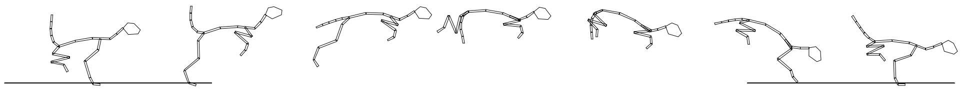
Interactive Control For Physically-Based Animation

Joseph Laszlo Michiel van de Panne Eugene Fiume

Department of Computer Science
University of Toronto[1]



Abstract

We propose the use of interactive, user-in-the-loop techniques for controlling physically-based animated characters. With a suitably designed interface, the continuous and discrete input actions afforded by a standard mouse and keyboard allow for the creation of a broad range of motions. We apply our techniques to interactively control planar dynamic simulations of a bounding cat, a gymnastic desk lamp, and a human character capable of walking, running, climbing, and various gymnastic behaviors. The interactive control techniques allows a performer's intuition and knowledge about motion planning to be readily exploited. Video games are the current target application of this work.

CR Categories: I.3.6 [Computer Graphics]: Methodology and Techniques— Interaction Techniques; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation; I.6.8 [Simulation and Modeling]: Types of Simulation—Animation

Keywords: physically based animation, user interfaces

1 Introduction

Interactive simulation has a long history in computer graphics, most notably in flight simulators and driving simulators. More recently, it has become possible to simulate the motion of articulated human models at rates approaching real-time. This creates new opportunities for experimenting with simulated character motions and behaviors, much as flight simulators have facilitated an unencumbered exploration of flying behaviors.

Unfortunately, while the controls of an airplane or an automobile are well known, the same cannot be said of controlling human or animal motions where the interface between our intentions and muscle actions is unobservable, complex, and ill-defined. Thus, in order to create a tool which allows us to interactively experiment with the dynamics of human and animal motions, we are faced with the task of designing an appropriate interface for animators. Such an interface needs to be sufficiently expressive to allow the creation of a large variety of motions while still being tractable to learn.

Performance animation and puppetry techniques demonstrate how well performers can manage the simultaneous control of a large number of degrees of freedom. However, they are fundamentally kinematic techniques; if considerations of physics are to be added, this is typically done as a post-process. As a result, they do not lend themselves well to giving a performer a sense of embodiment for animated figures whose dynamics may differ significantly from that of the performer. In contrast, the physics of our simulated characters constrains the evolution of their motions in significant ways.

We propose techniques for building appropriate interfaces for interactively-controlled physically-based animated characters. A variety of characters, motions, and interfaces are used to demonstrate the utility of this type of technique. Figure 1 shows an example interface for a simple articulated figure which serves as a starting point for our work and is illustrative of how a simple interface can provide effective motion control.

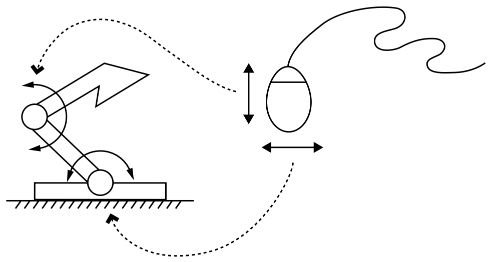


Figure 1: Interactive control for Luxo, the hopping lamp.

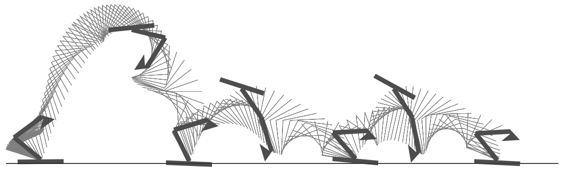


Figure 2: Example of an interactively controlled back head-springs and back-flip for Luxo.

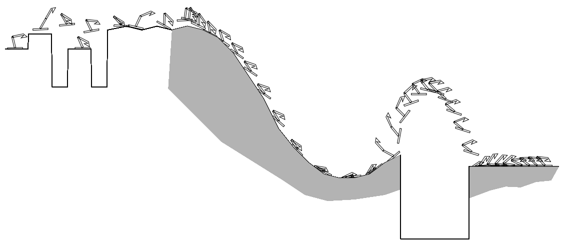


Figure 3: Example of an interactively controlled animation, consisting of hops across variable terrain and a carefully timed push off the ski jump.

[1]{jflaszlo|van|elf}@dgp.utoronto.ca
http://www.dgp.utoronto.ca/~ jflaszlo/interactive-control.html

This planar model of an animated desk lamp has a total of 5 degrees of freedom (DOF) and 2 actuated joints, capable of exerting joint torques. The motion is governed by the Newtonian laws of physics, the internal joint torques, the external ground forces, and gravity. The joint torques are computed using a proportional-derivative (PD) controller, namely $\tau = k_p(\theta_d - \theta) - k_d\dot{\theta}$. The motions of the two joints are controlled by linearly mapping the mouse position, (m_x, m_y), to the two desired joint angles, θ_d.

Using this interface, coordinated motions of the two joints correspond to tracing particular time-dependent curves with the mouse. A rapid mouse motion produces a correspondingly rapid joint motion. With this interface one can quickly learn how to perform a variety of interactively controlled jumps, shuffles, flips, and kips, as well as locomotion across variable terrain. With sufficient practice, the mouse actions become gestures rather than carefully-traced trajectories. The interface thus exploits both an animator's motor learning skills and their ability to reason about motion planning.

Figure 2 shows an example of a gymnastic tumbling motion created using the interface. This particular motion was created using several motion *checkpoints*. As will be detailed later, these facilitate correcting mistakes in executing particularly unstable or sensitive motions, allowing the simultion to be rolled back to previous points in time. Figure 3 shows a user-controlled motion over variable terrain and then a slide over a ski jump, in this case performed without the use of any checkpoints. The sliding on the ski hill is modelled by reducing the ground friction coefficient associated with the simulation, while the jump is a combined result of momentum coming off the lip of the jump and a user-controlled jump action.

This initial example necessarily provokes questions about scalability, given that for more complex characters such as a horse or a cat, one cannot hope to independently control as many input DOF as there are controllable DOF in the model. One possible solution is to carefully design appropriate one-to-many mappings from input DOF to output DOF. These mappings can take advantage of frequently occuring synergetic joint motions as well as known symmetry and phase relationships.

We shall also explore the use of discrete keystrokes to complement and/or replace continuous input DOF such as that provided by the mouse. These enrich the input space in two significant ways. First, keys can each be assigned their own action semantics, thereby allowing immediate access to a large selection of actions. This action repertoire can easily be further expanded if actions are selected based upon both the current choice of keystroke and the motion context of the keystroke. Second, each keystroke also defines *when* to perform an action as well as the selection of *what* action. The timing of keystrokes plays an important role in many of our prototype interfaces.

In its simplest form, our approach can be thought of as sitting squarely between existing virtual puppetry systems and physically-based animation. It brings physics to virtual puppetry, while bringing interactive interfaces to physically-based animation. The system allows for rapid, free-form exploration of the dynamic capabilities of a given physical character design.

The remainder of this paper is structured as follows. Section 2 reviews previous related work. Section 3 describes the motion primitives used in our prototype system. Section 4 illustrates a variety of results. Finally, section 5 provides conclusions and future work.

2 Previous Work

Building kinematic or dynamic motion models capable of reproducing the complex and graceful movements of humans and animals has long been recognized as a challenging problem. The book *Making Them Move*[3] provides a good interdisciplinary primer on some of the issues involved. Using physical simulation techniques to animate human figures was proposed as early as 1985[2]. Since then, many efforts have focussed on methods of computing appropriate control functions for the simulated actuators which will result in a desired motion being produced. Among the more popular methods have been iterative optimization techniques [8, 13, 19, 23, 29, 30], methods based on following kinematically-specified reference trajectories [15, 16], suitably-designed state machines [9], machine learning techniques[14], and hybrids[5, 12].

A number of efforts have examined the interactive control of dynamically-simulated articulated figures[10, 11] or procedurally-driven articulated figures[6]. The mode of user interaction used in these systems typically involves three steps: (1) setting or changing specific parameters (2) running the simulation, and (3) observing the result. This type of observe and edit tools is well suited to producing highly specific motions. However, the interaction is less immediate than we desire, and it does not lend a performer a sense of embodiment in a character.

Motion capture and virtual puppetry both allow for user-in-the-loop kinematic control over motions[17, 21, 24], and have proven effective for specific applications demanding real-time animation. The use of 2d user gestures to specify object motion[4] is in an interesting early example of interactive computer mediated animation. Physical animatronic puppets are another interesting precedent, but they cannot typically move in an uncontrained and dynamic fashion in their environment. The system described in [7] is a novel application of using a haptic feedback device for animation control using a mapping which interactively interpolates between a set of existing animations. Our work aims to expand the scope of interactive real-time interfaces by using physically-based simulations, as well as exploring interfaces which allow various degrees of motion abstraction. Such interfaces could then perhaps also be applied to the control of animatronic systems.

The work of Troy[25, 26, 27] proposes the use of manual manipulation of several input devices to perform low-level control of the movement of bipedal characters. The work documents experiments with a variety of input devices and input mappings as having been performed, although detailed methods and results are unfortunately not provided for the manual control method. Nevertheless, this work is among the first we know of that points out the potential of user-in-the-loop control methods for controlling unstable, dynamic motions such as walking.

Computer and video games offer a wide variety of interfaces based both on continuous-input devices (mice, joysticks, etc.) and button-presses and/or keystrokes. However, the current generation of games do not typically use physically-based character animation, nor do they allow much in the way of fine-grained motion control. Exceptions to the rule include fighting games such as *Die by the Sword*[20] and Tekken[18]. The former allows mouse and keyboard control of a physically-based model, limited to the motion of the sword arm. The latter, while kinematic in nature, affords relatively low-level control over character motions. Telerobotics systems[22] are a further suitable example of interactive control of dynamical systems, although the robots involved are typically anchored or highly stable, and are in general used in constrained settings not representative of many animation scenarios.

3 Motion Primitives

The motion primitives used to animate a character can be characterized along various dimensions, including their purpose and their implementation. In this section we provide a classification based largely on the various interface methods employed in our example scenarios.

The joints of our simulated articulated figures are all controlled by the use of PD controllers. Motion primitives thus control motions by varying the desired joint angles used by the PD controllers, as well as the associated stiffness and damping parameters.

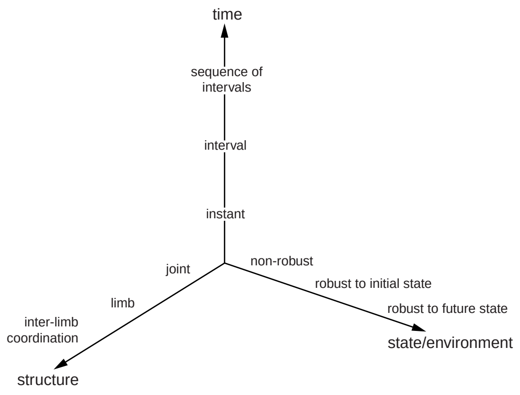


Figure 4: Three dimensions of control abstraction.

PD controllers provide a simple, low-level control mechanism which allows the direct specification of desired joint angles. Coping with more complex characters and motions necessitates some form of abstraction. Figure 4 shows three dimensions along which such motion abstraction can take place. The interfaces explored in this paper primarily explore abstractions in time and structure by using stored control sequences and coordinated joint motions, respectively. The remaining axis of abstraction indicates the desirability of motion primitives which perform correctly irrespective of variations in the initial state or variations in the environment. This third axis of abstraction is particularly challenging to address in an automated fashion and thus our examples rely on the user-in-the-loop to perform this kind of abstraction.

3.1 Continuous Control Actions

The most obvious way to control a set of desired joint angles is using an input device having an equivalent number of degrees of freedom. The mouse-based interface for the hopping lamp (Figure 1) is an illustration of this. It is interesting to note for this particular example that although cursor coordinates are linearly mapped to desired joint angles, a nonlinearity is introduced by the acceleration features present in most mouse-drivers. This does not seem to adversely impact the continuous control. In general, continuous control actions are any mappings which make use of continuously varying control parameters, and are thus not limited to direct mappings of input DOF to output DOF.

The availability of high DOF input devices such as data-gloves and 6 DOF tracking devices means that the continuous control of input DOF can potentially scale to control upwards of 20 degrees of freedom. However, it is perhaps unreasonable to assume that a performer can learn to simultaneously manipulate such a large number of DOF independently, given that precedents for interfaces in classical puppetry and virtual puppetry are typically not this ambitious.

3.2 Discrete Control Actions

Discrete actions, as implemented by keystrokes in our interfaces, allow for an arbitrary range of action semantics. Action games have long made extensive use of keystrokes for motion specification, although not at the level of detail that our interfaces strive to provide. The following list describes the various action semantics used in prototype interfaces, either alone or in various combinations. Some of the actions in this list refer directly to control actions, while others serve as meta-actions in that they modify parameters related to the simulation and the interface itself.

set joint position (absolute) Sets desired position of joint or a set of joints to a prespecified value(s). If all joints are set simultaneously in order to achieve a desired pose for the figure, this becomes a form of interactive dynamic keyframing.

adjust joint position (relative) Changes the desired position of a joint or set of joints, computed relative to current desired joint positions.

grasp, release Causes a hand or foot to grasp or release a nearby point (e.g., ladder rung) or to release a grasped point.

select IK target Selects the target point for a hand or foot to reach toward using a fixed-time duration IK trajectory, modelled with a Hermite curve. The IK solution is recomputed at every time step.

initiate pose sequence Initiate a prespecified sequence of full or partial desired poses.

select next control state Allows transitions between the states of a finite-state machine; useful for modelling many cyclical or otherwise well-structured motions, leaving the timing of the transitions to the performer.

rewind, reset state Restarts the simulation from a previous state checkpoint.

set joint stiffness and damping Sets the stiffness and damping parameters of the PD joint controllers to desired values.

select control mode Chooses a particular mapping for a continuous input device, such as which joints the mouse controls.

set simulation rate Speeds up or slows down the current rate of simulation; helps avoid a motion happening 'too fast' or 'too slow' to properly interact with it.

set state checkpoint Stores the system state (optionally during replay/review of a motion) so that simulation may be reset to the same state later if desired.

modify physical parameters Effects changes to simulation parameters such as gravity and friction.

toggle randomized motion Begins or halts the injection of small randomized movements, which are useful for introducing motion variation.

Our default model for arbitration among multiple actions which come into conflict is to allow the most recent action to pre-empt any ongoing actions. Ongoing actions such as IK-based trajectories or pose sequences are respectively preempted only by new IK-based trajectories or pose sequences.

3.3 State Machines

Given the cyclic or strongly structured nature of many motions, state machine models are useful in helping to simplify the complexities of interactive control. For example, they allow separate actions such as 'take left step' and 'take right step' to be merged into a single action 'take next step', where a state machine provides the necessary context to disambiguate the action. As with many other animations systems, state machines serve as the means to provide apriori knowledge about the sequencing of actions for particular classes of motion.

4 Implementation and Results

Our prototype system is based on a number of planar articulated figures. The planar dynamics for these figures can easily be computed at rates suitable for interaction (many in real-time) on most current PCs and offer the additional advantage of having all aspects of their motion visible in a single view, thereby providing unobstructed visual feedback for the performer. Our tests have been conducted primarily on a 450 Mhz Mac and a 366 Mhz PII PC. While hard-coded interfaces were used with the original prototyping system behind many of our results, our more recent system uses `Tcl` as a scripting language for specifying the details of any given interface. This facilitates rapid iteration on the design of any given interface, even potentially allowing changes during the course of a simulation.

4.1 Luxo Revisited

Using the continuous-mode mouse-based interface shown in Figure 1, the desklamp is capable of executing a large variety of hops, back-flips, a kip manoevre, head-stands, and motion across variable terrain. This particular interface has been tested on a large number of users, most of whom are capable of performing a number of the simpler movements within 10–15 minutes, given some instruction on using the interface. Increasing the stiffness of the joints or scaling up the mapping used for translating mouse position into desired joint angles results in the ability to perform more powerful, dynamic movements, although this also makes the character seem rather too strong during other motions.

We have additionally experimented with a keystroke-based interface using 14 keys, each key invoking a short sequence of prespecified desired poses of fixed duration. The various key actions result in a variety of hops and somersaults if executed from the appropriate initial conditions. The repertoire of action sequences and associated keystrokes are given in the Appendix. The animator or performer must choose when to execute keystrokes and by doing so selects the initial conditions. The initiation of a new action overrides any ongoing action.

The keystroke-based interface was created after gaining some experience with the continuous-mode interface. It provides an increased level of abstraction for creating motions and is easier to learn, while at the same time trading away some of the flexibility offered by the continuous-mode interface. Lastly, user-executed continuous motions can be recorded and then bound to a keystroke.

4.2 Animating a Cat

Experiments with a planar bounding cat and a planar trotting cat are a useful test of scalability for our interactive interface techniques. Figure 5 illustrates the planar cat as well as sets of desired angles assumed by the legs for particular keystrokes. In one control mode, the front and back legs each have 6 keys assigned to them, each of which drives the respective leg to one of the 6 positions illustrated in the figure. The keys chosen for each pose are assigned a spatial layout on the keyboard which reflects the layouts of the desired poses shown in the figure. An additional pose is provided which allows each leg to exert a larger pushing force than is otherwise available with the standard set of 6 poses. This can be achieved by temporarily increasing the stiffness of the associated leg joints, or by using a set of hyperextended joint angles as the desired joint positions. We use the latter implementation. This seventh overextended pose is invoked by holding the control key down when hitting the key associated with the backwards extended leg pose.

The animation sequence shown in Figure 6 was accomplished using 12 *checkpoints*. A checkpoint lets the performer restart the

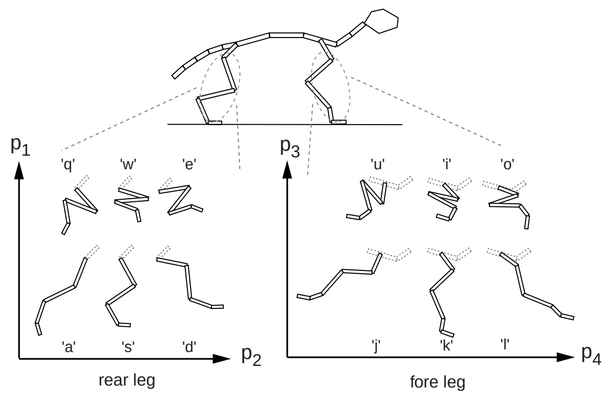


Figure 5: Parameterization of limb movements for cat.

simulation from a given point in time, allowing the piecewise interactive construction of sequences that would be too long or too error-prone to perform successfully in one uninterrupted attempt. Checkpoints can be created at fixed time intervals or at will by the performer using a keystroke. Some of the sequences between checkpoints required only 2 or 3 trials, while particularly difficult aspects of the motion required 10–20 trials, such as jumping the large gap and immediately climbing a set of steps (second-last row of Figure 6).

The cat weighs 5 kg and is approximately 50 cm long, as measured from the tip of the nose to the tip of the tail. Its small size leads to a short stride time and requires the simulation to be slowed down considerably from real-time in order to allow sufficient reaction time to properly control its motions. The cat motions shown in Figure 6 were controlled using a slowdown factor of up to $40\times$, which allows for 10–15 seconds to control each bound.

It is important to note that there is a 'sweet spot' in choosing the speed at which to interact with a character. Important features of the dynamics become unintuitive and uncontrollable if the interaction rate is either too slow or too fast. When the simulation rate is too fast, the user is unable to react quickly enough to correct errors before the motion becomes unsavable. When the motion is too slow, the user tends to lose a sense of the necessary rhythm and timing required to perform the motion successfully and lacks sufficient immediate feedback on the effects of the applied control actions. For basic bounding, a slowdown factor around 10, giving a bound time of 2-3 seconds is sufficient. For more complex motions such as leaping over obstacles, a factor of up to 40+ is required.

Figure 7 shows a trotting motion for a planar 4-legged cat model. The trotting was interactively controlled using only the mouse. The x, y mouse coordinates are used to linearly interpolate between predefined poses corresponding to the six leg poses shown in Figure 5. The poses are laid out in a virtual 2×3 grid and bilinear interpolation is applied between the nearest 4 poses according to the mouse position. The simplest control method assumes a fixed phase relationship among the 4-legs, allowing the mouse to simultaneously effect coordinated control of all legs. A more complex method uses the same mapping to control one leg at a time. This latter method met with less success, although was not pursued at length. The cat model is comprised of 30 articulated links, which makes it somewhat slow to simulate, given that we currently do not employ $O(n)$ forward dynamics methods.

4.3 Bipedal Locomotion

We have experimented with a number of bipedal systems which are capable of more human-like movements and behaviors such as walking and running. For these models, we make extensive use

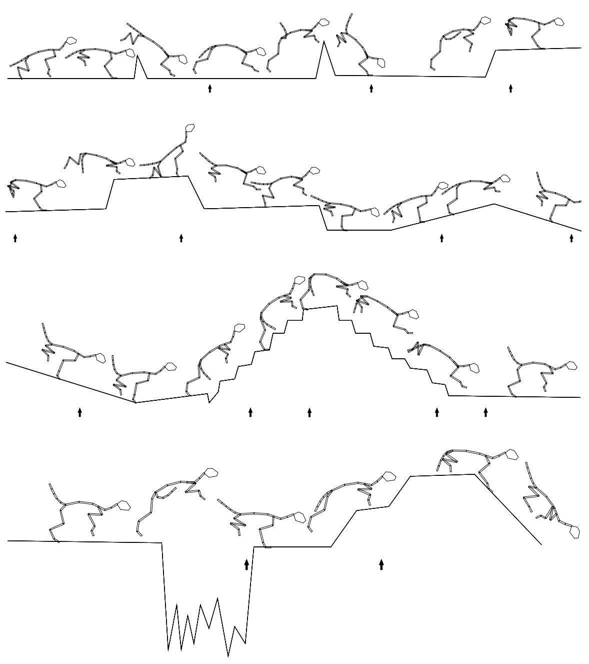


Figure 6: Cat bounding on variable terrain using piecewise interactive key-based control. The frames shown are manually selected for visual clarity and thus do not represent equal samples in time. The arrows indicate when the various checkpoints were used, denoting the position of the shoulders at the time of the checkpoint.

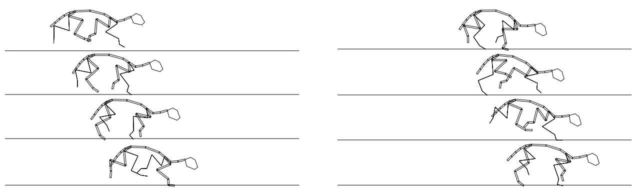


Figure 7: Cat trot using continuous mouse control. The animation reads from top-to-bottom, left-to-right. The first seven frames represent a complete motion cycle. The frames are equally spaced in time.

of a hybrid control technique which mixes continuous and discrete input actions in addition to purely discrete methods similar to those used with the cat and Luxo models. We have also experimented with a wide variety of other bipedal motions in addition to walking and running, including a number of motions such as a long jump attempt and a fall-recovery sequence that are readily explored using interactive control techniques.

Figure 8 shows the interface for an interactive walking control experiment. The mouse is used to control the desired angles for the hip and knee joints of the swing leg. A keypress is used to control when the exchange of stance and swing legs occurs and therefore changes the leg currently under mouse control. The stance leg assumes a straight position throughout the motion. The bipedal figure has human-like mass and dimensions, although it does not have a separate ankle joint. In our current implementation, joint limits are not enforced, although such constraints can easily be added to the simulation as desired.

An example of the resulting motion is shown in Figure 9. With some practice, a walk cycle can be sustained indefinitely. With significant practice, the walk can be controlled in real-time, although a simulation speed of 2–3 times slower than real-time provides a

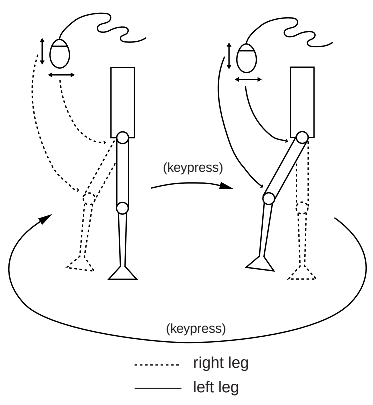


Figure 8: Interface for interactive control of bipdal walking.

sweet-spot for consistent interactive control. It is also possible to choose a particular (good) location for the mouse, thus fixing the desired joint angles for the swing leg, and achieve a marching gait by specifying only the the time to exchange swing and stance legs by pressing a key. This marching motion is quite robust and is able to traverse rugged terrain with reasonable reliability. Yet another mode of operation can be achieved by automatically triggering the swing-stance exchange when the forward lean of the torso exceeds a fixed threshold with respect to the vertical. With this automatic mechanism in place, it is then possible to transition from a marching walk to a run and back again by slowly moving only the mouse through an appropriate trajectory.

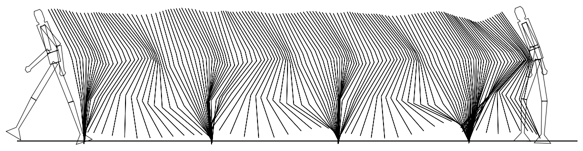


Figure 9: Bipedal walking motion

Figure 10 shows the results of a biped performing a long-jump after such an automatic run. This particular biped dates from earlier experiments and is smaller in size and mass than the more anthropomorphic biped used for the walking experiments. This motion makes use of the same interface as for the bipedal walking motion, shown in in Figure 8. A slowdown factor of up to 80 was necessary because of the small size of the character, as well as the precision required to achieve a final landing position having the legs extended and the correct body pitch. Approximately 20 trials are required to achieve a recognizable long jump, each beginning from a motion checkpoint one step before the final leap. However, we anticipate that the interface can be also be improved upon significantly by using a more reasonable default behavior for the uncontrolled leg.

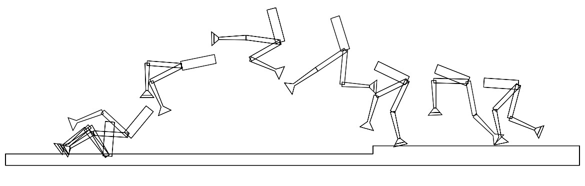


Figure 10: A long jump attempt.

4.4 Bipedal Gymnastics

Several other experiments were carried out using the bipedal figures with continuous-mode mouse control and one or more keys to select the mapping of the continuous input onto the model's desired joint angles. The basic types of motion investigated include a variety of climbing modes both with the bipedal model ”facing” the view plane and in profile in the view plane, and swinging modes both with arms together and separated. Nearly every mapping for these control modes uses the mouse y coordinate to simultaneously drive the motion of all limb joints (hips, knees, shoulders and elbows) in a coordinated fashion and the mouse x coordinate to drive the bending of the waist joint to alter the direction of the motion.

The control modes differ from each other primarily in the particular symmetries shared between the joints. Figure 11 illustrates two forms of symmetry used for climbing ”gaits” similar in pattern to those of a quadruped trotting and bounding. The mapping of the mouse x coordinate onto the waist joint is also shown. The control modes can produce interactive climbing when coupled with a state machine that grasps and releases the appropriate hands and feet each time a key is pressed (assuming that the hands and feet are touching a graspable surface). Swinging modes perform in a similar manner but use the mouse x or y coordinate to swing the arms either back-and-forth at the shoulder or in unison and can make use of either graspable surfaces or ropes that the user can extend and retract from each hand on demand. When used on the ground without grasping, these same modes of interaction can produce a range of gymnastic motions including handstands and different types of flips and summersaults, in addition to a continuously controlled running motion. Among the various interesting motions that are possible is a backflip done by running off a wall, a gymnastic kip from a supine position to a standing position and a series of giant swings as might be performed on a high bar. While not illustrated here, these motions are demonstrated in the video segments and CD-ROM animations associated with this paper.

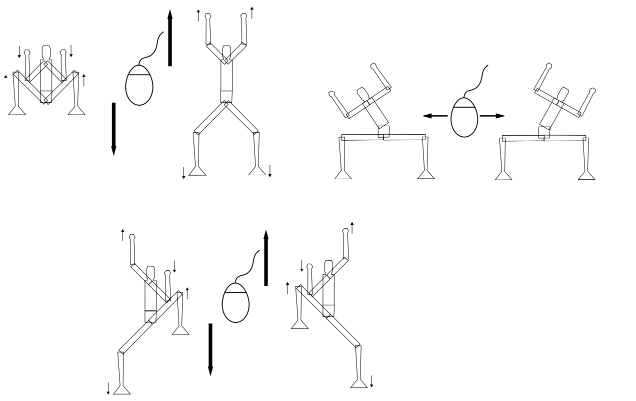


Figure 11: Control modes useful for climbing and gymnastics.

4.5 Using IK Primitives

Figures 12 and 13 illustrate interactively-controlled movements on a set of irregularly-spaced monkeybars and a ladder, respectively. These are movements which require more precise interactions of the hands and feet with the environment than most of the other motions discussed to date. To deal with this, we introduce motion primitives which use inverse kinematics (IK) to establish desired joint angles.

In general, IK provides a rich, abstract motion primitive that can appropriately hide the control complexity inherent in many semantically-simple goal-directed actions that an interactive character might want to perform. This reduces the associated learning curve the user faces in trying to discover how to perform the action(s) from first principles while still taking good advantage of the user's intuition about the motion.

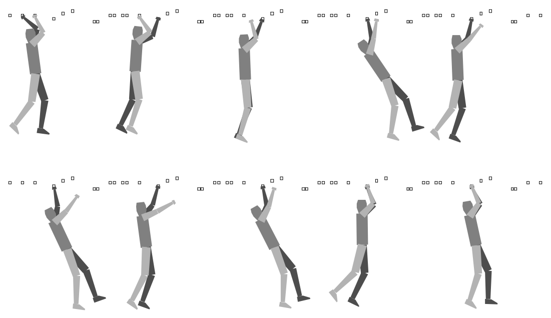


Figure 12: Traversing a set of irregularly-spaced monkeybars.

The interface for monkey-bar traversal consists of keystrokes and a state machine. IK-based trajectories for the hands and feet are invoked on keystrokes. The hand-over-hand motion across the monkeybars is controlled by keys which specify one of three actions for the next release-and-regrasp motion. The actions causes the hand to release its grasp on the bar and move towards the previous bar, the current bar, or the following bar. These actions can also be invoked even when the associated hand is not currently grasping a bar, which allows the figure to recover when a grasp manoevre fails due to bad timing. The interface does not currently safeguard against the premature execution of a regrasp motion with one hand while the other has not yet grasped a bar. The character will thus fall in such situations. A grasp on a new bar is enacted if the hand passes close to the target bar during the reaching action, where ‘close’ is defined to be a fixed tolerance of 4 cm in our example. Controlling the motion thus involves carefully choosing the time in a swing at which a new reach-and-grasp action should be initiated, as well as when to pull up with the current support arm. More information about the particulars of the interface is given in the Appendix.

The ladder climbing example is made up of a number of keys which serve to position the body using the hands and feet which are in contact with the ladder, as well as a key to initiate the next limb movement as determined by the state machine. The details of the interface are given in the appendix, as well as the specific keystroke sequence used to create Figure 13. Note, however, that the keystroke sequence by itself is insufficient to precisely recreate the given motion, as the timing of each keystroke is also important in all the motions discussed.

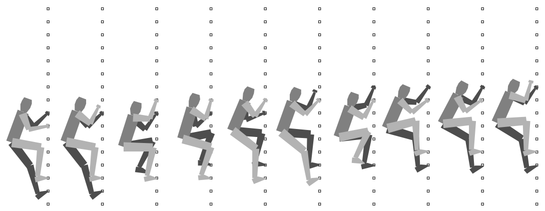


Figure 13: Climbing a ladder.

Finally, Figure 14 illustrates a standing up motion, followed by a few steps, a forward fall, crouching, standing up, and, lastly, a backwards fall. A set of 18 keys serves as the interface for this scenario, as documented in the appendix.

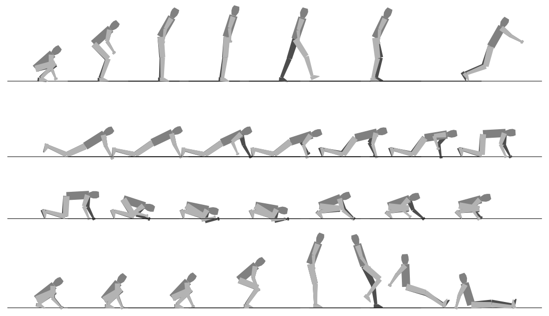


Figure 14: Fall recovery example.

5 Conclusions and Future Work

We have presented prototype interfaces for the interactive control of physically-based character animation. The techniques illustrate the feasibility of adding physics to virtual puppetry, or, alternatively, adding interactive interfaces to physically-based animation. They allow human intuition about motions to be exploited in the interactive creation of novel motions.

The results illustrate that dynamic motions of non-trivial articulated figure models can be reasonably controlled with user-in-the-loop techniques. Our experiments to date have focussed on first achieving a large action repertoire for planar figures, with the goal of using this experience as a suitable stepping stone towards 3D motion control. While it is not clear that the interaction techniques will scale to the type of nuanced 3D motion control need for foreground character animation in production animation, the interfaces could be readily applied to a new generation of physically-based interactive video games. The interfaces provide a compelling user experience — in fact, we found the interactive control experiments to be quite addictive.

One of the drawbacks of using interactive control is the effort required in both designing an appropriate interface and then learning to effectively use the interface. These two nested levels of experimentation necessitate a degree of expertise and patience. We are optimistic that tractable interfaces can be designed to control sylistic variations of complex motions and that animators or game players can learn these interfaces with appropriate training and practice.

Our work has many directions which require further investigation. We are still far from being able to reproduce nuanced dynamic motions for 3D human or animal characters[1, 28]. The large variety of high-DOF input devices currently available offers a possible avenue of exploration. Haptic devices may also play a useful role in constructing effective interfaces[7]. Nuanced performance may potentially require years of training, as it does for other arts (key-frame animation, dancing, music) and sports. We can perhaps expect that the instruments and interfaces required for composing motion to undergo continual evolution and improvement. A large community of users would offer the potential for a rapidly evolving set of interfaces for particular motions or characters.

Many dynamic motions would benefit from additional sensory feedback, such as an animated update of the location of the center of mass[25]. Going in the opposite direction, one could use an interactive environment like ours to conduct experiments as to the minimal subset of sensory variables required to successfully control a given motion. Questions regarding the transfer of skills between interfaces and between character designs are also important to address if broad adoption of interactive control techniques is to be feasible.

The derivation of high-level abstractions of motion control is of interest in biomechanics, animation, and robotics. The training data and insight gained from having a user-in-the-loop can potentially help in the design of autonomous controllers or high-level motion abstractions. A variety of hybrid manual/automatic control methods are also likely to be useful in animation scenarios.

Beyond its application to animation, we believe the system also has potential uses in exploring deeper issues involved in controlling motions for biomechanics, robotics, or animation. What constitutes a suitable motor primitive or ‘motor program’? How can these primitives be sequenced or overlaid in order to synthesize more complex motions? In what situations is a particular motion primitive useful? Our experimental system can serve as a tool towards exploring these questions by allowing interactive control over the execution and sequencing of user-designed motion-control primitives.

Acknowledements: We would like to thank all of the following for their help: the anonymous reviewers for their comments; the Imager lab at UBC for hosting the second author during much of the work on this paper; and David Mould for suggestions and assistance investigating the automatic bipedal marching and running motions. This work was supported by grants from NSERC and CITO.

A Appendix

Details of keystrokes interface for Luxo:

k	small hop
l	medium hop
o	large hop
i	high backward hop
u	medium backward hop
j	small backward hop
y	back somersault
s	sitting to upright (slow)
d	standing to sitting / LB to sitting
f	LB to standing (small height) / standing to sitting
e	LB to standing (medium height) / standing to LB
w	standing to sitting / sitting to LB / LB to standing
q	big jump from base to LB / fwd somersault from LB
a	LB to standing with small jump
	A,B = a single action that performs either A or B depending on initial state
	LB = lying on back

Interface and keystrokes for monkeybar example:

a	grasp rung previous to CR
s	grasp CR
d	grasp rung following CR
f	grasp rung two rungs following CR
q	release with both hands, relax arms
e	pull up using support arm
R	reset to initial state
t	toggle defn of support/grasp arm
	CR = closest rung

Interface and keystrokes for ladder climbing example:

q	release both hands, fall from ladder
f	grasp two rungs higher with next grasp arm
h	shift body up
b	lower body down
n	pull body in with arms
g	push body out with arms
j	push body out with legs
m	pull body in with legs
R	reset to initial state

Interface and keystrokes for the fall recovery example:

p	ST, prepare for forwards fall
o	ST, prepare for backwards fall
t	HK, step back with left arm
y	HK, step back with right arm
q	HK, shift body back
w	HK, bend elbows, prepare for push up
W	HK, straighten elbows, push up
1	CR, straighten hips, knees, ankles
2	CR, assume intermediate pose towards being upright
3	CR, assume final upright pose
c	ST, step backwards with left leg
v	ST, step forwards with left leg
b	ST, step backwards with right leg
n	ST, step forwards with right leg
j	ST, lean back at hips
R	reset to initial state
M	checkpoint current state
L	restart at checkpoint state
	ST = standing
	HK = on hands and knees
	CR = crouched

References

[1] K. Amaya, A. Bruderlin, and T. Calvert. Emotion from motion. In *Graphics Interface '96*, pages 222–229, May 1996.

[2] W. W. Armstrong and M. Green. The dynamics of articulated rigid bodies for purposes of animation. *Proceedings of Graphics Interface '85*, pages 407–415, 1985.

[3] N. I. Badler, B. Barsky, and D. Zeltzer. *Making Them Move*. Morgan Kaufmann Publishers Inc., 1991.

[4] R. M. Baecker. *Interactive computer-mediated animation*. PhD thesis, Massachusetts Institute of Technology, 1969.

[5] A. Bruderlin and T. W. Calvert. Goal-directed animation of human walking. *Proceedings of ACM SIGGRAPH*, 23(4):233–242, 1989.

[6] A. Bruderlin and T. W. Calvert. Interactive animation of personalized human locomotion. *Proceedings of Graphics Interface*, pages 17–23, 1993.

[7] B. R. Donald and F. Henle. Using Haptic Vector Fields for Animation Motion Control. Technical Report PCS-TR99-353, Dartmouth College, Computer Science, Hanover, NH, May 1999.

[8] J. Auslander et al. Further experience with controller-based automatic motion synthesis for articulated figures. *ACM Transactions on Graphics*, October 1995.

[9] J. K. Hodgins et al. Animating human athletics. *Proceedings of SIGGRAPH 95, ACM Computer Graphics*, pages 71–78, 1995.

[10] T. Ertl et al. Interactive control of biomechanical animation. *The Visual Computer*, pages 459–465, 1993.

[11] D. Forsey and J. Wilhelms. Techniques for interactive manipulation of articulated bodies using dynamic analysis. In *Proceedings of Graphics Interface '88*, pages 8–15, 1988.

[12] M. Girard. Interactive design of computer-animated legged animal motion. *IEEE Comptuer Graphics and Applications*, 7(6):39–51, June 1987.

[13] R. Grzeszczuk and D. Terzopoulos. Automated learning of muscle-actuated locomotion through control abstraction. *Proceedings of SIGGRAPH 95, ACM Computer Graphics*, pages 63–70, 1995.

[14] R. Grzeszczuk, D. Terzopoulos, and G. Hinton. Neuroanimator: Fast neural network emulation and control of physics-based models. *Proceedings of SIGGRAPH 98*, pages 9–20, July 1998. ISBN 0-89791-999-8. Held in Orlando, Florida.

[15] H. Ko and N. Badler. Animating human locomotion with inverse dynamics. *IEEE Computer Graphics and Applications*, pages 50–59, 1996.

[16] E. Kokkevis, D. Metaxas, and N. Badler. User-controlled physics-based animation for articulated figures. In *Proceedings of Computer Animation '96*, 1996.

[17] A. Menache. *Understanding Motion Capture for Computer Animation and Video Games*. Morgan Kaufmann, 1999.

[18] Namco. Tekken, tekken2, and tekken3. computer game, 1998, 1999.

[19] J. T. Ngo and J. Marks. Spacetime constraints revisited. *Proceedings of SIGGRAPH 93*, pages 343–350, 1993.

[20] Interplay Productions. Die by the sword. computer game, 1998.

[21] M. Rosenthal. Dynamic digital hosts. *Computer Graphics World's Digital Magic*, pages 39–42, August 1998.

[22] T. B. Sheridan. *Telerobotics, Automation, and Human Supervisory Control*. MIT Press, 1992.

[23] K. Sims. Evolving virtual creatures. *Proceedings of SIGGRAPH 94, ACM Computer Graphics*, pages 15–22, 1994.

[24] D. J. Sturman. Computer puppetry. *IEEE Computer Graphics and Applications*, 18(1):38–45, January/February 1998.

[25] J. Troy. *Dynamic Balance and Walking Control of Biped Mechanisms*. PhD thesis, Iowa State University, 1995.

[26] J. Troy. Real-time dynamic balancing and walking control of a 7-link planar biped. In *Proceedings of ASME Design Engineering Technical Conferences*, 1998.

[27] J. Troy and M. Vanderploeg. Interactive simulation and control of planar biped walking devices. In *Workshop on Simulation and Interaction in Virtual Environments*, pages 220–224, July 1995.

[28] M. Unuma, K. Anjyo, and R. Takeuchi. Fourier principles for emotion-based human figure animation. *Proceedings of SIGGRAPH 95, ACM Computer Graphics*, pages 91–96, 1995.

[29] M. van de Panne and E. Fiume. Sensor-actuator networks. *Proceedings of SIGGRAPH 93*, pages 335–342, 1993.

[30] M. van de Panne, R. Kim, and E. Fiume. Virtual wind-up toys for animation. *Proceedings of Graphics Interface '94*, pages 208–215, 1994.