Long Distance Distribution of Educational Augmented Reality Applications

H. Kaufmann[1], M. Csisinko[1], A. Totter[2]

[1]Institute of Software Technology and Interactive Systems, Vienna University of Technology, Austria.
[2]Organisation, Work and Technology Group, Swiss Federal Institute of Technology, Switzerland.

Abstract

For distance education utilizing shared Virtual or Augmented Reality (VR/AR) applications, reliable network distribution of educational content is of prime importance. In this paper we summarize the development of software components enabling stable and reliable distribution of an existing educational AR application for geometry education. Our efforts focus on three main areas: (1) For long distance distribution of Open Inventor scene graphs, throughout a wide area IP network, a TCP based network protocol was implemented in Distributed Open Inventor. (2) A tracking middleware was extended to support sending tracking data unicast instead or in addition to sending multicast messages. (3) Multiple adaptations in our geometry application were required to improve scalability, robustness and reliability. We present an early evaluation with high school students in a distant learning, distributed HMD setup and highlight final results.

Categories and Subject Descriptors (according to ACM CCS): K.3.1 [Computer Uses in Education]: Distance learning, I.3.2 [Graphics Systems]: Distributed/network graphics, K.3.1 [Computer Uses in Education]: Collaborative learning, H.5.1 [Multimedia Information Systems]: Artificial, augmented, and virtual realities.

1. Introduction

In order to use Virtual or Augmented Reality applications in realistic, educational settings, a large group of students must be able to participate either actively or passively in the activities taught in VR/AR. In distance learning with VR/AR, reliable network distribution and replication of educational content is of prime importance.

Our work is based on the educational Augmented Reality application Construct3D [KS06, KS03]. This system deploys Augmented Reality (AR) to provide a natural setting for face-to-face collaboration of teachers and students. The main advantage of using AR is that students actually see three dimensional objects which they until now had to calculate and construct with traditional (mostly pen and paper) methods (Figure 1). By working directly in 3D space, complex spatial problems and spatial relationships may be comprehended better and faster than with traditional methods. Our system utilizes collaborative augmented reality as a medium for teaching, and uses 3D dynamic geometry to facilitate mathematics and geometry education. Pedagogical aspects influenced the design of collaborative AR hardware setups, user interface design and content design as reported in [Kau04]. This paper focuses on the technical development and recent advancements enabling distribution in order to serve groups of students and pedagogical findings when using Construct3D in a distributed setup for distance education.

Figure 1: *Collaborative co-located work in Construct3D.*

When sharing a virtual workspace for collaboration with people at distant locations, distribution and replication of data has to be taken into account.
Ideally, data transmission should be fast to achieve fast response times. Especially in long distance distribution this aspect is crucial, as the travelling time depends on the distance to cover. In addition data transmission has to be (in most cases) reliable. The amount of transmitted data should be low to achieve fast response times, to prevent network congestion, and to increase efficiency. We distinguish between different types of distributed data:

- Input data distribution:
  Tracked input device data to visualize the actions and movements of other participants, especially those working in distant locations. This type of data is typically sent by a tracker server in VR/AR environments.
- High level application state distribution:
  Shared application state, in the form of compacted metadata to reduce the amount of transmitted data. Ideally these metadata suffice to regenerate the correct application state without actually transmitting the whole application.
- Output data and application content distribution:
  (Educational) application content or additional application data that needs to be shared.

To allow collaboration between distant users immersed in a common shared space, a consistent application state is required throughout all participating sites. This implies that each participant perceives a similar virtual world, although slight differences might be possible. In a teaching scenario, for example, user roles (teacher - student) could be defined which result in displaying additional information - such as the solution of a given 3D construction - to a teacher, whereas the student does not see the solution.

1.1. Contribution

Hesina [Hes01] introduced Distributed Open Inventor (DIV) and distribution features in Studierstube [SFH*02], our Augmented Reality software framework, but a series of remaining shortcomings had to be resolved. Existing functionality was restricted to local networks, because the network implementation makes use of multicast UDP. This networking mode, though theoretically ideal for the task at hand, lacks of support for long distance distribution: As a major drawback multicast UDP packets are in general not sent through arbitrary routers on the internet (unless part of the MBONE network [Eri94]). Therefore immediate distribution between school networks, without setting up multicast tunnels in cooperation with local administrators is not possible. In our experience gathered in past e-learning projects, these obstacles (which require time and effort of school personnel) usually prevent usage of the technology in an educational setting.

Our work is supposed to fill this gap, enhancing distribution features, overcoming the rigid restriction in terms of networking and offering the possibility to truly distribute Studierstube applications such as Construct3D to remote places. Due to the practical usage of DIV by Hesina for the past 5 years, the shortcomings (as mentioned above) became obvious. Increased interest in Construct3D and collaborative projects with partners in other countries required a flexible network implementation, breaking free from standard lab setups in a LAN. Efficient mechanisms for distributing AR/VR applications over long distances had to be implemented. This was done on 3 levels:
(1) Tracking middleware (OpenTracker [RS01]) was extended to send data of tracked input devices in unicast UDP mode in addition or instead of multicast UDP.
(2) Distributed Open Inventor [Hes01] was extended to send data using (reliable) TCP instead of reliable multicast UDP. It enables long distance distribution but also leads to a considerable performance increases in small networks compared to the multicast UDP implementation.
(3) Construct3D [KS03] (section 4) was selected to make in-depth long distance distribution tests. In addition to extending distribution functionality, we enhanced the replication behavior of the application. Initially the whole application state - as a scene graph containing all geometric objects - was transmitted which resulted in a high amount of transmitted data. To minimize the network load only state data is being transmitted which enables clients to rebuild the whole application state themselves. Therefore distribution is basically restricted to meta information in the form of command lists containing essential application states. Executing a command list generates the whole geometric construction and application state.

Another huge amount of work was spent on massively increasing robustness of present features by bug-fixing and reimplementation as well as extending them and introducing new functionality to push the application further. They are of major importance for a stable educational application but are mainly omitted in this context.
Finally an early evaluation of a distributed educational setup is presented which shows the usefulness of utilizing AR/VR applications, in example Construct3D, in distance education.

2. Related work

For the development of any educational, distributed VR/AR application, technological, domain specific, pedagogical and psychological aspects are of importance. Accordingly, literature from different and diverse research areas relates to our work: Tracking frameworks, distributed scene graphs, collaborative AR/VR, distributed virtual environments, desktop and immersive 3D modeling, educational 2D/3D applications, dynamic geometry and pedagogic theories such as constructivism or activity theory. We will briefly mention work related to the core parts of our work. For a comprehensive overview of related work regarding Construct3D we refer to [Kau04].

2.1. Tracking frameworks

With the wealth of different tracking systems and input devices available, it is impossible for application developers to deal with the details necessary to support each and every technology natively in their applications. Instead, it is desirable to add another level of abstraction, and try to encapsulate the details of the necessary software support for various tracking technologies in a tracking middleware. The goal of tracking middleware is to serve tracking (and other input) data to the application, independent of the underlying hard- and software. Several middleware systems for tracking devices have been developed.
VRPN (Virtual-Reality Private Network) [THS*01] is a wide spread device-independent and network-transparent framework for peripheral devices used in Virtual and Augmented Reality applications written in C++. Networking is built upon UDP and TCP. Depending on the reliable delivery property of the tracking data type, the protocol is chosen on a per message basis.
With OpenTracker [RS01] it is also possible to support distinct tracking device types by abstraction, to perform various preprocessing tasks (such as filtering) and network transmissions within a single framework. An OpenTracker client is integrated into the Studierstube [SFH*02] toolkit for tracking device support.

2.2. Distributed scene graphs

Current high-level 3D graphics libraries are engineered around the concept of a scene graph, a hierarchical object oriented data structure of graphical objects. Such a scene graph gives the programmer an integrated view of graphical and application specific data, and allows for rapid development of arbitrary 3D applications. Although shared memory systems are capable of directly sharing data, they have additional hardware requirements. Distributing and replicating scene graphs among heterogeneous computer systems does not require additional hardware.

The blue-c Distributed Scene Graph (bcDSG) [NLSG03] is based on OpenGL Performer. Distribution features are added on top of the blue-c framework and are not integrated into Performer. The scene graph can be divided into a shared and local partition. Shared parts have to be created using custom nodes, as standard Performer nodes do not support distribution. Scene graph synchronization is performed in a traversal operation at each rendering. This mechanism includes consistency, locking and ownership features.
Data transfer is done using UDP, enabling multicast support for more than two participating sites: While scene graph synchronization messages are transmitted to any participating site, locking operations are of unicast nature. Relying on multicast UDP and its routing deficients, the system will experience afore mentioned problems when used for large distance distribution. Its synchronization features are based on nodes as atomic units: Changing a single field causes the whole node contents to be transferred. This can be problematic, when having huge amount of data belonging to a single node.

Avango [Tra99] is also based on Performer. Similar to the Inventor toolkit, its own scene graph nodes act as field containers, storing data in terms of fields. In addition field connection and streaming mechanisms are introduced similar to existing concepts in Inventor. Distribution features are based on so-called distribution groups. To build a shared object, a local object has to be created and migrated to a distribution group. On the receiving end all group members reverse this process by creating a local copy of the distributed object.

Distributed Open Inventor is based on Open Inventor (OIV), a popular scene graph toolkit. Several implementations of adding distribution features to OIV exist:
Distributed Open Inventor (DIV) by Hesina [Hes01] is a stand alone open source add-on to OIV, and has also been integrated into the Studierstube framework [SRH03]. It enables sharing of a scene graph or parts of it in a network, which is a fundamental prerequisite for (distant) collaboration in AR/VR environments. If encapsulating application and its graphical object state altogether in a scene graph, distribution of that scene graph avoids the dual database problem [MF98]. Since DIV provides the basis of our work, we will describe some of the concepts implemented in DIV in detail:
The implementation makes use of the notification mechanism in OIV and observes occurred scene graph changes by sensors. On an atomic level changes of field values are monitored. For convenience, a special group called DivGroup denotes a subtree for distribution, offering the possibility to share several independent parts of a scene graph.
Usually a single master hosts the original copy of the scene graph for replication to guarantee total ordering of messages. The master is responsible for transmission of scene graph changes to the network. In this transmission the node name (where the field value change occured) is used as unique identifier and naming lies in the responsibility of the master. Scene graph modification messages transmitted by the master typically contain the name of the node where the change occurred with additional information such as (a) appropriate field data, if a field update occured or (b) structural information, if the update is of structural nature (involving group node operations).
Slaves process received changes and modify the scene graph. Initially, slaves are also capable of sending polling packets to the network, requesting the scene graph from the master. The master reacts on this message appropriately by transmitting the scene graph in its actual state. This is actually the implementation of a late joining feature. Networking is based on the ACE toolkit.

A similar approach to distributed Open Inventor was implemented by Pečiva [Peč02], also based on a master-slave architecture. Similar to Hesina, he extended the Open

Inventor source directly. Therefore scene graphs can be set up for distribution without replacing standard nodes by a customized counterpart.

2.3. Educational VR applications

Since the early 1990th researchers have been working on virtual reality applications for purely educational use ([DSL96, WB92] and many others).
In the area of mathematics education the most advanced immersive VR project is CyberMath [TN01]. CyberMath is an avatar-based shared virtual environment aimed at improving mathematics education. It is suitable for exploring and teaching mathematics in situations where both teacher and students are co-present or physically separated. It has been presented in a CAVE and exists as a desktop VR application. The recent VRmath system [YN04] is an online application that utilises desktop VR combined with the power of a Logo-like programming language, hypermedia and the Internet to facilitate learning of 3D geometry concepts and processes. A very good summary of educational VR applications is given by Mantovani [Man01].

2.4. Pedagogic theory

Constructivist theory provides a valid and reliable basis for a theory of learning in virtual environments [Osb97, Win93]. As constructivism underlines, learning takes place when students build conceptual models that are both consistent with what they already understand and with the new content.
The core commitment of a constructivist position is that knowledge is not transmitted directly from one knower to another but is actively built up by the learner. Learning is considered to be an active process in which learners "construct" their own knowledge by testing ideas and approaches based on their prior knowledge and experience, applying these to a new situation, and integrating the new knowledge gained with pre-existing intellectual constructs. This is supported through relevant, engaging learning activities, which involve problem-solving and critical thinking. We used activity theory [Eng99, TGG04] as a conceptual framework to design constructivist learning tasks for our evaluation. Details are given in subsection 5.2.

3. Distribution - Technical Design

In this section we provide a brief overview of our design of the three components that were extended to support long-distance distribution: OpenTracker, Distributed Open Inventor and Studierstube. In section 4 Construct3D is described.

3.1. Tracking data distribution in OpenTracker

OpenTracker [RS01] contains components providing tracking data transmission between several OpenTracker instances on different hosts. Just like Distributed Open Inventor these capabilities are built upon multicast UDP, which causes the earlier mentioned multicast-related problems.

Following the data flow principle of OpenTracker, tracking data is inserted into the data flow graph by means of a so called NetworkSource, while a NetworkSink transmits data to the network. This implies that network traffic concerning tracking data is unidirectional and of multicast nature: Payload data is always transmitted by a single NetworkSink and received simultaneously by one or more NetworkSources. It was rather straightforward to add unicast UDP as an additional networking protocol. A NetworkSink generating tracking data packets has to deliver them simultaneously to associated receivers. To know all receivers, the NetworkSink has to maintain a list of counterparts (each of them usually a NetworkSource of an OpenTracker instance on the receiver side).

The network topology on the logical level is a star. This topology implies that tracking data of several devices can be distributed by a single network, as long as data occurs on the same central location. Of course building several independent networks (consuming more network resources) is the alternative and more general way, as this allows distribution of tracking data occurring at different places. Establishing a tracking data network is initiated by NetworkSinks, similar to the traditional server-client scenario: Each client must have knowledge in advance about the server providing desired tracking data in terms of socket information (host and port).

3.2. Distributed Open Inventor

In general Distributed Open Inventor is utilized to distribute parts of a scene graph. Since scene graph data usually contains important application information, network communication must be reliable. To overcome the borders of private local networks, TCP as a very widespread and reliable network protocol was chosen. This implies a lot of changes to Hesina's implementation.

In contrast to multicast UDP, TCP allows only point-to-point communication. On the other hand no reliability treatment is necessary in TCP as this is implicitly taken care of in the protocol. To allow data delivery to all network nodes, the TCP implementation, considering its unicast nature, has to emulate multicast data delivery to comply to the requirements of Distributed Open Inventor. As any node might act as a server, a many-to-many property has to be taken into account.

Multicast data delivery with multiple senders is done by building up a logical network of so called true mesh topology. This is a network, where each peer is logically connected to each other peer. The main challenge of the TCP implementation is to establish and ensure true mesh topology at any time. Sending and receiving is, as mentioned before,

fairly simple: Data is automatically transmitted to each connection simultaneously. On the receiving end nothing special has to be taken into account. Processing order of data received from different connections is uncritical as the next higher network layer implies that critical data in terms of processing order is sent from exactly one peer at any time.
Whenever a peer joins the distribution network it must know at least one peer of the existing network. Otherwise it will be the only participant of a new network.
A new peer initially contacts the network by sending a special message identifying itself just after connection establishment. This identification contains the server port of the peer as each peer contains server and client functionality. The arrival of a new peer must be forwarded to all other participating sites of the network. All other peers establish connections to the new peer identifying themselves.
On receiving any identification message, the peer has to check, if another connection to the counterpart currently exists. If this is the case, the connection is closed immediately. Since network redundancy involving more than a single peer is effectively prevented in advance, connection closing should only be done in case of cycles. Livetime information (time to live (TTL)) is embedded in the message of a joining peer. It allows only a certain low number of hops between peers. This increases efficiency and helps to avoid infinite cycles.

3.3. Studierstube

The enhancements of Distributed Open Inventor have to be reflected in Studierstube. Studierstube applications are distributed automatically by a DivGroup, implicitly created as a parent of each application. Configuration of the distribution is handled by the core library with the help of a tool called session manager. Implicit distribution is implemented in favor of ensuring distribution capabilities without further intervention by the application programmer.

The session manager assigns master property to the participant who originally hosts a certain application. All other participating sites (slaves) receive the application scene graph via network due to the node transfer feature. Terminating the master results in reassigning master property to another participant. This master-slave property assignment is conducted autonomously and cannot be influenced by Studierstube instances. Another task of the session manager is to create network resources according to the requested networking mode. To do this, a generator produces network configuration data. In multicast UDP mode, a single multicast group address and associated port number is generated per application. In TCP mode, each peer is given a unique port number to allow running several applications on a single machine.
As the assignment of network resources lies in the responsibility of the session manager, it simply creates a list containing proper contact information of all other participants for each peer and includes this in reconfiguration messages sent to each participating site. This strategy guarantees highest chances to contact any of the other peers to successfully build up a network.

Further details about design and implementation of our approach are described in [Csi06].

4. Construct3D

Construct3D is based on the Studierstube AR system [SFH*02] and uses augmented reality to provide a natural setting for face-to-face collaboration of teachers and students. Based on an underlying distribution mechanism, Studierstube extends its support to multiple users working with multiple different display techniques in a shared workspace that features multiple applications and management techniques similar to a common 2D desktop [SRH03]. Studierstube applications are custom nodes which are part of the scene graph. As Construct3D is just another Studierstube application, it inherits automatically its distribution features.

4.1. Software design

Construct3D offers functions for the construction of points, two-dimensional geometric primitives and three-dimensional geometric objects. It provides functionality for planar and spatial geometric operations on these objects, allows measurements, features structuring of elements into layers and offers basic system functions.
Construct3D promotes and supports exploratory behavior through dynamic geometry. A fundamental property of dynamic geometry software is that dynamic behavior of a construction can be explored by interactively moving individual defining elements such as corner points of a rigid body. For example, moving a point lying on a sphere results in the change of the sphere's radius. It can be seen what parts of a construction change and which remain the same. The histories of constructions as well as dependencies between geometric objects are maintained. Experiencing what happens under movement allows better insight into a particular construction and geometry in general.
At its start Construct3D initializes a 3D window and the user interface. The menu system is mapped to a hand-held tracked panel called the personal interaction panel (PIP) [SG97]. The PIP allows the straightforward integration of conventional 2D interface elements like buttons, sliders, dials etc. as well as novel 3D interaction widgets (Figure 2). Passive haptic feedback from the physical props guides the user when interacting with the PIP, while the overlaid graphics allows the props to be used as multi-functional tools. Students can position written notes onto the tablet for instance that might help them during their work in the virtual environment.
All construction steps are carried out via direct manipulation in 3D using a stylus tracked with six degrees of freedom. In order to generate a new point the user clicks with his pen

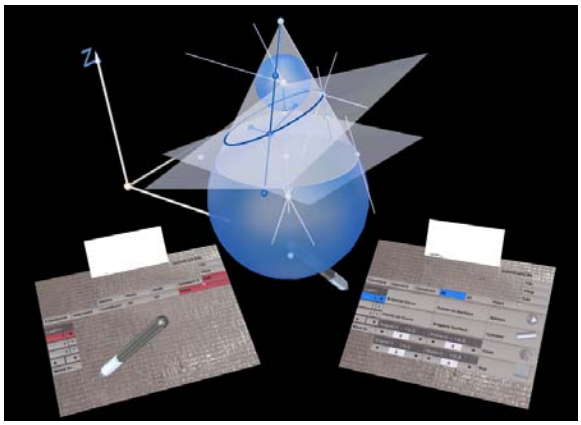

Figure 2: *Two users collaborate on a construction in Construct3D. To distinguish users' contribution each user is working within an own color scheme (note the differently colored menus).*

exactly at the location in 3D space where the point should appear. Users can easily switch between point mode (for setting new points) and selection mode (for selecting 3D objects). All 3D operations consistently support dynamic modifications of their input elements and re-evaluate the resulting elements accordingly. Necessary system operations such as selection and deselection of primitives, save, load, delete, undo, redo, export and import of VRML files are provided too. Details on the implementation, specifically the implementation of undo, redo and other features for multi-user environments are explained in detail in [Kau04].

The internal structure of the application's scene graph is rather simple. Avoiding the dual database problem [MF98], it encapsulates all of the application's data. Basically the scene graph hierarchy is composed by command lists storing all Construct3D operations. Geometric objects are the visible results of these operations. A command list and its interpretation (= the geometric elements) are represented by node kits. The command list represents the meta-state of the application, the node kits containing the geometric elements are rendered and represent the visual state.
These two distinct parts forming the application's scene graph are interrelated: Manipulation of geometry causes the generation of new commands in the command history list. On the other hand, the execution of commands generates deterministic results on visible geometry. Therefore the command history list is used for file operations (load/save) and for the undo/redo functionality.

An important property of the command history list is that it allows the complete regeneration of the application state at any specific time during the construction process. Therefore distributing the command history list alone is sufficient to rebuild the correct application state on any client computer. Other parts of the scene graph are excluded from distribution. As each action causes a modification of the command list position pointer, the latter is a key element in assisting the detection of changes caused by distribution. Whenever the shared pointer changes, actions have to be taken.

5. Evaluation

Being one of the longest actively developed educational AR applications, Construct3D has been used with teachers and students in more than 500 teaching lessons yet. Usability aspects of Construct3D and pedagogical content design have been evaluated in two previous evaluations as summarized in [Kau04] providing very good results and useful feedback. The current evaluation focuses on distributed, distant learning.

5.1. Collaborative, distributed hardware setup

The standard setup used for Construct3D supports two collaborating users wearing stereoscopic see-through head mounted displays (HMDs) (see Figure 1) providing a shared virtual space. The users interact with the system using pen and pad props. Both users see the same virtual objects as well as each others' pens and menu systems which allows a student or teacher to help the other user with the menu system for instance if necessary. The same is valid in a distance learning scenario since input device data is shared amongst remotely located users. Because of see-through head mounted displays they perceive their real bodies, gestures and actions and those of people outside the virtual space, i.e. a teacher, as well which is especially important for co-located work. Head and hands are tracked using an ARTTrack optical tracking system. In a co-located setup one dedicated host with 2 graphic ports renders stereoscopic views for both users. In distributed setups rendering as well as computation of the geometric objects is done locally on each participant's PC.
Our immersive setup that uses head mounted displays is most favored by teachers and students. The big advantage of this setup is that it allows users to actively "walk around" geometric objects which are fixed in space. Excited students sometimes lie down on the floor to view objects from below or step on a chair to look down from above. This is a unique feature of an HMD setup which cannot be provided by monitor or projection screen based hardware configurations. It actively involves students and therefore complies with constructivist learning theories. Geometric objects are not abstract anymore but in spatial relation to the learner's own body, they can be manipulated directly and are nearly tangible. We think these are key features to learning and to improving spatial abilities with Construct3D.

Other AR setups for educational use have been tested with Construct3D such as a basic desktop setup, semi-immersive, mobile and hybrid setups which are described in detail in [KS03].

5.2. The link to pedagogical theory

In the course of this study the technical requirements of Construct3D, the learning tasks as well as the evaluation methodology were aligned in accordance with pedagogical concepts and learning practices based on constructivism, combined with action-oriented learning such as real-problem solving, collaborative learning, exploratory learning and interdisciplinary learning, stemming from activity theory and the theory of expansive learning [Eng87, Eng99]. In particular, learning tasks had to

- be part of the actual curriculum in schools.
- represent a holistic real life problem. The description (instruction) of the task should be embedded in authentic (real life) context and not at the level of an abstract instruction.
- offer the possibility to be viewed from several perspectives. The focus on different perspectives should support the transfer of knowledge to other similar, but not identical problems.
- be available in multiple representations (different kind of visualisations of the task).
- meet the experience and interests of the students Ű which kinds of tasks they are already familiar with, what kind of problems might they be confronted with in the near future, etc.

The pedagogical theories also influenced the context in which the learning takes place. During the learning process, the following aspects were considered: Learning was an active process, and students collaboratively performed practical tasks to improve their procedural knowledge. Students structured and controlled the learning process. They chose the approach, and the methods for solving the task. The learning process should enable knowledge construction; students should develop their own ideas and approaches. They should be able to identify a contradiction or a conflict in the task. Students should investigate their learning with respect to methods used to organize their information and interpretation. Finally, they should analyze and evaluate their solution with respect to strengths and weaknesses.
The teacher acted as a coach, analysing students' strategies during the collaborative learning process, diagnosing mistakes and misunderstandings and supporting students.

5.3. Evaluation design

Early at the development phase we conducted a first evaluation with distributed Construct3D to investigate its usefulness for distributed collaborative learning and teaching. The evaluation was based on the methodological framework CIELT (Concept and Instruments for evaluating learning technologies [TGG04]). Within the evaluation design we discriminated three phases for the evaluation. The preparation-phase was the period before the actual evaluation sessions start. During the experiments, learning was observed and the assessment-phase concluded the experiments.

The evaluation design for the experiments is shown in Figure 3. To be able to derive information about the effectiveness

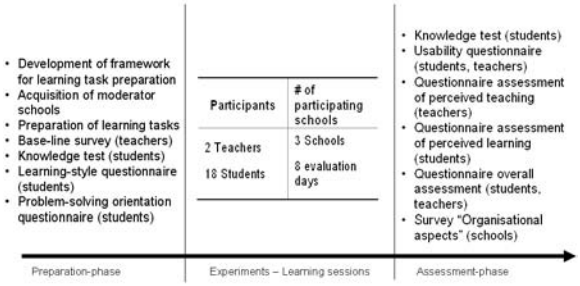

Figure 3: *Students working in a distributed HMD setup in two different labs.*

of distributed collaborative learning, we followed a quasi-experimental design for the geometry experiment, which involved splitting learner groups. Seven learner groups performed three geometry tasks in a face-to-face setting, working next to each other, whereas two learner groups performed these three tasks in a distributed setting (Figure 4). For the distributed setting each group member was located in a different room. A learner group consisted of two students and one teacher. In both settings both users were wearing HMDs, were tracked by an optical tracking system or a magnetic tracking system in the second lab, and a teacher was watching (in a different room) on a monitor.

For this early evaluation all students were located in rooms in the same building, therefore issues of latency in long distance collaboration were not faced at that stage. Long distance collaboration was tested throughout the development process and with the final implementation at an informal level. Application specific adaptations for improved robustness had to be implemented in case of delayed messages arriving from different sources (e.g. tracking data coming from a different source than application data). Large scale testing is part of our future work (see 6.1).

Figure 4: *Students working in a distributed HMD setup in two different labs.*

5.4. Learning tasks

Based on the above mentioned characteristics, three learning tasks were developed. Each group had to accomplish these tasks during the evaluation.

- The first learning task dealt with the wheels of an airplane, which were rotated into a shell in the hull of an airplane after its take off. The wheel in its start and end position were given. Students had to construct the axis of rotation and the angle of rotation for the wheel from its original position into the position in the hull. A screenshot of the given elements that they will face later in the virtual world was given as well.
- In the second task, a satellite dish had to be adjusted to point to the TV-SAT2 satellite. Students had to translate this real life problem into a geometric problem to be able to identify two angles, which are needed to adjust the satellite dish. Web links were presented with additional information about geostationary satellites; images were also given to help understand and translate the problem. The virtual scene in Construct3D showed a small model of the earth where all continents and seas could be seen, to help pupils find the correct places on earth and to immerse them further into the problem.
- In task 3 a rope was redirected from one given position to a final given position. Two deflection sheaves, which were drawn as circles, redirected the rope. These had to be constructed by the students. Deflection sheaves can be found in skiing lifts, elevators and many other machines. A draft was given to lead the students to a correct solution.

5.5. Evaluation results

Overall 18 students and 2 teachers participated in the evaluation of the C3D system. Students were between 17-18 years old and attended grade 11 or 12 of Austrian senior high school. They all head average to good computer experience.

The usability of the C3D system was measured with the ISONORM 9241/10 questionnaire [Prü97]. The overall usability of C3D was rated good, with M=2.37 (SD=0.42; with values ranging from "1=Completely adequate" to "5=Completely inadequate"). When analysed with respect to the two settings, distributed and face-to-face learning, there was no significant difference between the two groups, except for the principle of suitability for learning. There, groups in the distributed setting rated the suitability for learning statistically significant better than groups in the face-to-face setting (p=.01). We want to point out that due to the small sample, results have to be interpreted carefully. In an additional interview, students of the distributed setting mentioned that due to the distributed setting they concentrated better on what the other person said or did.

Open ended questions from the participants were analysed in a qualitative way to summarize the difficulties experienced while working with Construct3D. Most frequently students complained about an instable distributed system that crashed during the experiments. These difficulties were related to the very early trial of our implementation. We got very useful feedback which helped to make distribution very robust and develop the final system as described in this paper.

Participants also rated the perceived usefulness of Construct3D for meeting (1) learning, (2) communication and (3) collaboration needs (1=very good, 5=not at all). The mean values for those 3 categories were between 1 and 2.14. There was no statistically difference between the distributed and the face-to-face group.

Furthermore participants rated the perceived collaborative awareness (ranging from "always = 1" to "never = 5") based on findings from Caroll et al. [CNI*03]. The authors stated that three aspects of awareness have to be taken into account measuring the effectiveness of the collaboration. Students rated their awareness (while working with Construct3D) of other *working* students with 1.17 (std.dev. $\sigma = 0.707$). Their awareness of *interacting* colleagues was rated with 1.50 ($\sigma = 0.632$) and their awareness that other users are *thinking and planning* was rated with 2.06 ($\sigma = 0.873$). The good ratings could be explained by our very specific application design with respect to supporting multiple users. A different color scheme is used for every user which allows teachers and students to clearly distinguish between each user's contribution [KS06]. In co-located setups collaboration is supported by Augmented Reality, specifically see-through head mounted displays which enable users to see the movements of others.

To investigate the learning outcome we differentiated between successful and non-successful groups. The learning outcome was measured in two ways. Firstly, for each task a specific time frame was defined. After that, groups had to stop working on the task, but the teacher explained the solution. The time frame for the three geometry tasks was set for 45 minutes each to complete the task. Learning sessions either stopped when students solved the task or after the set period of time was reached. Second, a fixed, quasi-experimental design was used, following the traditional pre-knowledge test - intervention - post-knowledge test design. Before the actual experiment started, students had to fill in a multiple choice test, trying to find the correct answers for 8 geometry content related questions. The correct answers of this pre-test constituted the individual base-line for each student, providing information concerning knowledge about the geometry topics that each student had before the actual topics were taught. After performing the three tasks students had to fill in another multiple choice test, again trying to find the correct answers for 8 content related questions. The results of the knowledge pre- and post test could then be compared providing information about how much knowledge had been increased during the experiment.

In the geometry experiment 9 groups (2 students and 1

teacher) from three schools in Austria participated, performing 3 tasks. Five out of the nine groups solved all 3 tasks within the given time frame (45 minutes for each task). Based on the results of the knowledge pre- and post tests we investigated changes in the domain specific knowledge. In the pre-test students were able to answer on average 4 out of the 8 questions (M=4.67), after the experiments they were able to answer almost 6 out of the 8 questions (M=5.86). This difference between the pre- and post-test was statistically significant (p=.003). A positive correlation between pre- and post-test was found (r=.66, p=.003). These results provide first hints that learning in a distributed Construct3D setup has positive effects on the knowledge increase of students. In both co-located and distributed learning groups the knowledge gain was high. No difference was found, therefore distance learning did not effect knowledge gain in any negative way.

We were extremely surprised to see that students collaborated in the distributed, distance learning setup without any problems. Four participants from a school for highly-gifted students were extremely skilled collaborating in the distributed setup. They said that it's even easier than co-located collaboration because they can fully move around geometric objects and can freely interact with them without being considerate of other people who physically share virtual and real space with them in a co-located setting.

6. Conclusion

Running DIV on TCP enables long distance distribution without the effort of tunneling or relying on special infrastructure (MBONE [Eri94]). Depending on the amount and complexity of scene graph data, initial node transfer takes some time. But after this initialization state, interaction has to be fast and responsive.

Comparing the multicast UDP and TCP implementation, it is easily observable that TCP performance is over-topping multicast UDP, especially in small networks: Generating huge amounts of DIV updates by heavily manipulating the scene graph contents, network data throughput in the TCP implementation seems to be much better. On multicast UDP, the send queue gets comparatively quickly full, causing rendering thread blocking of the master. Consequently, interactive manipulation is not possible while having the render thread waiting for dequeuing to take place. Maintaining the same conditions (queue size) while running these massive stress tests, this blocking phenomenon could not be achieved on TCP. We assume that this is related to a sub-optimal implementation of multicast UDP in the ACE network library that we use.

Construct3D is fully benefiting from all distribution features: Multi-user functionality raises the demand for tracking data distribution. Supporting master transfer ability is desired for more flexible use cases. Generally speaking, a very high degree of flexibility is ensured by three orthogonal aspects:

- User configuration and user resources such as output devices, panels and pens can be freely specified.
- The host of the application (DIV master) can be selected without restriction and is completely independent of associated users and their resources. Startup order is completely insignificant and the master automatically migrates by session management on termination.
- Finally, by configuring OpenTracker properly, tracking data distribution (usually done on a separate tracking server) is independent of all other aspects.

Each Construct3D instance can be configured in multiple ways by defining the number of users, its associated resources, specifying application retrieval method (by distribution as slave or by file input as master) and tracking data obtaining strategy.

Further on a central and persistent Construct3D service can be established as a background process without the need of directly associated users and rendering output. This allows joining and leaving a persistent Construct3D learning experiment at any time. In contrast to this, dynamically migrating Construct3D application hosts with directly associated users and rendering tasks is also easily possible without difficult configuration effort, as contacting the session manager performs all bootstrapping.

6.1. Future work

Regarding the technical aspects we omitted the fact that our implementation of DIV supports using multicast UDP and TCP connections simultaneously in a hybrid network configuration. For example a simple hybrid network setup could consist of two local networks with multicast support (e.g. two local school networks or two university networks) which are connected using a reliable TCP connection. Extensive tests with hybrid network configurations are planned since they allow more efficient distribution between multicast enabled subnets.

Since the early evaluation in 2005 no further user studies have been conducted with distributed Construct3D. In order to simulate real classroom conditions, large scale testing of our implementation with a large number of client PCs (> 15) needs to be done. It should ideally be coupled with a large scale evaluation with high-school students.

7. Acknowledgements

The authors thank all participants of evaluations as well as other teachers and students for testing and giving useful feedback. Part of this research was funded by the Austrian Science Fund (FWF) contract P16803 and by the EU IST project Lab@Future (IST-2001-34204).

References

[CNI*03] CARROLL J. M., NEALE D. C., ISENHOUR P. L., ROSSON M. B., MCCRICKARD D. S.: Notification and awareness: synchronizing task-oriented collaborative activity. *International Journal of Human-Computer Studies 58* (2003), 605–632.

[Csi06] CSISINKO M.: *Long Distance Distribution of VR and AR Applications*. Master's thesis, 2006.

[DSL96] DEDE C., SALZMAN M. C., LOFTIN R. B.: Sciencespace: Virtual realities for learning complex and abstract scientific concepts. *Proceedings of IEEE VRAIS '96* (1996), 246–252.

[Eng87] ENGESTRÖM Y.: *Learning by Expanding: An Activity-Theoretical Approach to Developmental Research*. Helsinki: Orienta-Konsultit Oy, Finland, 1987.

[Eng99] ENGESTRÖM Y.: Activity theory and individual and social transformation. In *Perspectives on Activity Theory*, Engeström Y., Miettinen R., Punamäki R.-L., (Eds.). Cambridge University Press, UK, 1999.

[Eri94] ERIKSSON H.: MBONE: the multicast backbone. *Commun. ACM 37*, 8 (1994), 54–60.

[Hes01] HESINA G.: *Distributed Collaborative Augmented Reality*. PhD thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Austria, 2001.

[Kau04] KAUFMANN H.: *Geometry Education with Augmented Reality.* PhD thesis, Vienna University of Technology, Vienna, 2004.

[KS03] KAUFMANN H., SCHMALSTIEG D.: Mathematics and geometry education with collaborative augmented reality. *Computers & Graphics 27*, 3 (2003), 339–345.

[KS06] KAUFMANN H., SCHMALSTIEG D.: Designing immersive virtual reality for geometry education. In *Proceedings of IEEE Virtual Reality Conference 2006* (Alexandria, Virginia, USA, 2006), pp. 51–58.

[Man01] MANTOVANI F.: Vr learning: Potential and challenges for the use of 3d environments in education and training. In *Towards CyberPsychology: Mind, Cognitions and Society in the Internet Age*, Riva G., Galimberti C., (Eds.). IOS Press, Amsterdam, 2001.

[MF98] MACINTYRE B., FEINER S.: A distributed 3D graphics library. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1998), ACM Press, pp. 361–370.

[NLSG03] NAEF M., LAMBORAY E., STAADT O., GROSS M.: The blue-c distributed scene graph. In *EGVE '03: Proceedings of the workshop on Virtual environments 2003* (New York, USA, 2003), ACM Press, pp. 125–133.

[Osb97] OSBERG K. M.: *Constructivism in practice: the case for meaning-making in the virtual world*. PhD thesis, University of Washington, 1997.

[Peč02] PEČIVA J.: *Development of a Distributed Scene Toolkit Based on Open Inventor*. Master's thesis, 2002.

[Prü97] PRÜMPER J.: Der Benutzungsfragebogen ISONORM 9241/10: Ergebnisse zur Reliabilität und Validität. In *Software-Ergonomie '97*, Liskowsky R., (Ed.). Stuttgart, 1997.

[RS01] REITMAYR G., SCHMALSTIEG D.: An open software architecture for virtual reality interaction. In *VRST '01: Proceedings of the ACM symposium on Virtual reality software and technology* (New York, USA, 2001), ACM Press, pp. 47–54.

[SFH*02] SCHMALSTIEG D., FUHRMANN A., HESINA G., SZALAVARI Z. S., ENCARNACAO L. M., GERVAUTZ M., PURGATHOFER W.: The studierstube augmented reality project. *Presence - Teleoperators and Virtual Environments 11*, 1 (Feb 2002), 33–54.

[SG97] SZALAVARI Z. S., GERVAUTZ M.: The personal interaction panel - a two-handed interface for augmented reality. *Computer Graphics Forum 16*, 3 (1997), 335–346.

[SRH03] SCHMALSTIEG D., REITMAYR G., HESINA G.: Distributed applications for collaborative three-dimensional workspaces. *Presence - Teleoperators and Virtual Environments 12*, 1 (Feb 2003), 52–67.

[TGG04] TOTTER A., GRUND S., GROTE G.: A methodological approach to the evaluation of e-learning technologies. *Proceedings of ED-MEDIA, World Conference on Educational Multimedia, Hypermedia & Telecommunications* (Lugano, Switzerland, 2004), 4392–4397.

[THS*01] TAYLOR R. M., HUDSON T. C., SEEGER A., WEBER H., JULIANO J., HELSER A. T.: VRPN: a device-independent, network-transparent VR peripheral system. In *VRST '01 Proceedings* (New York, NY, USA, 2001), ACM Press, pp. 55–61.

[TN01] TAXEN G., NAEVE A.: Cybermath: Exploring open issues in vr-based learning. *Proceedings SIGGRAPH 2001 Educators Program* (2001), 49–51.

[Tra99] TRAMBEREND H.: Avocado: A Distributed Virtual Reality Framework. In *VR '99: Proceedings of the IEEE Virtual Reality* (Washington, DC, USA, 1999), IEEE Computer Society, p. 14.

[WB92] WINN W., BRICKEN M.: Designing virtual worlds for use in mathematics education: The example of experiential algebra. *Educational Technology 32*, 12 (1992), 12–19.

[Win93] WINN W.: A conceptual basis for educational applications of virtual reality. *Technical Report 93-9* (1993).

[YN04] YEH A., NASON R.: Vrmath: A 3d microworld for learning 3d geometry. In *Proceedings World Conference on Educational Multimedia, Hypermedia & Telecommunications* (Lugano, Switzerland, 2004).