An Improved Object Interaction Framework for Dynamic Collaborative Virtual Environments

R.Vinob Chander
Sree Nidhi Institute of Science and Technology
Yamnampet, Ghatkesar, Hyderabad, India

Abstract
This paper attempts to achieve better realistic dynamic interactions for collaborative virtual environments. Towards this, we create a general object interaction framework for dynamic interaction in physically realistic collaborative virtual environments. The realism of the interactive world is increased by using Newton Game Dynamics, a rigid body simulator that is open-source. Also, Inverse Kinematics is used to increase the interaction possibilities, and on-the-fly creation of new interactions. Further, we make use of separate servers for efficient space structuring and also use these multiple servers with multicasting for efficient distribution of both the interactive objects and the dynamic avatar interactions, keeping the network load as low as possible.

Keywords
CVEs, animation, simulation, inverse kinematics, interaction

1. INTRODUCTION

Trends towards networked applications and Computer Supported Collaborative Work, together with a wide interest for graphical systems and Virtual Environments, have in the recent years raised interest for research in the field of Collaborative Virtual Environments (CVEs).
CVEs are systems that allow multiple geographically distant users to share a common threedimensional Virtual Environment (VE). CVEs are a powerful tool for communication and collaboration, with potential applications ranging from entertainment and teleshopping to engineering and medicine. Therefore it is not surprising that in the recent years we have seen active research on this topic in both academic and industrial research establishments.

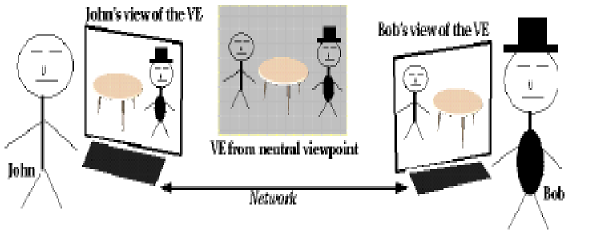

Figure 1 - Principles of Networked Collaborative Virtual Environments

Manuscript received September 5, 2010
Manuscript revised September 20, 2010

The concept of a CVE is fairly simple (Figure 1). Two or more users can view the Virtual Environment (VE) on their work stations. Each work station has a local copy of the VE. As the actions occur on one work station, they are propagated through the network to other work stations in order to keep allcopies of VE synchronized. The users become themselves part of the VE, placed within the VE at the position of their view point and represented by a graphical embodiment called avatar. This allows the users to see each other and visualize the other users' actions.
Despite the fairly simple concept, the design of NCVE systems involves a complex interaction of several domains of Computer Science:
− networking, involving communication of various types of data with varying requirements in terms of bitrate, error resilience and latency
− virtual environment simulation, involving visual data base management and rendering techniques with real time optimizations
− human-computer interaction, involving support of different devices and paradigms
− virtual human simulation, involving real time animation/deformation of bodies and faces
− artificial intelligence (in case autonomous virtual humans are involved), involving decision-making processes and autonomous behaviors.
Also, the state of such a CVE should be synchronized and distributed to all participants in order to give the users the illusion of being located in the same place at the same time [2].
In this undergoing work, we try to present a dynamic VE platform as in [1],
where every object can be interacted with by the users. It has the advantage of all objects to be able to interact with every object. This concept is in turn similar to the smart object approach [3]. Furthermore, our approach makes use of separate servers with multicasting, eliminating the server bottleneck problem in [1]. Realism is increased further between avatar interactions by using a rigid body simulator, which many of the existing CVE systems don't. And to increase on-the-fly creation of new interactions we make use of inverse kinematics and a different way of

transmitting the animation data when compared to that of [1]. Finally, we plan to test out system by integrating it into the DIVE framework [4].

2 Landmark NVCE Systems

In the following sections we present several representative NCVE systems. The systems presented here are either representative of a particular system category (academic, commercial, game) or they introduced novel approaches and developments at the time of their conception [5].

2.1 NPSNET

NPSNET is a networked virtual environment developed at the Computer Science Department of the Naval Postgraduate School (NPS) in Monterey, California. It is developed specifically for large-scale military simulations. It is one of the first NCVE systems and has known several generations of new and improved versions. NPSNET can be used to simulate an air, ground, nautical (surface or submersible) or virtual vehicle, as well as human subjects. A virtual vehicle, or stealth vehicle, is a vehicle that can navigate in the virtual world but has no graphical representation and is therefore not seen by others. The standard user interface devices for navigation include a flight control system (throttle and stick), a six degree of freedom SpaceBall, and/or a keyboard. The system models movement on the surface of the earth (land or sea), below the surface of the sea and in the atmosphere. Other entities in the simulation are controlled by users on other workstations, who can either be human participants, rule-based autonomous entities, or entities with scripted behavior. The virtual environment is populated not only by users' vehicles/bodies, but also by other static and dynamic objects that can produce movements and audio/visual effects.
NPSNET uses the Distributed Interactive Simulation (DIS2.03) protocol [IEEE 93] for application level communication among independently developed simulators (e.g.legacy aircraft simulators, constructive models, and real field instrumented vehicles).

2.2 DIVE

DIVE (Distributed Interactive Virtual Environment) is developed at the Swedish Institute of Computer Science. The DIVE system is a tool kit for building distributed VR applications in a heterogeneous network environment. It is another early system that continues to be developed and improved over the years. The DIVE run-time environment consists of a set of communicating processes, running on nodes distributed within a Local Area Network (LAN) or Wide Area Network (WAN). The processes, representing either human users or autonomous applications, have access to a number of databases, which they update concurrently. Each database contains a number of descriptions of graphical objects that together constitute a virtual world. Objects can be added or modified dynamically, and concurrently using a distributed locking mechanism. Multicast protocols are used for the communication simulating a large shared memory for a process group through the network.

2.3 BRICKNET

The BrickNet toolkit, developed at the National University of Singapore, provides functionalities geared towards enabling faster and easier creation of networked virtual worlds. It eliminates the need for the developer to learn about low level graphics, device handling and network programming by providing higher level support for graphical, behavioral and network modeling of virtual worlds. BrickNet provides the developer with a "virtual world shell" which is customized by populating it with objects of interest, by modifying its behavioral properties and by specifying the objects' network behavior. This enables the developer to quickly create networked virtual worlds. BrickNet applies the multiple server network topology Servers, distributed over network, communicate with one another to provide a service to the client without itsexplicit knowledge of multiple servers or inter-server communication. BrickNet introduces an object sharing strategy which sets it apart from the classic NCVE mindset. Instead of all users sharing the same virtual world, in BrickNet each user controls their own virtual world with a set of objects of their choice. They can then expose these objects to the others and share them, or choose to keep them private. The user can request to share other users' objects providing they are exposed. So, rather than a single shared environment, BrickNet is a set of "overlapping" user-owned environments that share certain segments as negotiated between the users. BrickNet does not incorporate any user representation, so the users are body-less in the virtual environment and their presence is manifested only implicitly through their actions on the objects.

2.4 VLNET

Virtual Life Network (VLNET) [Capin99, Pandzic97] is a client/server NCVE system that introduced high-level Virtual Humans for user representation and communication. The system also supports a modular architecture concept allowing extended system capabilities using plug-ins or to connect the system to complete applications, thus giving a graphical, networked front-end to those applications. In particular, gestural and facial communication are supported. The facial communication includes live video analysis for tracking of facial features, video texturing with face region tracking, predefined emotions and lip synchronization to the audio signal.

2.5 BLAXXUN COMMUNITY PLATFORM

This system from Blaxxun Interactive Inc. [Blaxxun] is the first commercialy successful NCVE system. It is used to create on-line 3D communities where people can meet, chat or even trade. Through a set of modules, the system supports a rich set of functionalities such as pre-programmed agents, personal homes that can be set up by participants, personalized avatars, access rights control, text-to-speech, etc. It is a client/server system. The client is a free of charge and comes in form of a Web browser plug-in, and the commercial benefit is obtained from customers installing the servers for their users. Blaxxun claims that the Community Server can support more then 20000 users simultaneously with a T3 (45Mbits) connection; these numbers are reduced by the server's CPU capacity. The company claims that a Pentium 500 machine running Linux supports approx. 2000 simultaneous users. It is however less clear how each client would cope with information from so many other clients.

2.6 DOOM

Doom is a computer game that reached tremendous popularity due largely to the networked aspect: groups of players can share the game through the network. In the first version the network traffic was not at all optimized and created large network traffic on the LANs where the game was played. It is remarkable that approx. 15 million copies of the shareware version of Doom were downloaded, and approx. 250000 copies of the full product were sold. Doom has been criticized for its content (basically, it is shooting monsters) but it is probably the most-used example of a NCVE system!

2.7 Avatar Evolution

Many of the initial CVEs used very simple graphical primitives to represent user embodiment. RING [6] used yellow spheres with green orientation vectors. DIVE used blockies (figure 2) to represent connected users. Virtual humans were introduced with NPSNET [7] (figure2) system. And finally, current CVE systems focus on virtual community building. (figure 2)

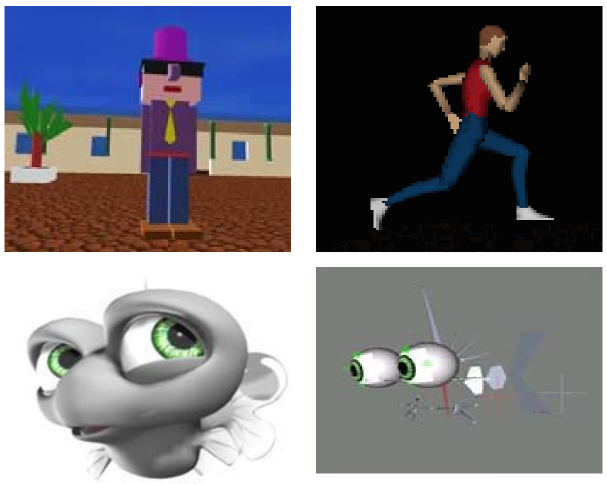
Figure 2 – Avatar Examples

3. Our Approach

We make use of the object interaction framework similar to the one described in [1]. The major differences from the referred framework are as follows. Object properties now don't have any actions defined on them. Action descriptions are removed and are part of the object behaviors. Furthermore, actions along with behaviors are defined in a scripting language.(figure 4)
The dynamism is identical with that of [1], handled by the interaction layer.
- We make use of Newton Game Dynamics for achieving realistic movements, rather than the Novodex physics engine. Newton Game Dynamics is open-source and can be used commercially, unlike the novodex engine. This saves some cost overhead in purchasing a commercial rigid body simulator.
- For collaboration and network setup we make use of multiple servers (figure 3) in contrast to a centralized one used in [1]. This way the
server bottleneck can be reduced to a great extent. Also, the clients are given illusion of the existence of a single server (although it is distributed). For the actual distribution of data, initially(at startup) clients connect to the server(s) using reliable TCP connection. The system thereafter falls back to UDP to keep the different clients synchronized.
Furthermore to reduce the network load, the server sends state updates to all interested clients at once using multicasting. Multicast transmission delivers source traffic to multiple receivers without adding any additional burden on the

source or the receivers while using the least amount of bandwidth of any competing network

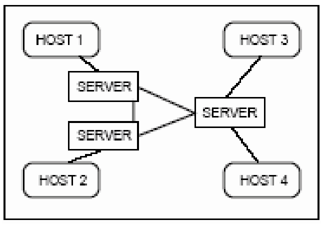

Figure 3 - Multiple Servers

technology. Additionally TCP keep- alive messages containing positional data are also regularly sent to ensure that, when many UDP messages do get lost, there is still some synchronization.

- On-the-fly creation of new instances and dynamic interactions are achieved using inverse kinematics as in [1]. But, in our approach during transmission of the animation data, apart from the position and orientation information, all the transformational information (like joints of the IK chain) are also sent so that the clients need not have to calculate the transformations of the intermediate joints in the chain. This way the clients may be) have to wait for some extra time, but they have the advantage of not

having to decode again the animation information. We also assume that the traffic is low (not during the peak hours) and that the clients have reasonable bandwidth speeds.

- We show the general object interaction platform based on the one proposed by Pieter [1]. The user sends interacts with the object controller that send the commands the objects perform. The interactions not only include avatar – object interaction, but also object-object interactions and hence independent of the application

The main advantage of this approach(Figure4) is that all the information needed to interact with an object is located at the object level instead of at the application level. As a result, the objects, their parts, and their behaviors are easy to modify, even at run time. This could be done by the application itself or by the application user, provided that the application developer provides support for it to the user. Object commands, actions, and behaviors can be added, removed or changed at run time by having these requests called by the interactive object manager , a part of the interaction layer that contains the objects. Furthermore, object and part constraints can be changed as well via the object manager. The interactive objects are also easy to reuse for the construction of other interactive objects. New objects unknown to the application and to objects that were already present, can me introduced in the application at any time and all other objects will be able to interact with them instantaneously. Even at the networking level we can use the interaction properties of the objects. The object format basically consists of three important parts: the object properties, the interaction properties, and the object's behaviors. Here, the object properties consist of a fill description of the object's structure and identity: its identifier, name, its parts with their graphical and physical data, how these parts are related to each other, and a short textual description of the object. However, the physical features associated with the object is represented separately in the world parameters. This would be easy for the rigid body simulator to identify the physical parameters easily associated with the object. Whereas in [1] some of the physical parameters are associated with the object itself. Another improvement is the removal of action descriptions from the object properties and associating it with the behavioral properties.

4. Results

The techniques proposed in this paper is currently being integrated into the DIVE framework. I am currently are performing tests on a LAN using Pentium IV 1.7 GHZ Computers with 512 MB RAM as clients and server. Computers with 512 MB RAM as clients and server.

5 Conclusions and Future Work

I have proposed an efficient way for achieving dynamic interactions in physically realistic collaborative virtual environments. I have modified the object interaction framework of [1] to get this efficiency. Basically, I eliminated the action descriptions from the object properties to reduce the object size. Further, I made use of the Newton Game Dynamics to achieve physical realism. Also, I made use of inverse kinematics to create on-the-fly creation of interactions. And multiple servers are used to represent the world and interaction layers in order to avoid the memory bottleneck problem that arise in using a centralized one. I plan to extend the work so as to achieve more realistic communication making use of gestures and facial animation. DIVE has some predefine gestures in its framework. But this is not the case with real worlds. I try to create gestures that are dynamic as extension to this work

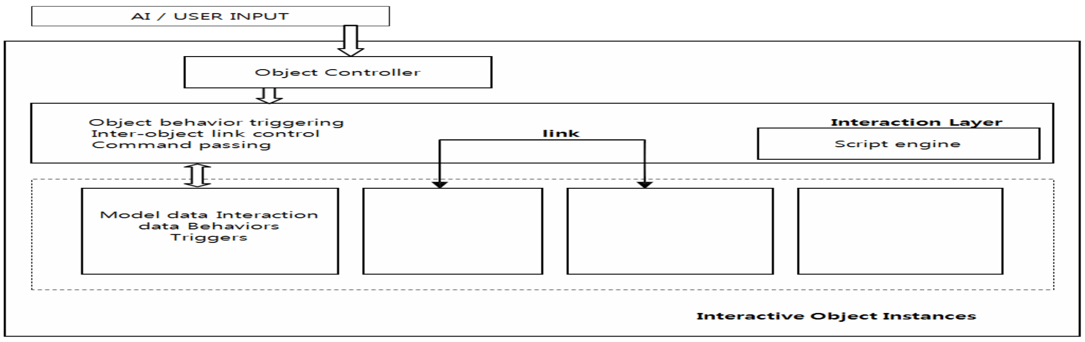
AI / USER INPUT
Object Controller
Object behavior triggering
Inter-object link control
Command passing
link
Interaction Layer
Script engine
Model data Interaction data Behaviors Triggers
Interactive Object Instances

Figure 4 - The interaction platform setup

6. References

[1] Pieter Jorisson, Maarten Wijnants, and Wim Lamotte,Dynamic Interactions in Physically Realistic Collaborative Virtual Environments, IEEE transactions on visualization and computer graphics, pp. 649-660, 2005.
[2] S Singhal and M.Zyda, Networked Virtual Environments:Design and Implementation, Addison-Wesley, 1999.
[3] M.Kallmann and D.Thalmann, "Modeling Objects for Interactive Tasks," proc. EGCAS '98 – Ninth Eurographics Workshop Animation and Simulation,1998.
[4] O.Hagsand, "Interactive Multiuser VEs in the DIVE system" IEEE Multimedia, pp. 30-39, spring 1996.
[5] Igor S. Pandzic et.al " Trends in Networked Collaborative Virtual Environments".
[6] T.Funkhouser, "RING: A client-server system for multi-user virtual Environments," proc. ACM 1995 Symp Interactive 3D Graphics, pp. 2-9, 2000.
[7] M.Macedonia et.al. "Exploiting Reality with multicast groups: A network architecture for arge scale virtual environments," Proc. IEEE Virtual Reality Ann. Int'l Symp., pp. 2-10,1995.
[8] http://secondlife.com
[9] www.there.com
[10] www.havok.com
[11] www.novodex.com
[12] www.ode.org
[13] www.physicsengine.com

R. Vinob chander is currently with Sree Nidhi Institute of Science and Technology, working as Associate Professor in the Department of IT. He has nine years of teaching experience. He did his BE(CSE) from Gulbarga University and ME(CSE) from Vinayaka Missions' University. He is also a Life Member of ISTE and IETE. His areas of research include graphics, animation, human computer interaction , collaborative virtual environments, and web services. He also has a couple of national and international conference papers to his credit.