# A Real-time Performance System for Virtual Theater

Qiong Wu
Computing Science
University of Alberta
Edmonton, Canada
qiong@cs.ualberta.ca

Pierre Boulanger
Computing Science
University of Alberta
Edmonton, Canada
pierreb@cs.ualberta.ca

Maryia Kazakevich
AICT
University of Alberta
Edmonton, Canada
maryia@ualberta.ca

Robyn Taylor
Computing Science
University of Alberta
Edmonton, Canada
rltaylor@ualberta.ca

## ABSTRACT

The idea of combining virtual reality technology and theatrical tradition to create virtual plays has captured artists' imaginations for some time. Using conventional technology, the use of virtual characters in a theatrical performance often integrates the predefined animations of virtual actors into the theater scene, resulting in a performance that can feel stilted and unresponsive due to its pre-programmed nature. This paper proposes a new system that allows actors to animate virtual characters in real time, resulting in a more flexible and interactive theatrical performance experience. Actors are sequestered at a remote site, invisible to the audience, and are digitized by a motion capture system. Using camera feeds to provide the remote actors with information about the behavior of the live actors and audience in the theater, the remote actors can adapt their virtual counterparts' behavior to react to live events in real-time, giving the illusion to the audience that the virtual characters are responsive to their actions.

The system integrates display peripherals, networked cameras, real-time motion capture system, gesture recognition, and a virtual environment development suite (Virtools) to create a true virtual theatrical performance environment. In the paper, we will present the various concepts developed so far and an example of a virtual theatrical performance called *Trickster at the Intersection* that was presented during Smart Graphics 2010 at Canada's Banff Centre.

## Categories and Subject Descriptors

H.5.1 [**Information interfaces and Presentation**]: Multimedia Information Systems—*Artificial, augmented, and virtual realities*; J.5 [**Computer Applications**]: Arts and Humanities—*Performing arts (e.g., dance, music)*

## General Terms

System, Performance, Human-Computer Interaction


*MM'10,* October 25–29, 2010, Firenze, Italy.


## Keywords

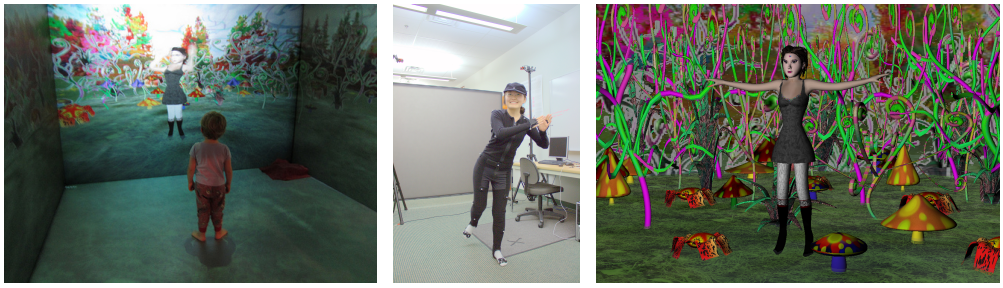
Virtual Theater, Motion Capture, Audience Interaction, Real-time 3D Graphics, Gesture Recognition

## 1. INTRODUCTION

Over the past decade, Virtual Reality (VR) have been used in a wide range of artistic applications including virtual theater. Virtual theater preserves the theatrical performance form through direct mediated interaction of avatars and objects, and it is known for its flexibility of interaction [1], and its potential to create novel theatrical elements [19] that would be impossible to do in the real-world. However, performances in virtual theater are often pre-recorded which make them pre-programmed, difficult to use, and lifeless compared to the spontaneity and improvisation of a real performance.

This paper presents a new system that preserves the interactivity of real theater by allowing for a better sense of presence and the ability of improvisation. The system features immersive display peripherals used as part of the theatrical set, networked cameras, real-time motion capture and gesture recognition, integrated into a virtual environment development suite (Virtools). The system allows natural interaction between the audience and the virtual characters without the aid of tethered input devices. In the current system, the audience can view the virtual actor in an immersive three-wall CAVE (see Figure 1(a)) to interact with a remote actor digitized in real-time by a motion capture system (Figure 1(b)) allowing the actor to control an avatar (see Figure 1(c)) and to create actions in the virtual world using gesture. While the audience does not know that the virtual character is controlled by a remote actor, the remote actor can see and listen in real time to the audience and other actors located on the real theater set by using networked cameras and microphones.

One of the key component of the system is the use of gesture recognition technology to create actions in the virtual world. For example, as the actor performs his/her gestures, one can create actions that materialize objects, stretch objects, turn them transparent or appear/disappear, and emit sparkles etc. In many ways, many gesture-triggered special effects can be created depending on the scenario of the play. By using gesture recognition based only on the actor's motion, one can smoothly integrate actions into the performance, making each performance a unique experience that can be controlled by the subtlety of the human actor and the reaction of the audience. In addition, remote cameras on the audience side can be used by the actor to observe the participant and to give them



(a) (b) (c)

**Figure 1: Virtual theatrical performance of the *Trickster at the Intersection*: (a) The AMMI CAVE is used to display the trickster and its virtual world immersively; (b) A performer in a motion capture suit who is responsible to drive the embodied trickster and to create his magic world using gesture recognition; A video camera located in the CAVE allows the performer to monitor the audience reaction and to improvise ways to surprise and trick them; (c) The trickster in his magic forest.**

the illusion of responsiveness. This new system opens the door to new forms of theatrical expression as well as standard theatrical plays. For example, plays that require imaginary characters, such as Shakespeare's *Macbeth*, *The Tempest*, and *Midsummer Night's Dream*, could use this technology to add realistic personifications of magic characters like the witches in *Macbeth*.

Two key issues are addressed in this paper. First, by using full-body tracking and gesture recognition, manipulation of the virtual characters and virtual world is intuitive and natural. The system is fully automatic and trivial to experience. The actors are not required to have specialized training and can animate the virtual characters or create the virtual world by simply moving naturally. Second, the system is truly interactive and fluid as the motion capture, the gesture recognition, and the rendering are performed in real-time (100 FPS).

In Section 2, we will review the pertinent literature. In Section 3, we will describe the system architecture and its application to virtual theater. In Section 4, we will describe a virtual theatrical performance called *Trickster at the Intersection* that was presented during Smart Graphics 2010 at Canada's Banff Centre. We will then conclude and discuss future work in Section 5.

## 2. RELATED WORK

Virtual plays have been around for decades. However, it has never been thought as a replacement of real theatrical performance. Reeve [15] pointed that key features of a typical theatrical rehearsal process can significantly improve the sense of presence for participants within a shared virtual environment. In [15], it is demonstrated that traditional shared Virtual Environments (VEs) for the production of theater have specific requirements to create a sense of presence. The level of presence is dependent on the actor-avatar, actor-space, and actor-actor relationships. Current virtual theater environments cannot achieve this sense of presence. The difficulty of changing normal theater to its virtual form, and to retain the original narrative is immediately evident.

There has been extensive research and literature on increasing user's commitment to the virtual environment and increasing user's feeling of "presence" by having user involved at different levels of interaction. The most naive interactions include the use of limited input resources, such as a simple mouse [21], a keyboard [10], video cameras[2], Nintendo Wiimotes [17], or a pressure sensor [20] etc, and the use of AI agents to interpret inputs. In these systems, once the input is interpreted, a pre-defined animation is played, during which a user has no control over its execution. Other systems use speech and gesture inputs instead. For example, Dow et al.[4] proposes to use remote human operators to play "behind-the-scenes" roles (type player's spoken utterances) to control a live embodied character, so that it is able to recognize user speech and gesture inputs. In such systems, users still passively play the pre-scripted narratives, and specifying interactions remains in the domain of programmers.

Higher level of interactions allow characters in a virtual theater to be treated as virtual actors and the novice user can construct narratives in which they appear [8]. For example, the "Virtual Theater" project of Barbara Hayes-Roth [7] and the IMPROV virtual actors system [14]. These systems allow users who would normally be excluded from the creation of play to be involved at different levels ranging from choosing their viewpoint to defining dynamic behavior of virtual actors using various tools, such as simple scripts, so that authoring virtual environments maybe performed by novice user. At a certain level, these systems simulate the improvisation characteristic of theater performance.

Recently, there have been several efforts to build virtual characters in a more expressive way that responds to user's direction [18, 12], in 2D multi-media environments as well as in virtual environments. Intuitive and expressive control of virtual character require high degrees of freedom (DOFs) input in order to be as expressive as real human actors. Ninomyia [13] allows user to control virtual marionette characters based on computer graphics using their hand and finger movements as if they were controlling marionettes in a theatrical play. The system recognizes hand gesture of the marionette manipulator and transforms it into motion of a marionette character. Using network, the system allows multiple marionettes to be included by allowing multiple users to join the networked virtual marionette theater. This system is excellent for marionette theater but would not allow actors or dancers to naturally interact with their virtual counterpart. To solve some of those issues, Cheok [3] describes an interactive virtual theater system based on embodied mixed reality space and wearable computers. Here again, there is no direct link between the actor's normal gesture and the virtual character expression. Other systems are capable of live performances, such as puppetry [5], where human actors directly control some features of the virtual character using hand movement. The ability to animate virtual characters realistically using real actors truly creates a sense of reality and presence that no AI agents have ever been able to match so far.

Application of real-time motion capture data in live performance has also been explored to create more direct link between the virtual character and live dancer/actor by several researchers [11, 6, 16]. However, they do not consider virtual environments and narratives

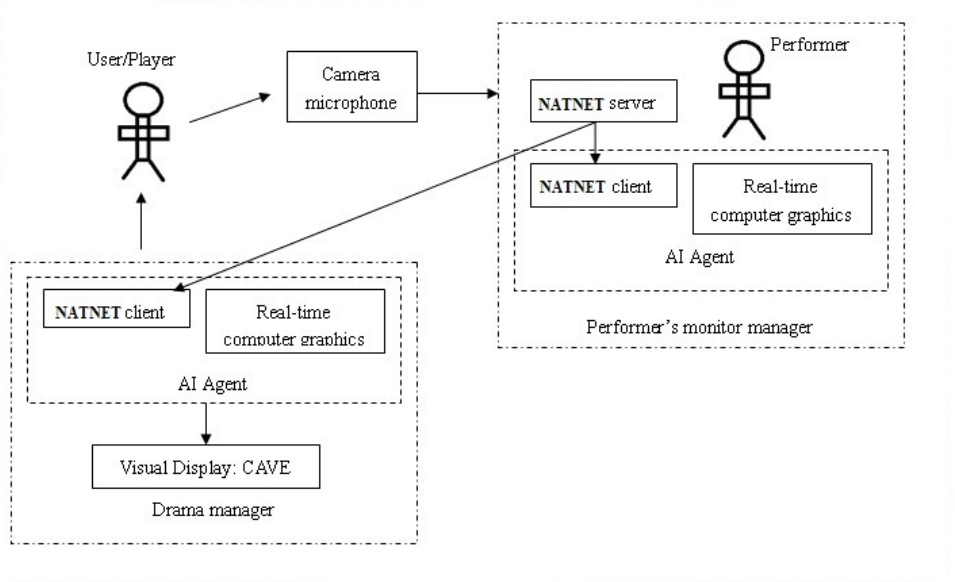



**Figure 2: System architecture for the virtual performance: *Trickster at the Intersection***

driven by performance as our system does. Actors also usually need specialized knowledge to use real-time motion capture in actual performances [6]. By using automated data processing and gesture recognition, we will demonstrate that it is possible for actors with limited technical knowledge to deliver high-quality performance and also drive actions in narrative by triggering computer-generated animations based on gesture recognition.

# 3. SYSTEM ARCHITECTURE

The system is basically composed of the following components. The avatar and scene rendering are performed on two networked high-end graphics PCs, one as a performer's monitor and the second one managing the display at the theater set. At the theater location, a display peripheral can be inserted in the set using transparent projection sheet technologies [1] or around the audience using a CAVE [2] like system. At the theater location the set is equipped with web cameras and microphones linked to the remote actor location using a standard network. At the remote location, the actor is tracked using an Optitrack motion tracking system calibrated to transmit skeleton information to the theater computer. The system is integrated using Virtools VR authoring tools from Dassault Systems. One can see at Figure 2 the architecture of the system. The communication between the motion tracking system and the rendering programs is performed using a modified version of the Open Source NATNET client/server utility.

On the performer's monitor, performers are able to see the reactions of the real audience as well as a feed showing a representation of their avatars in the virtual world that is being created during the performance, allowing them to monitor how their actions affect both the physical and the virtual worlds. Because the performance is driven by real-time motion data, this information need to be transmitted to both location via network (performer's monitor manager and theater manager). The virtual play that the audience sees on the set is always the same as the one performer sees remotely. This implies that the virtual performance must be synchronized in real-time on the performer's monitor as well as on the theater screen.

The virtual world in our project is actually a media layer where the interaction between audience and performer can be integrated and manipulated at will. In this way, we are able to have both audience and actor involved in the creation of the virtual world and how it behaves and changes.

The need for real-time is critical for this application. With motion capture system, high-speed internet, and two PCs each equipped with two high-end graphics cards (NVIDIA GeForce 5800), our system runs in real time at a speed of 100FPS. By careful integration and system optimization there is no delay between the remote actor performance and the live action on the theater set.

## 3.1 Virtools Framework

Our system is integrated using Virtools, a visual programming environment developed by Dassault Systems to design interactive virtual environment. It has many advanced modules (modeling, animation, behavior, physics, artificial intelligence, sound etc) where a user can simply drag-and-drop on an object to create a behavior. In Virtools, each virtual world is described as a COMposition (COM) file. This file contains a number of environment models (e.g., 3D objects, cameras, lights), existing scripts and/or user written scripts to define object's behavior and display configuration such as: CAVE, HMDs, and single or multiple monitors. There are primarily two steps involved in our virtual play construction: environment modeling and behavior specification.

### *3.1.1 Environment Modeling*

Environment modeling is the construction of static environment, starting with the virtual characters and adding other virtual entities, not only visible entities but also objects like sound, light, virtual camera etc. All visible virtual objects populated in our virtual environment can be modeled and textured using standard graphics package, such as Maya and 3ds Max. Each model contains information including geometry, texture, shaders etc. This object creation process may also define object animation (refer to "static" animation as it describes changes in the object that are not defined

[1] http://www.reintek.com/index.htm

[2] http://en.wikipedia.org/wiki/Cave_Automatic_Virtual_Environment

by interaction/behavior specified by the system [8]), such as inverse kinematics (IK) and automatic skinning.

The virtual character is the most important object in our virtual world, as it is the one that directly interacts with the user in real-time. In addition, the direct stimulus that triggers other animations must be carried out in the context of the narrative of the virtual play. The virtual character's body parts are connected using the skeleton joints technique. In order to precisely map the actor's movement to the virtual character skeleton, one must map the measured 3-D points on the body suite to the character skeleton using a calibration procedure, so that the model is in the same coordinate system and the body parts are the same size as the performer. Each joint of the skeleton has six DOFs, including the position in cartesian coordinates system and rotation represented using quaternion or Euler angles. In total, 24 joints are used to animate the virtual character from motion capture data. Once a character is finished modeling in Maya, it can be imported into Virtools and all its static properties and animation (e.g. IK and skinning) are preserved.

After importing the character into Virtools, the body parts of the character are hierarchically organized in a tree structure and parented by a single "root" node. Positioning the "root" node can properly carry positions of all children joints computed based on IK. Therefore, to animate a character, we only need to position the "root" node and then rotate other body joints relative to "root".

### *3.1.2 Behavior Specification*

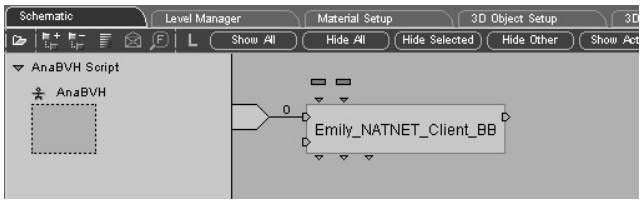



**Figure 3: The script of the virtual character is linked to the motion capture data client.**

Behavior specification defines the dynamic behavior of objects triggered by the performer's actions. In Virtools, object behavior can be defined using existing behavior module or user written module, called a Building Block. In our system, the behavior of virtual character is completely controlled by the performer's actions and is directly mapped to the real-time motion capture data. The behavior of other virtual entities are defined by the gesture recognition system. We implemented our behavior module as user-written building blocks (DLL file) in the Virtools environment. Applications of these user-written building blocks in Virtools are just like other building blocks provided by Virtools and must be compatible to the basic framework. Figure 3 shows an example of applying the building block developed for the project in a CMO file. Figure 3 shows the building block, "Emily_NATNET_Client_BB", that defines character behavior. "Emily_NATNET_Client_BB" mainly has two tasks. First, the skeleton data is streamed from the motion capture system using an Open Source NATNET client/server utility located at the remote site. NATNET is a specialized client/server utility that transmits skeleton data from the Optitrack Arena environment to the Virtools. The second task is to animate the virtual character by re-targeting the motion data from the NATNET server to the skeleton of the virtual character. The NATNET server IP and re-targeting array are two input parameters of that a user can specify for "Emily_NATNET_Client_BB" .

In order to navigate in larger space than the motion capture space, we had to add an extra wireless joystick to our system. The performer can use the joystick to control the virtual character to walk in the virtual world for long distances without having to move in the real-world. The basic functionalities of joystick buttons include moving forward/backward, left/right, and up/down. The motion of the virtual character is integrated relative to the performer's real movement. For example, if the current position of the performer is $\mathbf{V}_1$ (position vector), and the input from joystick is $\mathbf{V}_1$, the virtual character's position is $\mathbf{V}_1$+$\mathbf{V}_2$. As the motion capture system is not precise enough to capture finger movement, the audience will not be able to see when the performer's hand operates a joystick.

**Virtual Objects Behavior:** It is defined by gestures of the performer. Gestures of the performer can trigger the behavioral animations of other virtual objects, such as allowing objects to appear/disappear, fly, translate, and deform etc., so that the virtual character can push forward the actions in the narrative of the performance through his/her movement.

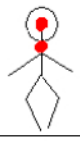
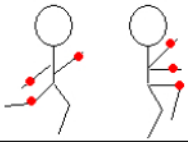
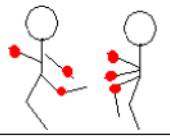
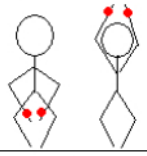
| Gesture | Motion | Action |
|---|---|---|
|  | **Lower head, head** is close to **chest** | **A bunch of plants** appear and grow at back |
|  | **Left leg is picked up, left wrist** is close to **left knee** (distance is less than 5), and **right wrist** is above **left wrist.** | **A bunch of plants** appear and grow on **right-hand side,** stop grow after certain threshold |
|  | **Right leg is picked up, right wrist** is close to **right knee** (distance is less than 5), and **left wrist** is above **right wrist** | **A bunch of plants** appear and grow on **left-hand side,** stop grow after certain threshold |
|  | **Two hands** close up(trigger actions when distance <3) | **Sparkles** coming out of hands |
|  | **Body turn** left/right more than 90 degree | Everything disappear and reset to beginning |

**Figure 4: Behavior description trigged by gestures. Left column: gestures that trigger action. Red dots are the featured body parts that define a certain gesture. Middle column: description of temporal features of each gesture. Right column: object behavior.**

We use the raw output of motion capture data for gesture recognition, including time-varying sequence of parameters describing positions and angles of relevant body joints. Gestures can be defined using not only static body posture (spatial features, e.g. position of body parts) but also temporal features (temporal events, e.g., time-varying pose) without having to train the system. Figure 4 shows a sample description of predefined gestures and its corresponding action in our system. For example, the second gesture is illustrated at Figure 4 and is defined as:

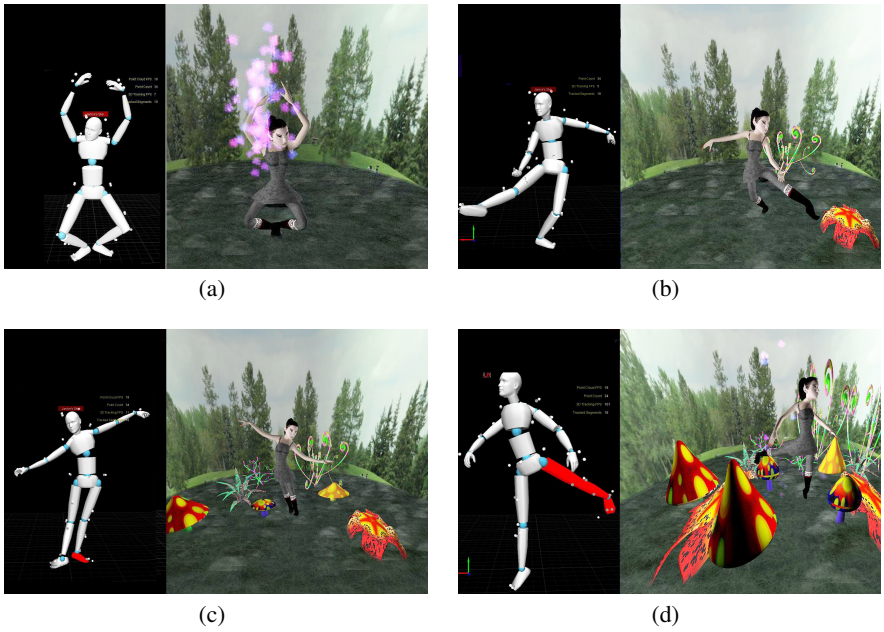


**Figure 5: Dynamic virtual world driven by the actor's performance. In each sub image, left is the skeleton computed from motion capture system, right is the rendered virtual play. Red body parts of the skeleton is caused by occlusion or movement of markers. (a) Hands close-up gesture triggers sparkles. (b) Lifting left hand and left foot triggers plants grow on left-hand side. (c) More plants appears by different gestures. (d) Gestures trigger stretching animation of plants to simulate growing effect.**

$$
\begin{aligned}
ACTION = \quad & ((leftknee, up) && AND \\
& (distance(leftknee, leftwrist) < threshold) && AND \\
& (above(leftwrist, rightwrist)))
\end{aligned}
$$

Each virtual entity in the virtual world includes a set of predefined properties, e.g., engagement conditions **C**, dynamic motion **Q**, geometric description **G** etc. When the engagement condition of a virtual object is satisfied by the motion of the the performer, a dynamic motion of the object is triggered which may cause changes to the virtual environment, such as meshes stretching (to simulate growth). Using the gesture recognition engine, actor can control the virtual character behavior but are also able to push forward the plots of the narrative.

## 4. TRICKSTER AT THE INTERSECTION

We demonstrated a sample application of our system at Smart Graphics 2010 held at Canada's Banff Center. The immersive and interactive play called *Trickster at the Intersection* explores the relationship between the real and the virtual world where a mythological creature called the Trickster plays with the audience uneasiness with the virtual world. Visitors to the performance can interact with the Trickster which at first gives the illusion of been a simple virtual character dancing around and creating his own world but occasionally surprises the audience by truly interacting with them – creating a sense of uneasiness that this Trickster may be real in some way. As the Trickster plays in his magical forest he is able to create whimsical trees, mushrooms, and plants. For example, when the visitor points to an area of the virtual world, the Trickster correspondingly points to the same area and commands flowers to grow in that area to welcome them. The experience brings the audience and the character together at the intersection between the physical and the virtual world. Contrary to the work by Mateas' and Stern's 2005 called "interactive drama" [9], in our system we replace the need for standard input devices (desktop, mouse and keyboard) for a much more expressive and intuitive tool that truly allows the performer to create a sense of presence of a real-virtual entity.

The gesture recognition system can be personalized to fit with the performer style. Only notable, recognizable and definable gestures are coded in our system, so that the behaviors of objects are not trigged by some random motion during the transition of different motion. An action script describing the specified gesture and corresponding behavior is present to the dancer before the show. Figure 4 shows the action script of our current system. Figure 5 shows the virtual play driven by the actor's performance. As shown in Figure 5, the movement of the virtual character is a mirrored motion from the real actor's motion. This is not a problem for the actor as it simulates how he would see himself in front of a mirror. A dancer who has never worked with the motion capture system before can easily understand his task and his role in the performance.

## 5. CONCLUSION AND FUTURE WORK

In this paper, we present a system that enables true real-time interaction between an audience and a virtual character. The character which is animated through the live performance of a real actor in a remote location can be used in a theater setting to add to the narrative of a play and to interact with the audience or with other actors. The virtual character can "see" and "listen" to the user through networked cameras and microphones installed at the theater location. This allow the performer to "respond" in real-time to the reaction of the audience. The short performance *Trickster at the Intersection* presented at the Banff Center was well received as the audience truly felt that they had a real interaction with the trickster. They really felt that the trickers was alive and present on the set. Considering the fact that the connection between the performer and the audience is based on a network connection, one could imagine that

more than one remote performer could be part of a virtual play and could interact with the real-actor present at the theater and the audience. In such a system, it becomes possible for the audience to immerse themselves and become part of the virtual world as the action takes place, and perhaps even choose to interfere with the action and help shape the narrative themselves, thereby becoming part of the performance and creating a truly interactive and collaborative version of virtual theater.

## 6. REFERENCES


[1] S. C. Ahn, I.-J. Kim, H.-G. Kim, Y.-M. Kwon, and H. Ko. Audience interaction for virtual reality theater and its implementation. In *VRST '01: Proceedings of the ACM symposium on Virtual reality software and technology*, pages 41–45, New York, NY, USA, 2001. ACM.

[2] J. Chai and J. K. Hodgins. Performance animation from low-dimensional control signals. *ACM Trans. Graph.*, 24(3):686–696, 2005.

[3] A. D. Cheok, W. Weihua, X. Yang, S. Prince, F. S. Wan, M. Billinghurst, and H. Kato. Interactive theatre experience in embodied + wearable mixed reality space. In *ISMAR '02: Proceedings of the 1st International Symposium on Mixed and Augmented Reality*, page 59, Washington, DC, USA, 2002. IEEE Computer Society.

[4] S. P. Dow, M. Mehta, B. MacIntyre, and M. Mateas. Eliza meets the wizard-of-oz: evaluating social acceptability. In *CHI '10: Proceedings of the 28th international conference on Human factors in computing systems*, pages 547–556, New York, NY, USA, 2010. ACM.

[5] L. Engler and C. Fijan. *Making puppets come alive: how to learn and teach hand puppetry*. Dover Publications, 1997.

[6] G. Giesekam. *staging the screen: the use of film an video in theatre*. Palgrave Macmillan, 2008.

[7] B. Hayes-Roth and L. Brownston. Multiagent collaboration in directed improvisation. In *Proceedings of the First International Conference on Multi-Agent Systems*, pages 148–154, 1995.

[8] Z. Hendricks, G. Marsden, and E. Blake. A meta-authoring tool for specifying interactions in virtual reality environments. In *AFRIGRAPH '03: Proceedings of the 2nd international conference on Computer graphics, virtual Reality, visualisation and interaction in Africa*, pages 171–180, New York, NY, USA, 2003. ACM.

[9] M. Mateas and A. Stern. Facade: An experiment in building a fully-realized interactive drama. In *Game developer's Conference: Game Design Track*, 2003.

[10] M. v. d. P. Matthew Thorne, David Burke. Motion doodle: An interface for sketching character motion. In *Proc. of SIGGRAPH '04 (Special issue of ACM Transactions on Graphics)*, 2004.

[11] W. S. Meador, T. J. Rogers, K. O'Neal, E. Kurt, and C. Cunningham. Mixing dance realities: collaborative development of live-motion capture in a performing arts environment. *Comput. Entertain.*, 2(2):12–12, 2004.

[12] C. P. N. Badler. *Simulating Humans: Computer Graphics, Animation and Control*. Oxford University Press, 1993.

[13] D. Ninomiya, K. Miyazaki, and R. Nakatsu. Networked virtual marionette theater. In *KES '08: Proceedings of the 12th international conference on Knowledge-Based Intelligent Information and Engineering Systems, Part I*, pages 440–447, Berlin, Heidelberg, 2008. Springer-Verlag.

[14] K. Perlin and A. Goldberg. Improv: a system for scripting interactive actors in virtual worlds. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 205–216, New York, NY, USA, 1996. ACM.

[15] C. Reeve. Presence in virtual theater. *Presence: Teleoper. Virtual Environ.*, 9(2):209–213, 2000.

[16] M. Shiba, A. Soga, and J. Salz. A virtual performance system and its application on a noh stage. In *VRST '09: Proceedings of the 16th ACM Symposium on Virtual Reality Software and Technology*, pages 267–268, New York, NY, USA, 2009. ACM.

[17] T. Shiratori and J. K. Hodgins. Accelerometer-based user interfaces for the control of a physically simulated character. *ACM Transactions on Graphics (SIGGRAPH Asia 2008)*, 27(5), 2008.

[18] S. Strassman. *Desktop Theater: Automatic generation of Expressive Animation, PhD thesis*. MIT Media Lab, 1991.

[19] S. Ullrich, H. Prendinger, and M. Ishizuka. Mpml3d: agent authoring language for virtual worlds. In *ACE '08: Proceedings of the 2008 International Conference on Advances in Computer Entertainment Technology*, pages 134–137, New York, NY, USA, 2008. ACM.

[20] K. Yin and D. K. Pai. Footsee: an interactive animation system. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 329–338, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[21] P. Zhao and M. van de Panne. User interfaces for interactive control of physics-based 3d characters. In *I3D '05: Proceedings of the 2005 symposium on Interactive 3D graphics and games*, pages 87–94, New York, NY, USA, 2005. ACM.