From Proceedings IEEE International Conference on Information Visualisation 2000, London, UK

DILEWA: The DIstributed Learning Environment Without Avatars

Michal Máša, Jirí Žára
Department of Computer Science and Engineering, Czech Technical University in Prague
{xmasam, zara}@fel.cvut.cz

Abstract

Nowadays multi-user VR systems are mostly aimed to social interaction, whereas exploitation of shared virtual environment for educational purposes has been mainly left aside. We introduce notion of tutor-based system and state main differences between tutor-based multi-user system and social interaction aimed one. In the proposed approach, one of the users is the tutor while the others represent auditors. This paper focuses on possible means for expressing tutor's role including special auditor's operating modes supported by mechanism of net-routes. Theoretical considerations have been tested on DILEWA experimental system.

1. Introduction

Most existing multi-user VR systems follow a fundamental pattern of shared virtual world(s) occupied by users perceiving each other with the help of avatars [4,5]. Such a model sufficiently meets needs for social interaction but this is definitely not the only way of exploiting virtual environments (VE). In this paper we focus on cases where one of the users plays a special role among the others – she/he acts as a *tutor*. The rest of users then represent her/his *auditors*. For such an arrangement, we introduce a notion of a *tutor-based* multi-user system. Such a system is mainly suitable for educational purpose, but not limited to it – multi-user cooperation in VE can make the most of it as well [6].

In section 2, we state main differences between tutor-based multi-user system and social interaction aimed one. Section 3 discusses resources for expressing tutor's role, especially in VRML, introducing special auditors' modes and mechanism of *net-routes.* Results are presented in Section 4, which deals with implementation issues of our experimental system called DILEWA.

2. Virtual reality as a learning environment

The *tutor-based* system introduces new requirements for how users should perceive each other. While every user is less or more equal in a multi-user system for social interaction, the tutor plays the key role and the auditors are less important in a tutor-based system. The auditors do not need to perceive each other and the only user to be perceived by the others is the tutor. However, the way she/he is perceived should be different from a typical avatar approach, because the auditors are interested in her/his activities rather than appearance. To provide the auditors with a comfortable method of observing the tutor's activities, it is reasonable to enable them to see the world through the tutor's eyes [7]. Naturally, the auditors should also be able to act independently to practice their own skills.

The following two main observations for tutor-based multi-user systems has been made:

1. Avatars redundancy – auditors do not need to perceive each other, but tutor. Absence of avatars drastically reduces network traffic, since most of network's capacity is usually used for transmitting information about avatars' position and orientation. Moreover, there could be such a virtual environment, where the presence of avatars is even undesirable. As an illustration, consider tutor exemplifying how to handle a device or describing a complex object, like a molecule, engine, or solar system. In such scenes, it makes no sense to express auditors' positions (e.g. with avatars), since this information is irrelevant for educational tasks.
2. Users operate in two modes – *tracing mode*, used to watch the world through the tutor's eyes and *self-control mode* used to act independently.

3. System characteristics

Implementation of a tutor based multi-user system has to consider the following basic issues:

1. 3D scene rendering
2. Capturing information about tutor's behavior (interaction and navigation in VE)
3. Broadcasting the information to auditors
4. Utilizing the information by auditors to support *tracing mode*

It is reasonable and time saving to utilize maximally already existing technologies instead of reinventing the wheel. VRML as a proved and progressive internet standard for 3D scene representation in conjunction with Java cross platform programming language seems to form powerful basis for both experimental and prototype application development.

3.1. DILEWA architecture

The architecture of the DILEWA experimental system is predetermined to meet the following requirements:

1. Cross platform application – VRML as the Internet standard and Java fulfils this issue.
2. Minimum development effort – to exploit existing cross platform WWW browsers extended by VRML plug-in for VRML scene presentation.

Here we come to an accomplished scheme composed of Java applet communicating by EAI [2] with VRML plug-in, both running in a WWW browser [5]. Java applet is responsible for communication issues, VRML plug-in renders the scene and handles user's interaction, EAI is employed for information exchange between Java applet and VRML plug-in. Plug-in updates applet about tutor's behavior and conversely, applet instructs plug-in to support *tracing mode*. Finally a WWW browser is exploited as an application host.

3.2. Tutor's behavior capturing

User's behavior in VE can be divided into two groups: interaction and navigation. Interaction means user's activities impacting the scene, while navigation means changes of position and orientation of user's viewpoint.

Interaction capturing

In VRML, user's interaction with virtual environment is internally accomplished by *sensor* nodes, classified into two categories: *environmental sensors* and *pointing-device sensors.* Environmental sensors get activated when specific conditions in a scene are met, as e.g. when avatar collides with objects or avatar enters specified region. Pointing-device sensors get activated when the user locates the pointing-device over objects. When activated, the sensor node generates one or more *events* of various data types. To affect the scene in response to the user interaction, these initial events are passed to other VRML nodes. Connection between the source and the destination node is accomplished by mechanism called ROUTE in VRML. Once a sensor has generated an *initial event*, the event is propagated producing the events along any attached ROUTEs to other nodes. Those nodes may respond by generating additional events; this process is called an *event cascade* in VRML.

Thus, capturing of tutor's interaction involves capturing of generated events. However, not all events generated during the event cascade have to be captured. Instead, it is sufficient to capture only initial ones, which after delivering to their original destination nodes in auditor's client are able to trigger identical event cascade.

Navigation capturing

Environmental sensor `ProximitySensor` covering the whole scene is appropriate to support this task, because it generates continuous events while a user navigates through the scene. These events contain information about current user's position and orientation. Sending this information to auditor's client and setting auditor's viewpoint accordingly is a keystone for tracing mode support.

3.3. Auditor's modes

This section describes the key difference between the tutor-based multi-user systems and the social interaction aimed ones. The difference lies in providing auditors with ability of switching between two operational modes: tracing mode and self-control mode. While the self-control mode enables auditors to navigate and operate like in an ordinary single-user world, switching in tracing mode locks their activities and lets them just watch the world through the tutor's eyes.

Tracing mode

In this case, tutor's behavior represented by events has to be delivered to auditor. We have designed a mechanism of *net-routes* for this task. Semantics of net-routes are the same as for VRML routes – they serve as a connection from generating to receiving nodes. Unlike VRML routes, net-routes are intended to connect nodes located in different clients.

To simulate tutor's interaction in auditor's client, net-routes are to be established between both clients. Corresponding net-route mate is established for each such a route from VRML scene, which is fed by sensor node's initial events. Once initial events have been delivered, they are able to trigger event cascade described above, to ensure the same behavior in both auditor's and tutor's clients.

Synchronization of auditor's viewpoint accordingly to tutor is guaranteed by the help of net-routes as well. Two net-routes transmitting current tutor's position and orientation are established from tutor's `ProximitySensor` node to auditor's `ViewPoint` node.

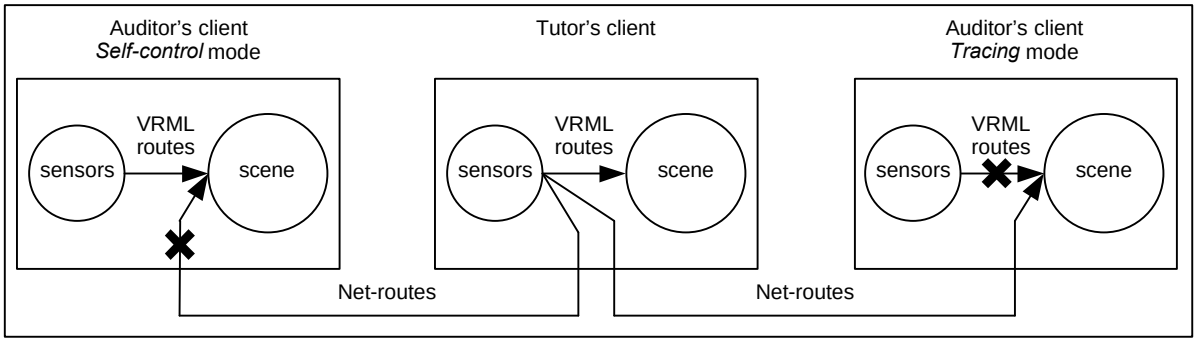

Figure 1. Auditor's modes and net-routes, the crosses mark deleted routes

However, synchronizing auditor's viewpoint is likely to activate the same environmental sensor nodes in auditor's client, which has been already activated in tutor's client. This would cause the same initial event being delivered twice – once along a VRML route in auditor's client and once along a net-route. To avoid this duplication, one of these routes has to be omitted. Deleting the original VRML routes in auditor's client has been proved to be the right solution due to the following reasons.

Firstly, events generated from environmental sensor nodes in tutor's and auditor's client can be different. Even more, some sensor nodes, which get activated in tutor's client, might not get activated in auditor's client. This inconsistency is caused by converting continuous tutor's movement into discrete values. While tutor's client interpolates between particular navigational steps, auditor's client just sets the viewpoint accordingly to these steps. Some intermediate states might be omitted, leading to some environmental sensors being not activated in both clients. `Collision` sensor serves as a good example; while easily activated by a tutor, it gets never activated in auditor's client, because browser adjusts tutor's location immediately, preventing tutor's client from sending collision location. The use of the net-routes ensures delivering of all initial events caused by a tutor, no matter if they have been generated simultaneously in auditor's client or not.

Secondly, it is desirable to prevent auditor from activating pointing-device sensor nodes in her/his client while a tutor is managing the scene. However, deletion of connected routes has the same effect – although sensors might be activated, they are not able to trigger any event cascade.

Therefore when auditor switches to tracing mode, the following actions take place:

1. Net-routes are established between her/his client and tutor's client. There are three types of net-routes:
   - Net-routes fed by environmental sensor nodes
   - Net-routes fed by pointing-device sensor nodes
   - Net-routes fed by navigational information
2. Corresponding VRML routes are temporarily deleted from the local scene.

Self-control mode

This mode provides auditor with single-user world experience. When switching from tracing mode, some types of net-routes are deleted and corresponding temporarily deleted VRML routes are restored. Decision about what types are to be deleted affects the consistency of the auditor's and tutor's scenes.

Net-routes fed by navigational information should be deleted, unless there is at least one another independent viewpoint an auditor can be switched to. Deletion of net-routes fed by pointing-device sensor nodes is more critical and we have to think about the purpose of the system. If the system is aimed for educational purposes, deleting net-routes is a proper solution, since we want provide auditor with ability to practice interaction with the scene independently. However, if the system has to serve for cooperation purposes, consistency of the worlds and preserving tutor's scene impacts are the essential requirements, that net-routes deletion would break. Similar considerations apply to net-routes fed by environmental sensor nodes, but these sensors typically play less important role in cooperation-designed scenes.

Thus, when auditor switches to self-control mode, the following actions take place:

1. Selected net-routes (accordingly to the purpose of the system) are temporarily deleted.
2. Corresponding VRML routes are restored

Figure 1 depicts two auditors' clients using tracing, and self-control modes, respectively. The crosses mark routes that are temporarily deleted.

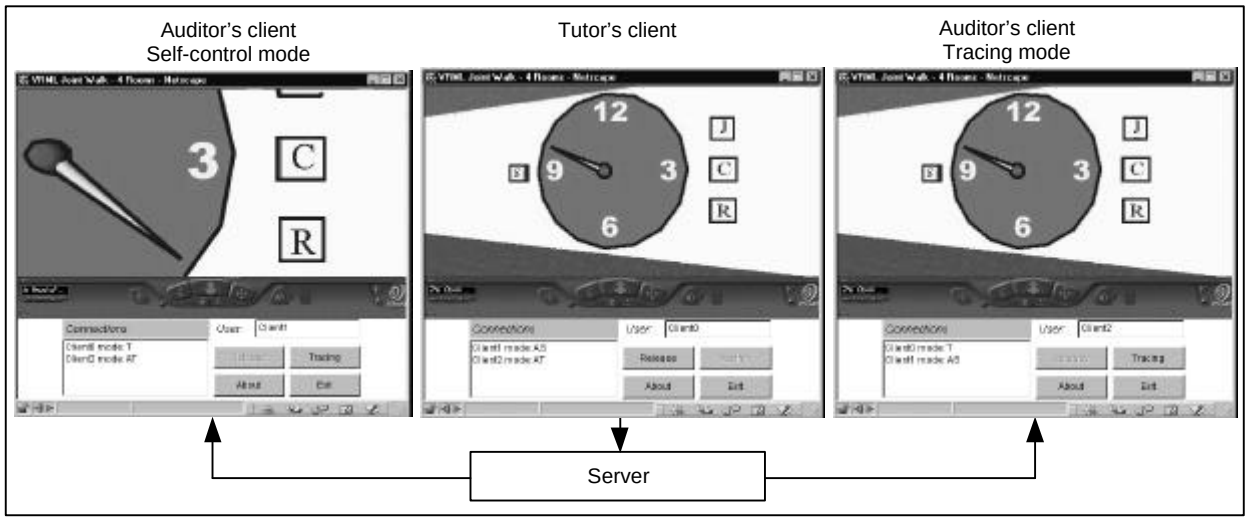

Figure 2. Sample scene with a stopwatch viewed in different modes. The leftmost window shows an independently working user while the other two are synchronized.

4. DILEWA implementation

This section deals with main implementation issues of our DILEWA experimental system: net-routes, DILEWA world authoring and user management.

4.1. Net-routes

Net-routes are nothing else, but unidirectional communication channels between two clients. Since VRML itself does not include any communication tools, Java applet supplies them. Because of security restrictions for Java applets running in a browser, applets cannot establish network connection to arbitrary network node, but the server they were downloaded from. This constrains communication scheme to follow client-server model.

Net-route mechanism is therefore implemented within two steps. Firstly, initial event is captured using EAI interface in tutor's client and sent via network to the server. Secondly, the server broadcasts this event to auditors' clients and events are delivered to receiving nodes using EAI.

4.2. DILEWA world authoring

Implementation of proposed features of the tutor-based system involves the following issues: capturing user's navigation, setting user's viewpoint, locating VRML routes fed by sensor nodes and obtaining the actual world time. Since current EAI implementation lacks direct support for this functionality, our approach is to extend a VRML world by extra nodes. These extra, but standard nodes are:

- `ProximitySensor`, covering the whole scene, for capturing tutor's navigation,
- additional `ViewPoint` for setting auditor's viewpoint,
- `Anchor` holding in its `parameter` field a list of VRML routes fed by sensor nodes,
- `TimeSensor` for providing the applet with VRML scene clock.

The Anchor node holds an extra list in its `MFString parameter` field. Particular route is described as a regular VRML route by four strings identifying source and destination node/event pairs. Since only nodes named by DEF statement are accessible through EAI, possible sensor nodes encapsulated in user PROTOs are inaccessible and their events cannot be captured. This can be overcome by exposing sensor's node events in PROTO's interface declaration.

Very common approach is to use a sensor node to activate `TimeSensor` node (note that `TimeSensor` node is not considered as generating initial events in such a case) by routing some `eventOut` to `eventIn startTime` of `TimeSensor`. Since the sensor generated event represents 'now' time, `TimeSensor` is activated immediately. This works fine in local client, but fails when attempting to transfer 'now' time along net-route to activate `TimeSensor` in different clients with different scene clock. While other systems overcome this issue by introducing delta time fields [3] or by generating `SFTime` event from `SFBool` event with the help of `Script` node [5], our solution is to substitute every `SFTime` event delivered along net-route by actual scene 'now' time using the extra `TimeSensor` node.

4.3. DILEWA user management and GUI

In DILEWA architecture, the server, implemented in Java, is responsible for event broadcasting and basic user management. Users are identified by their names/nicknames and the server maintains a list of active users and their roles and modes. The first logged user obtains the tutor role and the others become auditors. Each user can see the list of connected users; auditors can switch between tracing and self-control mode; the tutor can entrust an arbitrary user with tutor's role. This is accomplished by applet GUI as depicted on Figure 2.

5. Conclusion and future work

In this paper, we have presented possible exploitation of multi-user virtual environments for educational purposes. We have introduced notion of tutor-based system and pointed out basic requirements such a system should met. It has been proved that system can be implemented using VRML extended by newly designed mechanism of net-routes, which supply communication issues where VRML lacks.

Further improvements are still under the development e.g. event filtering, state encoding, and object locks in a scene. The first issue concerns the situation when distributed participants use machines with big differences in their real-time capabilities. High number of events coming from hi-end tutor's client can cause overloading and delays in other, possibly low-end clients. The second point helps late coming auditors to immediately update their local scene. The third problem is the key point for distance cooperation extension of the system. Although the primary task has been to use the tutor-based VE system for educational and presentational purposes, the possibility to use it for real cooperation is slightly different, but accomplishable challenge.

6. References

[1] The Virtual Reality Modeling Language. *International Standard* ISO/IEC 14772-1:1997, http://www.web3d.org/technicalinfo/specifications/vrml97/

[2] The Virtual Reality Modeling Language External Authoring Interface. *Committee Draft* ISO/IEC 14772-2, http://www.vrml.org/WorkingGroups/vrml-eai/Specification/

[3] *Living Worlds*, http://www.vrml.org/WorkingGroups/living-worlds/

[4] *Blaxxun Interactive*, http://www.blaxxun.com/

[5] G. Reitmayr: "Behind the scenes of a VRML multi-user architecture". *VRML99* – Fourth International Conference on the Virtual Reality Modeling Language and Web 3D Technologies, 23-26 February 1999, Paderborn, Germany.
http://www.c-lab.de/vrml99/
http://www.geometrek.com/

[6] D. Margery, B. Arnaldi, and N. Plouzeau: "A General Framework for Cooperative Manipulation in Virtual Environments". *Virtual Environments '99*, Proceedings of the Eurographics Workshop in Vienna, Austria, May 31-June 1, 1999, pp. 169-178.

[7] K. Engel, and T. Ertl: "Texture-based Volume Visualisation for Multiple Users on the World Wide Web". *Virtual Environments '99*, Proceedings of the Eurographics Workshop in Vienna, Austria, May 31-June 1, 1999, pp. 115-124.