EUROGRAPHICS 2002 / G. Drettakis and H.-P. Seidel
(Guest Editors)

Volume 21 (*2002*), *Number 3*

Local Physical Models for Interactive Character Animation

Sageev Oore[1] Demetri Terzopoulos[1,2] Geoffrey Hinton[1]

[1]Department of Computer Science, University of Toronto [2]Computer Science Department, New York University

Abstract

Our goal is to design and build a tool for the creation of expressive character animation. Virtual puppetry, also known as performance animation, is a technique in which the user interactively controls a character's motion. In this paper we introduce local physical models for performance animation and describe how they can augment an existing kinematic method to achieve very effective animation control. These models approximate specific physically-generated aspects of a character's motion. They automate certain behaviours, while still letting the user override such motion via a PD-controller if he so desires. Furthermore, they can be tuned to ignore certain undesirable effects, such as the risk of having a character fall over, by ignoring corresponding components of the force. Although local physical models are a quite simple approximation to real physical behaviour, we show that they are extremely useful for interactive character control, and contribute positively to the expressiveness of the character's motion. In this paper, we develop such models at the knees and ankles of an interactively-animated 3D anthropomorphic character, and demonstrate a resulting animation. This approach can be applied in a straightforward way to other joints.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism, Interaction Techniques

1. Introduction

In the presence of increasingly sophisticated animated entertainment and virtual simulations, efficient and high-quality interactive animation is becoming extremely important. The most common goal for such animation is primarily either *believability* (for entertainment applications) or *realism* (for simulations). Although they can be closely related, neither quality necessarily implies the other. For example, referring to the classic early computer animated film *Tony de Peltrie*[5], the great Disney animator Frank Thomas said that the lead character "did not amaze audiences because he was made up of so many intricate moves as much as the fact that he told a poignant story"[33].

Our primary goal is the creation of expressive, believable animation. A powerful way to achieve this is by virtual puppetry, the interactive creation of animation. From the user's perspective, virtual puppetry can be viewed analogously to a musical instrument, except instead of music, the user "plays" animation.

As believability is important to us, we are also very interested in the advantages that can be gained in this regard from having physically realistic elements in an animation. However, issues such as dynamic balancing make physics-based character animation very hard to control, even off-line. That, in conjunction with computational costs and interface issues, presently makes it simply unfeasible to simultaneously reconcile both *interactive* and *full physics-based* animation for expressive 3D character control.

A fully interactive system can, however, incorporate a partial, local physical simulation, as we will show. In this paper, we contribute a new approach to creating physics-based motion actuators. We demonstrate how an existing kinematically-based performance animation system can be enhanced with such actuators while still providing potential for real-time interaction. Furthermore, these actuators *simplify* the control process by having certain natural behaviours occur automatically, yet without constricting the user's ability to override them.

The current work represents an application of a physics-based approach to interactive character animation. We therefore begin by describing some related work in these two ar-

eas (Section 2), followed by an overview of our approach, and the basic kinematic control framework over the system (Sections 3, 4). Local physics-based models are developed and applied in Sections 5 and 6. Issues regarding these local controllers are discussed in Section 7, and results are presented in Section 8. Finally, future work and conclusions are presented in Sections 9 and 10.

2. Background

2.1. Real-time and Interactive animation

To control animation without feedback is– quite literally– choreographing and performing a dance in the dark. Computer puppetry transcends the limitations of many other animation techniques by allowing the animator to create motion in time and react to the created motion immediately, thus putting both the creation and the correction processes in the dimension where they occur. Performance animation thus allows a spontaneous and efficient creative process that can result in expressive animation. Some of the earliest work in this area was by the Jim Henson Company[37] and deGraf and Wharman[11]. Companies such as Medialab and Protozoa further developed this technology in the 90's for a wide range of characters and applications. Sturman[32] gives a comprehensive overview of this history, and deGraf and Yilmaz[10] provide additional examples of some of the wonderful CG creatures that were created at Protozoa.

The well-known difficulty with these approaches, however, arises when highly complex characters need to be controlled, such as articulated humans[23]. In such cases, a full motion capture studio is typically needed to record the performance of actors wearing sensors or markers[4, 25]. The resulting signals are then corrected for issues such as sensor noise[7], sensor occlusion[16] (in camera-based systems), and differences in proportion between the actors' bodies and those of the characters they are animating[14, 6, 26, 31]. Motion captured libraries generally require modification if they can at all be reused; various techniques have been proposed for editing and blending of motion parameter curves[40, 9, 22].

In contrast, our DIGITAL MARIONETTE system[27] requires only a bimanual input device based on a pair of motion trackers to allow multi-tracked performance animation of a 3D articulated character. In this paper, we describe how this system relies on a simple set of local, physics-based principles. First, however, we give some background on physics-based animation in general, and more specifically interactive physics-based animation.

2.2. Physics-based animation

Physically-simulated character animation has been a holy grail of computer graphics for many years, with some of the earliest work described by Wilhelms[39] and Armstrong, Green and Lake[1]. The advantages of full dynamic simulation lie in the unparalleled naturalness and realism of the motion. It is, however, extremely difficult to control.

One of the hardest aspects of dynamic character simulation is keeping the characters balanced. In the VIRYA system[38], a balance mode provided automatic external forces to counteract any motion for the trunk away from a desired orientation. However, this also made it impossible to walk properly since walking is based on falling forward[24]. van de Panne and Lamouret[35] describe a system in which external forces automatically help a character balance, and are then gradually removed as the character learns to balance himself. Laszlo et al[20] use a manually-tuned finite state machine over PD-controllers at the joints to control the dynamic simulation of a cyclic motion such as human walking. A predictive model increases the stability of the system. Hodgins et al[18] demonstrate controllers for running, diving and other complex activities, and Faloutsos et al[12] propose a framework for transitioning between multiple controllers. Hodgins and Pollard[19] present a method for adapting physical controllers to different characters, and Popović and Witkin[30] provide a paradigm and algorithm for transforming motions while preserving essential physical proporties. Dynamic and kinematic approaches have also been combined[13, 8, 2].

2.3. Interactive Physics-based Control

The work by Troy[34] is one of the earliest examples of interactive locomotion control for a dynamically simulated character. The user operates a 7-link planar biped within a virtual environment, with a feedback-based 2D balancing regulator, and a state-machine-PD controller for a walking motion. Laszlo, van de Panne and Fiume[21] have demonstrated a nice system for controlling interactive 2D-dynamic character simulations in various scenarios. For example, continuous mouse input sets the target states for PD controllers at the 2-joints of a Luxo character. In another example, a keyboard interface controls a planar bounding cat, with different keys triggering different equilibrium states for subsets of the parameters. Continuous and discrete controllers are combined to allow a planar biped to walk, run and do long jumps.

Most recently, van de Panne[36] has developed a computer game in which the player uses the mouse's x, y-translation to control the dynamic simulation of a 2-D skier on a variety of entertaining courses that include jumps and obstacles. The stiffnesses are set to reasonably high, constant values, and the length of the skis themselves provide the character with a relatively stable base. When the character falls and is outside the player's control, then the stiffnesses are automatically reduced to provide a more natural look. The user is taught the mapping by a series of follow-the-leader training examples.

The challenge of interactively controlled dynamic simulation of characters is extremely hard for many reasons. First, it is computationally difficult to achieve dynamic simulation of complex 3D systems in near real-time speeds. Planar characters, such as those used in the systems described above,

have both fewer degrees of freedom, and also involve significantly simpler collision checking facilitating real-time simulation. Second, finding trajectories *sufficiently close* to the desired ones, and which also subsume balancing throughout is another very difficult problem, both on-line and off-line. Third, unlike kinematic control, a dynamically simulated character can get into situations where it is not possible to detect an impending fall sufficiently early to be able to recover from it. Finally, non-static balancing, and dynamic behaviour in general, are sensitive to small variations in any of the parameters such as torque values or timing. Thus, not only do many parameters have to be controlled, but they have to be controlled accurately, particularly in the time domain (although slowdown factors can help in this regard[21]).

3. Approach

The systems described in the previous section are impressive in their use of full physical simulation. To achieve this, they need to keep the number of degrees of the freedom (DOF) of the system relatively small. Our criteria are somewhat different, as our primary goal is to maintain usability for expressive, believable animation. Thus, we require a sophisticated 3D articulated character, but we do not necessarily require a full physical simulation. With these guidelines in mind, we develop a simple physical approximation that takes into account certain behaviours of "interest", and allows for varying stiffness levels and viscosity, while ignoring some of the more global and complex or unpredictable dynamic interactions. For animation purposes, it turns out that this is not only sufficient in many cases, but in fact helps the user's control. Furthermore, it provides a direction for incremental modifications to the system with the long-term goal of controlling a full dynamic simulation. We now give an overview of the multi-tracking kinematic control framework.

4. System Overview

The DIGITAL MARIONETTE animation system is based on an interactive loop as shown in Figure 1. The user manipulates a bimanual input device; the input signal is filtered and mapped onto motion parameters of an articulated skeleton; responding to the immediate feedback, the user continues to control the animated character, as described below.

4.1. The Animated Character Output

The CG puppet we are controlling is an anthropomorphic, articulated body consisting of rigid links connected by joints, illustrated in Figure 3. Specifically, each hip is a ball-and-socket joint with three DOF; each knee is a hinge joint with one DOF; and the ankles are treated as hinge joints with one DOF each. The character's root is at the pelvis, and has three translation and three rotational DOF. The spine and the arms are modeled with another 17 DOF.

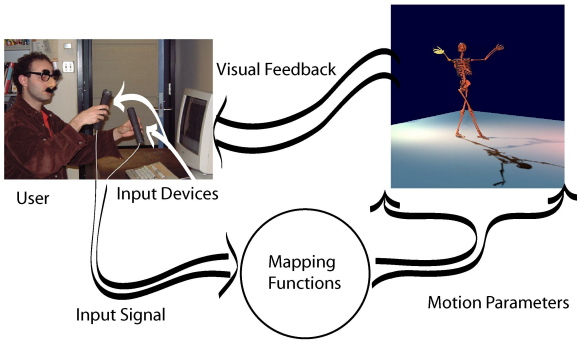

Figure 1: *Digital Marionette Interaction Loop: The user manipulates real-time input devices to interactively drive the motion of a computer graphics character, and is provided with immediate feedback displaying the animation as it is being created. The Digital Marionette system operates within a desktop environment.*

4.2. Input Device

Two Polhemus motion trackers[29] are embedded in cylindrical bamboo tubes, as shown in the upper left part of Figure 2. The customized wooden housing, although simple, provides the user visual and kinesthetic feedback cues[17] by establishing a tangible interface to inherent coordinate frames, which can then be matched to those of the character's bones, for example (also shown in Figure 2).

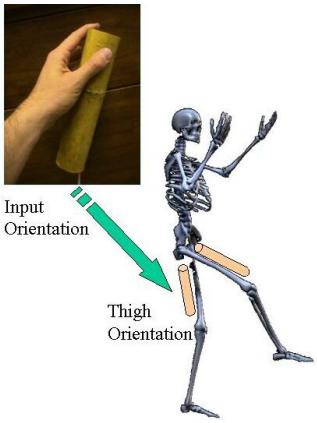

Figure 2: *Input Device to Output: The upper left photo shows the user holding one of the two input tubes. Although not visible, the tracker is in the top of the tube, near the user's fingers. On the right, cylinder icons next to the thigh bones schematically illustrate the mapping from input to hip rotation, e.g. keeping the thighs parallel to the tubes themselves. The long axis of the device corresponds in this case to the long axis of the bone.*

It is important to note that the feedback showing the motion of the character provides the basis for an input device

that is hand-held rather than worn. That is, as the user manipulates the input, he sees the resulting animation almost immediately and can therefore adapt and make corrections on-the-fly as necessary. The hand-held device is hence operated within a desktop environment, requiring a space of less than $1m^3$. The only required calibration is to define the approximate position of the trackers relative to the location of the source emitter in the user's environment, a simple procedure which only needs to be repeated if the position of the source relative to the user is changed significantly.

4.3. Filtering & Mapping

The raw input from the trackers is subject to both the jitter of the physical device in the user's hand, and more importantly, noise caused by the presence of electromagnetic devices in the environment. To filter it, let x_i represent the value received for one of the input parameters at time step t_i. Then we take the filtered value υ_i to be given by

$$\upsilon_i = \upsilon_{i-1} + (1-\alpha)(\upsilon_{i-1} - \upsilon_{i-2}) + \alpha(x_i - x_{i-1}) \qquad (1)$$

where α is effectively a viscosity parameter, set by the user to be in the range $0 < \alpha \leq 1$.

The filtered signal is then mapped to a subset of the character's parameters within a multi-tracking framework. That is, a full body motion is recorded in multiple passes, where each pass controls a "layer" or subset of the characters parameters. Typical layers of motion parameters for a character animation are shown in Figure 3. The leg and arm layers are controlled with a bimanual symmetric mapping (i.e. the left input device controls the left leg, and the right input device controls the right leg), and the spine and the head are controlled with an asymettric bimanual mapping in accordance with Guiard's Kinematic Chain theory[15] as applied within a computer graphics context[3].

The mappings are kinematic, in the sense that a filtered input signal $\vec{\upsilon}$ will map to a unique vector of joint-angle values $\vec{\theta}$ for any given layer of motion parameters. When controlling the legs, for example, the orientation of the input device is used to directly control the thigh bone orientation, as schematically illustrated in Figure 2, while the translations may be mapped to parameters such as the knee and ankle bends. For further details of the mappings, we refer the reader to Oore's thesis[27].

4.3.1. Locomotion and Ground Contact

To maintain a believable relationship of the virtual puppet to the ground, we define a set of potential contact points on the skeleton's feet, and impose the constraint that one of these points must always be touching the ground. Thus, as the character rotates at the hip, a point on one foot is constrained to stay at a fixed position relative to the floor, acting as the center of rotation. A mechanism is provided for switching feet as the next foot touches the ground. The ground model is implemented by determining, at each time step, which one of the potential contact points is touching the ground, and then adding a translational vector to the character's root position to keep that point fixed relative to the ground. By virtue of this ground contact, the puppet can locomote in any direction. Our method can accommodate an arbitrary number of potential contact points, and multiple ground levels of an uneven terrain. We currently use two contact points per foot—one at each of the heels and balls of the feet.

We now describe how and why the above approach can and needs to be enhanced by local physical models.

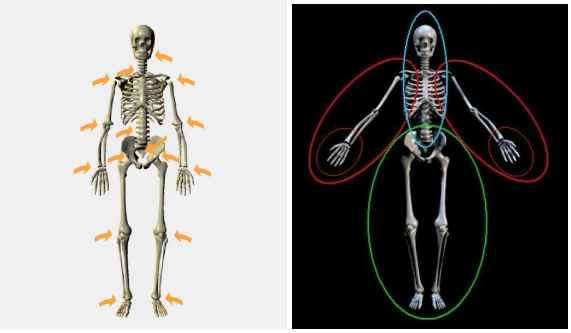

Figure 3: *Kinematic Layering Strategy: The arrows on the left figure indicate the articulation of the model. On the right, the circled regions mark groups of parameters that are recorded together in a single layer. Legs are usually recorded first, since they define the motion of the character's root. Spine and head are usually recorded simultaneously, followed by arms.*

5. Local Physical Models

5.1. Desired Properties

There are various ways in which physics can make an animation more natural. In particular, early use of the Digital Marionette system showed that a simple, isolated motion such as having the character raise his thigh and swing his corresponding lower leg naturally from the knee (a Tai Chi warm-up exercise) turned out to be quite hard to simulate by a direct kinematic control over the angle values. Natural swinging has a very smooth and recognizable displacement pattern, so that any unevenness is noticeable. Yet, many non-intentional motions involve some gravitationally-induced swinging (e.g. of a limb segment) about an articulation point. Furthermore, when stepping down, adjusting the ankle so that the foot lands on the heel, then comes down flat on the ground, and finally pushes off was quite difficult to achieve at every step as well, especially while trying to control other aspects of the character's motion.

The reason, of course, that we do not have to work to swing our leg is because we can relax and let gravity and momentum do it for us. The reason our foot comes down, stays

down, then pushes off, is also a result of physical principles. We would like such motions to occur equally naturally when animating, too. To give our model these benefits, one possibility is using a dynamic simulation. However, doing so involves a host of other control difficulties, as described earlier. Our goal, on the contrary, is to facilitate the puppeteer's control over subtleties in the motion, for example by making swinging occur naturally to simplify his task. The puppeteer should have the freedom to focus his cognitive resources on other, subtle aspects of the motion. In this light, using a full dynamic simulation would defeat our purpose, for instead of simplifying, it would add an extra layer of complexity (i.e. balance) to all motion. To solve this problem we introduce local dynamic models that approximate *selected, desired* physically-generated behaviours.

We thus begin by enumerating the specific benefits of physical simulation we wish to gain, and the advantages of kinematic control that we wish to retain. The desired motion and control properties for the virtual puppet's knee, for instance, can be summarized as follows:

a1 When a leg is lifted, then the corresponding shin should be able to swing naturally. During walking, this could lead to behaviour such as the foot occasionally extending beyond its "goal" orientation just before stepping down.
a2 When standing, the supporting knee should have some springiness to it, unless it's fully extended (locked).
a3 Regardless of how the above properties are implemented, the puppeteer should still be able to manipulate the knee into any desired position within the natural range of motion, and maintain it there for as long as he wants to.

Likewise, a similar set of desired properties can also be listed for the ankle:

b1 When the swinging leg strikes the ground— either with the heel or with the toes— and becomes the supporting leg, then the natural tendency should be for the rest of the foot to come down flat on the ground.
b2 The ankle of the supporting leg should have a springy quality when the weight is on the ball of the toes. (Note that we currently model the foot with a contact point at the heel and contact point at the toes; we do not model the rotation of the tarsal phalanges about the metatarsophalangeal joint).
b3 It should be easy to keep the foot of the supporting leg parallel to the ground, but also easy to bend the ankle to push off at the end of the support phase.
b4 Regardless of how the above properties are modeled, the puppeteer should still be able to manipulate the ankle into any desired position within the allowable range and maintain it there for as long as he wants to. This is to ensure that any "automatic" behavior added to the system does not reduce the extent of the puppeteer's potential control (including for motions which happen to be unrelated to walking). For example, it should not be too hard to make sure that the toes of the swinging leg clear the ground, or to walk on tiptoes as well.
b5 Finally, and very importantly, although gravity needs to be incorporated (e.g. in helping the foot to foot fall flat onto the ground) the puppeteer should not have to do any extra maneuvering in order to keep the puppet from falling over.

5.2. Augmenting a Kinematic Model With Local Physically-Based Properties

To see how these properties can be implemented by a local physical model, consider, as a typical example, a simple two-link structure with a hinge joint as shown in Figure 4, where link 1 is fixed rigidly to the ceiling at one end, and the

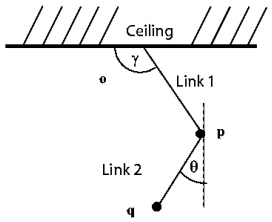

Figure 4: *Two Link Chain: Link 1 is attached to the ceiling at an angle γ at point* **o**, *and Link 2 is a attached to Link 1 at an angle θ by a hinge joint. A point mass m is assumed to be at* **q**.

orientation θ of link 2 is controlled directly according to a mapping†

$$\theta = \psi(\upsilon) \tag{2}$$

of input υ. Now suppose that a simple physically-inspired property is introduced so that as link 2 rotates about point **p**, the angular velocity $\dot{\theta}$ is maintained, so that $\dot{\theta}$ at time t_{i+1} is given by

$$\dot{\theta}(t_{i+1}) = (1-\lambda)\dot{\theta}(t_i) \tag{3}$$

where $0 \leq \lambda \leq 1$ is a damping coefficient. The state of the joint can then be updated by

$$\theta(t_{i+1}) = \theta(t_i) + \dot{\theta}(t_i)\Delta t \tag{4}$$

where $\Delta t = t_{i+1} - t_i$. This has effectively introduced behaviors of angular momentum and damping into the system, but it has also removed the user from the control loop (the motion is now parameterized only by the initial state $\theta(t_0)$ and λ). When physical properties are introduced, then a pure kinematic control mechanism is no longer applicable, and some new control must be provided within the context of physical simulation. Of course, the obvious framework within which to reincorporate interactive user control is by

† Note that the model assumes a function exists for specifying desired target positions for the character's joints. In our case this is provided by the multi-tracking kinematic mappings outlined earlier.

having the user cause accelerations by applying forces (or torques), as opposed to directly specifying (instantaneous) angular displacements. Instead of directly computing positions as in Eq (2), or velocity as in Eq (3), we use an underlying torque model, with a standard Euler-step dynamic simulation update:

$$\dot{\theta}(t_{i+1}) = \dot{\theta}(t_i) + \ddot{\theta}(t_i)\Delta t \quad (5)$$

$$\theta(t_{i+1}) = \theta(t_i) + \dot{\theta}(t_i)\Delta t \quad (6)$$

to give the state of the joint $\theta(t_{i+1})$ at time t_{i+1}, where $\ddot{\theta}$ is proportional to the sum of active torques.

Now a natural way to get a torque from a kinematic mapping ψ is by imagining an angular spring with stiffness k at joint $\mathbf{p}$ that has $\psi(\upsilon)$ as its equilibrium value:

$$\tau_{control} = k(\psi(\upsilon) - \theta) \quad (7)$$

The damping can be caused by a torque

$$\tau_{damping} = -\lambda\dot{\theta} \quad (8)$$

This, combined with $\tau_{control}$ is essentially a proportional derivative (PD) controller (setting the stiffness k will be described in Section 6.3).

To enable the natural swinging motion suggested previously, gravity needs to be added to our model. This is done by adding the following torque (computed as in Figure 5):

$$\tau_{gravity} = |g|m\sin\theta \cdot r \quad (9)$$

where r is the length of link 2, and a point mass m is assumed to be at $\mathbf{q}$.

Combining Eq (7), (8), and (9), we have the total torque simulated at joint $\mathbf{p}$ to be:

$$\begin{aligned}\tau_{total} &= \tau_{gravity} + \tau_{control} + \tau_{damping} \quad (10)\\ &= |g|mr\sin\theta(t_i) + k[\psi(h) - \theta(t_i)] - \lambda\dot{\theta}(t_i)\end{aligned}$$

A difference between the model described here and typical PD control is that we are not really in a physical simulation: returning to Figure 4, suppose γ is no longer fixed, but controlled kinematically by a mapping ψ_2, so $\gamma = \psi_2(\upsilon)$. We can still use Eqs (10,5,6) to control joint $\mathbf{p}$ by force, while using a kinematic simulation to control joint $\mathbf{o}$ directly. Of course, the motion of link 2 will no longer be physically realistic, because rotating link 1 actually accelerates link 2, however for our purposes this is not critical. Ultimately, our goal is to map from real-time input onto a real-time graphical animation: if the animation control works better with some physically-inspired properties than with none— as we have found to be the case in our own experience— then we make use of this. If we were to add all of the complex physical interactions that can occur between all the joints of a human articulated figure in 3 dimensions, then the simulation— just the act of balancing let alone locomotion— would become so hard to control in real-time that the benefit would be lost. So we just add what we can use.

6. Applying the Physics-Based Motion

6.1. Model for Knee Motion

A model for controlling the knee can be separated into two distinct cases, based on whether the corresponding foot is in the air or on the ground. When the foot is in the air, as in Figure 5, then the general form of the control is as given in Eq (10), where r now represents the length of the shin.

For the knee of the supporting leg, gravity is ignored, leaving a PD controller. This is just enough to provide the puppet with nice "soft" knees which are not too stiff.

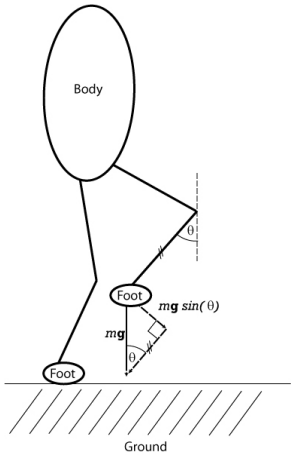

Figure 5: *Gravitational Torque at the Knee: The configuration at the knee here is essentially the same for our purposes as that shown in Fig 4. The force due to gravity is resolved into 2 components— one parallel and one perpendicular to the shin. The latter component creates a torque of magnitude proportional to $|g|m\sin\theta \cdot r$ where r is the distance from the knee to the ankle and m is the mass of the foot.*

6.2. Ankle Motion

An effective model for the ankle motion control is harder to achieve, because of the potential complexity introduced by ground contact, and because there are more "motion properties" we wish to obtain, and they are more subtle.

Motivated by the requirements listed in Section 5.1, we use a dynamically-based model, as shown in Figure 6, wherein the heel contact with the ground is treated as a hinge joint. The downward action of the foot which happens at heel strike results as follows (referring to Figure 6):

1. The force due to gravity, $\mathbf{g}$, is resolved into two components:
   (a) $\mathbf{g}_0$, which is perpendicular to the direction of the lower leg, and
   (b) $\mathbf{g}_1$, which is parallel to the leg, and has magnitude

$$|\mathbf{g}_1| = |\mathbf{g}|\cos(\alpha)$$

2. That second component, $\mathbf{g}_1$, is redrawn in Figure 6 at the ankle as $\mathbf{f}$, which is then resolved into two further components:
   (a) $\mathbf{f}_0$, which is parallel to the foot, and
   (b) $\mathbf{f}_1$, which is perpendicular to the foot, and has magnitude

$$|\mathbf{f}_1| = |\mathbf{f}|\cos(\beta - \pi/2)$$

This leaves us with the simple configuration shown in Figure 7. So the torque exerted at the ankle due to gravity is

$$\begin{aligned} \tau_g &= |\mathbf{f}_1| \cdot r \quad (11) \\ &= |\mathbf{f}|\cos(\beta - \pi/2) \cdot r \quad (12) \\ &= |\mathbf{g}|\cos(\alpha)\cos(\beta - \pi/2) \cdot r \end{aligned}$$

where r is the distance from the heel to the ankle. The total torque at the ankle is thus given by $\tau_g + \tau_{control} + \tau_{damping}$ where the latter two terms are as given in Eqs (7) and (8).

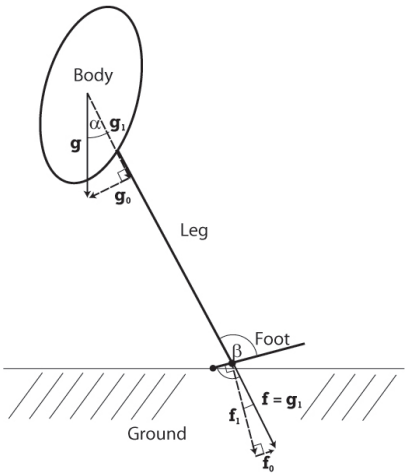

Figure 6: RESOLVING FORCES AT THE ANKLE
The leg represents the direction of the shin, below the knee. The only source of force considered in this diagram is gravity, applied on the mass of the body and acting on the centre of mass. The heel is considered a fixed hinge joint with the ground. A similar diagram can be drawn for the case when the toe is attached to the ground.

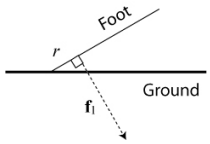

Figure 7: *Net Gravitational Force at the Ankle: r is the distance from the fulcrum (where the heel touches the ground) to the ankle (the point of application of the force).*

This model assists in getting the foot flat on the ground since gravity will push the mass of the skeleton down to apply a torque on the supporting foot as required. Furthermore, the perpendicular component g_0 of g (in Figure 6) is intentionally ignored, so that the character does not risk falling over.

The user is provided with a PD-controller, once again, to manipulate the joint angle at the ankle by specifying a target equilibrium joint angle value.

6.3. Stiffness Control

There is an important issue that still needs to be resolved: We want to facilitate natural swinging motion, yet we also want the user to be able to control high-acceleration motions. Natural swinging will occur when the dominant term in Eq (10) is the torque due to gravity, whereas for the user to control high-acceleration motions, the dominant term needs to be $\tau_{control}$. In order for $\tau_{control}$ to be large, looking at Eq (7) shows that either $|\psi(h) - \theta|$ must be large, or k must be large. But $|\psi(h) - \theta|$ is only really large if the user is vastly overshooting the target values, both when initiating motions as well as when stopping them, which is very difficult to do accurately. Suppose, then, that k is set to be very large. Then the joint motion will closely follow the target value trajectory $\psi(\upsilon)$ as controlled by the user. In this case, if the user simply wants the leg to swing freely, then he or she must draw just the right control path to do this. That is, the puppeteer uses an extremely high stiffness and then *pretends* to swing the leg. In fact, in the limiting case as $k \to \infty$, the model tends towards becoming kinematic control, but with additional gravitational effects[‡]. But this is exactly what we are trying to overcome in the first place!

The above argument shows that k must necessarily change over time somehow. The challenge is that all of our input degrees of freedom are already being allocated to the kinematic joint specification. We currently cannot have the puppeteer control both the stiffnesses and the target values independently and simultaneously. We have solved this issue for the knee and ankle control as follows.

6.3.1. Controlling Stiffness at the Knee

The solution currently adopted for the knee is to have the stiffness k vary depending on the angular distance of the target orientation from the world vertical axis. That is, the user input υ determines a knee angle $\psi(\upsilon)$ relative to the thigh, which, depending on the rest of the puppet's configuration, specifies an orientation $\psi_{WCS}(\upsilon)$ in the world coordinate system (WCS) for the shin. It is this target orientation, relative to the vertical axis in WCS, that determines the stiffness, which we can now write $k(\psi_{WCS}(\upsilon))$. It should be emphasized that the stiffness is a function of the user-specified target orientation, not the current orientation. Although this is still a reduction of a degree of freedom in the

[‡] In this case, the kinematic process would be faster and more numerically stable, however.

user-controllable parameters, from a user's point of view is more attractive than any of the fixed-stiffness solutions.

When $\psi_{WCS}(\upsilon)$ is near 0, i.e. nearly vertical, then the stiffness is very small, allowing the knee to swing almost freely. When we reduce the stiffness, we also make a corresponding reduction in the damping parameter λ. As $\psi_{WCS}(\upsilon)$ approaches a diagonal, the stiffness plateaus to a higher value. The rationale behind this is that when the shin is intended to point downwards, we let gravity do the work of keeping it there, and therefore we can relax the corresponding muscles, allowing it to swing on its own momentum. On the other hand, if it is oriented at a larger angle from the vertical, then either it is at the edge of a swing phase (which the puppeteer could achieve by making $\psi_{WCS}(\upsilon) = 0$, and consequently k is small), or else we intend it to be there, so $\psi_{WCS}(\upsilon) \approx \theta$ with $k \gg 0$. Originally $k(\psi_{WCS}(\upsilon))$ was a piecewise linear function, but this created unnatural accelerations as the leg approached horizontal target values, so a tight gaussian-shaped function is currently being used, with k approaching 0 only in the region where $\psi_{WCS}(\upsilon)$ is near 0.

For the supporting leg, the solution is simpler, in that the stiffness can remain constant.

6.3.2. Controlling Stiffness of the Ankle.

Regarding the ankle motion, the question is how to allow the supporting one to be loose as the rest of the body rotates around it, while allowing for much larger stiffness during the plantar flexion occurring as the supporting leg pushes off the ground. This is different from the knee, in that at one moment the ankle controller needs to be completely relaxed, while at the next moment, a large torque needs to be exerted.

In this case, the current ankle pitch ζ (in WCS) (that is, its angular difference from the global *xz*-plane) is tested for whether it is within some very small tolerance of being horizontal. If that is the case, then no gravitational torque is applied, so that the foot does not rapidly oscillate between the heel and toe contact points. Furthermore, if the pitch of the supporting ankle is already horizontal, to within $\pi/20$ radians, and if the difference $|\psi_\zeta - \zeta|$ between the pitch of the target orientation and the current pitch is less than $\pi/10$ radians, then a pure kinematic simulation is used to keep the ankle perfectly level[§]. Otherwise we revert to a constant, fairly high stiffness. The ankle target angle is directly controlled relative to the ground.

Thus the user only needs to specify the ankle to remain *approximately* horizontal (once it already nearly is) to ensure that it will stay completely horizontal. It also means that the puppeteer can still override this behaviour by overshooting the target value; if he or she sets a target value which is more than $\pi/10$ radians from the current position, then the usual torque simulation will kick in.

[§] Originally another physical simulation with different parameters was used at this point to have the same effect, but it required a much smaller stepsize for numerical stability, so the momentary kinematic solution was more convenient.

7. Discussion: Equilibrium-Point Control

PD-control is a specific instance of equilibrium-point control. As such, it has the nice characteristic that it can convert a kinematic or posture-based control into a dynamic control mechanism. For any gesture made by the user specifying a desired posture of the character, the difference between the character's current posture and the desired one can be used to calculate the resulting torques at each of the joints. The spring model given by Eq (7) could be replaced by a wide variety of functions[1, 28] while preserving this characteristic.

Regardless of the torque function, another property of equilibrium-point control is that the mapping from input to output signal is non-stationary, in the sense that an essential quality of inertia and gravity is that the character will sometimes move without input from the user, while at other times the user's motion will cause the character to stay still. This is not necessarily unintuitive, but it does have both advantages and disadvantages. The advantages, as described earlier, are that many desirable effects occur naturally. The disadvantage is the potential cognitive complexity of controlling this behaviour in certain situations.

8. Results

The expressiveness of the animation created with our system has been demonstrated in a wide variety of contexts, ranging from live theatre to television. A professional puppeteer was also brought in to watch an experienced user working with our system, and one of her first questions was whether the ankle motion was happening automatically or not at every step. When told that this was indeed the case (due to the physical models), her next question was, "But can you make the puppet stand on his toes?", She greatly appreciated that this, too, was possible, as it demonstrated that the user's control had not been sacrificed to achieve this automated motion. The puppeteer also appreciated the slight bump that occurs as the character steps down onto the ground.

Figure 8 shows sample frames from a one-minute long demonstration animation in which the character dances to some slow music. All of the animated parameters of this sequence were created in a total of under 10 minutes using our system, with no re-takes necessary. This is extremely efficient compared to traditional animation techniques, and the local physics-based models for the legs were critical in making such effective control possible.

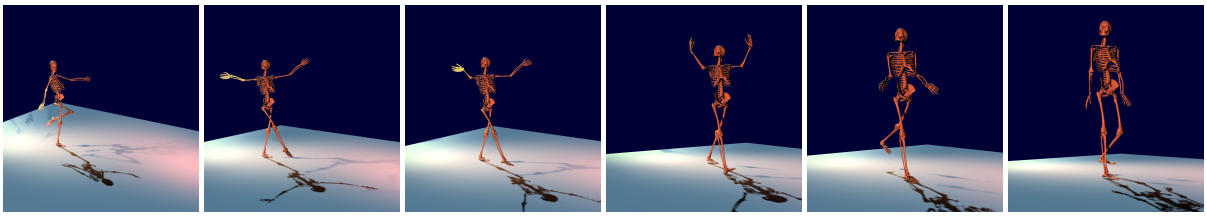

Figure 8: *Digital Marionette Dance... Still frames.*
These images are frames taken from an interactively-generated dance animation sequence created using the Digital Marionette interface with local physically-based motion actuators. All animated parameters of the one-minute long sequence were generated in a total of 10 minutes at a desktop environment. The frames shown here and in the following sequence were sampled about one to two seconds apart (going across rows), highlighting some of the postures achieved by the character.

9. Future Work

9.1. Extending the Local Physical Models

A natural extension will be adding a physical model to the hips and arms. When adding the hip model, some care may be needed once again to avoid having balancing issues, while still allowing interaction between the hip and knee motion.

One limitation of the system is in the kinematic ground contact model. Since exactly one point is always fixed to the ground, double support is difficult to control accurately, while zero-support is not currently possible. We will explore additional physics-based models to overcome these issues, and facilitate motions such as running and jumping.

As mentioned previously, the local physical models do not exert forces that make the character fall down; although advantageous in most circumstances (i.e. the character is generally intended to stay upright), sometimes such downwards forces would in fact be desirable. For example, if the character were to trip over an object, or simply lean too far backward or forward, it would be helpful if the physical model could simulate the appropriate falling, rather than leaving it up to the user to generate the realistic details of such motions. Indeed, precisely therein lies the strength of a full physical simulation, so it would be desirable to integrate the interactively-controlled local models with full dynamic controllers, e.g. within a framework such as that proposed by Faloutsos et al[12].

9.2. Extending the Interface and Control

We will explore the possibility of providing the user with a mechanism for varying stiffness/relaxation independently during interactive physical control. There are a number of ways to potentially incorporate this additional parameter into the control interface. A feasible and natural solution would be through the use of grip-strength controllers, having the obvious advantage that the user's "tension" gets mapped onto tension of the character, although specifying the joint on which to apply this could be tricky. Interestingly, perhaps even two discrete stiffness control values, corresponding to high and low (zero) stiffness, would still give significant advantage over a single constant value. Although a grip-strength control may be able to sense both continuous and discrete signals, the latter could even be accomplished with simple buttons that can be held and released.

In general, an important direction of future work is to continually increase the fidelity of the dynamic simulation while maintaining usability. As discussed earlier, a full dynamic simulation is currently simply unfeasible. However, the powerful advantage of local physical models are that they allow the possibility of identifying and solving the difficulties of dynamic control issue incrementally, making intermediate stages both functional and manageable for animation. It is this kind of approach which will further the field of interactive physically based animation. As computational resources also increase, this may ultimately lead to interactive control over full dynamic simulations, with additional applications in robotics, biomechanics, and computer-assisted medicine.

10. Conclusion

Our primary goal was to provide an efficient, powerful and satisfying interface for expressive character animation. To achieve this, we introduced local physical models for performance animation and described how they can augment an existing kinematic method. These models approximate specific physically-generated aspects of a character's motion by automating certain behaviours, while still letting the user override such motion via a PD-controller if he so desires. Furthermore, they can be tuned to ignore certain undesirable effects, such as the risk of having a character fall over, by ignoring corresponding components of the force. Although local physical models are a quite simple approximation to real physical behaviour, they are extremely helpful both in improving the quality of the resulting motion, as well as in creating a significantly more usable interface. We develop such models at the knees and ankles of an interactively-animated 3D humanoid character, and show a resulting animation. This approach can be applied in a straightforward way to other joints.

11. Acknowledgements

We thank Chakra Chennubhotla for invaluable discussions and feedback. We thank Joe Laszlo and Michael Neff for their help in creating the video. We thank Petros Faloutsos and Victor Ng for providing the DANCE platform and support. We thank the referees for their helpful comments and suggestions.

References

[1] William Armstrong, Mark Green, and R. Lake. Near-real-time control of human figure models. *IEEE Computer Graphics and Applications*, 7(6):52–61, June 1987.

[2] Norman I. Badler, Cary B. Phillips, and Bonnie Lynn Webber. *Simulating humans : Computer Graphics Animation and Control*. Oxford University Press, New York, 1993.

[3] Ravin Balakrishnan. *Issues in Bimanual Interaction for Computer Graphics*. PhD thesis, University of Toronto, 2001.

[4] V. Benquey and L. Juppé. The use of real-time performance animation in the production process. In *ACM SIGGRAPH 97 : Course Notes*, volume 1, Los Angeles, 1997. ACM Press.

[5] P. Bergeron and P. Lachapelle. Controlling facial expressions and body movements in the computer-generated animated short Tony De Peltrie. In *Siggraph 85 Advanced Computer Animation Seminar Notes*, NY, July 1985. ACM Press.

[6] R. Bindiganavale and N. I. Badler. Motion abstraction and mapping with spatial constraints. In *CAPTECH '98: Workshop on Modelling and Motion Capture Techniques for Virtual Environments*, pages 70–82, Geneva, November 1998.

[7] B. Bodenheimer, C. Rose, S. Rosenthal, and J. Pella. The process of motion capture: Dealing with the data. In *Computer Animation and Simulation '97 : Proceedings of the Eurographics Workshop in Budapest, Hungary*, pages 3–18, 1997.

[8] A. Bruderlin and T. W. Calvert. Goal-directed, dynamic animation of human walking. *Computer Graphics*, 23(3):233–242, 1989.

[9] A. Bruderlin and L. Williams. Motion signal processing. In *Computer Graphics*, pages 97–104, 1995. Annual Conference Series, ACM.

[10] Brad de Graf and Emre Yilmaz. Puppetology: Science or cult? *Animation World*, 3(11), February 1999.

[11] B. deGraf. Notes on human facial animation. In *SIGGRAPH 89 : Course Notes*, volume 22, pages 10–11, Boston, 1989.

[12] P. Faloutsos, M. van de Panne, and D. Terzopoulos. Composable controllers for physics-based character animation. In *Proceedings of ACM SIGGRAPH 2001*, August 2001.

[13] M. Girard and A.A. Maciejewski. Computational modeling for the computer animation of legged figures. In *Proceedings of SIGGRAPH 85*, volume 20, pages 263–270, 1985.

[14] M. Gleicher. Retargetting motion to new characters. In *Proceedings of SIGGRAPH 98*, pages 33–42. ACM Press, 1998.

[15] Yves Guiard. Asymmetric division of labor in human skilled bimanual action: The kinematic chain as a model. *Journal of Motor Behaviour*, 19(4):486–517, 1987.

[16] L. Herda, P. Fua, R. Plänkers, R. Boulic, and D. Thalmann. Local and Global Skeleton Fitting Techniques for Optical Motion Capture. In *Computer Animation*, Philadelphia, May 2000.

[17] Ken Hinckley. *Haptic Issues for Virtual Manipulation*. PhD thesis, University of Virginia, 1996.

[18] J. K. Hodgins, W. L. Wooten, D. C. Brogan, and J. F. O'Brien. Animating human athletics. In *Proceedings of SIGGRAPH 95*, pages 71–78, Los Angeles, CA, 1995.

[19] Jessica K. Hodgins and Nancy S. Pollard. Adapting simulated behaviours for new characters. In *Proceedings of SIGGRAPH 97*, Los Angeles, CA, 1997.

[20] J. F. Laszlo, M. van de Panne, and E. Fiume. Limit cycle control and its applications to the animation of balancing and walking. In *Proceedings of SIGGRAPH 96*, pages 155–162, New Orleans, August 1996.

[21] Joseph F. Laszlo, M. van de Panne, and E. Fiume. Interactive control for physically-based animation. In *Proceedings of SIGGRAPH 2000*. ACM SIGGRAPH, 2000.

[22] Jehee Lee and Sung Yong Shin. A hierarchical approach to interactive motion editing for human-like figures. In *Proceedings of SIGGRAPH 99*, pages 39–48, August 1999.

[23] R. Maiocchi. 3-D Character Animation Using Motion Capture. In N. Magnenat Thalmann and D. Thalmann, editors, *Interactive Computer Animation*. Prentice Hall Europe, London, 1996.

[24] T.A. McMahon. *International Journal of Robotics Research*, 3(2):4–28, 1984.

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

[26] J. O'Brien, R. Bodenheimer, G. Brostow, and J. Hodgins. Automatic joint parameter estimation from magnetic motion capture data. In *Graphics Interface*, pages 53–60, May 2000.

[27] Sageev Oore. *Digital Marionette: Augmenting Kinematics with Physics for Multi-Track Desktop Performance Animation*. PhD thesis, University of Toronto, Toronto, Canada, 2002.

[28] S. Plagenhoef. *Patterns of Human Motion: A Cinematographic Analysis*. Prentice-Hall, Englewood Cliffs, NJ, 1971.

[29] Polhemus. `www.polhemus.com`.

[30] Jovan Popović and Andrew Witkin. Physically based motion transformation. ACM Press / ACM Siggraph / Addison Wesley Longman, 1999.

[31] M.-C. Silaghi, R. Pläankers, R. Boulic, P. Fua, and D. Thalmann. Local and Global Skeleton Fitting Techniques for Optical Motion Capture. In Nadia Magnenat-Thalmann and Daniel Thalmann, editors, *CAPTECH '98: Workshop on Modelling and Motion Capture Techniques for Virtual Environments*, Geneva, Switzerland, November 1998.

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

[33] Frank Thomas. The future of character animation by computer. In *SIGGRAPH 87 : Course Notes*, volume 5, 1987.

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

[35] M. van de Panne and A. Lamouret. Guided optimization for balanced locomotion. In *Computer Animation and Simulation '95 : Proceedings of the Eurographics Workshop (Netherlands)*, pages 165–177, Wien, Austria, 1995. Springer-Verlag.

[36] Michiel van de Panne. www.motionplayground.com, 2001.

[37] Graham Walters. The story of waldo c. graphic. In *SIGGRAPH 89 : Course Notes*, volume 4, pages 65–79, 1989.

[38] Jane Wilhelms. Virya - a motion control editor for kinematic and dynamic animation. In *Graphics Interface '86*, May 1986.

[39] Jane Wilhelms. Using dynamic analysis for realistic animation of articulated bodies. *IEEE Computer Graphics and Applications*, 7(6):12–27, 1987.

[40] Andrew Witkin and Zoran Popović. Motion warping. In *Proceedings of SIGGRAPH 95*, pages 105–108. ACM, Inc., 1995.