Workspace Awareness in Relaxed WYSIWIS Systems

WU, SHIN - TING
LUIZ GONZAGA DA SILVEIRA JR.

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

Abstract. CoMo is a prototype of a collaborative 3D geometric modeling environment designed to support temporally and geographically dispersed work teams. In this environment the participants may work in different parts of the space and have distinct perspective viewing of the space. Maintaining workspace awareness becomes, therefore, much more difficult. In this paper we present graphics solutions that we adopted for enhancing the 3D workspace awareness in CoMo. We reused several individual awareness mechanisms provided by the MTK graphics library designed to single-user applications. Appropriate communication and distribution platform for coordinating individual actions was devised. The main contribution of this paper is to show how group awareness can be implemented by adequately reusing the existing individual awareness facilities.

1 Introduction

The potential benefits of systems that facilitate group working on product design seems to be huge. The typical scenario for collaborative modeling is a shared workspace, where a geographically dispersed small group of participants works in a loosely-coupled mode for constructing and modifying an application-dependent 3D-model over the Internet. Two issues underlie the usability of such a system are robustness and responsiveness.

CoMo [15], acronym for *Co*laborative *Mo*deler, is a prototype of an heterogeneous collaborative 3D geometric modeling system designed to meet a tradeoff solution for keeping the consistency of 3D-model and for preserving the system/network time response. The basic idea consists in separating geometric modeling and interaction activities (rendering and manipulation). 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. Graphics interactions, on their turn, are performed on the replicated graphical model residing in local machines.

An important feature of CoMo is that the rendering mode at each participating machine are tailorable to the local computing power. Besides, participants may be working in different parts of the space with distinct aim points. This approach reduces the group focus that characterizes strict-WYSIWIS (*What You See Is What I See*) oriented systems and makes the design of 3D workspace awareness a challenging issue. Workspace awareness is knowledge about others' interaction with a shared workspace [10]. This includes the understanding of who is in the workspace (participant awareness), where they are working (location and proximity awareness), what they are looking at (perspective awareness), and what they are doing (activity awareness). The way how the workspace awareness could be enhanced depends greatly on the tasks involved, the users, and the environment. Once it is recognized that the workspace awareness is an important factor in the usability of interactive groupware (it helps users to simplify verbal communication, helps them coordinate actions, and helps them anticipate others' actions and intentions), we have designed, on the basis of the existing theoretical and practical results, a set of awareness widgets to CoMo system.

The implementation of such widgets require two kinds of functionalities: communication services and 3D graphics facilities. A freeware implementation of CORBA standard [7], called MICO [14], and MTK are used to provide such capabilities. CORBA standard provides abstractions for distributed object-oriented programming into heterogeneous distributed systems, whereas MTK is an enhanced loosely-coupled 3D graphics library equipped with 3D interaction facilities to enable effective 3D interactions [15]. MTK provides several functions for implementing widgets that enable maintaining individual awareness while users interact. It was originally designed for single-user purposes.

The objective of this paper is twofold: to show how to enhance a 3D graphics library for visually providing appropriate group awareness and to show how the implementation of group awareness widgets can be benefit from 3D individual awareness widgets designed for loosely-coupled graphics interfaces. Our results indicate, for example, that 3D Cursor and draggers in MTK is valuable not only for individual manipulations but also for conveying 3D workspace awareness.

In the next section, we give a summary of work that influences the design of our widgets. In section 3 the architecture of CoMo is briefly described, whereas some important features for individual awareness supported by MTK are given in section 4. The designed awareness widgets are presented in Section 5. Some snapshots of the CoMo interface are collected in Section 6 to try to convey the dynamics of CoMo and its visual feedback. Finally, some reflections about our work are summarized in section 7.

2 Related Work

There is a number of collections of papers that address computer supported cooperative work concerns in a range of aspects, including hardware, software, services and social issues [13]. The systems typically provide two basic services: communication channel and a shared workspace. In this section we restrict our discussion to the work related with the shared *workspace awareness*.

Observational studies on face-to-face collaborations have found that in coordinating work activities all of our perceptual abilities are used to maintain knowledge about others' interactions with shared artifacts [16]. Gutwin and Greenberg [9] have shown that better support for workspace awareness in groupware systems can improve the usability of these shared computational workspaces.

Several CSCW projects have considered support for 2D workspace awareness. One approach is to include new interface components – *awareness widgets* – in the user interface to support loosely coupled styles of collaboration. This means that the co-workers can perform their activities in a parallel, but coordinating, way.

The most known 2D awareness widgets are the radar views, multi-user scrollbars, telepointers, WYSIWID (*what you see is what i do*) view, and teleports. Gutwin et al.[10] conducted experiments that examined the usability of these widgets added to a real-time distributed groupware system for construction activities (moving, arranging, and aligning objects in 2D space) among nine pairs of computer science students. Some important findings of their work are:

- "... the idea of workspace awareness should cover more than just knowledge of others' interactions with the workspace; it also includes knowledge of the state of the workspace and its artifacts, and of your own actions within that context... Global overviews seem particularly useful in this light";

- "It is commonly thought that distraction is caused by perceptual information that draws our attention, but distraction may have as much to do with interpretation difficulty ... the awareness information must be easily interpretable regardless of where it is presented ... one way of simplifying interpretation is to build on people's existing knowledge of the workspace";

- they reinforced the principle stated by Dourish and Belloti [3] that awareness information should be passively collected and distributed by the system, rather than explicitly generated by the participants.

Later, an experiment to compare two versions of a groupware interface – one with basic miniature and the enhanced miniature or a radar view – was thoroughly performed and they concluded that radar view providing "indications of other person's location, the location of his and her cursor, and the objects that he or she moved ... helped people complete some tasks more quickly and more efficiently" [9].

The rapid emergence of 3D virtual environments propelled researches in 3D manipulation metaphors [1, 17, 2] and 3D workspace awareness widgets. Recently, Dyck and Gutwin [4] did studies on some 3D workspace awareness widgets for maintaining the locations, perspectives, and the proximities of co-workers. Their experiences suggest that the embodiment enhancements (explicit representation of the embodiment of each co-worker in the workspace), participant list enhancements (explicit representation of data about the co-worker), and alternate views of the workspace (explicit representation of different perspectives on the workspace) play an important role in supporting such kind of awareness.

On the basis of the reported empirical experiments, we designed three awareness widgets with use of MTK [11, 15] for supporting 3D construction tasks among a small size of participants.

3 CoMo: A Collaborative Environment

The underlying architecture of CoMo 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.

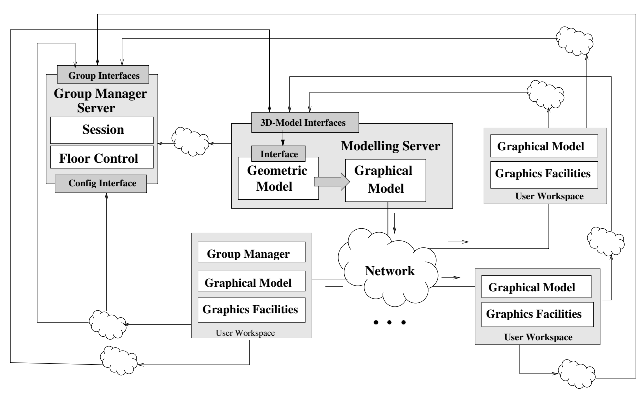


Figure 1: An overview of our proposal

In this approach, the *geometric modeling kernel* resides in a central server. In the current version, an instantiation technique, extrusion and boolean operations are provided for constructing polyhedral objects. Conventional geometric transformations (translation, rotation, and scaling) are also supported.

The visual *user workspace* interface is very simple. It consists of one scene window, one 3D radial window, and one participant list window. The workspace at each participating machine runs under the control of a X window system. We used the built-in set of widgets provided by GTK+ [8] to implement these windows. It is because that GTK+ provides a uniform handling of local and networked events. A participant directly interacts with objects in the scene window by clicking and dragging the box/sphere draggers provided by MTK, which, on its turn, accesses the functionalities of OpenGL [12] for efficient rendering (Section 4).

The separation of *geometric modeling kernel* from *user workspace* allows a participant to personalize her/his interface by setting rendering and manipulation parameters in accordance with local hardware capabilities. Furthermore, since the geometric decisions are under central control, design issues for a robust monolithic modeling system are applicable to the geometric kernel. A *graphical representation* of the underlying 3D geometric model is replicated and sent to the participating machines on demand. In this way, graphics hardware acceleration features may be better explored to maximize local performance.

Several users may interact together with the shared 3D model. For avoiding conflicts in inputs and providing group awareness, there is a *group manager server*. This server holds a repository containing information about users connected to session, policies for group joins, and a floor control mechanism. For simplicity, and without deviating from the main objective of our work, we consider that the floor control strategy is 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 communication channels are supported by the available *distributed object environment* – a freeware MICO [14], which is an implementation of the CORBA specification [7]. CORBA is an object-oriented infrastructure that allows objects communications, independent of specific platform and techniques used to implement these objects. The CORBA core hides low-level details of platform-specific networking interfaces, allowing developers to focus on application development, instead of having to build the network communication infrastructure. The CORBA objects and their interfaces (methods) are specified via IDL (Interface Definition Language). Therefore, they can be transparently invoked in the remote way. The CORBA specification includes several services, such as Naming, Transactions, Concurrency Control, and Event, that facilitate the development of distributed object applications, are also included.

In the implementation of CoMo, Naming Service is used to name and categorize the geometric, graphics, and group information, while Event Service is applied to decouple the event producers and consumers and to provide facilities for reliable one-to-many communication through one (or more) event channel [6]. More than one event channel are allocated for active and passive asynchronous data sending.

The Event Service is used in graphical model replication. We adopted the push-model configuration with one event channel: the graphics model residing in the 3D-model manager is the master and the replicated models in the user workspaces are slaves. Whenever a change occurs in the geometric data, the master graphics model posts the event-data (updated graphics data) into the channel and the slaves (the slave/replicated graphics model) consumes the event-data from this channel (Figure 2).

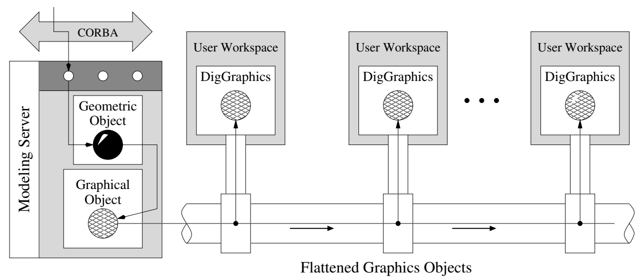


Figure 2: Graphics objects replication

Nevertheless, we also have graphical information that must be replicated but not be retained for enhancing the group awareness, such as the current viewing volume of each participant and their interaction artifacts, as will be explained in section 5. For this reason, there is a second event channel for conveying these data. Whenever a participant changes the state of her/his interaction artifact, the local graphics model multicasts the updated data to others' machines.

A third event channel is necessary for the *group manager server* to communicate with all the participating machines whenever the state of the group session is changed (e.g. joining or leaving of a participant).

4 MTK: A Graphics Toolkit

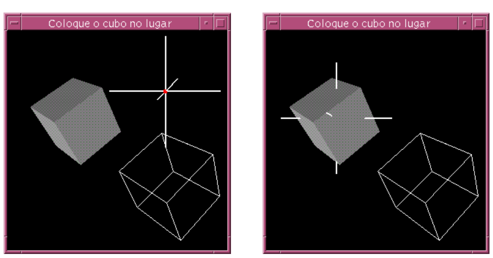
We give in this section a brief description of MTK, also designed and implemented by our research group at State University of Campinas. This toolkit plays an important role in the design and implementation of our awareness widgets.

MTK [11, 15] is a C++ programming library aiming at providing a simple, direct interface to the fundamental operations of 3D graphics rendering and interactions. Its design differs essentially from the most known higher level 3D graphics toolkits, such as Open Inventor [18]. Instead of integrating application and the user interface into the same development environment, MTK provides a loosely-coupled graphics development environment.

MTK includes a subset of the conventional 3D graphics library functionalities, namely the set of basic geometric objects, display list (wrapped by the `Geometric Models` class), multiple viewing (wrapped by the `Cameras` class), multiple lighting (wrapped by the `Lights` class), and picking/selection (wrapped by the `Selection` class). In this way, for visualization purposes, the coupling of an application-specific 3D model and MTK may be reduced to a data format conversion problem.

Additionally, MTK also provides interaction facilities: (1) a 3D-cursor for indicating visually a virtual 3D-mouse position (`3D Cursor` class), (2) a set of pictorial representations to improve the 3D depth perception (`Guides` class), (3) a set of pictorial representations that are relevant to realize 3D interaction metaphors (`Draggers` class), and (4) a way to customize the relationship between a geometric element and a 3D interaction metaphor (`Constraints` class).

The application-dependent geometry is useful in moving controllably a virtual 3D-mouse on the surface of any object in 3D scenes with a 2D-mouse. The position of the 2D-mouse is "unprojected" in 3D space through an implicit function that defines the surface. A 3D-mouse can also move freely in a 3D scene. In this case, we constrained the 2D-mouse movements on either xy- or yz-plane in R^3 according with the predefined operation mode. The position of the virtual 3D-mouse is visually represented on the screen by the conventional 2D-cursor in cross-hair shape. All these methods for managing a virtual 3D-mouse are encapsulated in a `3D Cursor` object. Figure 3 shows a 3D-cursor on the (a) outside and (b) inside of an object in a 3D scene.

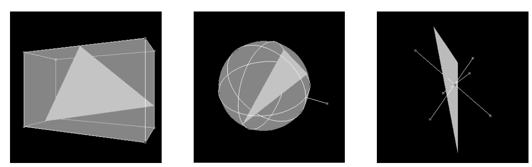


(a) (b)

Figure 3: 3D Cursor

Both `Draggers` and `Graphics Models` objects are selectable when a user picks on their elements – `Handles`/(Sensitive) `Parts` and `MtkElement` objects, respectively – and respond by delegating the request to application-dependent objects for handling them properly. They also have capabilities for redrawing by themselves when their attributes are changed. At this point, one may argue what is the difference between them. The difference lies in the way that they handle the subsequent input events. Predefined subsequent 2D positioning events are captured by a dragger and transformed into 3D points. Then, the point is passed to the client for further processing. Whereas a geometric model is passive, in the sense that it just passes the event to the client for specialized dealing. The unambiguous 2D–3D mapping in a dragger is achieved by using a `Constraints` object to which each sensitive part must refer. Currently, four subclasses of `Draggers` are implemented:

- box: with a box geometrical representation. It comprises twenty-six sensitive parts: eight vertices, twelve edges, and six faces (Figure 4a).
- sphere: with a spherical geometrical representation. It comprises six sensitive parts: the sphere surface, three orthogonal rings, a handle, and the handle vertex (Figure 4b).
- reference frame: with a three-orthogonal-axis geometrical representation. It comprises thirteen sensitive parts: six edges and seven vertices (Figure 4c).



(a) (b) (c)

Figure 4: Draggers

It is worth noting that the visual feedback of draggers and geometric models is totally decoupled.

Summarizing, MTK is comprised of the `MtkCore` class and eight "independent" abstract classes: `Graphics Models`, `Cameras`, `Lights`, `Selection`, `Guides`, `Constraints`, `Draggers`, and `3D Cursor` (Figure 5). The `MtkCore` coordinates the interaction between these classes by delegating requests in order to realize a semantic action. For example, the `Selection` passes to the `MtkCore` the identifier of selected elements and the method `cameraPointer()` in `MtkCore` is responsible for deciding whether a part of a dragger or a graphics model should be notified to handle the subsequent events.

5 Awareness Widgets

Our design is based on the hypothesis that a good visual feedback (perception) for individual actions in a single-user

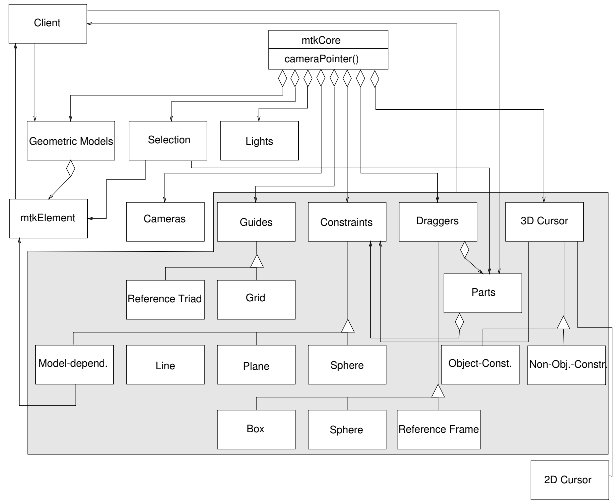


Figure 5: MTK framework

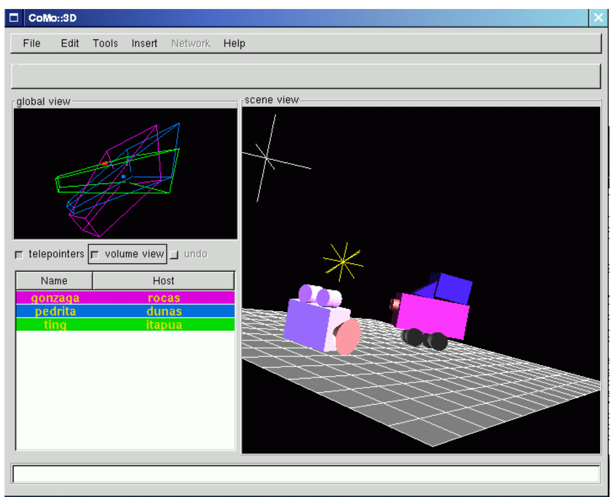


Figure 6: CoMo's Interface.

application may be valuable in a collaborative workspace, when more than one user have a common object of interest. Otherwise, it is sufficient to keep a simplified view of the entire workspace in order to maintain "group awareness" and to facilitate activities synchronization whenever it is required.

Hence, in addition to the conventional drawing area (the scene view) where users can interact individually and directly with the 3D model with the use of draggers or 3D cursors, two windows are included in the interface of CoMo for conveying a coarse global view of the state of the workspace: a listbox that contains the participant names and a second drawing area (global view) where a simplified version of the global overview of the 3D shared workspace is presented (Figure 6).

Each participant's screen view occupied most of their screen. By manipulating the camera parameters, participants can refer to particular objects or have a particular view of the workspace while performing their own goal tasks. A participant can also adjust her/his viewing parameters to be the same as the ones of another participant by clicking on the corresponding name in the participant list. The user can also back to her/his previous view by undoing the action.

To each co-worker is assigned a unique color when she/he joins to the working session. This color is consistent across other participant's views and across the windows in each individual workstation.

5.1 Participant List

The paricipant list is useful for locating a particular co-worker. From it we can promptly find out which users are currently logged in and which color each of them is assigned to. From the color attribute, we may locate not only their position in the shared workspace but also their current activities, as explained in Section 5.2.

The control of participant flux is carried out by the *group manager server*, which passively updates its data base, generates and distributes an event to all of the participating machines whenever a participant joins to or leaves a working session. Because the window manager GTK+ supports networked events, the invocation of the appropriate event handler for updating the participant list at each participating machine is easily programmable.

Differently from the traditional participant lists, which typically contain images or model of each participant, our participant list is a simple scrolled list with colored participant's names (at the bottom of Figure 7). This reduces the occupation of screen space, without missing the basic group awareness information. We will see that from the list a user may identify from her/his assigned color the location and activities of each co-worker in the 3D shared workspace. Observe that in the *global view* window the volume view of each participant is visualized.

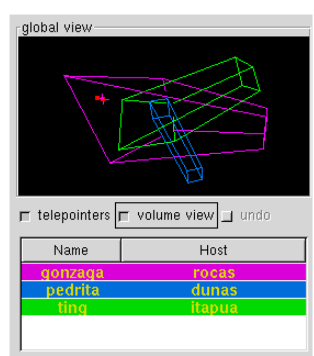


Figure 7: Global view and participant list.

5.2 Global View

Because of the narrow field view offered by computer displays, the co-workers can easily be out-of-view in a relaxed WYSIWIS system. As a solution for this frequent awareness missing, alternate views of the shared 3D workspace are recommended. Dyck and Gutwin [4] developed the Grand Tour widget, a 3D version of the radar view. The Grand Tour shows the view of a distant camera that is constantly in motion (orbital view).

We developed a similar widget called *global view* to provide a view of the entire workspace. The main differential of our approach relies on the possibility in setting the desired viewing parameters on demand. Following the recommendation of Ellis et al [5] that "a good group interface should depict overall group activity and at the same time not be overly distracting", only the currently active 3D cursors and draggers are shown in the *global view*. Whenever a user interacts with them, a networked event is generated and multicasted to all the rest of participating machines for maintaining an ongoing awareness of others.

Besides the currently active 3D cursors and draggers, the volume view of each participant is displayed in a colored wireframe box in the *global view*, in order to enhance the perspective awareness.

5.3 Teledraggers

As already mentioned, the draggers indeed emulate 3D input device. On the basis of registered constraining functions, they are able to map a 2D input point into a 3D point. Moreover, they can redraw by themselves when their attributes are changed by the application. One useful application of a dragger in a single-user system is the pictorial representation of a 3D manipulation metaphor [13], which is called *manipulator*. It captures a sequence of 2D input points which is translated by the manipulator in a semantic value, such as displacement, rotation angle, etc. This information is then applied not only on the dragger itself but also on the corresponding controlled 3D model. For this reason, from the dragger movements the user can infer the activities that the other participant performs in the shared 3D workspace.

As already mentioned, the 3D application model resides in the *geometric (modeling) server*. Hence, the activity awareness of CoMo may benefit from the model-dragger decoupling design of MTK. An immediate semantic feedback may be provided while the user is interacting with a dragger, even though a geometric server needs longer time to accomplish a task, as illustrated in the sequence diagram of Figure 8. Observe that the information of the dragger is replicated to the other participating machines via another event channel. Hence, the latency (the update speed of an image in response to a user action) of a dragger is less than the latency of any 3D model.

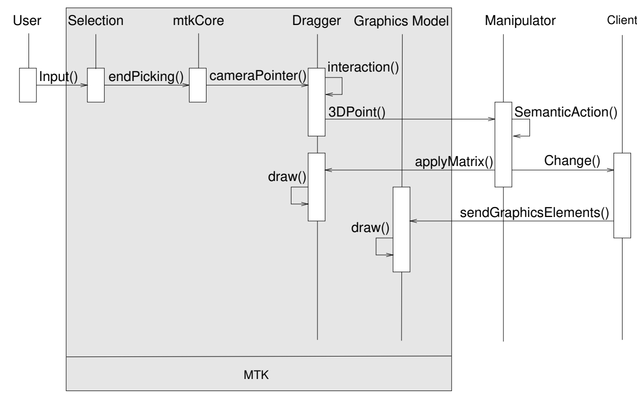


Figure 8: Interaction sequence

For avoiding visual clutter in the scene view, the state of teledraggers manipulated by other participants can only be followed from the *global view*. Figure 9, where we can see three teledraggers associated to three distinct participants, is a zooming view of the scene presented in Figure 7 from one participating machine.

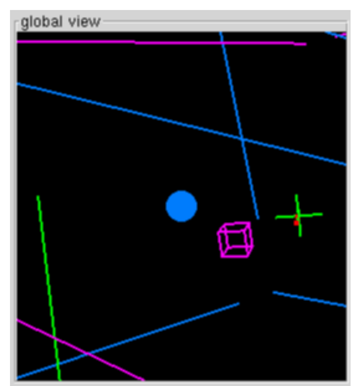


Figure 9: 3 teledraggers in the global view.

5.4 A 3D Telecursor

In MTK a 3D cursor is a positioning widget that represents the position of a virtual 3D-mouse under control of a user. In general, it conveys the point of interest of its user. The movements of virtual 3D-mouses are restricted to xy-, yz- or xz-planes at time. As shown in Figure 3, the 3D cursor is visually represented by the conventional 2D-cursor in cross-hair shape whose "unprojected" axes are parallel to the axes of viewing coordinate system of its user. This implies that the orientation of cursor axes is dependent on the viewing direction of its user. This leads us to modify the shape of the 3D cursor slightly for making it more suitable to indicate the viewing direction of its user. We simply added a point on the extreme of the negative z-axis of the old version of 3D cursor, as illustrated in Figure 10.

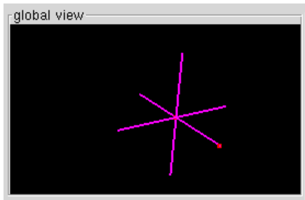


Figure 10: A new shape of 3D Cursor

For the same reason given in section 5.3, the 3D telecursors associated to the other participants are only displayed in the *global view* with the corresponding color attribute.

6 Results

In this section we attempt to show the dynamics of the interface of CoMo through some snapshots.

We consider that there are three participants (Gonzaga, Ting, Pedrita) sharing a 3D workspace comprising one toy car, one camera, and a lighting source. Reference axes and a reference plane are included to enhance 3D perception. Gonzaga is working on a machine with Creator 3D graphics accelerator (UltraSPARC/60 – rocas), Ting is working with a machine without graphics accelerator (UltraSPARC/1 – buzios), and Pedrita is logged at a machine with Elite 3D graphics accelerator (UltraSPARC/10 – dunas). For machines with and without graphics accelerator, Gouraud and wireframe shading were, respectively, chosen (Figure 11).

Observe that in the *global view* window at each participating machine the volume view and the interaction tool of each participant is presented. Ting (in green) is interacting with a 3D cursor, Gonzaga (in magenta) with a box teledragger, and Pedrita (in blue) with a sphere teledragger (Figure 9).

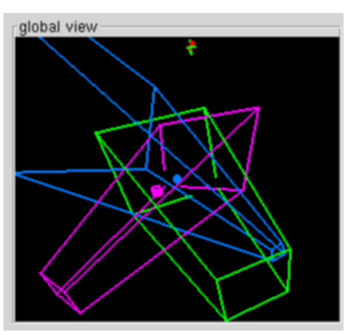


Figure 12: *Global view* of Gonzaga.

Figure 12 shows the consistent updates in the *global view* at the machine where Gonzaga works, after Pedrita having changed her viewing parameters and Ting moves her 3D cursor (observe that it is outside of her view field). This ensures both the perspective and the activity awareness.

7 Concluding Remarks

On the basis of existing studies on the workspace awareness in real-time groupware systems, a solution for enhancing the 3D workspace awareness in a collaborative geometric modeller, typically with a small size of participants, was presented. A scrolled participant list, an alternative for 3D radial view, volume view, 3D telecursors, and 3D teledraggers have been used in our implementation.

To the best of our knowledge, 3D telecursors and 3D teledraggers are new interaction techniques that we have developed. A 3D telecursor results from the combination of the functionalities of a classical 3D telepointer (providing the location awareness) and the nose ray (identifying the line of sight) [4]. From our observations, a 3D teledragger has shown to be effective in supporting task awareness. Moreover, because of a small size of participants, it is possible to display all of the interaction artifacts in the 3D radial view, without window clutter. Moreover, we may avoid the users being distracted when they are performing their own goal tasks in the scene sub-window.

Nevertheless, from our point of view the main contribution of this work is to have demonstrated that, conceptually, several individual awareness mechanisms designed to single-user applications can be easily adapted to provide workspace awareness. The reuse is greatly facilitated if the implementation of those individual awareness mechnisms follows the loosely-coupled approach, such as the architecture of MTK.

As further work, experiments must be carefully conducted to evaluate the usability of the interface of CoMo, and consequently, the two new widgets.

References

[1] Eric A. Bier. Snap-dragging in three dimensions. In Rich Riesenfeld and Carlo Sequin, editors, *Proceedings of the 1990 Symposium on Interactive 3D Graphics*, pages 193–204, Snowbird, Utah, 25-28 March, 1990, march 1990.

[2] D. Brookshire Conner, Scott S. Snibbe, Kenneth P. Herndon, Daniel C. Robbins, Robert C. Zeleznik, and Andries van Dam. Three-dimensional widgets. In *ACM Symposium on Interactive 3D Graphics, Special Issue of Computer Graphics*, volume 26, pages 183–188, 1992.

[3] P. Dourish and V. Bellotti. Awareness and coordination in shared workspaces. In *Conference on Com-*

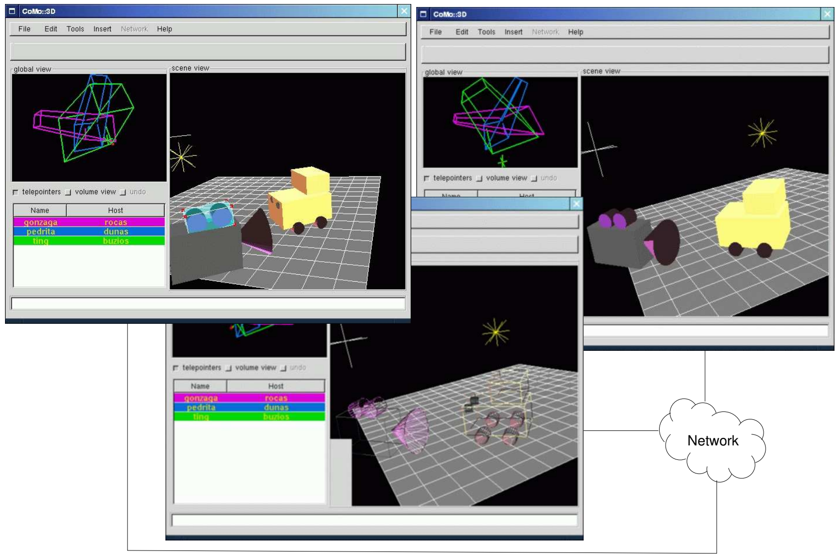


Figure 11: An overview of a working session.

puter Supported Cooperative Work (CSCW'92), pages 107–114, Toronto, Ontario, 1992. ACM Press.

[4] Jeff Dyck and Carl Gutwin. Awareness in 3d collaborative workspaces. In *ACM Graphics Interface (GI'2002)*, `http://hci.usask.ca/publications/2002/groupspace-gi/index.xml`, 2002. to appear.

[5] C. Ellis, S. Gibbs, and G. Rein. Groupware: Some issues and experiences. *Communications of the ACM*, 34(1):38–58, 91.

[6] OMG Object Management Group. Corba event service specification - v1.0. `http://www.omg.org`, 1993.

[7] OMG Object Management Group. The common object request broker: Architecture and specification - version 2.0, 1995.

[8] GTK+. Gtk+ - the gimp toolkit. `http://www.gtk.org/`.

[9] Carl Gutwin and Saul Greenberg. The effects of workspace awareness support on the usability of real-time distributed groupware. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 6(3):243–281, Sept. 1999.

[10] Carl Gutwin, Mark Roseman, and Saul Greenberg. A usability study of awareness widgets in a shared workspace groupware system. In *Computer Supported Cooperative Work*, pages 258–267, 1996.

[11] 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.

[12] Jackie Neider, Tom Davis, and Mason Woo. *OpenGL Programming Guide: The Official Guide to Learning OpenGL, release 1*. Addison-Wesely, 1993. ISBN 0-201-63274-8.

[13] Jenny Preece, Yvonne Rogers, Helen Sharp, and David Benyon. *Human-Computer Interaction*. Addison-Wesley Pub Co, 1994.

[14] Arno Puder and Kay Roemer. *MICO: An Open Source CORBA Implementation*. Morgan Kaufmann Publishers, Mar 2000.

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

[16] J.C. Tang. Findings from observational studies of collaborative work. *International Journal of Man-Machine Studies*, 34(2):143–160, 1991.

[17] Maarten van Emmerik. A direct manipulation technique for specifying 3D object transformations with a 2D input device. *Computer Graphics Forum*, 9(4):355–361, 1990.

[18] J. Wernecke. *The Inventor Mentor*. Addison Wesley, 1994.