# Composable Controllers for a Virtual Stuntman

Petros Faloutsos Michiel van de Panne Demetri Terzopoulos
University of Toronto, Department of Computer Science
Techical report: CSRG-408

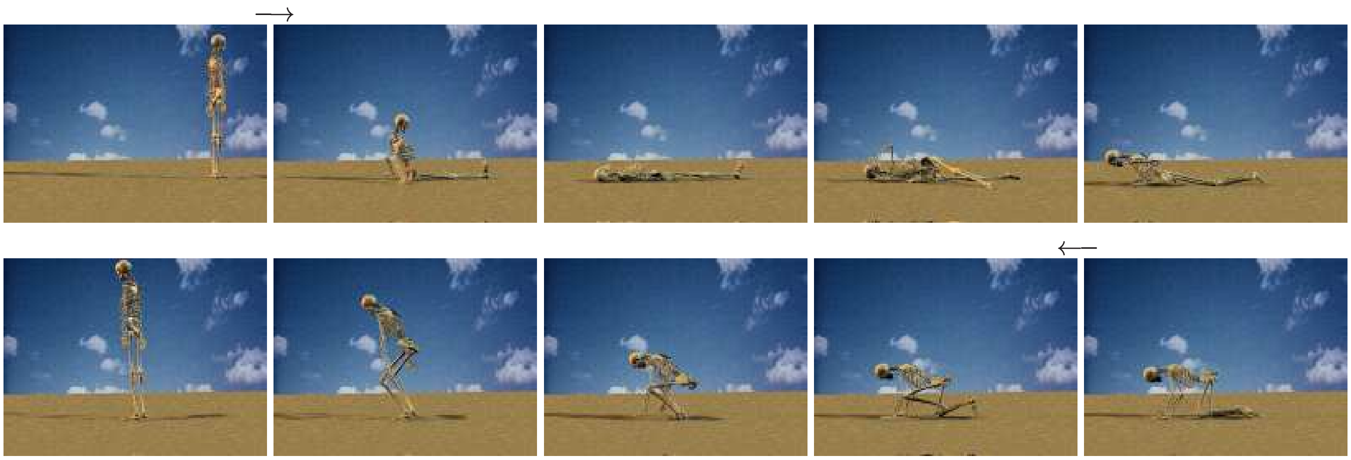


Figure 1: A dynamic "virtual stuntman" falls to the ground, rolls over, and rises to an erect position, balancing in gravity.

## Abstract

The implementation of a multi-talented, dynamic virtual actor is a significant challenge given the difficulty of designing controllers even for very specific motions. Divide-and-conquer methods are a logical approach to this problem, but this requires a community of control practitioners as well as a means of integrating the fruits of their labor. We propose a framework for composable controllers to address the latter requirement, which in turn is also an important prerequisite for the former. The composition framework is tested on two novel families of controllers. The first is aimed at the basic motions which need to be at the core of any skill repertoire, such as balance and fall recovery. The second explores a practical application of a synthetic actor, namely the execution of dangerous stunts by a dynamic digital stuntman.

**Keywords:** Computer Animation, Character Animation, Physics-Based Animation Control, Physics-Based Modeling

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

## 1 Introduction

Despite the considerable history of progress in animating virtual humans [3, 5], physics-based animated characters with a large repertoire of motor skills have so far been elusive. This may seem surprising in view of the recent successes in implementing a slew of specialist controllers capable of realistically synthesizing the complex dynamics of walking, of running, of diving, and of various gymnastic maneuvers, impressively demonstrated in the work of Hodgins and her co-workers [10].

While a divide-and-conquer strategy is clearly prudent in coping with the enormous variety of controlled motions that humans and other animals may perform, little effort has been directed at how the resulting control solutions may be integrated to yield composite controllers with significantly broader functionalities. For example, if researcher A creates a walking controller for a dynamic character while researcher B creates a running controller for the same model, it would be highly beneficial if they could share their controllers (perhaps through an e-mail exchange) and easily create an integrated controller enabling the character to walk and run. This is a difficult problem, but its resolution would help pave the way towards controller libraries for dynamic animation to which communities of practitioners could contribute and utilize.

In this paper, we propose a simple yet non-restrictive and apparently effective framework for specifying specialist controllers and composing them into more general controllers. In our framework, individual controllers are black boxes which encapsulate control knowledge possibly gleaned from the biomechanics literature. Individual controllers must be able to determine two things: (1) a controller should be able to determine whether or not it can take the dynamic character from its current state to some desired goal state, and (2) an active controller should be able to determine whether it is operating nominally, whether it has succeeded, or whether it has failed. Any controller that can answer these questions may be added to a pool of controllers which is then managed by a *super-*

*visor controller* whose goal is to achieve non-trivial control tasks. We engineer a software system that implements our framework and show how it facilitates the sharing of controllers. Our system is built on top of DANCE [18],[1] a portable, extensible object-oriented modeling and animation system that implements a versatile plug-in mechanism. Individual controllers and the composition method are implemented as DANCE plug-ins.

We propose the challenge of implementing a dynamic *virtual stuntman* which should eventually be capable of emulating the complete repertoire of motor skills exhibited by stunt people in westerns and other film genres. In this paper, we make substantial progress towards this long-term goal by paying particular attention to dynamic falling due to various disturbances, getting up after a fall, and other complex combinations of motor control skills often associated with movie stuntmanship. We model our first-generation virtual stuntman as an articulated skeleton in 3D whose physical parameters reflect anthropometric data consistent with a fully-fleshed adult male. Fig. 1 illustrates the virtual stuntman autonomously performing a complex control sequence which is composed of 5 individual controllers responsible for falling, rolling-over, getting up, and balancing in gravity. The dynamic figure is pushed backwards by a sizable external force, its arms react protectively to cushion the impact with the ground, the figure lies in a supine position, rolls over to a prone position, pushes itself up on all fours, rises to its feet, and finally balances once again in place. A subsequent disturbance will elicit similar though by no means identical behavior, because the initial conditions will not be the same. Control sequences of such breadth for fully dynamic articulated figures are unprecedented in the physics-based animation literature.

After reviewing related prior work in the next section, we present the details of our control framework in Section3. Section 5 presents the details of the example in Fig. 1 along with several other examples that demonstrate the effectiveness of our framework. Section 6 concludes the paper and discusses avenues for future research opened up by our work.

## 2 Previous work

The simulation and animation of human characters is a challenging problem in many respects. Comprehensive solutions must aspire to distill and integrate knowledge from biomechanics, robotics, control, and animation. Models for human motion must also meet a particularly high standard, given our familiarity with what the results should look like. Not surprisingly, the divide-and-conquer approach is evident in most approaches, focusing efforts on reproducing particular motions in order to yield a tractable problem and to allow for comparative analysis.

The biomechanics literature is a useful source of predictive models for specific motions, typically based on experimental data supplemented by careful analysis. These models target applications such as medical diagnosis, the understanding and treatment of motor control problems, the analysis of accidents and disabilities, and high-performance athletics. Computer simulation is becoming an increasingly useful tool in this domain as the motion models evolve to become more complex and comprehensive [20, 21, 23]. Given the challenge of achieving high-fidelity motion models for individual motions, there have been fewer efforts towards integrated solutions applicable to multiple motions. [20] is one such example.

Robotics research has made remarkable progress in the successful design of a variety of legged robots [22] and, more recently, bipedal robots with anthropomorphic aspirations [17]. Despite their limited motion repertoire and rather deliberate movements, these robotic systems are truly engineering marvels. The work in [1] provides a good summary of behavioral architectures explored in the context of robotics. For the time being, however, computer simulations remain a more economical and flexible tool for exploring dynamic models of human and animal motion.

Computer animation is to a large extent unencumbered by the exacting fidelity requirements of biomechanical models and the mechanical limitations of robotic systems. This has spawned a great variety of kinematic and dynamic models for character motion [3, 4, 5]. Although kinematic models are currently widely used, dynamic models have the potential to be significantly more accurate and general models of motion. Dynamically simulated characters were first proposed some 15 years ago [2, 28] and since then have progressed in sophistication in a variety of directions. Controllers for specific motions such as walking, running, vaulting, cycling, and other human motions [11, 16, 29] have been successfully designed.

Dynamically simulated characters equipped with an integrated, wide-ranging repertoire of skills currently remain an unachieved goal. Some positive steps in this direction are evident, however. Examples include an integrated repertoire of motor controllers for biomechanically animated fish [24], a methodology for controller design and integration applicable to simple figures [27], a demonstration of successful integration for selected diving and gymnastic motions [29], and adapting a controller designed for one character to work on another character [12]. The work of Wooten [29] is the most relevant as an example of a sequence of successive transitions between several controllers for human motions such as leaping, tumbling, landing, and balancing. Transitions are realized by including the end state of some controllers in the starting states of other controllers. A *digital biomechanics laboratory* is proposed by BDI[15] as a tool for simulating a wide range of human motion. This currently remains ambitious work-in-progress.

Our work is aimed at building dynamic human characters with an integrated action repertoire. Unlike previous work aimed specifically at athletic endeavors, our methodology is to begin with a core set of simple motions, including balancing, small steps, falling reactions, recovery from falls, standing up from a chair, and others. At the same time, we propose a framework to facilitate the exchange and integration of multiple controllers.

## 3 Controller Composition Framework

As was stated in the introduction, our framework considers individual controllers as black boxes which are managed by a simple supervisor controller. When no controller is active, the supervisor polls the pool of controllers querying each whether it can handle the transition of the dynamic character from its current state to the desired state. The first controller that answers affirmatively is activated and it remains active until it has succeeded, failed, or until the desired state changes. The power of this scheme stems from the following attributes:

- *Simplicity*: The composition method is straightforward and easy to implement. It does not appreciably burden the controller design task.
- *Generality*: The composition method does not restrict the design of individual controllers. Each controller can be as primitive or as sophisticated as as its designer wishes.

A controller within the pool of available controllers can be as simple as a constant force, or as complex as a structured hierarchy of multiple levels of control abstraction. For example, as more controllers are added to the system, we may wish to group all the walking and running controllers together into a cluster that can be treated as one controller. This grouping may happen at various levels,

[1] DANCE is freely available for non-commercial use via the URL: www.dgp.toronto.edu/software/dance.html

where the first level correspond to the paramerization of controllers. In any case, instead of trying to design a single super-controller, knowledge about how to control the dynamic character is left for designers to encapsulate within their control algorithms, since they are best equipped to determine the conditions under which these algorithms are able to operate.

## 3.1 Controller abstraction

Our method requires controllers to define *pre-conditions*, *post-conditions* and *expected performance*. *Pre-conditions* are a set of conditions over the state of the character and the environment. If these conditions are met then the controller can operate and lead the character to satisfying the *post-conditions*. The *post-conditions* define a range of states for the final state of the character after the execution of the controller, assuming that the preconditions were met. In other words the controller realizes a mapping between a domain of input states to a range of output states for the character.

Because of unexpected changes in the environment this mapping may not always succeed, which motivates the notion of *expected performance*. The controller should be able to evaluate its performance in order to detect failure at any point during its operation. To do this, the controller must have knowledge of the current and expected state of the character or the environment at all times.

Defining pre-conditions, post-conditions , and the expected performance for complex characters, motions, and environments is not always a straightforward task; however, the biomechanics literature provides insights into appropriate specifications for a number of everyday and athletic motions, as we will discuss later in the paper. In any case, Controllers that adhere to these specifications can form a pool of available controllers managed by a simple supervising controller. Fig. 2 presents an overview of how the supervising controller works and its interaction with the individual controllers at every time step of the simulation.

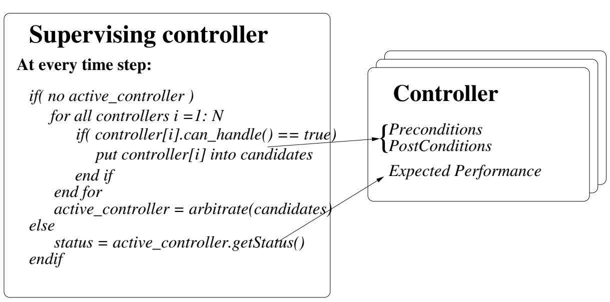



Figure 2: Controller selection and arbitration during simulation.

Before we elaborate on pre-conditions, post-conditions, and expected performance in subsequent sections, let us define the following quantities and symbols: The *state* $\mathbf{q} = [\mathbf{x}\ \dot{\mathbf{x}}]'$ of a figure is the vector of generalized positions $\mathbf{x}$ and velocities $\dot{\mathbf{x}}$, where the dot indicates a time derivative. The position and velocity of the center of mass are denoted as $\mathbf{c}$ and $\dot{\mathbf{c}}$ respectively. The *base of support* of a figure (often called the *support polygon*) is denoted as $\mathcal{S}$. It is represented by a polygon that surrounds the foot or feet that are in contact with the ground at any given time.

## 3.2 Pre-conditions

In general, Pre-conditions are relationships and constraints involving a number of different parameters. We have used the following parameters in our work:

- The input state $\mathbf{q_i}$ of the figure. Most of our controllers can operate within a small region of the state space which we denote $\mathcal{R}(\mathbf{q}_i)$.
- Environmental parameters. These include the contact points between the character and the ground, as well as the normal of the ground and the amount of friction at the contact points. In the following we denote conditions (generally indicated by the letter $C$) on the environment parameters as $C_e$.
- The balance of the figure. Usually, this is indicated by the relative position and velocity between the figure's center of mass $\mathbf{c}$ and the base of support. Typically, if the projection of $\mathbf{c}$ along the gravity vector $g$ does not intersect the base of support $\mathcal{S}$, the figure is out of balance. We denote the balance conditions as $C_b(\mathcal{S}, \mathbf{g}, \mathbf{c}, \dot{\mathbf{c}})$.
- A target state $\mathbf{q}_t$, or in general a target region of the state space $\mathcal{R}(\mathbf{q}_t)$, which can be provided by the user.

Pre-conditions consist of unions of instances of the above conditions and are denoted

$$\mathcal{P} = C(\mathcal{R}(\mathbf{q}_i), \mathcal{R}(\mathbf{q}_t), C_b, C_e). \tag{1}$$

## 3.3 Post-conditions

Successful operation of a controller brings the character from an initial state, as defined by the pre-conditions, to a desired state or a desired region $\mathcal{R}(\mathbf{q}_o)$ in the state space. This region along with balance $C_b$ and possibly environmental constraints $C_e$ form the post-conditions of a controller:

$$\mathcal{O} = C(\mathcal{R}(\mathbf{q}_o), C_b, C_e). \tag{2}$$

## 3.4 Expected performance

Our framework permits the automatic selection of the appropriate controller based on the information provided by the controllers themselves. Only the individual controllers can detect whether they are operating normally or they are going to fail. Failure in our case means that the controller cannot meet its post-conditions $\mathcal{O}$. A sudden change in the environment, a change in the target state by the user or badly designed preconditions can make the controller fail. The sooner a controller can detect failure the sooner another more appropriate controller can take over. That is very important for making a character behave naturally. For example, the character should not attempt to continue a walking gait if it has lost its balance and it is falling. In our implementation the expected performance $\mathcal{E}$ consist of expressions similar to those of the pre-conditions $\mathcal{P}$. In particular if the controller successfully completes its task in the time interval $[t_1, t_2]$ then:

$$\mathcal{E}(t_1) \in \mathcal{P} \quad \texttt{and} \quad \mathcal{E}(t_2) \in \mathcal{O}.$$

In accordance with definition (1), the expected performance of a controller at time $t$ can be defined as:

$$\mathcal{E} = C(\mathbf{q}(t), \mathcal{R}(\mathbf{q}_i), \mathcal{R}(\mathbf{q}_t), C_b, C_e). \tag{3}$$

## 3.5 Choosing pre/post-conditions

Designing the pre-conditions of a specific controller can be a difficult problem. For some cases the answer lies within the biomechanics literature [6, 19]. For example Pai and Patton [19] present a comprehensive study of balance in the sagittal plane and identify the conditions under which a human can compensate for postural disturbances and maintain balance without stepping. For some other cases, the preconditions are trivially defined by the desired motion itself. Certain controllers function as intermediate stages between other controllers. If controller B is the intermediate step between A and C then the post-conditions of A dictate the pre-conditions of B and similarly the pre-conditions of C define the post-conditions of B. Finally, in some cases the preconditions are computed by experimentation. For example a simple balance controller based on an inverted pendulum model [8] has intrinsic stability that can tolerate small disturbances. After the controller has been designed testing under disturbances of increasing magnitude can yield an approximation of the pre-conditions and the post-conditions.

## 3.6 Transitions

Another issue in composing controllers is the large number of possible transitions between different motions. There are three different ways to implement transitions [29]. Assuming $n$ different controllers the most complete method requires $n^2$ transition controllers that can take the system from the end state of one controller to the start state of another controller. A simpler way is to implement $2n$ transitions, $n$ from every controller to a common intermediate state and $n$ transitions from the intermediate state to each of the $n$ controllers. Alternatively, we can eliminate transitions altogether by making sure that the end state of a controller falls within the start states of another controller. Of course the last two methods may not be appropriate for all cases.

Our system does not regard transitions as a separate function. Whether a controller implements a transition or a new action is irrelevant from the system's point of view. Therefore any of the aforementioned transition methods may be used as long as it is implemented in accordance with our controller abstraction. In Fig. 1, for example, the controller that rolls the character over to a prone position serves as a transition between the previous controller that maintains the figure in a supine position and the subsequent controller that raises the figure to an upright stance.

Fig. 3 illustrates possible transitions between controllers. The thick lines indicate the following controller sequence: balance → fall → maintain → roll → rise → balance.

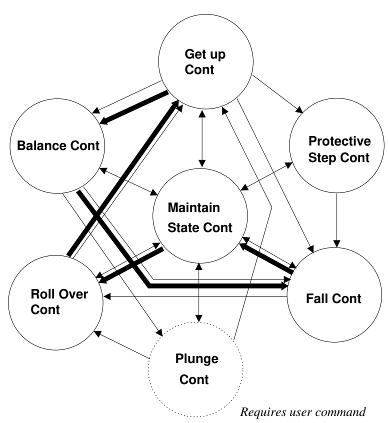



Figure 3: Transitions between controllers.

# 4 Implementation

The design and implementation of a software system that can be used as a general tool for animation and which enables the sharing of controllers as reusable modules is an interesting challenge from a software engineering point of view. Our system, which is implemented on top of DANCE [18], addresses both issues.

## 4.1 DANCE

An integral part of our software, DANCE is a system that supports the animation and control of articulated objects and characters. Its purpose is to provide a portable platform that researchers can use to implement animation and control techniques with minimal design and implementation overhead.

The core of the system implements data structures for hierarchically defined articulated creatures. The fundamental objects of DANCE are *Articulated Objects*, *Actuators*, and *Simulators*, which are loadable as plug-ins in accordance with simple APIs.

Articulated objects are objects with skeleton hierarchies. They have kinematic properties and, usually, fully dynamic physical properties as well. Our virtual stuntman, which will be described shortly, is a dynamic articulated object within DANCE.

An actuator is a generic concept that includes anything that can exert forces or, in general, interact in any way with articulated objects or other actuators. For example, gravity, the ground, the collision mechanism, the supervisor controller and individual controllers are implemented as actuators. DANCE places no restrictions on the complexity of the controllers.

Simulators compute the equations of motion of all the dynamic characters and other objects in DANCE. DANCE offers built in support for SD/FAST, a commercial system which produces optimized simulation code [13]. However, any simulator that follows a simple API can be dynamically loaded into the system. Our simulators are automatically produced by SD/FAST [13] from description files. They use Kane's method for computing articulated dynamics and a fourth order explicit Runge-Kutta time integrator for numerically simulating the motions.

Actuators and simulators are implemented as DANCE plug-ins. This allows the user to dynamically load controllers and simulators at runtime. In addition, researchers can exchange, simulators, and controllers in the form of dynamically linked pieces of code.

### 4.1.1 Collisions

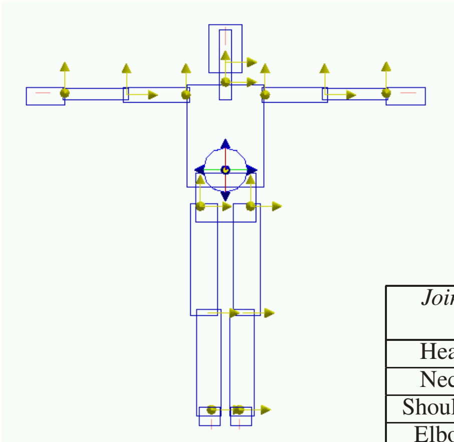
Self and object collisions are handled by the *Collision* actuator. This actuator works on pairs of objects. The DANCE API allows it to work with objects that have different simulators. Collision detection is based on a library [9] using oriented bounding boxes. Collision resolution uses a penalty method that corrects geometry interpenetration using spring forces. As with all penalty methods, it can make the system stiff, but it has performed well in our experiments to date.

## 4.2 Virtual stuntman

### 4.2.1 Dynamic model

Fig. 4 depicts our 3D human model in bounding box form for clarity. The yellow arrows indicate the joint positions and axes of rotational degrees of freedom which are also presented in the table. There are 37 degrees of freedom, six of which correspond to the global translation and rotation parameters. The dynamic properties of the model, such as mass and moments of inertia, are taken from the biomechanics literature and correspond to a full-flesh human being.

The model is equipped with natural limits both on the motion of the joints and the strength of its muscles. However, DANCE has no built in muscle model and does not enforce the limits automatically. Users can implement the model they prefer and it is up to them to respect the limits of the model. Our plug-in control scheme uses rotational spring forces for control and enforces the limits on the joints with exponential springs.



| *Joint* | *Rotational DOFs* |
|---|---|
| Head | 1 |
| Neck | 3 |
| Shoulder | 2 |
| Elbow | 2 |
| Wrist | 2 |
| Trunk | 3 |
| Hip | 3 |
| Knee | 1 |
| Ankle | 2 |

Figure 4: Dynamic model and its degrees of freedom (DOFs).

#### 4.2.2 Pose and continuous control

Most of the controllers for our virtual stuntman are based on pose control, which has often been used both for articulated objects [26, 25] and soft objects [7]. Pose control is based on cyclic or acyclic finite state machines with time transitions between the states. Each state of the controller can be static or depend on feedback parameters. For some of our controllers, we use continuous control, in the sense that the control parameters are tightly coupled with some of the feedback sensors. From our experience, pose controllers are more suitable for periodic motions such as walking and jumping while continuous control is better for motions such as balancing in place.

We designed several controllers based in part on experimental studies of how humans detect loss of balance [19] and extensive analysis of protective and falling behaviors [6]. The resulting parameterized controllers have been enhanced with appropriate *pre-conditions, post-conditions*, and *expected performance* and have been integrated using an arbitration-based supervising controller.

#### 4.2.3 Sensors

Each controller has full access to the internal data structures of DANCE including all the information associated with any character or object in the system. This way the controllers can define arbitrary sensors that keep track of necessary information such as state parameters for feedback loops, the state of the environment etc. For efficiency, the supervisor controller calculates a number of common sensor values that are available to all the controllers. These sensors include:

- Position, velocity, acceleration of the center of mass.
- The support polygon which is the convex hull of the feet.
- Position of the center of mass with respect to the support polygon.
- Links that are in contact with the ground.
- Facing-vector, up-vector considered with respect to the coordinate system of the hip.

#### 4.2.4 Command interface

Currently, the user directs the voluntary motions of the virtual stuntman by entering command strings to the supervisor controller which, in turn, forwards the string to every controller that it queries during the search phase. For example the controller that makes the character jump off the horse responds to the string “jump-off-horse”. We have found that using a string instead of a desired state is more intuitive as a user interface.

## 5 Results

Within the scope of our prototype system is a composite controller that is capable of handling a large number of everyday tasks, such as walking, balancing, bending, falling, and sitting. While motion capture solutions based on blending and warping techniques may give satisfactory results for such tasks in the short term, controller based approaches reveal more about the physics, planning, and control of such motions and they are the basis for more general, long-term solutions. We present our results in two parts—everyday actions and stunts.

### 5.1 Everyday actions

We have begun our implementation with the simple tasks of standing, recovering balance when pushed, and falling. An autonomous human agent should be able to balance, standing naturally in place. Should loss of balance occur, the character should react naturally either with a restoring motion or with a protective falling behavior depending on which action is appropriate in each case. Affording a dynamic articulated figure with natural reactions to loss of balance or impending falls is an essential step towards believable autonomous characters. Biomechanics research has performed extensive studies on the subject [14] since falling is the cause of a large number of everyday and sport-related injuries.

#### 5.1.1 Balancing

A *balance* controller is responsible for maintaining an upright natural motion. This controller is based roughly on an inverted pendulum model [8]. Despite the fact that the body of the character is not as rigid as the inverted pendulum hypothesis suggests, the approximation works well in practice.

The pre-conditions and post-conditions of this controller are straightforward. The body has to be in a balanced upright position, the velocity and acceleration of the center of mass should not exceed certain threshold values as explained in [19], and both feet must maintain contact with the ground at all times. The controller can tolerate small perturbations of the posture and the velocity/acceleration of the center of mass by stiffening the ankle joints. For larger accelerations of the center of mass, the controller actively actuates the ankle joint to reduce the acceleration of the center of mass. The post-conditions are similar to the pre-conditions. As an example we present both in mathematical form using the notation defined in Section 3:

$\mathcal{P}$ :

| | |
|---|---|
| Acceleration: | $\ddot{\mathbf{c}} < 0.1\ \mathrm{m/sec}^2$. |
| Velocity: | $\dot{\mathbf{c}} < 0.3\ \mathrm{m/sec}$. |
| Balance: | $\mathrm{projection}(\mathbf{c}) \in \mathcal{S}$. |
| Posture: | $(1/n)\sum_i \lvert \mathbf{q}[i] - \mathbf{q}_0 \rvert < 0.1$ rad, where $i = (\mathrm{thigh}, \mathrm{knee}, \mathrm{waist})$ and $n$ is a normalization parameter. |

$\mathcal{O}$ :

| | |
|---|---|
| Acceleration: | $\ddot{\mathbf{c}} < 0.01\ \mathrm{m/sec}^2$. |
| Velocity: | $\dot{\mathbf{c}} < 0.05\ \mathrm{m/sec}$. |

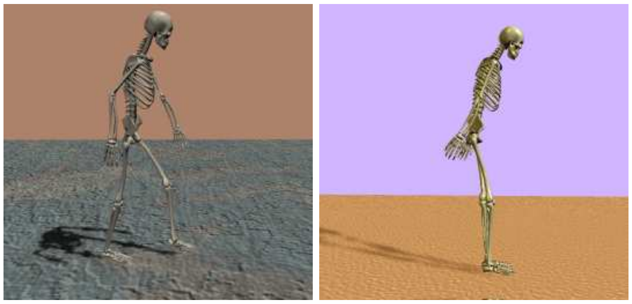


Figure 5: Responses of a human character to various pushes.

Balance: projection$(\mathbf{c}) \in \mathcal{S}$.
Posture: $(1/n) \sum_i |\mathbf{q}[i] - \mathbf{q}_0| < 0.1$ rad, where $i =$ (thigh, knee, waist) and $n$ is a normalization parameter.

Fig. 5 shows examples of how a character may attempt to maintain balance in response to external disturbances by shifting its weight, taking a step or bending at the waist. If the character cannot maintain balance, however, it should fall in a natural, protective manner. The next section describes the fall controller that we have implemented.

#### 5.1.2 Falling

The manner in which people fall depends on a number of factors such as their physique, their age and their training. For example, the work in [14] shows that, during a fall, older people tend to impact their hip first, while teenagers falling under the same conditions tend to impact their hands first. Our *fall* controller is designed with the average person in mind. Its main function is to absorb the shock of the impact using mostly the hands. The second frame in Fig. 1 also demonstrates the action of the fall controller within the complex virtual stuntman sequence.

The pre-conditionsof the fall controller are defined in accordance with those of the balance controller. Situations that are beyond the capabilities of the latter should be handled by the fall controller. Our implementation of the fall controller can handle falls in any direction. Fig. 6 depicts three situations handled by the fall controller.

The most natural action for a person that has fallen down is to get up as soon as possible. Again, the manner in which people get up depends on a number of different factors, such as age, athletic ability, etc. We have implemented three different ways for our figure to get up. Two of those were designed for the average person, while the third may be categorized as a vigorous athletic maneuver or stunt, and is therefore presented in Section 5.2.

#### 5.1.3 Sitting and rising from a chair

A very common everyday task performed by all people is sitting down or rising up from a chair. We have implemented a controller that can do both depending on the instructions of the animator. Apart from the command string supplied by the user, the pre-conditions are either a balanced upright posture or a balanced sitting posture. The post-conditions are similarly defined. The resulting action is illustrated in Fig. 7.

#### 5.1.4 Rising from a supine position

Getting up off the ground is a difficult motion to work with. It involves rapid changes of the contact points and significant shifting of the figure's weight. In addition, the frictional properties of the ground model can influence the motion.

The pre-conditions for this controller are straightforward. The character must be lying on its back flat on the ground within some tolerance. Similarly the post-conditions are that the character should be on its feet with its center of mass within the support polygon. Then it would be up to another controller to take over and bring the character from a crouching position to a standing one. A snapshot of the motions is shown in Fig. 8.

#### 5.1.5 Rolling over

When lying on their back, some people may choose to roll-over to a prone position before they attempt to get up. We have implemented a *roll-over* controller that can emulate this action. The fourth frame in Fig. 1 demonstrated the action of the roll-over controller.

The pre-conditions of the roll-over controller require a supine posture, and no movement of the center of mass. They could easily include a desired state, so that the controller will attempt to function only if the animator wishes. The post-conditions of the roll controller are fairly simple and they include any prone position for which the character is fairly straightened; i.e., no crossing of legs or arms, etc.

#### 5.1.6 Rising from a prone position

Frames 5–9 in Fig. 1 demonstrate the action of a controller that enables the virtual stuntman to rise from the prone position. When lying face-down, the pre-conditions can be fairly relaxed. Our controller assumes that is has the time to change the state of the character to one from which it knows how to rise. As long as the figure is not lying on its arms and the ground is relatively flat it will attempt to get up. The post-conditions are chosen such that they satisfy the pre-conditions of the *balance* controller.

### 5.2 Stunts

Apart from everyday actions, we want our virtual stuntman to be able to do a variety of other voluntary actions dictated by the animator. Such actions can potentially include vigorous and/or physically dangerous actions. If a large number of researchers contribute controllers the character can eventually be equipped with many impressive abilities demonstrated by experienced stunt-people in the movies.

#### 5.2.1 Kip move

An elaborate athletic motion, known as the kip, that is sometimes observed in martial arts films is depicted in Fig. 9. The controller is based on a pose controller whose pre-conditions include a fairly large variation of supine positions. As before, the first part of the controller makes sure that the character assumes a position suitable for performing the kip. The larger part of the motion is ballistic, which focuses the control mainly at the kick off and landing phases. The last part of the controller applies continues control which brings the stuntman to an erect position from which the balance controller can take over. In other words the post-conditions of the kip controller satisfy the pre-conditions of the balance controller.

#### 5.2.2 Plunge and roll

Fig. 10 shows the stuntman performing a suicidal dive down stairs. This is a stunt that any real stuntperson would probably prefer to avoid doing. The character can be instructed to dive forward at an angle controlled by the user. When the hands contact the ground a roll over the head maneuver is performed. The pre-conditions of this controller are defined be an upright position and little movement of the center of mass.

Figure 6: Falling in different directions

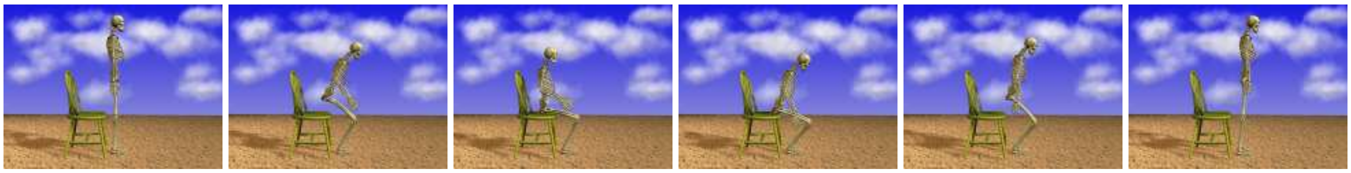


Figure 7: Sitting and rising from a chair

Figure 8: Rising from a supine position on the ground and balancing erect in gravity.

Figure 9: Kip move: A more vigorous way of getting up from the supine position as in the first frame of Fig. 8.

### 5.2.3 Jumping off a horse

Our system DANCE allows for the co-existence and interaction of both dynamic and kinematic objects. We have implemented a collision actuator, described in Section 4.1.1, which models the interaction between colliding objects. Based on this we have implemented a controller that allows the character to jump off the back of a horse and possibly attack another character. The horse is animated kinematically, while the figure is dynamic. This test case nicely exploits the features of physics-based techniques. For example, we can easily create variations of the result by simply changing the initial position of the character under attack. Fig. 11 shows snapshots of an animation created this way. Similarly we have constructed an example[2] of a character tackling an unsuspecting bandit.

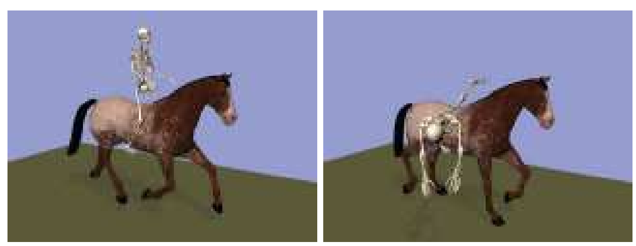


Figure 11: Jumping off a moving horse.

## 6 Conclusion

The challenges of physics-based controller design plus the technical obstacles that researchers face when attempting to share their algorithms has hindered progress in the important area of physics-based character animation. This paper has presented a methodology for ameliorating the problem. We have proposed a general physics-based controller composition technique implemented within a freely available system for modeling and animating articulated characters. We believe that our work can form the basis of a system that will allow researchers to exchange and compose controllers. It is our hope that collective efforts among numerous practitioners will result in complex composite controllers capable of synthesizing human-like motor behaviors.

[2] Please see the video tape

## References


[1] Ronald C. Arkin. *Behavioral Robotics*. MIT Press, 1998.

[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.

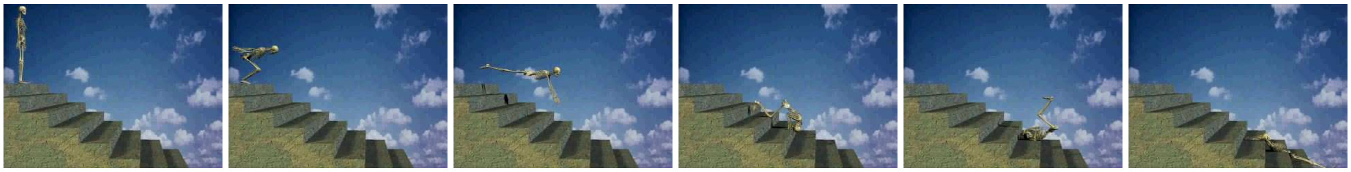


Figure 10: Ouch!


[3] N. Badler, C. Phillips, and B. Webber. *Simulating Humans: Computer Graphics, Animation, and Control*. Oxford University Press, 1993.

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

[5] Tolga Capin, Igor Pandzic, Nadia Magnenat Thalmann, and Daniel Thalmann. *Avatars in Networked Virtual Environments*. John Wiley & Sons, 1999.

[6] M. C. Do, Y. Breniere, and P. Brenguier. A biomechanical study of balance recovery during the fall forward. *Journal of Biomechanics*, 15(12):933–939, 1982.

[7] Petros Faloutsos, Michiel van de Panne, and Demetri Terzopoulos. Dynamic free-form deformations for animation synthesis. *IEEE Transactions on Visualization and Computer Graphics*, 3(3):201–214, 1997.

[8] R. C Fitzpatrick, J. L. Taylor, and D. I. McCloskey. Ankle stiffness of standing humans in response to imperceptible perturbation: reflex and task-dependent components. *Journal of Physiology*, 454:533–547, 1992.

[9] Stefan Gottschalk, Ming Lin, and Dinesh Manocha. OBB-Tree: A hierarchical structure for rapid interference detection. In *Computer Graphics (SIGGRAPH '96 Proceedings)*, pages 171–180, 1996.

[10] J. K. Hodgins, W. L. Wooten, D. C. Brogan, and J. F. O'Brien. Animating human athletics. In *Computer Graphics (SIGGRAPH '95 Proceedings)*, pages 71–78, 1995.

[11] J. K. Hodgins, W. L. Wooten, D. C. Brogan, and J. F. O'Brien. Animating human athletics. *Proceedings of SIGGRAPH '95, ACM Computer Graphics*, pages 71–78, 1995.

[12] Jessica K. Hodgins and Nancy S. Pollard. Adapting simulated behaviors for new characters. *Proceedings of SIGGRAPH 97*, pages 153–162, August 1997.

[13] Michael G. Hollars, Dan E. Rosenthal, and Michael A. Sherman. Sd/fast. Symbolic Dynamics, Inc., 1991.

[14] E. T Hsiao and S. N Robinovitch. Common protective movements govern unexpected falls from standing height. *Journal of biomechanics*, 31:1–9, 1998.

[15] Boston Dynamics Inc. The digital biomechanics laboratory. `www.bdi.com`, 1998.

[16] Joseph F. Laszlo, Michiel van de Panne, and Eugene Fiume. Limit cycle control and its application to the animation of balancing and walking. *Proceedings of SIGGRAPH 96*, pages 155–162, August 1996.

[17] Honda Motor Co. Ltd. www.honda.co.jp/english/technology/robot/.

[18] Victor Ng and Petros Faloutsos. Dance: Dynamic animation and control environment. Software system, www.dgp.toronto.edu/DGP/DGPSoftware.html.

[19] Yi-Chung Pai and James Patton. Center of mass velocity-position predictions for balance control. *Journal of biomechanics*, 30(4):347–354, 1997.

[20] Marcus G. Pandy and Frank C. Anderson. Three-dimensional computer simulation of jumping and walking using the same model. In *Proceedings of the VIIth International Symposium on Computer Simulation in Biomechanics*, August 1999.

[21] Marcus G. Pandy, Felix E. Zajac, Eunsup Sim, and William S. Levine. An optimal control model for maximum-height human jumping. *Journal of Biomechanics*, 23(12):1185–1198, 1990.

[22] M. H. Raibert. *Legged Robots that Balance*. MIT Press, 1986.

[23] Cecile Smeesters, Wilson C. Hayes, and Thomas A. McMahon. Determining fall direction and impact location for various disturbances and gait speeds using the articulated total body model. In *Proceedings of the VIIth International Symposium on Computer Simulation in Biomechanics*, August 1999.

[24] Xiaoyuan Tu and Demetri Terzopoulos. Artificial fishes: Physics, locomotion, perception, behavior. *Proceedings of SIGGRAPH'94*, pages 43–50, 1994.

[25] M. van de Panne. Parameterized gait synthesis. *IEEE Computer Graphics and Applications*, pages 40–49, March 1996.

[26] M. van de Panne, R. Kim, and E. Fiume. Virtual wind-up toys for animation. *Graphics Interface*, pages 208–315, 1994.

[27] Michiel van de Panne, Eugene Fiume, and Zvonko Vranesic. Reusable motion synthesis using state-space controllers. *Computer Graphics (SIGGRAPH '90 Proceedings)*, 24(4):225–234, August 1990. ISBN 0-201-50933-4. Held in Dallas, Texas.

[28] Jane Wilhelms and Brian A. Barsky. Using dynamic analysis to animate articulated bodies such as humans and robots. In *Graphics Interface '85*, pages 97–104, May 1985.

[29] Wayne Wooten. *Simulation of Leaping, Tumbling, Landing, and Balancing Humans*. PhD thesis, Georgia Institute of Technology, March 1998.