Interactive Articulated Character Dancing Animation in Real Time

Chong Chen, Chang-Zhi Li and Bao-Gang Hu
Institute of Automation, Chinese Academy of Science, China
{chenchong, czli, hubg}@nlpr.ia.ac.cn

Abstract

In this paper, an interactive platform is introduced that users can create dancing animation of models with various topologies, including humans, flowers, animals and etc. After the key postures of dancing are set interactively by animators, the program produces the final smooth animation automatically. What is more, certain posture pools have been designed for users in advance, which greatly reduces the labor burden of creating individual postures and facilitates operating the system. After users have stored the individual postures, our system will calculate the transition between those postures with dynamic theories, and produce the dancing animation smoothly in real time. Two different physical models are applied in this system. In the interaction procedure, the models are set as spring model, which could propagate the force at one joint to the entire skeleton, driving the motion of the whole model correspondingly. In the process of animating, we make good use of elastic joint model to sequence and polish the interpolation. In addition, some natural movements can be easily simulated in the system, such as the waving motion of trees in the winds, with the spring model and elastic joint model working together.

CR Categories: I.3.5 [Computer Graphics]: Physically based modelling; I.3.7 [Computer Graphics]: Animation

Keywords: Physically based animation, Interactive, Spring Model, Simulation, Elastic Joint Model, Inverse Kinetics

1 Introduction

From *Steamboat Willie* by Disney in 1928, the first sound cartoon in the world, the cartoon industry has made rapid progress in the past years. Nowadays there are about fifty to sixty cartoons produced each week just in Japan. Generation after generation, both children and adults are impressed by lively characters, interesting stories, various subjects and wonderful pictures of cartoons. However, the traditional key-frame technique to produce cartoon manually is an extremely great burden for animators. With the development of computer science, how to aid animators in producing cartoons with computer is an active and promising research field.

Our approach is based on the dynamic equations of motion. As physically based animations are preeminent to produce realistic motions, and moreover, motions is created automatically according to the inherent physics theories in such an animation system. That is, in this system, what animator need to do is to regulate some physical parameters and specify the beginning and ending postures, and the system will produce the final animation automatically.

The goal of this work is to create an expressive and recreational animation system. We finally focus on the dancing animation, for dancing is very representative. Furthermore, it generally exhibits repeated patterns in dancing, which facilitate the design of postures. We also want the system to provide an easy-to-use interface, and grant users the power to create motions representing their individualities. We also intend to offer a system for animating universal models, including humanoid biped, quadruped, myriapod insects, plants, and so forth.

As a matter of fact, to generate the animation completely by computers, without any involvement of human intelligence is not possible at present [Lu and Zhang 2002]. In this system, users should first choose a dancer from what we have provided in the model pool. Because our method is based only on dynamic physics theories and not related with the topology of dancer models, the dancing characters can be various. Then users may set individual postures directly by dragging the selected joint of the skeleton to the desired position interactively. Subsequently, the individual postures are stored as key-frames of the final dancing animation. We also afford users some most ordinary postures in the posture pools of each model to make the creation procedure easier and more standard. At last, we apply the elastic joint model to the dancer, interpolating between the key-frames and producing dancing animation in real time. Because our model is a second-order dynamic system, the final animation is smooth and realistic, avoiding the jerky switches usually caused by direct linear interpolation.

The remainder of this paper is organized as follows. After reviewing the related work in Section 2, we will briefly describe our whole system in Section 3. We introduce the application of spring model and the pre-set posture pools in the system to realize the interactive control in Section 4, and the elastic joint model to smooth the dancing animation in Section 5. Some resulting animations will be shown in Section 6. We conclude the paper and point out the future work directions at last.

2 Related Work

A common technique to create character-dancing animation is motion capture, that the motions of artificial characters were choreographed with identical and slightly deformed copies of real-life motions. For instance, Li and his colleagues [Li et al. 2002] designed a two-level statistical model to characterize the dynamic and stochastic nature of the figure motion. Firstly they created motion texture for synthesizing the dancers' motion. Then under the constraints of the beginning pose and ending pose, the system could find the lowest cost path automatically and smooth all the transitions. But their algorithm did not allow the poses to be edited by users to deviate from the original ones too much. Kim et al. [Kim et al. 2003], however, presented a novel scheme to synthesizing new motions from the example motions while preserving their rhythmic pattern. Their scheme first extracted the basic movements and their transitions from example motions, and then constructed a movement transition graph representing the example motions. However motion capture method has disadvantages on interactivity and adjustability, since captured data are difficult to segment and edit. [Nakata 2002]

Oore et al. [Oore et al. 2002] introduced local physical model - the DIGITAL MARIONETTE animation system, which could automatically produce some behaviors and let users control the virtual puppetry via a PD-controller. There are also inertia parameter M, elastic coefficient K, and damping matrix C in their system, but they

only developed models for the knees and ankles of an interactively-animated 3D anthropomorphic character. What's more, there were no such poses as leaning or bowing in their final animation, which would lead to the unbalance of the character. In addition, the virtual puppetry could not jump, that is, at least one point of the two feet should be fixed on ground, and in the interactive control, users could only set some limited physical properties of the joints. Such kind of work is also in [McMillan et al. 1995], and their algorithm is O(*N*).

On the other hand, many people create some animation system with intelligence, such as [Tu et al. 2000]. Recently Lu [Lu and Zhang 2002] is engaged in the research of full life cycle computer aided animation generation technique, that is, from the story in natural language to the demonstration of pictures, the whole process is done with the aid of computers. This study is related with artificial intelligence, natural language understanding, and computer graphics. It will greatly reduce the workload of producing animation, however nowadays the process must have people's interference.

In the respects of interaction, frameworks for creating and animating the elastically deformable characters interactively have been developed as in [Capell et al. 2002][Turner and Gobbetti 1998]. Dontcheva et al. [Dontcheva et al. 2003] created an action-based animation system, in which the animator's real-world, expressive motions were mapped into the character's virtual world. Some widgets were used to control the user features, which mapped to character features. Visual feedback maintained a tight coupling between the animator and the character. In J. Popovic's [Popovic et al. 2000] system, an animator could select bodies at any time and simply dragged them to desired locations. In response, the system computed the required physical parameters and simulated the resulting motion. While in our system, all the physical parameters are initially defined or interactively regulated by users in real time. The system just computes the torque caused by displacement and torsion through the second-order dynamic equations.

3 System Overview

First of all, we should create the character models with the help of some modelling softwares. In our practice, we just apply the skeletons to the prepared model meshes through MilkShape 3D©. Of course, the skeleton can be designed with more details, which means more accurate control of the character and the more elaborate animation. However it also means more burdens in creating the postures and more computation cost in animation. We should achieve a balance in the dilemma soundly.

The system of this work is divided into two levels: the high level for interactive control and animate actors and the low level for regulating the dynamical motions and physical properties of each joint. Animators can interactively control the characters at the high-level intuitively, with no awareness of the inner potential mechanisms. The system will automatically produce the final animation through the low-level.

As can be seen from Figure 1, we design the system as a linear spring model and a second-order dynamic elastic model. The connected springs can propagate the local deformation throughout the whole skeleton, which is desirable for the interactive control globally. On the other hand, the elastic model is created as a linear dynamic system (LDS) as in [Li et al. 2002]. All the transitions are computed locally, in the parent joint coordinates. Consequently, the system should carry out the relative angular displacement from global absolute position coordinates. We also simplify the LDS in order to create the animation in real time, keeping animation realistic at the same time.

Our system is dynamic, interactive and real-time. If animators could design the postures according to their understanding of certain music, they will choreograph these postures to music with their own individuality and express their inner preferences as well.

4 Interactive Controls

4.1 Spring Model

Skeleton animation consists of changing joint angles. In a typical system based on inverse kinematics, animators specify the end positions of some parts of the skeleton, and the system can compute the necessary joint angles of other parts to put the specified parts to the desired position automatically. [Magnenat-Thalmann and Thalmann 1990] Our system also involves inverse kinematics, for users select and drag certain joint to the target position, and system should put all the character skeleton to the proper position. Our method is to model the skeleton with spring model because connected springs can propagate forces among the whole skeleton, maintaining the minimum energy of the system.

In practice, spring model has been widely adopted to simulate deformable objects, for example the clothes [Meyer et al. 2001][Provot 1995]. There are two kinds of spring model, with or without mass. The spring model with mass can be used to create more real animation, but it is also time-consuming, which is unsuitable to be utilized in interactive animation. In our system, the prime purpose is to permit users to set and store the appropriate postures as they want in real time. Accordingly we adopt the spring model without mass to alleviate the computation cost.

The original method is to minimize the energy of the whole model. Because the joints in this system are created without mass, there is only potential energy and no kinetic energy here. A case in point is the example of a simple spring system like Figure 2. According to the theories of potential energy in a spring system, we should minimize the energy

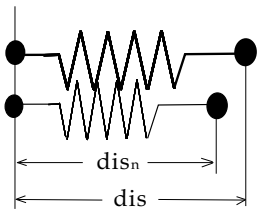


Figure 2: Energy in a spring system

$$E_p = \frac{1}{2}k(dis - dis_n)^2 \tag{1}$$

where dis is the current distance between two joints, while dis_n is the rest length of the spring. And the final algorithm is as follows.

$$D = P^2 - P^1 \tag{2}$$

$$dis = \|D\| \tag{3}$$

$$ext = dis - dis_n \tag{4}$$

$$\triangle X = \frac{ext}{dis}D \tag{5}$$

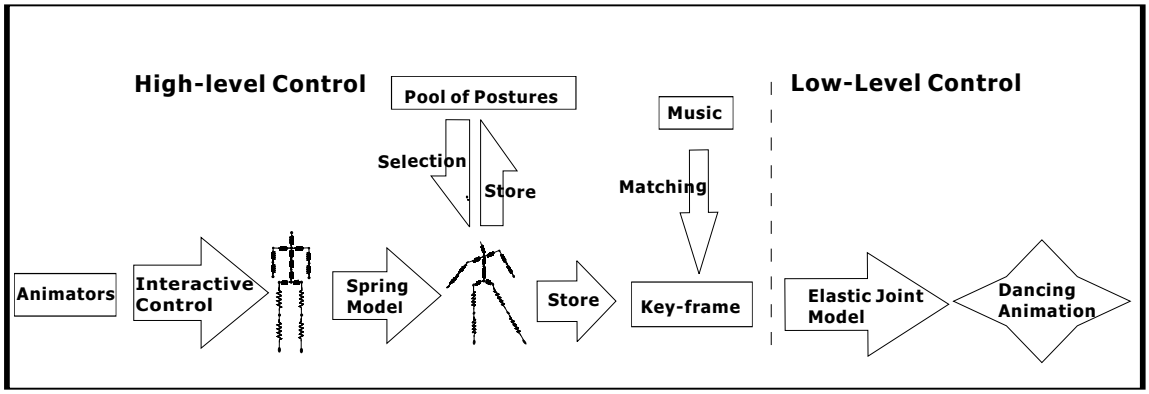


Figure 1: Flow chart for the interactive system

$$P^2 = P^1 + h * \Delta X \tag{6}$$

here

$$D = \begin{pmatrix} dis_x \\ dis_y \\ dis_z \end{pmatrix} = \begin{pmatrix} p_x^2 - p_x^1 \\ p_y^2 - p_y^1 \\ p_z^2 - p_z^1 \end{pmatrix}$$

$$dis = \sqrt{dis_x^2 + dis_y^2 + dis_z^2} \tag{7}$$

and h is the step coefficient.

From above, we carry out the effect of one neighbor joint on another. But each joint in the system, except the root joint and leaf joints, has one parent joint and at least one child joint, and the sum of these effects must be taken into consideration. Here we calculate the sum of forces on every joint according to Hooke's law.

$$F = k * \Delta X \tag{8}$$

We set $k = 1$ to facilitate the computation. We can also figure out *E* from above equation. As can be seen, *F* and *E* could be treated as the same to some extent.

Then the sum of forces on each joint is calculated recursively throughout the whole skeleton, as shown Figure 3, to figure out the final displacements of every joint in the skeleton. It produces the forces driving the movements of the entire skeleton, which will be illustrated in the next section.

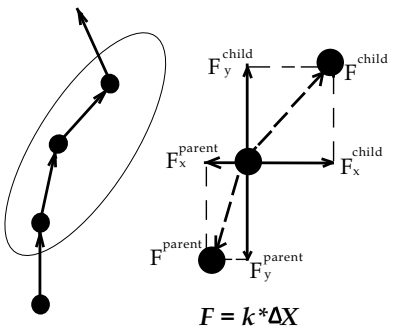


Figure 3: Force on a joint

Initially, we set $h = 1$ directly, because the system energy is kept *zero* in such a case. But there is unexpected vibration while dragging a joint to move the entire skeleton. When $h = 1$, the points move so far at a step that the system is unstable. Therefore we set h less to slow down the movement of each point.

However as can be expected, an inherent shortcoming of spring model, the elongation, appears, which may result in unacceptable deformation in our models. Ultimately we found that $h = 0.5$ is a good setting, It is the critical point. When it is greater, there is vibration while skeleton moving, while if less than that value, there is no such phenomenon. But the less h is, the greater the elongation is. As a result, we finally set $h = 0.5$.

4.1.1 Making Dancing Models Stiffer

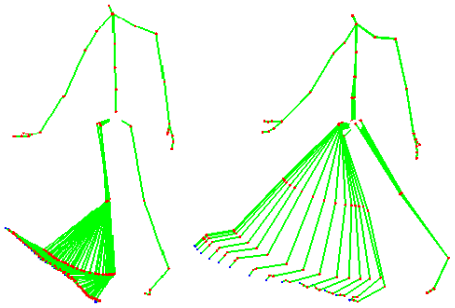
In the beginning experiments, another problem with the spring model is also found in the system. If we just provide springs between the immediately neighbors, the model looks extremely flexible. We then impose additional springs in the neighbor 4 and 6 joints, to make the model stiffer. As a matter of fact, to set spring between two joints is to maintain their initial distance constant during the interaction procedure.

In Figure 4, the blue point in foot indicates the joint selected and dragged by the mouse. As it shows, because we have imposed springs between the 2, 4 and 6 neighbor joints, the leg seems stiff. In contrast, if only the force produced by the springs between the immediate 2 neighbors is calculated, the whole skeleton is particularly flexible and unrealistic.

4.1.2 Fixing Joints in the Skeleton

As illustrated previously, the spring model is universal to model subjects of different topologies, and we regard any joint in the skeleton having 6 DOFs (degree of freedom), unattached to each other. But in most cases, this is not true. For example, knees in human body can not bend forward and there are interlinkings between crotch, knee and ankle. Furthermore, many articulations are only of limited ranges of DOF. For instance, we cannot turn our heads for 360^o.

On the other hand, after more springs are added between neighbor several joints, the skeletons are much stiffer. If the ankle is dragged, the whole leg will follow the motion and it is something like a stick.



(a) low stiffness (b) high stiffness

(On the left is the skeleton with springs imposed just between the nearest 2 neighbor joints, while on the right is that with springs among the 2, 4 and 6 neighbor joints.)

Figure 4: Dragging the foot of a human skeleton

Then we grant animators the ability to select and fix some joints. For example, animators can finely adjust the position and orientation of wrist by fixing the joints shoulder and elbow. This function will permit animators to design postures with more accuracy.

4.1.3 Reducing the Elongation

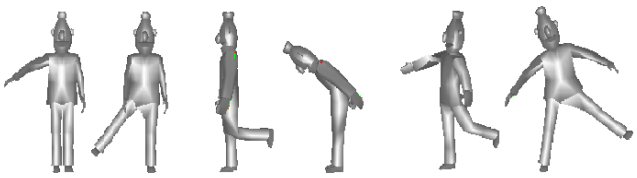
Due to the application of spring model, the displacement of a joint is proportional to the altered distance between its neighbor joints and it. There is also the phenomenon of "super-elongated" in our model. But because of the less number of joints in our system compared to the one finished by Provot [Provot 1995], we do not adopt his method of apply a dynamic inverse procedure. In our accomplishment, we set a threshold d_{max}, for example $d_{max} = 0.05$, which means that the length of the springs cannot exceed their rest length by 5% or more. The system will iterate from the springs exceeding this threshold and propagate the force among the whole skeleton.

The criterion to end the iteration is diverse, for instance, the iteration can be stopped when all the springs converge to satisfy that all springs do not exceed their rest length by 5%, or by computing for a certain period time, say 0.1 second. However we choose to predefine the times of iteration N, to make our algorism easier. The experiments show that $N = 10$ is an ideal setting. When N is less than 10, the "super-elongated" is also obvious in some cases, while if N is greater, we might achieve more satisfying results, but it is comparatively with more computation cost.

4.2 Postures Pools

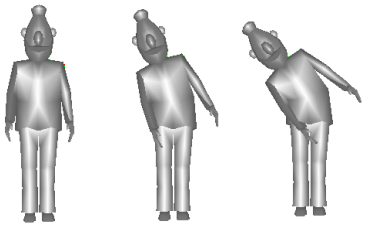
On the other hand, we have pre-set some basic postures for users. These postures are the most common ones as we can imagine, including bending the waist, the elbows, the knees, the wrists, or the ankles, nodding head, raising the arms or the legs. What's more, the postures from the pools can be combined into a new complicated posture. Some are shown in Figure 5. We also set them with extent coefficients to demonstrate the similar posture with different exaggeration, exhibited in Figure 6. Hence, users can obtain a group of similar postures by adjusting the extent parameter.

Of course, animators could set their own posture pools by creating individual postures and storing them into the pools. Users need not to design every posture each time, for they can choose the appropriate postures from the posture pools to create their dancing. With the help of these settings, users can produce animations more effectively.



(From left to right, they are postures *right arm raised up*, *right leg raised up*, *left knee bent backwards*, and two complicated postures combined from the elemental postures.)

Figure 5: Some elemental postures and the combined postures



(The postures are the model *Human at rest*, *leaning right* with 50% and 100% degree of exaggeration respectively.)

Figure 6: Postures with increasing exaggeration extent

5 Dancing Animation

In the above section, we have designed the individual postures of animation, and then we will animate the model by interpolating between these postures. But the linear interpolation will result in jerky motions in the animation. Consequently, we adopt the second-order dynamic equations.

In the elastic model, every joint is regarded as rigid link and elastic hinge, and the hinge determines DOF of the joint. A joint is also considered as a spherical joint, so an angular spring is set in every hinge. When the joint rotates, there will be angular displacement in the joints. The angular spring will produce a torque proportional to the angle, to force the joint back to its rest position. In the model, all the computation is localized. As a result, we should translate the absolute coordinates to those in the parent coordinates, and extract the the spacial angular displacement from 3D position coordinates, as shown in Figure 7.

$$\Theta^2_{rel} = (P^3 - P^2) \times (P^2 - P^1) \quad (9)$$

$$\alpha^2_{rel} = \alpha^2 - \alpha^1 \quad (10)$$

$$\Phi = \begin{matrix} \Theta_{rel} \\ \alpha_{rel} \end{matrix} \quad (11)$$

As can be seen, Θ_{rel} is a vector quantity, indicating the rotation angels in two orientations, and α_{rel} is a scalar quantity that can be

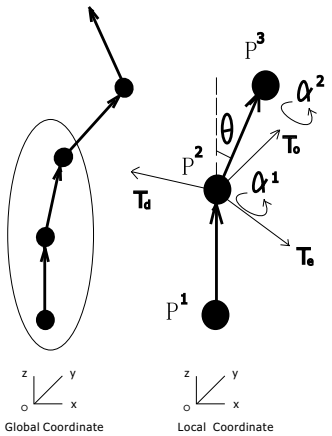


Figure 7: From spring model to elastic joint model

thought as the torsion angel around the hinge. And Φ is the final angular vector under computation in this system.

First of all, we apply the unconstrained rigid body motion equations to analyze the dynamics of each joint.

$$\dot{\phi}(t) = \omega(t) \tag{12}$$

$$\dot{L}(t) = \tau(t) \tag{13}$$

In the above equation (13), $\omega(t)$ is the angular velocity, $L(t) = m\omega(t)$ is the angular momentum, and m is the rotation inertia of bone of the hinge.

$\tau(t) = \tau_e(t) + \tau_d(t) + \tau_o(t)$ is the total torque, the sum of elastic torque , damping torque and external torque.

$\tau_e(t) = -k\,\phi(t)$ is the elastic torque, which according to *Hooke*'s principle, is proportional to the negative angular displacement, and k is the elastic coefficient.

$\tau_d(t) = -c\,\dot{\phi}(t)$ is the first-order damping, representing friction or the energy loss of the system, and c is the damping coefficient.

$\tau_o(t)$ is the external torque, which is the only motivation of the whole system.

From the above analysis, the whole model can be deduced as a second-order linear dynamic system. The equation for the movement of every joint is as follows.

$$m\ddot{\phi}(t) + c\dot{\phi}(t) + k\phi(t) = \tau_o(t) \tag{14}$$

5.1 Simplification

In the practical design, every joint is thought independent on other joints, including the immediate parent-joint and child-joints in the skeleton. That is, we assume that when computing a joint, other joints are fixed in their present positions, and only the joint currently under consideration is active and can move freely. Therefore, there is no coupling between joints, and all the coefficient matrixes in the equations are simplified to cross matrixes as follows:

$$\begin{bmatrix} m_1 & \dots & 0 \\ \vdots & \ddots & 0 \\ 0 & \dots & m_N \end{bmatrix} \begin{bmatrix} \ddot{\phi}_1(t) \\ \vdots \\ \ddot{\phi}_N(t) \end{bmatrix} + \begin{bmatrix} c_1 & \dots & 0 \\ \vdots & \ddots & 0 \\ 0 & \dots & c_N \end{bmatrix}$$

$$\begin{bmatrix} \dot{\phi}_1(t) \\ \vdots \\ \dot{\phi}_N(t) \end{bmatrix} + \begin{bmatrix} k_1 & \dots & 0 \\ \vdots & \ddots & 0 \\ 0 & \dots & k_N \end{bmatrix} \begin{bmatrix} \phi_1(t) \\ \vdots \\ \phi_N(t) \end{bmatrix} = \begin{bmatrix} \tau_{o1}(t) \\ \vdots \\ \tau_{oN}(t) \end{bmatrix}$$

or briefly:

$$M\,\ddot{\Phi}(t) + C\,\dot{\Phi}(t) + K\,\Phi(t) = T_o(t) \tag{15}$$

Here M is the inertial matrix, C is the linear damping matrix, and K is the flexibility parameter matrix. $\Phi(t)$ is the angular displacement vector, and $T_o(t)$ is external torque vector.

Such presupposition is not accurate in theory. But because the motion range of every joint, relatively with neighbor joints, is small, the velocity of the whole skeleton is not very fast, and above all, the simplification highly decrease the computation cost, we ultimately achieve satisfying smooth animations in real time.

5.2 Physical Parameters

As the animation is based on angular springs, and the ultimate motion source is the external torque produced from the displacement in interaction. It is also the only impetus of the whole system. The actor can smoothly change its postures as a response of the second-order dynamic system with different external motivations.

On the other hand, all the physical parameters in the formula can be adjusted by animators in real time, and the system will demonstrate different styles of choreography accordingly. For example, if we increase the inertia coefficient, the actor looks clumy; while deceasing it, it is brisk. The greater the damp coefficient is, the faster the system loses its energy; in contrast, the less it is, crazier the character appears is. When stiff coefficient is enhanced, the motion seems rigid, and when it is cut down, the animation is softer.

6 Results

All the animations are achieved on a 1.4G PC with Intel Pentuim 4 CPU, 128M RAM. The interactive control is in real time and the final dancing is smooth. Many objects of various topologies can be animated in this system. For example, *Human* is a 3D model with 754 faces and 39 joints, while *Lily* is with 9248 faces and 29 joint. Although there are actually more than 100 joints in a human body, in this system, the number of joints is enough, and more joints also lead to more designing and computing cost.

Some final dancing animations are displayed in Figure 8 and Figure 9.

Furthermore as discussed in the previous sections, although there are spring model and elastic joint model in the system, the two models operate at different time. However, when the two models work together, this system can simulate some natural movements, as Feng et al. [Feng et al. 1998] do. Animators can introduce wind into the system, which causes displacement in some joints, just as some joints are selected and dragged interactively through mouse motions. The spring model is responsible for the effects of wind on the whole skeleton. At the same time, the elastic joint model will

be activated by the angular displacement, and produce a force compelling the skeleton to its initial equilibrium. As the elastic joint model is a second-order dynamic system, the simulation is based on physics and the results are realistic. But this kind of simulation is not fine enough, especially for the objects such as flowers that have complicated and detailed petals. However it is good at simulating the models with clear skeletons, such as willows, grass, stems of flowers and branches of trees. The simulation result is exhibited in Figure 10.

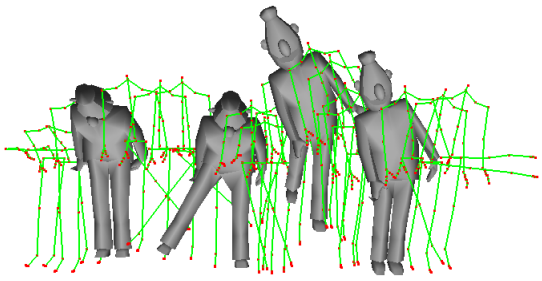


Figure 8: Dancing animation sequences of *Human*

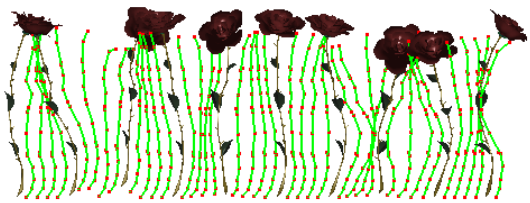


Figure 9: Dancing animation sequences of *Rose*

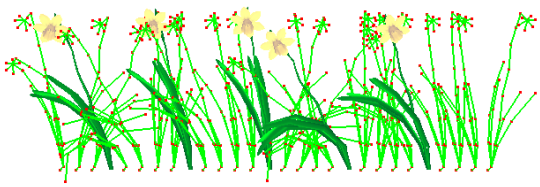


Figure 10: Motion sequences of *Lily* in storm

7 Conclusions and Future Work

In this paper, a system has been described for creating and manipulating articulated character dancing animation. Animators can produce choreography of their own personality, and even design dancing to music to express their understanding of certain music freely. What is more, this system can achieve diverse types of dancing animations by changing the physical parameters, including stiffness, damping, and inertial coefficients of the joints.

Spring model is widely adopted to animate deformable objects, and "it is easy to implement, is highly parallelizable, and involves few computations." [Meyer et al. 2001] We apply spring model to interactively control and design postures, and the results are equally effective and satisfying. For the elastic joint model, we make some assumptions to simplify the computation of LDS and achieve the real-time animation while keeping motions realistic.

There are also many problems remaining in this system. The spring model is universal for objects of various topologies, but we also want to add some constrains to the articulated models to grant them their natural characteristics, as M. Naganand and S.Ferguson [Naganand and Ferguson 1998] have done. For example, all the human-like models have the characters of human, and exhibit the required physical likeness that is necessary for a good human animation. Then it will be much easier and friendlier for animators to create animations through this system.

Spring model and elastic joint model are applied to the system to produce animation in real time, but the two models are generally local models. Consequently, we are also desirous to add some other physical models into our system to realize controlling the whole skeleton globally. A case in point is the introduction of *Mode Shape* in mechanism, which will greatly facilitate the animation production and simplify the computation to improve the real-time performance.

Furthermore, dancing is the most impressive when performed to music. Animators can create dancing animations to music, but it may take them much time with our current system. For animators must listen to the certain music many times to extract the rhythm and pattern by themselves, and then design the postures accordingly and particularly. How to aid animators in creating choreography to music with computers is a promising and arduous task for our future improvement. It involves in extracting rhythm, patterns and even feelings of music by computer automatically, and it is also a significative exploration in music visualization [Kim and Hwang 1999][Smith and Williams 1997]. We plan to use *MIDI* (Musical Instrument Digital Interface) as the input music, for its unique advantages of containing the control information for a musical composition.

Acknowledgment

We would like to acknowledge Jun Teng for his profound suggestions and technology instruction in implementing the system, and Yujiu Yang for his constructive advices in finishing the paper. The flower and human models shown in our paper is downloaded from the website of Toucan Corporation. (URL: http://www.toucan.co.jp)

References

CAPELL, S., GREEN, S., CURLESS, B., DUCHAMP, T., AND POPOVI, Z. 2002. Interactive skeleton-driven dynamic deformations. *ACM Trans. Gr. 21*, 3, 586–593. (Proc. SIGGRAPH'02).

DONTCHEVA, M., YNGVE, G. D., AND POPOVIC, Z. 2003. Layered acting for character animation. *ACM Trans. Gr. 22*, 3, 409–416.

FENG, J.-H., CHEN, Y.-Y., YAN, T., AND WU, E.-H. 1998. Going with wind - physically based animation of trees. *Chinese Journal of Computers 21*, 9, 769–773. (Chinese).

KIM, G., AND HWANG, J. 1999. Musical motion: A medium for uniting visualization and control of music in the virtual environment. In *Proc. of VSMM99*.

KIM, T., PARK, S. I., AND SHIN, S. Y. 2003. Rhythmic-motion synthesis based on motion-beat analysis. *ACM Trans. Gr. 22*, 3, 392–401. (Proc. SIGGRAPH'03).

LI, Y., WANG, T., AND SHUM, H.-Y. 2002. Motion texture: A two-level statistical model for character motion synthesis. *ACM Trans. Gr. 21*, 3, 465–475. (Proc. SIGGRAPH'02).

LU, R.-Q., AND ZHANG, S.-M. 2002. From story to animationfull life cycle computer aided animation generation. *Acta Automatica Sinica 28*, 5, 321–348. (Chinese).

MAGNENAT-THALMANN, N., AND THALMANN, D. 1990. *Computer Animation: Theory and Practice*. Springer-Verlag.

MCMILLAN, S., ORIN, D. E., AND MCGHEE, R. B. 1995. Efficient dynamic simulation of an underwater vehicle with a robotic manipulator. 1194–1206.

MEYER, M., DEBUNNE, G., DESBRUN, M., AND BARR, A. H. 2001. Interactive animation of cloth-like objects in virtual reality. *The Journal of Visualization and Computer Animation 12*, 1, 1–12.

NAGANAND, M., AND FERGUSON, S. 1998. Specialised constraints for an inverse kinematics animation system applied to articulated figures. In *Proceedings of Eurographics'98*, 215–223.

NAKATA, T. 2002. Generation of whole-body expressive movement based on somatical theories. In *Proceedings of the second international workshop on Epigemetic Robotics*, 105–114.

OORE, S., TERZOPOULOS, D., AND HINTO, G. 2002. Local physical models for interactive character animation. 337–346. (Proc. Eurographics'02).

POPOVIC, J., SEITZ, S. M., ERDMANN, M., POPOVIC, Z., AND WITKIN, A. 2000. Interactive manipulation of rigid body simulations. In *Proceedings of SIGGRAPH 2000*, 209–218.

PROVOT, X. 1995. Deformation constraints in a mass-spring model to describe rigid cloth behavior. In *Graphics Interface*, 147–154.

SMITH, S., AND WILLIAMS, G. 1997. A visualization of music. In *Proc. Conference on Visualization'97*.

TU, X.-Y., CHEN, H.-J., AND TU, X.-Y. 2000. Methods of artificial life - modelling and production of "artificial fish". In *Software World*, 51–53.

TURNER, R., AND GOBBETTI, E. 1998. Interactive construction and animation of layered elastically deformable characters. 135–152.