CO-OPERATIVE AND CONCURRENT BLENDING MOTION GENERATORS

Vincent Bonnafous, Eric Menou, Jean-Pierre Jessel, René Caubet

IRIT
University Paul Sabatier, 118 route de Narbonne
31062 Toulouse
France
{bonnafou|menou|jessel|caubet}@irit.fr www.irit.fr

ABSTRACT

In this paper we will be describing a new animation architecture and its implementation in our system LIVE. This model introduces a new blending layer approach which uses several motion generators on the same character at the same time. Despite the numerous studies done, animating characters or complex objects in a virtual world remains a difficult problem. In fact, the complexity of the data used to represent the characters (often articulated rigid bodies) makes their control by an animator difficult and using only one technique to generate motion tends to limit the quality of the animation. The solution will probably be provided by the co-operative or concurrency use of several motion control methods.

Keywords: character animation, articulated rigid bodies, motion generators, kinematics, dynamic, motion blending.

1. INTRODUCTION

Animating virtual humans with actions that reflect real human motion is still a challenge. In fact, the complexity of the data used to represent the characters makes their control difficult. Several motion control methods (called MCM) or motion generators or motor skills exist [Thalmann91] to control virtual humans. Each of them has its own advantages but none are adapted to all situations. In fact, not all the actions of a virtual human can be generated in real time using only one technique.

Our approach is not to use just one, even improved, to produce an animation, but to use multiple motion generators to generate realistic animation, taking into account our real time constraints. We can use several motion generators co-operatively or concurrently. In co-operative ways, several methods are used on several different parts of the characters. In the concurrent approach, two or more methods are used on the same element of a character. In this last case, the techniques are blended. To do so, we have developed a specific layer called the blending layer. This layer participates in a global project with a new layer model for the animation software. We implement this model in a platform called “Life In Virtual Environment” (LIVE). This program is inserted between a behavior model and a rendering system. It is also integrated in a virtual environment.

In this paper, we will describe our layer model and more particularly the blending layer. To synchronize the different methods during the animation, we use the task concept for which, the animation is divided into elementary bricks named tasks.

This software works in real time environment with virtual humans or with articulated rigid bodies; it is not designed for real-time interaction. Motion capture is not used in real-time interactions, this method is only used to record a sequence. The capture sequence is stored in a database, and then this sequence is used with other methods (inverse kinematics and dynamics) to generate the motion.

Our final goal is to create a system able to respond to orders from a behavioral model or a user and to synthesize the corresponding animation. Designing such a system is a difficult objective, but this paper describes how we begin to solve the problem in our particular approach. We try to create smart avatars as described in [Badler98]. In real-time applications, avatars are human representations driven directly by

a real person. In general, an embodied character that acts from its own motivations is often called an agent. Balder calls an avatar controlled via instructions from a live participant a smart avatar.

In section 2, we will present a background of existing motion generators and the different approaches where the authors use more than one motion generator at the same time. In section 3, the layer architecture of LIVE is presented and each layer described. Section 4 presents our implementation and our results. The last section concludes and presents the directions for our further work.

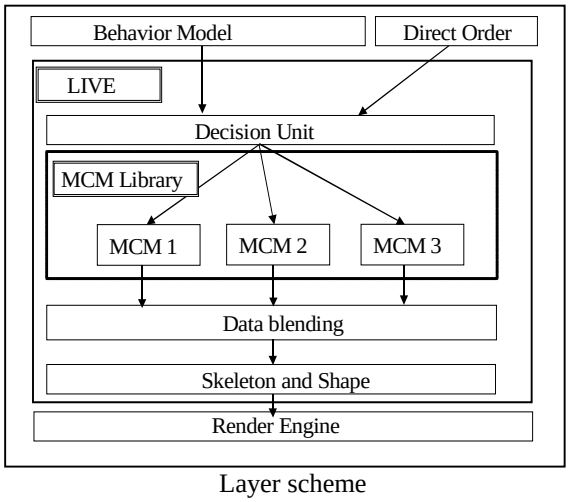
2. BACKGROUND

Among the numerous papers on motion control methods, we can point out the studies of [Thalmann91], [Boulic97] and [Maiocci96] for motion capture, [Barzel88] and [Green91] for dynamics and [Welman93] for inverse kinematics.
One approach is to adapt an existing motion to new situations or characters. A system introduced by Witkin and Popovic, that modifies time-warped joint trajectories by using keyframes with a scaling factor and an offset [Witkin95]. Other researchers have created parametric behavior from several sample in order sequences to adapt motions [Rose98].
Some authors have used more than one motion generator to produce motion. In [Zoran95], the author uses a simplified dynamic model to edit motion for behavior such as running and jumping. In this paper, two methods (direct kinematics and dynamics) are used to produce a new motion. In [Zordan99], the upper-body motion of a running human is modified; a new motion is produced with motion capture and inverse kinematics. [Boulic92] presents an approach with both direct and inverse kinematics control.
The AGENTlib software architecture presents a good approach to the subject of motion generators mixing. In [Boulic97], the authors present a management of action combination using keyframe sequence, inverse kinematics and motion capture. They also use a task system with priority and a transition system between each task. The AGENTlib architecture is grouped in a global project called ACE [Kallmann00].
A similar project, that exists at the University of Pennsylvania includes a low-level motor skills with Jack [Badler99], a mid-level parallel automata controller, and a high-level conceptual representation for driving humans through complex tasks with a Parameterized Action Representation (PAR) [Badler98][Levison94] and an expressive motion engine (EMOTE) [Zhao00][Badler98]. In the same approach Perlin has developed a human animation system Improv [Perlin96] which includes scripting language to provide a sufficiently powerful interface for instructing virtual humans. In [Blumberg95], the authors discuss the problem of building autonomous animated creatures for interactive virtual environments which are capable of being directed at multiple levels.
In a high level application, scenario languages are defined for animation and simulation by [Multon98], [Donikian99] and [Funge99]. In their system the actions are described by a scenario and an animation system generates the corresponding motion.

3. LAYER MODEL

3.1 PRESENTATION

LIVE is software that works on articulated rigid bodies, such as human bodies. The goal of the system is to answer an order by generating the corresponding animation. For example, if a virtual human is ordered to “go to the chair and sit down”, the system may synthesize an animation where the human walks to the chair and sits down on it. LIVE is inserted between a behavior model and a rendering engine. Our model generates an animation using several motion generators. We use these methods in a co-operative or concurrent way.
Here we present the architecture of LIVE. The application is constructed on a layered model. Its structure presents many advantages. For instance, each layer is independent, and can be modified without altering the others. That way, they can evolve easier and faster.



Layer scheme
Figure 1

Figure 1 represents the hierarchical layered architecture. The treatment is performed from top to bottom. The information is processed successively by each layer. A layer takes the information from the layer above, transforms it and sends the result to the layer below.

There are four layers in LIVE. The decision unit receives an order and chooses in the MCM library the best motion generators to achieve the animation. In this layer, we use a low-level script language; this script calls a MCMs sequence to generate the motion. This layer is currently being developed. The MCM library is a set of several Motion Control Method (MCM) or motion generators. To date, we have four methods, which are described in section 3.4. The blending layer is used to collect sets of data and mix them and is described in section 3.5. The representation of articulated rigid bodies is described in section 3.2.
The animation process is simple. When the behavior model sends an action, the decision unit splits it into a set of tasks. For each frame, each active task is treated by motion generators and sends data to the blending layer. The blending layer processes data and sends the final data to the skeleton representation layer which finally provides a mesh to the render engine.

3.2 SKELETON AND SHAPE

LIVE is based on articulated rigid bodies. To represent these bodies, we use three layers [Chadwick89] : a skeleton layer, a muscle layer and a shape layer. The most important is the skeleton layer because it stores the position of the character's bones.
This skeleton layer is an articulated hierarchy which provides the foundation to control the motion of the character. The skeleton is represented by a hierarchy of bones. Any two consecutive bones in the hierarchy are connected by a ball and socket joint (three degrees of freedom). The orientation of each joint gives the global position of the characters. In fact, each bone stores its rotation with regard to its parent in a euler angle, and the character has a vector for its position.
The muscle layer is used with the dynamic motion generators. Springs are used to simulate the muscles. The springs are attached to the bones.
The shape layer is very simple, we associate a triangular mesh to each bone.

3.3 TASK

To perform the animation, we introduce a concept of task. The task is the elementary brick of the animation. To generate an animation, the decision unit gives a set of tasks to the MCM library. In the simulation loop, each motion generator executes its associated tasks.
The structure of a task is simple. Each task is associated to one motion generators and is applied to a set of bones. A start and an end time specify the time interval during which the task is performed. The task can be stopped at any moment by a simple kill command.
When two or more tasks involve the same bone at the same time, the motion generators execute the animation independently for every task in separate buffers and sends the new orientation of the bone for each task to the blending layer. This layer calculates the final orientation for the bone. For this, each task has a priority level (high, medium, low) and a weight function. Operation of the priority system is described in section 3.5.
Each task has specific parameters depending on its associated motion generators. For the inverse kinematics, the task has the final desired articular configuration. For the direct kinematics method, we add two data sets, the name of the keyframe sequence and an initialization time. The method uses this initialization time to create a smooth motion between the current position of the characters and the first keyframe of the sequence. In the case of a cyclic keyframe sequence, a search method finds the frame where the position between current state and the frame state is closest.
A state is associated to each task to manage its execution. The description of each state is described in figure 2. A task generates an animation only when it is in the RUN state.

WAIT	State when current time is greater than task start time.
START	State when the task begins. This state is used for initialisation.
RUN	Normal state during the animation.
KILL	End of the task. This state is used for termination.
STANDBY	When other tasks have a greater priority on the same element at the same time, the state of the task turns to standby.

Task States
Figure 2

3.4 MCM LIBRARY

We describe here the library of motion generators. We use only simple motion generators to control the characters. It is the blending of several motion generators which generates complex animation.
The first goal of the application is not the creation of new motion generators. Therefore, we apply existing algorithms to implement our motion method. The library currently includes four methods.

3.4.1 DYNAMIC

The dynamic method is important in animation because it is the only method that generates animation using physical laws. The drawback is that it is not fast. Generally, Newton's law or virtual work principals are used to simulate the dynamics. To simplify the development, we chose to use an existing library of dynamics, Math Engine [MathEngine]. This library is free for research and provides rapid results. Each character has dynamic properties. A dynamics world is created based on these parameters. At the beginning of the animation, all the bones are inactive. When we animate a set of bones with this method, the bones become active in the dynamics world and their orientation is computed by the library at each frame.

3.4.2 REAL-TIME MOTION CAPTURE

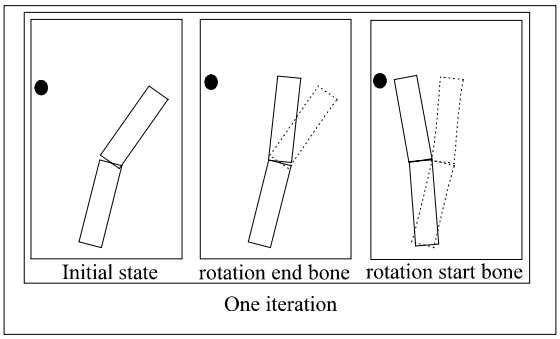
Motion capture is often used to animate a character. This method supplies data from the real world using sensors. In LIVE, we use a space mouse and some Polhemus sensors. Each sensor is associated to a bone and the position and orientation of the sensors provide the new position and orientation of the bone. Motion capture is used only to create a keyframe sequence for a character.

3.4.3 PLAY BACK MOTION CAPTURE

Play back motion capture consists in playing a keyframe sequence. This method provides the value of the orientation of each associated bone at any time by the help of a set of keys. Values are interpolated between two keys with a slerp algorithm. At the present time, we use a simple linear interpolator. Each character has a library of keyframe sequences representing different motions. Two kinds of sequences are used by this technique, simple sequences and cyclic sequences.
To store a sequence, we can use a very simple method, simply by recording the animation. The animation of a set of bones is recorded between a start time and end time. After recording, the keyframe sequence is stored in the character library.

3.4.4 INVERSE KINEMATICS

Taking the environment into account is difficult using only motion capture or dynamic. Inverse kinematics is efficient to adapt an existing motion to the environment. In this task, we define a chain of bones and a goal. To solve the problem, matrix inversion or optimisation methods are generally used. As the matrix inversion is an expensive process, we choose an optimisation method called Cyclic Coordinate Descent (CCD) [Welman93].

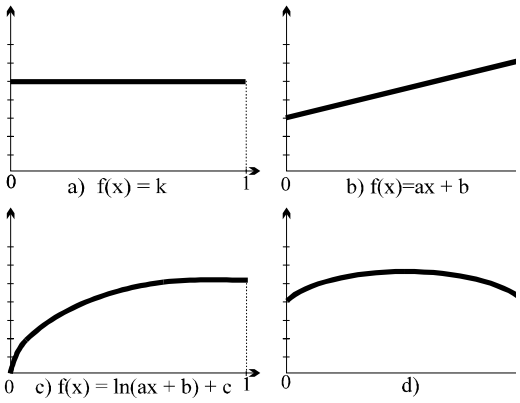


Cyclic Coordinate Descent method
Figure 3

The CCD method minimizes the error on the position of the final effector by adjusting the orientation of each bone one by one. The idea is to adjust the orientation of each joint, starting from the fist and going through each joint until the last, so that the final effector location matches the goal location. The process is iterated until the error on the location of the final effector is lower than a threshold (see figure 3).
This method also allows, a path to be followed. The path is defined with a list of points and computed via a Hermite interpolation. Thus, part of the character can follow a specific path in the virtual world.

3.5 DATA BLENDING

The blending layer is important in LIVE because it allows the motion generators to be used co-operatively and concurrently. At each time of motion, tasks control a set of bone and for each bone sends an orientation to the data blending. The layer receives a set of orientations for each bone and computes the final orientation according to a set of parameters. To do so, we use a priority system.

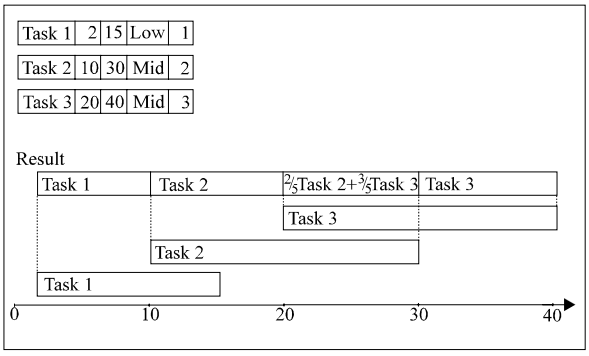


Weight Functions
Figure 4

There are two cases: either a bone is associated to a motion generator and there is not a conflict; or a bone is associated to a set of motion generators and there is a conflict which the layer must resolve with this priority system. When there is not a conflict, the blending layer does nothing ,the layer only receives one euler angle and sends it on as a final orientation. There are also no conflict when we use several motion generators on different bones; this is a co-operative process. For example, using dynamic for the balancing motion of the arms, a keyframes sequence for the legs and inverse kinematics for the head. If two motion generators are associated to the same bone, it creates conflict. In this case, the methods are used with a concurrent process. This conflict has to be resolved by the blending layer. In this case, the blending layer uses the information provided by the tasks. Each task has a priority level (high, medium or low) and a weight function (see figure 4).
When conflict occurs, the layer receives two or more orientations for one bone. The treatment is as follows: the process compares the priority of each task and keeps the higher priority tasks. If only one orientation remains, the conflict resolution is over, and the layer sends this orientation to the shape layer. Otherwise, the process computes a weighted average of the orientations returned by tasks with the higher priority. The orientation is given by:

$$ori = \frac{\sum_{i=1}^{nb\_task} weight\_task(i).ori\_task(i)}{\sum_{i=1}^{nb\_task} weight\_task(i)}$$

where nb_task is the number of tasks with higher priority and weight_task(i) is the weight of the i^{th} task (given by the weight function) and ori_task(i) is the orientation returned by the task i.



Priority Process
Figure 5

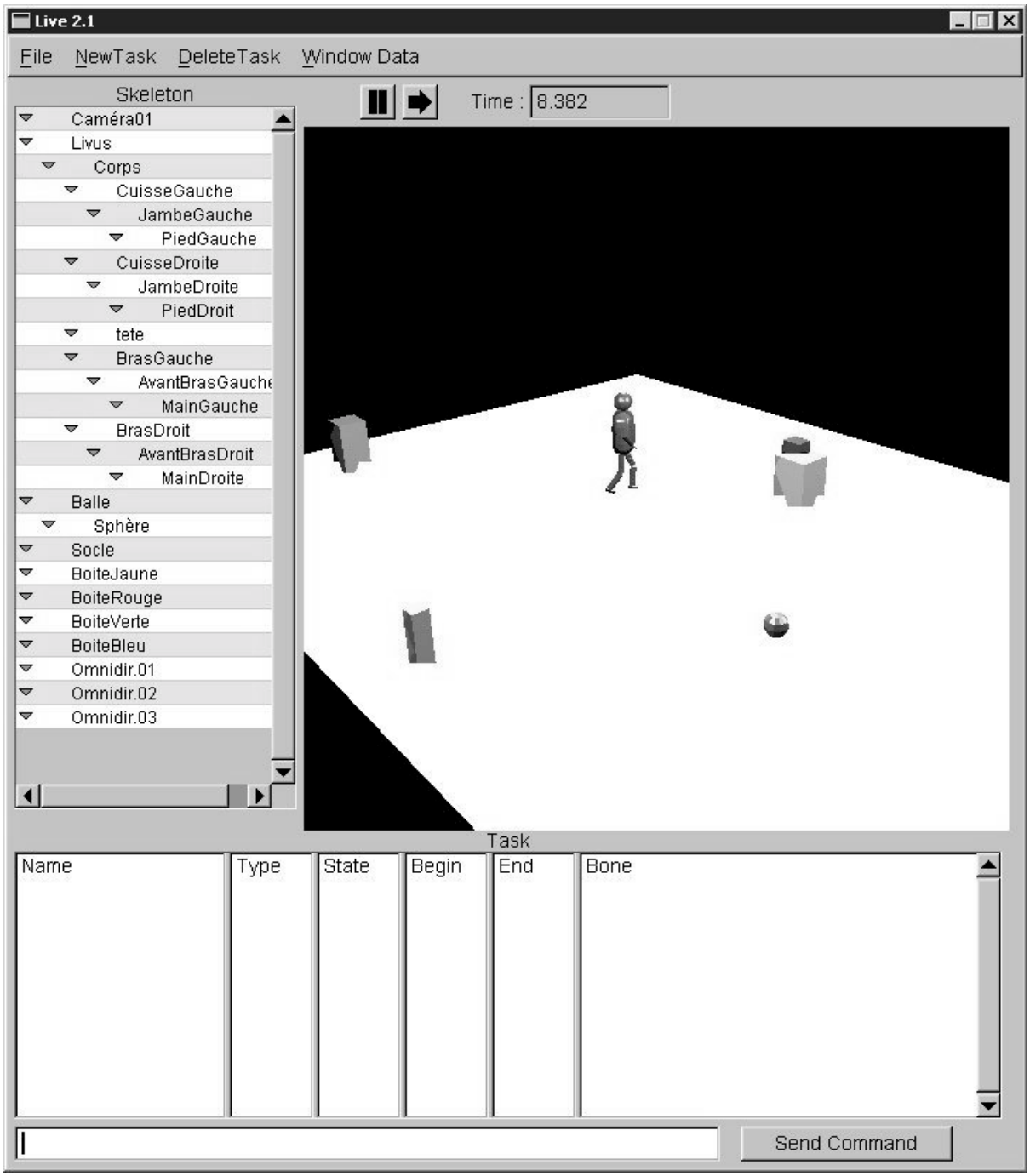
Figure 5 illustrates the selection of the task with regard to the priority. From time 2 to 10, we only have one task, so no conflict occurs. From 10 to 20, the priority level of task 2 is higher than the priority of task 1, so task 2 is selected and the state of task 1 becomes standby. From 20 to 30, two tasks are active at the same time. In this case weights are used to compute the orientation. The final orientation is given by the equation: ori = (2.ori_task2 + 3.ori_task3) / 5 where task2_orientation and task3_orientation are the orientations sent by task 2 and task 3.
The coefficient of each term is calculated with the weight function. Figure 5 shows four examples of weight functions. They depend on time, 0 is the beginning and 1 is the end time of the task. Example a is the most common because the weight is constant during the animation. In the other cases, it changes during the animation. In cases b and c the weight grows with time. Thus, the task takes on a higher priority in the animation. In case d, we can have a smooth transition between the tasks because the weight is lower at the beginning and at the end of the task.

4. RESULTS

We implemented our layer model in C++. For the interface, we used FLTK. In figure 6, we can see a screen capture of the interface. We also see the tree representation of the scene on the left.
First space, we test each MCM individually. In this case, the results are correct. The results are different in the concurrency way. In this case, if we choose two or more motion generators with divergent motion on the same bone then the results are not what we were waiting for. In this way, we can produce realistic movements, if we choose the good blending of motion generators. In figure 6, a human walks to grasp a red box. In this case, we send a command “walk to the red box” and “grasp the red box”. For each command, the system executes a specific script. The walk script uses several motion generators, a cyclic motion capture and an inverse kinematic at the end of motion. With this command, the human character executes the action. The quality of animation depends on the quality of capture and complexity of the script.

5. CONCLUSION

We have presented a new architecture for the real-time animation of virtual humans. We have also presented a new technique for blending motion control technique with a priority system.
With our application, many new movements can be created with the MCM combination. Co-operation works well and produces realistic movements. Concurrency, on the other hand generally produces wrong movements with an uncoordinated state, a better choice of method and weight function could produce better results.
We are presently working on decision layer which would choose a set of tasks to perform an action as



Live Interface
Figure 6

described by [Balder98]. We are also working to introduce new transitions between tasks.

REFERENCES

[Badler99] Badler, N; Palmer S.P, Bindiganavale, R : Animation Control for REAL-TIME VIRTUAL HUMANS, *Communication of the ACM*, Vol.42, No.8, pp 65-73, 1999.

[Badler98] Badler, N; Chi, D; Chopra, S : Virtual Human Animation Based on Movement Observation and Cognitive Behavior Models, In Proccedings of the Computer Animation, pp 128-137, 1998.

[Badler98] Badler, N ; Bindiganavale, R ; Bourne, J ; Palmer, M ; Shi, J ; Schuler, W : A Parameterized Action Representation for Virtual Human Agents, *Workshop on embodied conversational characters*, Lake Tahoe, California, 1998.

[Barzel88] Barzel, R : A Modeling System Based On Dynamic Constraints, *ACM Computer Graphics*, Vol.22, No.4, 1998.

[Blumberg95] Blumberg, M.B, Galyean, T.A : Multi-Level Direction of Autonomous Creatures for Real-Time Environments, *Siggraph*, pp.47-54;1995.

[Boulic92] Boulic, R : Combined Direct and Inverse Kinematic Control for Articulated Figure Motion Editing, *Computer Graphics Forum*, Vol.2, No.4, pp.179-188, 1992.

[Boulic97] Boulic, R; Becheiraz, P; Emering, L; Thalmann, D : Integration of Motion Control Techniques for Virtual Human and Avatar Real-Time *Animation, ACM International Symposium VRST*, 1997.

[Brudelin95] Brudelin, A : Motion Signal Processing, *Siggraph*, pp.97-104,1995.

[Chadwick89] Chadwick, J.E : Layer Construction for Deformable Animated Character, *Siggraph*, pp.179-188, 1989.

[Donikian99] Donikian, S; Devillers, F; Moreau, G : The Kernel of a Scenario Language for Animation and Simulation, *Computer Animation and Simulation, Eurographics Animation Workshop*, 1999.

[Funge99] Funge, J; Tu, X; Terzopoulos, D : Cognitive Modeling : Knowledge, Reasoning and Planning for Intelligent Characters, *Siggraph*, 1999.

[Green91] Green, M : Using dynamics in computer animation, *Making Them Move*, Morgan Kaufmann publishers, pp.281-314, 1991.

[Kallman00] Kallmann , M; Monzani, J.S; Caicedo, A; Thalmann, D : ACE: A Plateform for the Real Time Simulation of Virtual Human Agents, *EGCAS'2000, 11th Eurographics Workshop on Animation and Simulation*, 2000.

[Levison94] Levison, L; Balder, N : How Animated Agents Perform Task: Connecting Planning and Manipulation Through Object-Specific Reasoning, *Presented at the AAAI Spring Symposium : Toward Physical Interaction and Manipulation*, 1994

[Magnenat91] Magnenat-Thalmann, N : Complex Models for Animating Synthetic Actors, *IEEE Computer Graphics & Application*, pp.32-44,1991.

[Maiocchi96] Maiocchi, R : 3D character Animation Using Motion Capture, in *Advanced Interactive Animation*, Thalmann&Magnenat-Thalmann (eds), Prentice-Hall Europe, ISBN 0-13-518306-X, 1996.

[MathEngine] www.mathengine.com.

[Multon98] Multon, F; Controle du Mouvement des Humanoides de Synthese, *these de l'université de Rennes 1*,1998.

[Perlin96] Perlin, K, Goldberg, A : Improv : A System for Scripting Interactive Actors in Virtual Worlds, *Computer Graphics*, pp.205-216, 1996.

[Rose98] Rose, C; Cohen, M; Bodenheimer, B : Verbs and adverbs: Multidimensional motion interpolation, *IEEE Computer Graphics and Applications*, Vol.18, No.5, pp.32-40, 1998.

[Thalmann91] Magnenat-Thalmann, N; Thalmann, D : Complex Models for Animating Synthetic Actors, *IEEE Computer Graphics & Applications*, pp.32-44, 1991.

[Wang91] Wang; Chan : A Combined Optimization Method for Solving the Inverse Kinematics Problem of Mechanical Manipulators. *IEEE Transactions on Robotics and Automation*, Vol. 7, No. 4, pp. 489-499, 1991.

[Witkin95] Witkin, A; Popovic, Z; Motion Warping, *Siggraph*, pp.105-108, 1995.

[Welman93] Welman, C : Inverse kinematics and geometric constraints for articulated figure manipulation, *master thesis*, 1993.

[Zhao00] Zhao, L; Costa, M, Badler, N : Interpreting Movement Manner, *Procedings of Computer Animation 2000*, 2000.

[Zoran99] Zoran, P; Witkin, A : Physically based motion transformation, *Siggraph*, 1999.

[Zordan99] Zordan, V.B; Hodgins, K : Tracking and Modifying Upper-body Human Motion Data with Dynamic Simulation, *Computer Animation and Simulation, Eurographics Workshop*, 1999.