The MaDViWorld Project

An Attempt to Apply a Collaborative Virtual World Paradigm to the Internet

Patrik FUHRER
Department of Informatics, University of Fribourg
1700 Fribourg, Switzerland
patrik.fuhrer@unifr.ch

and

Jacques PASQUIER-ROCHA
Department of Informatics, University of Fribourg
1700 Fribourg, Switzerland
jacques.pasquier@unifr.ch

ABSTRACT

MaDViWorld is an object oriented software framework supporting the implementation of fully distributed virtual worlds on the Internet. While the World Wide Web proposes a document paradigm with HTTP servers containing documents consulted by users with the help of browser applications, MaDViWorld supports a much richer paradigm based on room servers hosting spaces populated by full-fledged objects, that avatar applications can activate, move and share transparently.

Within this context, the MaDViWorld project main goal is to provide its users with the appropriate software environment for creating all kinds of new collaborative objects and for sharing them transparently with others. The present paper illustrates this process with several examples from projects recently accomplished at the DIUF (Department of Informatics of the University of Fribourg, Switzerland).

Keywords: Virtual World, Collaborative Work, Mobile Objects and Distributed Software Framework

1. INTRODUCTION

The Software Engineering Group at the DIUF has developed an object oriented distributed framework supporting massively distributed virtual worlds, called MaDViWorld. The general concepts have been introduced in [11] and an insight of the global framework architecture is provided by [12]. The theoretical foundation as well as a complete description of more specific aspects such as the distributed events model, code mobility or security are extensively discussed in [9].

The goal of this paper is to illustrate how MaDViWorld technology can be used on the Internet in order to apply a much richer collaborative paradigm than the classical document one proposed by the World Wide Web (WWW). The paper is organized as follows. Section 2 reviews some of the main virtual world concepts such as subspaces, avatars and objects and further explains the paradigm shift between classical WWW browsing and MaDViWorld usage. Section 3 provides a sample of the objects that can be created within MaDViWorld. Finally, Section 4 wraps up the paper by describing future possible collaborative worlds based on MaDViWorld technology.

2. BASIC VIRTUAL WORLD CONCEPTS

For the further comprehension of this paper, the following four terms need to be briefly explained:

1. *Avatars* are the virtual representation of the users. Concretely, an avatar is a tool that allows a given user to move through the world, to interact with its inhabitants and objects and that lets the other users know where she is and what she is doing. Among people working on virtual reality and cyberspace interfaces (see [5, 18, 19]), the word Avatar is used to describe the "object" (icon, two or three-dimensional photo, design, picture or animation) representing the user in a shared virtual reality. In other words, an avatar is an instantiation of the user's body in the computerized medium. In text-based virtual realities, such as MUDs[1] and MOOs[2] (see [4, 17]), avatars consist of a short description which is displayed to the users whose avatars "look" at them.

[1]MUD stands for Multi User Dungeon.
[2]MOO is the acronym for MUD Object-Oriented.

2. In order to distinguish between near and distant elements, it is essential to divide the world into subspaces where the users might or might not enter and in which all interactions take place. Let us call such subspaces *rooms*.

3. Rooms are connected by *doors*, which an avatar can use for moving from one room to another.

4. *Objects* populate the rooms. They are not just simple passive data objects, but full-fledged objects (single or multi user) avatars can execute and share (e.g. games, whiteboards). Furthermore, in a distributed world, objects should be "physically" mobile, i.e. transparently movable from one room on a given server to another room hosted on a different machine. In MaDViWorld, mobility is either performed autonomously by the object itself or passively with the help of the avatar transporting it in her *bag*.

The conceptual model, that emerges from these considerations is illustrated below with the help of a simple typical scenario.

A Typical Scenario

The starting point is a virtual world composed of two rooms, R1 and R2, hosted on two different machines. Let us comment, step by step, the scenario illustrated by Figure 1.

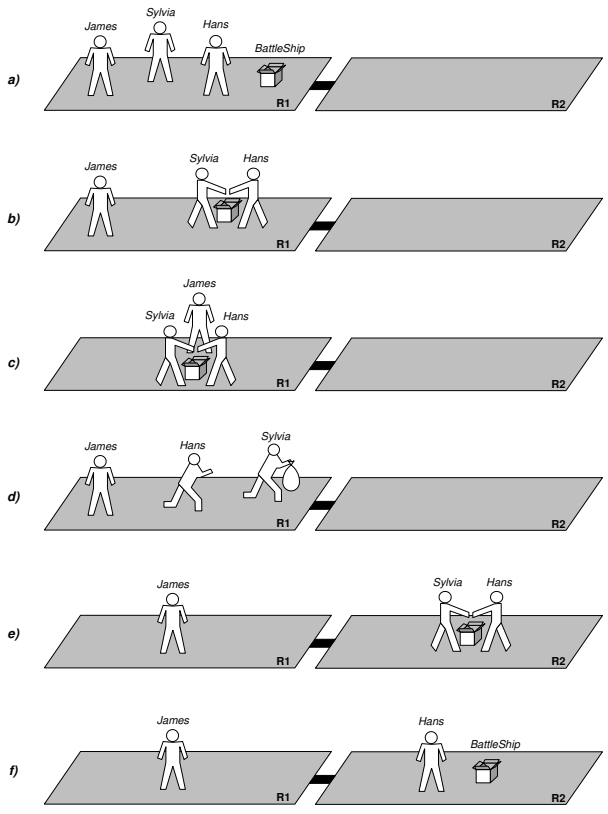

Fig. 1: A typical scenario in MaDViWorld

- Figure 1a): The virtual world is shared by three avatars: James, Sylvia and Hans, all present in the same room R1. There is a battleship game object in this room.
- Figure 1b): Sylvia and Hans both launch the battleship game and start playing it.
- Figure 1c): James also launches the battleship game. As it is a two players game, he becomes an observer of the game and can only watch how his two roommates play.
- Figure 1d): Sylvia and Hans decide to finish their game in room R2. Sylvia takes the battleship object and puts it in her bag.
- Figure 1e): Sylvia and Hans move to the empty room R2. Sylvia puts the game she had in her bag into the room. Then both Hans and Sylvia reactivate the game and go on from the point they stopped before. James is now alone in room R1.
- Figure 1f): The game is finished and Sylvia logged off the world. James and Hans are still inhabiting the world, each in a different room.

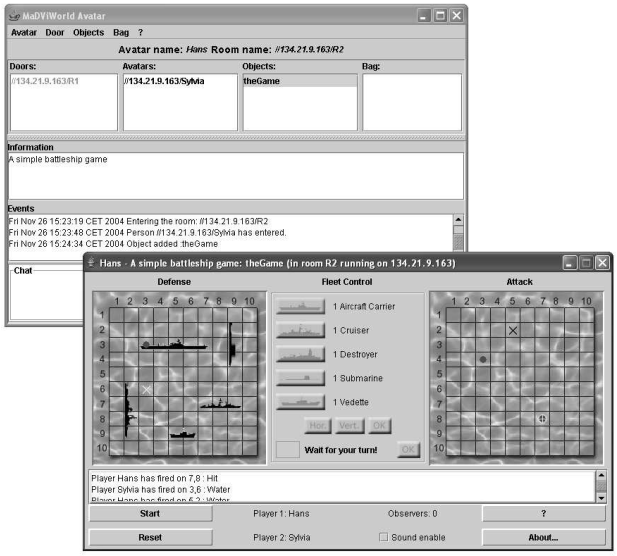

Fig. 2: Hans' avatar application and battleship object GUI

Although very simple, the preceding story reveals several interesting points:

- MaDViWorld's powerful remote event mechanism plays an important role at two levels in the scenario. On the one hand, thanks to it, the avatars are aware of their environment. James immediately knows that Sylvia and Hans left the room. Hans sees when Sylvia puts the battleship object in room R2 (see Figure 2). On the other hand, the event mechanism is used to update the graphical user interface of the objects. This allows each move to be displayed immediately on each logged avatar's board, player or observer.
- The battleship object has "physically" been carried from room R1 to room R2 by the avatar Sylvia. Note

that R2 is hosted by another machine than R1 and that the machine hosting R2 had no prior knowledge of this kind of object.

- The state of the game has not been lost during its transfer from R1 to R2.

Document versus Virtual World Paradigm

At this stage, it is worth comparing the virtual world paradigm just presented above with the document one usually applied when browsing the web:

- Within the *document paradigm*, documents, often active ones able to react to various user actions, are made available on one or several servers, and client applications (e.g., web browsers) can be used to interact with them. Typically, each user copies the documents onto her local machine and her interactions with them have no direct repercussions on the other connected users. In particular, a user never directly modifies the original document. The underlying metaphor is the one of a huge cross-referenced book where each user browses through the pages totally unaware of other users performing the same task at the same moment. All actions are asynchronous and, thus, there is no need for a central server to coordinate user interactions with the pages of the book or to take care of an event redistribution mechanism. The main advantage of this approach is that it allows a truly distributed architecture with thousands of http servers interconnected all over the world. If a crash occurs, only the pages hosted by the failed or the no longer reachable servers become momentarily unavailable. The whole system is extremely robust and, since the connection of new decentralized servers is always possible, there is no limit to its growth.

- Within the *virtual world paradigm*, multiple users and active objects interact in the same space and therefore have a direct impact on each other. Within such systems, if a user interacts with an object, the other connected users can see her and start a dialog with her. Moreover, it is possible for a user to modify some properties of the world and all the other users present in the same subspace must immediately be made aware of it. It is worth noting that applications of the virtual world paradigm range from simple textual chat to sophisticated 3D virtual worlds (e.g. [1]) used for military simulations.

Implementation

At the implementation level, systems based on a distributed[3] virtual world metaphor are clearly the most complex ones. Indeed, the users interact directly with the original objects of the system and the resulting events must be correctly synchronized and forwarded in order to maintain the consistency of the world. To face these issues, to support scalability when the virtual world grows very large and to allow for code mobility when objects are moving, virtual world developers have to choose carefully an appropriate software architecture. It is out of the scope of this paper to review the abundant literature on the subject (see [15, 14]) or to present in details the solution proposed by MaDViWorld. The interested reader is referred to [9] for an in depth presentation of the MaDViWorld software architecture and to [13, 8, 6, 20] for a discussion of other systems. It is however possible to summarize MaDViWorld 's mains choices as follows:

- *Room server applications* are set up on networked machines the same way as HTTP servers are for the web. A room server application allows for creating an arbitrary number of rooms with various properties on a given machine and for connecting them with others (including on different room servers) through doors.

- Active objects are implemented as *simple Java classes* with the only need of respecting a minimal set of conventions[4]. Furthermore, a user-friendly *setup wizard* application allows for installing newly programmed objects within a given room.

- *Avatars* are simple client applications, which can connect to a given room, see its contents and interact with its objects and other connected avatars. They basically play the same role as classical browsers within the World Wide Web and they do not need to present a sophisticated 3D interface.

3. MORE ON OBJECTS

The MaDViWorld framework provides a default implementation both for a simple avatar application (see Figure 2) and for the room server application. These two default applications allow for a given amount of customization from their users (e.g. by defining the rooms' access rights and security policy, by setting them up with one's own collection of objects or by connecting them through doors). It would even be possible for an experienced Java programmer to use the carefully designed hooks of the framework in order to extend or even to fully override the default implementation, introducing for example rooms with 2D or 3D representations.

It is, however, not the main goal of the MaDViWorld project to go into this direction. We truly believe that the actual implementation is sufficient in order to design exciting worlds at the important condition of disposing of a rich enough variety of objects to populate them. This is the reason why we concentrated on facilitating as much as possible the process of programming new types of object with the ultimate goal of instigating a rich community of object creators. Furthermore, in order to bootstrap this process, we launched a series of student projects[5] with the only requirement of validating

[3]In the context of virtual worlds, "distributed" means that the architecture must not be limited to a single central server containing the whole virtual world and guaranteeing its consistency with many clients connected to it. It is imperative that distinct clusters of subspaces might be distributed on separate servers for scalability purpose.

[4]The interested reader is referred to the MaDViWorld's official web site [10], where she can both download the code and consult a step by step cookbook guide to programming MaDViWorld objects.

[5]Bachelor or Master level projects realized at the DIUF (see [10]).

the framework by programming new "useful" objects. The next subsections briefly present some of them.

Resource Sharing Objects

The objects are executed either on the machine hosting their containing room or on the one where the avatar application is running. This allows for *resource sharing*. Objects needing a lot of computing power and memory are put in a room hosted by a powerful computer and they are remotely controlled by their thin GUIs launched by avatar clients. A little example illustrating this feature is the fibonacci number calculator. Other ones can easily been imagined, for example from mathematical topics such as fractal calculation, cryptography or linear programming solvers.

Collaborative Objects

The MaDViWorld framework offers all what is needed in order to build *collaborative objects*. Indeed, objects can easily be shared by several users and events transparently broadcasted. This allows for the creation of a large variety of objects supporting collaboration among the virtual world users. These objects range from simple shared whiteboards to sophisticated collaborative editors and "chat" utilities. The whiteboard object is an illustrative example from the MaDViWorld programming cookbook guide, while prototypes of a simple collaborative editor and of a powerful chat object (see Figure 3) have been realized in two separate projects.

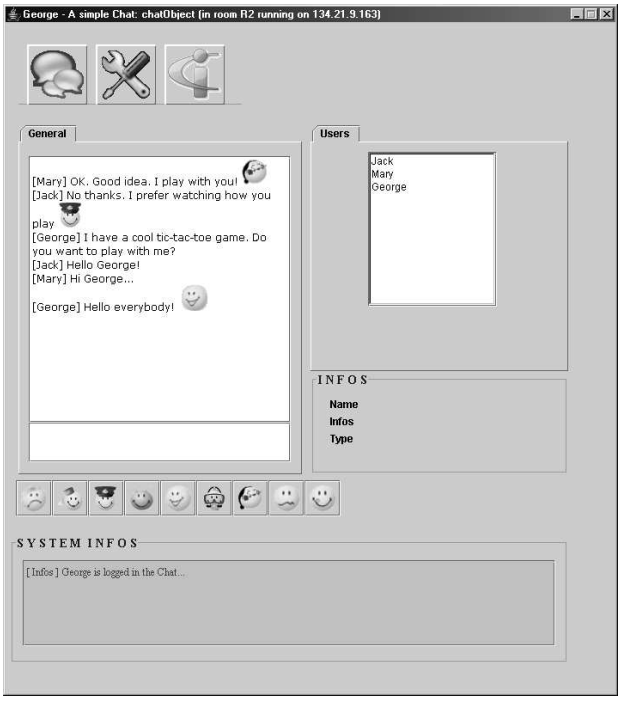

Fig. 3: A chat object

Multi-player games are also part of this category of objects. Existing examples of multi-user games are the "battleship" game (see Figure 2), the "tic-tac-toe" game, the "minesweeper" game[6] and even a complex "Metal Panic" game composed of three complementary *inter-communicating objects* : robot factories, customizable fighting robots and fighting arenas.

It would also be possible to imagine objects which would sense their environments and adapt their states in order to anticipate the needs of their users, e.g. a whiteboard which would adapt its size to the number of avatars present in a given room.

Inter-Communicating Objects

The remote event mechanism model can also be used in order to make objects communicate with each other. A possible application consists in producing so called "social" objects. For example, one can create a virtual pets community. The avatars owning these pets have to play with them, clean or feed them in order to keep them healthy. If a member of the community dies, the other pets living in the same room are affected by the death of their friend and their "life capital" decreases. The GUI of such an object is illustrated by Figure 5.

Fig. 4: A virtual pet in MaDViWorld

Other applications of the communication between objects are the robots, factories and arenas of the "Metal Panic" game, and MadTunes, an audio player accessing MusicRack objects containing several music files.

Agent Objects

The MaDViWorld framework also allows for the creation of so-called *mobile software agents* (see [7, 3]). At the software engineering level (design, programming and especially debugging), these objects are some of the most difficult to deal with. Nevertheless, two prototypical agents have successfully been developed with the help of the framework fa-

[6]Essentially a single-user game. It might, however, range in the collaborative objects category if one considers the avatars watching how someone else plays.

cilities: The first one is an agent called "Explorer" that draws a sophisticated interactive map[7] of a given virtual world by visiting all its rooms. The second one is an agent called "Matchmaker" that fixes meeting with other agents of the same type on behalf of their respective owners (see Figure 5).

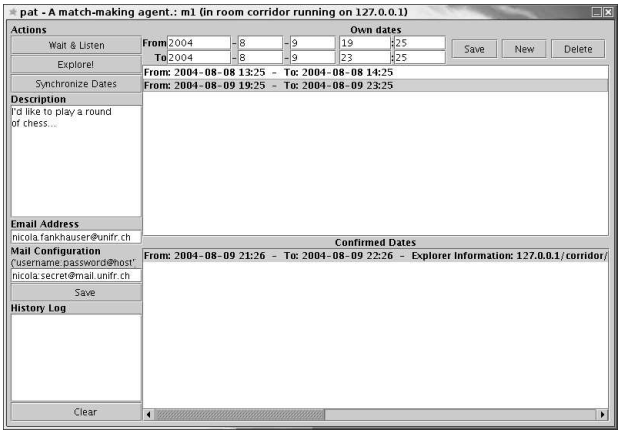

Fig. 5: Screenshot of a "Matchmaker" agent

Summary

Our experience within the various projects partially described in the preceding subsections proved that it is rather simple for an average Java programmer using the framework to develop her own objects with the freedom of deciding if her new object will:

- be stateful or stateless, determining if the internal state of the object is carried around when the objects moves or not;
- be single- or multi-user;
- take advantage of the distributed event mechanism in order to "inter-communicate" with other objects or not;
- take advantage of their intrinsic mobility in order to behave as mobile agents or not;
- be rather specific (for example a given game) or very generic (e.g. a chat, a witheboard or a collaborative editor, which could then be installed by default in every room).

4. CONCLUSION

The current version of MaDViWorld is a fully functional one and has been extensively tested within various student projects. Furthermore, a small set of generic objects are already available. The next step would be to integrate the work already done within a coherent and "interesting" world. Two possible candidates are sketched below: the first one ranges in the area of entertainment and the second one deals with e-learning.

[7]The map appears as a graph, where the vertices express either the room server hosts, the rooms, the connected avatars or the objects in the rooms, while the edges represent either inclusion (e.g. an object in a room) or connection relationships (e.g. rooms linked by a door).

Gameworld

This virtual environment consists of a set of rooms full of active collaborative game objects, ranging from single user arcade games to sophisticated multi-user ones (card games for instance). After having paid a fee, the users are allowed to visit the rooms; to watch other users play; to try out some demo versions of the games; or even to join a game and to exchange their impressions about it. Later, if she is interested, a user can even copy a given game object onto her own machine by getting the right to clone it. A slightly modified version of this world would be to replace the cloneable game objects by active pieces of art that would be unique in the sense that one could only move them around, not copy them.

Eduworld

A more ambitious project is to build up a *distributed learning environment* on the top of the MaDViWorld framework. While Figure 6 sketches the conceptual model of such a world, its key elements are enumerated below.

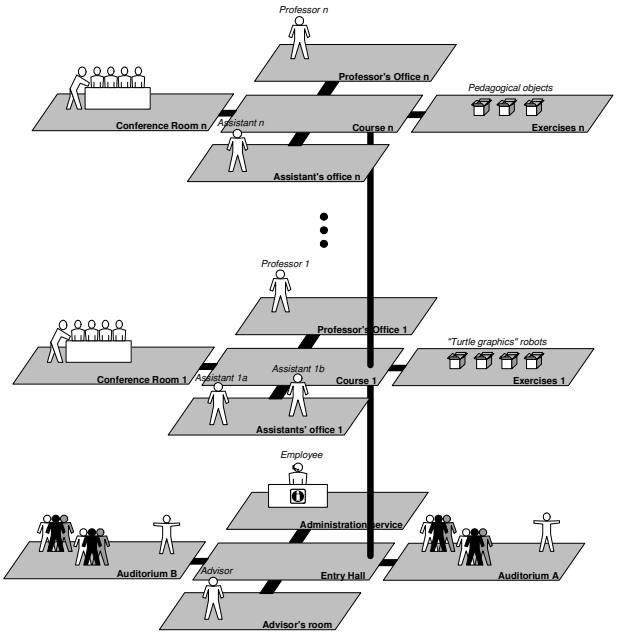

Fig. 6: Distributed learning environment conceptual model

- *Individual professors' offices* are used in order to receive students for private discussions. We propose to physically decentralize them on the professors' private machines.
- *Assistants' offices* are rooms used by the assistants of a given professor in order to receive individual students for questioning about their on-going homework. The functions of these rooms are close to the former ones and we also propose to decentralize them.
- *Conference rooms* are associated to a professor's group and are used by both the professor and his assistants

in order to have an open discussion with several students at once. They can also serve for more classical *ex cathedra* courses. These rooms can either be decentralized on a machine associated with a given professor's group or put on a larger department's server.

- *Exercises rooms* are the most interesting ones, since they contain the *active pedagogical objects* associated with a given course. For instance, programmable drawing robots could be used in order to teach algorithmic concepts. This idea is analogous to the turtle graphics methodology adopted by Logo [16, 2]. Adapted to a virtual world environment such a learning strategy would lead to the following scenario. Each student clones the 'exercise of the day' robot and takes it into her virtual office, running on her own physical machine. She then tries to instruct the robot to do a given drawing. Once she is finished, the student puts her programmed robot in another room for correction (the assistants' office for instance). A reasonable solution is to put these rooms on the same server as the conference ones. They will not overload this machine, since the real work will always take place on the students' individual machines.
- Administrative rooms provide various central services (registration, accreditation, etc.) and would typically run on a larger department (or even university) server.

It is our hope that such worlds will be built in the near future.

5. REFERENCES

[1] Paradise project web site. [online]. http://www.dsg.stanford.edu/paradise.html (accessed January 14, 2004).

[2] H. Abelson and A. A. diSessa. *Turtle Geometry: The Computer as a Medium for Exploring Mathematics*. MIT Press, September 1986.

[3] J. M. Bradshaw. *Software Agents*. AAAI Press, 1997.

[4] L. P. Burka. The MUDdex. [online], 1993. http://www.linnaean.org/~lpb/muddex/ (accessed November 26, 2004).

[5] J.-C. H. (ed.). *Virtual Worlds: Synthetic Universes, Digital Life, and Complexity*. Perseus Books, Reading, Massachusetts, USA, 1999.

[6] V. N. et al. The COVEN project: Exploring applicative, technical, and usage dimensions of collaborative virtual environments. *Presence*, 8(2):218–236, 1999.

[7] S. Franklin and A. Graesser. Is it an agent, or just a program?: A taxonomy for autonomous agents. In *Proceedings of the Third International Workshop on Agent Theories, Architectures and Languages*. Springer-Verlag, 1996.

[8] E. Frécon and M. Stenius. DIVE: A scalable network architecture for distributed virtual environments. *Distributed Systems Engineering*, 5(3):91–100, September 1998.

[9] P. Fuhrer. *Distributed Virtual Worlds - Abstract Model and Design of the MaDViWorld Software Framework*. PhD thesis, Department of Informatics, University of Fribourg, Switzerland, Nr. 1458, September 2004.

[10] P. Fuhrer. MaDViWorld (Massively Distributed Virtual Worlds). [online], 2004. http://diuf.unifr.ch/softeng/projects/madviworld/ (accessed November 26, 2004).

[11] P. Fuhrer, G. K. Mostéfaoui, and J. Pasquier-Rocha. MaDViWorld : a software framework for massively distributed virtual worlds. *Software - Practice And Experience*, 32(7):645–668, June 2002.

[12] P. Fuhrer and J. Pasquier-Rocha. Massively distributed virtual worlds: A framework approach. In E. A. Nicolas Guelfi and G. Reggio, editors, *Scientific Engineering for Distributed Java Applications*, volume 2604 of *Lecture Notes in Computer Science*, pages 111–121. International Workshop, FIDJI 2002 Luxembourg-Kirchberg, Luxembourg, November 2002, Springer-Verlag, March 2003.

[13] C. M. Greenhalgh. Awareness-based communication management in the massive systems. *Distributed Systems Engineering*, 5(3):129–137, September 1998.

[14] R. Kazman. Load balancing, latency management and separation of concerns in a distributed virtual world. In A. Y. Zomaya, editor, *Parallel Computing: Paradigms and Applications*. International Thomson Publishing, November 1995.

[15] K. L. Morse, L. Bic, and M. Dillencourt. Interest management in large-scale virtual environments. *Presence*, 9(1):52–68, 2000.

[16] S. A. Papert. *Mindstorms: Children, Computers and Powerful Ideas*. Basic Books, 2nd edition, March 1999.

[17] E. Reid. *Cultural Formations in Text-Based Virtual Realities*. Masters thesis, English Department, University of Melbourne, January 1994.

[18] S. Singhal and M. Zyda. *Networked Virtual Environments: Design and Implementation*. Addison-Wesley, 1999.

[19] J. Smed, T. Kaukoranta, and H. Hakonen. A review on networking and multiplayer computer games. Technical Report Technical Report 454, Turku Centre for Computer Science, April 2002.

[20] R. C. Waters, D. B. Anderson, J. W. Barrus, D. C. Brogan, M. A. Casey, S. G. McKeown, T. Nitta, I. B. Sterns, and W. S. Yerazunis. Diamond park and spline: Social virtual reality with 3D animation, spoken interaction and runtime extendability. *Presence*, 6(4):461–481, August 1997.