협업 설계에서의 다중해상도 모델링 응용

김 태 성† · 한 정 현††

요 약

본 논문은 역할 기반 뷰잉이라는 방법을 기반으로 하여 협업 설계에서의 정보 보호에 대한 기본 구조를 제안한다. 역할 기반 뷰잉은 다중 해상 기하 모델과 보안 모델을 조합하여 달성된다. 주어진 3차원 모델은 기하적으로 분할되며, 분할된 각 모델을 이용하여 다중 해상 메쉬 계층 구조가 생성된다. 협업 설계 환경에서 각 디자이너의 접근 권한에 알맞은 모델의 생성은 접근 통제 방법에 의해 이루어진다.

Application of Multi-Resolution Modeling in Collaborative Design

Taeseong Kim† · JungHyun Han††

ABSTRACT

This paper provides a framework for information assurance within collaborative design, based on a technique we call role-based viewing. Such role-based viewing is achieved through integration of multi-resolution geometry and security models. 3D models are geometrically partitioned, and the partitioning is used to create multi-resolution mesh hierarchies. Extracting a model suitable for access rights for individual designers within a collaborative design environment is driven by an elaborate access control mechanism.

키워드 : 협업/분산 설계(Collaborative/Distributed Design), 접근 통제(Access Control), 다중해상도 모델링(Multi-resolution Modeling), 역할 기반 뷰잉(Role-based Viewing)

1. Introduction

Information assurance (IA) refers to methodologies to protect and defend information and information systems by ensuring their availability, confidentiality, integrity, non-repudiation, authentication, access control, etc. [1] In collaborative design, IA is mission-critical. Suppose a team of designers working collaboratively on a 3D assembly model. Each designer has a different set of security privileges and no one on the team may have the "need to know" the details of the entire design. In collaboration, designers must interface with others' components, but do so in a way that provides each designer with only the level of information he or she is permitted to have about each of the components.

Among various issues in IA, *access control* is critical for the purpose. We will present a combination of *multi-resolution geometry* and *access control models*. (Figure 1) illustrates the conceptual architecture of the secure collaborative design system, which can be stated as follows :

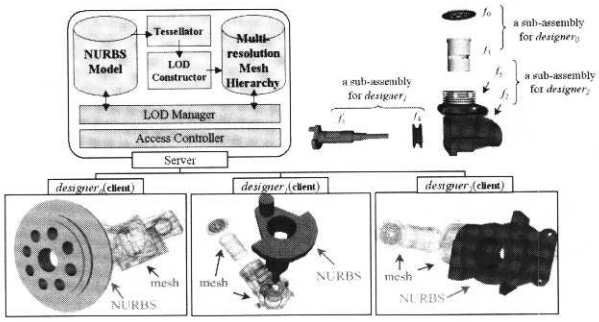
- An assembly model consists of a set of component parts, possibly grouped into sub-assemblies.
- Each component part is represented as a NURBS-based boundary model.
- Design is performed collaboratively by a group of designers working on different (possibly geographically distant) workstations. A standard client-server architecture is assumed, where the *collaborative CAD server* maintains and synchronizes the master design model.
- The collaborative CAD server tessellates the master model into polygon meshes. Multi-resolution mesh hierarchies are constructed, which are used in generating *role-based views* for designers working at the client hosts.
- Depending on the accessibility privilege of a designer, an appropriately simplified model is extracted from the multi-resolution hierarchies, and provided to the designer at the collaborative CAD client. In (Figure 1),

※ This work was supported by Grant No. 1999-2-515-001-5 from the Basic Research Program of the Korea Science & Engineering Foundation.
† 준 회 원 : 성균관대학교 대학원 정보통신공학부
†† 종신회원 : 성균관대학교 정보통신공학부 교수
논문접수 : 2003년 6월 16일, 심사완료 : 2003년 10월 1일

the client side NURBS model being designed is shaded whereas the other components (in meshes) of the master model are rendered as wire frames, just for illustration.

- When a component part or sub-assembly gets modified, the server reconstructs only the corresponding (changed) portion of the hierarchy, and then passes these updates to the other clients according to their accessibility privileges.

Aside from digital 3D watermarking [18-20], research on how to provide IA to distributed collaborative designers is largely non-existent. The authors believe that this work represents the first attempt to provide IA to computer-aided design and collaborative engineering.



(Figure 1) Secure Collaborative Design System Architecture

2. Related Work

There has been a vast body of work on concurrent engineering and collaborative design. The existing work most relevant to our efforts deals with real-time 3D collaboration and communication. Distributed Virtual Environments (DVEs) [2-6] have been developed for real-time interactions between distributed collaborators in a number of different domains. Immersive environments such as CAVE [7] have been developed which also support real-time interaction, but they do not necessarily support collaborative CAD.

This paper focuses on *access control* which is the process of limiting access to resources of a system only to authorized users, programs, or processes, and therefore preventing activity that might lead to a breach of the system's security. In CAD and collaborative design contexts, a most relevant work in the domain of collaborative assembly design can be found in Shyamsundar and Gadh [8]. Their work could be taken as a simple implementation of information-hiding techniques, but lacks an elaborate access control mechanism. Further, it will be desirable to provide finer-grained levels of detail than envelops.

Polygon meshes lend themselves to fast rendering algorithms, which are hardware-accelerated in most platforms. Many applications, including CAD, require highly detailed models to maintain a convincing level of realism. However, the number of polygons is often greater than that we can afford. Therefore, *mesh simplification* is adopted for efficient rendering, transmission, and various computations. The most common use of mesh simplification is to generate *multi-resolution* models or various *levels of detail* (*LOD*). A most recent survey on mesh simplification can be found in [9].

The most popular polygon-reduction technique is *edge collapse* or simply *ecol* (more generally, vertex merging or vertex pair contraction) where two end vertices are collapsed into a single one. Repeated applications of *ecol* generate a simplified mesh. *Vertex split* or simply *vsplit* is the inverse operation of *ecol*. Hoppe proposed *progressive mesh* (*PM*) [10], which consists of a coarse base mesh (created by a sequence of *ecol* operations) and a sequence of *vsplit* operations. Applying a subset of *vsplit* operations to the base mesh creates an intermediate simplification. The *vsplit* and *ecol* operations are known to be fast enough to apply at runtime, therefore supporting dynamic simplification.

Previous works on mesh simplification and LOD techniques often mention the possibility of applying the techniques to collaborative design. To date, however, their use has been limited to the areas such as redundant geometry reduction, realtime rendering, streaming 3D data over the network, etc.

3. Role-based Viewing

In *role-based viewing*, each user sees a shared 3D model in a distinct resolution, which is determined by the user's *role*.

3.1 Role-based View

A *role-based view* is a tailored 3D model which is customized for a specific user based on the roles defining the user's access permissions on the model. In this way, the role-based view does not compromise sensitive model information which the user is not allowed to see (or see in detail).

Consider the component f_0 in (Figure 1), which is being

edited by *designer*$_0$. Suppose that *designer*$_0$ wants to hide the design details of f_0 from other participating designers, i.e. *designer*$_1$ and *designer*$_2$. Our solution to the problem is to present f_0 to them in some "lower" resolutions. (Figure 2) shows three different resolutions or LODs of f_0. (Figure 2)(a) is a full-resolution model, which *designer*$_0$ sees and may also be presented to, for example, project supervisors.

The set of holes in f_0 might be critical features which *designer*$_0$ wants to hide from *designer*$_1$. Then, all holes are removed from the original model, and the model in (Figure 2)(b) is presented to *designer*$_1$. Suppose that *designer*$_2$ is a supplier from another organization. Then, the model in (Figure 2)(b) can be again simplified to generate the crude model in (Figure 2)(c), which just presents the outline of f_0 to *designer*$_2$. Those are examples of role-based views.

(a) Original model (for *designer*$_0$) (b) Genus-reduced model (for *designer*$_1$) (c) Simplified Model (for *designer*$_2$)

(Figure 2) Role-based View Examples of f_0

3.2 Access Control Policies

Existing *access control policies* are briefly surveyed in this subsection. Access control policies commonly found in contemporary systems can be classified as follows [11].

- Discretionary Access Control
- Mandatory Access Control
- Role-based Access Control

Discretionary Access Control (DAC) was originally introduced by Lampson[12], where the access of a user to an object is governed on the basis of authorizations that specify the access mode (e.g. read, write, or execute) the user is allowed on the object.

Mandatory Access Control (MAC) [13] policies control dissemination of information by associating users and objects with *security levels*. The security level associated with an object reflects the *sensitivity* of the information, i.e. the potential damage that could result from unauthorized disclosure of the information. The security level associated with a user reflects the user's *trustworthiness* not to disclose sensitive information to users not cleared to see it. MAC policies assert that a user can access an object only if the user has a security level higher than or equal to that of the object. For example, suppose that the security levels consist of Top Secret (TS), Secret(S), Confidential(C), and Unclassified(U), and that TS > S > C > U, where > denotes "has a higher security level than." This is often called the "read down" principle.

In Role-Based Access Control (RBAC) [14], system administrators create *roles* according to the job functions in an organization, grant permissions (access authorizations) to the roles, and then assign users to the roles. The permissions associated with a role tend to change much less frequently than the users who fill the job function that role represents. Users can also be easily reassigned to different roles as needs change. These features have made RBAC attractive, and numerous software products such as Microsoft's Windows NT currently support it.

Our security framework is essentially based on embodiment of a MAC policy within an RBAC framework. *Roles*, $R = \{r_0, r_1, \cdots, r_m\}$, are abstract objects that define both the specific users allowed to access resources and the extent to which the resources are accessed.

The engineers (designers, process engineers, project supervisors, etc.) correspond to a set of *actors* $A = \{a_0, a_1, \cdots, a_n\}$, each of which will be assigned to a set of roles. *Actor-Role Assignment*, AR, is a many-to-many relation of actors to roles : $AR \subseteq A \times R$.

The entire assembly design is represented as a *solid model* M. A collaborative engineering environment enables multiple engineers (actors) to simultaneously work with M. Let $b(M)$ represent the boundary of M. *Model-Role Assignment*, MR, is a many-to-many relation assigning points on $b(M)$ to roles : $MR \subseteq b(M) \times R$, where each point on $b(M)$ is assigned to at least one role, i.e., $\forall p \in b(M)\ \exists\ r \in R,\ (p,\ r) \in MR$.

It is impractical to assign $b(M)$ to roles point-by-point. Hence, we define a set of *security features*, $SF = \{f_0, f_1, \cdots, f_k\}$, where each f_i is a topologically connected point set on $b(M)$ and $\cup\ SF = b(M)$. The Model-Role Assignment can then be simplified to be the relation associating security features with roles : $MR \subseteq SF \times R$.

Partitioning $b(M)$ into security features SF can be done either by the project supervisor (working as an administrator) or by the designers in charge of the components or sub-assemblies to be partitioned. In (Figure 1), the assembly model is partitioned into 6 security features f_0, f_1, f_2, f_3, f_4 and f_5, where $\{f_3, f_4, f_5\}$ is a set of mating features.

3.3 Access Matrix

Access matrix is a popular *conceptual model* that specifies the rights that each user possesses for each object. In a large system, the access matrix is usually enormous in size and sparse. Therefore, compact access control lists (ACL) are often used to implement the access matrix.

In the collaborative CAD context, however, an access matrix is constructed and maintained "for each design session," and consequently the matrix is dense because every component/sub-assembly is supposed to be visible to virtually all participating designers (probably in different LODs). Therefore, we adopt a matrix implementation as illustrated in (Figure 3), which is for the collaborative assembly design example in (Figure 1). There is a row in this matrix for each role, and a column for each security feature. For simplicity, only three roles, r_0, r_1 and r_2, are created.

	f_0(TS)	f_1(C)	f_2(S)	f_3(U)	f_4(U)	f_5(U)
r_0(TS)	w	r	r	r	r	r
r_1(S)	r	w	r	r	r	r
r_2(C)	r	r	w	r	r	r

(Figure 3) Access Matrix

Such an access matrix is obviously an RBAC implementation. To embody a MAC policy in it, let us associate both roles and security features with *security levels* using the simple hierarchy of TS > S > C > U. (In fact, boundary partitioning is followed by associating each feature with a specific security level.)

Each cell of the access matrix distinguishes between *read* and *write* authorizations. It is reasonable to assume that write permission of a feature is exclusively given to a single role. In contrast, read permissions of a feature should be given to all roles. For the remainder of this paper, we focus on read permissions.

A typical scenario within this RBAC + MAC system would be that, for example, a C-level feature is visible to S-level role whereas a TS-level feature is invisible. Rather than this "all or nothing" read permissions, our objective is to assign a "continuous" *degree of visibility* between a feature and a role, i.e. the method presented in this paper may generate a "full" resolution version of the C-level feature and a "lower" resolution version of the TS-level feature to the S-level role.

The administrator not only constructs the access matrix and registers it into an authorization database, but also performs Actor-Role Assignment *AR*. Suppose that, in the simple example of (Figure 1) and (Figure 3), actors *designer*$_0$, *designer*$_1$, and *designer*$_2$ are assigned to roles r_0, r_1, and r_2 respectively. Let us focus on f_0. The write permission given to r_0 implies the full read permission, regardless of the security levels associated to r_0 and f_0. Therefore, *designer*$_0$ who has the write permission on f_0 sees a full resolution of f_0. This is the view given in (Figure 2)(a). In contrast, *designer*$_1$ takes r_1's security level S, and it is lower than the level TS of f_0. Therefore *designer*$_1$ should see a simplified model. It might be the view given in (Figure 2)(b). Finally, *designer*$_2$'s security level C is far lower than the level TS of f_0, and therefore *designer*$_2$ might see a drastically simplified model, which might be the view given in (Figure 2)(c). Such a "continuous" role-based viewing technique is discussed in the next section.

4. Role-Based View Generation

To an actor a, role-based viewing presents a new model M', which is generated from the original assembly model M such that its security features are appropriately obfuscated based on the actor a's roles. If the roles give the actor full permissions to see certain features, then the resulting model M' includes those features with the same fidelity as in M ; if not, the features must be obfuscated so as to hide from a what a does not have permissions to see.

The input to role-based viewing consists of an actor a, the Actor-Role Assignment (AR), access matrix, and multi-resolution mesh hierarchies for the entire assembly.

4.1 Multi-resolution Mesh Hierarchy

Numerous mesh simplification approaches have been proposed in computer graphics literature. In our system, an object's topology is less important than its overall appearance for rendering. We also need an algorithm capable of drastic simplification since runtime performance is crucial. Therefore, topology-modifying simplification is a reasonable choice. Further, topology modification such as *genus reduction* often plays an important role in hiding the design-detail of a component/sub-assembly.

In a collaborative design system where a number of designers collaborate simultaneously, it is more storage-efficient to have a single dynamic/continuous hierarchy rather than multiple discrete LODs. Further, an appropriate LOD need often be transmitted to each client depending not only on each designer's access privilege but also on each

client's computing capability. A continuous hierarchy guarantees extremely fine granularity in the sense that a distinct LOD can be presented to each actor. Therefore, the progressive mesh(PM) discussed in Section 2.3 is a reasonable choice.

4.2 Genus Reduction in Feature-based Design

A problem of PM is that it assumes manifold topology, and consequently is not compatible with topology-modifying simplification. Its solution can be found by utilizing *feature-based design* capabilities, which most of contemporary CAD systems support.

Let us consider solid modeling. Features are classified into positive/additive and negative/subtractive features. The negative features lead to depressions such as holes. In the first stage of our simplification process, such negative features may be removed from the original model, and then topology-preserving simplification (*ecol*) is applied at the second stage. Note that the topology-preserving simplification enables drastic polygon reduction because genus is reduced at the first stage. Such an integration of feature-based genus reduction and topology-preserving simplification is much faster than topology-modifying simplification algorithms such as [15]. (Figure 2)(b) shows a model with negative features removed, and (Figure 2)(c) shows the result of applying mesh simplification to the model in (Figure 2)(b).

4.3 Role-based Viewing integrated with MAC

A role-based view is generated "security features by features." We distinguish between *genus-reducible* security features from others. In the context of feature-based design, for example, a security feature is genus-reducible if it contains a non-empty set of negative design features whose dimensions are below some predetermined threshold values. For a genus-reducible security feature, two mesh data structures are constructed : one is a plain mesh for the entire security feature, and the other is a PM of the genus-reduced model. If a security feature is not genus-reducible, it is just represented as a PM.

We have a PM per a security feature. As discussed in Section 2.3, a PM data structure consists of a base mesh and a list of *vsplit* nodes. The *vsplit* list can be conceptually illustrated as a forest of binary vertex trees as shown in (Figure 4)(a). Each PM node corresponds to a vertex. Therefore, a *vsplit* operation splits a vertex into two new vertices corresponding to its two children.

The problem of how much of a security feature is made visible to a role is reduced to the task of what subtrees of its PM to select, or how to choose a "vertex front"[16] of the PM. (All vertices of a simplified mesh extracted from a PM constitute a vertex front in the PM's hierarchical structure, as depicted in (Figure 4)(b) and (Figure 4)(c)). The solution to the task requires understanding of the mesh simplification method we adopted.

Garland and Heckbert [17] proposed a mesh simplification algorithm based on *quadric error metrics* (*QEM*). It proceeds by repeatedly merging vertex pairs, each of which is not necessarily connected by an edge, i.e. it modifies topology. We use a slight modification of the algorithm : QEM coupled with *ecol*, not the general vertex merging. A QEM is associated with each vertex and represents the sum of the squared distances from the vertex to the neighboring triangles.

Error caused by an *ecol* operation is easily obtained by summing the QEMs of the two vertices being merged, and the sum is assigned to the new vertex as a QEM. All *ecol* candidates are sorted in a priority queue, and the simplification algorithm selects the edge with the "lowest error" and then performs *ecol*. The algorithm then updates the errors of all edges involving the merged vertices and repeats the simplification.

As *ecol*s are selected basically in order of increasing errors, the inverse operations *vsplit*s are roughly listed in order of decreasing error values. In PM, all leaf nodes have error 0, and one of root nodes will have the maximum error e_{max}. The range [0, e_{max}] is normalized into the range [0, 1]. Such a normalized error is depicted for each node in (Figure 4). (For implementation purpose, the error values of all root nodes are made 1.00.)

MAC policy allows us to have as many levels of security as needed. Let us denote the highest level as l_{max}, the lowest level as l_{min}, the level assigned to a role as $\underline{l_r}$, and the level assigned to a security feature as l_f. Our MAC policy asserts that, if $l_r < l_f$, the full-resolution version of the feature is presented : ① If the security feature is genus-reducible, the plain mesh for the entire security feature is transmitted. ② Otherwise, the vertex front is formed with all "leaf nodes" of the security feature's PM.

When $l_r < l_f$, the vertex front should be composed of "internal nodes" of PM. Let us define the *degree of visibility* α mentioned in Section 3.3. If $l_r < l_f$, α is set using a *distance metric*, which is defined as follows :

- $(l_f - l_r - 1)/(l_{max} - l_{min})$ if feature-based genus reduction has been performed
- $(l_f - l_r)/(l_{max} - l_{min})$ otherwise

The reason for two metrics will be discussed soon. The point is that, as the second metric clearly says, a larger α value is computed when the distance between l_f and l_r is longer. Obviously, the larger α value is, the lower resolution is required. In fact, degree of visibility is a misnomer, and α actually denotes the degree of *invisibility*.

Note that the α value computed as above is also normalized into the range of [0, 1]. Therefore, it can be directly used to determine the vertex front in PM where *ecol* errors have also been normalized. In the list implementation of PM, simple list operations are invoked to select a subset of *vsplit* nodes whose error values are greater than or equal to α. The base coarse mesh followed by the selected *vsplit* nodes are transmitted to clients, and a simplified mesh is rendered. (Figure 4)(b) shows the vertex front determined by $\alpha = 0.64$, and (Figure 4)(c) by $\alpha = 0.54$. Compare the two vertex fronts. As 0.64 is larger than 0.54, a lower resolution should be presented for the case of $\alpha = 0.64$. Therefore the vertex front of $\alpha = 0.64$ lies higher than that of $\alpha = 0.54$.

There can be many other ways to obtain the vertex front. A simpler way is to make α determine the percentage of *vsplit* nodes. For example, if α is 0.7, 30%(= 1-0.7) of the *vsplit* nodes are selected. However, our experiments showed that the elaborate mechanism based on QEM values leads to "more expectable" degradation of the model fidelity.

Let us now discuss why we need two metrics for α. Suppose that $l_f - l_r = 1$, i.e. the role's security level is just one degree lower than that of the security feature. If feature-based genus reduction has been performed, the PM represents an already-simplified model. Therefore, it is reasonable, when $l_f - l_r = 1$, to present the full PM, i.e. the vertex front should consist of all leaf nodes of PM. It is achieved when $\alpha = 0$. For that purpose, we subtract 1 from $l_f - l_r$ to set $\alpha = (l_f - l_r - 1)/(l_{max} - l_{min})$ to 0.

When the level difference between a role and a security feature is extremely large, we could make the security feature completely deleted or replaced with a simple convex hull or bounding box. For example, if $\alpha = 1$, i.e. if $l_f = l_{max}$ and $l_r = l_{min}$, we could simply make the security feature invisible. It is implementation dependent.

5. Implementation and Results

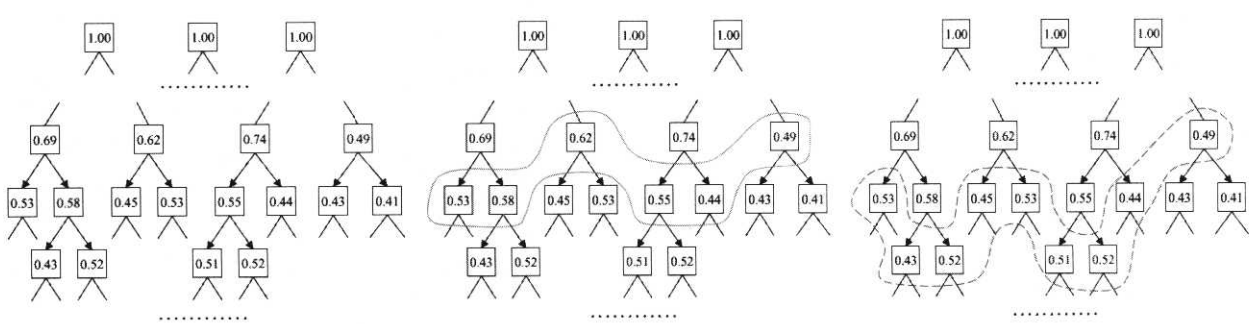
To test the approach we have described in this paper, a prototype system was developed. For collaborative design, it is imperative to make the system independent of platforms and operating systems. The system has been developed using OpenGL on Solaris 2.7~2.8 and Windows, and using both Mesa and nVidia's native OpenGL drivers on Linux operating systems.

Example : Motorcycle Engine Assembly

(Figure 5) shows three role-based views of the motorcycle engine assembly. (Figure 5)(a) and (Figure 5)(b) assume that $designer_0$ is editing a sub-assembly named "cylinder body" and $designer_1$ is editing another sub-assembly named "cylinder head."

(Figure 5)(a) is a view for $designer_0$, the "cylinder block" designer. Suppose $designer_0$ has lower security level than the security levels "crank case" and "cylinder head" have. Therefore, appropriately degenerated resolutions of the components in "crank case" and "cylinder head" are given to $designer_0$. The components in "cylinder body" are presented in full resolutions.

In contrast, (Figure 5)(b) is a view for $designer_1$, the "cylinder head" designer. A number of components in

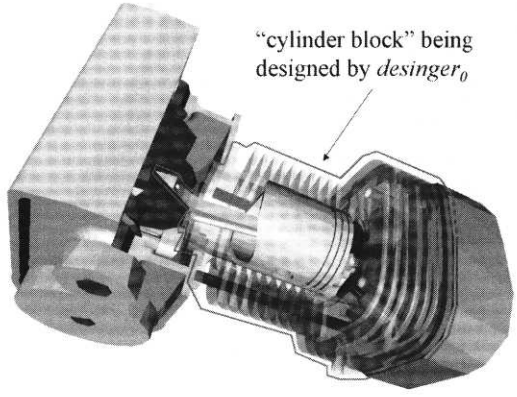


(a) Vertex Tree (b) Boundary Nodes with $\alpha = 0.64$ (c) Boundary Nodes with $\alpha = 0.36$

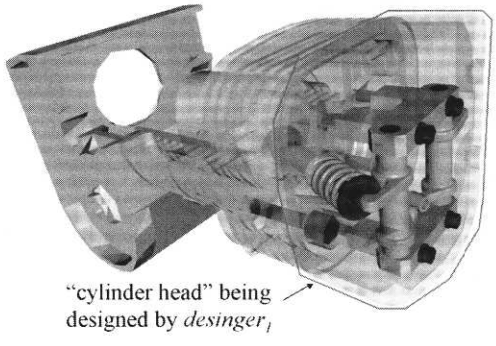
(Figure 4) Vertex Tree and Boundary Nodes Examples

"crank case" are completely deleted in the figure. It is because *designer*$_1$ is associated with the minimum security level whereas the hidden components are associated with the maximum level.

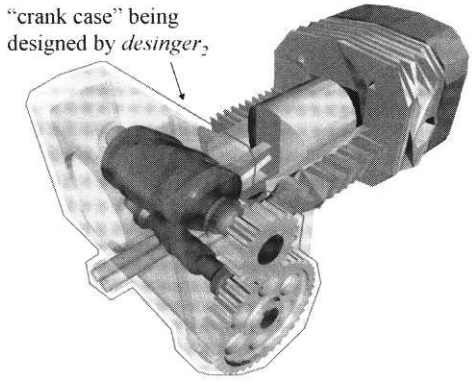
We assume that *designer*$_2$ is a member of different team from the one *designer*$_0$ and *designer*$_1$ belong to. The *designer*$_2$ is in charge of only the components in "crank case." (Figure 5)(c) is a view for the *designer*$_2$, and therefore all components except the ones in "crank case" are presented in low resolutions.



(a) Role-based View for *designer*$_0$



(b) Role-based View for *designer*$_1$



(c) Role-based View for *designer*$_3$

(Figure 5) Motorcycle Engine Assembly Examples

6. Conclusions and Future Work

This paper has presented a new technique, *role-based viewing*, for collaborative 3D assembly design. By incorporating security with collaborative design, the costs and risks incurred by multi-organizational collaboration can be reduced. Aside form digital 3D watermarking, research on how to provide security issues to distributed collaborative design is largely non-existent. The authors believe that this work is the first of its kind in the field of collaborative CAD and engineering.

Our present implementation focuses on meshes only. In the same access control framework, however, we can augment meshes with NURBS. Then, multi-resolution analysis (MRA) based on wavelet decomposition would be needed. With this extension, a low resolution NURBS model is transmitted to clients, and it enables *design comments* or *design suggestions* from the other designers : They can suggest some design changes to the owner of the model, for example, by moving some control points of the NURBS model.

References

[1] William Stallings, *Network and Internetwork Security*, IEEE Press, 1995.

[2] John W. Barrus and Richard C. Waters. Locales : Supporting Large Multiuser Virtual Environments. *IEEE Computer graphics and Application*, 16(6), pp.50-57, 1996.

[3] J. M. S. Dias, R. Galli, A. C. Almeida, C. A. C. Bello and J. M. Rebordao, mWorld : A Multiuser 3D Virtual Environment, *IEEE Computer Graphics and Applications*, 17(2), pp.55-65, 1997.

[4] H. Eriksson, MBONE : The Multicast Backbone. *Communications of the ACM*, 37(8), pp.54-60, August, 1994.

[5] Sankar Jayaram, Uma Jayaram, Yong Wang, H. Tirumali, K. Lyons and P. Hart, VADE : a Virtual Assembly Design Environment, *IEEE Computer Graphics and Applications*, 19(6), 1999.

[6] M. Macedonia, M. Zyda, D. Pratt, P. Barham and S. Zeswitz, NPSNET : A Network Software Architecture for Large-Scale Virtual Environments, *Presence*, 3(4), pp. 265-287, 1994.

[7] Carolina Cruz-Neira, Daniel J. Sandin and Thomas A. DeFanti. Surround-screen projection-based virtual reality : the design and implementation of the cave, In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, ACM Press, pp.135-

142, 1993.

[8] N. Shyamsundar and R. Gadh, Internet-based Collaborative Product Design with Assembly Features and Virtual Design Spaces, *CAD*, 33, pp.637-651, 2001.

[9] David P. Luebke. A Developer's Survey of Polygonal Simplification Algorithms, *IEEE Computer Graphics and Applications*, 21(3), pp.24-35, 2001.

[10] Hugues Hoppe, Progressive Meshes. In *Proceedings of the 23th annual conference on Computer graphics and interactive techniques*, ACM Press, pp.99-108, 1996.

[11] Ravi S. Sandhu and Pierrangela Samarati, Access Control : Principles and Practice, *IEEE Communications*, 32(9), pp.40-48, 1994.

[12] Butler Lampson, Protection, In *Proceedings of the 5th Annual Princeton Conference on Information Sciences and Systems*, Princeton University, pp.437-443, 1971.

[13] Sylvia L. Osborn, Ravi S. Sandhu and Qamar Munawer, Configuring Role-Based Access Control to Enforce Mandatory and Discretionary Access Control Policies, *Information and System Security*, 3(2), pp.85-106, 2000.

[14] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein and Charles E. Youman, Role-Based Access Control Models, *IEEE Computer*, 29(2), pp.38-47, 1996.

[15] J. El-Sana and A. Varshney, Controlled Simplification of Genus for Polygonal Models, In *IEEE Visualization*, pp. 403-412, 1997.

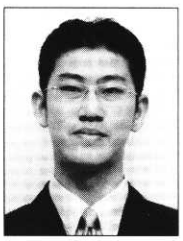
[16] Hugues Hoppe, View-Dependent Refinement of Progressive Meshes, In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, ACM Press, pp.189-198, 1997.

[17] M. Garland and P. S. Heckbert, Surface Simplification Using Quadric Error Metrics, *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, ACM Press, Addison-Wesley Publishing Co., pp.209-216, 1997.

[18] Ryutarou Ohbuchi, Hiroshi Masuda, and Masaki Aono, A Shape-Preserving Data Embedding Algorithm for NURBS Curves and Surfaces, In *Computer Graphics International*, pp.180-187, 1999.

[19] Ryutarou Ohbuchi, Shigeo Takahashi, Takahiko iyazawa, and Akio Mukaiyama, Watermarking 3D Polygonal Meshes in the Mesh Spectral Domain, In *B. Watson and J. W. Buchanan, editors, Proceedings of Graphics Interface 2001*, pp.9-18, 2001.

[20] Emil Praun, Hugues Hoppe and Adam Finkelstein, Robust mesh watermarking, In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., pp.49-56, 1999.



김 태 성

e-mail : falcons@ece.skku.ac.kr

2001년 성균관대학교 전기전자 및 컴퓨터공학부(학사)

2002년 미국 Drexel University, Department of Computer Science 교환 연구원

2001년~현재 성균관대학교 대학원정보통신공학부 컴퓨터공학 전공 공학석사

관심분야 : Collaborative Design, Geometry Modeling, Realtime Rendering, Mobile 3D graphics 등

한 정 현

e-mail : han@ece.skku.ac.kr

1988년 서울대학교 컴퓨터공학과(학사)

1991년 미국 University of Cincinnati, Department of Computer Science (공학석사)

1996년 미국 University of Southern California, Department of Computer Science(공학박사)

1996년~1997년 미국 상무성 National Institute of Standards and Technology (NIST) Manufacturing Systems Integration Division 연구원

2002년 미국 Drexel University, Department of Computer Science 연구조교수

1997년~현재 성균관대학교 정보통신공학부 컴퓨터공학 전공 부교수

2001년~현재 산업자원부 지정 (성균관대학교 소재) 게임기술 개발지원센터 센터장

관심분야 : Geometric Modeling, Realtime Rendering, 3D Game Design and Development