**Intervision3D: Online 3D Visualisation and Conferencing**

Felix Heidrich¹, Peter Russell², Thomas Stachelhaus  
Lehrgebiet CAAD, RWTH Aachen University  
http://caad.arch.rwth-achen.de  
¹heidrich@caad.arch.rwth-aachen.de, ²peter_russell@mac.com

The use of Internet communication technologies in distributed teams has been carried out for well over 10 years. In this time, various methods to communicate and transfer information have been developed. A large amount of effort has been placed on enabling normal conversation to take place and it could be said, that with technologies like Skype, this is established. This enables planning partners to discuss, but we still need to convey what they are discussing. In short, the contents are still lacking. Technologies exist to allow users to share files or images, however this does not nearly reach the intensity or quality of discussions when partners are sitting together in front of a drawing or model. At best, screen sharing allows participants to see the same image but with low resolution and bad system response. The goal of the project is to allow distributed team members to discuss design issues with a common 3D model where participants can manipulate the model together in real time. The speed of the system is also buttressed by the simplicity of the application: as a Java applet, it is possible to start the Intervision3D system in any browser or as a separate applet on any system. Files can be imported and then rendered using the JOGL Engine (Java Bindings for Open GL). JOGL allows the full Open GL suite to be used in rendering the model including lighting and textures: even normal PCs can do this quite well. The first implementation of the system is within an existing internet-based Design Studio and the paper elucidates how the first uses of the system have (partially) helped to increase the exchange of design ideas over the Internet.

**Keywords:** Virtual design studio; shared models; CSCW.
Background

Internet-based Design Studios are not new. In the early 1990s, the first ideas about establishing asynchronous communication using information technologies existed (Wojtowicz et al., 1992). As the World Wide Web became established, architectural educators also flocked to the Internet with ideas about establishing design studios over the Internet. Some of these were one-shot affairs (Donath et al., 1999) while others, such as the Netzentwurf, have run continuously for 10 years (Forgber, Russell, 1999; Elger et al., 2006). The techniques for using the Internet as a communication tool have become so widespread, that making an Internet telephone call is widely referred to as “Skyping”.

Additionally, over the last 10 years, the portrayal of architecture using the Internet has been extensively explored. The uses of limited pixels, but extensible interactivity, have allowed architects to create new forms of architectural drawings (Russell et al., 1999). Furthermore, new media technologies such as Flash and VRML have enabled new possibilities in conveying architectural ideas.

However, in the design studio, the bulk of the communication involves not presentation, but investigation and interpretation. The design studio calls for a dialogue between students and tutors as well as among the students themselves. In the classical studio setting, this is easily carried out at the student’s desk or together around a table in a conference setting.

In an internet-based design studio, the rules are quite different. Firstly, establishing the presence of the studio partners has always been difficult. Simple methods can be used to establish reciprocal feedback as to the online presence and in the Netzentwurf, these have been quite successful. Indeed, the Netzentwurf platform is referred to a place and not merely a website (Russell, 2001). Secondly, it is often difficult to discuss student’s work: it is not clear where the work is and in what form it is presented. As well, it is difficult to be certain that both student and tutor are looking at the same thing. In part, this can be alleviated by the use of virtual computers such as VLC. This allows both (or many) users to look at the same computer screen. Thus, the student can use whichever browser or media software fits the presentation. The other users watch passively.

This type of communal viewing is well suited to viewing web pages, but can be cumbersome to set up and is also plagued by latency problems. As well, it does not address one of the most salient types of presentation: the architectural model. When students build physical models, they must be presented by photos placed on the web or by using a web cam to show live images to the other person. This has drawbacks in that only one person can control the view and the range of viewing positions is limited. When students build 3D digital models, an image or film must be made from the model in order to view it. These are static and cannot be changed rapidly, in response to criticism that might occur. Some types of 3D models can be co-viewed, but this is done with extra viewer software, which raises the old problem of platform compatibility (Windows vs. Mac. vs. Linux). In light of all these limitations, the authors undertook to create a web-based 3D viewing software that allows multiple users to communally view and comment on 3D models.

Netzentwurf

The Netzentwurf concept has been carried out since 1997 using a neutral website which serves as an information node for projects carried out between universities. As of the fall of 2006, over 1900 people from 15 universities have taken part in over 80 Projects on the platform. The platform is equipped with a “Who’s Online” feature which allows users to not only see who is also online, but to initiate informal communication through chat functionality. Further internet-based communication is possible through the embedded flash communication server. This allows participants to start individual audio or
videoconferences in preset “communication rooms”. The goal of the platform is to support the same kind of informal discussions that occur in traditional design studio settings (see figure 1).

An additional feature of the platform is a skills list where members can list their individual skills. These include language, computer programs as well as other field specific skills such as urban planning or civil engineering. Members can then search for other members with certain skills. This allows the entire membership to evolve into a kind of community where individuals are not just a face and a name, but have a profile. Other features of the platform include a booking system for tutorial slots and multi-language support (in addition to English, Dutch, German, Chinese and Japanese menu structures are fully supported).

The heart of the platform is the set of project management tools. Essentially, they allow tutors to establish projects and to offer them to the platform. Members can join these projects and provide links as to the web location of the project documentation. The projects, either from individual students or groups of students spanning different universities, can be located anywhere on the web. However, for many students with little or no HTML skills, the platform provides a simple content management system (CMS). The CMS allows the students to create web page structures and to arrange their uploaded images with text in a preset webpage structure.
The CMS also provides an automatic workbook webpage, which simply documents who uploaded what in which week, which is helpful for tracking the student’s progress. A separate Diary tool allows students to log when their webpage has been updated as well as to search for other student’s “new work”.

The limitation of the CMS is that students can only upload finished images of their models, either real or virtual. The tutors as well as the students sought a way to collectively get “inside” the buildings. This is the motivation for Intervision3D.

**Intervision3D**

There have been attempts to use VRML technologies to allow many users simultaneously to experience a 3D model. Beside the standard problems of computing power, these experiments were also beset by problems relating to avatars. Firstly, the avatars rarely matched the expectations of the users. Modern immersive 3D environments such as Second Life have improved on the situation, but not by much. We are still a long way from the Metaverse of Neal Stephenson’s Snow Crash (1992) Secondly, the attention of the various users is not clear. Much as when two people view a webpage separately, it is often the case that the various users are looking in different directions, or even wandering away from the part of the model being discussed.

The Intervision3D concept uses a moderated discussion paradigm. That is, two or more users share a view of the model with only one user in control of the view as a moderator. This ensures that all of the users have the same viewpoint. As well, it negates the need for avatars or other 3D presence indication and location.

Technically, the Intervision3D system consists of a server and Java Applets that are started locally on each user’s computer. The process starts when a user uploads a 3D Studio document (3DS) to the server. Currently, Intervision3D supports only 3DS files, but support for other formats is planned. Once the file is uploaded, the user can start a session with the 3DS file (or another that is residing on the server) and give it a unique name. The Java applet is started and the file is downloaded to the user’s computer as part of the Applet startup. The model is displayed in a window within the web browser. For the server, the first user is the moderator.

A second user can then go to the Intervision3D start page and choose to join a running session. The Java Applet is started in the second web browser and the file is downloaded to the second computer. The second user is then shown the view of the first user and is registered as a passive viewer.

When the moderator changes the view, this information is sent to the server and then relayed to the other session participants. The model information itself resides on each single computer and all that is being relayed is the location and viewing angle of the model. As a result, the amount of traffic after the initial download is minimal. This means that bandwidth is not a problem to shared viewing.

**3.1. Intervision3D Interface**

The Intervision3D Interface is contained entirely within the Java Applet. It consists of principally a 3D display and a sidebar with controls and information. The Sidebar itself is divided into three areas. The first area (figure 2a) is a list of saved views as well as buttons to manage or create new views. The second area is a list of the users taking part in the session. The moderator is indicated after the nickname, unless the user herself is the moderator, in which case a button to pass off the moderator function to another user appears (figure 2b). The third area is a color picker for the markup in the view mode. The user’s color is also indicated in the user list.

Because the users can all be on different platforms with different browsers and different displays, the aspect ratios of the displays are likely different. This is indicated to other users by creating shadow bars at the sides or top and bottom of the display. This indicates to the users which parts of the display are visible to all session participants. This is relayed from the server to each applet and then rendered locally (figure 2d).
The navigation is relatively intuitive. Users cannot change the focal length at the moment, but are presented with a normal human eye field of view. Forward and backward movements are done with the up and down arrow keys. However, the movement is not in the direction of the view, but in the direction of the mouse pointer, which allows users to easily zoom in on a part of the model. Holding the space bar allows panning; the control key allows users to rotate the viewing angle (look around). This allows users to set up views within the model itself.

When the user clicks on the model, the applet parses this and places a red ball on the nearest plane. This is then used as a rotation point for the whole model (figure 2e). This way the users can “swing around” a certain part of the model, without having to rezoom or pan back to the part of the model they were interested in. When a user clicks on the view where no model is present (e.g. in the sky), the viewer rotates the user’s view (see figure 3).

As the users explore and discuss the model, it is possible to save a certain view. This disables the navigation controls in the model and allows the users to markup the view instead. Thus, the users can draw on the view and annotate or highlight certain aspects of the model. With the escape key, the moderator can return to the navigation controls, however the view and the annotations remain saved on the server. This way, the users can return to certain views or switch between them, according to the needs of the discussion (see figure 4).

The rendering of the views is done using the JOGL Engine (Java Bindings for Open GL). JOGL allows the full Open GL suite to be used in rendering the model including lighting and textures: even normal PCs can do this quite well as most graphic
cards support OpenGL. This means that the actual computing is done to great part on a dedicated graphic card and not on the computer itself or the server. As such, latency in updating the views to all members of a session is limited for the most part to the network speed. What is important to note is the real-time aspect of the shared model – this is rarely found in other modeling software and until now, always application or system dependent.

**Summary**

Initial tests with users have generally brought positive feedback for the system. Apart from some general system-related problems, there are some issues, which need to be addressed. The first is that because all users need to have a copy of the model, bandwidth restrictions for some users will mean that startup times take minutes as opposed to seconds. However, in a university setting, this should not normally be a problem. Another aspect of the system design is that sooner or later, there will have to be a model management interface before the number of uploaded models increases to overwhelm the pull down menu interface. Standard versioning concepts will also need to be implemented to ensure that all partners know which version of the project they are discussing.
Currently, the users must use a separate communication tool such as Skype to speak with one another. This has drawbacks in that other Skype users often then interrupt the users when they need to concentrate on the Intervision3D users. The implementation of the Netzentwurf Flash Communication Server audio channel into the interface is on the “To do” list.

The session management is, at the moment, entirely open. It will probably be desirable to allow open or closed sessions, so that the number of participants is limited and limited to those who should take part. As well, at the moment the first moderator has no way to recover the moderator function once it is passed off to another user. If they refuse (or forget) to pass the function back, all other users are powerless. The creation of a session manager would allow someone to have control of the moderator function and to retrieve it if necessary.

The above-mentioned aspects are being implemented into the project. Other features such as export of views and markups and being able to save discussion paths are also desirable. Camera paths in the model would also allow students to set up a presentation before the online meeting.

It must be noted that like the Netzentwurf, the Intervision3D system is a self-financed project. This means that limited resources are available to improve the system. On the other hand, the previous work on the Netzentwurf platform as well as Intervision3D have shown that limited resources means keeping things simple and by doing this, it usually means that they work.

References


