Multimodal Interaction in a Collaborative Virtual Brainstorming Environment

Fabian Di Fiore, Peter Vandoren, and Frank Van Reeth

Expertise Center for Digital Media
Limburgs Universitair Centrum
Universitaire Campus
B-3590 Diepenbeek, Belgium
{fabian.difiore, peter.vandoren, frank.vanreeth}@luc.ac.be
http://www.edm.luc.ac.be

Abstract. In this paper we present our work in setting up a collaborative virtual environment (CVE) framework which is built to support collaborative creative meetings for geographically dispersed participants. Similar to real life, we rely on the use of quick drawings or sketches as a means of communication to convey new ideas, thoughts or other metadata to other individuals.
Furthermore, we concentrate on facilitating the (collaborative) interaction process through the use of four modalities. The first modality is direct manipulation, which is suitable for direct interaction with the networked environment. Secondly, we look at interaction through gesturing symbols in order not to distract the user's attention from the meeting. As a third modality we consider interaction through menu and widget manipulation. A fourth modality is established by a camera interface.
We expect that the combination of the intuitive interface and the real-time visualisation of the virtual environment leads to a better understanding and realisation of one's ideas in an early phase of the cooperation.

Keywords: collaborative virtual environment, human-computer interaction, multimodal interaction, sketching

1 Introduction and Motivation

People who cooperate in teams often need to meet together to do some brainstorming, or to form, convey, readjust or hit upon (new) ideas. During brainstorming sessions, it is a common practice for participants to use quick drawings or sketches as these convey more than only words can say. However, when the participants are located in geographically dispersed locations, these kinds of brainstorming sessions are impracticable.

'Collaborative Virtual Environments' (CVEs), in general, are applications that provide distributed virtual reality technology to support cooperative work. They consist of virtual spaces that enable participants to collaborate and share

objects as if the participants were present in the same place. Currently, CVEs are used for many purposes, some of which include collaborative design [1], manipulation [2], education, simulation, and training. However, on-line games still are the most common form of on-line virtual environments in use today. Games such as Sony's 'EverQuest' [3] and 'World of Warcraft' [4] are designed to be played by thousands of users worldwide every day.

In this paper we introduce our work in setting up a collaborative virtual environment (CVE) framework which is built to support collaborative creative meetings. Regarding collaborative working, roughly two categories can be identified. In the first, users are located in the same room and, for example, use displays projected on walls or tables as a spatially continuous extension of their own workspace [5]. Our work, however, extends the concept of 'Designer's Outpost' [6] in which participants are *remotely present* while sharing one virtual workbench.

The strength of the proposed environment is in the fact that we, just as in real life, use quick drawings or sketches as a means of communication to convey new ideas, thoughts or other meta-data to other individuals. We expect that by using our system the occurrence of misunderstandings can be minimised or cleared up promptly in an early phase of the cooperation.

This text is organised as follows. In section 2 we elaborate on the collaborative environment that we envisage. Section 3 focusses on facilitating the (collaborative) interaction process through different modalities. Currently, we concentrate on four modalities in particular, which are direct manipulation, interaction through gesturing symbols, menu and widget manipulation, and a camera interface. We end with our conclusions and topics for future research (section 4).

Related work on highlighted aspects will be mentioned in the appropriate sections.

2 Collaborative Virtual Environment Framework

The collaborative environment we envisage consists of a virtual workbench that can be shared by a group of users (who are remotely present) in order to support collaborative creative meetings. In the following subsections we discuss the underlying virtual environment framework and the collaborative setup.

2.1 Virtual Environment Framework

Virtual Environments (VEs) are computer generated, two-dimensional or three-dimensional environments that create the effect of an interactive world in which the user is immersed. In previous research, our research group has developed a code framework for interacting in virtual environments [7,8,9]. In this work, we rely on their framework, of which we'll now give a brief overview.

The framework in its current state can be regarded as a black box that can be used to create VEs that require all sorts of (multimodal) interaction possibilities. Currently, it supports the use of several 3D input devices (e.g. a 3D

mouse [10], a MicroScribe [11], 3D trackers), speech input and haptic feedback (e.g. PHANToM [12]).

As different modalities, such as haptic interface, direct manipulation, interaction through widgets, or speech input, present their information in a specific manner, an interface is developed for each of these interaction techniques. These interfaces should be considered in a sense of a high-level structured syntax of which the designer of the environment can use the supported functionalities.

The data passed into the framework is similar for all created interfaces and is represented by *interaction events*. All events that are generated by the interaction techniques are sent to a central dispatching unit, called a *task conductor*. From there, the interaction events are redirected to the appropriate (application specific) event handling code.

Similar to other interaction techniques, the user interface will also generate and send interaction events to the task conductor in order to perform the necessary tasks.

2.2 Collaborative Setup

Our server-client based CVE system consists of a server application, and for each participating site a client application (GUI). The functions of the server include management of the parties (joining, quitting), sharing of application data (adding, deleting, ...), and floor control to assure mutual exclusion. The main role of each client is to serve as the collaborative interface with the user and to act as the visible part of the collaborative system.

The collaborative environment that we envisage consists of a virtual workbench that can be shared by the users. For each participant, this virtual workbench is projected onto his/her real desk. Equipped with a set of interaction devices and some basic functionalities, this virtual workbench allows participants to brainstorm with other users in an intuitive manner. Figure 1 shows one possible set-up. In this case the projected working area can be enlarged or reduced depending on the available space and need, just by tuning the projector. Other set-ups are supported as well including projecting from underneath the desk or simply using a conventional display.

In order to obtain a one-to-one mapping between the projected workbench and the user's real working area, we provided simple calibration in which the user selects the corners of the projected workbench. As a result, when using for example a 3D pointing device, besides the obvious one-to-one mapping between the real tip of the device and it's projected visual representation, we also can detect whether the tip touches the workbench or not. This will be elucidated in section 3.1.

At this moment, two different network communication protocols are being used simultaneously: TCP/IP and RTP (real-time protocol). The first is used as a reliable means of communication and deals with data that in any case needs to be delivered, whereas the latter is employed to take care of time-critical data.

In the next section we concentrate on the different modalities that are being used.

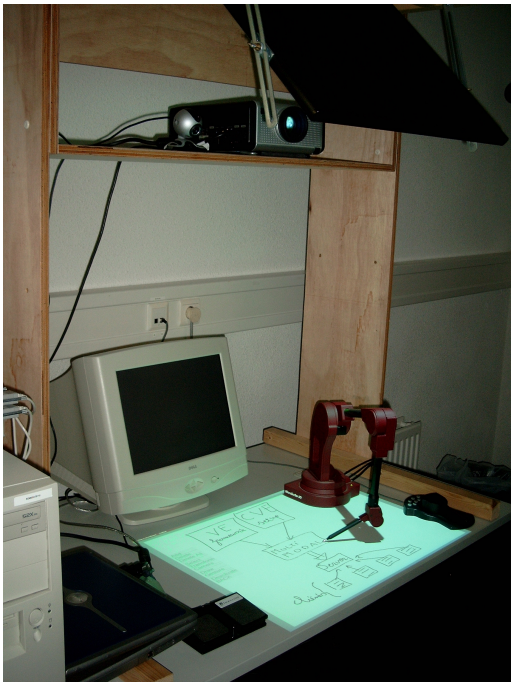


Fig. 1. A possible set-up for displaying the shared virtual workbench.

3 Multimodal Interaction

In a multimodal application, input and output are provided by using different devices. Due to their specific characteristics, each modality has its own needs with regard to the used input device.

Currently, we concentrate on four modalities in particular to facilitate the collaborative interaction process. The first modality is direct manipulation, which is suitable for direct interaction with the 3D virtual environment. Secondly, we look at interaction through gesturing symbols in order to facilitate a sketching process. As a third modality we consider interaction through menu and widget manipulation. A fourth modality is established by a camera interface as a means to incorporate real-life objects into the virtual workbench.

The functionality of the first three modalities is provided using a MicroScribe [11]. The reasons for using a MicroScribe are threefold. Firstly, we need a *tracking device* in order to be able to intuitively put down sketchy remarks (section 3.1). The MicroScribe is an accurate, affordable 3D digitising system (usually employed to digitise 3D clay models). However, its high accuracy (0.009" or 0.23 mm) and large workspace size (50" or 1.27 m), provide the user also with a natural interaction method for sketching in 2D/3D [13]. The device is depicted in the top-right corner in figure 2. Secondly, the same argument stands up for gesturing understandable symbols (section 3.2). Thirdly, an easy-to-use and comfortable

pointer device is required for several 'conventional' pointing operations such as menu interaction (see section 3.3) and object selection. Since the user's dominant hand is already holding the MicroScribe, it is not convenient to switch to another input device (e.g. mouse) for menu interaction. Therefore, we chose to use the MicroScribe to interact as well with 3D menus (figure 5(b)).

Since the MicroScribe also comes with two foot pedals, which are very useful for e.g. mode selection, several different interactions (sketching, selecting menu items, selecting objects, ...) can be performed successively in an arbitrary order using the same device. This approach considerably reduces the number of times the user has to release the MicroScribe.

Other conventional (tracking) devices could be used as well [14,15,7], however, from our experiences [13] we discovered they either only work fine in laboratory environments, or are unnatural to use. For example, a conventional (pressure-sensitive) stylus is easy-to-use but depends on a — often small — fixed size working area. Kobayashi et al. [16] and more recently, Oka et al. [17] introduced a fast and robust method for tracking a user's hands and multiple fingertips. Their method is capable of tracking multiple fingertips in a reliable manner even in a complex background under dynamically changing lighting conditions without any markers. Although it feels very natural to the user, a major drawback of this method is that tracking fingers still lacks high accuracy.

Note that the PHANToM [12] is a worthy alternative to the MicroScribe. However, a formal user test carried out by Raymaekers and Coninx pointed out that in a virtual environment the "point-and-click" metaphor should be used rather than a pushing metaphor [8]. As a result, employing a force feedback device only as a tracking or pointing device while ignoring force feedback is quite overkill.
In the following subsections we elaborate on the different modalities.

3.1 Direct Manipulation: Collaborative Sketching

In our CVE we rely on the use of sketches as a means of communication to convey certain ideas to other individuals.

Therefore, we must keep an important constraint in mind, which is the ease of use of the sketching tool. The simplicity of the classic pen, which is also available as an input device for the computer, is an important feature that needs to be taken into account in the implementation of our sketching tool. In other words, a sketch-drawing tool needs to work in a natural and intuitive way, which implies that the tool has to work in real-time and that the movement of the pen has to be closely tied to the resulting curve.

Creating Sketches. In our system, the creation of a stroke is done interactively by sampling the MicroScribe along the trail of the stroke. This only happens when the tip of the MicroScribe touches the workbench (section 2.2). In order to allow for real-time high-level manipulations on the strokes, the individual pixels that make up the stroke are not used. Instead, a high-level internal

representation, using cubic Bézier curves, is created. We use the solution of [18]. While sampling the MicroScribe we simultaneously perform an iterative curve fitting technique based on least-squared-error estimation. Existing curve drawing solutions mostly take recourse to a 'batch' curve fitting solution, in which the fitting is done after the stroke drawing is finished, whereas our fitting is done on-the-fly while the curve is being drawn.

The curves themselves are drawn as solid lines. Figure 2 illustrates the use of a MicroScribe as a tracking device in order to intuitively put down sketchy remarks. Note also there's a one-to-one mapping between the projected workbench and the tip of the MicroScribe (i.e. real working area), through an initial calibration.

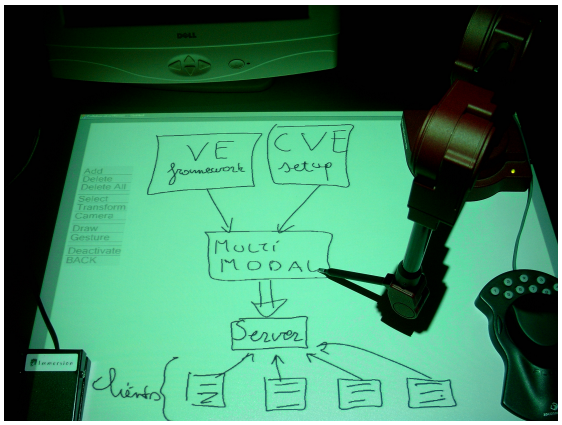


Fig. 2. Employing the MicroScribe as a tracking device in order to intuitively put down sketchy remarks.

We also added support for grouping and performing transformations on (parts of) the sketches. Existing applications transform drawings on a per-pixel basis which results in artifacts because the transformed parts are cut out and then pasted at a new position. In our case the transformation tools (translate, rotate, scale, ...) only affect the underlying geometric description of the selected (parts of the) sketches.

Notice that the user does not have to worry about picking, clicking and dragging control points associated with the underlying curve primitives. That way we preserve the same freedom of sketching a user has when using the "pencil-and-paper" approach.

Furthermore, we provided navigation support by means of a SpaceMouse [10]. This is a 3D optical mouse with 11 programmable buttons which allows us to simultaneously pan, zoom or rotate the virtual camera using only one hand. The SpaceMouse is visible in the lower right corner of figure 2.

Collaborative Sketching. In order to modify existing sketches, they temporarily need to be assigned to the requesting user. The user concerned only has to 'touch' a particular sketch after which an explicit locking request is sent to the server. The server for its part takes care of the request and notifies all other parties in case of acceptance. The requesting client, however, gets informed in any case. All communication involved happens transparently to the user.

By using instantaneous visual feedback (e.g. using different colours or flashing points), users always are aware of which sketches currently (i) are being modified by others, (ii) are attributed to the user himself, or (iii) are unlocked.

Sketches consist of cubic Bézier curves. Consequently, whenever sketches are being drawn or modified, only the affected control points need to be sent to the other participants. In case of transforming sketches or requests for locking/unlocking/..., only specific data is sent over the network.

For these operations, we make use of TCP/IP as a communication protocol. As a result, we are assured that all data arrives in the right order, *and* no data gets lost.

3.2 Direct Manipulation through Gesturing

When making a sketch, it is sometimes desirable to manipulate the result. Different techniques to manipulate the sketch can be encountered in the literature. For instance, in Teddy [19], a program which allows the creation of a 3D object out of 2D sketches, a sketch is modified by drawing extra lines onto the sketch. This manipulation then deforms the sketch.

In our case, we have investigated the interface that is needed to perform typical re-occurring operations including transforming sketches (i.e. translating, rotating and scaling), selecting sketches, deleting them, ... without having to turn to menus (and other widgets) which take up a lot of time and distract the user's attention.

We thus need an intuitive user interface for performing these operations. Since the user is already using a sketching interface by means of a MicroScribe, a *gesture interface* comes to mind.

By not using a WIMP interface, but a sketching interface, the user can perform the different operations without having to resort to a button or another interface element that distracts the attention from the sketching process.

Gesture interfaces are used in different applications, such as whiteboard tools [20]. Landay and Myers propose an interactive tool which allows designers to sketch an interface, and transforms it into a complete, operational interface [21]. Traditional gesture interfaces have as disadvantage that they do not offer an accurate means to provide input, while the interface that is generated by the tool of [21] has lost the look and feel of a sketch interface.

In our approach, the purpose of gestures can be regarded as *gesturing the functionality* and not gesturing the widgets themselves. That is, gestures are not turned into widgets since otherwise, the user has to perform twice as much interactions as needed: (i) gesturing the widget, and (ii) interacting with the generated widget. Instead, upon recognising a gesture, the system switches its

mode to the corresponding operation. Furthermore, by changing the pointer, the user gets instantaneous feedback about the current mode.

Creating Gestures. The gesturing interface is built upon our intuitive sketching tool (section 3.1). Gestures are created in the same manner as "normal" gestures: the user just draws a gesture, e.g. like an arrow, using the MicroScribe. The difference with a normal gesture is interpretation of the gesture. For each gesture a particular action which performs the manipulation is defined. For instance, drawing an arrow, pointing to the right (figure 3(a)) indicates that the selected object(s) can be moved to another position. To be more precise, as soon as a horizontal arrow is drawn (and recognised), our system switches to horizontal translation mode (as defined by the direction of the arrow). From now on, the user can precisely position the object by moving the pen of the MicroScribe: the further the user moves the pen away, the further the object is moved.

Several gestures for translating, rotating, scaling and deleting are shown in figure 3.

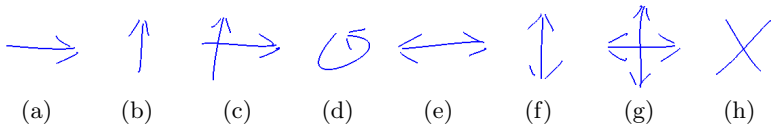


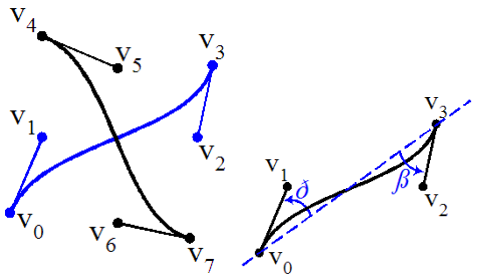
Fig. 3. Several gestures. a) Horizontal translation. b) Depth translation. c) Horizontal and depth translation. d) Rotation. e) Horizontal scale. f) Depth scale. g) Horizontal and depth scale. h) Delete selected object(s).

Recognition of Gestures. Because our gestures are represented by curves, it is easier to exploit the associated curve data instead of using for example a neural network that anyhow has to be trained [22].

Our idea is to induce the intended gesture from (i) the geometrical position of the control points, (ii) the order of appearance of the control points, and (iii) the number of sketches which make up the gesture. That way, we also do not impose any constraints on the user, such as the drawing speed or the size of the gesture.

In order to clarify our approach, we amplify on recognising the 'delete' gesture (figure 3(h)).

Based on the number of strokes drawn by the user, our system will perform some test in order to recognise this gesture. In this case, one possible gesture is the one indicating the 'delete' functionality since it typically consists of two strokes, as shown in figure 4(a). If one of the tests fails, the system proceeds to test the next possible gesture (in this example, also consisting out of two strokes).



(a) (b)

Fig. 4. Recognition process of a 'delete' gesture. a) Gesture made by the user shown with underlying control points. b) Close-up of the first line (indicated in blue in figure (a)).

At first, we test if the two strokes are (nearly) straight lines. We'll show this by means of the first line (figure 4(b)). Since our strokes are represented by cubic Béziers, straight lines occur when all control points are co-linear. That is, ideally, angles ϑ and β should be 0. As users almost never draw real straight lines, angles up to 35 degrees are tolerated. As a result, following equation should be satisfied (the same idea counts for the second line):

$$\vartheta \simeq \arccos\left(\frac{\overrightarrow{v_0v_3}}{\|\overrightarrow{v_0v_3}\|} \cdot \frac{\overrightarrow{v_0v_1}}{\|\overrightarrow{v_0v_1}\|}\right) \leq 35^\circ$$

Now that we know that the gesture consists of straight lines, we have to check whether the slope of both lines is correct: the slope of $\overrightarrow{v_0v_3}$ should approximately equal 1 whereas the slope of $\overrightarrow{v_4v_7}$ should approximate -1. This is examined by following equations:

$$\frac{v_{3,z} - v_{0,z}}{v_{3,x} - v_{0,x}} \approx 1 \text{ and } \frac{v_{7,z} - v_{4,z}}{v_{7,x} - v_{4,x}} \approx -1$$

As a final test, we need to be sure that the two lines actually cross each other. This is a straightforward operation and comes down to whether or not finding an intersection point. Both t_a and t_b, see following equations, should be in the range 0..1.

$$t_a = \frac{((v_{7,x} - v_{4,x}) * (v_{0,y} - v_{4,y})) - ((v_{7,y} - v_{4,y}) * (v_{0,x} - v_{4,x}))}{((v_{7,y} - v_{4,y}) * (v_{3,x} - v_{0,x})) - ((v_{7,x} - v_{4,x}) * (v_{3,y} - v_{0,y}))}$$

$$t_b = \frac{((v_{3,x} - v_{0,x}) * (v_{0,y} - v_{4,y})) - ((v_{3,y} - v_{0,y}) * (v_{0,x} - v_{4,x}))}{((v_{7,y} - v_{4,y}) * (v_{3,x} - v_{0,x})) - ((v_{7,x} - v_{4,x}) * (v_{3,y} - v_{0,y}))}$$

The recognition process of all other gestures involves similar operations.

We end this section by stating that gesturing the functionality is both easy to create and use and does not distract the user's attention from the brainstorming process. This is particularly due to (i) the use of gestures which are intuitive and easy to remember, (ii) the analytical recognition process which does not involve any training period, and (iii) the user never has to release the MicroScribe for performing typical re-occurring manipulations.

Interpretation of Gestures. A problem that arises in this kind of interfaces is how to specify the mode that the user is working with. The user can either be drawing the sketch, gesturing or using the gesture.

Likewise, the transition between the gesturing mode and the usage of the gesture should be defined. One could stop the gesturing mode when a gesture is recognised but this could introduce a new problem when drawing a gesture which is a subset or a superset of another one. For example in figure 3 one can see clearly that the gesture for making horizontal translations is a part of the gesture for scaling and hence the system has no knowledge whether the gesture is completed.

As a solution, we use the MicroScribe's foot pedals as a means for switching between modes. Moreover, by changing the pointer visual feedback about the current mode is given to the user as well.

3.3 Menu and Widget Interaction

Menu and widget interaction form an essential part of virtual environments. Different approaches to incorporate menus in a virtual environment can be identified in the literature. In JDCAD [23] a "Daisy menu" is used in which a sphere shaped menu is applied to manipulate 3D objects. Others suggested pie or radial menus [24,16,25] in which menu items are represented by slices of the pie. A drawback of the latter is that they still are two-dimensional. Some research into 3D user interfaces has been conducted by Anderson et al. [26] who developed a complete toolkit to create user interfaces for VEs (e.g. the FLIGHT project). A 3D user interface can, for instance, be useful to prevent the need for alternating between using a mouse and using a 3D device when working with a 3D application.

As our main goal is to fully integrate the widget set into the virtual environment, we chose the hybrid approach as presented by [8]. The resulting user interface contains properties based on 2D as well as 3D environments and applications. The UI elements can be arbitrarily positioned and are initially semi-transparent, thus not occluding parts of the environment. However, when the pointer approaches the UI element, it fades in to become opaque. This feature is illustrated in figure 5(a) by means of a virtual pointer.

In our system we also provided a MicroScribe interface for the menu interaction in order to let the user carry on with the same input device. This is shown in figure 5(b) in which a menu item is selected by 'touching' its projected version using the MicroScribe's tip.

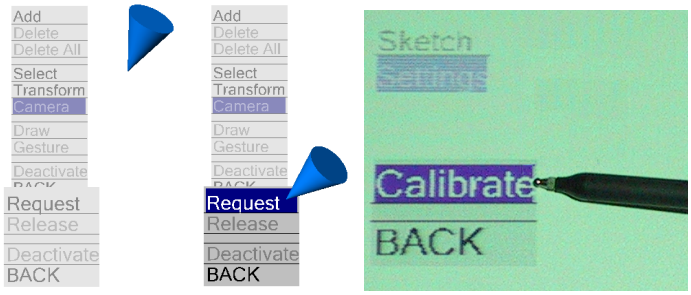


Fig. 5. a) This illustrates the use of the menus used in our environment. UI elements are initially semitransparent, but, when the pointer approaches them, they fade in to become opaque. b) Employing the MicroScribe as a pointing device in order to intuitively select a menu item.

3.4 Camera Interaction

Video encoding and transmission is traditionally primarily used for video conferencing, video-on-demand, and/or broadcasting applications. Research concerning the use of video streams to represent a users appearance in a virtual environment has been reported upon in [27,28,29,30,31,32]. While these applications themselves are a major subject of research, the combination of video communication with (collaborative) virtual environment technology is still relatively unexplored.

We believe that the incorporation of real-time video streams can create a surplus value to collaborative work applications. In this work, we employ the use of a camera as an extra means to incorporate real-life objects into the virtual workbench. For example, similar to situations in real life, while one hand is used for drawing, the other (free hand) can serve for showing objects or pointing. This is depicted in figure 6.

When aiming the camera towards the projected workbench itself, the camera obviously records the projected workbench and consequently retransmits it, causing visual artifacts. This problem could be solved by explicitly extracting only the hand or fingers from the recorded images using computer vision techniques. However, at this moment, we avoid this issue by working in the same way as weathermen do: the user moves his hand in another region with a neutral background while at the same time watching the projected workbench; the camera is aimed towards this neutral background while the recorded images immediately get blended into the virtual environment.

Given the real-time nature of these data streams, an obvious choice for the underlying protocol is the Real-Time Transmission Protocol or RTP, as it handles issues such as synchronisation and packet ordering internally. Our implementation currently employs the JRTP [33] library.

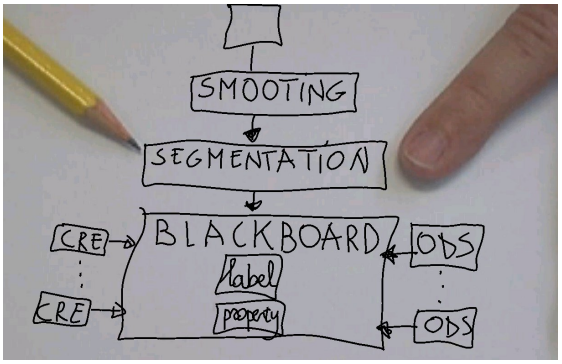


Fig. 6. This snapshot illustrates the use of a camera as a means to incorporate real objects into the virtual workbench. In this particular example, a pencil and a finger are used to emphasise one part of the diagram.

4 Conclusions and Future Research

We presented our work in setting up a collaborative virtual environment (CVE) framework which is built to support collaborative creative meetings. We concentrated on the use of quick drawings or sketches as a means of communication to convey new ideas, thoughts or other meta-data to other individuals.

Four different modalities enable the user's interaction with the virtual environment. Direct manipulation allows a very straightforward interaction with the virtual objects. Next, gesturing permits additional interaction without having to switch to another interface. For more infrequent tasks, widgets and menus are integrated into the environment and manipulated using the same devices. Finally, a camera interface brings the real world of one user into another's virtual world.

Several cooperative working sessions were held among different location. From the user's point of view, our collaborative environment is easy to use, without the need for training sessions. Particularly, the instantaneous visual feedback from both the sketching tool and the camera was appreciated.

In the near future, we would like to incorporate speech into our CVE — this is currently under development; at the moment we use a standard non-integrated tool [34]. Furthermore, we plan to (further) investigate the use of two-handed input, and to incorporate tangible mixed reality interaction.

Acknowledgements

We gratefully express our gratitude to the European Fund for Regional Development and the Flemish Government which are kindly funding part of the research reported in this work.

Our appreciations also go out to Tom Jehaes and Peter Quax for dealing with some implementation issues, and Johan Claes for the valuable reviews.

Furthermore, we would like to thank Jori Liesenborgs for freely putting available to us his JRTP library, and ANDROME NV for helping us to code and decode the video streams [35].

References

1. Luo, Y., Sánchez, D., Bennasar, A., Fornés, J., Serra, J.C., Huéscar, J.M.: Visualization for cooperative architecture design systems. In: Proceedings of Sixth International Conference on Information Visualisation (IV2002), IEEE (2002) 497–501
2. Rekimoto, J.: Pick-and-drop: A direct manipulation technique for multiple computer environments. In: Proceedings of the 10th annual ACM symposium on User interface software and technology, ACM Press (1997) 31–39
3. Sony: Sony Online Entertainment: everquest. World Wide Web, `http://www.everquest.com/` (2004)
4. Blizzard: Blizzard Entertainment: World of Warcraft. World Wide Web, `http://www.blizzard.com/wow/` (2004)
5. Rekimoto, J., Saitoh, M.: Augmented surfaces: A spatially continuous workspace for hybrid computing environments. In: In Proceedings of Computer-Human Interaction (CHI1999). (1999) 378–385
6. Everitt, K.M., Klemmer, S.R., Lee, R., Landay, J.A.: Two worlds apart: Bridging the gap between physical and virtual media for distributed design collaboration. In: CHI Letters, Human Factors in Computing Systems (CHI2003). Volume 5. (2003) 553–560
7. Raymaekers, C., De Weyer, T., Coninx, K., Van Reeth, F., Flerackers, E.: ICOME: an Immersive Collaborative 3D Object Modelling Environment. In: Proceedings of Virtual Reality (VR1999). Volume 4. (1999) 129–138
8. Raymaekers, C., Coninx, K.: Menu interactions in a desktop haptic environment. In: Proceedings of Eurohaptics. (2001) 49–53
9. De Boeck, J., Raymaekers, C., Cuppens, E., De Weyer, T., Coninx, K.: Task-based abstraction of haptic and multisensory applications. In: Proceedings of EuroHaptics (to be published). (2004)
10. 3Dconnexion. World Wide Web, `http://www.3dconnexion.com/` (2004)
11. Immersion. World Wide Web, `http://www.immersion.com/digitizer/` (2004)
12. SensAble. World Wide Web, `http://www.sensable.com/` (2004)
13. De Weyer, T., Coninx, K., Van Reeth, F.: Intuitive modeling and integration of imaginative 3D-scenes in the theatre. In Richir, S., Richard, P., Taravel, B., eds.: Proceedings of Virtual Reality International Conference (VRIC2001). (2001) 167–173
14. Sachs, E., Robert, A., Stoops, D.: 3-Draw: a tool for designing 3D shapes. In: IEEE Computer Graphics and Applications. Volume 11. (1991) 18–26
15. Zachmann: Distortion correction of magnetic fields for position tracking. In: Proceedings of Computer Graphics International (CGI1997). (1997) 213–220
16. Kobayashi, M., Koike, H.: Enhanced desk, integrating paper documents and digital documents. In: In Proceedings of Asian Pacific Computer Human Interaction. (1998) 57–62
17. Oka, K., Sato, Y., Koike, H.: Real-time fingertip tracking and gesture recognition. In: IEEE Computer Graphics and Applications. Volume 22. (2002) 64–71

18. Vansichem, G., Wauters, E., Van Reeth, F.: Real-time modeled drawing and manipulation of stylized cartoon characters. In: Proceedings of the IASTED International Conference on Computer Graphics and Imaging, Honolulu, HI, USA, IASTED (2001) 44–49
19. Igarashi, T., Matsuoka, S., Tanaka, H.: Teddy: A sketching interface for 3D freeform design. In: Proceedings of SIGGRAPH, ACM (1999) 409–416
20. Moran, T.P., van Melle, W.: Tivilo: Integrating structured domain objects into a free-form whiteboard environment. In: Proceedings of Computer-Human Interaction (CHI2000). Volume CHI 2000 Extended Abstracts., The Hague, NL, ACM (2000) 20–21
21. Landay, J.A., Myers, B.A.: Interactive sketching for the early stages of user interface design. In: Proceedings of Computer-Human Interaction (CHI1995), Vancouver, CA, ACM (1995) 43–50
22. Bimber, O.: Rudiments for a 3D freehand sketch based human-computer interface for immersive virtual environments. In: Proceedings of Virtual Reality Software and Technology (VRST1999), ACM (1999) 182–183
23. Green, M., Halliday, S.: A geometric modelling and animation system for virtual reality. Communications of the ACM **39** (1996) 46–53
24. Deering, M.: The holosketch VR sketching system. Communications of the ACM **39** (1996) 54–61
25. Kurtenbach, G., Fitzmaurice, G., Owens, R., Bauel, T.: The hotbox: Efficient access to a large number of menuitems. In: Proceedings of Computer-Human Interaction (CHI2000). (2000) 231–237
26. Anderson, T., Breckenridge, A., Davidson, G.: FBG: A graphical and haptic user interface for creating graphical, haptic user interfaces. In: Proceedings of the Fourth PHANToM Users Group Workshop (PUG99). (1999)
27. Insley, J., Sandin, D., DeFanti, T.: Using video to create avatars in virtual reality. In: Proceedings of SIGGRAPH, Los Angeles CA (1997) 128–128
28. Yura, S., Usaka, T., Sakamura, K.: Video avatar: Embedded video for collaborative virtual environment. In: Proceedings of the IEEE International Conference on Multimedia Computing and Systems. Volume 2. (1999) 433–438
29. Ogi, T., Yamada, T., Tamagawa, K., Hirose, M.: Video avatar communication in a networked virtual environment. In: Proceedings of INET2000. (2000)
30. Rajan, V., Subramanian, S., Johnson, A., Keenan, D., Sandin, D., DeFanti, T.: A realistic video avatar system for networked virtual environments. In: Proceedings of IPT 2002, Orlando, FL (2002)
31. Quax, P., Jehaes, T., Jorissen, P., Lamotte, W.: A multi-user framework supporting video-based avatars. In: Proceedings of the 2nd workshop on Network and system support for games, ACM Press (2003) 137–147
32. Quax, P., Jehaes, T., Flerackers, C., Lamotte, W.: Scalable transmission of avatar video streams in virtual environments. In: Proceedings of the IEEE International Conference on Multimedia & Expo (ICME2004) (to be published). (2004)
33. Liesenborgs, J., Lamotte, W., Van Reeth, F.: Voice over IP with JVOIPLIB and JRTPLIB. In 26th Annual IEEE Conference on Local Computer Networks (2001)
34. Skype. World Wide Web, `http://www.skype.com/` (2004)
35. ANDROME NV. World Wide Web, `http://www.androme.com/` (2004)