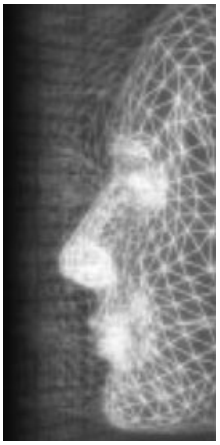
COMPUTER ANIMATION AND VIRTUAL WORLDS
Comp. Anim. Virtual Worlds 2006; **17**: 551–564
Published online 24 July 2006 in Wiley InterScience
(www.interscience.wiley.com) DOI: 10.1002/cav.154

Procedural rhythmic character animation: an interactive Chinese lion dance†



By Tsai-Yen Li and Je-Ren Chen*

The creation of a stylistic animation through the use of high-level controls has always been a design goal for computer animation software. In this paper, we propose a procedural animation system, called rhythmic character animation playacting (RhyCAP), which allows a designer to interactively direct animated characters by adjusting rhythmic parameters such as tempo, exaggeration, and timing. The motions thus generated reflect the intention of the director and also adapt to environmental obstacle constraints. We use a sequence of martial-art steps in the performance of a Chinese lion dance to illustrate the effectiveness of the system. The animation is generated by composition of common motion elements, concisely represented in an action graph. We have implemented an animation control program that allows Chinese lion dance to be choreographed interactively. This authoring tool also serves as a useful means for preserving this part of world cultural heritage. Copyright © 2006 John Wiley & Sons, Ltd.

Received: 20 March 2006; Revised: 22 May 2006; Accepted: 29 May 2006

KEY WORDS: rhythmic character animation; motion planning; Chinese lion dance; high-level animation control

Introduction

Despite the recent progress in the field of computer animation, synthesizing an expressive character animation through the use of high-level user controls remains a challenge. The key to creating expressive styles of motion is held within the minds of sophisticated animators. Traditional animators create convincing character motions by specifying how actions are spaced according to the personality, characteristics, and mannerisms of a character with the physics of movement as well as the aesthetic aspect of the animation. From the lowest level pose-to-pose construction to the overlapping action refinement within a character movement, animators express strong or weak motion controls by varying temporal distribution of motion elements. From a director's point of view, developing a character motion is very similar to the composition of music or the choreography of a dance in that all three involve rhythmic progression. However, the realization of this understanding in an expressive animation is a labor-intensive part of an animation production.

An alternative way to acquire character motions is through the use of motion capture techniques. Modern motion capture facilities are able to record the motions of the human figure and re-target them to other characters in real time. However, once an appropriate motion clip has been captured, it is difficult to modify the motion for reuse without referring to its original semantic meaning in regard to aspects such as the satisfaction of kinematic or geometric constraints. Thus, the automatic generation of character motions from existing data through higher-level user commands remains difficult.

In this work, rather than interpolating data from examples of motion, we consider a procedural method for generating character motions. Our aim is to construct the individual components of the behavior of an actor into elementary actions as understood according to the principles of traditional animation. In providing parameters for the subjective factors of a motion via stylized high-level animation controls, we aim to

*Correspondence to: Tsai-Yen Li, National Chengchi University, Computer Science Department, 64, Sec. 2, Zhi-Nan Rd., Taipei, Taiwan, Province of China. E-mail: li@nccu.edu.tw

†This study is supported in part by National Science Council, R.O.C. under contract no. NSC 94-2213-E-004-006. The authors wish to thank the editor and anonymous reviewers for their insightful comments on an earlier version of the manuscript.

develop an animation system that can create a rhythmic character animation from high-level user commands.

Our aim will be illustrated by a character animation of a Chinese lion dance.[1] Chinese lion dances are a traditional Chinese art performance performed at holiday celebrations such as the Chinese Lunar New Year. A typical lion costume consists of a large lion head operated by one dancer and a symbolic body costume operated by another (see Figure 1). The dancers are usually accompanied by several drummers. It is a great challenge to perform a lion dance since it requires intensive practice over a long time to produce seamless coordination between the dancers. Thus, realizing this performing art via the use of computer animation also presents a challenge to animators because of the expectations of the audience in regard to the degree of expressiveness and coordination of movement in the performance of a show.

We have not been able to find systems in the literature that are able to generate this type of animation interactively with high-level user commands. In this work, we have implemented such an interactive system to demonstrate that the goal of high-level controls for rhythmic animations can be achieved for the specific application of a Chinese lion dance. Although the motion repository is specific for the performance of the actions in martial art forms, the general concepts of composing animations from organized motion elements that are generated procedurally with rhythmic control parameters should also be applicable to other types of character animations.

Figure 1. An image of the Chinese lion dance generated by the proposed system.

In short, the technical contributions of this work are threefold. First, we propose an interactive procedural animation system that allows the users to present their animated design through three rhythmic parameters and through high-level commands. Second, with the feature of being easy to use, the animation-authoring tool should be able to help a regular user without trained animation skills to create an expressive animation through the interactive user interface. Third, we propose a concise representation of motion controls, called action graph, for all procedural compositions of elementary actions, and we have used the Chinese lion dance as an example to illustrate this idea. Nevertheless, as in all procedural animation systems, realizing different types of motions will require different procedurals to be designed specifically and the animation principles will still be applied in general. In addition to the technical contributions, our authoring tool also serves as a means to help in preserving the cultural heritage of the Chinese lion dance.

We organize the rest of the paper as follows: in Related Work Section, we review the approaches to animation control found in previous work. In System Architecture Section, we present the system architecture and approach that we have used in this work. In Chinese Lion Dance Section, we use a Chinese lion dance as an example to illustrate our two-level procedural animation control. In Experimental Results Section, we outline our implementation with examples. In Discussions and Conclusions Section, we discuss the limitations and future work for the system.

Related Work

Previous research pertaining to human figure animation can be divided into three major categories: *simulation-based*, *example-based*, and *procedural*.

Simulation-Based Human Motion

Generating convincing character locomotion is a conventional problem in computer animation. Many recent efforts in synthesizing human motions with user controls address the issues of physics/dynamics-based character animation. Liu and Popovic[2] generated character motions with dynamic constraints by the detection of phrases in the input motion with the use of a motion-sketching tool. Hodgins *et al.*[3] simulated several forms of physics-based representations of athletics with

DOI: 10.1002/cav

higher-level behaviors from global control. The virtual stuntman in Ref. [4] is able to perform various repertoires by integrating two-level state-space motor controllers. Sims, with an optimized control algorithm, generated creatures with high degrees of freedom (DOF) under the control of certain objective functions.[5] Hsieh and Luciani[6] used a simplified dynamic model to assist computer choreography utilizing a number of dance verbs, which is similar to the idea of actions used in this paper. Due to the high DOF required of a character, simulation-based approaches are expensive in either temporal or spatial complexities. In addition, the biomechanical model used to describe the human is still not precise enough to portray convincing character motions. Consequently, post-modifications to the generated animation are usually required. In addition, it is not easy to apply such techniques to real-time animation systems in general.

Example-Based Human Motion

In contrast, research efforts in relation to extracting motion style are more recent. Brand and Hertzmann used style machines[7] to learn stylistic motion patterns from motion capture data, while Bregler *et al*.[8] captured the motion styles from traditional cartoons and retargeted it to new characters. Kim *et al*.[9] synthesized new motions from example motions while preserving rhythmic patterns. In contrast to the machine learning and pattern recognition techniques, Rose *et al*.[10] created character motions from existing animation segments by interpolating 'verbs' motion with multi-dimensional 'adverbs'. Isla *et al*.[11] also modeled a motor system synthesizing character animation within a one-dimensional adverb space. Urtasun *et al*.[12] extracted motion style from captured data utilizing the principle component analysis method and synthesize new motions from a motion library. However, rather than expressing an abstract emotional motion style directly or extracting it from existing data, our motion-style presentation is defined with three concrete rhythmic parameters that are more intuitive to end-users.

Procedural Character Modeling and Animation

Creating feasible key poses and then interpolating them is an efficient way to generate plausible character animations. If the key poses and interpolations are defined in an algorithmic manner with appropriate parameterization, we call it *procedural animation*. Due to the complexity of describing the motions produced in such a manner, most of the research in this area has been on the generation of the most common locomotion: walking. Bruderlin and Calvert[13] developed a system that allowed the user to procedurally specify most parameters of locomotion and arm swings through the use of a graphical user interface. An interactive control hierarchy was adapted to produce a variety of personalized human walks. Girard and Maciejewski[14] used a mix of kinematic and dynamic methods to simulate human locomotion. The motion of the lower body was defined in a procedural manner while the motion of the whole body was computed with simplified dynamics. Sun and Metaxas[15] automated gait generation procedurally by matching limbs with an elevation angle dataset. Van de Panne[16] created a whole-body animation by formulating it as a global optimization problem on the center-of-mass trajectory based on the placement and timing of footprints. Li *et al*.[17] described a method to plan and generate the lower-body motion of a human procedurally. In our work, the key poses are identified specifically for the performance of a lion dance with the traditional animation principles.

Above the level of locomotion in simulation for animated characters, research such as Refs. [18,19] proposed several hierarchical behavior models for human or life-like characters. Another similar approach by Blumberg and Galyean[20] decomposed a motion behavior into layers: motivation, task, and motor. In our work, the animation control is constructed by a two-level control framework: a playacting level and a pose-to-pose level as described in the next section.

Since the movement of the human body is the synergy of over 600 muscles and 200 bones, an articulated human figure will contain more than a hundred DOFs that need to be controlled. However, most research uses a simplified kinematics model to produce the motions. Research such as that by IKAN[21] and Koga *et al*.[22] sought to find a suitable inverse kinematics (IK) algorithm to reduce the complexity of manipulation of human limbs in the accurate performance of typical human tasks. Rose *et al*.[23] interpolated motions with an artist-directed IK algorithm in a multi-dimensional motion space. Our method of generating motion procedurally is also based on a simplified kinematic model. The postures of the characters are created by a rapid IK solver similar to the one used in Refs. [21,22].

DOI: 10.1002/cav

System Architecture

The proposed rhythmic character animation playacting (RhyCAP) system is based on a two-level animation control framework consisting of four main modules as shown in Figure 2. In the controlling flow, the director module first derives control of stylized animation from its upstream and then manages component controls with appropriate rhythmic parameters sent to the downstream. In the following subsections we will examine the animation controls at both levels.

Playacting Level

We assume that the proposed system is initialized with several human characters as the actors and a performing space in a 3D virtual environment. In such a stylized character animation system, a user should be able to direct the animation by specifying a sequence of desired motion repertoires such as the human tasks or maneuvers and modulating the motion style by varying its rhythm. Therefore, the system begins its control pipeline with a Playacting Console interface accepting play commands from the pre-choreographed character repertoires. The user interface has both a script mode for a pre-defined listing of plays and an interactive mode for run-time user improvisation. During run-time interpretation, the playacting console allows the user to vary the rhythmic parameters of the repertoire playacting and specify the target toward which the character should move.

Once a play command is issued from the playacting console, the *Director* module is invoked to coordinate the actors in the scene. Since the characters should follow human behavior rules while performing the requested actions, the director has to be able to conduct global synchronization and centralized control over the actors. In order to generate character motions that can reach the user-specified goal while avoiding collisions with environmental obstacles, we assume that the director will acquire a collision-free path generated by the *Path Planner* module. The director can then conduct *behavior controls* (BCs), such as steering and following, over the actors along this path. In addition, the director is also responsible for specifying the rhythmic motion style for the play commands sent to each dancer.

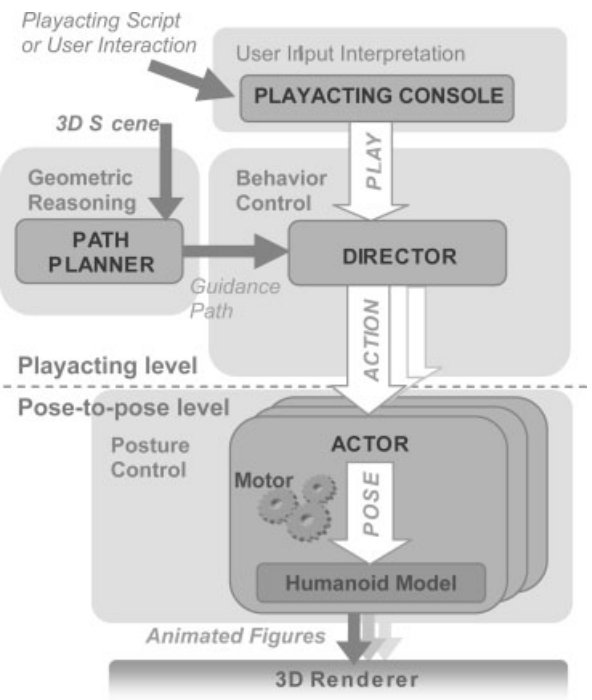


Figure 2. The schematic view of the rhythmic character animation playacting (RhyCAP) system.

Pose-to-Pose Level

At the pose-to-pose level, the Actor module behaves as a local control system with a humanoid model and its motor drivers. The objective of the actor module is to implement a rhythmic posture controller (PC) for a given action. Each controller is in charge of generating a sequence of frames for the desired action and then piping them to the real-time 3D renderer module to generate the final character animation.

The objective of the PC is to generate plausible human-like motions for the articulated characters. Unlike the PD controllers in dynamics-based approaches or motion synthesis from extracted features in example-based approaches, RhyCAP generates the motions for an action in a procedural manner. The animation procedure is implemented as a sequence of pose compositions similar to the pose-to-pose development process in traditional animation production. Although the design of animation procedures can be example dependent, they are efficient because of light computation requirements. The procedure could also be made effective based on the knowledge of animators from long-standing experience. For example, all the actions may be decomposed into several common motion components that are controlled by a limited

DOI: 10.1002/cav

number of motion controllers. In addition, a procedural approach will also facilitate the control of rhythmic motion parameters as described below.

Rhythmic Motion Styling

The term 'rhythm' is defined as the aspect of music comprising all the elements (as accent, meter, and tempo) that relate to forward movement.[24] Changing the rhythm of some music affects the strong and weak presentation as well as the speed of the melody. One can draw an analogy between the rhythm of music and character motions since similar actions may consist of the same progression of poses but with different tempo, exaggeration, and timing for each pose. Varying the rhythm of an action yields different motion styles. The RhyCAP abstracts and parameterizes three major rhythmic features (tempo, exaggeration, and timing) for the control of a stylized character motion.

In the RhyCAP system, the 'tempo' parameter determines the duration of each action. Note that since RhyCAP treats different forms as actions that consist of different arrangements of key pose sequences, changing the tempo of the controller does not change an action from one form into another. For example, speeding up a walk action dose not result in a run action since a fast walk and a slow run are both meaningful in the presentation of the character animation.

When presenting a character motion, animators usually exaggerate some constructing poses in order to achieve a special emphasis on a certain action. In RhyCAP, the 'exaggeration' parameter indicates how close these key poses are to the extremes of the limbs or the body. For instance, raising one's hand with a fully exaggerated 'raise hand' pose extends the hand to the extreme.

Regulating the timing of a rhythm is subtle but results in a significant effect on both visual and aural senses. The speed of an action, that is, timing, is said to give both physical and emotional meaning to movement.[25] When presenting a character motion, animators usually allocate appropriate amounts of time for the anticipation of an action, for the action itself, and for the reaction to the action, respectively. For example, a drown-out preparation may imply a sudden movement with a slow heavy damping to terminate the action. Relating these principles to a jump action, a deep sink introduces a swift leap and then some extra time is induced to indicate the necessity to keep steady after landing. RhyCAP achieves these effects with a single 'timing' parameter that is used to rearrange the durations of these phases in an action.

These three rhythm factors depict a motion-style space as illustrated in Figure 3. The speed of an action is matched with the global tempo in the depth axis. The vertical axis represents the exaggeration of the key poses, and the horizontal axis ranges the timing from fluent to fully anticipated. In general, normal human motions use an energy-minimized style as at the lower-left 'moderate' corner. At the same time, the action in cartoons emphasizing character motions is closer to the upper-right 'accentuated' area. The term 'rhythmic motion styling (RMS)' in this paper means specifying the location of a specific style in the rhythmic motion-style space via the three rhythmic parameters.

Chinese Lion Dance

During a lion dance, the dancers enact a series of rehearsed dance forms by following the rhythm and tempo of the drumbeats. The dancers and drummers may revise the progress of the performance or place additional emphasis on some parts or steps according to the atmosphere of the scene or the reaction of the audience. Usually, the performance ends with a sequence of steps called 'Chai Chin' with the lion moves out to props representing food. The objective of the RhyCAP system is to output a stylized, collision-free lion dance animation according to interactive user inputs.

Character Model

Each of our humanoid lion dancers is modeled as a hierarchical linkage with a tree structure of nodes connected by arcs. We simplify the complex anthropomorphic structure into 15 major nodes as shown in

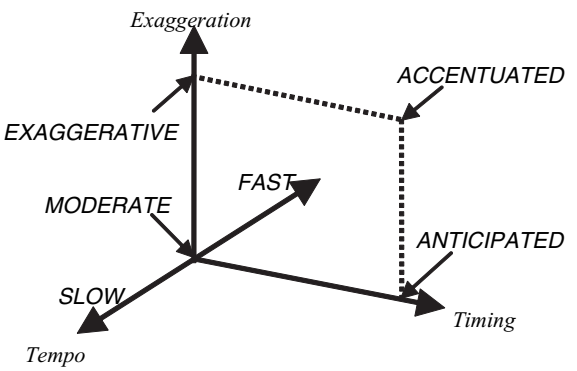


Figure 3. The rhythmic motion-style space.

DOI: 10.1002/cav

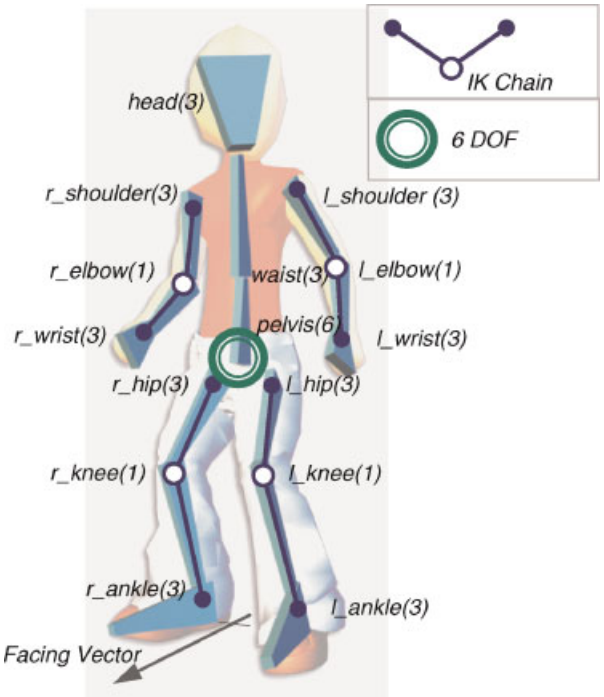


Figure 4. Humanoid kinematics model (Each joint's DOF is shown in the parenthesis).

Figure 4. Each node of the simplified kinematic model here represents a crucial joint for animating human figures in a lion dance. There are a total of 40 DOFs (12 in the torso and head, 28 in the limbs) in the model of the lion dancer.

Generally speaking, a lion dance performance mainly consists of martial actions that require certain parts of the body to reach specific positions with distinct postures. We first decoupled our articulated humanoid into two portions, torso and limbs, in order to perform the poses. In RhyCAP, an action is performed by specifying the end points of the limbs and the root of the torso (pelvis) in the kinematic hierarchy of the humanoid. The pelvis has six DOF while the arms and the legs, each rigged with 7-DOF IK chains, allow six DOF at their end-effectors. We also designed a generic IK solver similar to Ref. [21] so as to specify the postures in Cartesian space instead of in the less-intuitive joint space. During run-time action, we also maintained a facing vector on the top of the hierarchy to represent the direction that the dancer is heading toward, and this vector may not be aligned with the orientation of the pelvis.

There are nine primary stances in a Chinese lion dance.[1] We chose five simpler stances, as shown in Figure 5, which all incorporate an action of stepping on the ground without jumping, rolling, or tumbling. We also chose five basic types of gesture for manipulation of the lion head. We then choreographed these stances and gestures into a specific sequence in the lion dance. The stances (AS) represent the posture of the martial stances by specifying the positions and rotations of the both feet and the pelvis. The gestures (AG) are settled relatively to the lower body. We treated all AGs in a both-hand mode since the head dancer operates the lion head with both hands at all times.

Play Script

The teaching of the steps for the various dances is not well documented. The director of a play teaches the lion dancers by describing the actions in terms of the various martial forms. In RhyCAP, we use the play command to choreograph the stances and gestures. A play is a

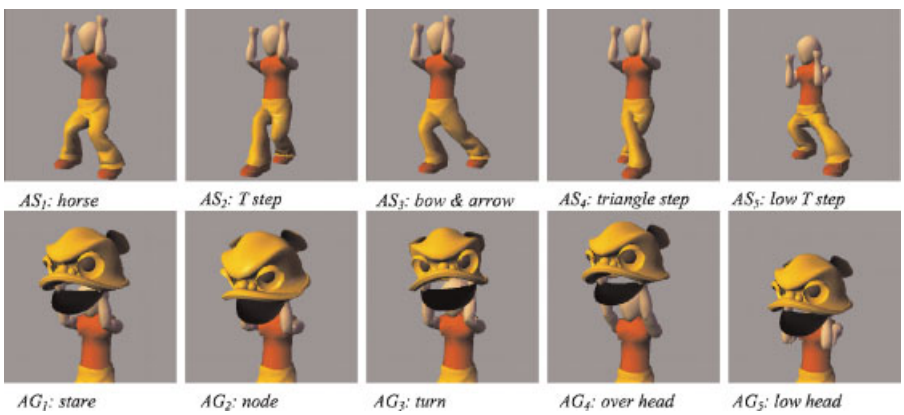


Figure 5. Available stances and gestures in the RhyCAP system.

DOI: 10.1002/cav

```
[PLIST]
1
[PLAY]
4
[ACTION]
4164 ---- ---- ---- 4264 ---- ---- ---- 3164 ---- ---- ---- 1294 ---- ---- ----
---- 2264 ---- ---- ---- 2164 ---- ---- ---- 1064 ---- ---- ---- 3094 ---- ----
[BEAT]
B-BB B-B- B-B- M---
M
M-B- B-B- B-B- M---
M
```

Figure 6. An example of the play command consisting of four actions.

playacting script that defines not only the martial form sequence but also the corresponding drumbeats.

Figure 6 shows an example play with four beats as defined by our system. The minimal unit of the time in the play is the sixteenth note, that is, a quarter of a beat. The martial forms and drum beat patterns are defined in the Action and Beat sections, respectively. The two tracks in the action section represent the stance and gesture actions, respectively. We use a four-digit number to represent an action. The first digit stands for the destined martial form as referred to in Figure 5. The second digit denotes the stance leg (the fixed leg on the ground during a stepping action) for the stance actions (one for the right and two for the left) or the facing direction for the gesture actions. The third digit represents the default exaggeration value for the action, and the last digit represents the total number of the sixteenth notes that the action takes. For example, the set of digits '4164' represents a 'triangle step' action with a right stance leg and an exaggeration value of 0.6 (normalized between 0.0 and 1.0) for the duration of one beat.

The drum beat section has four tracks: a drum track, a strike track (striking of the drum), a gong track, and a cymbal track. 'B' denotes percussion, and 'M' denotes mute. All the '-' symbols in each track stand for the period of the sustention of an action or a sound.

Real-Time Path Planning and Following

In RhyCAP, we implemented two BC for the lion dancers: the *leading* behavior for the head dancer and the *following* behavior for the tail dancer. Both controls are based on a collision-free path generated by a path planner in real time for the lion to reach the user-specified goal.

The path planner that we have adopted utilizes a potential-field-based algorithm to search for a feasible path.[26] This technique has been shown to be efficient in low-dimensional search spaces.[27] Since the generated path is only a reference path for the lion to follow, the robot (the subject for planning) in this planner is modeled as a circle with an appropriate radius that can enclose the geometry of a dancer. Due to the symmetry of a circle, the planning problem can be simplified into a two-dimensional search problem by an expansion of the boundary of the obstacles in the environment.

As in most motion planners, the best-first planning (BFP) process consists of two steps: *preprocessing* and *query*. In the preprocessing step, we use the NF1 navigation function proposed in Ref. [27] to fill the potential field. In the query step, the planner uses the potential field as a heuristic to search for a feasible path in a best-first fashion on a uniform grid. The planner is called whenever a new goal is specified at run time and a feasible path is returned if one exists for the given grid resolution. A path is comprised of a sequence of configurations in the following form: $\tau = q_1, q_2, q_3, \ldots, q_n$ where $q_i = (x_i, z_i)$.

The leading behavior of the head dancer is achieved by maintaining a facing vector ($\mathbf{F}^{\mathrm{H}}$) along the tangential direction of a reference point along the generated path. Since we do not incorporate jump actions in the system at this time, the lion always moves one step at a time. We use the facing vector to determine a new pelvis location $\left(P'^{\mathrm{H}}_{\mathrm{pelvis}}\right)$ which is then used to determine the location of the stepping leg.

The motions of the tail dancer are very similar to that of head dancer except that the tail dancer is constrained by the distance between the head dancer and himself. This BC consists of three steps: determining the facing vector ($\mathbf{F}^{\mathrm{T}}$), a new pelvis location $\left(P'^{\mathrm{T}}_{\mathrm{pelvis}}\right)$, and a new free leg location $\left(P'^{\mathrm{T}}_{\mathrm{free}}\right)$, as illustrated in Figure 7. First,

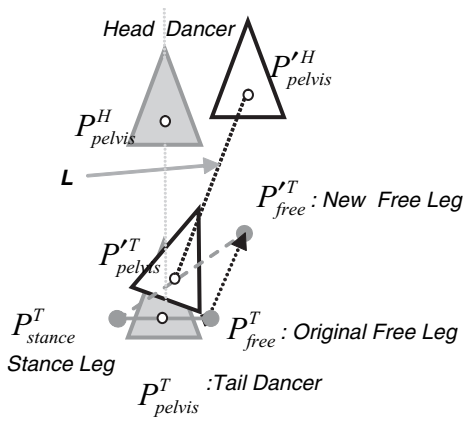


Figure 7. The following behavior of the tail dancer.

DOI: 10.1002/cav

we make the tail dancer face the new pelvis location of the head dancer: $\mathbf{F}^{\mathrm{T}} = P'^{\mathrm{H}}_{\mathrm{pelvis}} - P^{\mathrm{T}}_{\mathrm{pelvis}}$. In line with this direction, we place the pelvis of the tail dancer by a fixed distance L behind the head. That is,

$$P'^{\mathrm{T}}_{\mathrm{pelvis}} = P'^{\mathrm{H}}_{\mathrm{pelvis}} - \frac{\mathbf{F}^{\mathrm{T}}}{|\mathbf{F}^{\mathrm{T}}|}L$$

Since we prefer to keep the pelvis of the tail dancer in the middle of two legs for balance, the new free leg location can then be computed with the following formula:

$$P'^{\mathrm{T}}_{\mathrm{free}} = P^{\mathrm{T}}_{\mathrm{stance}} + 2\left(P'^{\mathrm{T}}_{\mathrm{pelvis}} - P^{\mathrm{T}}_{\mathrm{stance}}\right).$$

Once the new locations of the pelvis and the free leg for both dancers are determined, we can then manipulate the corresponding end-effectors by the pose controllers with appropriate heights.

Pose controller		Parameter	Direction
Foot posers (PF)			
PF_1	Shift leg	EX, Dir	Horizontal
PF_2	Lift leg	EX	Vertical
PF_3	Lower leg	EX	Vertical
Torso posers (PT)			
PT_1	Shift between	EX, Bias	Horizontal
PT_2	Lift up	EX	Vertical
PT_3	Lower down	EX	Vertical
Hand posers (PH)			
PH_1	Aim	EX, Dir	Polar coord.
PH_2	Push	EX	Polar coord.
PH_3	Pull	EX	Polar coord.
PH_4	Spin	EX	Polar coord.

Table I. List of posers (pose controllers)

Action Graph

Generally speaking, to affect a performance of an action requires the combination of precise key poses with an appropriate transition from the current martial form to the destined martial form (e.g., from a 'Horse' stance to a 'Bow and Arrow' stance). However, the number of transitions between actions becomes greater as the number of actions available increases. For example, for the five stance actions, there are 20 possible transitions that need to be implemented in our system if we want to create these motions procedurally. However, we think that these transitions can be decomposed into a few fundamental motion controllers that are common for all of these different actions. Thus, we can design an action by composing the exact sequence of atomic controllers, called *Posers*. Each poser represents a single action that shifts the limbs or torso horizontally or vertically.

Currently, we have designed nine posers that can be used to compose all the actions that our dancers are able to perform: three for the foot posers (PF), three for the torso posers (PT), and another four controllers for the hand posers (PH) as shown in Table 1. The PFs and PTs are work in coordination in the functioning of the lower-body stances. The PHs are used to compose the gestures for the upper body. Note that since the head dancer operates the lion head in combination with the movement of the lower body, the PHs are moved by the polar coordinate system relative to the local frame of the lower body.

We represent the transitions of these stances by a concise graph called *stance action graph* (SAG) as shown in Figure 8. In SAG, each node represents a torso or PF while an arc between two nodes represents the transition between two sets of actions. The arcs are denoted by (A_{From}, A_{To}), where A_{From} and A_{To} stand for the previous martial forms and destined martial forms, respectively and '*' stands for wildcard. The graph is cyclic, and each action begins and ends at the same node PT_1. The arcs are directed and each of them is associated with a priority. These priorities are used for the nodes with multiple outgoing arcs. When there exists more than one arc that satisfies the triggering condition of a transition, the arc with the highest priority is chosen.

The action graph shares similar ideas as the motion graph proposed in Ref. [28]. The nodes in a motion graph represent key frames, and the arcs denote the motion clips between key frames. For the action graph, the nodes represent motion controllers that generate motion clips, and the arcs represent the list of possible triggering actions. The motion graph is used to search for a synthesized path while the action graph is mainly used to encode animation procedures in a concise way.

We use an example to demonstrate the composition of a transition between two martial forms with SAG. Assume that our dancer is performing a series of 'Triangle Stepping' actions as denoted by (AS_4, AS_4), and we begin at the PT_1 node. Since every stance action starts by the movement of one's center of mass onto the stance leg, PT_1 is invoked to shift the torso onto the stance leg as the first node in this composition. Next, the transition proceeds to PT_2, PF_2, and then to PT_3 as our previous martial form is not an AS_5.

DOI: 10.1002/cav

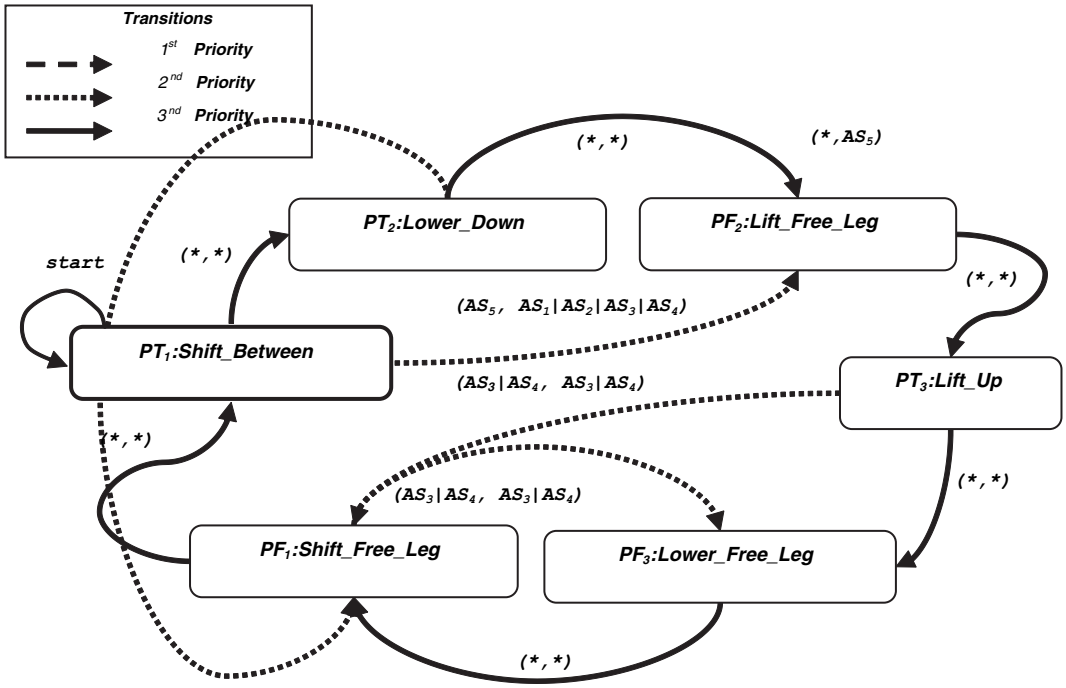


Figure 8. The stance action graph (SAG).

At PT_3, we transit to PF_1 for an additional key pose of the free leg to prevent intersection between the two legs since our previous martial form and target martial form are both in an AS_4 type. For the same reason, the transition proceeds to PF_3 and then PF_1 to affect the lowering down and shifting of the free leg to the destined location. Finally, the torso is moved to the middle of the new stance by PT_1 again for the start of the next stance action. The resulting transition from one AS_4 to another AS_4 is therefore as follows: $PT_1 \rightarrow PT_2 \rightarrow PF_2 \rightarrow PT_3 \rightarrow PF_1 \rightarrow PF_3 \rightarrow PF_1 \rightarrow PT_1$ (as shown in Figure 9). We implemented SAG into procedures so that we are able to compose an appropriate series of posers to generate plausible key poses between any two martial forms in an action in run time. Similarly, procedures for implementing the gesture actions can be concisely represented as a gesture action graph (GAG) consisting of four nodes. With this type of action graph, one can not only express the rules of composition in a concise format but also reduce the complexity of implementing these procedures.

Rhythmic Motion Styling

In RhyCAP, the control of rhythmic motion is achieved by adjusting the tempo, exaggeration, and timing parameters for each action. The tempo parameter is used to determine the duration of an action. In Section 3, we described how the duration of an action is denoted

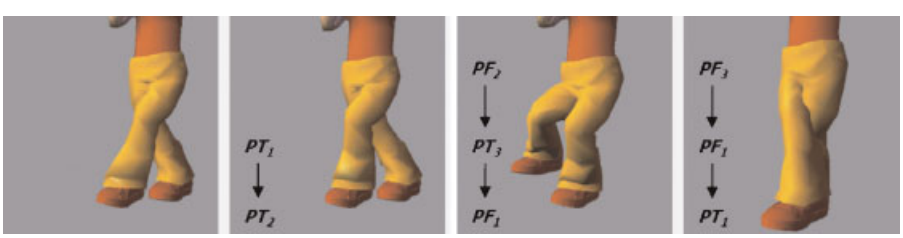


Figure 9. Key poses from AS_4 to AS_4.

DOI: 10.1002/cav

by the number of sixteenth notes (the quarter of a beat) that it takes in a play script. The duration of a sixteenth note $T_{1/16}$ with the tempo of R_{TEMPO} is defined as

$$T_{1/16} = \frac{R_{\text{TEMPO}}}{60(\text{sec})} \times \frac{1}{4}\text{beat}$$

Thus, the duration of an action T_{A} that lasts for a number of B_{A} notes can be computed as follows:

$$T_{\text{A}} = T_{1/16} \times B_{\text{A}}$$

The exaggeration parameter in RhyCAP determines how close the postures are to their extremes. Each poser in an action has a default exaggeration value that ranges from 0.0 to 1.0. In run time, a user can magnify or scale down the degree of exaggeration for each poser by specifying the exaggeration value R_{EX}.

The drawing of unevenly spaced key poses is of essence for the representation of an expressive character movement. Since a properly timed preparation of an action can enable the viewer to better understand a rapid action, a very common technique to express character motions is by partitioning an action into three phases: *anticipation*, *action*, and *reaction*.[29] The anticipation and action phases consist of active motions while the reaction is a passive side effect of the first two phases.

Our current implementation partitions the posers in an action only into the anticipation phase and the action phase. The regulation of the timing between poses is accomplished by arranging the proportion of the two phases inside an action. The passive reaction phase is ignored in the current system and reconsidered as part of post-processing after the motion is exported to an animation package for rendering.

The distribution of the two phases is determined by the timing parameter R_{TIMING}, ranging from 0.0 to 1.0. If we assume that the duration for an action is T_{A}, then its anticipation phase T_{Anti} and action phase T_{Act} may be computed with the following formula: $T_{\text{Anti}} = (R_{\text{TIMING}} \times C + 0.5) \times T_{\text{A}}$, $T_{\text{Act}} = T_{\text{A}} - T_{\text{Anti}}$ where C is a constant empirically set to 0.4 because we define that a moderate action has a 50–50 distribution between these two phases while a fully anticipated action is expected to have a relation of 90–10.

Experimental Results

Implementation

Each of the modules in our RhyCAP system is developed through the use of the C++ programming language. Scene graph management, user event interaction (mouse and keyboard), and real-time rendering, including the lighting and view port control, are developed using the Open Inventor API.[30] Drum beats are produced and synthesized using the Midiio library,[31] a MIDI software interface. A minor issue is that the graphical user interface is substituted with a GLUT[32] binding widget, called SoGLUT Render Area,[33] so that our RhyCAP system is able to work on both the GNU/Linux and Win32 environments. The 3D geometric models including the dancers, the lion head, the costume for the lion body, and the surrounding scene were created using Alias Maya. The animations generated by the system can be exported to Maya for further processing such as the addition of secondary or reactive motions. A sample ray-traced image generated by Maya is depicted in Figure 1.

Examples

Figure 10 shows the sequences of three stance actions (AS_1, AS_3) of a lion dancer with different degrees of the

Figure 10. A stance action from AS_1 to AS_3 with different R_{EX} and R_{TIMING}.

DOI: 10.1002/cav

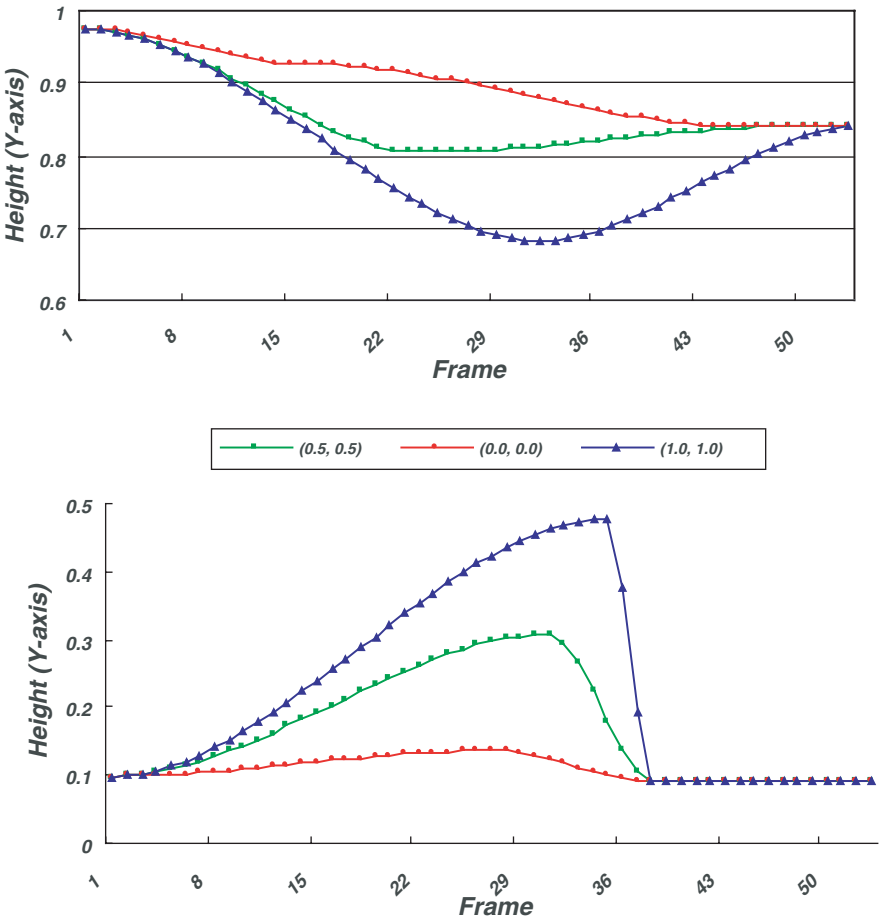


Figure 11. The plots of the heights with respect to the pelvis (left) and left ankle (right) for the three accentuated stance actions from AS_1 to AS_3.

accentuation for each row. The (R_{EX}, R_{TIMING}) values from the bottom row to the top row are (0.0, 0.0), (0.5, 0.5), and (1.0, 1.0), respectively. We also plot the positions of the pelvis and left ankle of these three accentuated actions into curves as shown in Figure 11.

In Figure 12, we show an example path generated by the path planner for the lion to pursue a butterfly. The path is comprised of six configurations for the lion to follow. The red dotted (darker) line traces the pelvis of the head dancer while the green dotted (lighter) line traces the pelvis of the tail dancer. Note that tail dancer follows the head dancer by a dynamic adjustment of the position of his or her pelvis to meet the distance constraint between the two dancers.

In Figure 13, we display snapshots of a lion dance sequence that is enacted by the coordination of the two lion dancers under high-level control through the RhyCAP interface. This lion dance sequence was defined using the play command described in Subsection 4.2. The martial actions and drum beat patterns in each sequence were choreographed by referring to the literal material in Chinese lion dance.[1] Figure 14 shows two snapshots of the graphical user interface of the authoring tool under the user's interactive control.[†] The lion pursues a butterfly (the user-specified 'Chin') along a path that is generated and maintained in real time. The system also allows the user to compose sequences by selecting appropriate actions and setting the rhythmic

[†]A video showing the system in action as well as the post-rendered animations can be found at http://imlab.cs.nccu.edu.tw/~jeren/project/LionDance/

DOI: 10.1002/cav

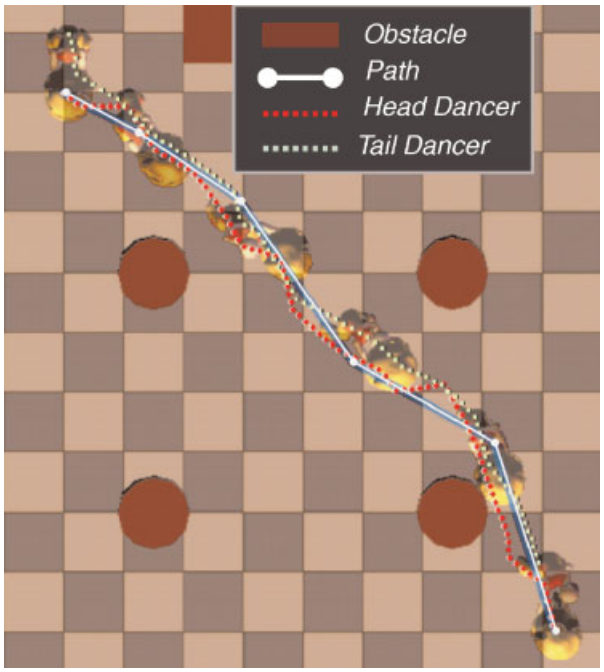


Figure 12. Traces of the lion dancers along a planned path.

parameters interactively through the control panel at the bottom.

Discussions and Conclusions

Discussions

For most procedural animation systems, a main limitation is that the realism of the generated animation, to some degree, depends on the appropriate design of animation procedures, which may not be easily accumulated. Nevertheless, because of their reusability and the ability to adapt to the environment, a well-designed procedure can generate a class of similar and adaptive motions with ease. By allowing a user to specify motion control via rhythmic parameters as proposed in RhyCAP, the expressiveness of the animation can be greatly improved, especially for animations that possess rhythmic motion patterns. However, these animation procedures, although reusable, still require

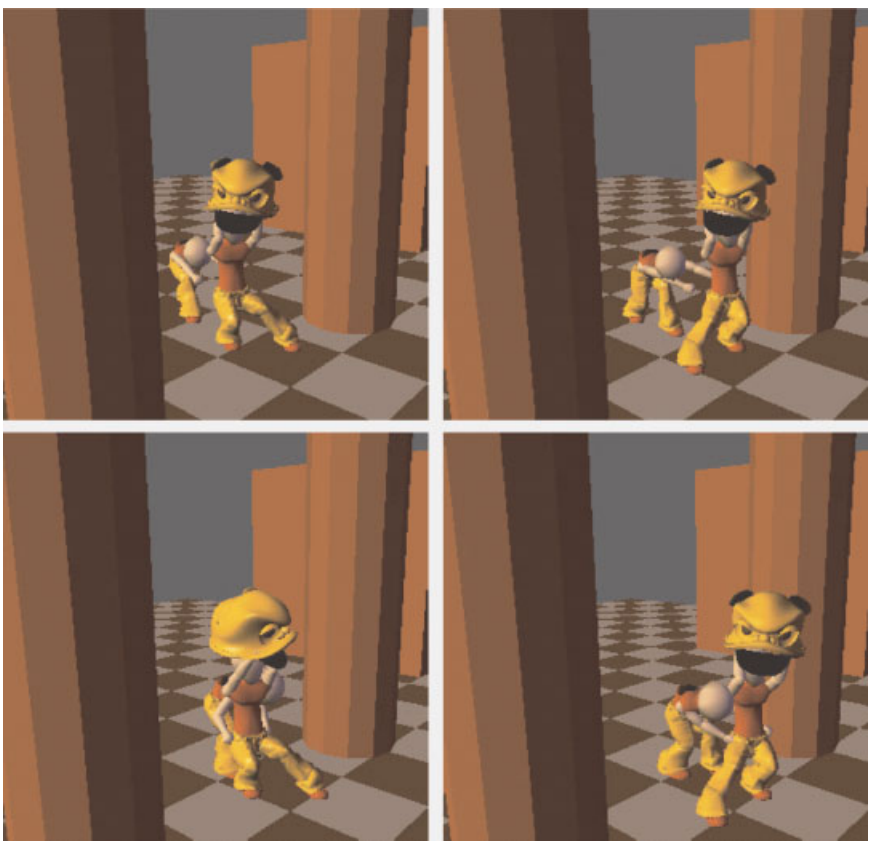


Figure 13. Snapshot of the 'Sneaking Lion' play.

DOI: 10.1002/cav

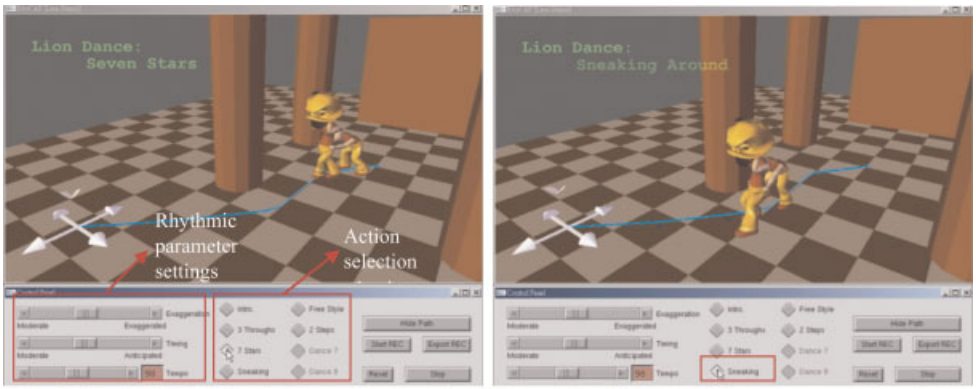


Figure 14. Snapshots of the authoring tool for directing the lion dance animation with various actions and rhythmic parameters. The lion is pursuing a butterfly (in white) along a real-time path.

careful design by animation programmers. Similarly, although the concept of an action graph may be used to represent the steps of composing actions, decomposing these actions into elementary motion controllers so as to construct the action graphs also requires the expertise of good animators. The construction of more procedures for character animation under these principles is an ongoing effort to enrich the repository of motion procedures. A more general framework for the flexible specification of animation procedures for human characters is also under development.

For the example of the lion dance in this research, although the current motion repository is adequate for the composition of a typical lion dance performed on a flat ground, certain further work is required to represent the full range of motions possible in such dances. First, a few, more difficult stance actions need to be modeled to make the repository of motion procedures complete and the generated shows more diverse. For example, in addition to pursuing a symbolic goal to amuse the audience, one popular lion dance includes moves whereby the lion steps on the tops of poles of various heights. Movement along the top of the poles requires careful planning and seamless coordination between the two dancers. Generating such a dance will present another interesting and challenging motion planning problem that deserves further study.

Conclusions

The creation of expressive character animations such as a Chinese lion dance has usually been a difficult and time-consuming task because of the lack of good animation tools to provide high-level controls. In this work, we have proposed a rhythmic animation control system, RhyCAP, that allows the users to create a Chinese lion dance by embedding the tips of traditional animation in the posture control. We also designed two BC modules (leading and following) to direct the dancers during a playact. The control mechanism in RhyCAP is very intuitive for the users wishing to animate dynamic lion dancers even though they may not be professionally trained in traditional animation skills. Thus, the developed system can also be used as an animation prototyping tool for an animation director to choreograph a Chinese lion dance. Although RhyCAP is currently designed specifically for the generation of a Chinese lion dance, the concept of applying rhythmic controls to procedural animations is generic and can be extended to other types of character animations in the future.

References

1. Tseng GC. *The Art of the Lion Dance*. Shu-Cheng: Taipei, 1997.
2. Liu CK, Popovic Z. Synthesis of complex dynamic character motion from simple animations. *In Proceedings of ACM SIGGRAPH*, 2002; **21**(3): 408–416.
3. Hodgins JK, Wooten WL, Brogan DC, O'Brien JF. Animating human athletics. *In Proceedings of ACM SIGGRAPH*, 1995; 71–78.
4. Faloutsos P, Panne MVD, Terzopoulos D. The virtual stuntman: Dynamic characters with a repertoire of autonomous motor skills. *Computers and Graphics* 2001; **25**(6): 933– 953.
5. Sims K. Evolving virtual creatures. *In Proceedings of SIGGRAPH*, 1994; 15–22.

DOI: 10.1002/cav

6. Hsieh CH, Luciani A. Generating dance verbs and assisting computer choreography. *Proceedings of the 13th Annual ACM International Conference on Multimedia*, 2005.
7. Brand ME, Hertzmann A. Style machines. *In Proceedings of ACM SIGGRAPH*, 2000; 183–192.
8. Bregler C, Loeb L, Chuang E, Deshpande H. Turning to the masters: Motion capturing cartoons. *In Proceedings of ACM SIGGRAPH*, 2002; **21**(3): 399–407.
9. Kim TH, Park SI, Shin SY. Rhythmic-motion synthesis based on motion-beat analysis. *In Proceedings of ACM SIGGRAPH*, 2003; **22**(3): 392–401.
10. Rose C, Cohen M, Bodenheimer B. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics and Applications* 1998; **18**(5): 32–40.
11. Isla D, Burke R, Downie M, Blumberg B. A layered brain architecture for synthetic creatures. *Proceedings of 2001 International Joint Conference on Artificial Intelligence (IJCAI2001)*, Seattle, WA, August 2001.
12. Urtasun R, Glardon P, Boulic R, Thalmann D, Fua P. Style-based motion synthesis. *Computer Graphics Forum* 2004; **23**(4): 799–812.
13. Bruderlin A, Calvert TW. Gal-directed, dynamic animation of human walking. *In Proceedings of SIGGRAPH*, 1989; **23**(3): 233–242.
14. Girard M, Maciejewski AA. Computational modeling for the computer animation of legged figures. *Computer Graphics (SIGGRAPH '85 Proceedings)* 1985; 263–270.
15. Sun HC, Metaxas DN. Automating gait animation. *In Proceedings of ACM SIGGRAPH*, 2001; 261–270.
16. van de Panne M. From footprints to animation. *Computer Graphics Forum* 1997; **16**(4): 211–224.
17. Li TY, Chen PF, Huang PZ. Motion planning for humanoid walking in a layered environment. *Proceedings of the 2003 International Conference on Robotics and Automation (ICRA2003)*, September 2003.
18. Reynolds CW. Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics (SIGGRAPH '87 Proceedings)* 1987; **21**(4): 25–34.
19. Reynolds CW. Steering behaviors for autonomous characters. *Proceedings of Game Developers Conference 1999*, Miller Freeman Game Group, San Francisco, California, pp. 763–782. 1999;
20. Blumberg BM, Galyean TA. Multi-level direction of autonomous creatures for real-time virtual environments. *In Proceedings of ACM SIGGRAPH*, 1995; 47–54.
21. Tolani D, Goswami A, Badler NI. Real-time inverse kinematics techniques for anthropomorphic limbs. *Graphical Models and Image Processing* 2000; **62**(5): 353–388.
22. Koga Y, Kondo K, Kuffner J, Latombe JC. Planning motions with intentions. *In Proceedings of ACM SIGGRAPH*, 1994; 395–408.
23. Rose C, Sloan P, Cohen M. Artist-directed inverse kinematics using radial basis function interpolation. *Computer Graphics Forum, Eurographics* 2001; **20**(3): 239–250.
24. Merriam-Webster's Collegiate Dictionary. Merriam-Webster, Eleventh Edition, July 2003.
25. Lasseter J. Principles of traditional animation applied to 3D computer animation. *In Proceedings of ACM SIGGRAPH*, 1987; **21**(4): 35–44.
26. Barraquand J, Langlois B, Latombe JC. Numerical potential field techniques for robot path planning. *IEEE Transactions on Systems, Man, and Cybernetics* 1992; **22**(2): 224–241.
27. Latombe J-C. *Robot Motion Planning*. Kluwer Academic Publisher: Boston, MA, 1991.
28. Kovar L, Gleicher M, Pighin F. Motion graphs. *ACM Transactions on Graphics* 2002; **21**(3): 473–482.
29. Parent R. Computer animation: Algorithms and techniques. 12.6:10-11, Morgan Kaufmann, August 2001.
30. Open Inventor, An object-oriented toolkit for interactive 3D Graphics. URL: http://oss.sgi.com/projects/inventor/
31. Midiio, The cross-platform MIDI software interface for C++ programming. URL: http://midiio.sapp.org/
32. GLUT, The OpenGL Utility Toolkit. URL: http://www.opengl.org/rsources /libraries/glut.html
33. SoGLUT RenderArea. URL: http://www.fit.vutbr.cz/~peciva/projects/SoSome RenderArea/index.php3

DOI: 10.1002/cav