SmurVEbox: A Smart Multi-User Real-Time Virtual Environment for Generating Character Animations

Rüdiger Beimler,* Gerd Bruder,† Frank Steinicke‡
Immersive Media Group
Department of Computer Science
University of Würzburg

ABSTRACT

Animating virtual characters is a complex task, which requires professional animators and performers, expensive motion capture systems, or considerable amounts of time to generate convincing results. In this paper we introduce the SmurVEbox, which is a cost-effective animating system that encompasses many important aspects of animating virtual characters by providing a novel shared user experience. SmurVEbox is a collaborative environment for generating character animations in real time, which has the potential to enhance the computer animation process. Our setup allows animators and performers to cooperate on the same virtual animation sequence in real time. Performers are able to communicate with the animator in the real space while simultaneously perceiving the effects of their actions on the virtual character in the virtual space. The animator can refine actions of a performer in real time so that both collaborate together on the same animation of a virtual character. We describe the setup and present a simple application.

Categories and Subject Descriptors

I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism—*Virtual reality*; H.5.1 [**Information Interfaces and Presentation**]: Multimedia Information Systems—*Artificial, augmented, and virtual realities*

General Terms

Computer Graphics, Computer Animation

Keywords

Virtual Reality, Character Animation, Real-Time, Motion Capture, Multi-Touch, Collaborative Environment

*ruediger.beimler@uni-wuerzburg.de
†gerd.bruder@uni-wuerzburg.de
‡frank.steinicke@uni-wuerzburg.de

Laval Virtual VRIC'13, March 20-22, 2013 Laval, France

1. INTRODUCTION

Three-dimensional (3D) computer animation is the process by which sequences of computer-generated images of 3D scenes are created, which give the illusion of moving objects when viewed sequentially [15]. In essence, computer animation is a digital successor to the stop motion techniques used in traditional animation with 3D models and frame-by-frame animation of 2D illustrations. Computer-generated animations provide more control and more flexibility at the cost of increased complexity of specifying the movement of objects through a scene [15]. In this context, the importance of real-time animation of virtual character has increased considerably over the past decade. However, defining animations of virtual characters is still a complex task. Usually, professional animators and performers work in and with expensive setups, e. g., motion capture systems, and professional software, resulting in a considerable amount of time until a realistic motion of a virtual character can be achieved [15, 19]. Although photorealism is often not necessary, many different aspects of behavior need to be considered if a virtual character should appear recognizably natural in expression [19].

Humans are adept at recognizing physically plausible movements and behavior, so the actions and appearance of a virtual character must match the expectations of the observer [17]. Hence, movements of a virtual character must be natural as well as contextually appropriate in such a way that they respond with the appropriate reaction and in the proper time frame to stimuli [19]. There is a large body of research on how various aspects of character animation, such as locomotion and facial animation, can be generated from an algorithmic perspective [9, 15]. However, the integration of all movement and behavior aspects into one animation leads to enormous complexities. 3D scenes and virtual characters are modeled traditionally with dedicated 3D modeling and animation applications and 3D models are rigged with virtual skeletons, and animated via keyframing or motion capture approaches [15]. The resulting motion sequences are usually refined and improved by the animator to complete the illusion of natural motion [29]. Although several 3D modeling applications provide sophisticated interfaces for specifying the models and their movements, describing animations is still a very time-consuming task [9, 15, 19].

In this paper we introduce the SmurVEbox, which provides a novel user experience for animators as well as performers and supports them while defining computer animations of virtual characters. The SmurVEbox is a collabo-

rative environment for generating computer animations in real time, which has the potential to enhance the computer animation process as well as the naturalness of a character's actions and movements within a virtual environment (VE). This approach is realized by combining several cost-effective technologies and techniques. For instance, movements of a performer are tracked by a Kinect-based interface, streamed and mapped onto rigged virtual characters within a VE. The interactions of a second performer or operator are captured via a touch-based interface and are applied to the virtual characters within the same VE. All users receive real-time audio-visual feedback via different virtual reality (VR) displays and can review their actions or movements in a shared virtual space.

The remainder of this paper is structured as follows. Section 2 resumes background information and related work. Section 3 introduces the SmurVEbox framework. Section 4 describes a use case of our setup. Section 5 concludes the paper and gives an overview of future work.

2. BACKGROUND

In this section we resume background information on techniques and technologies for generating and obtaining realistic computer animations of virtual characters.

2.1 Virtual Characters

For specifying 3D computer animations of virtual characters, usually the animator has to create a simplified representation of a character's anatomy. This process is referred to as *rigging*. The resulting hierarchical set of interconnected bones is called *skeleton* or *rig* [4, 7, 15]. *Skeletal animation* is the technique in which the virtual character is represented in two parts: (i) a surface representation used to draw the character (called *skin* or *mesh*) and (ii) the skeleton used to animate this mesh [2]. For virtual characters, parts of the skeletal model may correspond to actual bones, but skeletal animation is also used to animate other objects and parameters, such as facial features [16].

In addition there are a number of approaches for smooth-binding the skin objects to the underlying skeleton joints [12]. This process needs to be done prior to object animation, whether by hand or automated via software algorithms.

Thereafter, the position and orientation of each segment of the skeletal model is controlled by *animation variables* (so-called *avars*) [15]. The computer uses the skeletal model to compute the exact position and orientation of the character based on the values of the avars. Thus, by changing the values over time, the animator can specify motions of the character from frame to frame. Typical characters use up to 1000 avars, which often result in rather complex control procedures [15].

2.2 Computer Animation Techniques

Traditionally, animators set the avars directly, either for every frame or at strategic points (*keyframes*) in time and let the computer interpolate or "tween" between them, a process called *keyframing*. In order to specify the animation states, *forward kinematics* can be applied, in which the poses of parts of the model are calculated from hierarchical joint information of an articulated model. In contrast, with *inverse kinematics* the orientation of articulated parts is calculated from the desired pose of end-effectors in the model hierarchy. An alternative or supplement to keyframing is called *motion capture*, which makes use of live action tracking. When computer animation is driven by motion capture, a real performer acts out the scene as if he or she were the character to be animated. The performer's motions are recorded and applied to the virtual character. Both keyframing and motion capture have advantages and limitations. Keyframe animation can produce motions that would be difficult or impossible to act out, while motion capture can reproduce the subtleties of a particular actor. Thus motion capture is appropriate in situations where believable, realistic behavior and action is required.

Since both methods – keyframing and motion capture – require trained professionals and various efforts in setting up the production environment, more intuitive workflows were evolved [26]. With this software-implementation artists are able to produce results without the need for time-consuming training. More recent approaches in simplifying the animation procedure via interactive systems were for example in the fields of manipulating shapes in two-dimensional images [10] or by setting up postures in three-dimensional space and interactively animate the already set-up character with a control cursor [11].

2.3 Motion Capture

Motion capture has established itself as the dominant technique for computer animations in movie and game production. In order to capture physical movements of a performer, different full-body tracking technologies can be exploited. Motion capture technologies can be classified into inside-in (e. g., [13]), inside-out (e. g., [20]) or outside-in (e. g, [30]) systems. At the moment, the most popular method are optical outside-in tracking systems, which can be very precise, although they suffer from occlusion issues, are often expensive and usually not portable. Recent advances in the field of markerless outside-in tracking technologies, e. g., the Microsoft Kinect, lead to new possibilities for low-cost real-time performance tracking. While professional tracking systems provide superior precision and accuracy [18], with the markerless, inexpensive full-body skeleton tracking of the Microsoft Kinect it is now possible to get real-time 3D data about a performer's body and its pose that allows animators to capture natural and realistic 3D movements. Moreover, multiple users can simultaneously be tracked and distinguished by the Kinect, which provides new possibilities for computer animation environments.

2.4 Stereoscopy and Multi-Touch

Interactive design and review of 3D models and character animations requires accurate spatial perception of 3D geometry, which is often not sufficiently supported in traditional Desktop environments [8, 21], and results in a low learning curve for new modelers and animators. Head tracking and stereoscopic visualization have often been found to improve spatial perception and interaction in 3D VEs [25], but introduce challenges when interacting with 2D mouse input [22, 24]. Multi-touch interaction has recently received considerable attention due to the potential of intuitive and near-natural interaction with mono- and stereoscopic objects relative to display surfaces [5]. Multi-touch surfaces support input with multiple fingers and/or hands, which can be realized with various technologies, such as capacitive sensing or analysis of infrared (IR) or color images [1]. Although

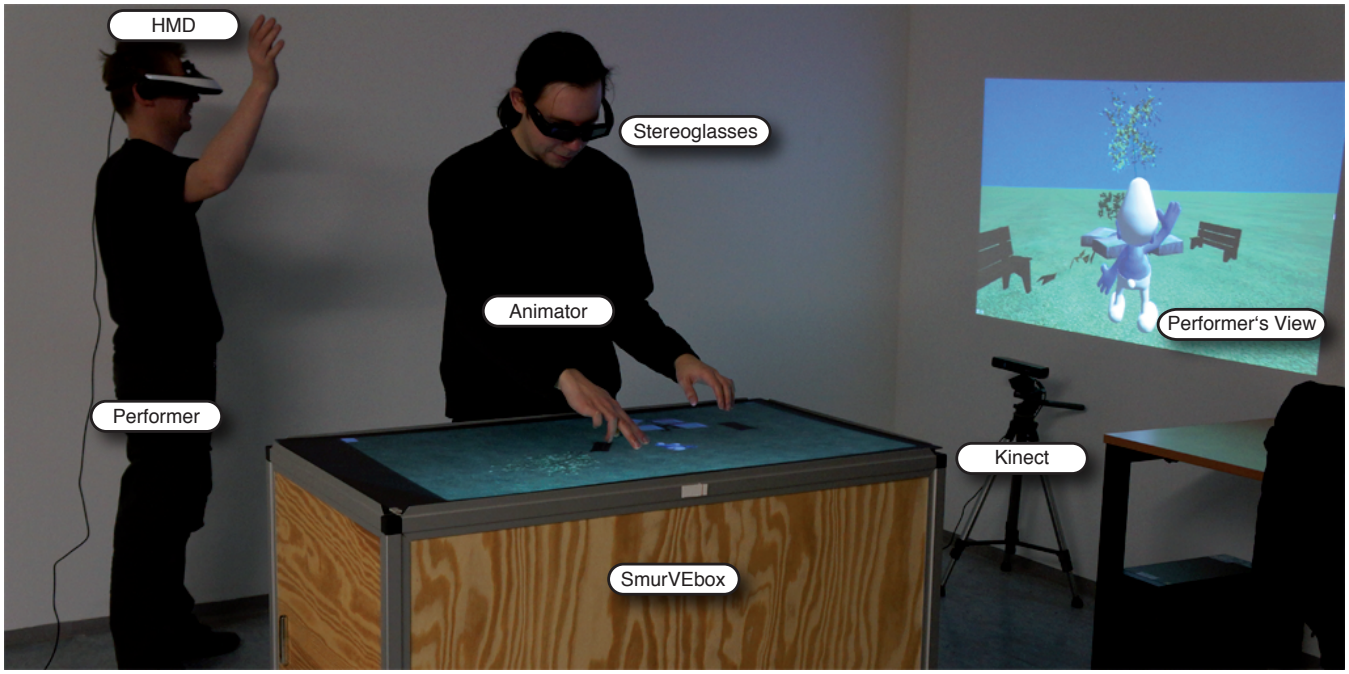


Figure 1: Illustration of the SmurVEbox setup: A performer is immersed in the virtual world and perceives his actions from an ego-referenced perspective on a head-mounted display (HMD). The performer's view is also displayed on a projection screen for the animator's reference. The animator interacts with the SmurVEbox via multi-touch gestures to initiate or control animations of characters or objects in the virtual scene.

touch-input is limited by the physical constraints of the touch surface, users do not have to use obstructive devices for interaction, such that these technologies can provide an unencumbered solution for intuitive and natural interaction. The ability to directly touch graphical elements while getting passive haptic feedback with collocated or augmented visual-motor responses about touch interactions from the touch surface has been shown to be very appealing for novice as well as expert users, and has the potential to improve interaction in virtual and mixed reality setups [27].

3. SMART COLLABORATIVE REAL-TIME ANIMATION SYSTEM

This section describes the hard- and software setup used for the implementation of the developed collaborative real-time animation setup.

3.1 Design Decisions

The basic considerations for the development of our collaborative animation system were:

- The setup should allow animators as well as performers to collaborate in the same setup, supporting interactions and assignments of duties between them for the collaborative generation of complex character animations in a single virtual interaction space.
- Live motion capture and preview should be supported to map biomechanical motion onto one of multiple virtual characters. Additionally, animations should not be limited to motion capture, but provide artistic freedom to integrate additional interaction modes, e. g., based on direct touch interactions.
- Live feedback should be provided to performers as well as animators such that they can evaluate and readapt their performance in response to dynamically changing VEs in real time.
- Following a rapid prototyping approach instead of aiming for highest accuracy and precision of animation recording, inexpensive hard- and software solutions are preferred over professional integrated solutions.

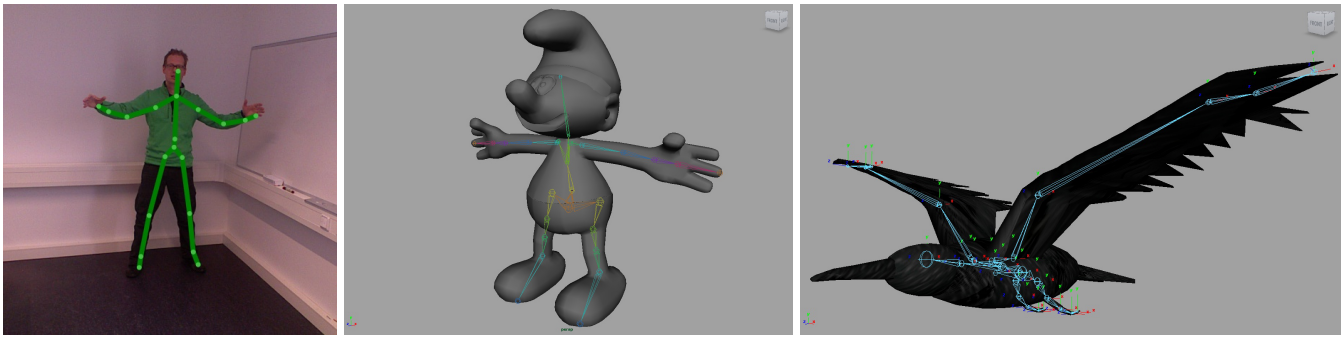
Basically, there are two different paradigms for the animation system. First, the animator can specify "global" animation parameters of characters or the scene on the SmurVEbox, whereas a performer can define motions of a virtual character on a "local" basis (see Figure 1). The SmurVEbox is based on the *smARTbox* approach [6], which combines stereoscopic visualization with multi-touch interaction in a portable box. In the following sections we describe all extensions that we applied to the hardware setup as well as the software implementations with which we addressed the design considerations described above.

3.2 Hardware Setup

This section describes the hardware components of the SmurVEbox. The entire setup was built at a total cost of less than USD 5,000.

3.2.1 Stereoscopic Visualization

The top side of the SmurVEbox consists of a 62 cm × 112 cm back projection screen with a gain of 1.6. For stereoscopic display we use an Optoma GT720 projector with a resolution of 1280 × 800 pixels at a refresh rate of 120Hz, which supports active stereoscopic display with inexpensive



(a) (b) (c)

Figure 2: Examples for skeleton re-targeting with (a) joint positions tracked by a Microsoft Kinect, as well as rigged virtual character models of (b) a smurf and (c) a raven.

DLP-based shutter glasses. The projector has a wide-angle converter lens. The image is projected from the projector in the base of the table to the back projection screen via a mirror that is mounted at an angle of 45 degrees. The left and right eye of a user receive different views to the same virtual scene rendered from slightly different perspectives which are generated from the eye positions of the viewer in front of the display surface. We either use a Microsoft Kinect to track the user's head pose while interacting with the SmurVEbox, or switch to a marker-based iotracker optical tracking system for improved performance. Using active shutter glasses, the images are displayed frame-sequentially to the eyes of a user with a rate of 60Hz per eye. Virtual content can be displayed with different stereoscopic parallax, i. e., negative, zero or positive parallax, resulting in objects appearing in front, on top, or behind the display surface, which enables out-of-box interaction concepts as described in Section 4. The virtual environment is rendered on an Intel Core i7 computer with 3.40GHz processors, 8GB of main memory, and an nVidia Quadro 4000 graphics card.

3.2.2 Multi-Touch Detection

We converted the back projection screen at the top of the SmurVEbox to a touch-sensitive input surface using Rear Diffused Illumination (Rear-DI) [14]. Six clusters of high-power IR LEDs illuminate the entire screen from below. The acrylic glass surface serves as diffusing layer. In the case an object comes in contact with the surface, e. g., a finger or palm, it reflects the IR light back to the base of the SmurVEbox. The reflected light is captured by a PointGrey Dragonfly 2 camera, which we equipped with an IR band-pass filter and a wide-angle lens. The camera captures 8-bit monochrome images with a resolution of 1024×768 pixels at a rate of 30 frames per second. We make use of a modified version of NUI Group's Community Core Vision (CCV) software[1], which receives the video stream from the camera to detect touch points. Detected touch points of one or multiple finger as well as fiducials are provided to applications via the TUIO[2] protocol, which is widely used with tangible multi-touch surfaces.

3.2.3 Motion Capture with Kinect

As illustrated in Figure 1, we make use of a Microsoft Kinect for real-time tracking of performers interacting with the setup. The Kinect provides full-body tracking data at a frequency of 30Hz. This allows us to track the approximated head position and orientation of one or multiple users wearing Sony HMZ-T1 HMDs or using a stereoscopic projection screen as reference. The tracked head pose can then be used to provide head-coupled rendering and the generation of stereoscopic views. Full-body skeletal information of the performers immersed in the virtual scene is tracked via the Kinect and mapped to virtual characters displayed on the SmurVEbox to provide an animator with the ability to develop animations based on scene-aware virtual characters, i. e., reacting and responding to actions in the VE and the state of a user's body in the real world.

3.3 Software Implementation

The software implementation of the SmurVEbox is based on Microsoft Windows 7 and a compilation of software libraries and frameworks, which are explained in the following sections.

3.3.1 Virtual Environment

We make use of Unity Technologies' proprietary Unity 3D game engine for the generation and audio-visual rendering of virtual environments to be used in the SmurVEbox setup. Unity 3D provides a simple development environment for virtual scenes, animations and interactions, which supports multiple cameras and viewports, as well as a broad range of game assets, dynamic geometry and rigged characters. In order to synchronize virtual camera objects with the movements of animators or performers using the physical setup, we integrated the MiddleVR for Unity software framework[3]. MiddleVR supports streaming of full-body Kinect tracking data to Unity 3D, as well as streaming of optical marker tracking information from our iotracker system using the Virtual Reality Peripheral Network (VRPN) protocol [23]. Head-coupled rendering parameters are computed by MiddleVR and streamed via a plugin to Unity 3D for each user's virtual scene cameras.

3.3.2 Touch Interaction

We map multi-touch gestures from the responsive touch surface of the SmurVEbox setup to Unity 3D by using the

[1] http://ccv.nuigroup.com/
[2] http://www.tuio.org/
[3] http://www.imin-vr.com/middlevr/

community edition of xTUIO's uniTUIO[4], which is a simple script library for streaming TUIO events. The library interprets received TUIO touch events and maps the corresponding touch coordinates into the screen space of the Unity 3D game environment. We implemented single-finger touches to initiate object selections and translations in the virtual 3D scene, as well as two-finger pinch gestures for rotations and scalings (see Figure 3). By controlling multiple degrees-of-freedom continuously and simultaneously using multi-finger and multi-hand input, users can animate multiple virtual objects at the same time.

3.3.3 Full-Body Animations

We use the Microsoft Kinect for Windows SDK for markerless skeletal tracking of one or multiple performers in the SmurVEbox environment. To access this data from within the Unity 3D virtual environment, we integrated the Kinect Wrapper Package that was released by the Entertainment Technology Center of Carnegie Mellon University. We generated and adapted rigged virtual characters using the Autodesk Maya software by placing Joint-Deformer Objects to shape a Bipedal Skeleton and by binding the mesh geometry to the appropriate joints via smooth skinning (see Figure 2). We imported the resulting rigged characters as assets into Unity 3D and mapped the Kinect's skeleton information to the joints of animated scene characters in real time.

4. USE CASES: ANIMATING VIRTUAL CHARACTERS

In this section we report observations that we made while animators and performers used our setup for generating several animations of virtual characters such as a smurf or a bird.

4.1 Procedure

As illustrated in Figures 2 and 3 we tested our setup for specifying a simple character animation sequence in a virtual environment. A performer and an animator collaborate during the generation of the animation. The role of the performer is to act and gesture in front of a Kinect device while receiving immediate feedback of his or her body movements and the animator's actions on the virtual character. We choose a third person's view or so-called "over-shoulder" perspective behind the virtual character for the performer's view. Thus the performer is able to initiate, correct and adapt character movements in real time, and even to perform simple interactions with scene objects as supported by the Unity 3D game engine. The performer can switch the perspective to a head-centered ego-centric view if desired. The tracked user can perform any kind of movements such as walking, jumping or crawling, and convey different styles or behavior if desired. The animator in front of the SmurVEbox receives an overview of the entire scene from a bird's eye perspective. Both can interact with the virtual character and other game objects dynamically through multi-touch gestures like pan, zoom or rotate gestures, and they can, for instance, move and rotate objects within the scene. Animator and performer can explore the scene in a collaborative way and can communicate with each other in order to agree on different decision about the animation process.

[4]http://www.xtuio.com/

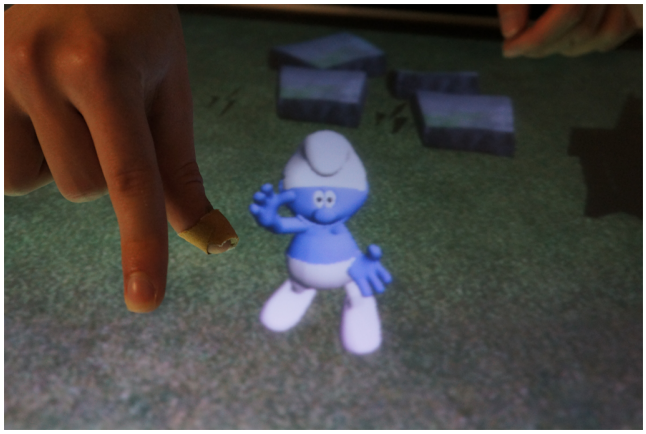


Figure 3: Multi-touch interaction performed by an animator who can specify position, orientation and scale of objects (a smurf in this case) in the VE, while motions of the character can be simultaneously defined by a performer.

4.1.1 Animating a Smurf

Two-legged and -armed characters are commonly known as *bipeds* [7], which can range from faithful representations of human avatars to fictional characters, such as the smurf which we chose to use as a near-human test character. Since the smurf character supports similar forms of locomotion and interaction, e. g., walking upright or the ability to touch objects with both hands, we assumed that animating the smurf would be a straightforward process. We provided the scene with various objects for the smurf to interact with.

4.1.2 Animating a Bird

Non-bipedal characters like quadrupeds or even more distinguished skeleton systems can sometimes be treated analogous to bipedal behavior, e. g. four-legged characters behave and move similar to two synchronized bipeds [28, 29]. In our setup we tested to what degree the skeletal system of a bird can be broken down to this basic approach. We manually fitted a raven-like polygonal model with the same skeleton structure and naming convention as applied for the smurf model, but altered the pose and scale of the joints according to the form of the bird model (see Figure 2).

4.2 Feedback

We informally collected feedback from three animators and performers, who have tested our system for about 1 hour. We used the think-aloud protocol as well as interviews. The most striking feedback of using our setup was that it was "really fun and engaging to use the setup". When synchronizing the action in the scene both the animator and the performer were communicating extensively. With that tasks could be split up between animator and performer; both were experiencing the scene individually and collaborated effectively. Artistic freedom of the animator was maintained since there were no constraints regarding restriction of movements, e. g., through the fitted Kinect rig. We found that only a small space was required and actually used for acting and interacting with the scene. Of course, the Kinect provides only very limited motion tracking ca-

pabilities. However, tracking of limbs was stable even in situations when performers were pointing towards the tracking sensor. Overall, the performers acknowledged that they were surprised about the range of freedom they had for acting. The real-time capacity of the whole system turned out satisfactory so that live action could be adapted and corrected where necessary. Setting up an additional projector, which displayed the performer's view to the animator, was evaluated as beneficial for both, in particular, since this projection was often used even by the animator. For future work we plan to incorporate this view into the view on the SmurVEbox such that the animator can see the performer's view without averting the gaze from the SmurVEbox. As expected, the visual feedback of the animated character on the SmurVEbox was evaluated as very responsive and helpful by the animator. The choice of the performer's scene camera angle from a third-person's view turned out to be more helpful compared to a first-person's view after informal tests, since according to a performer, it was much easier to evaluate the action than from an ego-centric view.

5. CONCLUSION

In this paper we introduced the SmurVEbox collaborative virtual environment for generating computer animations in real time. We described the hardware setup as well as the software implementation of our shared interaction space, and presented a use case that benefits from the ability of a performer and an operator to collaborate in the same environment when generating computer animations. Initial feedback collected from a few animators and performers was throughout positive. In particular, users acknowledged the immediate feedback and the collaborative work supported by the framework. With the current setup, animators and performers can design and sketch rough animation prototypes for further development using professional 3D Desktop-based animation software. In the future, we aim to improve animation export capabilities of our system. Moreover, we plan to reduce occlusion issues and improve the motion tracking in our setup using a setup consisting of multiple Kinects (cf. [3]).

6. REFERENCES

[1] W. Baxton and B. Myers. A study in two-handed input. In *Proceedings of ACM CHI*, pages 321–326, 1986.

[2] N. Burtnyk and M. Wein. Interactive skeleton techniques for enhancing motion dynamics in key frame animations. *Communications of the ACM*, 19:564–569, 1976.

[3] D. A. Butler, S. Izadi, O. Hilliges, D. Molyneaux, S. Hodges, and D. Kim. Shake'n'sense: reducing interference for overlapping structured light depth cameras. In *Proceedings of ACM CHI*, pages 1933–1936, 2012.

[4] J. Denavit and R. Hartenberg. A kinematic notation for lower-pair mechanisms based on matrices. *Transactions of the ASME Journal of Applied Mechanics*, 22:215–221, 1955.

[5] J. Edelmann and S. Fleck. The DabR - a multitouch system for intuitive 3D scene navigation. In *Proceedings of the 3DTV Conference*, pages 1–4, 2009.

[6] M. Fischbach, M. E. Latoschik, G. Bruder, and F. Steinicke. smARTbox: Out-of-the-box technologies for interactive art and exhibition. In *Proceedings of the Virtual Reality International Conference (VRIC)*, pages 1–7, 2012.

[7] M. Girard and A. A. Maciejewski. Computational modeling for the computer animation of legged figures. In *Proceedings of ACM SIGGRAPH*, pages 263–270, 1985.

[8] K. Hinckley, J. Tullio, R. Pausch, D. Proffitt, and N. Kassell. Usability analysis of 3D rotation techniques. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*, pages 1–10, 1997.

[9] J. K. Hodgins, W. L. Wooten, D. C. Brogan, and J. F. O'Brien. Animating human athletics. In *Proceedings of ACM SIGGRAPH*, pages 71–78, 1995.

[10] T. Igarashi, T. Moscovich, and J. Hughes. As-rigid-as-possible shape manipulation. In *SIGGRAPH'05 Proceedings of the 32nd annual conference on Computer graphics and interactive techniques*, pages 1134–1141, 2005.

[11] T. Igarashi, T. Moscovich, and J. Hughes. Spatial keyframing for performance-driven animation. In *SIGGRAPH'06 Proceedings of the 33rd annual conference on Computer graphics and interactive techniques*, page Article No.17, 2006.

[12] L. Kavan and O. Sorkine. Elasticity-inspired deformers for character articulation. *ACM Transactions on Graphics (TOG)*, 31:Article No.196, 2012.

[13] K. Kunze, M. Barry, E. A. Heinz, P. Lukowicz, D. Majoe, and J. Gutknecht. Towards recognizing Tai Chi - an initial experiment using wearable sensors. In *Proceedings of IFAWC*, pages 1–6, 2006.

[14] C. Müller-Tomfelde and J. Schöning. Building interactive multi-touch surfaces. In C. Müller-Tomfelde, editor, *Tabletops - Horizontal Interactive Displays*, Human-Computer Interaction Series, pages 27–49. Springer, 2010.

[15] R. Parent. *Computer Animation Algorithms and Techniques*. Morgan Kaufmann, 2008.

[16] F. Parke. Computer generated animation of faces. In *Proceedings of the ACM National Conference*, pages 451–457, 1972.

[17] M. H. Raibert and J. K. Hodgins. Animation of dynamic legged locomotion. In *Proceedings of ACM SIGGRAPH*, pages 349–358, 1991.

[18] H. Sato and M. Cohen. Using motion capture for real-time augmented reality scenes. In *Proceedings of the International Conference on Humans and Computers (HC)*, pages 58–62, 2010.

[19] A. Shapiro. Building a character animation system. In *Proceedings of the International Conference on Motion in Games (MIG)*, pages 98–109, 2011.

[20] T. Shiratori, H. S. Park, L. Sigal, Y. Sheikh, and J. K. Hodgins. Motion capture from body-mounted cameras. In *Proceedings of ACM SIGGRAPH*, pages 1–10, 2011.

[21] F. Steinicke, G. Bruder, and S. Kuhl. Realistic perspective projections for virtual objects and environments. *ACM Transactions on Graphics (TOG)*, 30:1–10, 2011.

[22] F. Steinicke, T. Ropinski, G. Bruder, and K. Hinrichs. Interscopic user interface concepts for fish tank virtual reality systems. In *Proceedings of IEEE Virtual Reality (VR)*, pages 27–34, 2007.

[23] R. M. Taylor II, T. C. Hudson, A. Seeger, H. Weber, J. Juliano, and A. T. Helser. VRPN: A device-independent, network-transparent VR peripheral system. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST)*, pages 55–61, 2001.

[24] R. J. Teather and W. Stuerzlinger. Guidelines for 3D positioning techniques. In *Proceedings of the conference on Future Play*, pages 61–68, 2007.

[25] L. Terziman, M. Marchal, M. Emily, F. Multon, B. Arnaldi, and A. Lecuyer. Shake-your-head: revisiting walking-in-place for desktop virtual reality. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST)*, pages 27–34, 2010.

[26] M. Thorne, D. Burke, and M. van de Panne. Motion doodles: an interface for sketching character motion. In *SIGGRAPH'04 Proceedings of the 31st annual conference on Computer graphics and interactive techniques*, pages 424–431, 2004.

[27] D. Valkov, F. Steinicke, G. Bruder, K. Hinrichs, J. Schöening, F. Daiber, and A. Krüger. Touching floating objects in projection-based virtual reality environments. In *Proceedings of the Joint Virtual Reality Conference (JVRC)*, pages 17–24, 2010.

[28] A. Vögele, M. Hermann, B. Krüger, and R. Klein. Interactive steering of mesh animations. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, pages 53–58, 2012.

[29] R. Williams. *The Animator's Survival Kit*. Faber and Faber, 2001.

[30] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Computing Surveys (CSUR)*, 38:1–45, 2006.