Using Collaborative Interactive Objects and Animation to Enable Dynamic Interactions in Collaborative Virtual Environments

Pieter Jorissen Maarten Wijnants Wim Lamotte

Expertise Centre for Digital Media
Limburgs Universitair Centrum
Universitaire Campus, B-3590 Diepenbeek, Belgium

{pieter.jorissen, maarten.wijnants, wim.lamotte}@luc.ac.be

Abstract

This work introduces a new general object interaction scheme for dynamic collaborative virtual environments. The idea is to construct a world using only collaborative interactive objects that contain their own interaction information. As a result, the object interactions are application independent and only a single scheme is required to handle all interactions in the virtual world. Furthermore, we present a new and efficient way for human users to dynamically interact in the virtual world through their avatar. In particular, we show how inverse kinematics can be used to increase the interaction possibilities and realism in collaborative virtual environments. This will normally also result in a higher feeling of immersion for connected users. For both the collaborative interaction objects approach and the distribution of the dynamic avatar interactions, we try to keep the network load as low as possible. Finally, we demonstrate our techniques by incorporating them into the ALVIC framework.

Keywords: interaction, distributed virtual environment, collaboration, animation, inverse kinematics

1 Introduction

Creating immersive environments has been a research topic for several years now. Experiencing totally immersive and realistic virtual worlds in which we can interact like in the real world is still not possible. Mixing navigation and meaningful interaction within these Virtual Environments (VE) remains a difficult problem in research. One of the most important parts of computer simulation applications are the actual 3D interactions with objects within these synthetic worlds. A subject closely related to this is the representation of the interactions. Allowing multiple participants, connected through a computer network, to inhabit the virtual world and to interact with the objects herein, makes the interaction mechanism even more complex. After all, the state of such a Collaborative Virtual Environment (CVE) should be synchronized and distributed to all participating sites in order to give the users the illusion of being located in the same place at the same time [1]. As a special case, to be able to see what other participants are doing in the VE, their state (especially their positions and actions) should be distributed among all users as well.

In contrast to managing and distributing the state of the virtual world, animation is not an essential enabler of interaction in CVEs. However, proper use of animation can dramatically increase the user's feeling of immersion while interacting in these worlds [2]. This is especially true for the animation of avatars, since they are the user's means of interaction in the VE [3]. It is through his avatar that a user can, for example pick up objects or point at locations in the virtual world. In order to achieve even the slightest level of realism, such actions clearly have to be accompanied by appropriate animations. However, since CVEs are real-time applications, no appeal can be made to off-line techniques to take care of the animation in these virtual worlds. Fortunately, real-time character animation has improved drastically over the past few years. Thanks to recent advances in computer hardware, it is currently possible to produce results in real-time that were formerly only achievable in off-line animation [4].

In the context of CVEs however, animation has received relatively little attention. This is manifested in the fact that advanced animation techniques are only slowly finding their way into

CVEs. A possible explanation of this could be the computational complexity of these more advanced techniques. Another possible explanation could be the inherent networking aspect of CVEs. Since avatars are a part of the shared world offered by a CVE, their state has to be distributed to other connected users. Advanced animation techniques tend to produce relatively large quantities of data that need to be transmitted over the network, which is often considered unacceptable by CVE designers. In this work we show a solution to at least a part of that problem.

2 Related Work

2.1 Dynamic Virtual Environments

Current research of interactivity within VEs is primarily concentrated on user navigation and actor-object interaction using direct interaction techniques [5] [6] [7]. As a result, many of the current VEs limit their interactive behavior to executing predefined animations when a particular event occurs, or allowing translations and rotations of objects using direct interaction techniques [7]. More advanced actor-object interactions are commonly handled by programming each possibility specifically for each case [8]. This approach is far from general and definitely not runtime adjustable.

Modelling virtual world objects is a very important part of the VE development process. Although many mechanisms for describing the visual elements of objects exist, only a few systems permit the dynamics or interaction properties of the objects to be described as well [9]. A first step into the description of application independent actor-object interaction was taken in [10] where Levinson used an object specific reasoning module, creating a relational table to inform actors on object purpose and functionality and used it mainly for simulating the task of grasping. The first time all functional and behavioral information of objects was described at object level was in the Smart Object approach presented in [11] [12]. Here, Kallmann et al. propose a framework for general interactions between virtual actors and objects in the virtual world. The idea is that all information necessary for a virtual actor to interact with an object is included in the object's description. This is achieved by using a combination of predefined plans specified through scripted commands.

2.2 User Embodiment

Many of the first CVEs used very simple avatars to represent connected users in the virtual world [13]. For example, RING [14] used yellow spheres with green orientation vectors as user embodiments, and early versions of MASSIVE and DIVE used avatars that were composed only of a few basic geometric shapes (so-called blockies [15], see figure 1(a)). Articulated human-like avatars were introduced a few years later in the NPSNET system [16]. It soon became clear that integrating these so-called virtual humans (see figure 1(b) for an example) in CVEs increased the natural interaction within these environments and generally also resulted in a higher feeling of immersion for connected users [17]. Futhermore, [2] demonstrates that when these virtual humans are able to perform animations, the feeling of immersion is increased even further. As a result, it should not come as a surprise that almost all recent CVEs use animated human-like avatars. Examples include SPLINE [18], later versions of the DIVE platform [15], the Virtual Life Network (VLNET) system [3] and most current Massively Multiplayer Online Games (MMOGs). Finally, the last few years we are witnessing the emergence of CVEs which focus on virtual community building. These CVEs, like for example Second Life [19] and There [20], attach great importance to user embodiment and thus offer their users very advanced avatars that are often also very customizable (see figure 1(c)).

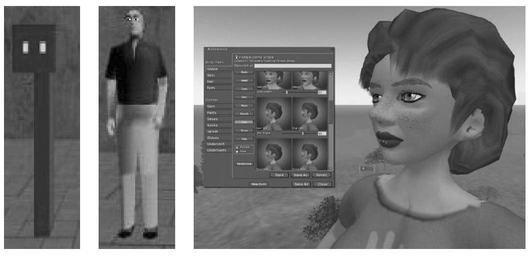
(a) (b) (c)

Figure 1: avatars: (a) Blockie; (b) Virtual humans; (c) Avatar customizability (Second Life).

2.3 Real-time Character Animation

Until recently, VE designers had to rely on simple representations like 3D hierarchic articulated objects or single mesh characters to create avatars that are able to perform animations [21]. There are major disadvantages associated with both techniques [4]. When animating 3D hierarchic articulated objects, unrealistic gaps can appear between the seperate parts of the model. Single mesh characters on the other hand require very large animation files. Furthermore, this representation lacks flexibility since on the fly creation of new animations is impossible. Thanks to recent advances in computer hardware however, skeletal animation has found its way into real-time applications [4] [21]. Skeletal animation combines the advantages of both 3D hierarchic articulated objects and single mesh characters without inheriting any of their drawbacks. As a result, it has become a standard in the field of real-time character animation which has been adopted by almost all recent CVEs.

Skeletal animation requires that models consist of seperate layers which each have their own physical and geometric properties. After appropriate constraints are enforced between the different layers, animating the model comes down to controlling its skeleton (the undermost layer of the model) [22]. This can be done in several ways. Examples include motion capturing, keyframing, forward and inverse kinematics and dynamics ([22] [23]) . Almost all current CVE systems rely solely on keyframing to take care of the animation in the virtual world. Examples include all the CVEs already cited in this section, except for the VLNET system. VLNET's main focus is on the integration of virtual humans in VEs, and it therefore has a very advanced animation engine which supports among other things inverse kinematics and facial animation [3].

2.4 Interactive Object Approach

In this work we present a totally dynamic VE where every object can be interacted with by human users. Also, in contrast to the Smart Object approach, where only avatar-object interactions are described, we would like *all* objects within the virtual world to be able to interact with every other object. We therefore propose a new general object interaction scheme for dynamic VEs in section 3 that uses a similar object functionality description as does the Smart Object approach, but extends it to all objects in the VE. Furthermore, we make no distinction on what kind of objects (agents, human users or world objects) are interacting, resulting in more dynamic VEs which also allow object-object and object-actor interactions. Section 4 presents a new and efficient way for human users to dynamically interact in the virtual world through their avatar. For both the collaborative interaction objects approach and the distribution of the dynamic avatar interactions, we try to keep the network load as low as possible. Finally, we describe the results of integrating our work into the ALVIC CVE framework which was first introduced in [24] in section 5, and we present our conclusions and future work in section 6.

3 Creating a Collaborative Interaction World

3.1 Interactive Objects

The starting point of our interactive object framework was given in [25]: "How different interaction

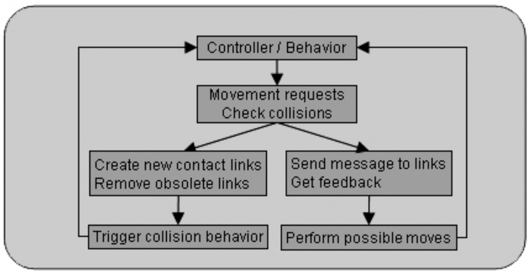

Figure 2: basic interaction scheme.

requests are processed, as well as how many participants may interact with an object concurrently, is highly dependent on the object itself". Generalizing the principles used in the Smart Object approach, we created a general object description capable of describing all interaction information necessary for objects to interact with all other actors in the virtual environment (object or interactor). A full description of the object format is beyond the scope of this paper, however the main structure is:

Object properties consisting of a full part description (id, file, parent part, and transformation information), part constraints, object constraints, actions and variables.

Interaction properties describing commands that can be handled by the object, a specification of the interaction zones and triggers.

Object behaviors containing all the scripts describing the object's behavior and all trigger-command pairs showing what command to be called when triggered, or what scripts to be executed.

Each object part can have its own animations, actions and constraints. Object constraints define constraints that are dependant on more than one part. Object variables can be defined to store data or state information. The triggering occurs for example when something collides with a defined interaction zone of an object (collision trigger), when a condition on an object variable is met (state trigger) or when a certain time is reached (time trigger). How an object reacts to a trigger or a called command is described in the behavior part of its description. Triggers can be used to invoke actions as well as behaviors. Actions are defined as combinations of object part movements or movements of the object itself, over a certain time. Behaviors on the other hand are described in a scripting language and can be used to initiate more advanced interaction behavior such as collision handling or the triggering of other objects' behavior. Scripts can access object variables and can initiate actions, animations and other behaviors. The entire object description is done in XML which makes it easy to construct, check for errors and to parse. Furthermore it is easy read and easy to understand.

The main advantage of this approach 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 or their parts and their behavior are easy to modify, even at runtime. Secondly, the objects are easy to reuse in other applications or just partly for the construction of other interactive objects. The scripts for opening a door can for example also be used for a closet. Furthermore, new or objects can be introduced in the application at any time. High-level planners could easily use the object information for planning how an AI actor can perform tasks with the objects. At the networking level, we can use the information, for example a part's constraints, to see if it can move or rotate, hereby concluding if frequent updates should be sent.

3.2 Interactive Worlds

To create totally dynamic virtual worlds, we construct our world only from the interactive objects described in the previous subsection. Objects are interconnected by links that create a structure for the world and provide a means of communication. An interaction layer is responsible for creating and maintaining the objects in the virtual environment. It is the central control unit for all object interaction in the VE. First of all it creates necessary links between objects and maintains these for as long as they are required. The most important links the interaction layer can create are contact links, attach links, controller links and parent/child links. Secondly the interaction layer is responsible for checking all object triggers. Furthermore it is responsible for the actual execution of object functionality (scripts and actions). Figure 2 shows the basic interaction scheme. As an example, consider the case where an object hits another object while moving. When the collision occurs, a contact link between the objects is set up by the interaction layer. The object then sends a move message containing the movement parameters through the link. Subsequently, the interaction layer checks the contact and movement behavior and constraints of all objects that are connected through contact and attach links. Finally the interaction layer calculates what behavior is possible for all the objects involved, calculates the parameters and triggers all necessary behavior. If in the next step, after movement and the first step of the triggered behavior of all the objects, the objects are no longer in touch, the contact link is removed.

All interactions in the VE are handled in a similar way. To manipulate an object in a VE, a controller can be attached to it. A controller is a higher-level part of the application, which takes for example user input or output from an AI module and converts it into commands that are sent to the object. A controller is connected to an object by a controller link, which makes it possible to send commands to the objects via the interaction layer. This approach is very different from the Smart Object approach where actors (human or AI) are not subjected to the same interaction rules as the objects they manipulate. As a result, our interaction scheme is a lot more general and the only one necessary for an entire interactive VE. Of course more work is necessary in the modelling stage, but the reusability and flexibility of designed objects by far outweighs this disadvantage.

3.3 Distributing the Collaborative World

To allow remote users to simultaneously interact and collaborate within a VE, they must be informed by network messages whenever the state of the VE changes. We decided to use a client/server architecture where the server runs the simulation of the interactive world. This means that the execution of scripts and all the other simulation data is calculated at the centralized server. Furthermore, the server continuously checks for interaction requests from clients and processes them as they arrive. After processing an interaction request, the server uses information from the interaction layer (which uses object interaction information) to see which object parts have changed state and must therefore be communicated to clients.

Tests have been performed with both TCP and UDP to check which network protocol was best suited for transmitting the state updates. Although TCP provided a reliable stream, the delays it introduced were too great to make it of any use for direct interaction in a VE. On the other hand, the delays introduced by sending updates with UDP were practically imperceptible. Unfortunately, UDP is an unreliable protocol that gives no guaranties whatsoever about the actual delivery of sent packets. In the case of non-essential updates however, this has proven to be acceptable. We have therefore chosen to use a hybrid approach. At startup, clients connect to the server with TCP to receive the current state of the virtual. The system thereafter falls back on UDP to keep the different clients synchronized. Furthermore, to place less load on the network, the server uses multicasting to send state updates to all interested clients at once.

4 Using Animation to Support Interaction

4.1 *Real-time Inverse Kinematics*

Most current animation engines for CVEs use skeletal animation but are limited to displaying a restricted set of predefined animations since they rely solely on keyframing to animate their models. The main advantage of displaying animations this way is that it is computationally inexpensive. The major drawback of the keyframing approach however is its lack of flexibility. Although real-time adjustment of basic animation parameters (like for example the speed at which an animation is to be displayed) can be supported, it is impossible to create new animations on the fly. This makes this approach suitable for displaying animations which can be predefined easily and which require very little real-time tweaking, like for example running or jumping. It is however not flexible enough to enable more complex interactions in the VE, like for example grabbing objects or pointing with the hand. Such interactions require animations that can be tailored in real-time to the current interaction situation. Consider the case of object grabbing as an example. If the animation engine relies solely on keyframing, allowing users to grab objects at different heights in the VE requires the availability of multiple predefined grabbing animations, which can cost an animator a lot of time to create. This problem can be circumvented by placing objects at one or possibly a few fixed heights in the VE and predefining very specific and detailed animations for grabbing at these heights, but this approach clearly limits the freedom of the VE designer. Furthermore, even when such interaction restrictions are applied, the results are still not always satisfactory. Imagine a user who is not perfectly aligned with the object he is picking up for example. So in order to create truly dynamic and interactive virtual worlds, more advanced animation techniques have to be introduced in CVEs. We believe real-time inverse kinematics is a good candidate.

Inverse kinematics (IK) allows an animator to directly specify the desired position and orientation of the loose end of a joint chain (the so-called end-effector). The animation system will subsequently compute how the different joints in the chain have to be translated and rotated so that the end-effector will reach the specified position and orientation [23]. A possible example of an IK chain could be the arm of a virtual human, with the hand of that arm being the end-effector. It should be apparent however that performing IK has a relatively high computational cost associated with it. While this limited its use to off-line applications in the past, todays computers are capable of doing IK in real-time [4].

Integrating IK in CVE animation engines can compensate for their current lack of flexibility, since it enables on the fly creation of new animations while the user is interacting in the virtual world. As an example, reconsider the case of object grabbing. By using IK, a grabbing animation for the arm can be created on the fly that takes the height at which the object is located into account. It suffices to feed the position of the object to the IK algorithm and to save the current pose of the model and the pose calculated by the algorithm as respectively the first and second keyframe of a new animation. This new animation can then be displayed by the animation engine as if it were a predefined animation. If appropriate constraints are applied to the different parts of the IK chain [23], this approach generally yields more realistic results than displaying a predefined grabbing animation. Furthermore, no grabbing animations need to be predefined, and no restrictions need to be applied to the height at which objects can be located in the VE. Another advantage of IK is that it allows direct control over the end-effector of the IK chain of an animated object (typically the hand of an avatar's right arm). We hereby note that we do not focus on simulating humans realistically, but in contrast try to make an interaction scheme for all kinds of animated avatars.

In order to be as general as possible, we propose seperating the IK related information of a model from the actual model data and its keyframes. We therefore accompany our model files with XML files that contain information about their IK possibilities. These XML files first of all indicate which joints of the model form an IK chain. Secondly, these files contain a description of a bounding box which the IK chain will never leave while it is being animated through inverse kinematics. And finally, degree of freedom (DOF) constraints can be specified for all the joints in the IK chain which will be taken into account by the IK algorithm. This way we can prevent joints from rotating into positions that are physically impossible to achieve. The main advantage of using such seperate IK files is that the IK algorithm is not restricted to avatars but can be applied to any object in the VE. Furthermore, the separation of IK information from actual model data increases model reusability. For example, if we would like both a left- and right-handed avatar, we would need to create two (very small) XML files, but we could use the same animated model file.

It should be noted that the results presented in this section can also be achieved by using other animation techniques like for example inverse dynamics. Some of these techniques are even capable of producing visually more realistic interaction animations than IK. Unfortunately, these techniques also have a much larger computational cost associated with them. Since processor time is still a scarce resource, we believe that at the moment IK is the most suitable animation technique to enable complex, dynamic interactions for all animated models in a virtual world.

4.2 Distributing Animation Related Information

Animating articulated objects in CVEs introduces new network traffic in these systems since animation related information will need to be distributed among connected users. There are several ways animation data can be represented for transmission [3]. Some of these representations place a large load on the network but do not require a lot of processor time to encode/decode. Others are computationally expensive to encode/decode, but require only small update messages.

We tried to keep the new network traffic introduced by animation as low as possible by using compact update messages. When keyframing is used to animate a model, only a high-level description of the currently active animations is sent over the network. When receiving such a message, remote sites must start interpolating locally to be able to display these animations. Similarly, instead of transmitting the transformations of all the joints in the IK chain when animating using inverse kinematics, only the position of the end-effector of the chain is sent. Upon arrival of such messages, remote hosts will need to calculate the transformations of the intermediate joints in the chain themselves by feeding the received end-effector position to their local IK algoritm. It should be apparent that this approach minimizes the network load but increases the necessary processor time in the decoding phase. Note that there is also a small accuracy loss associated with distributing animation data this way, but this normally will be acceptable for most multi-user applications. For example, the currently displayed frame of an active animation (such as running) might not be identical at all participating sites. We believe that showing the fact that the avatar is running is more than enough to see what he is doing, and that exact synchronization of the animations is not a necessity.

5 Test Results

For testing our interactive avatars in a dynamic CVE, we coupled our collaborative interaction world to the ALVIC multi-user framework described in [24]. All objects present in the VE are defined as interactive objects. We have added two different interactive objects to the avatars, one for the avatar's body and one for its interactive IK chain, which in this setup was the avatar's right arm. Instead of moving the avatars immediately, we use the information of the collaborative interaction object for moving and manipulation. This way, the two objects are always synchronized.

By combining our interactive objects framework with our animation engine which supports both keyframing and IK, meaningful and realistic interaction became possible in the virtual world. One example is opening a door by pressing a button in the VE. Since IK allows direct control over the end-effector of an IK chain, users can move their avatar's hand to the button. When the interactive object associated with the hand and the button (which itself is coupled to an interactive object) collide, the behaviour of the button is triggered. This behaviour in turn triggers the script controlling the door, which starts a keyframed animation to open the door. It is however also possible to open the door manually by moving our avatar's hand against it.

At first we used keyboard input to directly control the right hand of the avatar. However, this approach soon proved to be far from intuitive and efficient. A possible explanation could be that in this approach the user has, except for visual information, no reference of where his avatar's hand is positioned. Also, the context switch between controlling the arm and avatar navigation was not very clear since we also used the keyboard to move around in the virtual world. Users explicitly had to press a keyboard button to switch between direct control mode and navigation mode, which often resulted in situations where users tried to move forward while they were actually controlling the arm and vice versa. We therefore provided support for the MicroScribe-3D input device [26] (see figure 3) to control the IK chain. Based on the observation that the MicroScribe-3D is constrained in its movements, we automatically map the movement space of the MicroScribe-3D onto the IK bounding box of the model (see figure 3). This mapping is dependant of the model used, which ensures that input from the MicroScribe-3D is always correctly transformed to the corresponding position in the IK bounding box. This position is subsequently fed to the IK algorithm which will try to move the end-effector as close to this target position as possible.

Thanks to this mapping, the pose of the avatar's arm approximately corresponds to the pose of the user's real arm when he is using the device (see figure 4). This turned out to allow more intuitive control over the IK chain, which confirms the view

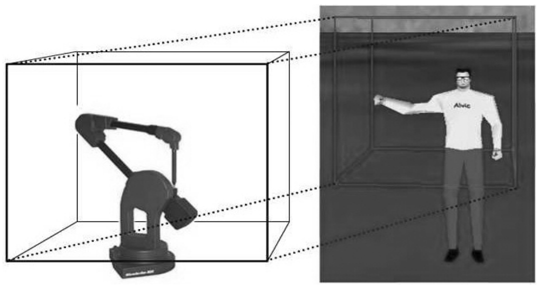

Figure 3: The MicroScribe-3D range is mapped onto the model's IK bounding.

given in [27] and [28] stating that the inclusion of motion relative to a spatial reference or the user's own body in interaction techniques decreases the cognitive load of the user. Furthermore, the context switch between controlling the arm and navigation also became much more clear by introducing the MicroScribe-3D. One disadvantage of the Microscribe-3D is that it has no force-feedback, making it possible for the user to keep its hand moving while the avatar's hand is stopped by for example a collision. We believe in this case however that the visual feedback given to the user suffices.

On a LAN, the network delays introduced by the deployed client/server architecture are practically unnoticeable. This is mainly due to the efficiency of UDP and the decision to keep interaction update messages as small as possible. Furthermore, only distributing the animation identifiers and the position of the end-effector of the IK chain to communicate the avatar's current state and actions works very well. The exact pose of the avatar may not be completely identical at all participating sites, but we do not believe that this is necessary in this kind of application.

6 Conclusions and Future work

We have presented a new general object interaction scheme capable of handling all interactions in a CVE. The scheme uses objects that contain their own interaction information and behaviour. Links between the objects are used to structure the world and to allow for communication. A special interaction layer controls all objects and is responsible for creating and destroying the links between them and the handling and the triggering of all their behaviour. Furthermore, this layer is also responsible for determining the update rate of every part of the objects. By treating all parts of VEs (actors, objects and the world itself) equally, our interaction scheme is far more general than the Smart Object Approach. Furthermore, we have shown how the proper use of animation, and in particular inverse kinematics, can increase the interaction possibilities in these VEs. Inverse kinematics enables on the fly creation of new animations that are tailored to the current interaction situation. This results in more realistic animations and allows for more heterogeneous virtual worlds, which usually also enhances the feeling of immersion for connected users. We have shown that our approach works by integrating and testing them in the ALVIC framework.

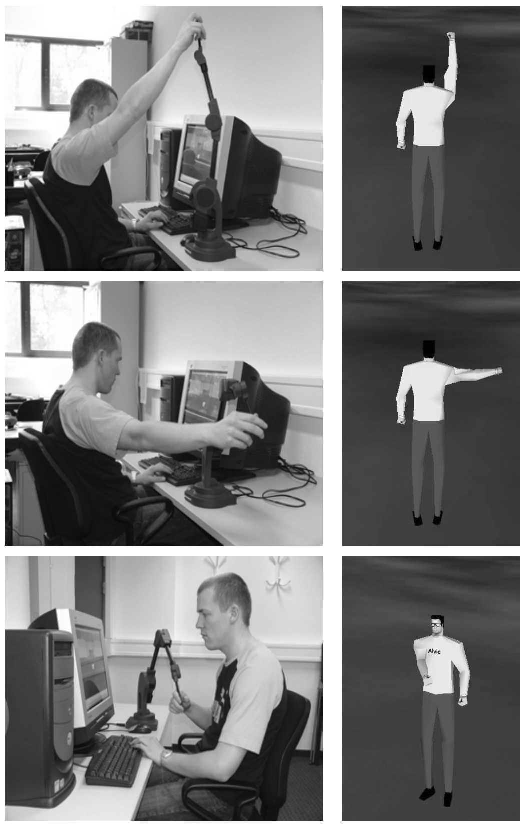

Figure 4: Images of a user who is controlling his avatar's arm with the MicroScribe-3D.

In the future we would first of all extend our interaction framework to support more kinds of models and extend our script engine to support more functionality. Furthermore we shall create a modeler for creating our collaborative interactive objects making it easier to create dynamic worlds. Thirdly we want to add force feedback support to our framework and replace the Microscribe-3D input device by a force feedback device and see if user interactions are improved. Finally, some tests on scalability and network traffic will be performed.

Acknowledgments

Part of this research was funded by the IWT project number 020339 and the Flemish Government.

References

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

[2] J. Casanueva and E. Blake. The Effects of Avatars on Co-presence in a Collaborative Virtual Environment. *Technical Report CS01-02-00, Department of Computer Science, University of Cape Town, South Africa*, 2001.

[3] T. Capin, I. Pandzic, D. Thalmann, and N. Thalmann. Realistic Avatars and Autonomous Virtual Humans in VLNET Networked Virtual Environments. *Virtual Worlds on the Internet (J. Vince and R. Earnshaw, eds.) IEEE Computer Society, Los Alamitos*, pages 157–173, 1998.

[4] E. Anderson. Real-Time Character Animation for Computer Games. *National Centre for Computer Animation, Bournemouth University*, 2001.

[5] C. Hand. A Survey of 3D Interaction Techniques. *Computer Graphics Forum 16(5)*, pages 269–281, 1997.

[6] M. Mine, F. Brooks Jr., and C. Sequin. Moving Objects in Space Exploiting Proprioception in Virtual Environment Interaction. In *Proceedings of SIGGRAPH'97*, Los Angeles, 1997.

[7] D. Bowman. *Interaction Techniques for Common Tasks in Immersive Virtual Environments: Design, Evaluation and Application*. PhD thesis, Georgia Institute of Technology, 1999.

[8] S. Smith, D. Duke, and J. Willans. Designing World Objects for Usable Virtual Environments. In *Workshop on Design, Specification and Verification of Interactive Systems (DSVIS'00)*, pages 309–319, 2000.

[9] S. Pettifer. *An Operating Environment for Large Scale Virtual Reality*. PhD thesis, University of Manchester, 1999.

[10] L. Levinson. *Connecting Planning and Acting: Towards an Architecture for Object-Specific Reasoning*. PhD thesis, University of Pennsylvania, 1996.

[11] M. Kallmann and D. Thalmann. Modeling Objects for Interactive Tasks. In *EGCAS'98 - 9th Eurographics Workshop on Animation and Simulation*, Lisbon, 1998.

[12] M. Kallmann and D. Thalmann. Direct 3D Interaction with Smart Objects. In *Proceedings of ACM VRST'99*, London, 1999.

[13] N. Magnenat-Thalmann and C. Joslin. The Evolution of Virtual Humans in NVE Systems. *ICAT2000*, pages 2–9, Oct 2000.

[14] T. Funkhouser. RING: A Client-Server System for Multi-User Virtual Environments. In *Proceedings of the 1995 symposium on Interactive 3D graphics*, pages 85–92. ACM Press, 1995.

[15] S. Benford, J. Bowers, L. Fahlén, C. Greenhalgh, and D. Snowdon. User embodiment in collaborative virtual environments. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 242–249. ACM Press/Addison-Wesley Publishing Co., 1995.

[16] M. Macedonia, M. Zyda, D. Pratt, D. Brutzman, and P. Barham. Exploiting Reality with Multicast Groups: A Network Architecture for Large-scale Virtual Environments. In *Proceedings of IEEE Virtual Reality Annual International Symposium*, pages 2–10, 1995.

[17] D. Thalmann. The Role of Virtual Humans in Virtual Environment Technology and Interfaces. In *Proceedings of Joint EC-NSF Advanced Research Workshop*, 1999.

[18] R. Waters, D. Anderson, J. Barrus, D. Brogan, M. Casey, S. McKeown, T. Nitta, I. Sterns, and W. Yerazunis. Diamond Park and Spline: A Social Virtual Reality System with 3D Animation, Spoken Interaction, and Runtime Modifiability. In *Presence: Teleoperators and Virtual Environments, Vol. 6, No. 4*, pages 461–480, August 1997.

[19] Second Life: Your World. Your Imagination. World Wide Web, `http://secondlife.com/`, May 2004.

[20] Welcome to There!! World Wide Web, `http://www.there.com/index.html`, May 2004.

[21] J. Lander. On Creating Cool Real-Time 3D. *Gamasutra*, 1(8), October 1997.

[22] T. Giang, R. Mooney, C. Peters, and C. O'Sullivan. Real-Time Character Animation Techniques. *Technical Report TCD-CS-2000-06*, February 2000.

[23] C. Welman. *Inverse Kinematics and Geometric Constraints for Articulated Figure Manipulation*. PhD thesis, School of Computer Science, Simon Fraser University, 1989.

[24] P. Quax, T. Jehaes, P. Jorissen, and W. Lamotte. A multi-user framework supporting video-based avatars. In *Proceedings of the 2nd workshop on Network and system support for games*, pages 137–147. ACM Press, 2003.

[25] W. Broll. Interacting in Distributed Collaborative Virtual Environments. In *Proceedings of the IEEE Virtual Reality International Symposium*, pages 148–155, Los Almitos, 1995.

[26] Immersion Website. World Wide Web, `http://www.immersion.com/digitizer/products/index.php`, May 2004.

[27] K. Hinckley, R. Pausch, J. Goble, and N. Kassell. Moving Objects in Space: Exploiting Proprioception in Virtual-Environment Interaction. In *Proceedings of the 7th annual ACM symposium on User interface software and technology*, 1994.

[28] M. Mine, F. Brooks Jr., and C. Sequin. A survey of design issues in spatial input. In *Proceedings of SIGGRAPH 97*, 1997.