Improving the Responsiveness in Multiplatform Collaborative Environments

Luiz Gonzaga da Silveira Jr
UNISINOS, V3D Studios, Brazil
lgonzaga@unisinos.br

Wu Shin-Ting
UNICAMP, Brazil
ting@dca.fee.unicamp.br

ABSTRACT

This paper addresses the design issues that may improve the responsiveness of a multi-platform collaborative modeling system, for which robustness and awareness are necessary requirements. The key points of our proposal are, whenever possible, (1) to reduce as much as possible the granularity of the transmission data over network; (2) to simplify as much as possible the functional feedbacks for fast screen update; and (3) to avoid as much as possible the network accesses for synchronization. On the basis of these hypotheses, we explore the features of a hybrid groupware architecture and show the feasibility of our proposal. Several latencies are measured to validate our assumptions.

Keywords: Collaborative Systems, Groupware, Geometric Modeling, Geometric Robustness, Awareness.

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 an internet network. Concerning with the underlying system architecture one may distinguish three approaches: centralized, replicated [2], and hybrid one [9].

In the centralized architecture only one instance of the shared application runs in a central server, while end-user workspaces display the same scene from the central server and managing the input events. It makes the 3D-model concurrency control simpler to be implemented, but the interactivity might be compromised. Beside, this approach may generate substantial overload both in the central server and network due to the continuous traffic and processing.

In the replicated architecture, one instance of the shared application runs locally on each end user's workspace. The system's response time may be enhanced, once the network traffic is relatively lighter. The benefits of a replicated architecture must, however, be balanced against the homogeneous numerical computation offered by the centralized one, when we migrate to a heterogeneous computing environment. Under heterogeneous platform, we understand internetwork of computers equipped with distinct hardwares (CPU, display technologies, memory and mainly GPUs), under different operating systems, compiler implementations and capabilities for float-point computing.

WSCG'2010, February 1 – February 4, 2010
Plzen, Czech Republic

To achieve consistency across the heterogeneous machines, without disregarding their individual performance, we proposed in [3, 9] a hybrid architecture for collaborative applications, as a tradeoff solution for keeping the consistency of 3D-model (geometric robustness) and keeping 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.

Besides the robustness, the separation of the geometric and the graphical model makes the rendering mode in each participating machine tailorable to the local computing power. Though, the participants may not only be working in different parts of the space with distinct viewpoints, but also calibrate the rendering parameters to the acceptable interactivity level.

In this paper we consider the aspect that helps ensure usability of any interactive system: the responsiveness. We will show that in the hybrid architecture we may control the granularity and the frequency of the transmitted information without sacrificing interactivity and robustness. Moreover, since the rendering mode may be tuned in the local workspace to fit the hardware capabilities. Personalized rendering may be set to compensate the latencies of different network transmission rates and the distinct processing performance of low ou high end GPUs used together. To validate our proposal, we have integrated our solutions in the multiplatform collaborative geometric modeler called CoMo (Collaborative Geometric Modeler) [9]. Several measures of latency have been performed and compared with the range of acceptable values proposed by Nielsen [7].

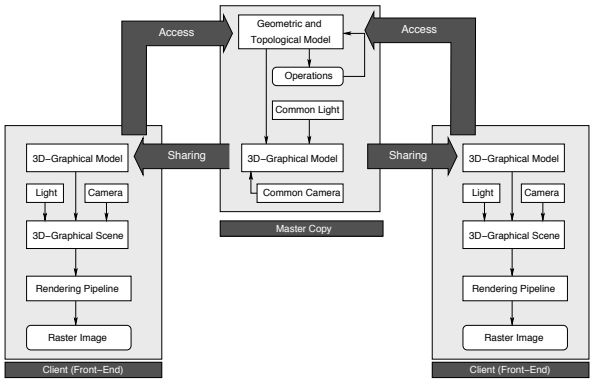

Figure 1: The conceptual model of a collaborative modeling system.

2 DESIGN ISSUES

In [9], we presented a hybrid architecture for collaborative 3D modeling systems, where two equivalent data are kept in the *geometric modeling kernel*: a geometric and a graphical data (Figure 1).

The geometric data is common to all end-users (interactive applications), whereas the 3D graphics data are replicable for visualization and manipulation in each end user workspace (*interactive application*). In this approach, we may take advantages of the well-known robust geometric algorithms [5], designed for monolithic modeling applications, to enhance the robustness of the entire heterogeneous platform.

To provide a more versatile way to refer to an object residing in the *geometric modeling kernel*, a proxy [1] is designed to make the user workspace communicate with a representative of the geometric model, rather than with the geometric model itself. We reused rendering and interaction functionalities provided by the Manipulation Toolkit *MTK* [4], running on top of OpenGL [6]. This is because that a main differential of *MTK* with respect to the other known graphical toolkits is its loosely decoupling of the application and the graphical models, although it provides efficient direct manipulation mechanisms via 3D-metaphors.

Several users may interact with the shared 3D model simultaneously. To avoid/solve resource contention of potential conflicts that may arise from simultaneous accesses of a shared application by various end users and to support group awareness, it is also devised in our architecture *floor control mechanisms* residing in a *group manager server*. The adopted infrastructure for objects communications over a network, independent of specific platform and techniques used to implement these objects, is based on the the Common Object Request Broker Architecture (CORBA) [8].

One drawback of the hybrid architecture, is that the group awareness may be drastically reduced and the system's usability may be deteriorated. As a solution, we proposed to integrate two awareness sub-windows in the user interface of each interactive application (on the left side of each application interface in Figure 2), in addition to the conventional drawing area (*scene view*) where users can interact with the 3D model through the 3D-metaphors (on the right side of each application interface in Figure 2). The awareness windows are responsible for conveying the global view of the ongoing activities: one is a listbox that contains the participant names (on the bottom left) and the other is the second drawing area (*global view* on the top left) where a simplified version of the global overview of the 3D shared workspace is presented.

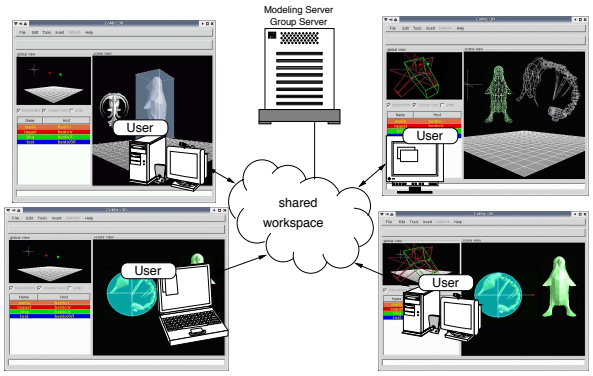

Figure 2: A Sample of The Modeller User Interface

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 user 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 [7]: 0.1s - the sistem is reacting instantaneously, no feedback is required; 1.0s - limit for the user's flow of thought to keep uninterrupted, despite the noticeable delay; 10s - the limit for keeping the user's attention focused on the dialogue, visual feedback is required.

3 THE SYSTEM RESPONSIVENESS

The emphasis on the design of an end-user workspace application for a hybrid architecture based system is its tailorability to each local computing power and the accessibility to all geometric functions provided by a geometric modeling server and the knowledge of the actions of the other users that share the same application model.

For making each instance of the end user workspace an interactive application, we have proposed useful awareness feedbacks, four problems must be solved:

1. System latency;
2. Event handling;

3. Remote pointing and viewing volumes;

4. Graphic states synchronization;

3.1 System latency

One of the challenging issues that we must circumvent is to devise a technique for interacting with a remote object through its corresponding graphical object in the user workspace, whose latency can be expressed as

$$t_{latency} = t_{t1} + t_m + t_{t2} + t_r, \quad (1)$$

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

Eq. 1 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, a hybrid architecture supports it.

Our network-independent solution for optimizing the interaction performance is a graphical object–dragger loosely coupling interaction paradigm that is supported by *MTK*. Under a dragger, we understand an object that has a pictorial representation and can map the 2D inputs from the pointing devices to motions in three dimensions. With the Mediator design pattern [1], we define the class of objects, called Manipulator, to control the interaction between each pair dragger–graphical object. In the graphical object–dragger interaction model, two interaction loops may be distinguished: user–dragger–geometrical model–graphical model–user and user–dragger–user (Figure 3).

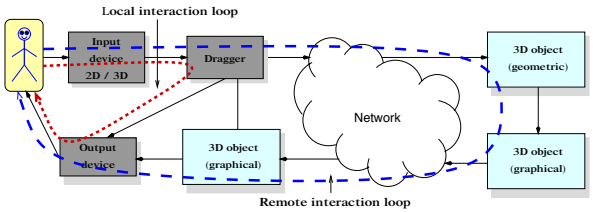

Figure 3: Interaction loops.

It is important to remark that the time response of the two interaction loops are different. Whereas the response time of the first loop is given by Eq. 1, the latency of the second loop may be expressed by

$$t_{dragger\_latency} = t_{lp} + t_r, \quad (2)$$

where t_r and t_{lp} are, respectively the local processing and rendering time.

Observing in Figure 3, by a sequence of events viewpoint, a manipulator generates a unit of information that is a semantically valid transformation, which is applied on the the dragger for local visual feedback. In parallel, the sequence of transformations is concatenated into a unique transformation for updating a 3D geometric object in the geometric server. This approach relieves the communication traffic.

3.2 Event Handling

The cost for loosely decoupling the geometric and graphical model in the context of interactions is the increase in the complexity of event handling. In addition to the user input events that can be handled by any interaction techniques toolkit, there are events from the communication channel that may also affect the context of the *scene view* sub-window, as illustrated in Figure 3.

An algorithm is necessary to extend the standard dispatching code for selecting the correct window for each of these events. A solution is to use an interaction that allows some portions of the event dispatching code to be modified by application programmers, provided by all GUI SDKs. On top of the window system, a Chain of Responsibility [1] defines an object that decouples the sender of events from the windows whose handlers an event should be forwarded.

3.3 Remote Pointing and Viewing Volume

Whenever a user interacts with the pointing device, the sequence of actions is collected and mapped into a meaningful unit of information. This unit of information is broadcasted to all the rest of participating machines for updating the state of the replicated manipulator. It guarantees the location awareness. Moreover, if the viewing parameters at each user workspace application are modified, they must also be multicasted to all the participating machines for redisplaying the context of the global view window and maintaining the perspective awareness. We propose an Observer design pattern [1] to define the object that performs this passive replication: the original manipulator/view parameters is the `subject` that all their replica (`observer`) must keep on observing. This approach keeps all client applications consistently updated.

It is worth observing that the latency of the manipulators is the most critical one, since the user interacts continually with them and the obtained units of information must be constantly transmitted over the network. This latency depends on the transmission time of the replicated data t_{t1} and the rendering time t_r in each participating machine (Figure 4):

$$t_{rep\_latency} = t_{t1} + t_r. \quad (3)$$

To make this latency as lower as possible, we suggest to use the wireframe rendering mode in the global view window and to adopt the simplest graphical representation to the replicas. The pictorial representations of

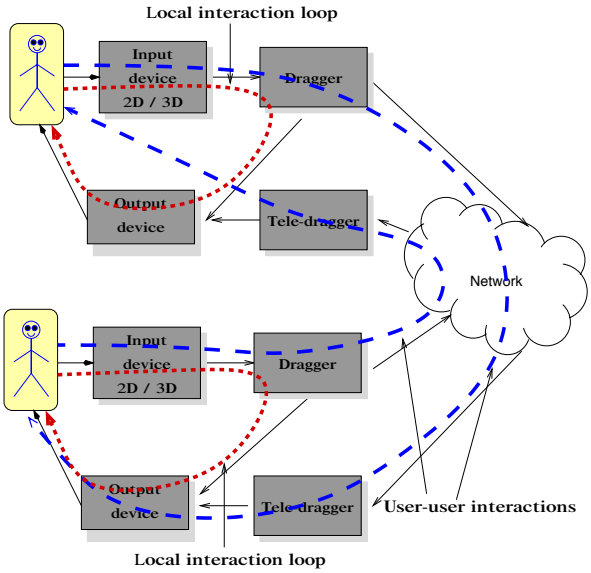

Figure 4: User–user interactions.

the pointing devices and the volume view that we chose are, respectively, colored graphical representations of the manipulators and colored wireframe boxes. Furthermore, we allocate an event channel only for replications of the manipulators and view volumes.

3.4 Graphics Attributes Synchronization

For providing appropriate visual feedbacks, positions of a pointing device that a user manages should be shown in the *scene view* sub-window. At the same time, for the sake of group awareness, these positions must be sent to all instances of the user workspace application, inclusive the instance that generates the event, in order to update the content of every *global view* sub-window. This means that both sub-windows are selected and their corresponding event handlers are invoked for, concurrently, redisplaying. These handlers need to access correct graphics states for correctly re-rendering the graphics objects. Otherwise, incorrect drawings may be generated.

The solution that we propose for synchronizing the graphics states with the selected windows is to explicitly issue the command that make the graphics states of any focused window as the current states. In this way, the handler can always draw the objects with the expected attributes in each window.

4 EXPERIMENTS AND CONCLUDING REMARKS

To validate our proposal, we implemented a new version of CoMo with extended functionalities, installed in different machines (Sparc with Elite 3D, and PCs with FX6600 and GeForce 8600GT), and measured latency parameters over 1.0Mb/s, 10.0 Mb/s and 1.0Gb/s LANs. We collected some measure about latencies parameters proposed in this paper.

The dragger latency lied in the range 0.1–1.0s for all machines. While, the results collected for object manipulation shows the transmission rate did not affect the object latency. It's because the size of the geometric data was much smaller than the network capacity. It is one of the advantages of replicated and hybrid architectures over a centralized architecture. The tele-dragger latency has been the most critical one, since the user interacts continually with it. Therefore, we shows that the latency of the tele-draggers is dominantly dependent on the network transmission rate.

Based on experiment, our proposed solutions is close, but does not satisfy completely yet, the recommended interactivity metrics. We have work in thee system reimplementation, considering lightweight communication protocols.

The main contribution of this work is to demonstrate that it is feasible to design an interactive and usable system in a heterogeneous multi-platform environment, where the shared data consistency must be ensured. Besides, rendering parameters and display can be adapted accordingly with local system resource to assure individual performance. We believe that a hybrid architecture may becomes a good alternative for any multi-platform application whose the most important requirements are the robustness and the adaptability to local computational resources.

REFERENCES

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

[2] S. Greenberg, S. Hayne, and R. Rada. *Designing Groupware for Real-Time Drawing*. McGraw Hill, 1995.

[3] L.G. Silveira Jr and S.-T. Wu. An object-oriented groupware framework for developing collaborative 3d-modelers. In Thierry Priol e Jamie Painter, editor, *Third Eurographics Workshop on Parallel Graphics & Visualisation*, pages 103–114, Girona, ESP, Sept. 2000.

[4] M. de G. Malheiros, F. N. Fernandes, and S. T. Wu. Mtk: A direct 3d manipulation toolkit. In *SCCG'98 Proceedings*, pages 81–88, Brastilava, april 1998.

[5] D. Michelucci. An introduction to the robustness issue. In *Swiss Conference of CAD/CAM*, pages 214–221, Neuchâtel, Switzerland, Feb. 1999.

[6] J. Neider, T. Davis, and M. Woo. *OpenGL - Programming Guide - Release 1*. Addison Wesley Co., 1993.

[7] Jakob Nielsen. *Usability Engineering*. Morgan Kaufmann Publishers, 1994.

[8] Object Management Group (OMG). The common object request broker: Architecture and specification, Dez 2001. Version 2.6.

[9] L.G. Silveira Jr and S.-T. Wu. Towards consistency in a heterogeneous collaborative geometric modeling environmen. In *Proceedings of SIACG 2002*, pages 139–148, Guimarães, Portugal, 1–5 July 2002.