

PEER-TO-PEER TECHNIQUES APPLIED TO THE COOPERATIVE VISUALIZATION OF CAD MODELS

Diego BORRO¹, Iñigo RECIO², Carlos PEDRINACI³, Hector SANCHEZ⁴ and
Alejandro GARCIA-ALONSO⁵

¹CEIT(Centro de Estudios e Investigaciones Técnicas de Guipúzcoa), B. Ibaeta,
20009 San Sebastián (Spain)

²OPTENET, Paseo Mikeletegi, 58 - 1ª planta, 20009 San Sebastián (Spain)

³PARQUE TECNOLÓGICO MIRAMON, Paseo Mikeletegi, 53
20009 San Sebastián (Spain)

⁴CENTRO UNIVERSITARIO DE MÉRIDA, Universidad de Extremadura, Santa
Teresa de Jornet 38, 06800 Mérida (Spain)

⁵FACULTAD DE INFORMÁTICA, Universidad del País Vasco, P. Manuel Lardizábal
1, 20009 San Sebastián (Spain)

ABSTRACT

We describe a specific problem within the foundry production area and its need for a cooperative visualization and geometry verification tool. We discuss the opportunity of building a specific application and its features: communication architecture and user interface. The conclusion is that it is possible to develop very specific and powerful products for computer supported cooperative visualization and verification using Java 3D. We show the techniques we are using to keep our prototype useful even in very narrow bandwidth conditions. As a consequence of the experience gathered we have developed a new public Java package. This package makes easier the development of cooperative applications based on the Internet and can be adapted to different and specific visualization or simulation problems.

INTRODUCTION

Next we present the foundry process, a problem that requires a cooperative visualization and geometry verification tool, and its requirements.

We have analysed a process that involves people placed at three different sites. We will refer to them as the customer, the caster, and the modeller. The customer orders certain number of identical parts from the caster and gives him the drawings that define the part. The caster passes the drawings to the modeller, who machines the model that is used to make moulds that will contain the melted steel.

The design cycle has the following steps: *design the model*, machine the model, make the mould of each part, cast each part, break the mould, remove residues and finish the part, and check the part.

Both parties are interested in using the first model to create valid parts: if the cast generates wrong parts, the iteration cycle must be repeated. The cycle involves a first software related process and then several production processes. Production processes have a lot of manual work associated with them. So, the creation of the design requires a close cooperation between engineers from the caster's factory and designers from the modeller's one. Customers are seldom involved in these discussions, but they could be. Up to now that collaboration requires daily, or even more frequent, trips from the modeller's to the caster's because it is not enough to

send the design and talk over the telephone: *both parties need to chat viewing the same aspects of the design.*

Trips can be saved using an application for cooperative visualization: as one user changes the 3D view in a local screen, the same view appears in the screen of all the other users. The tool must also provide some means to query geometric properties within the model and other verification tools. A list has been defined with all the requirements as they were defined by a group of users assisted by the developers of the application [1].

We have considered the following types of cooperative applications related to the foundry area: General-purpose cooperative tools, CAD cooperative tools, and CAD cooperative and design integration tools. We have also analysed applications from other areas that allow users to visualize a common environment. All this data has been compiled in a report [1].

After seeing the disadvantages of the previous applications (no real-time, high cost, etc.) we decided to develop a custom-made cooperative visualization tool suitable for a restricted set of users, following their needs as close as possible. The following sections describe the 3DSHARED prototype, which follows the GNU license¹.

3DSHARED is a freeware cooperative visualization application developed in the computer science faculty of San Sebastian – Spain. It is totally independent of the platform and it uses the Internet protocols to communicate with the hosts that participate in a cooperative session.

It is composed of two modules: **3D API** and **Collaborative API**. The 3D API takes care of the aspects related with the visualization of three-dimensional models. The Collaborative API ensures the correct communication among the members of the cooperative session. As the design and implementation of both modules is independent, 3DSHARED offers the functionalities of both modules in the same interface. Consequently, the cooperative visualization of three-dimensional models is allowed.

DYNAMIC CLIENT/SERVER ARCHITECTURE

The Collaborative API represents a new communication architecture based on two well-known architectures: client/server and distributed [2].

In the client/server architecture, the server manages the transfer of messages among the participants. It receives messages from clients and replies them to the other hosts. This is a centralized management. Although this architecture is easily implemented, a server must firstly be started in order to allow the connection of clients [3]. Even more, a failure in the server causes the whole system failure of the system and this kind of architecture is poorly scalable. These problems can be solved with the hybrid client/server architecture [4]. However, these solutions need to develop and support two different applications (the server and the client).

¹ Code and documentation (installation, user manual, etc) are available on the web <http://scsx01.sc.ehu.es/ccwweb3d/3dshared>.

In the distributed architecture, all the hosts develop the same function. Therefore, only one application exists. As there is no server role, it is necessary to hold connections among all the hosts [5]. However, in our implementation, most of these connections are useless during the session because each host will only use the connections with the host that owns the control of the system. Therefore, the application loses the management simplicity.

We have designed an hybrid architecture, that avoids these problems: the **dynamic client/server** architecture, that avoids these problems. It is based on a “peer to peer” strategy [6] and any node can take the server role. At any time a client host can ask for the control of the system and shift from client role to server role. The server node, also called controller, is the manager of the communications. If the server node falls down there is a replacement mechanism that assigns the server role automatically to a client host.

Depending on the situation of the collaborative session the Collaborative API acts either as client or server, in a way that is transparent to the user. This avoids the development and co-existence of two different modules. If the system control is transferred to another host, the old controller host switches the role with the new controller (see Figure 1).

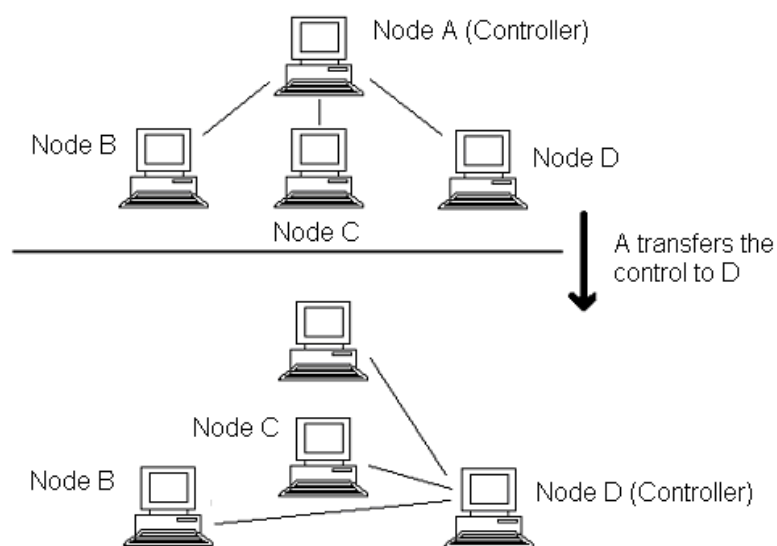


Figure 1. Dynamic client/server architecture

When a participant wants to join to the session, it can connect to any participant. If this host is not the controller, the application intelligently redirects the connection to the controller. A user can be accepted, rejected or asked to join a cooperative session. This allows a robust management of the group.

The access control to the cooperative session is managed by the **token mechanism** that establishes that only the controller can interact with the model. Changing the control over the system can be accomplished for two reasons: the controller does not want to own the control and gives it to a client host; a client host wants to become the

controller and requests it. This provides a higher flexibility to the system: the capability of transferring the token dynamically during a cooperative session.

COMMUNICATION OF INTERACTIONS

In a collaborative session, the controller host must communicate the interactions its user is performing over the model. Following these commands the participant hosts can generate the same scene on their respective screens.

3DSHARED uses a short message policy. These messages are sent only when there is a change in the state of the scene (position, orientation or colour change, etc.). As a consequence, the information flow among the participants is low and constant, achieving in most cases nearly real-time synchronization.

Using TCP/IP protocol, secure and ordered transmission is provided. Therefore, it is not necessary to use specific hardware like multicast routers or D class IP addresses or other solutions like *JavaGroups* [7] or *Ensemble* [8].

The possibility of sending text messages has been integrated in the application. This functionality (*chat*) is very useful in collaborative applications.

The user of the controller node is able to manipulate the user interface in order to change the visualization parameters. These interactions are captured by the 3D API. Each controller's action over the interface generates an update message that is queued in order to be sent to the clients.

We have been very concerned about the lack of bandwidth for the communications. Thus we have built an option to control the data transfer flow: the controller can uphold the transfer of messages. So clients will not receive changes in the scene as they occur. Changes are queued and they are sent only when the user of the controller node requests the application to send them. Consequently, the 3D API provides two communication types: **continuous mode** and **send by order mode**. Viewing the object in motion is an important cue to help our mind in the recognition of 3-D models; for this reason continuous mode is the most used. However, if the Internet is very busy, then it is worthwhile to lose this cue to avoid the annoying effect of display lag.

The Collaborative API of the participant hosts receives the messages. These messages are interpreted. If a message announces the opening of a file, the system checks whether that file is locally stored or not. If it is not, its transfer is requested and the application is interrupted until transmission is finished. If a message encapsulates an update of the scene, it is queued and the 3D API interprets it and reflects the changes in the scene that encapsulates it. The communication process can be seen in the Figure 2.

The Collaborative API is a general purpose API [10]. It contains the client/server architecture and the communication protocol but not the message policy. This feature allows its implantation in several domains. It is only required to define the message policy and how the 3D API must act when it receives the message.

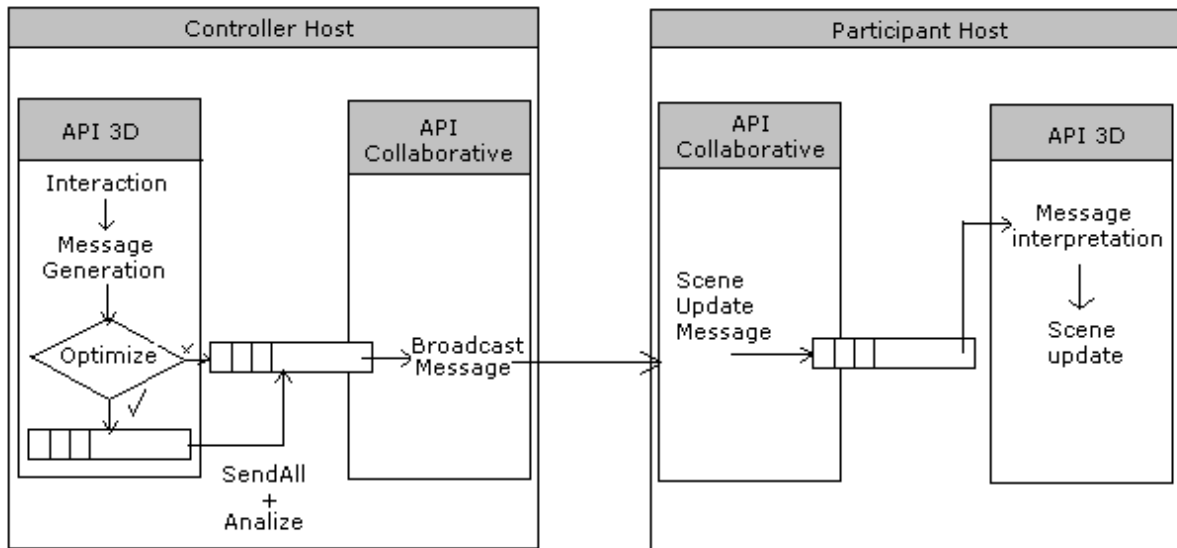


Figure 2. Communication of the update messages among the 3D API and the Collaborative API

USER INTERFACE OF THE “3D-Shared” PROTOTYPE

Each user will have one instance of the same program, which is presented in Figure 3. It has three communication tools: a chat tool, a file transfer tool and a 3D-window where the same view for all the participants in the session appears.

The chat tool helps to check the connections. It is usually needed when the Internet is very busy. It also provides a simple mechanism to log the comments along the session. It can be used instead of the telephone or as a supplement to voice communication when a collaborative session is held between people of different languages, a usual problem in Europe. The list of connected nodes is available in the interface.

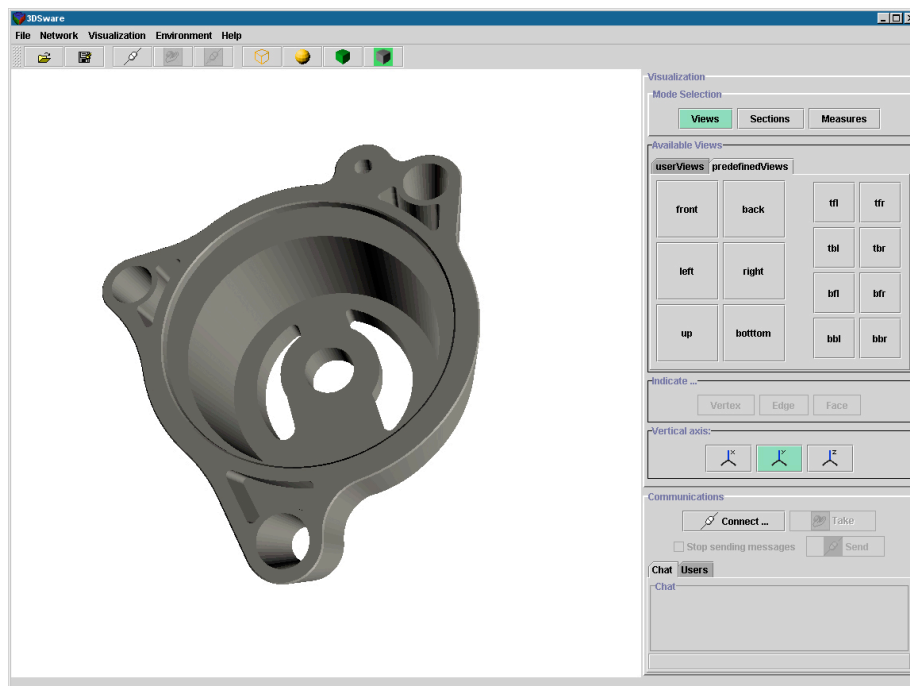


Figure 3 Cooperative Visualization Tool: 3Dshared.

The file transfer is required nearly in every session, so we have integrated a tool to make this task more comfortable.

The cooperative visualization window is used to: verify concurrently the design of the model and signal a particular feature to the other participants (sometimes to explain a doubt, other times to ask about a detail).

Now we will refer to some visualization features of the application. The controller can select any of the 14 predefined views, change the viewing conditions freely, or select one of the user views. The user can store or remove views from a list, which can be reused in several sessions. Switch buttons define which axe of the model is mapped vertically on the screen. Cutting planes that remove part of the model are used to see the inside and get a better understanding of the structure of the model. Verification may be done visually, but the caster usually needs to make some measurements to check angles, thickness, radiuses, distances, etc., while the discussion with the modeller is carried out in the CSCW session. All of these functionalities are sent from the controller to all clients.

EXPERIENCES IN COOPERATIVE VISUALIZATION

We have tested the application with a model made of 2.400 polygons in a Pentium III at 550Mhz., with an ASUS V3800 (32MB, TNT2 chipset) graphic card, and it achieves a frame rate of 49 frames per second. A model made of 50.000 polygons achieves 6 fps. As the designs that are required to verify the foundry models have tens of thousands of polygons, it is strongly recommended the use of a graphic accelerator card. Currently the program loads models from files that follow the "*Wavefront (.obj)*" format and stereo-lithography (.stl) format. It will easily accept any other format loadable by Java 3D. We have compared the prototype running locally against another viewer developed with OpenGL in C++ and both perform very alike.

Several experiments have been made to measure the speed in the transfer of the graphics file and the updating messages of the scene among several hosts. For this, two nodes have been connected with different configurations in the physical access to the network: both nodes connected by a local network, a node connected to a local network and another connected to the Internet by an ISP provider and a modem, and both nodes connected by modems and ISP providers.

The achieved results in the first configuration have been positive. The more significant experiences made in public were two: in December 2000 fourteen users connected in a cooperative session using the University of Deusto network, and, in September 2001 3DSHARED was exhibited on a open day sponsored by the Miramon Technology Park.

With regard to the second configuration, the spent time in the connection protocol among two hosts varies between five and ten seconds. The required time to transfer the updating messages is very low; usually the latency is negligible. However, the time required to transfer the model file in the case that the other host doesn't have it locally goes beyond the five minutes for medium size models (500K). Here resides the higher latency of the application. For resolving this we have decided to compress

the model file before its transmission. Using this technique compress ratios of 3:1 and 4:1 can be achieved in some models.

With both hosts connected to ISP providers and physical access to the network by modems the behaviour has been very similar. A live demonstration with four nodes was made during the ACM CSCW 2000 Conference [9]. Two of them located in Bilbao and San Sebastian connected to an ISP provider and the other two hosts connected to the Conference local network in Philadelphia. Similar results were achieved in a presentation to fifty Small Business owners of the Machinery sector realized in March 2001 in Elgoibar.

These experiences of the communication architecture have been made using relatively small polygonal models and computers with similar GPUs installed on them. It is evident that if larger models are used in dispersed computers with different features, some of them will be unable to reproduce the entire animation. They must either skip intermediate states of the animation or simplify the polygonal model.

CONCLUSIONS AND FUTURE WORK

The following points summarize our achievements:

- the application demonstrates that it is possible to create easily cooperative 3D visualization products directed to real and specific needs in the mechanical design industry, using Java 3D,
- the product can be used in all the systems that support Java 3D; we have tested: Linux and Microsoft Windows,
- the proposed networking architecture allows the cooperative visualization and collaborative work through the Internet
- it allows new users to connect and disconnect within the session in a very simple way, because they do not need to know which is the server node,
- the application uses some techniques that allow real cooperative visualization even with very low bandwidth,
- we have provided data from field tests regarding the performance of this type of applications which shows that these tools can be used even with very low bandwidth and a very low hardware cost in the physical access to the network.

However, the exposed experiences of the previous section concludes that the main difficulties are in:

- the time used in the transfer of the model file
- the maintenance of a similar frame rate in all hosts
- the possibility of saturating the interactive visualization capacity when assembly CAD models are visualized

Consequently, we have seen a research line that analyses the problem of the CAD models from its transmission by the network and interactive visualization point of view.

The parametric surface based models, specifically NURBS based models are precise and compact (they need less storage memory) and commonly used in CAD systems. In the Internet oriented applications, because of its implicit compression, they are more convenient than the polygonal meshes because the amount of information to be transferred among hosts is reduced.

With regard to the interactive visualization, its description allows to select the optimal level of detail in runtime. Thus, it is possible to manage the quality of the generated image in function of either the degree of interactivity to achieve or the hardware features of a particular host. This means that, in different hosts participating in a collaborative session over the same 3D model, its polygonal approximation can be made up a different number of polygons.

ACKNOWLEDGMENTS

We thank Luis Matey from CEIT, Amaia Bernarás from San Sebastián Technology Park and the people from the companies Mein and Ormola, especially Patricia Caballero, Javier Arcelus and Alejandro Sudupe, for their dedication, advice and eagerness concerning our work, which was priceless for the definition of the requirements that the product should provide. The Basque Government and the Diputación Foral de Guipuzcoa have sponsored this work.

REFERENCES

- [1] Borro, D.: « Sistema de Realidad Virtual Multiusuario para la Interacción cooperativa en 3D », available at http://scsx01.sc.ehu.es/ccww3d/docs/pfc_borro.doc
- [2] Gossweiler, R., Laferriere, R. J., Keller, M. L., Pausch, R.: « An Introductory Tutorial for Developing Multi-User Virtual Environments », Presence: Teleoperators and Virtual Environments vol. 3, nº. 4, pp. 255-264, 1994.
- [3] Brutzman, D.: « Graphics Internetworking: Bottlenecks and Breakthroughs », in Digital Illusions. Dodsworth, C. (ed.), pp. 61-97, Addison Wesley, 1997.
- [4] Saar, K.: « VIRTUS: A collaborative multi-user platform », Proceedings of the VRML'99 Symposium, pp. 141-152, Paderborn, Germany, 1999.
- [5] Macedonia, M. R., Zyda, M. J.: « A Taxonomy for Networked Virtual Environments », IEEE Multimedia vol. 4, nº. 1, pp. 48-56, Jan.-Mar., 1997.
- [6] Clark, D.: « Face-to-Face with Peer-to-Peer Networking ». Computer, vol. 34, nº. 1, pp. 18-21, January 2001.
- [7] Ban, B.: « JavaGroups – A Reliable Multicast Communication Toolkit for Java », Project Home Page: www.cs.cornell.edu/Info/Projects/JavaGroupsNew/index.html, 1999.
- [8] Clark, T.: « Ensemble – Distributed Communication System », Project Home Page: www.cs.cornell.edu/Info/Projects/ensemble, 2002.
- [9] Borro, D., Recio, I., García-Alonso, A., Sánchez, H., Matey, L.: « CSCW for foundry design using Java3D », Demonstration at ACM 2000 Conference on Computer Supported Cooperative Work, Philadelphia, 2000.
- [10] Pedrinaci, C.: « Integración y validación de un sistema de visualización cooperativa en Internet », available at http://scsx01.sc.ehu.es/ccww3d/docs/pfc_pedrinaci.doc.