Puppet Show: Intuitive Puppet Interfaces for Expressive Character Control

Devin Hunt devin@gatech.edu
Jamie Moore jmore@gatech.edu
Alex West alexwest@gatech.edu

and Michael Nitsche
michael.nitsche@lcc.gatech.edu

School of Literature, Communication and Culture
Georgia Institute of Technology

Abstract

We argue that evolving game art like virtual performances and Machinima depend on better character control. That is why *Puppet Show* does not focus on new or optimized gameplay but aims to extend the expressive range of characters in an existing game world. *Puppet Show* is an interface plug-in for Epic's *Unreal Tournament 2004*. *Puppet Show* uses an external Java program to feed visual input from a webcam onto the character animation of customized *Unreal* game avatars. It adds computer vision as another input device for physical puppeteering of the virtual game characters. Computer vision allows for an intuitive mapping of body movements onto virtual characters and gives human actors better access to an expressive virtual performance. We argue that this exemplifies an interface trend towards more expressive input options that support a higher level of expression in video games.

Keywords

Real-time, animation, video game, machinima, interface design

1. Machinima, Performance, and Puppets

This paper will first outline the goals and context of the *Puppet Show* project. It will point towards the value for such a tool especially for the practices of virtual performance and Machinima. Then, it will highlight the lack of interfaces for subtle animation control in these emerging cultural formats. This lack imposes heavy restrictions upon consumer-level in-game performances and affects their expressive range. We introduce and discuss *Puppet Show* as a project addressing this issue and outline its design philosophy as well as its implementation. The closing remarks will contextualize our work within the current development of video game interfaces.

1.1. Aim

We argue that 3D video games offer a rich platform for artistic expression but also that their range of expression for virtual actors is underused. We see the lack of intuitive interfaces as one reason for this limitation. Consequently, *Puppet Show* is an experimental game project to increase the expressive range and control of players in existing game worlds. The project was conducted at the Experimental Game Lab at the Georgia Institute of Technology. It has two main goals:

1) remove the physical middleware (such as keyboards, mice, and joysticks) and provide users with the ability to map basic physical performance onto their virtual persona
2) provide this kind of interactive access not as a single-standing prototype but as a plug-in for an existing game engine allowing the use of preexistent resources

Puppet Show aimed at robust performance and simplicity. It uses a consumer level webcam as an input device and combines it with the wide-spread *Unreal Tournament 2004* game. Players can download the plug-in together with custom-built characters and gain additional control over the avatars' performances. Basic in-game controls are kept active and players can navigate, jump, or fly through the game world using the usual keyboard/ mouse interfaces that are the standard input devices for *Unreal Tournament 2004* but *Puppet Show* adds the webcam as additional input device for the control of game characters.

The game as such stays intact but the character's performance is enhanced. More control not necessarily improves game performance but enriches the virtual performer's range of expressions and the player's control over them without reducing the in-game functionality. This form of added depth of virtual performance is useful especially for evolving communities that use game worlds to stage their own events.

Players in these communities have started to utilize game engines for the production of Machinima, virtual theater and performances, as well as political demonstrations and art exhibitions. All of these activities grow from pre-existent game worlds and their artistic dynamics but add their own interpretation and use. This development highlights a growing trend towards games as tools for artistic experimentation. But the artistic practice depends on the expressive vocabulary provided by the commercial game. The game has to focus on engaging gameplay; the notion of virtual acting is only one among many components that support this focus.

1.2. Virtual Performance

Despite this tension between goal-oriented gameplay and expressive artistic interpretation, video games have been acknowledged as platforms for virtual performances since the earliest academic work on new media [5, 6]. Since these days, the question of the expression has also been a question of the interface [4, 18].

Technology and expression have been interconnected not only since the dawn of digital media, but the gradual development of computer interfaces illustrates the limitations of new media and their steady adjustments. Insofar, improvements in this area are also a sign for the maturing of the computer as expressive media. Depending on the state of the technology, virtual performances have been realized in a very wide range of computer systems from textual MUD pieces and IRC chats [1], to video-game performances in and for game-worlds [17], to virtual reality theater productions and installations that can use high-end technology to carry out their performance [2]. The more the technology allowed for detailed representation, the more the performance involved virtual actors represented as avatars that operate on the virtual stage (from [9] to [14]).

Assuming a certain level of media literacy, effective acting in a virtual space technically depends on two main factors: the virtual environment and the acting character have to be expressive; and the interface for the inter-acting human has to be appropriate to transmit the necessary subtleties and personality. Accordingly, each virtual performance depends on specific interface metaphors: from text-typing in MUD worlds, to spatial navigation with mouse and keyboard in games, to the whole range of

experimental input devices in the high-end virtual theatre range. With the dawn of 3D video games and their notion of cinematic representation, a new form of performances in virtual worlds has emerged: Machinima.

2. Machinima and Games

Machinima is the technique to create computer-generated imagery in real-time using game technology. Marino defined it as 'Filmmaking + Animation + Game Tech = Machinima' [10]. Because it utilizes 3D games as render engines, it is closely connected to the strengths and disadvantages of the employed engine. On the one hand, Machinima can make use of the game's content, its artwork, its functionality, and more and more of the game's specialized tools for cinematics and real-time image capturing. On the other hand, it often is limited by the game's features, its visual quality, interaction design, animation system, and input options [13]. The technique hovers between an expression of gaming and one that is based on gaming but feeds into other formats. Machinima can be the in-game cut-scene and the player-recorded game event as well as the live performance and the animation shorts that happen to use a game engine, which is barely recognizable operating beneath the customized content.

These 3D game engines allow for high visual details, dynamic lighting, physics, sounds, and ever more elaborate audio-visual qualities. Sadly, the development of interface tools has not kept up with that of the graphic power of modern virtual environments. While the performances have grown from text-based communication to direct actor body control, from MUDs to high-end 3D engines, keyboard and mouse remain the dominating input devices throughout. Game consoles offer more innovative devices but lack access to the code base for necessary adjustments. Computers have become more and more ubiquitous and their expressive power keeps growing, as the rise of Machinima illustrates, at the same time the limitations of keyboards and mice for effective virtual performances become more and more apparent.

Two mayor Machinima game engines – *The Movies* and *The Sims 2* – set a current trend in Machinima, where the actors' movements are driven by Artificial Intelligence and pre-fabricated animation libraries. This excludes the direct control over the specific character behavior and focuses user-control on selected sections, e.g. camera control or editing.

A more traditional approach sees the Machinima producer and player as controller of every depicted event, including the animation sequences. Here, Machinima carries the element of performance in its generation [8] as well as that of puppeteering in many of its character controls [12]. In fact, the element of puppeteering and control has become a topic in itself for some Machinima pieces such as *The Puppet* (Tom "Awaken" Jantol in Half-Life 2, 2005) or *Bot* (Tom "Eggman" Palmer, in *Unreal* Tournament 2004, 2004). Players control the characters either via scripting commands in the game engine or through real-time interfaces in a game session. Control of animation and ease of access co-define the range and expressive vocabulary of these pieces.

Because Machinima is so closely tied to the underlying game engines, the available animations usually cover only game actions (shooting, racing, etc) and interfaces are optimized for the fast action-driven gameplay (mouse for looking around, designated buttons for shooting, jumping, striving). As a result, many Machinima pieces remain within their game engine's dramatic setting and depict variations of typical in-game events. To allow for a wider range of Machinima, we have to ask: How can one leave this state of limited game control and allow for an intuitive yet more subtle input?

The second question has to be: How can access be improved without applying elusive and inaccessible techniques? Machinima praises itself as “computer animation for the masses” because it gives access to highly elaborate Computer Generated Imagery via the easily accessibly video game platform. Its community is still driven by a grassroots relatively low-tech guerilla film-making attitude. Although there are first professional groups consolidating, any tools has to be easily accessible to be of value to the overall community.

3. Related Work

The outlined deficit in interface advancement has created a constricted interface pipeline for actors to project their intended expressions into virtual environments. Adapting current interface artifacts to representative controls has created a muddy interface relationship from actor – to a physical abstract interface – to virtual avatar. This interface relationship has restricted the role and interactive range of the actor and affects the artistry of machinima productions as well as virtual performances and video games. Meaningful *acting* in game environments is restricted and remains on the most rudimentary level [14].

Specialized game interfaces, such as the popular *DDR* floor pads and music instrument-like controllers (from the bongos of *Donkey Konga* to the maracas of *Samba De Amigo* to the guitars of *Guitar Hero*) provide the gamer with an intuitive interface for specific gameplay. Sadly, specialized interfaces like these can rarely be extrapolated outside of their original purpose. Even within subgenres specialized interfaces can rarely be interchanged. These context-dependent interfaces rarely offer direct control over animation or character attributes and therefore add little to the animation range of in-game characters. Instead, they concentrate on a specific interface experience for the human performer in order to optimize the gameplay.

Addressing the need for more detailed animation control, various researchers have worked in the area of interface design for real-time animation control that maps physical “body” action onto virtual characters. The following are a selected pioneering few of the many possible reference points.

3.1. Sound and Touch

Since the 90ies, Ken Perlin has worked on procedural real-time character animation [15]. During the 2005 Machinima Festival he presented a variation of his *Peepz!* project with two avatars acting out an improvised dramatic scene – both controlled by a human player using a music keyboard. The keyboards not only generated the musical score but also controlled the character animation and offered a fresh form of virtual character performance.

Replacing the music input with voice, Nakatsu/ Tosa/ Ochi’s *Romeo and Juliet in Hades* analyzed the emotional tension in the human player’s voice and mapped it on the behavior of the virtual avatar [11]. *Romeo and Juliet in Hades* is a high-end interactive movie production that also applies large projection screens and motion tracking. Still, a direct mapping of a certain voice timbre onto a related animation was difficult to control by the human actor.

The *Synthetic Characters Group* at MIT worked with a number of puppet-like interfaces, for example, in the *SWAMPED* project. Here, the group around Bruce Blumberg used a physical bird puppet as input device and main controller. At the same

time, they traced the virtual avatar's emotional state and adjusted the camera work accordingly [20]. Such a combination of character control, emotional avatar state, and cinematic presentation offers interesting venues for Machinima, but it concentrated on the agent behavior more than on the visual expression. MIT's *Tangible Media Group* is even more focused on the development of physical interfaces (see e.g. [16]) albeit with even less focus on a cinematic presentation.

3.2. Computer Vision

Mainstream computer vision interfaces have been becoming more widely available especially within the realms of interactive gaming, the Sony *Eyetoy*™ for their Playstation 2 console being the most prevalent. The *Eyetoy*™ Labanotation study conducted by Loke/ Larssen/ Robertson examined the range of motion that the motion games required and how gamers fulfilled the interactive cues presented to them [7]. This indicates that players are indeed willing to engage in a highly emotional performance for such an interface.

But most vision-based games are also reaction-based, thus, almost all interaction involves the user moving a limb into a specified quadrant on the screen in a timely manner. In this regard, the camera directs the player more than the player the virtual character. Still, the Eyetoy™ motion tracking system and the associated games excel because they are both simple and subsequently so intuitive that they encourage a meaningful input from the player. For the world of Personal Computers, the webcam has become extremely wide-spread, technically reliable, and affordable device.

Most of the aforementioned projects are unique experiments, often depending on expensive hardware and prototype software, none ties into pre-existing game engines. Their interfaces can offer very expressive bridges between the physical world of the performing actor and the virtual one of the expressing avatar, but they remain unique and highly specialized. Their agenda was not an increase of expression in commercial video games – as it is the case with *Puppet Show*.

In order to remain affordable and simple, yet intuitive *Puppet Show* had to use existing and easily accessible hard and software. Otherwise, the aim of increasing the general expressive range of an existing game engine would be compromised. Thus, the main interface was a standard webcam and a Java-application that feeds into a basic modification of the *Unreal* engine's characters.

4. The Puppet Show Project

4.1. Concept

The design philosophy of *Puppet Show* was to increase the expressive range of an existing game engine by mapping basic physical actor input onto the virtual performer. The original paradigm we applied is that of a hand puppeteer, but – in fact – it extends beyond any single puppet interface.

Hand and body puppetry establishes an interface relationship from the actor – to a physical puppet – to a virtual avatar. Such a relationship supports a more organic interface: each facet mimics the other. For example, Hoysniemi has shown the effective connection of movement and virtual puppet behavior in studies with children [3].

Because the actor controls the virtual avatar with intuitive cues, a richer projection of expressive qualities is easier. A more organic interface spawns a more

organic expression. So far, this mapping was provided in high-tech laboratories utilizing expensive hardware and exclusive software, which would not feed into the existing game community. In the case of Machinima most of these experimental projects left little immediate marks in the community.

That is why *Puppet Show* aimed at a basic yet very flexible interface improvement. By emphasizing mimicry between simple hand puppetry and avatar animations users can quickly form intuitive links between the two elements. *Puppet Show* subsequently allows users to map otherwise complex expressive animations onto virtual avatars through simple physical motion.

4.2. Content

The prototype setting was loosely modeled after the two grumpy old men, Waldorf and Statler, of *The Muppet Show*. They never leave their seats in the muppets' theater but never stop commenting on the ongoing show below. They express themselves mainly with posture, arm, and head movements and their witty dialogue. In some way, they are the opposites to the athletic fast-paced action heroes of the original *Unreal Tournament 2004* game. *Unreal* supports mainly spatial movement such as running, jumping, crouching. Sound capabilities are also available through the included voice chat feature. But *Unreal* does not allow the more subtle character control needed for two grumpy old men. Thus, our prototype focused on the control of a character's head, mouth, and arms.

We created a sample level as well as customized characters but a main feature of *Puppet Show* is the fact that it ties seamlessly into the *Unreal* game world. The entire game content stays intact and can be used as a backdrop for the new-found character control. Because *Unreal* is a preferred game engine for Machinima creators, *Puppet Show* can directly transfer into their production circles.

4.3. Implementation

4.3.1. Overview

Puppet Show is a composite of three major elements: the physical puppets or any color-coded physical feature, the computer vision interface, and the *Unreal* modification. The interaction of all these units produces a fluid interface for real time avatar animation. Its components were designed with simplicity and flexibility as the utmost priorities.

Because the *Puppet Show* server operates over an Internet link, both the server and the *Unreal* client can function on a single computer and the system can be run on one consumer-level computer with a standard webcam attached. The most common setup, though, uses two computers – one running the *Puppet Show* server and the other running a modified *Unreal* client. This model has been most effective for the performance itself, because it provides the best overview of both the webcam interface and the resulting character animations to the player.

We chose color blob tracking as our computer vision standard instead of the popular motion tracking, the method used in the *Eyetoy*™ games. Blob tracking is unique in its ability to identify multiple points of interest based solely on the apparent color of the surface. *Puppet Show* includes a calibration tool. Within seconds, players can calibrate any colored object to be traced by the system. As a result, *Puppet Show* leaves a great deal of freedom to the human performers in regard of what exactly they want to use as control elements. Colored gloves, t-shirts, paper puppets, any simple colored object can be used to control animations. In that way, *Puppet Show*'s design doctrine encourages users to tailor puppets to his or her expressive needs.

Video captured by the webcam is then translated into abstract values and transmitted to the connected modified *Unreal* client. The *Unreal* client translates the received values into bone transformations and subsequently alters the 3d character.

4.3.2. Computer Vision and Blob Tracking

Puppet Show uses a very selective blob tracking system for accurate object tracking. First the image is scaled down so that we can perform an image analysis on it without using too much processing power. Then the system filters out all the colors that are farther than the calibrated threshold in a color cube. Each color value represents a vector in its own dimension. A three dimensional distance is calculated and if that distance is greater than the threshold, the pixel is turned black. Then, the image is further filtered on luminosity using the V3GA Filter.

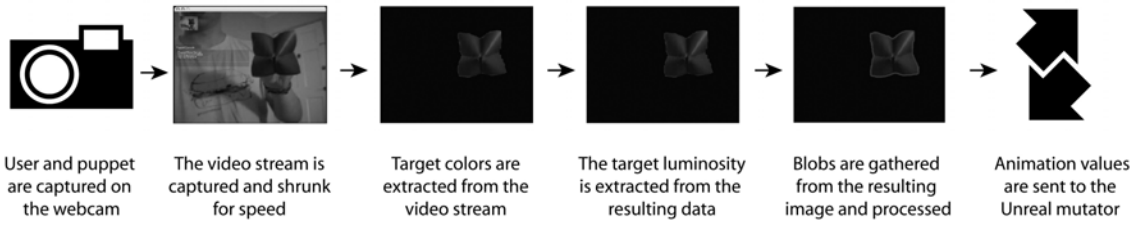

Figure 1 – Color tracking in *Puppet Show*

The resulting image contains only the significant surfaces of the puppetry object and can easily be selected.

4.3.3.Calculations and Manipulations within Unreal

Once the information about the color blobs is calculated, the system has all the information needed to correctly modify the avatar's bones. The information is formatted in a unique string and sent to an *Unreal* Client over a TCP/IP connection. The client parses the data from the string and stores all the data. The motions are applied individually to their appropriate bones, so that all other character controls are not affected. This allows complex movements because multiple actions can be applied to the skeleton at the same time. Players can move their virtual characters while controlling their head movement, for example.

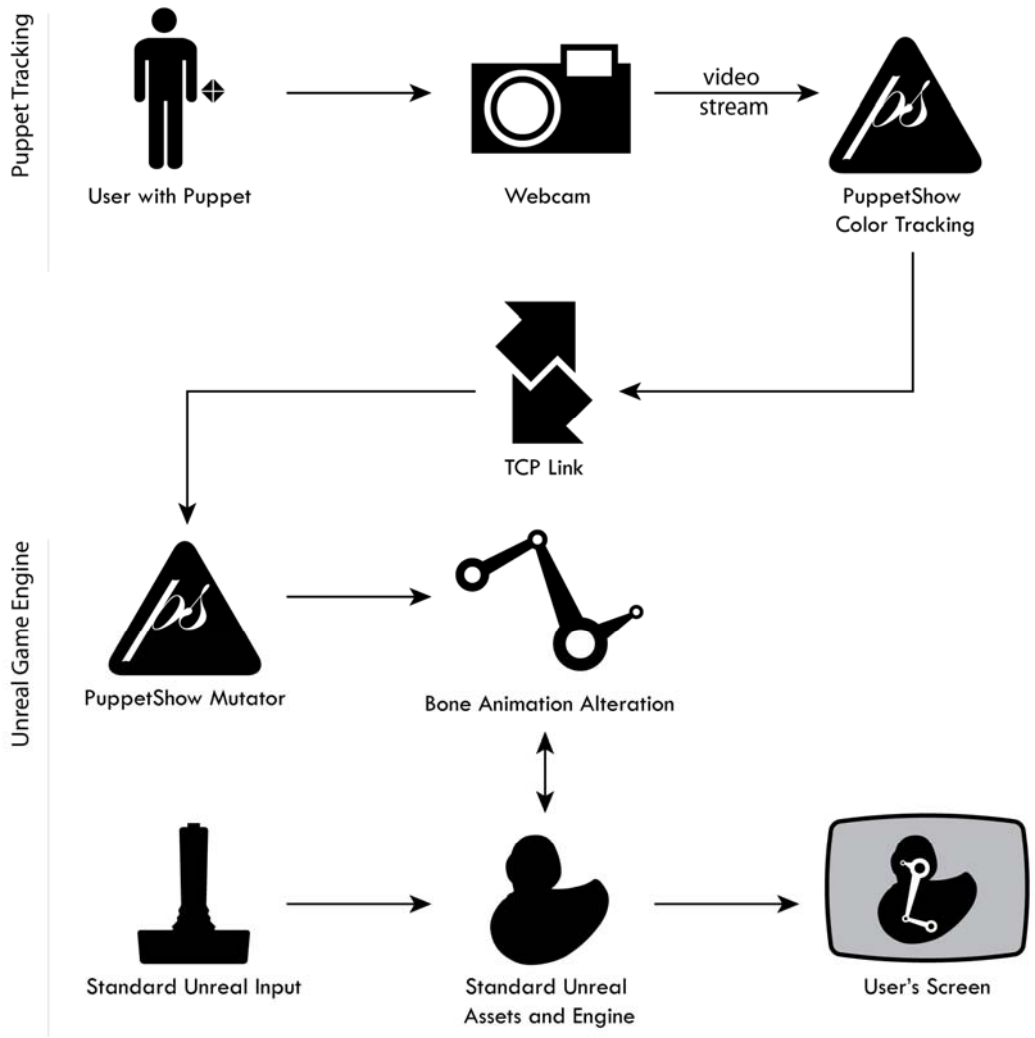

Figure 2 – The *Puppet Show* pipeline from visual input to in-game effect in a live performance

A video stream of the user with the puppet is captured by a generic webcam and fed into the *Puppet Show* color tracking system. The resulting animation values are fed into the *Unreal* mutator via a TCP stream. The mutator affects the avatars animation skeleton directly, keeping any animation from the native game engine untouched. This new animation skeleton is then passed back to the *Unreal* system for rendering on the user screen.

4.4. Result

In order to demonstrate *Puppet Show*, we modeled two characters – a duck and a panda bear – as well as a basic experimental game level. We imported them into *Unreal* and mapped different animation controls onto the characters' skeletons.

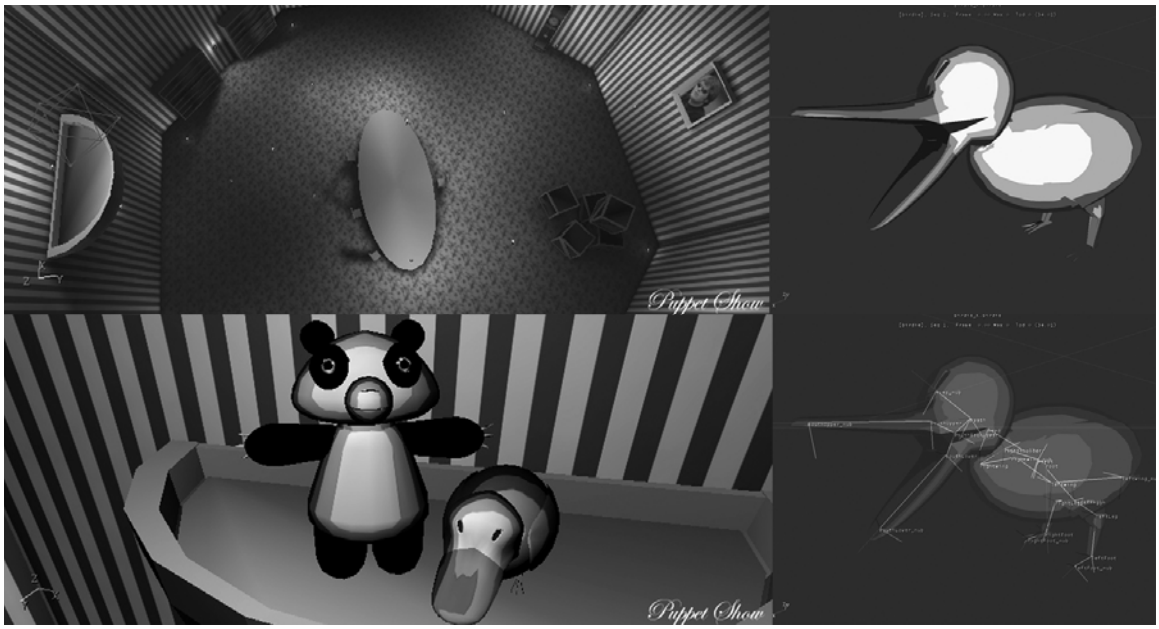

Figure 3 – The arena/ stage (top left); the two customized characters on their "virtual balcony" in the example level of (left); the customized skeleton structure of one character (right)

A number of problems remain unresolved: the computer vision interface does not yet apply automatically affect all *Unreal* characters but needs specialized skeleton structures. The range of animations is also still limited. One cannot control every aspect of the character via the webcam input, but only selected ones.

Still, *Puppet Show* provides a relatively robust, functional additional avatar control tool for players and performers using the *Unreal* game engine. It works with all the game content, the 3D worlds, the sound and chat options, the multi-player set-up, the visual effects of the game, in short: with the whole expressive palette of the video game. As such, *Puppet Show* adds its share to the expressive vocabulary of the *Unreal* engine and extends its value as a virtual performance tool for the emerging field of Machinima.

5. Conclusion and outlook

The overall goal of *Puppet Show* was to increase the control of a human performer over a virtual character in an existing game engine by offering more physical input options. It chose a color-tracking computer vision interface and plugged it successfully into the *Unreal Tournament 2004* game via custom-built character skeletons. The result is an additional level of character control and a higher expressive range for the player.

Puppet Show is a step into a much needed direction. We strongly believe that the current state of character controls in video games is insufficient for more elaborate acting as it is needed in the production of Machinima. We will provide *Puppet Show* as a free-downloadable tool to the community to allow them to play, improve, and express themselves better in digital worlds – but also to alter and improve the tool itself. This follows the tradition of player-generated tools for more effective Machinima production. Like all of these projects, *Puppet Show* has to grow in future iterations and needs some further improvements.

The realities of the game engine force limitations to the mapping of physical onto virtual performance and form obstacles to overcome. The limited range of commercial interfaces is one of these obstacles. We see a promising development in this field e.g. in the light of recent and upcoming game console interfaces. Nintendo's Wii controller as well as the motion control features in Sony's PS3 controller both allow for more physical input options.

Another restriction resides in the animation systems of existing game engines. Too often they rely on pre-fabricated animation sequences and lack more freedom in the avatar control. Again, there have been promising improvements in the form of increased use of physics in character animation. *Unreal*'s use of Karma physics and its rigid body simulation, or the physics implemented in Valve's *Source* engine both directly affect the character animation. So far, this too often plays out only in a character's death sequence, but the technical improvements pave the way for a more elaborate animation system.

Finally, game engines begin to include features such as lip-synch and a character's gaze. Such an development towards more elaborate real-time virtual actors, implies also improvement of overall gesture and therefore the control structure of these gestures. An agitated character that gives a perfectly lip-synched speech is also expected to show the according body movements with an equal level of detail. The depiction of emotion will be a constant challenge for game developers and call for innovative interface metaphors.

All of these developments allow for a richer virtual performance in game engines, which in return should have positive impact on the quality of Machinima and virtual performances. Yet, utilizing these new options is still too rare. Springel [19] proposed a future of virtual dramatic environments for self expression – if that is the goal, then *Puppet Show* offers a model for a simple yet effective mapping of expressions into a virtual, dynamic world.

References

1. Danet, B., Wachenhauser, T., Cividalli, A., Bechar-Isreali, H., Rosenbaum-Tamari, Y. Curtain Time 20:00 GMT: Experiments with Virtual Theater on Internet Relay Chat. *Journal of Computer-Mediated Communication*, 1995, 1 (2)
2. Giannachi, G. *Virtual Theatres: An Introduction*. Routledge, London, New York, 2004.
3. Hoeysniemi, J., Haemaelaeinen, P. and Turkki, L. Wizard of Oz Prototyping of Computer Vision Based Action Games for Children *IDC 2004*, ACM Press, College Park, 2004, 27-34.
4. Laurel, B. *The Art of Human-Computer Interface Design*. Addison-Wesley Publishing Company, Reading, 1990.
5. Laurel, B. *Computers as Theatre*. Addison-Wesley Publishing Company, Reading/ Mass, 1991.
6. Laurel, B. Toward the Design of a Computer-Based Interactive Fantasy System, Ohio State University, 1986.
7. Loke, L., Larssen, A.T. and Robertson, T. Labanotation for Design of Movement-Based Interaction *Second Australasian Conference on Interactive Entertainment*, ACM Press, Sydney, 2005, 113-120.
8. Lowood, H. Real-time Performance: Machinima and Game Studies. *The International Digital Media & Arts Association Journal*, 2005, *2* (1). 10-18.

9. Magnenat Thalmann, N. *Synthetic Actors in Computer-generated 3D Films*. Springer-Verlag, Berlin/ Heidelberg/ New York, 1990.
10. Marino, P. *3D Game-Based Filmmaking: The Art of Machinima*. Paraglyph Press, Scottsdale, AZ, 2004.
11. Nakatsu, R., Tosa, N. and Ochi, T. Interactive Movie System with Multi-Person Participation and Anytime Interaction Capabilities. Nikatsu, R., Altman, E. and Pinhanez, C. eds. *Sixth ACM International Conference on Multimedia: Technologies for Interactive Movies*, ACM Press, Bristol, 1998, 129-137.
12. Nitsche, M. Film live: An Excursion into Machinima. in Bushoff, B. ed. *Developing Interactive Content: sagas_sagasnet_reader*, High Text, Munich, 2005, 210-243.
13. Nitsche, M. and Thomas, M. Play it again: Film Performance, Virtual Environments and Game Engines. in Beardon, C. and Carver, G. eds. *New Visions in Performance: The Impact of Digital Technologies*, Swets & Zeitlinger, Lisse, 2004, 121-139.
14. Perlin, K. Can there be a Form between a Game and a Story? in Wardrip-Fruin, N. and Harrigan, P. eds. *First Person: New Media as Story, Performance, and Game*, MIT Press, Cambridge, MA, 2004, 12-18.
15. Perlin, K. and Goldberg, A. *Improv: A System for Scripting Interactive Actors in Virtual Worlds*. ACM Press, New York, 1996.
16. Ratti, C., Ishii, H., Frenchman, D., Wang, Y. and Piper, B. Tangible User Interfaces (TUIs): A Novel Paradigm for GIS. *Transactions in GIS*, 2004, *8* (4). 407-421.
17. Salen, K. Telefragging Monster Movies. in King, L. ed. *Game On. The History and Culture of Videogames*, Laurence King Publ., London, 2002, 98-112.
18. Slater, M., Howell, J., Steed, A., Pertaub, D.-P., Gaurau, M. and Springel, S. *Acting in Virtual Reality*. ACM Press, New York, 2000.
19. Springel, S. *'The Virtual Theatre' Immersive Participatory Drama Research at the Centre for Communications Systems Research, Cambridge University*. Nikatsu, R., Altman, E. and Pinhanez, C. eds. *Sixth ACM International Conference on Multimedia: Technologies for Interactive Movies*, ACM Press, Bristol, 1998, 43-49.
20. Tomlinson, W.M., Jr. Interactivity and Emotion through Cinematography *Media Lab*, Massachusetts Institute of Technology, 1999.