# Expressing Animated Performances through Puppeteering

Takaaki Shiratori* Microsoft Research Asia
Moshe Mahler† Disney Research, Pittsburgh
Warren Trezevant‡ Autodesk, Inc.
Jessica K. Hodgins† Carnegie Mellon University, Disney Research, Pittsburgh

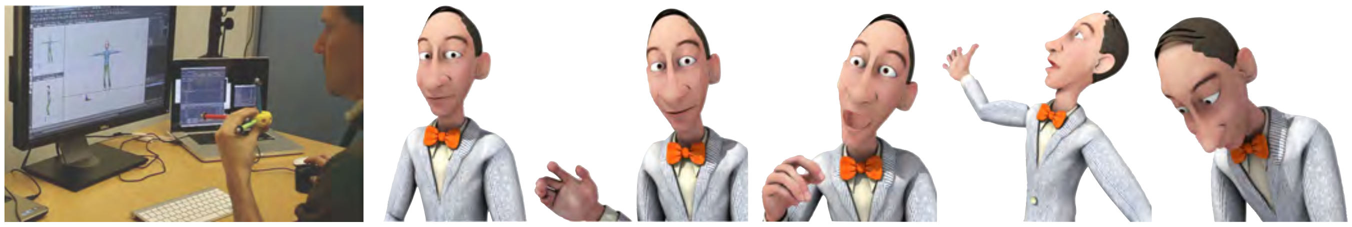


Figure 1: Animator puppeteering a character for a dialog, “Minimum Wage,” and frames from the resulting animation.

## Abstract

An essential form of communication between the director and the animators early in the animation pipeline is rough cut at the motion (a blocked-in animation). This version of the character’s performance allows the director and animators to discuss how the character will play his/her role in each scene. However, blocked-in animation is also quite time consuming to construct, with short scenes requiring many hours of preparation between presentations. In this paper, we present a puppeteering interface for creating blocked-in motion for characters and various simulation effects more quickly than is possible in a keyframing interface. The animator manipulates one of a set of tracked objects in a motion capture system to control a few degrees of freedom of the character on each take. We explore the design space for the 3D puppeteering interface with a set of seven professional animators using a “think-aloud” protocol. We present a number of animations that they created and compare the time required to create similar animations in our 3D user interface and a commercial keyframing interface.

**Index Terms:** I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction techniques; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

## 1 Introduction

The heart of a computer animation is the performance of the characters as they play their part in the narrative arc. Through their motions, they tell the story, express their emotions and motivation, and convey their personality. This performance is designed in a process called *blocking-in*: working from audio and the storyboards, the animator creates a rough version of the motion.

Animators often spend one or more days to create a 10-second blocked-in animation with keyframing in a mouse-based animation system. Particularly in commercial productions, the blocked-in animation serves as a form of visual communication between the animator and the director, and the animator refines and redoes the blocked-in motion until the director is satisfied that it conveys the essence of the performance. With several days required between iterations, this process can be slow to converge. On the other hand, for live action movies, a director can guide the performance of actors, and the actors can rehearse their performance in real time. A user interface that would allow for such a quick revision cycle for animation productions would not only reduce the cost of the productions but might also result in a higher quality product by allowing more iterations of refinement.

In this paper, we describe a 3D puppeteering interface that allows an animator to “perform” the motion of a character or a simulation effect for the blocking phase rather than “keyframing” it in one of the commercially available animation software packages (Figure 1). We hypothesize that the animator can perform motion much more quickly through puppeteering than he/she can animate in traditional keyframing software. Performing the motion allows the animator to quickly get a feeling for the timing of the performance while maintaining approximate control of the poses. A number of animators have observed that the timing of actions is the key element in conveying the performance of the character [10, 19].

In addition to keyframing, alternative approaches used to convey the proposed performance include motion capture, selecting motion from a database, or videotaping the animator’s performance. Our preliminary interviews with animators indicated that puppeteering showed significant promise for several reasons. It allows motions that are not physically realistic and non-human—a requirement for many character animations as well as for secondary animations such as fluid effects. Puppeteering provides the animator with full control over the design and timing of the motion whereas selecting from even a large database is necessarily more restrictive. Creating the blocked-in motion for the actual character geometry and rig brings it far closer to the final animation than a video of a human performance or motion capture.

We chose to use an iterative refinement process with a set of animators to better understand what they expected and needed from a puppeteering interface for blocked-in motion. Iterative refinement is the seminal idea of iterating between an interface refinement phase and a user test phase and has been shown to result in significantly better systems in many domains. Seven professionally trained animators and two experienced animators created a variety of animations with our system. We explored a number of key dimensions for the interface design with seven animators:

- Should the actions be motion-captured (a direct mapping from the actor to the human-like degrees of freedom (DOFs) on the character) or puppeteered (control of DOFs without a direct anatomical mapping)?

- What tools allow the most effective performance: a single manipulated rigid object, markers on the hands or gloves, or cus-

*takaakis@microsoft.com
†{mo, jkh}@disneyresearch.com
‡warren.trezevant@autodesk.com

tom designed objects?

- How should the synchronization of motions across multiple takes (layering) or between joints be accomplished?

The iterative refinement process provided us with the answer to these interface design questions. It also generated a list of necessary user interface features such as control over the editable range, controllable delay, enabling/disabling particular DOFs, multiple live views, and saving and restoring settings.

Our test animators demonstrated that puppeteering could provide significant time savings by allowing them to rapidly explore the elements of a performance. With the puppeteering interface, an animator is now able to produce a credible performance of a 15 second scene in less than an hour, rather than over a period of several hours or longer. A number of the animations were created twice, once in the puppeteering interface and once in a traditional keyframing interface. In all cases, the puppeteering interface was more efficient and the animators felt that the animations were of equal effectiveness in conveying the important elements of their intended performance.

Through the iterative refinement process, we learned about a number of factors that are important to animators in the design of an interface for blocking in motion: (1) the motion should be puppeteered, not motion captured. The physical constraints of motion capture are not a good match to the exaggerated motion required of most animated characters. (2) Blocking in motion is truly a performance and as such, the speed with which the animator can record the timing and poses of the performance is critical for capturing the performance that is in his/her head. Many of the design decisions of the user interface were driven by the need for speed of capture as the animator moved through the DOFs of the character. (3) Synergies or correlations between joints are essential for natural human motion and are similarly important for animated characters. Our animators wanted to tie joints together with offsets in angle and timing so that they could more quickly capture a full performance.

## 2 Related Work

The field of animation interfaces is diverse, with approaches ranging from sketching (*e.g.*, [5]) to full body, real time motion capture (*e.g.*, [17]). In this section, we discuss related work in 3D puppeteering interfaces. Most of these interfaces have been directed at naive users rather than professional animators, giving the work a different flavor than that presented here. The previous work focused on providing appropriate scaffolding to reduce the complexity of the system (including correlation maps, implicit editing, and physics). These constraints are unacceptable to professionals as they require full control over the character to express the nuances of the performance.

Oore and colleagues [16] developed a layered editing system with two six-DOF motion trackers. A character skeleton was subdivided into layers (legs, arms, spine and head), and the users could animate the layers through multiple takes. Dontcheva and colleagues [2] also developed a layered acting system. Their key contribution was the derivation of an implicit relationship between the animator's and the character's motions. That technique is very valuable for novice users who require a walk-up-and-use interface but do not need precise control over the character's motion. But it is inappropriate for professional animators who are willing to invest more time in specifying the motion in exchange for precise control. Interestingly, we both found that simple tracked objects were the most effective, despite these differences in application.

Neff and colleagues [14] introduce a correlation map in which users specify a correlation between motion of multiple degrees of freedom and a single 2D input from a mouse with linear interpolation. The paper also contributed relative mapping and overlays. While very effective for novices, these features were not found to be useful by our professional animator testing team. Laszlo and colleagues [11] developed a system to control a physically simulated character with a mouse. Their system actuates joints with torques according to 2D input. The goal of their system is interactive control of a physically simulated character for entertainment, and therefore is different from ours.

There have been a number of interfaces that used special purpose input devices for the creation of poses. Esposito and colleagues [3] created an instrumented puppet called the *Monkey* that allowed the users to specify poses by positioning the joints of the device. Knep and colleagues [7] used an armature to perform stop motion directly into the computer for rendering. Feng and colleagues [4] used a puppet to specify key poses and retrieved a matching motion from a database. They used a stereo camera to identify the positions of the puppet joints. Yoshizaki and colleagues [22] used actuated joints for their puppet interface to improve the usability through gravity compensation and biomechanical constraints. These interfaces could also be used for blocked-in motion, although the poses would have to be positioned along a timeline to create the final animation. Such input devices can also be used to specify motion of a character rather than poses. Jim Henson Productions and Pacific Data Images created a puppeteering device to animate a character called Waldo C. Graphic in real time for a TV show [20]. Motion of each DOF of the device was directly mapped to the corresponding DOF of the character. The drawback of these approaches is that the system is applicable only to characters that have the same DOFs as input devices.

Real-time motion capture has been a focus of research for many years with one of the key issues being how to map to a character with non-human dimensions. Shin and colleagues [17] addressed this problem by identifying the key features of the motion that should be preserved by hand and then maintaining those in each section of motion. Our approach is different in that the motion does not have to be performed in one take and the animator has the freedom to re-define that mapping in between takes.

Other systems have been developed that leverage a domain-specific database to map from low-dimensional input to complete motion. (see Chai and Hodgins's work [1] for an example using motion capture). Johnson and colleagues [6] developed an instrumented puppet to control a bird-like character via selections from a set of pre-determined motion patterns. Komura and colleagues [8] used an instrumented glove to select among and modify a number of pre-programmed gaits. Luo and colleagues [12] similarly used a sensing glove to control a simple digital puppet directly. Input finger motion was recognized with a database consisting of sensor readings of six finger motions. Numaguchi and colleagues [15] used motion of an instrumented puppet to retrieve motion capture data from a database. Input puppet motion is matched to human motion in a database based on the subspaces of the puppet and human motion. Although these approaches were intuitive and efficient for naive users, the domain-specific databases limit types of animations to be designed.

A few of the mouse or pen-based interfaces have emphasized the capture of the timing of the motion in real time rather than just pose creation. Yamane and Nakamura [21] developed a pin-and-drag mouse interface. The users could pin any body part and move the unconstrained body part freely. This interface enabled users to easily specify poses and timings to create animations without any reference motion. Igarashi and colleagues [5] developed a spatial keyframing system. Their insight was to define key poses and associate them with locations in the workspace of the mouse so that the mouse position could be used to fully control a relatively complex character at desired timings. As with our system, the key contribution of these work is defining the mapping between a low dimensional input and the complex motion of the character.

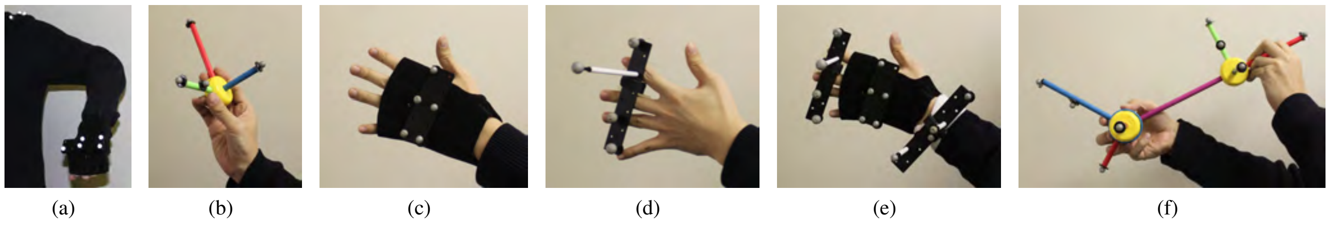


Figure 2: The various tracked objects and marker sets that we tested. (a) A motion capture setting including markers on the shoulder and hand that our animators thought was inefficient, (b) a TinkerToy$^{TM}$construction with marker settings that work for most joints/rig, (c, d) sweat band-based constructions with settings that work for mapping translational motion such as IK nodes, and (e, f) modifications of earlier structures with special settings for eyebrows, and arms.

## 3 Method: Iterative Design

Via a process of iterative refinement, we explore a number of different possible parameters for the puppeteering interface. We had a primary animator and six secondary animators who participated in our iterative refinement of the 3D user interface. All seven were professionally trained animators. The primary animator has experience creating keyframe animations for a number of feature-length films. He used the interface for a total of about 20 hours over several design revisions. The secondary animators used the later versions of the interface for 3–10 hours each. Collectively, they had prior experience in the creation of full length features, artistic shorts and video games via motion capture and keyframing. With each animator, we used a "think aloud" methodology. The animators talked about their thought about processes and actions while they constructed a set of blocked-in motions. We provided them with rigged characters, the audio clips for a set of short speeches, as well as suggestions for whole body motions (*e.g.*, jump, walk, dance) for them to interpret.

We now discuss the design questions that were analyzed during the iterative refinement process: input devices, puppeteered DOFs and interface features.

### 3.1 Input Devices

Our initial design included not only devices that could be held and rotated or positioned but also strategies for instrumenting the hand/wrist and arm/shoulder complexes of the animator. This second class of techniques allowed direct manipulation of the corresponding body parts of the character and was much closer to motion capture than to puppeteering (Figure 2(a)). We had assumed that these marker sets would provide an intuitive way to manipulate the arms of the character given the high number of DOFs involved and the power of inverse kinematics (IK). Our primary animator found the motion capture setups too cumbersome to use, in part because it was more time consuming to switch between animating the various DOFs of the character with these marker sets. He felt that it was essential to be able to cycle through the animatable DOFs of the character quickly while the performance and timing remained "in his head." Therefore he almost always reached for small, easy-to-grasp objects to puppeteer his animations. The other animators were offered the same choices and also chose to primarily use the simple input device.

The cost of choosing puppeteering over motion capture, however, is that, just as with keyframing, we lose some of the innate instincts that the animator possesses about how his body moves and instead have to rely on a developed talent for puppeteering. Some of our test animators were better able to transfer their animation skills to this novel (to them) art form than others.

Although we experimented with a variety of tracked objects, the simple Tinkertoy$^{TM}$construction proved to generally be the best, in part because it was easy to grasp and intuitive to orient (Figure 2(b)). The other tracked objects that worked well used a rigid marker set mounted on a sweatband and structure (Figures 2 (c-e)), as those were also easy to pick up and put down. One animator also experimented with a pivot joint for the arm mapping (Figure 2(f)).

### 3.2 Puppeteered DOFs

Intertwined with the design decisions about the controllable DOFs is the rigging of the character (the mapping of degrees of freedom of the skeleton to controllable variables). The interface allowed the animator to create arbitrary mappings from tracked objects to DOFs in the character. However, our animators quickly converged on a relatively small set of mappings that they found most effective. These could be divided into three categories: forward kinematics (FK), IK, and blendshapes (two deformations for the character's face which can be interpolated to create an animation). As with a keyframing interface, the rigs could be toggled on and off, so the arms, for example, could easily be animated via either IK or FK. The animators generally chose between FK and IK in the same way as a keyframe animator would, using FK for free space motions and IK for constrained motions (feet in contact with the floor, for example).

The controllable DOFs that were used to generate the results shown in the video are listed below (Figure 3).

- FK for pelvis, spine, neck, shoulder, elbow, wrist, finger, hip, knee, and ankle joints of all characters and wings and tail for flying characters (*e.g.*, a dragon).
- IK for hands and feet, and flap for the positions of elbows.
- Blendshapes for jaw and eyebrows.
- Position and orientation of a camera for rendering.
- Position of particles for fluid animation.

### 3.3 Overall Workflow

As he worked with the interface, our primary animator settled on a workflow that was largely followed by our other animators:

1. Setup: Confirm that parameters for the puppeteering interface are appropriate for the desired motion.
2. Rehearsal: Experiment with various performances (without recording).
3. Recording: Hit *record* button to capture. Recording starts with a three second pre-roll for synchronization. Preserve the capture if it was satisfactory. Otherwise repeat recording step.
4. Layering: move to the next DOF that should be animated (generally moving from root outward to limbs and head).
5. Post-processing: Do any required post-processing in a keyframing interface after the puppeteered animation is complete.

Details of the interface features used in these steps are described in the next section.

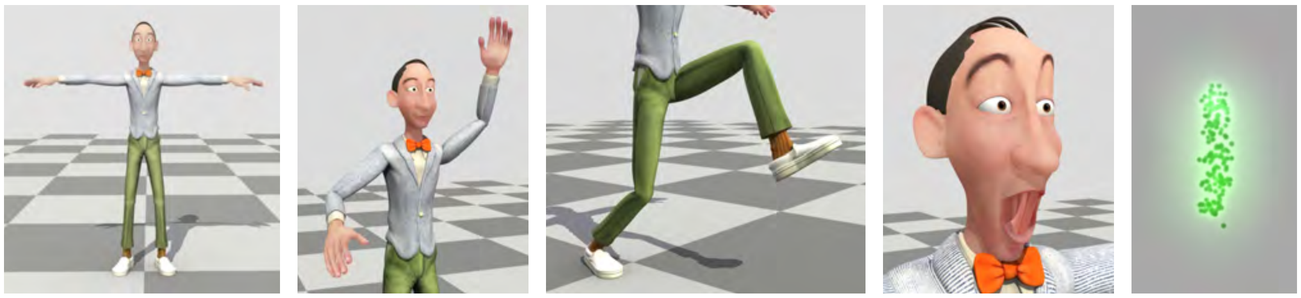


Figure 3: Examples of puppeteered DOFs. (a) Neutral pose of our human character, (b) arms, (c) legs, (d) eyebrows and jaw, and (e) particles.

## 3.4 System Features

Based on the animators' think-aloud comments, we added numerous features to the interface that allowed them full control over the characters.

**Flexible Mapping:** In the initial phases of the design, we assumed that the number of DOFs that could be controlled simultaneously would be limited by a maximum of six DOFs/hand. In contrast, the skeleton of a fully rigged character will have 50 or more DOFs. However, in our initial testing with our primary animator, it quickly became obvious that the number of controllable DOFs would be far more limited than we had expected. Our experiments indicated that using a tracked object to control the six DOFs of both position and orientation did not produce natural looking motion. The animator most often chose only to use one hand at a time and the number of DOFs that he could control effectively was generally limited to two position or three orientation DOFs. Therefore, the final design of the interface enables the animators to quickly choose which DOF(s) they want to control and to layer motions together from multiple takes.

**Synergies:** Our primary animator also requested the ability to map a single tracked object to several different DOFs on the character simultaneously. Although not part of our original interface design, this feature for enforcing synergies proved to be very effective. For example, the animator might map the orientation of the tracked object to both the left and right shoulder orientations but with an offset so that one shoulder would move further than the other. This mapping allowed the animator to create a nearly but not perfectly symmetric motion for the shoulders during a jump, for example. Our secondary animators also used this feature extensively.

**Controllable Delay:** One of the secondary animators working at a commercial video game studio pointed out that it would be helpful to have the ability to design dynamic effects such as animals' tails and passively moving objects such as trees blowing in the wind. We implemented this functionality by imposing a time delay on the motion. The animator first assigned several joints to a single tracked object. The tracked object motion was mapped to the parent node without delay, and was mapped to the child nodes with delay (Figure 4). This functionality is motivated by a technique used in keyframing where the parent node motion is created and then copied to the child nodes with latency.

**Layering:** Because the number of DOFs that could be puppeteered in one take was so limited, we quickly realized that it was crucial to facilitate layering of multiple takes each of which animates a small number of DOFs. For example, our animators first specified the path of the pelvis as a position on the ground plane (two DOFs) and in a second layering pass, added the up/down trajectory for the pelvis motion (one DOF). Similarly, they found it very effective to control the pelvis, spine, and head/neck rotations via a sequence of takes with one or two DOFs animated in each. We included a 3-2-1 countdown to allow the animator time to position the character for the start of the motion. Audio was played to facilitate synchronization for those sequences that included it. Our approach to this problem is similar to previous layering techniques [16, 2].

**Editing:** Comments from our primary animator led us to realize that it was quite frustrating to have to repeat a complete performance when the unsatisfactory motion had occurred only in a small window of time. We added the ability to re-perform a portion of the sequence with blends provided between the start and end points of the old motion and the new. This functionality was not perfect in that it was difficult to hit a pose at the beginning and the end of the edited sequence. We provided a grayed out figure in the Maya interface to show the start and end poses from the original performance, which made it much easier to create a match that could be seamlessly blended in post-processing.

**Sensitivity and Home Position:** The sensitivities and home position (*i.e.*, offset for DOFs) of the mapping between the animator's motion and that of the character proved to be crucial for agile control of the character. Our first prototype used a minimum and maximum position to compute sensitivities and home position for calibration. Based on feedback from our primary animator, we quickly switched to calibration by manually specifying a home position and sensitivities for the movement.

Our animators also used different sensitivities for the different DOFs of a single joint/rig. For example, the spine was usually animated with less sensitivity side-to-side than front-to-back, corresponding approximately to the relative magnitude of the range of motions of the two DOFs. Some animators also set a sensitivity for time to create fast motion such as flapping the wings of birds.

For translation, our animators liked a view-dependent mapping that mapped a tracked object's motion to target DOFs with respect to the current view. Given the user-specified home position ${}^{\mathcal{M}}\mathbf{p}_0$ and the current position ${}^{\mathcal{M}}\mathbf{p}$ of the tracked object in a motion capture coordinate system $\mathcal{M}$, the displacement in an animation coordinate system $\mathcal{A}$ is represented as

$$\Delta^{\mathcal{A}}\mathbf{p}(t) = \mathbf{C}^{\mathsf{T}}\mathbf{S}\left({}^{\mathcal{M}}\mathbf{p}(t) - {}^{\mathcal{M}}\mathbf{p}_0\right), \tag{1}$$

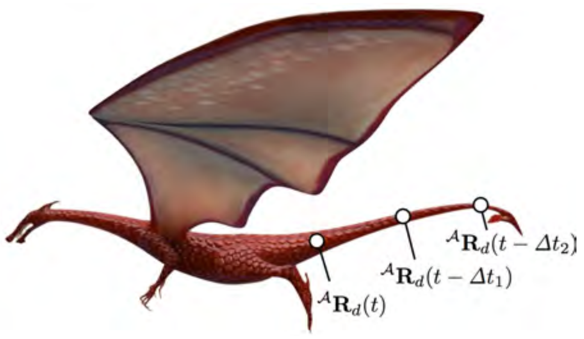
where $\mathbf{C}$ represents the orientation matrix of the viewing camera in the animation coordinate system, and $\mathbf{S}$ is a diagonal matrix consisting of sensitivities ($\mathbf{S} = \mathrm{diag}(s_x, s_y, s_z)$). The mapped position of the target DOFs in $\mathcal{A}$ is computed as

$${}^{\mathcal{A}}\mathbf{p}_d\left(s_t t\right) = {}^{\mathcal{A}}\mathbf{p}_d(0) + \Delta^{\mathcal{A}}\mathbf{p}(t), \tag{2}$$

where $s_t$ is a sensitivity for time.

For rotation, our animators preferred a view-independent mapping. Therefore, the rotation for the puppeteered joint/rig, $\mathbf{R}_d$, is computed as

$$\mathbf{R}_d\left(s_t t\right) = S\left({}^{\mathcal{M}}\mathbf{R}(t)\,{}^{\mathcal{M}}\mathbf{R}_0^{\mathsf{T}}\right)\mathbf{R}_d(0), \tag{3}$$



$^{\mathcal{A}}\mathbf{R}_d(t-\Delta t_2)$
$^{\mathcal{A}}\mathbf{R}_d(t-\Delta t_1)$
$^{\mathcal{A}}\mathbf{R}_d(t)$

Figure 4: Example usage of the controllable delay feature. $\Delta t_1$ and $\Delta t_2$ are manually specified latencies.

where $^{\mathcal{M}}\mathbf{R}_0$ and $^{\mathcal{M}}\mathbf{R}$ represent the user-specified home orientation and the current orientation of the tracked object, respectively, and $S(\cdot)$ is a function that converts an input rotation matrix in the motion capture coordinate system into Euler angles, multiplies the angles by sensitivities, and returns a scaled rotation matrix in the animation coordinate system. Which approach the animators chose was determined by the behavior being animated. For example, local orientation is useful for specifying joint angles and global orientation is used for root orientation. Therefore, the interface allowed the animators to choose global or local mapping, and the global orientation of the target DOFs in the animation coordinate system, $^{\mathcal{A}}\mathbf{R}_d$, is computed as

$$^{\mathcal{A}}\mathbf{R}_d(t) = \begin{cases} \mathbf{R}_d(t) & \text{if global mapping} \\ \mathbf{R}_d(t)^{\mathcal{A}}\mathbf{R}_p(t) & \text{if local mapping} \end{cases} \quad (4)$$

where $^{\mathcal{A}}\mathbf{R}_p$ represents the global orientation of the parent DOFs in the animation coordinate system.

One advantage to this approach was that the animator could now ground the character in the workspace. For example, our primary animator often placed the tracked object on the table and mapped that position to a standing posture with the feet on the ground. Then he could control the height of the pelvis knowing that whenever the tracked object was on the ground, the feet would be as well (assuming no animation on the legs at this stage). Our secondary animators also adopted this trick.

Our primary animator also found that he wanted different sensitivities for different scenes. Subtle animations such as eyebrows required more detailed motion and therefore decreased sensitivities, while bigger animations such as walking required large motions and therefore increased sensitivities. Sensitivities also depended on camera shots. Close shots required smaller motions and therefore decreased sensitivities. We added functionality for saving/loading home positions and sensitivities on a per-joint or a per-scene basis to facilitate reuse of parameters.

This observation is an interesting consequence of the fixed size of the animator's workspace and the kinematics and physiology of the human arm [13]. Accuracy improves and tremor is reduced for the center of the workspace (Figure 5 top and middle). Moving away from the movement fovea in the workspace leads to increased variability or inaccurate control as the kinematics of the human arm do not allow arbitrary paths near the edges of the human workspace (Figure 5 bottom). Several of our animators chose home positions that allowed them to brace their forearm or elbow on the table.

**Particle Animation:** Particle animation is a useful visual effect, but requires many parameters such as the location and rate of emitters. A secondary animator working at an animation studio suggested that these parameters could be puppeteered and added this functionality by leveraging Maya's simulation engines to compute the resulting motion.

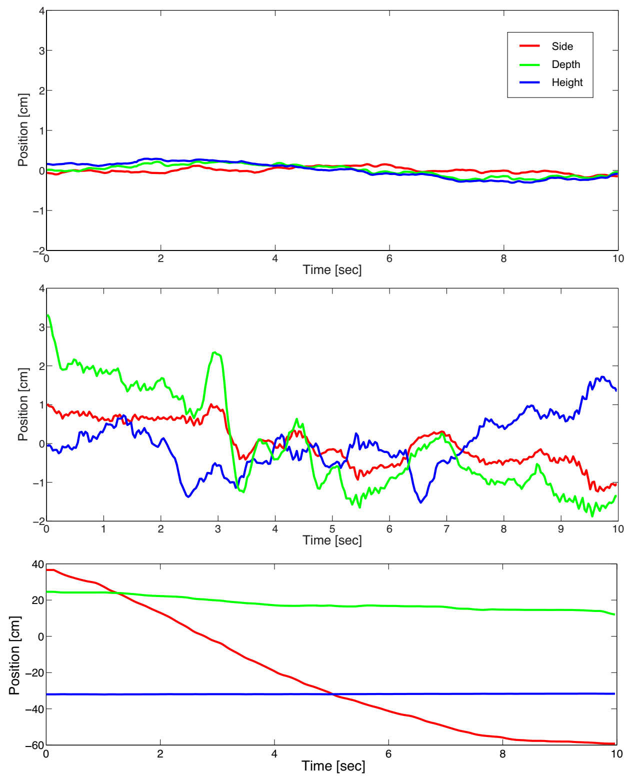


Figure 5: Top: a motion in which the animator attempted to hold a pose around the center of the workspace, and middle: near the edge. The resting tremor is obvious in the data and in the resulting motion of the character. Bottom: a motion in which the animator attempted to create a straight line across the workspace. The curves at the end occur because of the kinematics of the human arm.

Motion of a particle can be created based on Equations (2) and (4). Animators may use displacement mapping to create motion of child particles that mimic the parent particle trajectory. Given the position, $^{\mathcal{A}}\mathbf{p}_p$, and the orientation, $^{\mathcal{A}}\mathbf{R}_p$, of the parent particle trajectory, the child particle position, $^{\mathcal{A}}\mathbf{p}_c$, is specified with the displacement computed with Equation (1) as

$$^{\mathcal{A}}\mathbf{p}_c(s_t t) = {}^{\mathcal{A}}\mathbf{p}_p(s_t t) + {}^{\mathcal{A}}\mathbf{R}_p^{\mathsf{T}}(s_t t)\Delta^{\mathcal{A}}\mathbf{p}_p(t). \quad (5)$$

Animators can also specify the number of particles to be puppeteered and randomize the aforementioned parameters with manually specified range if necessary. These randomization settings allowed for a wide variety of particle motion and enabled the animators to quickly test behaviors of fluid animation without physical simulation.

**Post-processing:** The animator has the editing functionality of a standard animation system available for post-processing of the captured performance. For example, motions that should be tightly synchronized can be aligned and the scale of motions can be adjusted in time or space to create a stronger performance. We found that the standard curve editing tools in Maya were effective for these operations. Our animators also used the spline fitting functionality to simplify the puppeteered curves prior to editing (as with motion capture, our curves have one key per frame by default).

One editing operation proved particularly important and that was to damp out the residual motion for a held pose. This animation element was named a *moving hold* by Thomas and Johnston [19] and it is often used to draw attention to the main action of storytelling. The ability to hold a pose is of course trivial in a keyframing interface, but in a puppeteering interface residual motion of the hands

Table 1: Comparison of the time required to create an animation with the puppeteering interface (P) with Maya keyframing (K).

| Animation (animator ID) | P timing (# of frames) | K timing (# of frames) | Order |
|---|---|---|---|
| Flying penguin (Animator 1) | 30 min (300) | 45 min (300) | K → P |
| Stretching penguin (Animator 1) | 40 min (300) | 50 min (300) | P → K |
| Drinking man (Animator 1) | 40 min (540) | 80 min (540) | P → K |
| Flying dragon (Animator 2) | 20 min (112) | 20 min (40) | P → K |

becomes jitter in the motion of the character. Because we were generally mapping up from smaller/lighter DOFs on the animator's body to larger/more massive DOFs for the character (animator's hands to character's pelvis, for example), the tremor in the animator's hands was magnified in the character's motion and made the motion appear unnatural.

## 4 Experiments

We used an OptiTrack system produced by Natural Point with eight cameras for the puppeteering interface. The cameras were mounted on four tripods positioned to the left/right and front/back of the workspace. Four cameras were positioned high on the tripods (approximately 2.5 m) and four were positioned lower (approximately 1.5 m). The puppeteering interface was implemented as a Maya plug-in, leveraging the basic viewing manipulation functionality of Maya and was connected to the motion capture system via VRPN [18]. The post-processing was done with Maya 2011.

### 4.1 User Performance

We conducted an informal user study with two additional animators to compare the puppeteering interface with a conventional keyframing interface. Animator 1 was an undergraduate animation student, and had experience with Maya keyframing. Animator 2 was a professional artist working on a video game production, and had experience with Maya keyframing and motion capture. We provided them with three character models: a human model (56 DOFs), a penguin model (48 DOFs), and a dragon model (69 DOFs).

We first demonstrated each interface feature to the animators, and then they spent approximately an hour getting accustomed to the interface. After this practice phase, we asked them to produce an animation of their design. We also asked them to create the same animation by keyframing in Maya. After the sessions, we asked them to comment on the quality of the animations and the two interfaces.

Table 1 compares the time required for the puppeteering and keyframing interfaces. Animator 1 created three animations. The puppeteered animations were visually similar to the keyframed animations, but required less time to create. Animator 2 created a flying dragon animation. Though the two animations required the same amount of time, the resulting puppeteered motion was several times longer. He commented that he preferred the puppeteered animation because the natural variability of human motion added a lifelike quality to the repetitive flying motion.

### 4.2 Case Study

Figures 1 and 6 show six sample results from the puppeteering interface. The animation in Figure 1 was created by a professional animator working at a commercial animation studio (one of the secondary animators) The first and third animations in Figure 6 were created by an independent professional animator who has experience with Maya keyframing and motion capture (also one of the secondary animators in our iterative refinement design process).

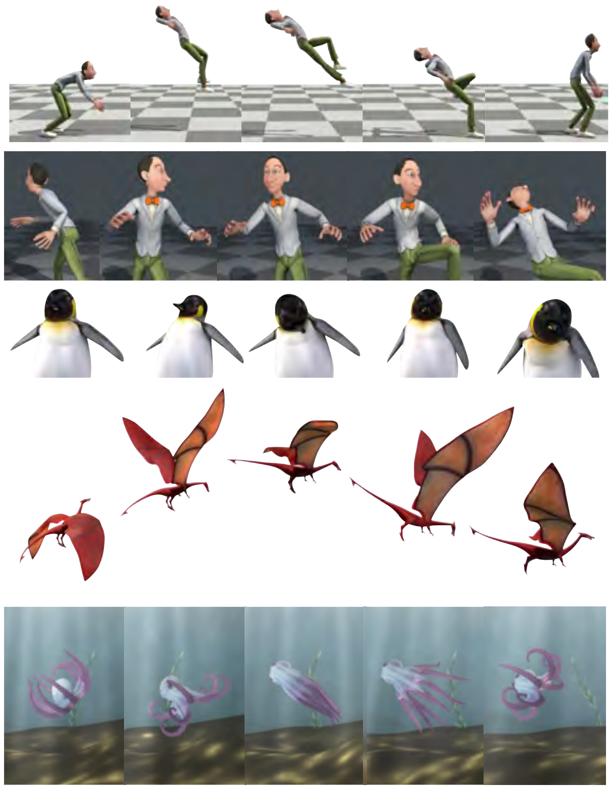


Figure 6: A sequence of images illustrating five of the puppeteered motions.

The second and fourth animations were created by professional animators working at another commercial animation studio. The last animation was created by Animator 1. Table 2 summarizes time and operations the animators used.

These examples illustrate that the interface performs well for creating a number of different kinds of motion: upper body performances in sync with pre-recorded lines of dialog, full body motions such as jumps and walks, and motions of characters with kinematics significantly different from that of an adult human. In creating these examples, our animators puppeteered many different DOFs: mouth, eyebrows, hands, neck, back, shoulder, elbow, wrist, hips, knees, and pelvis. The root, spine and neck of the scared man, the wings and tail of the flying dragon, and the seaweeds and the legs of the octopus animation were animated with controllable delay. The ink animation of the octopus was created with the particle control function. The animators used many layers to create these animations, performing ten or more takes as they moved through the DOFs, usually working outward from the core motion of the root (the pelvis).

The motions were first puppeteered and then refined or enhanced in Maya. The refinements were simple and quick to perform. For the scared man animation, the root, spine and head motions were puppeteered to explore different poses and motions and then the arm and leg motions were added with keyframing in Maya. For motions with held poses, any residual noise caused by hand tremor was removed with a filter. Occasional glitches from the motion capture Euler angles were removed by editing the curves. With the exception of these three editing functions, all the motions shown are raw puppeteered motion. The secondary animators commented that puppeteering the root motion was useful to explore different acting choices and to figure out the appropriate timing of the performances.

Figure 7 illustrates four different acting choices that Anima-

Table 2: Time chart for each blocked-in animation.

| Animation | Puppeteering | Post-processing |
|---|---|---|
| Dialog "Min. Wage" | 60 min. Root, spine, shoulders, R. elbow, head, foot IK, jaw. | 1 min. spline fitting |
| Long jump | 20 min. Root, head, spine, foot IK, shoulders, elbows. | 10 min. Pelvis cleanup for landing and takeoff spline fitting (only pelvis). |
| Scared man | 5 min. Root, spine, neck. | 15 min. Edit arms and legs. |
| Penguin | 20 min. Root, spine, head, shoulders. | 10 min. Edit curve of spine, edit head on final, move to a camera, spline fitting (spine and head). |
| Dragon | 35 min. Root, neck, head, wings. | 5 min. Edit curves of root. |
| Octopus | 60 min. Root, eight legs (four DOFs each), head, three seaweeds (11 DOFs each). | 10 min. Edit curves of legs. |

tor 1 created for a single dialog. Each animation was created with 20 minutes of puppeteering and 20 minutes of post-processing in Maya. While the motions are different, each animation is compelling for the dialog. These animations indicate that the animator could quickly test out a variety of performances with the puppeteering interface.

Figure 8 illustrates two whole body motions that were performed via puppeteering and traditional whole body motion capture. The puppeteered motions were performed by one of the secondary animators The motion capture actor was shown the rendered puppeteered clip and asked to perform the motion in as similar a fashion as possible. The green and red trajectories illustrate the trajectory of the pelvis in the image plane. The left image shows a walk sequence. The puppeteered character has a larger up/down oscillation of the pelvis and is able to achieve a greater compression of the ankle joint than would be possible for an actor. The right figure shows a jump. The puppeteered character jumps higher and has a much longer hang time as illustrated by the highest dark green circle. The puppeteered character also holds a few poses for longer during the preparation for the jump (darker green circles) while the motion capture character holds just the start and end pose and moves continuously throughout the rest of the motion.

These examples illustrate key differences between the motion used typically for animated films and that produced by motion capture. The motions that our animators chose to perform contained significant violations of physics (hang time), constraints of natural human motion (joint angles), and the flow of natural human motion (holding poses rather than minimizing jerk). These intentional violations are often present in the final keyframed animation as well. This graphical evidence provided further proof that puppeteering is a better choice for creating motion for animated films than motion capture mapping.

## 5 Discussion

In this paper, we have presented a design for a puppeteering interface that allows an animator to quickly block in motion for a shot. We took an iterative refinement approach with professional animators, and designed an interface that gave them full control of the character's motion via puppeteering. This interface allowed them to very quickly experiment with and discard different performances.

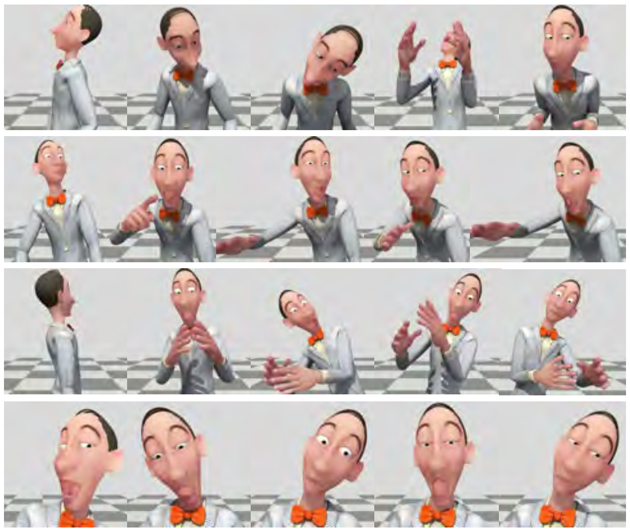


Figure 7: Four acting choices for a dialog, "Are you a good dog?"

Our animators could make reasonable blocked-in animations with the puppeteering interface more quickly and efficiently than with a typical keyframing interface. They could also easily express their imagination and test out a wide variety of animations for a single scenario/dialog. We believe that the rapid turnaround provided by such an interface could make a fundamental change in the level and quality of communication between an animator and the director of a production. The design of the interface was fundamentally altered through a testing process with a group of seven professional animators. The resulting interface was viewed as an effective tool by those animators and they used it to produce a number of animations with different characters, behaviors, and style of motion. The user study results and the comments from our subjects provide support for our hypothesis that puppeteering motion gives a better feeling for the timing of the performance and makes the blocking-in process more efficient than a traditional keyframing interface.

Puppeteering is in fact much more powerful than motion capture in that the animated character no longer has to adhere to the constraints of physics or human physiology. A motion capture actor moves his or her body in the real world and therefore can only produce physically correct patterns of human motion for each limb. We identified a number of classes of actions where these violations frequently occur: a reduction in gravity, extreme joint torques, increased range of motion or telescoping limbs for the appearance of a rubbery skeleton or squash-and-stretch, and characters with kinematics that are far from those of an adult human. A more subtle difference between keyframe animation and motion capture is in the style of the motion. Classical animation tends to consist of a series of held poses while motion capture is necessarily always in motion as actors cannot remain completely still. Puppeteering has the ability to achieve all of these characteristics: fluid motions, held poses, and physically impossible motions.

The trade-off for this extra freedom is that the animator must have the skills to keep the motion on model for a particular character, just as with a keyframing interface. Although our animators were able to learn the interface quickly, it was also apparent that puppeteering does not precisely match the skill set required for keyframing and that it would take some time for them to become truly proficient in the puppeteering of motions.

Although puppeteering appears to be an intuitive and effective

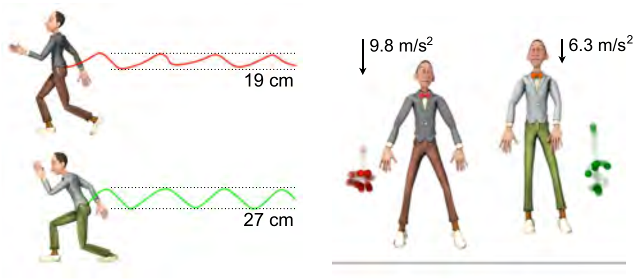


Figure 8: Images and traces of animated motion that illustrate the differences between the puppeteered motion (green) and full body motion capture (brown). Left: The oscillation of the pelvis is 19 cm for the motion capture and 27 cm for the puppeteered motion. Right: Gravity is 6.3 m/s$^2$ for the puppeteered jump while it is 9.8 m/s$^2$ for the motion capture jump.

way to create blocked-in motion, there are a few types of motions for which the interface does not work well. Most of these are motions that involve tight synchronization between the DOFs of the character or involve contact with the environment. For example, if a character is to experience a big impact, there are many joints which must respond with exactly the same timing. Puppeteering makes it difficult to hit that timing exactly but motions could be aligned in time as a post-processing step in Maya. The precision required for contact with the environment also presented difficulties. For some motions such as standing, a good choice of home position made it easy to puppeteer the motion as the table provided the needed constraint. But a motion such as grasping would be difficult to puppeteer as the hand must be precisely positioned with respect to the object when the grasp is performed or the physics of the grasp will be violated. However, motions involving precise contact can be created or refined with IK in commercial animation software such as Maya.

Another type of motion that the puppeteering interface was not effective for was facial motion. Blendshapes could be puppeteered if the character was rigged for them, but keyframing blendshapes would probably afford more precise control.

Although the results are compelling, we have yet to use this in an actual production and therefore do not have definitive numbers about the time savings that might result. However, the rapidity with which performances can be explored should allow the animator to try out more alternatives and to iterate much more frequently with the director to converge on the best performance for a particular scene. The performance aspect of the interface may even permit an interactive session with the director and animator trading off control of the degrees of freedom of the character to show how the motion should be "more like this."

We also speculate that perhaps an interface such as this that removes the hands from the keyboard and mouse might help reduce the frequency of repetitive strain injury in the animation community by greatly increasing the variety of hand and wrist motions performed in creating an animation.

Although the examples reported here focused on blocked-in animation, puppeteering could also be used in other domains where it is necessary to quickly provide a simple version of motion. For example, it could be also used to choreograph stage performances where the motion of the actors around the stage must be blocked in. Puppeteering is also useful for creating motion for simple robots such as the Keepon robot [9] in real time or for more complex robots if the motion is created in layers and optimized to be within the mechanical limits of the robots. Extending the puppeteering interface for such purposes would be an interesting future direction.

## Acknowledgements

This work was done when Takaaki Shiratori was at Disney Research, Pittsburgh and Warren Trezevant was at Pixar Animation Studios. These animations were created at Pixar Animation Studios, Walt Disney Animation Studios, and Black Rock Studio.

## References

[1] J. Chai and J. K. Hodgins. Performance animation from low-dimensional control signals. *ACM Trans. on Graphics*, 24(3), 2005.
[2] M. Dontcheva, G. Yngve, and Z. Popović. Layered acting for character animation. *ACM Trans. on Graphics*, 22(3), 2003.
[3] C. Esposito, W. B. Paley, and J. Ong. Of mice and monkeys: a specialized input device for virtual body animation. In *Proc. ACM SIGGRAPH Symp. on Interactive 3D Graphics*, 1995.
[4] T.-C. Feng, P. Gunawardane, J. Davis, and B. Jiang. Motion capture data retrieval using an artist's doll. In *Proc. Int'l Conf. on Pattern Recognition*, 2008.
[5] T. Igarashi, T. Moscovich, and J. F. Hughes. Spatial keyframing for performance-driven animation. In *Proc. ACM SIGGRAPH/Eurographics Symp. on Computer Animation*, 2005.
[6] M. P. Johnson, A. Wilson, B. Blumberg, C. Kline, and A. Bobick. Sympathetic interfaces: Using a plush toy to direct synthetic characters. In *Proc. ACM SIGCHI Conf. on Human Factors in Computing Systems*, 1999.
[7] B. Knep, C. Hayes, R. Sayre, and T. Williams. Dinosaur input device. In *Proc. ACM SIGCHI Conf. on Human Factors in Computing Systems*, 1995.
[8] T. Komura and W.-C. Lam. Real-time locomotion control by sensing gloves. *Computer Animation and Virtual Worlds*, 17:513–525, December 2006.
[9] H. Kozima, M. Michalowski, and C. Nakagawa. Keepon. *Int'l J. of Social Robots*, 1(1):3–18, 2009.
[10] J. Lasseter. Principles of traditional animation applied to 3D computer animation. *Computer Graphics*, 21(4), 1987.
[11] J. Laszlo, M. van de Panne, and E. Fiume. Interactive control for physically-based animation. In *Proc. ACM SIGGRAPH*, 2000.
[12] Z. Luo, I.-M. Chen, S. H. Yeo, C.-C. Lin, and T.-Y. Li. Building hand motion-based character animation: The case of puppetry. In *Proc. Int'l Conf. on Cyberworlds*, 2010.
[13] J. Marshall and E. G. Walsh. Physiological tremor. *J. of Neurology, Neurosurgery and Psychiatry*, 19, 1956.
[14] M. Neff, I. Albrecht, and H.-P. Seidel. Layered performance animation with correlation maps. *Computer Graphics Forum*, 26(3), 2007.
[15] N. Numaguchi, A. Nakazawa, T. Shiratori, and J. K. Hodgins. A puppet interface for retrieval of motion capture data. In *Proc. ACM SIGGRAPH/Eurographics Symp. on Computer Animation*, 2011.
[16] S. Oore, D. Terzopoulos, and G. Hinton. A desktop input device and interface for interactive 3D character animation. In *Proc. Graphics Interface*, 2002.
[17] H. J. Shin, J. Lee, S. Y. Shin, and M. Gleicher. Computer puppetry: An importance-based approach. *ACM Trans. on Graphics*, 20(2), 2001.
[18] R. M. Taylor, T. C. Hudson, A. Seeger, H. Weber, J. Juliano, and A. T. Helser. VRPN: a device-independent, network-transparent VR peripheral system. In *Proc. ACM Symp. on Virtual Reality Software and Technology*, 2001.
[19] F. Thomas and O. Johnston. *The Illusion of Life: Disney Animation*. Disney Editions, 1995.
[20] G. Walters. The story of Waldo C. Graphic. In *ACM SIGGRAPH Course Notes: 3D Character Animation by Computer*, 1989.
[21] K. Yamane and Y. Nakamura. Natural motion animation through constraining and deconstraining at will. *IEEE Trans. on Visualization and Computer Graphics*, 9(3), 2003.
[22] W. Yoshizaki, Y. Sugiura, A. C. Chiou, S. Hashimoto, M. Inami, T. Igarashi, Y. Akazawa, K. Kawachi, S. Kagami, and M. Mochimaru. An actuated physical puppet as an input device for controlling a digital manikin. In *Proc. ACM SIGCHI Conf. on Human Factors in Computing Systems*, 2011.