Towards Consistency in a Heterogeneous Collaborative Geometric Modeling Environment

Luiz Gonzaga da Silveira Jr
Electrical and Computer Engineering Faculty
State University of Campinas
P.O.Box 6101 13083-970
Campinas, SP - Brazil
gonzaga@dca.fee.unicamp.br

Wu, Shin-Ting
Electrical and Computer Engineering Faculty
State University of Campinas
P.O.Box 6101 13083-970
Campinas, SP - Brazil
ting@dca.fee.unicamp.br

Abstract

This paper presents a hybrid system architecture for collaborative geometric modeling applications, as a tradeoff solution between replicated and centralized approaches. It addresses the geometric inconsistency that may be presented by a replicated-based architecture and the tight visualization coupling in the centralized one. The proposed approach provides a infrastructure to ensure the same geometric processing for a highly heterogeneous computing environment. Moreover, it allows a decoupled visualization and interaction on each user's workspace. A prototype has been implemented and evaluated concerning with geometric consistency and responsiveness.

Keywords

geometric modeling, cross-platform consistency, CSCW, distributed heterogeneous system, interactive 3D graphics interface, design pattern.

1. Introduction

The typical scenario for collaborative modeling is a shared workspace, where a dispersed group of users (end-users) work together for creating and modifying an application-dependent 3D-model over the Internet. Concerning with the underlying architecture one may distinguish three approaches: centralized, replicated and hybrid one [Greenberg95].

In the centralized architecture only one instance of the shared application runs in a central server. On the end-users workspaces the local processes are restricted to displaying the output from the central server and managing the input events. It makes the concurrency control simple to be implemented, but the interactivity might be compromised. Any end-user interaction generates a sequence of events, which must be collected and routed to the central server where they are handled. And the same output of the shared application must be broadcasted to all participating machines for visualization (Figure 1). This may generate substantial overload both in the central server and network due to the continuous traffic and processing. Examples of applications based on the centralized architecture are NetMeeting [Microsoft99], SharedX[Garfinkel94], and XTV [Chung94].

In the replicated architecture, one instance of the shared application runs locally on each end-user's workspace. Both the input events and the graphics output are handled

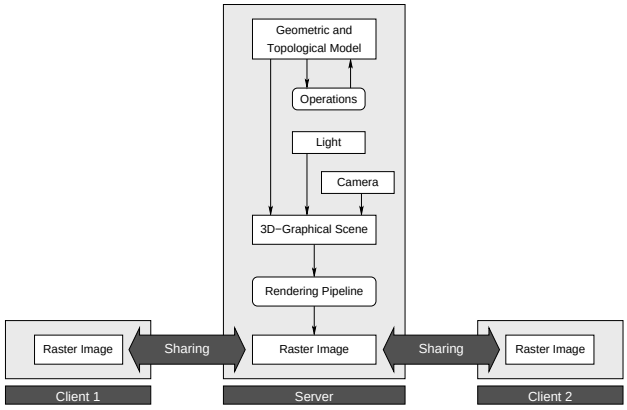

Figure 1. Centralized architecture

locally. This approach favors the design of a user interface more specialized to the local computing environment, since the application and the interface are part of the same front-end process. Additionally, UI researchers point out that a tighter integration between an application and its interface lets the application provide more naturally semantic feedback while the user is interacting [Zeleznik93]. Moreover, the system response-time may be enhanced, once the network traffic from the events is relatively lighter. GroupKit [Roseman96] and DistEdit [Knister90] are examples of applications that have replicated architecture.

By presenting favorable network communication features, the replicated architecture has been adopted by most collaborative 3D geometric modeling systems, such as Repo-3D [MacIntyre98], Distributed Open Inventor (DIV) [Hesina99], IntelligentBox[Okada98], TOBACO [von Lukas00], and CoCreate OneSpace [Mackrell99]. The benefits of a replicated architecture must, however, be balanced against the homogeneous numerical computation offered by the centralized one, when we migrate from a homogeneous to a heterogeneous computing environment. Under heterogeneous computing environment we understand a network of computers with different CPU speed, computing power, computing precision, memory, graphics processing, and display capabilities. And our focus is on the consistency across such machines, without completely disregarding its performance.

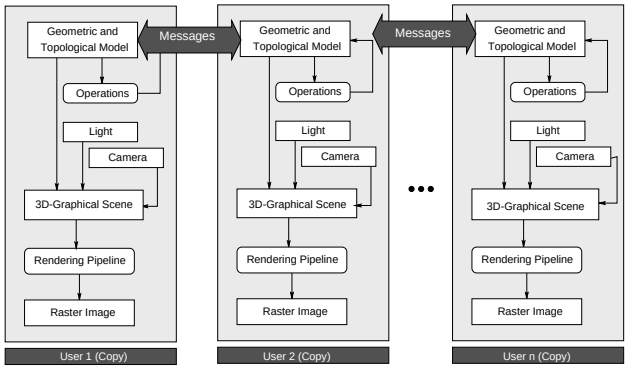

Figure 2. Replicated architecture

We propose a hybrid architecture for collaborative applications, as a tradeoff solution for keeping the consistency of 3D-model and for preserving the system usability. The hybrid architecture results from the combination of the both centralized and replicated architectures. The basic idea consists in separating application-dependent model from graphics functionalities. More precisely, it consists in separating geometric modeling and interaction (rendering and manipulation) activities. All geometric/topological operations are performed in a *geometric/topological model* located in a central *modeling server*, as response to networked requests from the participating machines (clients). Graphics interactions and visualization, on their turn, are carried out on the replicated graphical model residing in each client's machine. In this approach, we may take advantages of the well-known robust geometric algorithms [Michelucci99] designed for monolithic modeling applications to enhance the robustness of the entire heterogeneous platform.

For validating our proposal, we have implemented a multiplatform collaborative geometric modeler called `CoMo` (acronym for Collaborative Geometric Modeler). We analyzed its responsiveness on the basis of the metrics presented in [Nielsen94] and its robustness from the standpoint of heterogeneity of numerical computing environment [Hoffman89]. The preliminary tests indicate that the data consistency is ensured and, when suitably optimized, each local workspace has a response-time above the limit of the user's acceptance.

In the next section, we give a brief explanation of the design issues that we addressed. In section 3 our proposed architecture is given. A prototype, `CoMo`, that we used for evaluating our proposal is succinctly described in section 4. Four experiments are presented in section 5. In the first experiment, we tested the cross-platform consistency of the entire computing platform by installing the model server in different machines. In the remaining experiments, we tested the responsiveness of the system in two distinct situations: tight and loose application–graphics coupling.

2. Design Issues

To our best knowledge, a few collaborative geometric modeling systems are designed specifically with cross-platform consistency and heterogeneity in focus. There are various reasons. First, due to its very specialized nature, most of collaborative geometric modeling systems were designed for a homogeneous computing environment (one machine) with increased processor load and cost of application licenses. Second, because of highest bandwidth required between the application and the interface, an appropriate 3D interaction architecture for a single user system is still questionable. Third, a broad range of issues must be taken into consideration in the design of a CSCW (acronym for Computer Suported Cooperative Work) system. It depends on the tasks involved, the users, and the environment. The focus of most researchers has been on the distributed infrastructure and group facilitors.

As already stated, in our work we are centering on the two design issues: cross-platform consistency and responsiveness in a heterogeneous computing environment.

2.1. Cross-Platform Consistency

Two main sources of inconsistencies may affect collaborative geometric modeling applications: concurrent manipulations and inaccuracy of floating point arithmetic. While concurrency problems have been tackled in the development of general-purpose collaborative applications [Koch94, Strom98], geometric inconsistency problems have only been addressed for monolithic geometric modeling applications [Michelucci99].

The problems caused by floating-point arithmetic in geometric computation is serious. Due to its approximation nature, there may be imprecision that lead to contradictory information about modeled objects. For example, note in Figure 3 that $a =_\epsilon b$, $b =_\epsilon c$, but $a \neq_\epsilon c$. In this case, the classical transitivity property for Euclidean geometry is violated. Such a violation leads to inconsistent decisions. In order to improve the robustness, several approaches were proposed for consistent interpretation of imprecise numerical results in a single-user system [Bruderlin90a] [Hoffman89] [Segal90]. It expects that for all legitimate inputs, consistent results are delivered.

In replicated-based systems, inconsistency problems may be worsen, once the replicated data may be processed by

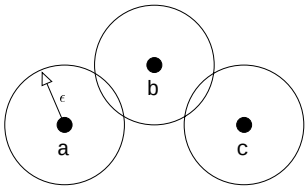

Figure 3. Violation of transitivity property.

machines with different computing power and representation precision. Slight differences in the processing may deliver contradictory numerical data and yield distinct logical decisions. Yet, it is difficult to conceive a way to guarantee that the output from one machine will be the same as the output from another, even when each instance of an application is individually robust and when the same floating point computation standard is adopted.

Figure 4 exemplifies the results of robust intersections between two polygons (F_1 and F_2) and a straight line (E_1). In the first replicated application (Figure 4.a) the intersection is a point on the boundary of the two polygons, and, in the second one (Figure 4.b), the result are two points, V_r and V_s, in the interior of F_1 and F_2.

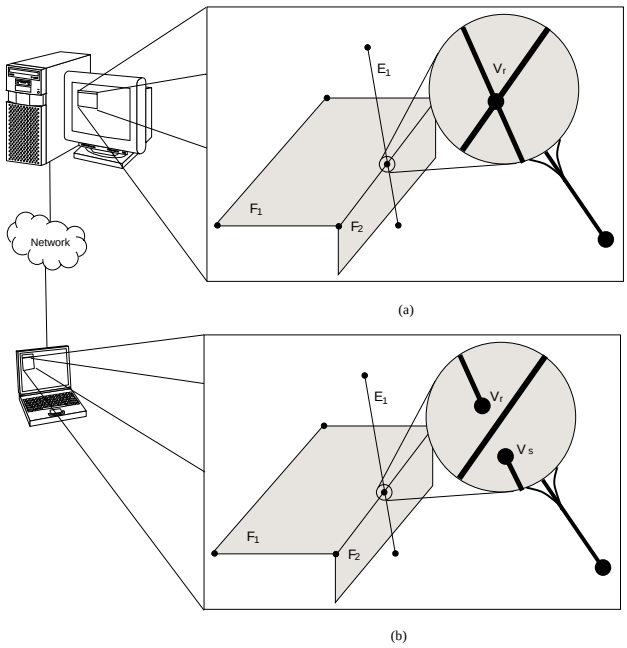

Figure 4. Inconsistencies in the replicated architecture

2.2. Responsiveness

To have wide acceptance, any interactive system must be designed with usability in mind. Usability concerns making systems easy to learn and easy to use. There are several usability metrics to assess system's performance, such as time to complete a task, time spent on errors, and ratio of successes to failures, and so on [Preece94].

For 3D graphics interactive systems, some of usability metrics are functions of system's *latency*. Latency is related with the update speed of an image in response to a user action. It plays an important role in the fluidity of end-users interactions with their applications. The latency must be the lowest as possible. As computer cannot provide fairly immediate response, three important latency limits have been identified regarding the reaction and behavior of end-users [Nielsen94]:

- **0.1 sec.** the limit for having the user feels that the system is reacting instantaneously. Hence, no special feedback is necessary.
- **1.0 sec.** the limit for the user's flow of thought to keep uninterrupted, despite the noticeable delay. Normally, no special feedback is necessary during delays in the range between 0.1 and 1.0 second, although the user does lose the feeling of operating directly on the data.
- **10 sec.** the limit for keeping the user's attention focused on the dialogue. From the usability point-of-view, a visual feedback is required when a system will take significant amounts of time to perform a task.

On the basis of these limits we will evaluate the acceptability of our proposed hybrid architecture. We will show that in our architecture we adopt a very conventional way for updating the screen images in response to the user's actions. Callbacks are used to alter a graphical model in each participating machine.

3. A Hybrid Architecture

Figure 5 presents an overview of our proposal for a collaborative geometric modeling architecture. It comprises three distinct classes of application on the top of a high-level distribution platform: geometric modeling server, user workspace application, and group manager server.

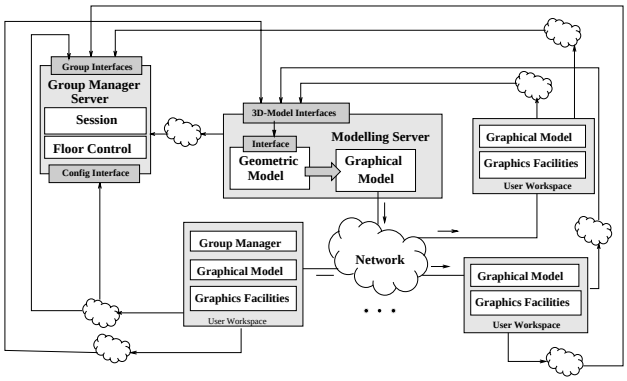

Figure 5. The conceptual model of our proposal

In this approach, the *geometric modeling kernel* resides in a central server and end-users interact with a simplified model, renderable on any machine, through either a

3D-cursor or 3D-graphics metaphors. This separation allows an end-user to set her/his rendering and manipulation parameters in accordance with local hardware capabilities, without compromising the geometric processing (Figure 6).

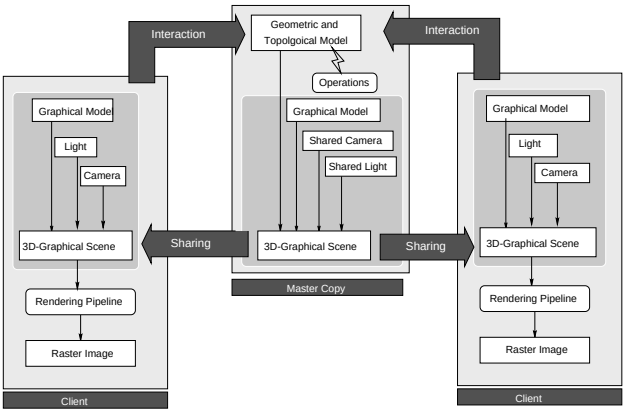

Figure 6. Hybrid architecture

Since the geometric decisions are under central control, it suffices to integrate in the *geometric modeling kernel* the well-known strategies to cope with robustness problems such as perturbation schemes [Edelsbrunner88], symbolic computation [Yap90], and interval arithmetic [Comba93]. A *graphical representation* of the underlying 3D geometric model is replicated and sent to the participating machines on demand. In this way, the geometric data consistency is ensured across the network, independent of local numerical computation power, and the graphics hardware features may be better explored at each workspace.

In our environment, several users may interact with the shared 3D model at once. Some control strategies are, therefore, necessary to avoid concurrency problems. In order to allow end-users to take turns in interacting with a shared 3D model, it is devised in our architecture *floor control mechanisms*, residing in a *group manager server*. The floor control mechanism is responsible for avoiding/solving resource contention of potential conflicts that may arise from simultaneous access of a shared application by various end-users. The most common strategy works on first-come-first-served basis.

3.1. Geometric Modeling Server

Figure 7 presents the conceptual model of the architecture of our geometric modeling server.

The geometric modeler is application-dependent and tailorable to the rendering and manipulation functionalities provided by the user workspace. One of the key elements that bridge the geometric modeler and the user workspaces is the *graphical model*. The graphical model is a hierarchical structure consisting of graphics elements necessary for rendering an image, such as geometric data, texture, and color. These elements are supported by the most low-level 3D graphics library.

The geometric server not only holds `Application-Dependent Model`, but also the operations on them. A representative of the geometric server is created in each participating machine whenever interactions on the geometrical model are required, in accordance with the `Proxy` design pattern [Gamma95]. The concurrency control is accomplished by intercepting the user's requests between the model manager and the user workspace application. These requests are granted in a sequence imposed by the floor control mechanism.

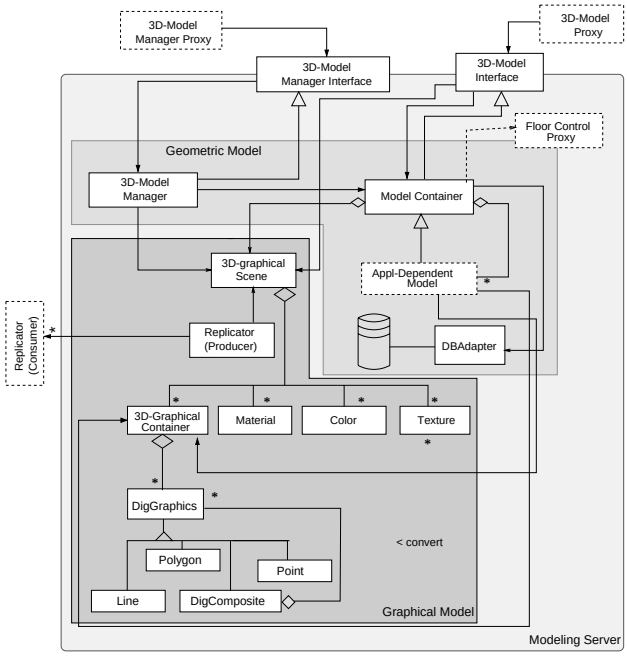

Figure 7. A geometric modeling server

Whenever a change is occurred in the `Application-Dependent Model`, its corresponding graphics data, `DigGraphics`, are also updated. While the geometric data and methods reside in a server, the graphics data are replicated for each end-user workspace where they are rendered. In this way, the speed with which each client re-renders the graphics objects after any view change depends only on the local graphics hardware power. Nothing needs to be transmitted over the network. If the geometric data are updated in a server, the changed part must be converted into graphics data format and multicasted from the server to the clients. In this case, instead of the whole graphics scene or complex native 3D geometric data, only the modified graphics data are transmitted.

To be flexible enough for accommodating distinguishing features of a variety of geometric representation schemes, the geometric data and geometric methods are customizable by the application developers [Pree95]. Heuristics that can ameliorate the robustness problems may be included in the geometric operations.

3.2. User Workspace

Our emphasis on the design of user workspace applications is the tailorability to each local computing power and the

accessibility to all geometric functions provided by a geometric modeling server. Its conceptual model is sketched in Figure 8.

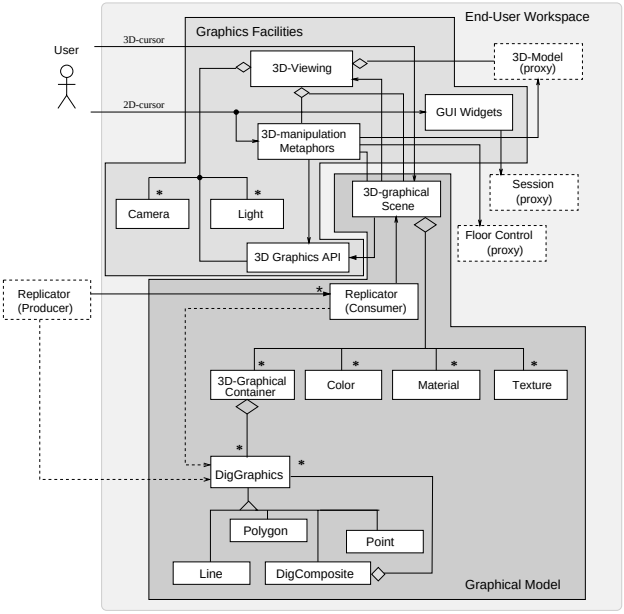

Figure 8. A user workspace

As already mentioned, to provide a more versatile way to refer to an object residing in the geometric server, a representative of the 3D geometric server is instantiated on demand at each user's workspace. For rendering and interaction purposes, we reused the functionalities available in the manipulation toolkit `MTK` [Wu99].

Besides the Gouraud rendering pipeline, MTK provides facilities to implement *metaphors*, which convey operations on a 3D object through a 2D cursor in a more familiar and accessible pictorial representation, called *dragger*. MTK also permits the users to interact with 3D objects via a 3D cursor. A main differential concept introduced in MTK is the model-graphics decoupling.

The model-dragger decoupling may enhance the performance of the system concerning with usability. An immediate semantic feedback may be provided while the user is interacting with the system, even though a geometric server needs longer time to accomplish a task, as illustrated in the sequence diagram of Figure 9. Observe in this diagram that, when an event is captured by a manipulator, it generates a semantic value, such as displacement and rotation angle. This value is, then, used for updating a 3D geometric object in the geometric server and its corresponding dragger. As draggers reside in the local machine, their latency is less than the latency of any 3D model.

The latency of a dragger is given by

$$t_{dragger\_latency} = t_{lp} + t_r, \qquad (1)$$

where t_r and t_{lp} are, respectively the local processing and rendering time. Whereas the visual feedback of a geometric object requires remote processing

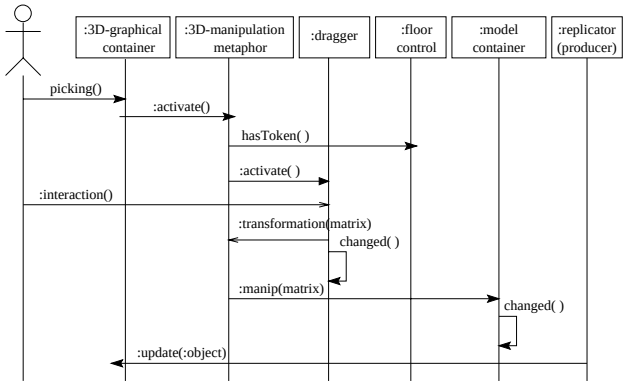

Figure 9. Interaction sequence

$$t_{model\_latency} = t_{lp} + t_{t1} + t_m + t_{t2} + t_r, \qquad (2)$$

where t_{t1}, t_m, and t_{t2} are, respectively, the transmission time of a user's request, the processing time in the central server, and the transmission time of the server's response.

The equation (2) suggests us that when an application is separated from the user interface, we have several ways to improve its latency: (1) we may locally adjust the rendering parameters (t_r) in order to counterbalance the delays in a network (t_{t1} and t_{t2}); (2) we may invest in the processing power of the central server (t_m); or (3) we may invest in a higher bandwidth network. Whatever is the solution, it is possible in our architecture to support different levels of rendering, tailoring to the local computing power, without sacrificing the geometric model processing capabilities of the central server.

The implementation of a 3D cursor in the context of a model-graphics decoupled architecture is not trivial at the first glance. Usually, the set of primitives provided by a geometric modeler is much more specialized than the one supported by our graphical model. This difference may yield discrepancies between what you see and what you get, if the updates of the application and graphics data are not appropriately coordinated. In the design of MTK, the paradigm "what you see is a simplified form of what you get" is adopted. The positions of the 3D cursor on a surface of a 3D object is computed by the application and not by MTK. The 3D cursor's movements are constrained on the tangent plane of a surface or on the tangent direction of a curve at each current point. Thus, its latency is governed by

$$t_{cursor\_latency} = t_{t1} + t_m + t_{t2} + t_r. \qquad (3)$$

Different from a single-user interface, a interface for a collaborative environment must also consider assisting human-human interaction and the cultural and social background of the participants. Providing some convenient ways of identifying "contextually" the participants and what they are doing is an important issue [Ellis94]. Audio, video, or even images have mostly been used for enhancing this group awareness and improving the attention and

mood of the participants. Despite its importance, group awareness is not the main concern of this paper.

3.3. Group Manager Server

The *group manager server* comprises the session and floor control services (Figure 10). It holds information about work session, membership policies, and the role of each member in the group in order to coordinate the teamwork. We distinguished three classes of end-users: members – have only viewing access right and may manipulate their own viewing and lighting parameters, collaborators – are specialized members that have the viewing and manipulating access right, and chairwoman/chairman – coordinates the session and its internal activities.

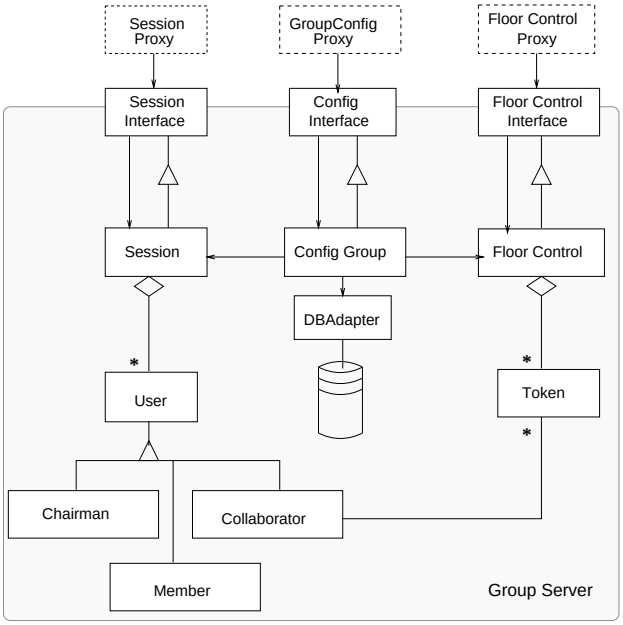

Figure 10. A group server

A work session is built by a group of end-users (members) working together on a specific problem, according with integrity criteria (minimum and maximum number of members in a session) and a membership policy (establishes how an end-user joins to a session) established by the system administrator via `ConfigInterface`. The administrator application may run remotely and access all of server services through their proxies.

3.4. Distributed Platform

In our proposal the heterogeneous computing systems are networked and the information are routed among them. For example, the updated graphical model is sent to each participating machine, and the user's requests are sent either to *geometric modeling server* or to *group manager server*. Hence, it is necessary to provide an infrastructure for distributing the information to every user's workspace. CORBA is an object-oriented infrastructure that allows objects communications, independent of platforms and techniques used to implement these objects. This is achieved by transparently invoking remote methods via the IDL (Interface Definition Language) [Group95]. In a CORBA specification several services, such as Naming, Transactions, Concurrency Control, and Event, that facilitate the development of distributed object applications are also included.

4. CoMo: A Prototype

For validating some usability aspects of our proposed architecture, we implemented the CoMo – Collaborative Geometric Modeler. Aware of the complexity of such an architecture, several public-domain softwares have been used and we simplified the geometric functionalities and user interface at great deal.

In the *geometric modeling server*, an instantiation mechanism is used to create a new object and a boundary representation [Mortenson85, Mäntylä88] is chosen to describe internally the object data.

Only three classes of polyhedral objects may be instantiated: wedge, cube, and faceted cylinder. Geometric transformations (translation, rotation, and scaling) are supported via a transparent sphere dragger by interacting with a 2D cursor. Moreover, the boolean operations, namely intersection, union, and difference, are also supported.

The *group manager server* holds a repository containing information about users connected to session, policies for group joins, and a floor control mechanism. For simplicity and without loss of the objective of our work, we consider that the floor control strategy is imposed by the system. It works on the basis of token-passing. Each manipulable geometric object has its own token. When a token is associated to a member, she/he becomes the owner of the corresponding object and nobody can access that object until its release.

The *user workspace* interface is very simple. It consists of one scene hierarchy tree window and one 3D scene window, runnable under the control of a X window system. We used the built-in set of widgets provided by GTK+ [Mattis93] to implement these two windows, because GTK+ provides a uniform handling of local and networked events. A participant directly interacts with objects in the 3D scene window by clicking and dragging the sphere draggers or the 3D cursor. MTK, on its turn, accesses the functionalities of OpenGL [Neider93] for efficient rendering.

For implementing a *distributed object environment* we used a freeware MICO [Puder00] – an implementation of a CORBA specification. Naming Service is used to name and categorize the geometric, graphics, and group information. Event Service decouples the event producers and consumers and provides facilities for reliable one-to-many communication through one (or more) event-channel(s) [Group93]. It was useful in implementing graphical model replication. We adopted the push-model configuration with three event-channels. One channel is for the communication between the `DigGraphics` (master) residing in the geometric modeling server and their

replications (slaves) at user's workspaces. Whenever a change occurs in the geometric data, the master `DigGraphics` posts the event-data (updated graphics data) into the channel and the slaves consumes the event-data from this channel (Figure 11). Only changed objects are replicated.

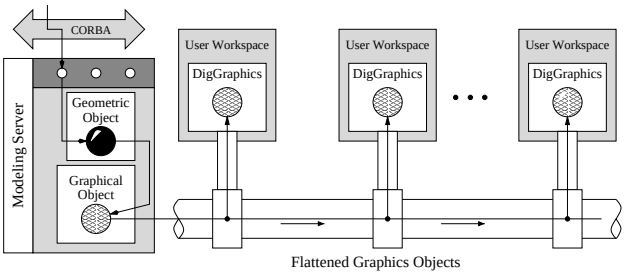

Figure 11. Graphics objects replication

5. Performance Evaluation

For evaluating the performance of our proposed architecture, we installed CoMo in the following sub-network of our laboratory with machines of different numerical precision: 1 UltraSPARC/10 with Creator/3D, 1 UltraSPARC/10 without graphics accelerator (Mesa3D 2.4), 1 UltraSPARC/1 with Creator/3D, 1 UltraEnterprise 450 without graphics accelerator, and 1 PC Pentium II 300MHz with Linux 2.4.3 and Mesa3D 2.4 over two network buses in normal use condition. The UltraSPARC/1 and PC are connected on a 10Mb/sec bus, while the UltraSPARC/10 are connected on a 100Mb/sec bus.

Two groups of experiment were carried out. The objective of the first is to evaluate the cross-platform consistency of CoMo. The second group of experiments attempted to measure the latency of our system and we compared the preliminary results with the user tolerance limits given in Section 2.2.

5.1. Cross-Platform Consistency

We performed two distinct tests: we have installed the geometric modeling server in two machines of distinct architectures and carried out separately a (boolean) difference operation between two cubes that are $d = 10^{-24}$ apart. The strategy that we used for enhance the robustness of the boolean operations is based on the two-ε-tolerance technique [Bruderlin90b].

Because of the precision of machines, the Ultra/SUN machines consistently delivered as a result two non-intersecting cubes. While in the PC Pentium II, a vertex was generated as an intersection of the two cubes. It was expected, once the geometric operations are dependent on the architecture of a machine where the geometric modeling server resides. However, the central control of the geometric computation guarantees that the same consistent result is propagated to all participating machines. This leads that the same 3D geometric model is consistently perceived by the end-users at any kind of machine in our hybrid architecture.

5.2. Responsiveness

One important variable in the evaluation of the responsiveness of an interactive system is the latency. For analyzing performance concerning with the latency, we chose the UltraEnterprise to be the host of the geometric modeling server and carried out several tests by varying the complexity in geometric functionalities and the rendering mode.

5.2.1. Latency of a Dragger

The equation (1) expresses the latency of a dragger. Since a dragger is processed in the local machine and its geometry is designed as simple as possible, its latency depends only on the graphics hardware support in each user workspace.

Figure 13 summarizes the average latency in the participating machines in relation to the number of facets in the scene. For machines with graphics accelerator (UltraSPARC/1 and UltraSPARC/10, on the bottom of Figure 12), Gouraud shading was applied, and for machines without graphics accelerator (PC and UltraSPARC/10, at the top right corner of Figure 12), wireframe visualization was used. Note that the latency lies in the range 0.1–1.0s for machines with graphics accelerator and 1.0–2.0s for machines without graphics accelerator. Despite 1.0–2.0s is not an ideal, it is an acceptable range. Surprisingly, we also noted that the latency tends to be constant when the number of facets is greater than 50.

We may conclude, as expected, that the responsiveness may be improved by sacrificing the image quality. The decreasing in this quality does not, however, affect the geometric power of the environment as a whole.

5.2.2. Latency of a Geometric Object

The equation (2) tells us that the latency of a geometric model involves more variables. We performed an experiment consisting basically in rotating randomly an object via the sphere dragger. We measured the latency of geometric models in each participating machine and the representative average experimental data are given in Figure 14. Comparing with the graphics in Figure 13, we note that the overhead comprising the network transmission and the geometric processing is minimal, once the 3D-models are of reduced size (5 facets).

5.2.3. Latency of the 3D Cursor

From equation (3), the latency of the 3D cursor is the most critical one, since the user interacts continually with it and its position in 3D may depend strongly on the geometric modeling server. It may overload network by constantly transmitting cursors positions.

We measured the average latency of the 3D cursor in each user workspace, by moving it randomly in the scene window depicted at the top left corner of Figure 12. The semantics of the movement is defined by pressing one of the 2D mouse button: when the middle button is pressed, the movement of the cursor is constrained on yz-plane; when the left button is pressed, the movement of the cursor is constrained on the geometry of a 3D model; and when

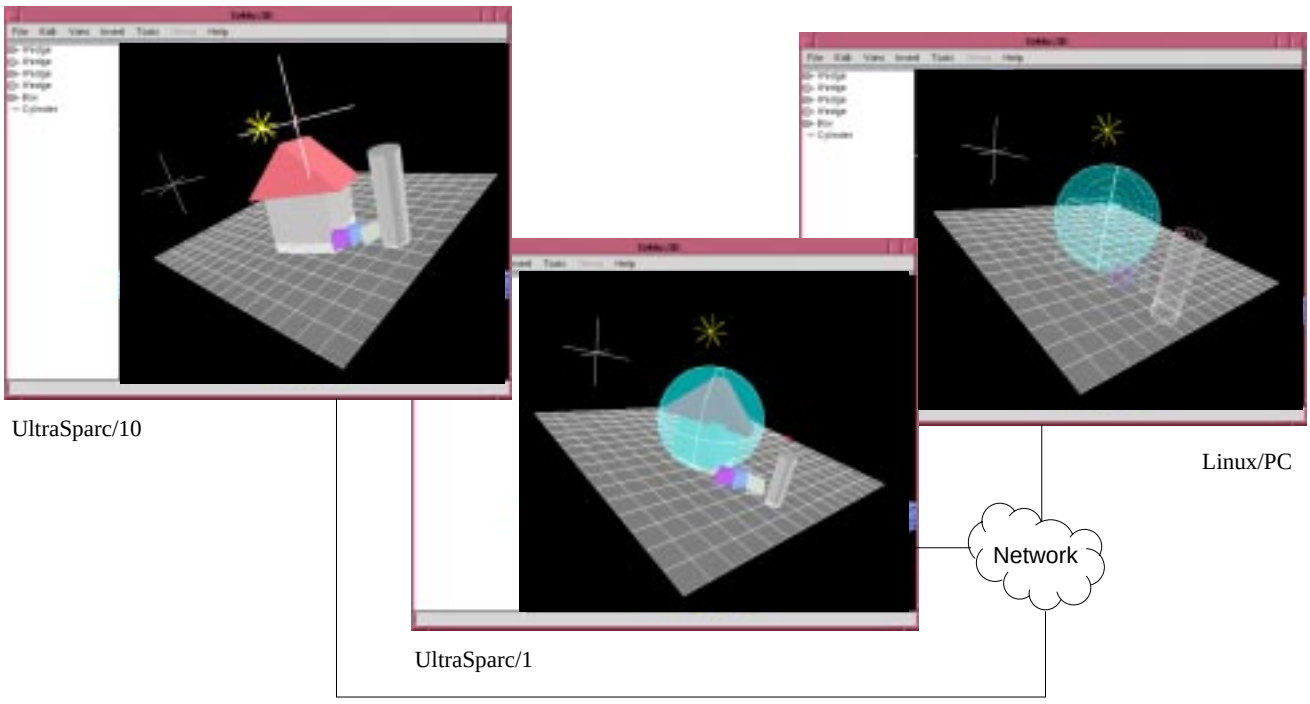

Figure 12. A snapshot of CoMo.

no button is pressed, the movement is constrained on xy-plane. From our experiments, on the machines with graphics accelerator no visual discontinuity on the cursor movements was perceived in a scene with up 200 facets.

6. Concluding Remarks

Aiming at cross-platform consistency and responsiveness, we proposed a hybrid architecture for collaborative geometric modeling environments. The main feature of our propose is decoupling the 3D model from its visualization, which permits us to locally configure the graphics resolution and still maintains the geometric precision across the network.

With the rapid advance in the network technology, we believe that the latency due to network transmission tends to be negligible. This supposition is somehow confirmed by our preliminary experiments. We noted that the decisive factor is still the graphics rendering capabilities in each participating machine. As further work we plan to include objects with more complex geometry and to evaluate the performance of the environment over Internet.

References

[Bruderlin90a] Bruderlin, B. *Detecting Ambiguities: An Optimistic Approach to Robustness Problems in Computation Geometry*. Technical Report UUCS-90-003, Computer Science Department, University of Utah, Salt Lake City. (1990a).

[Bruderlin90b] Bruderlin, B. *Robust Regularized Set Operations on Polyhedra*. Technical Report UUCS-90-004, Computer Science Department, University of Utah, Salt Lake City. (1990b).

[Chung94] Chung, G., Jeffay, K., & Abdel-Wahab, H. Dynamic participation in computer-based conferencing system. *Journal of Computer Communications*, 17(1), 7–16. (1994).

[Comba93] Comba, J. & Stolfi, J. Affine arithmetic and its applications to computer graphics. In *Brazilian Symposium on Computer Graphics and Image Processing (VI SIBGRAPI)* (pp. 9–18). (1993).

[Edelsbrunner88] Edelsbrunner, H. & Mucke, E. P. Simulation of simplicity: A technique to cope with degenerate cases in geometric algorithms. In *Symposium on Computational Geometry* (pp. 118–133). (1988).

[Ellis94] Ellis, C. & Wainer, J. A conceptual model of groupware. In A. Press (Ed.), *CSCW'94* (pp. 79–88). (1994).

[Gamma95] Gamma, E., Helm, R., Johnson, R., & Vlissides, J. *Desin Patterns – Elements of Reusable Object-Oriented Software*. Addisson Wesley. (1995).

[Garfinkel94] Garfinkel, D., Wleti, B., & Yip, T. Hp sharedx: A tool for real-time collaboration. *HP Journal*, 45(2), 23–36. (1994).

[Greenberg95] Greenberg, S., Hayne, S., & Rada, R. *Designing Groupware for Real-Time Drawing*. McGraw Hill. (1995).

[Group93] Group, O. O. M. Corba event service specification - v1.0. `http://www.omg.org`. (1993).

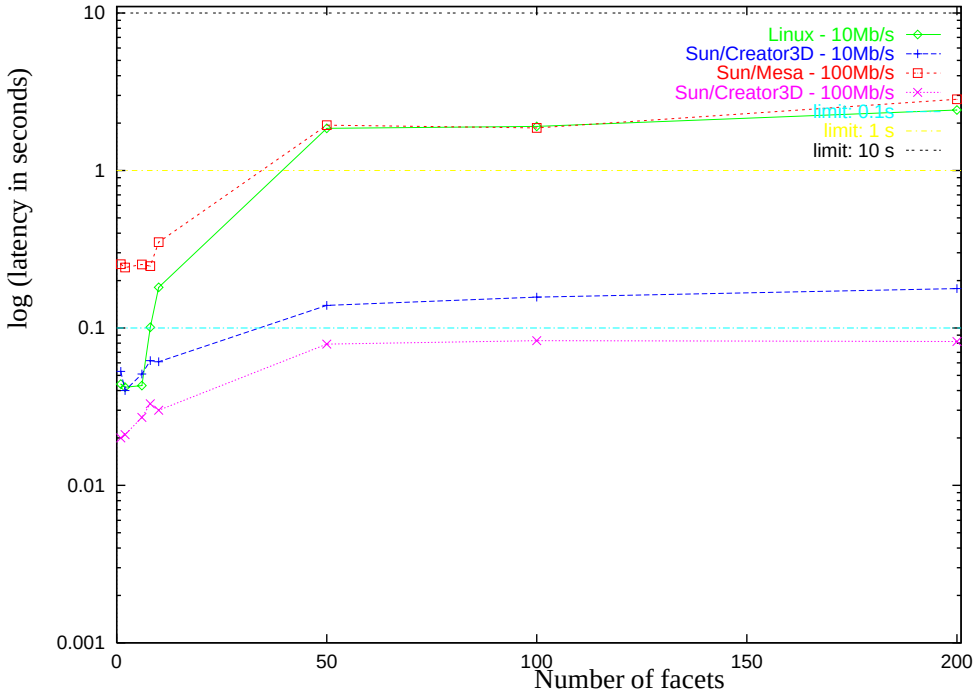

Figure 13. Dragger latency × number of facets

[Group95] Group, O. O. M. The common object request broker: Architecture and specification - version 2.0. (1995).

[Hesina99] Hesina, G., Schmalstieg, D., Fuhrmann, A., & Purgathofer, W. Distributed open inventor: A practical approach to distributed 3d graphics. In *Virtual Reality Software & Technology '99 (VRST'99)*: ACM. (1999).

[Hoffman89] Hoffman, C. The problems of accuracy and robustness in geometric computation. *Computer*, 22, 31–42. (1989).

[Knister90] Knister, M. & Prakash, A. Distedit: A distributed toolkit for supporting multiple group editors. In *Third Conf. Computer-Supported Cooperative Work* (pp. 343–355). (1990).

[Koch94] Koch, H. & Theel, O. An efficent data replication management architecture exploiting the separation of policy and mechanism. TR THD-BS-1994-1, Dept. of Computer Science, THD, Germany. (1994).

[MacIntyre98] MacIntyre, B. & Feiner, S. A distributed 3d graphics library. In *SIGGRAPH '98* (pp. 361–370). (1998).

[Mackrell99] Mackrell Cocreate onespace - a collaborative design infrastructure. `http://www.cocreate.com`. (1999).

[Mäntylä88] Mäntylä, M. *An Introduction to Solid Modeling*. USA: Computer Science Press Inc. (1988).

[Mattis93] Mattis, P., Kimball, S., & MacDonald, J. Gtk+: The GIMP toolkit. `http://www.gtk.org`. (1993).

[Michelucci99] Michelucci, D. An introduction to the robustness issue. In *Swiss Conference of CAD/CAM* (pp. 214–221). Neuchâtel, Switzerland. (1999).

[Microsoft99] Microsoft, C. The Windows NetMeeting Zone. `http://www.netmeeting-zone.com`. (1999).

[Mortenson85] Mortenson, M. *Geometric Modeling*. USA: John Wiley & Sons. (1985).

[Neider93] Neider, J., Davis, T., & Woo, M. *OpenGL - Programming Guide - Release 1*. Addison Wesley Co. (1993).

[Nielsen94] Nielsen, J. *Usability Engineering*. Morgan Kaufmann Publishers. (1994).

[Okada98] Okada, Y. & Tanaka, Y. Collaborative environments of intelligentbox for distributed 3d graphics applications. *The Visual Computer*, 14, 140–152. (1998).

[Pree95] Pree, W. *Design Patterns for Object-Oriented Software Development*. Addison Wesley/ACM Press. (1995).

[Preece94] Preece, J., Rogers, Y., Sharp, H., & Benyon, D. *Human-Computer Interaction*. Addison-Wesley Pub Co. (1994).

[Puder00] Puder, A. & Römer, K. *MICO: An Open Source CORBA Implementation*. Morgan Kaufmann Pub. (2000).

[Roseman96] Roseman, M. & Greenberg, S. Building Real Time Groupware with GroupKit - A Groupware

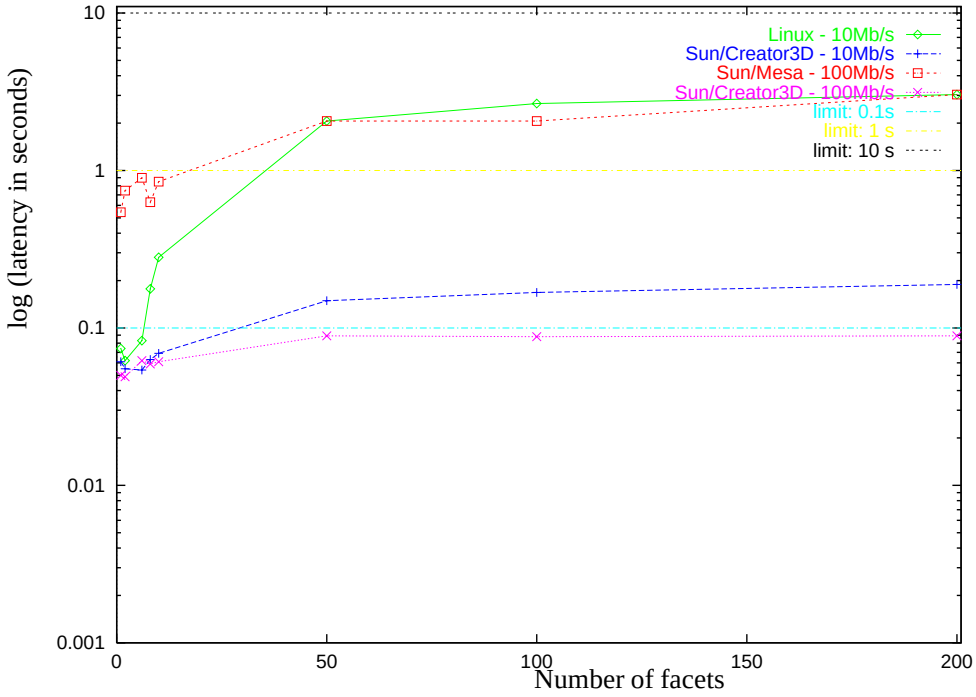

Figure 14. Model latency × number of facets

Toolkit. *ACM Trans. Computer-Human Interaction*, 3(1), 66–106. (1996).

[Segal90] Segal, M. Using tolerances to guarantee valid polyhedral modeling results. *ACM Computer Graphics*, 24(4), 105–114. (1990).

[Strom98] Strom, R., Banavar, G., Miller, K., Prakash, A., & Ward, M. Concurrency control and view notification algorithms for collaborative replicated objects. *IEEE Transactions on Computers*, 47(4), 458–471. (1998).

[von Lukas00] von Lukas, U. Tobaco - cooperative modeling with corba. `http://www.rostock.zgdv.de/ZGDV/Abteilungen/zr1/Projekte/TOBACO`, ZGDV - Rostok, Germany. (2000).

[Wu99] Wu, S. T. & Malheiros, M. G. A framework for interactive 3d geometric modelers. (submitted). (1999).

[Yap90] Yap, C. K. Symbolic treatment of geometric degeneracies. *Journal of Symbolic Computation*, 10, 349–370. (1990).

[Zeleznik93] Zeleznik, R. C., Herndon, K. P., Robbins, D. C., Huang, N., Meyer, T., Parker, N., & Hughes, J. F. An interactive 3d toolkit for constructing 3d widgets. In *Proceedings of SIGGRAPH'93*, volume 27 (4) (pp. 81–84). (1993).