:	1.0476	$\pi/3$	= 1.0476
mass-based xy moment:	0.0000	0	
mass-based xz moment:	0.0000	0	
mass-based yz moment:	0.0000	0	

PHYSICAL PROPERTIES OF tank

THE PROPERTIES OF MASS			
total mass:	12.5650	4π	= 12.5663
mass by area:	12.5650	4π	= 12.5663
mass by volume:	0.0000		
mass-based centroid:	(0.0000,	0.0000,	0.0000)
mass-based xx moment:	9.4257	3π	= 9.4248
mass-based yy moment:	6.8069	$13\pi/6$	= 6.8068
mass-based zz moment:	6.8069	$13\pi/6$	= 6.8068
mass-based xy moment:	0.0000	0	
mass-based xz moment:	0.0000	0	
mass-based yz moment:	0.0000	0	

PHYSICAL PROPERTIES OF fulltank

THE PROPERTIES OF MASS	COMPUTED		EXACT
total mass:	15.7065	5π	= 15.7079
mass by area:	12.5650	4π	= 12.5663
mass by volume:	3.1415	π	= 3.1416
mass-based centroid:	(1.0000,	1.0000,	1.0000)
mass-based xx moment:	42.4096	$27\pi/2$	= 42.4115
mass-based yy moment:	39.2675	$75\pi/6$	= 39.2699
mass-based zz moment:	39.2675	$75\pi/6$	= 39.2699
mass-based xy moment:	15.7065	5π	= 15.7079
mass-based xz moment:	15.7065	5π	= 15.7079
mass-based yz moment:	15.7065	5π	= 15.7079

The section concludes with a more appropriate representation for the prior configuration. In this case, there is a single component defined with both solid and shell characteristics scaled and translated to its final position.

Table 2.

PHYSICAL PROPERTIES OF cyl

THE PROPERTIES OF AREA	COMPUTED		EXACT
total surface area:	12.5650	4π	= 12.5663
weight per unit area:	1.0000		
projected area in xy plane:	1.9994	2	
projected area in xz plane:	1.9994	2	
projected area in yz plane:	3.1415	π	= 3.1416
surface-based centroid:	(1.0000, 1.0000, 1.0000)		
area-based xx moment:	34.5557	11π	= 34.5575
area-based yy moment:	31.9369	$61\pi/6$	= 31.9395
area-based zz moment:	31.9369	$61\pi/6$	= 31.9395
area-based xy product:	12.5650	4	= 12.5663
area-based xz product:	12.5650	4	= 12.5663
area-based yz product:	12.5650	4	= 12.5663
THE PROPERTIES OF VOLUME			
total volume:	3.1415	π	= 3.1416
weight per unit volume:	1.0000		
volume-based centroid:	(1.0000, 1.0000, 1.0000)		
volume-based xx moment:	7.8539	$5\pi/2$	= 7.8540
volume-based yy moment:	7.3305	$7\pi/3$	= 7.3304
volume-based zz moment:	7.3305	$7\pi/3$	= 7.3304
volume-based xy product:	3.1415	π	= 3.1416
volume-based xz product:	3.1415	π	= 3.1416
volume-based yz product:	3.1415	π	= 3.1416
THE PROPERTIES OF MASS			
total mass:	15.7065	5π	= 15.7079
mass by area:	12.5650	4π	= 12.5663
mass by volume:	3.1415	π	= 3.1416
mass-based centroid:	(1.0000, 1.0000, 1.0000)		
mass-based xx moment:	42.4096	$27\pi/2$	= 42.4115
mass-based yy moment:	39.2674	$75\pi/6$	= 39.2699
mass-based zz moment:	39.2674	$75\pi/6$	= 39.2699
mass-based xy moment:	15.7065	5π	= 15.7079
mass-based xz moment:	15.7065	5π	= 15.7079
mass-based yz moment:	15.7065	5π	= 15.7079

3 On-line Help Systems: Authoring and Using

Until recently, one of the least developed portions of many systems was the help provided to the users of the system. In addition, tools for system developers to easily create documentation and help facilities were also lacking. However, much current research has begun to address the issues of design and implementation that arise in on-line help systems. A current list of such issues and references can be found in [6].

Such recent research has shown that the major uses of help systems will be three fold: first, for the presentation of summary and tutorial information, secondly as a general source of on-line system capabilities and finally, as a detailed compendium of system operation. For her master's project and as her contribution to this grant, research assistant Jan Spangler looked into the development of such a system for *SMART*. A complete system called *ManualWriter* was the result. The appendix contains a major portion of her master's project report on that system.

References

1. *Solid Modeler For Aerorspace Vehicle Preliminary Design*, M.L. McMillan, J.J. Rehder, A.W. Wilhite, J.L. Schwing, J.L. Spangler and J.C. Mills, presented to *AIAA* Conference, August 1987, also *NASA* Technical Report.

2. *Geometric Modeling for Computer Aided Design*, J. L. Schwing, Progress Report for NCCI-99, 1986.

3. **Advanced Calculus**, A.E. Taylor, Blaisdell Publishing, 1955.

4. **Weight Engineers Handbook**, Society of Allied Weight Engineers, 1976.

5. *Geometric Modeling for Computer Aided Design*, J. L. Schwing, Progress Report for NCCI-99, 1987.

6. *On-Line Help Systems: Design and Implementation Issues*, G. Kearsley, R.L. Campbell, J. Elkerton, W. Judd and J. Walker, panel discussion, **Human Facotrs in Computing Systems**, SIGCHI Conference Proceedings, pp. 287 - 289, 1988.

Appendix: *ManualWriter* Report

ManualWriter: A Document Development Environment

By Jan L. Spangler

Master's Project

TABLE OF CONTENTS

ILLUSTRATIONS

ManualWriter: A Document Development Environment

This paper describes a prototype system called ManualWriter, which provides a set of facilities for the authorship, maintenance and display of technical documentation. ManualWriter was designed to support the documentation of the Solid Modeling Aerospace Research Tool (SMART), a conceptual design system being developed at Nasa Langley Research Center.

BACKGROUND

Writing good documentation, especially good user manuals for computer software, is hard. The user manual must provide information to all possible users, ranging from the naive to the expert. Furthermore, these readers will want to use the manual in different ways at different times. Early on they will want summaries and tutorial information; later they may want to browse; finally they may want a reference manual.

Recently there has been an increasing trend to make documentation available on-line. Most of these systems try to take advantage of the interactive environment provided by the computer to tailor the material to the needs of the reader. The recent availability of more powerful workstations with high resolution graphic displays has contributed to the creation of sophisticated document display interfaces.

As a result, during the past year interest in on-line document display systems has accelerated sharply. The electronic documents are often referred to as "hypertext". Most hypertext systems can be characterized by the following features[1]:

* Information is chunked into small units. Each unit may contain textual information. In some systems, units may also contain other forms of information such as graphic images, sound and animation.

* Units of information are displayed one per window.

* Units of information are interconnected by links. Users navigate in a hypertext database by selecting links to travel from unit to unit.

* Users build the hyperdocument by creating, editing, and linking units.

In addition, a browser is an important component of many systems. A browser displays some or all of the hyperdocument as a graph, to show the user which unit he is viewing and how it fits into the larger hyperdocument. Figure 1 which was taken from [2] illustrates how a hypertext browser provides a direct two dimensional view of the database.

SCOPE OF SYSTEM

The objective of ManualWriter is to provide a development environment for the writers of SMART's user documentation. This documentation will include a user's manual, a programmer's manual, and an installation guide. Development of the ManualWriter system began with an examination of the characteristics of these manuals, their life cycles, and the processes used to create them:

* The user and programmer manuals will probably be

longer than 100 pages.

* SMART, which is in its third year of development, is a large system that will continue to evolve. Manuals will need to keep pace with changes made to SMART.

* SMART's software development team (6 people) will write the documentation.

For these reasons, ManualWriter is designed for large documents with long life cycles. There is concern for not only the productivity of the individual, but also the productivity of groups.

Research shows that with large documents most of the effort goes into preparing the content of the document[3]:

> In large documentation projects , the development (writing) stages take over half the project resources, while final production takes much less (less than 10 percent according to some estimates). Therefore, to improve productivity and reduce costs in large documentation projects, you need to concentrate on the development cycle; relatively small gains come from improving the final production cycle.

Therefore, the ManualWriter design effort focused on helping the team of authors to organize their material to be readable in a dynamic way. Production issues (e.g. typesetting, layout) are largely ignored.

REQUIREMENTS ANALYSIS

Requirements for document display include:

* Provide browsing capabilities so that readers can rapidly move among manual sections to find information of interest.

* Use graphics to show interconnections between document sections and overall organization of the document.

* Maintain consistency of format when presenting document sections.

* Allow user to access SMART's user manual while running SMART.

* Maintain consistency between the document display interface and SMART's interface.

* Limit the amount which must be learned in order to use the system and capitalize on those conventions with which the user may already be familiar.

To achieve high productivity, authors must be able to enter material into the document display system easily, quickly and in a natural way. To provide this capability the authoring environment should have the following features:

* A database for storing documents.

* A set of cut/paste operations which allow authors to change or add to the document's hierarchical structure.

* A set of templates for entering text so that consistency is maintained among document sections.

* Tools to support group collaboration.

* An on-line help facility.

DESIGN DECISIONS

Configuration

The ManualWriter documentation system runs on the same hardware used by SMART - the Silicon Graphics IRIS 2400, a standalone high-performance graphics workstation. One disadvantage of installing ManualWriter on the IRIS 2400 is that the workstation is a scarce resource which is already in great demand by SMART's end-users and the software development team. Adding a document development package increases the machine's already heavy workload. However, two criteria dictate use of the IRIS workstation:

* It is desirable that the documentation system's interface be consistent with that of SMART. SMART's method of display interaction requires the capabilities of the IRIS workstation.

* The end user should have the capability of accessing the user manual from within the SMART application.

Both SMART and ManualWriter are written in the C language using a UNIX operating system.

Architecture

The architecture of the documentation system is diagrammed in Figure 2. The central component is the documentation itself, which is arranged in a database of independent records.

This database is created and maintained using ManualWriter. End-user retrieval from the database is handled using ManualBrowser, the interface for on-line document display. The database can also be printed as conventional paper documents.

Document Database

The document is organized as a hierarchical database of interconnected records. A record is also referred to as a node. Each node is a unit of information retrievable from the database. Writers create a separate node for each section of the document or any item of information that the reader might need to access directly. Nodes contain the subject matter from which documents are constructed. A document is formed by linking nodes together.

The underlying data structure of a ManualWriter document is a tree. Most other hypertext systems (e.g. Xanadu, Hyperties, Texnet, and NoteCards) use a directed graph (network) to structure the document. One advantage of a directed graph is that a network structure allows more flexible cross-referencing among nodes than does a tree. On the other hand, trees provide a more natural structure for organizing levels of abstraction. Furthermore, the command language for navigation in a tree-oriented system is simple[2]:

> From any node, the most one can do is go to the parent, a sibling, or a child. This simplicity also diminishes the disorientation problem, since a simpler cognitive model of the information space will suffice.

Perhaps the most decisive factor in choosing a hierarchical organizational scheme for ManualWriter is that a sophisticated graphical interface for manipulating tree structures had already been developed for the SMART application For a description of SMART's tree-based interface, readers are referred to [4].Several benefits accrue from adopting a similar tree-oriented organization for ManualWriter's database:

* Users need only learn one organizational scheme since both systems use trees to structure data.

* Software created for SMART can be re-used in ManualWriter.

* The authors (SMART's programming team) are familar with the tree-editing operations which are needed to maintain a hierarchical database.

Logically the database consists of any number of documents, where each document contains a number of document sections. Physically there is a UNIX directory for each document. The contents of a document's directory are listed below (see Figure 3):

* A "configuration" file containing the names of every file in which a section of the document's text has been stored. In addition, the configuration file stores the links which interconnect the document sections. Status information about each document section (text file) is also recorded: 1) a short description of the text; 2) the author's name; 3) whether the file contains no text, a rough draft, a smooth draft, or final copy; and 4) the format of the document section.

* A text file for each section entered into the document.

* A "ScrapBook" subdirectory containing document sections that have been written but not yet entered into the document's hierarchical structure.

End-user Delivery Interface

ManualWriter produces documents which can be displayed at a terminal or printed on paper. On-line delivery is more important because it provides more flexible access to the documentation.

On-line delivery of the document database is managed by ManualBrowser. From within the SMART application users access the facilities of ManualBrowser to display SMART's user manual. Writers often use ManualBrowser while in the authoring environment to see how a particular section will appear to its readers.

USER INTERFACE

Screen Layout

The screen is divided into windows that are used to display different kinds of information (see Figure 4):

* The main section of the display provides two views of the document being worked on. The document is represented as a tree graph. Each node is labeled by the title of the document section that it represents. The left view displays the entire table-of-contents tree along with a view rectangle. The right view contains that portion of tree contained in the view rectangle scaled larger. Thus, the two

views are respectively referred to a the "table-of-contents view" and the "detailed view".

* The bottom left region is used to display status information about the document: title, supervisor, stage of development (rough draft, smooth draft, or final copy), last modification, size of the database, directory where it is stored. This region is also used to display error messages.

* The menu bar at the top of the screen presents the authoring tools provided by ManualWriter. Some facilities (Retrieve, Author, Organize, and Review) have lower level operations which are accessed from pull-down menus.

Navigating

The user denotes where an action is to occur by clicking on the appropriate node in the "detailed view" of the table-of-contents tree. When a document becomes large, only a portion of the tree can be viewed in detail at one time. The simultaneous presentation of global and close-up views ensures that users are always aware of where in the overall document a magnified subtree fits. The mouse buttons are used to control what portion of the document is represented in the "detailed view".

CREATING AND EDITING DOCUMENTS

In creating documents, writers must manage four different kinds of information:

structure - the organization of material into a chapter-section-subsection hierarchy;

content - the subject matter of the document;

format - the appearance of the document; and

status - information needed to supervise the document development process.

ManualWriter facilities allow authors to consider each aspect of the document relatively independently of the others.

Organizing the document

The writer imposes structure on the document by directly manipulating the table-of-contents tree. A set of standard cut/paste operations are used to edit the tree's structure. The user denotes where the operation is to occur by moving the cursor to the appropriate node in the tree. The level of object (chapter, section, sub-section) to be operated upon can be inferred from the node's position in the tree diagram. Being able to see the tree helps in visualizing the structure of the document.

The user can move a whole sub-tree of the document simply by cutting it's root node from the tree and then pasting it back in a new location. As a result, the user can quickly evaluate a number of different organizations for material.

ManualWriter offers the writers much flexibility in their approach to the authoring process. Writers can construct the document in a top-down fashion by using the structure editing facilities to define a framework of chapters and sections before any text has been entered for them. This framework can be easily restructured during document composition. ManualWriter also supports a bottom-up approach to document development. Parts of the document can be written before knowing exactly where they are going to be placed in the document. Sections that have not been linked to the overall document are stored in a "ScrapBook"; later these sections can be cut from the ScrapBook and pasted at a specific position in the table-of-contents tree.

For a more complete description of the cut/paste commands, the reader is referred to Manual Writer's user manual (see Appendix).

Content Editing

Text is added to a node via "template editor". A template is a predefined set of sub-titles. The template editor allows the user to enter up to one screenful of text under each subtitle. The purpose of the template editors is to maintain a consistency among document sections. Listed below are the templates which are available:

templates	subtitles
User Manual Page	Button, Purpose, Usage, Description, Comments
Prog Manual Page	Name, Specification, Description, See Also, Notes
Free Form	user defines own subtitles

The template editor is screen oriented and employs a user interface similar to the UNIX vi editor. It was implemented using the UNIX curses package.

Figure 5 shows the template editor. Notice that a summary of the editing commands appear in the right-hand panel. Also displayed is status information concerning the document section being edited (lower left-hand corner).

Supervising the Document's Development

Manual Writer provides a set of operations for managing the document development process:

* The "display information" operation causes status information on on indicated document sections to appear. The user indicates which sections are of interest simply by clicking on nodes in the table-of-contents tree. The information shown includes : title, author, UNIX owner and group, permissons, last modification and draft status.

* The "change locks" operation allows a section to be specified as read only (locked). A color-coded tree is presented in which the nodes of locked document sections are colored green and unlocked sections are lilac. The user toggles the section between locked and unlocked simply by clicking on its node.

* An operation is provided for changing the individual or group owner of a document section.

* Writers record a document section's current stage of completion using the "change draft" command. These stages are: blank, rough draft, smooth draft, final copy. Each node in the table-of-contents tree is color-coded to reflect its respective draft status.

VIEWING A DOCUMENT

The interface to on-line documentation, called ManualBrowser, is conceptually similar to SMART's interface for displaying design configurations. Dual viewports display the two views of the table-of-contents tree. The reader navigates through the document by using mouse buttons to control the portion of the tree displayed in the detailed view. Once the relevant topic is located, the user clicks on its node. The textual content of the section is displayed (see Figure 6). Subtitles and keywords are highlighted. A mouse-sensitive control panel allows the user to either scroll forwards or return to the table-of-contents tree.

Topics are rarely independent, and part of using a manual well lies in deciding what to read. The tree shows the local context of a section, in what part of the document it appears, and what other sections are nearby in the document.

The reader can produce hard copy of any section in the database. The print utility does a walk of the tree that represents the document starting from the selected node and going down the hierarchy. ManualWriter's user manual which is provided as an appendix is an example of hard copy produced by the system.

EVALUATION

The first test of ManualWriter has been in developing an on-line help facility for the system. The documentation group consisted of one person, the author of this paper. I found ManualWriter's approach to document development particularly successful in the following areas:

* It is easy to try out different ways of organizing the document's content.

* The system's modular approach to document development helps the writer break down the writing task into manageable units.

* With the IRIS workstation's display hardware and the ManualBrowser interface, on-line delivery is an acceptable alternative to paper.

The most frustrating thing about using the system is its limited capabilities for editing document content (i.e. the small set of commands for editing text and lack of tools for creating diagrams.)

REFERENCES

1. Akscyn, Robert M., McCracken, Donald L., and Yoder, Elise A. "KMS: A Distributed Hypermedia System for Managing Knowledge In Organizations," Communications of the ACM31, 7(July 1988), 820 - 835.

2. Conklin, E.J. "Hypertext: an introduction and survey," IEEE Computer 2, 9(September 1987), 17 - 41.

3. Walker, Janet H. "Document Development with Concordia," IEEE Computr, (January 1988), 48 - 59.

4. Schwing, James L. and Spangler, Jan. "Geometic Modeling for Computer-Aided Design," (December 1986).

ILLUSTRATIONS

ORIGINAL PAGE IS
OF POOR QUALITY

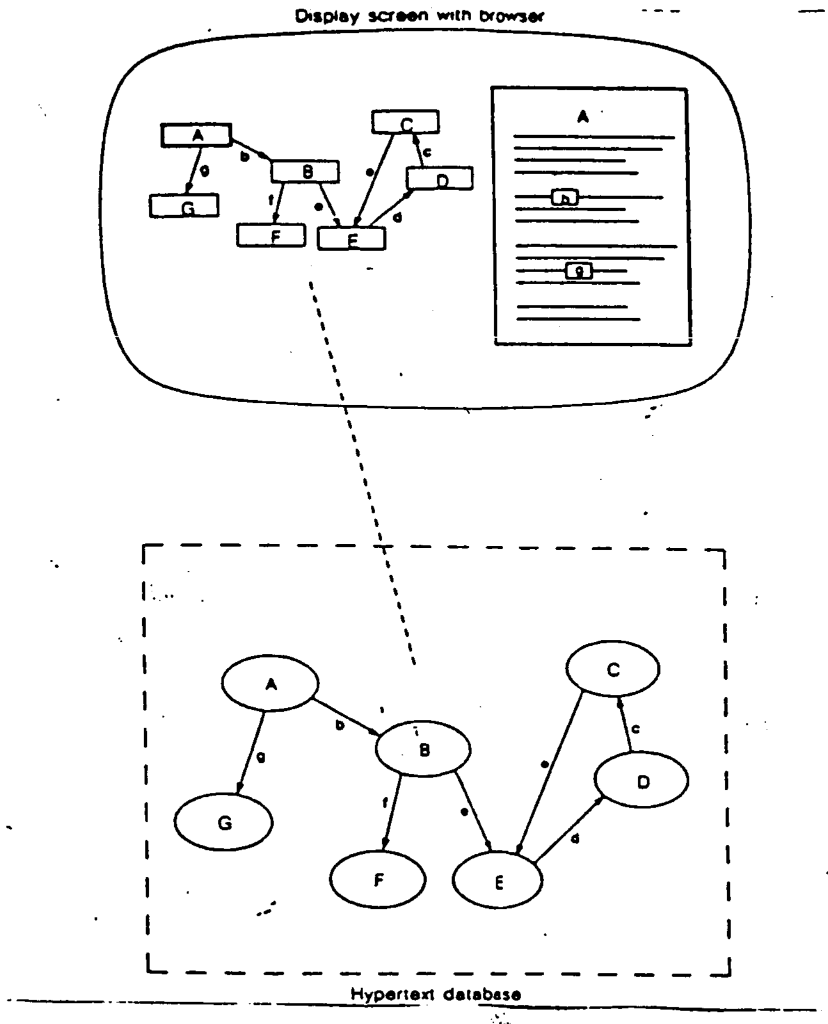


Figure 1. The screen at the top illustrates how a hypertext browser provides a direct two-dimensional graphic view of the underlying database. In this illustration, the node "A" has been selected for full display of its contents. Notice that in the browser view you can tell not only which nodes are linked to A but also how the subnetwork fits into the larger hyperdocument. (Of course, hyperdocuments of any size cannot be shown all at once in a browser—only portions can be displayed.)

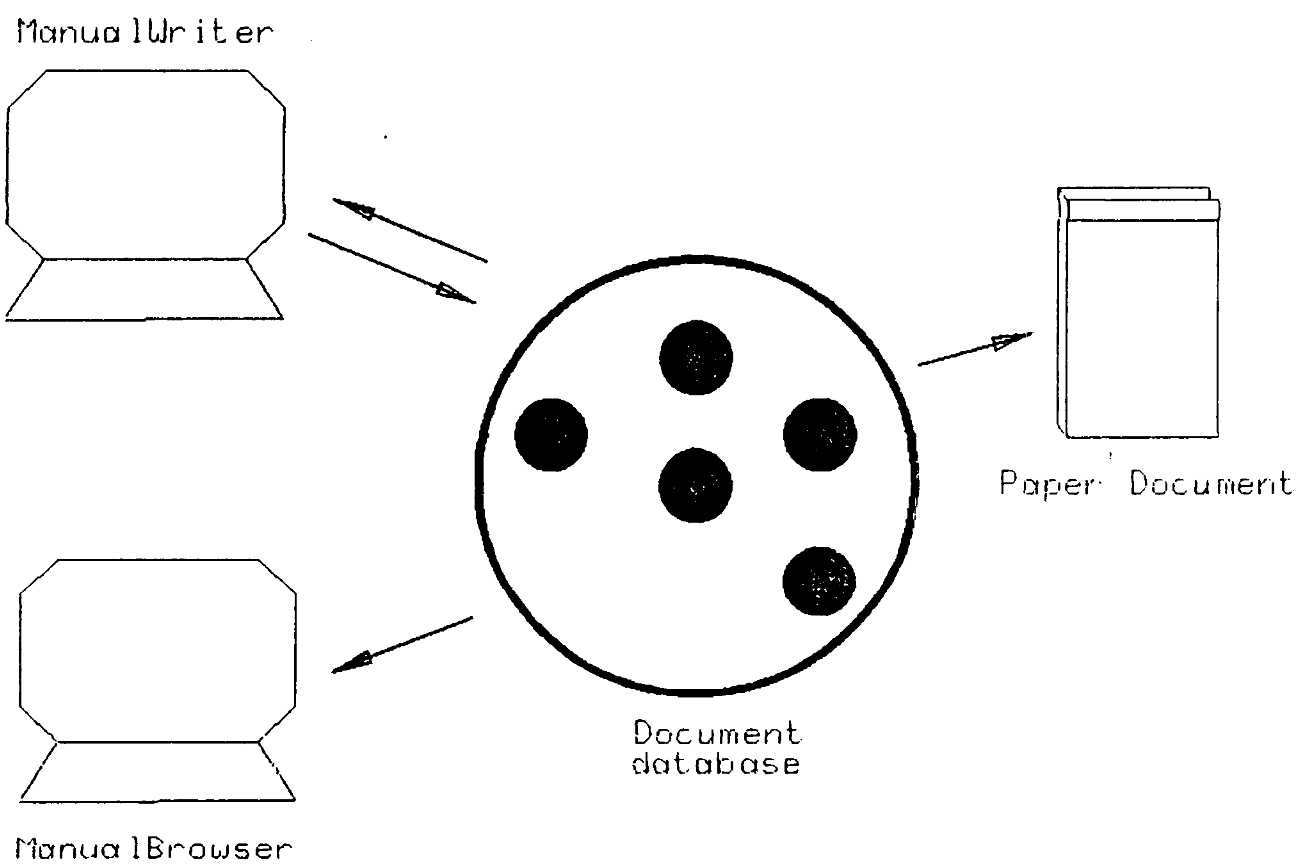


FIGURE 2. Documentation system architecture.

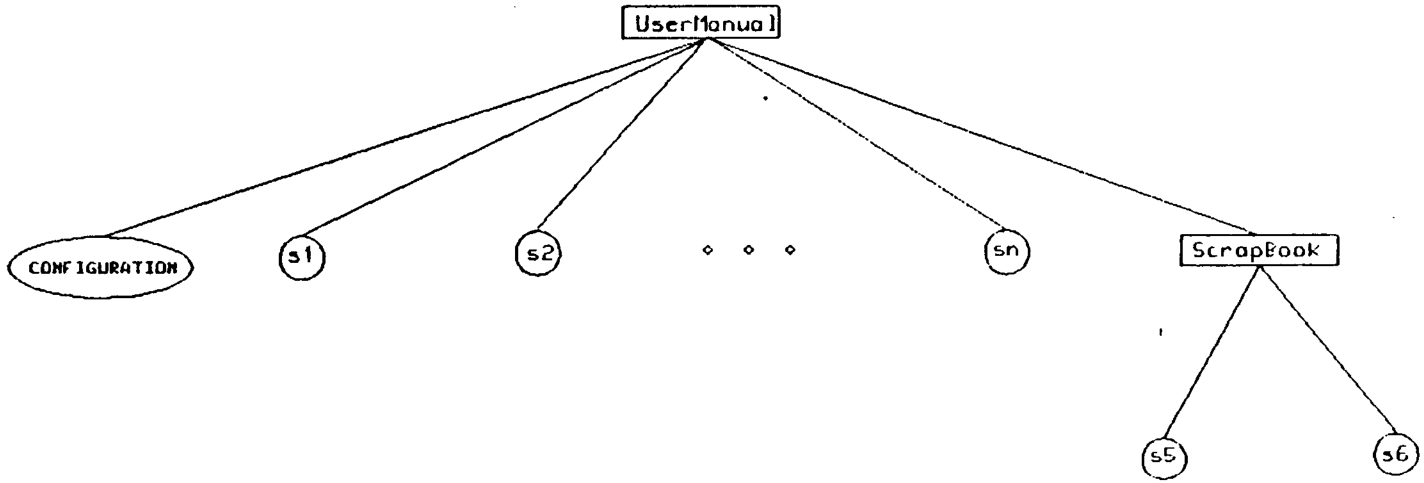
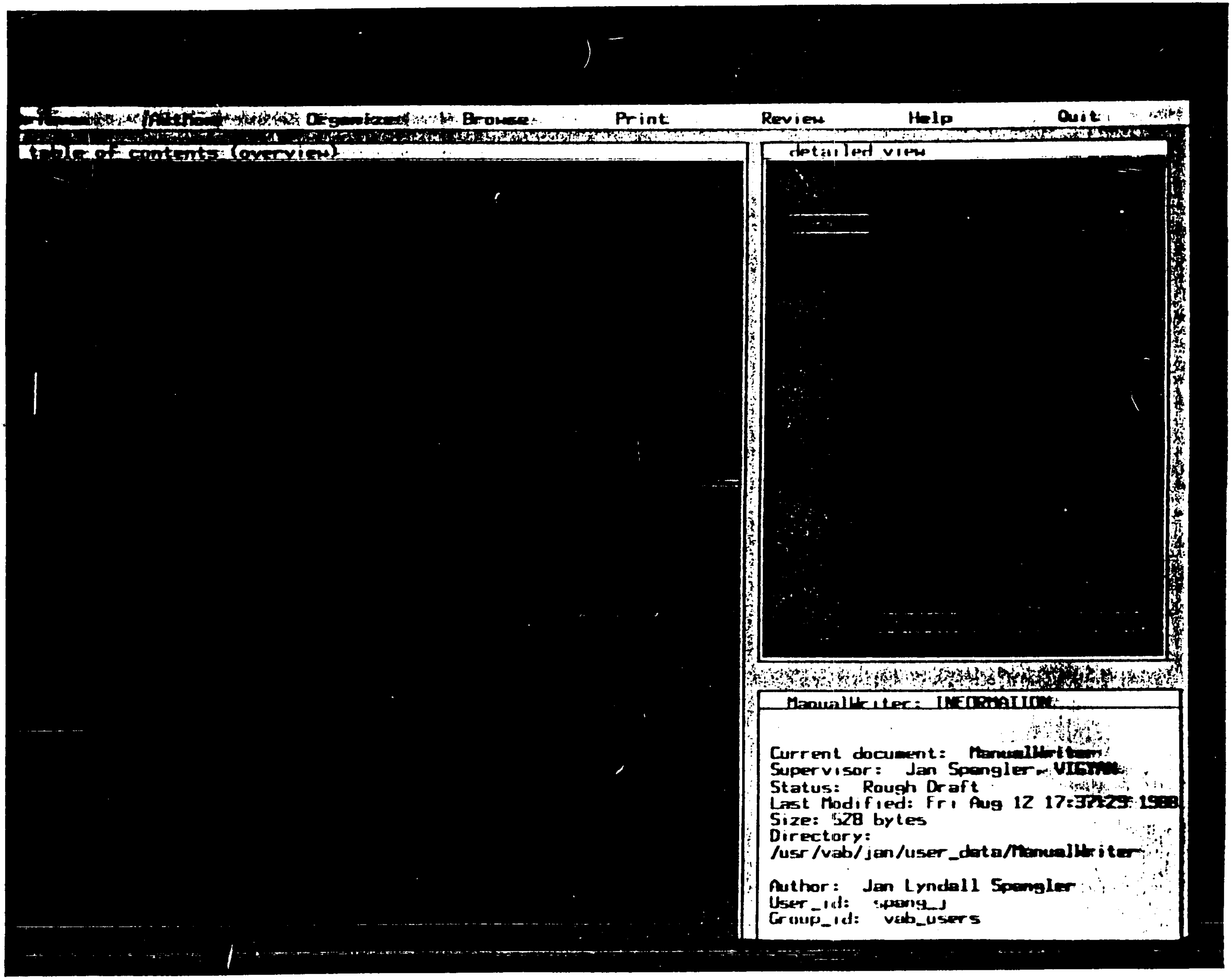


FIGURE 3. The UNIX file directory system serves as a free mechanism for creating hierarchical structures among nodes:

* CONFIGURATION is a file which stores the UserManual's structure.
* s1 ... sn are files containing the document's textual content.
* The ScrapBook directory stores sections that are not currently entered into the UserManual's section/subsection hierarchy.



4 ManualWriter's table-of-contents tree

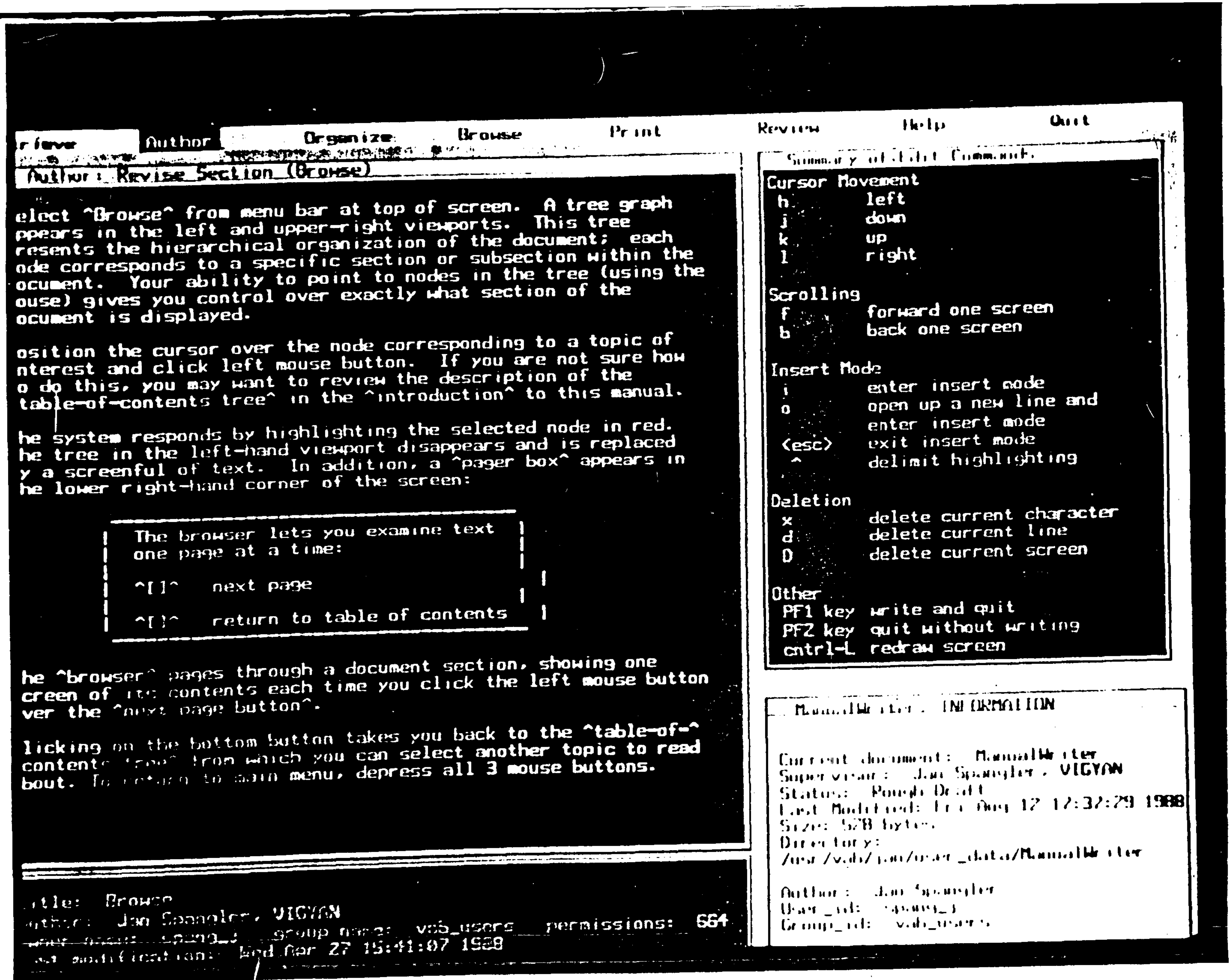
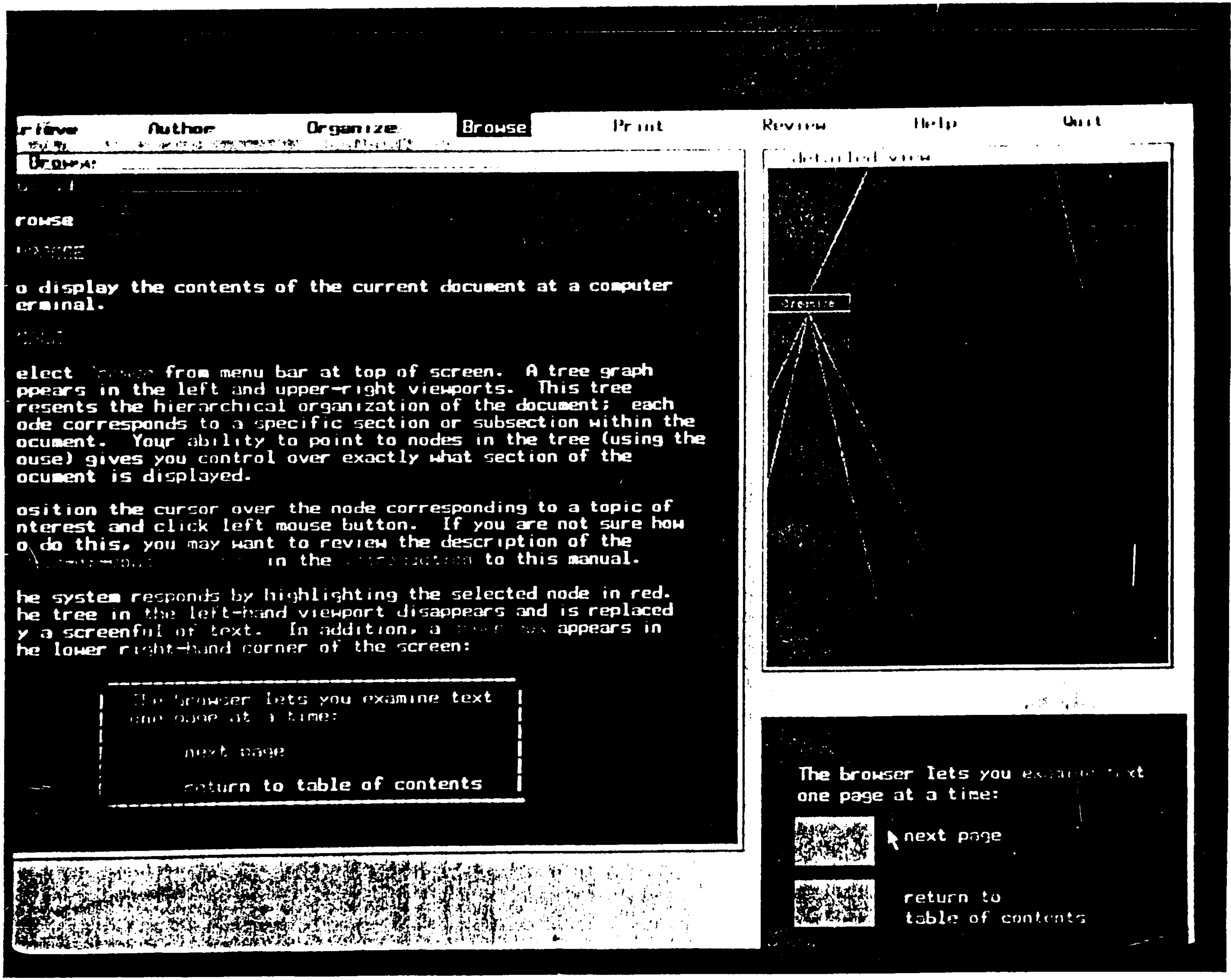


FIGURE 5 Template Editor


Author
Organize
Browse
Print
Review
Help
Quit
rowse
o display the contents of the current document at a computer
erminal.
elect from menu bar at top of screen. A tree graph
ppears in the left and upper-right viewports. This tree
resents the hierarchical organization of the document; each
ode corresponds to a specific section or subsection within the
ocument. Your ability to point to nodes in the tree (using the
ouse) gives you control over exactly what section of the
ocument is displayed.
osition the cursor over the node corresponding to a topic of
nterest and click left mouse button. If you are not sure how
o do this, you may want to review the description of the
in the to this manual.
he system responds by highlighting the selected node in red.
he tree in the left-hand viewport disappears and is replaced
y a screenful of text. In addition, a appears in
he lower right-hand corner of the screen:
The browser lets you examine text
one page at a time:
next page
return to table of contents
The browser lets you
one page at a time:
next page
return to
table of contents