Control of Articulated Figures Animations Using Petri Nets

ALESSANDRO DE LIMA BICHO
ALBERTO BARBOSA RAPOSO
LÉO PINI MAGALHÃES
Department of Computer Engineering and Industrial Automation - DCA
School of Electrical and Computer Engineering - FEEC
State University of Campinas - UNICAMP
Caixa Postal 6101, 13083-970 Campinas, SP, Brasil
{bicho, alberto, leopini}@dca.fee.unicamp.br

Abstract. In this paper we explore the use of Petri Nets as a tool to control the movements of articulated figures in computer animations. This approach permits us to describe the animation sequence by means of the treatment of events present in its execution. An advantage of this method is that the control may be abstracted in different levels, spanning from the definition of the relation among limbs for a single movement to behavioral directives. In addition, our treatment of events hides the mathematical model that describes the movement in fact, allowing the animators to choose the better technique for their applications. In this paper we use an inverse kinematics tool for this purpose. The use of Petri Nets also allows previewing the behavior of the animation before starting any shot.

1 Introduction

This paper investigates one of the facets of computer animation, which is the control of articulated figures movements. Movement control strategies can be thought as driven by two categories of algorithms, one directed to specific aspects of animating characters' parameters–e.g., a movement equation–and another to high level control aspects, where animation intentions are expressed as, for example, goal directed control strategies. Figure 1 illustrates this idea [1].

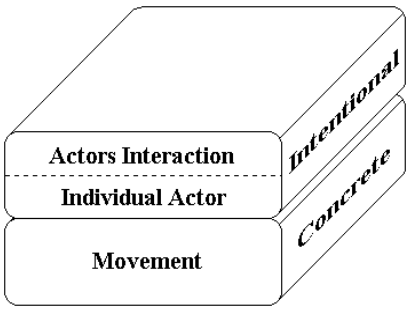


Figure 1 Movement modeling techniques framework.

At the concrete level, movements are considered strictly according to their mathematical modeling. Paradigms like key-frames interpolation, kinematics and dynamic models and biological techniques are examples of this class of solutions. The intentional level, on the other hand, is built on top of the concrete level and uses techniques that support more abstract directives, allowing to define events, restrictions, reactions, etc. Representative techniques in this context are behavior-based models, reactive systems, etc. The main objective at this level is not to define how movements will be performed, but the sequence of events that causes or affects a movement.

In parallel to these two levels, we also proposed the division of the animation control problem into two parts, local control and global control [2]. The local control suggests a mathematical tool, such as inverse kinematics, for modeling an individual character (e.g., a limb of an articulated figure). The global control, on the other hand, suggests a logical approach to manage the interaction among the characters (e.g., arms and legs of a biped figure). The definition of what is local or global control, however, is dependent of the context, since there can be different abstraction levels for control. For instance, both the relation between two actors and the relation between the leg and the foot of a biped figure can be considered global control, depending on the abstraction level we intend to work.

In this paper, we explore the use of Petri Nets (PNs) as a global control tool, used on top of an inverse kinematics tool (IKAN–Inverse Kinematics using Analytical Methods) [3], which implements the local control of articulated figures in the concrete level. In the

following section we review some techniques related to our work. In Section 3 we discuss PNs fundamentals and present the use of this tool for the control of articulated figures animations. Section 4 presents some examples of the PN-based control in different abstraction levels to biped figures. Finally, Section 5 contains the conclusions of this work.

2 Related Work

Many approaches for the control of articulated figures movements have been proposed in the literature. The work of Tmovic and McGhee suggested the application of automata theory of finite-state to the analysis and synthesis of bioengineering systems [4]. Using this approach, the behavior of a natural leg during a steady walk may be modeled by means of a finite-state model. Therefore, it is possible to design a finite-state controller to coordinate ankle and knee motion in prosthesis. This theory has been applied to animation systems.

Zeltzer created a hierarchical model in which the movement is obtained by "motor control programs", which are finite-state machines for executing a particular class of movements, such as "walk" or "run" (there are parameters that allow for different performances) [5]. These programs represent the highest abstraction level of the model. Their states invoke a fixed set of lower abstraction level programs called "local motor programs" (LMPs) to manipulate the joints. LMPs are also finite-state machines that access the joints by changing parameter values. A drawback of this approach is that the animator looses artistic control in favor of automatic motion synthesis.

The work of Fishwick and Porr [6] presented a method for combining discrete event modeling methods with key-frame computer animation. This approach uses PNs to represent the system dynamics at a fairly high abstraction level, and therefore complex systems can be represented as networks or hierarchies of discrete event and continuous models. The focus was to study existing methods in computer simulation, such as PNs, that may be used to aid the graphics community. The authors presented a common example in the literature about synchronization among processes in the operating systems theory (the dining philosophers' scenario) and produced an animation by key-framing method which was not very realistic.

Kalra and Barr [7] introduced a representation of time in which simulation can be neatly partitioned into sub-behaviors connected through events. They formalized the concepts of events and created time primitives called *event units* that may be hierarchically organized to construct motion sequences. This approach provides a partitioning for the problem of motion design, namely, a hierarchical scheme to compose motion behaviors from time primitives and a programming model for organizing animation.

Camargo et al. [2] divided the computer simulation control problem into two parts, namely, a local control problem and a global control problem, as previously mentioned. They proposed an event-oriented scheme to solve the global control problem using concepts related to DEDS (Discrete Event Dynamic System) and ESM (Extended State Machines). This approach used an extension of the "Space-Time Constraints" paradigm of computer modeled animation, named "Space-Time-Event Constraints" paradigm. In the presented example, the walking model for an articulated biped figure, a complex tree figure is split into several smaller blocks, simplifying the initial problem.

Our approach uses a formal framework based on PNs as a global control tool that is better than the ESM because it allows an animator to preview the behavior of an animation even before starting any shot. With PNs it is possible to define a hierarchical model of the actor movements, like that proposed by Zeltzer [5], and it is also suited to event-based systems, like that of Kalra and Barr [7]. Fishwick and Porr [6] also used PNs in their approach. They used the key-framing method rather than inverse kinematics as the local control tool and their transitions, rather than the places, were temporized. Moreover, the abstraction level used in that work does not concern the interaction among limbs of an articulated figure, differently from our approach.

3 Control Using Discrete Events

In animation environments, the movements of articulated figures may be controlled using the time as a frame clock, whose ticks indicate the moments when the behavior of the system must be changed. In spite of that, behaviors in an animation system may also be controlled using event-based tools in a more abstract level. An event means an important point in an animation or physical simulation. The control by means of events enables the animator to easily specify a desired animation sequence, because the treatment using the events domain facilitates changes in parts of an animation without having to remodel all or a large part of the sequence [7]. It also provides a nice composition methodology so that complex figures can be created by combining simpler figures, defining a hierarchical model that facilitates the movement control.

In the work of Magalhães et al. [8] the use of PNs as a modeling and analysis tool for animation environments

was presented, showing to be suited for systems with concurrency, synchronization and event conflicts. In the present paper we apply this methodology to control articulated figures, since behavior of the animation of these structures have such characteristics. In the following we discuss PNs fundamentals and present the approach for the control using this tool.

3.1 Petri Nets Fundamentals

Petri Nets [9, 10] are a modeling tool applicable to a variety of fields and systems, specially suited for systems with concurrency, synchronization and event conflicts. Formally, a PN can be defined as a 5-tuple (P, T, F, w, M_0), where: $P = \{P_1, ..., P_m\}$ is a finite set of places; $T = \{t_1, ..., t_n\}$ is a finite set of transitions; $F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs; $w: F \rightarrow \{1, 2, ...\}$ is a weight function; $M_0: P \rightarrow \{0, 1, 2, ...\}$ is the initial marking; with $(P \cap T) = \varnothing$ and $(P \cup T) \neq \varnothing$.

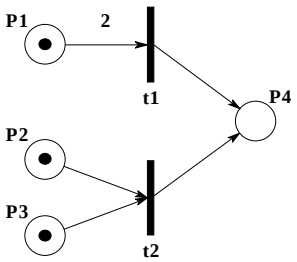
In a PN model, states are associated to places and marks (also called tokens), and events to transitions. A transition t is said to be enabled if each input place $P_i \in {}^\bullet t$ is marked with at least $w(P_i, t)$, which is the weight of the arc between P_i and t. Once enabled, a transition will fire when its associated event occurs. Firing transition t, $w(P_i, t)$ tokens are removed from each input place P_i and $w(t, P_o)$ tokens are added to each output place $P_o \in t^\bullet$. Here, ${}^\bullet t$ and $t^\bullet$ means, respectively, the set of input and output places of transition t.

A very useful notation for PNs is the graphical notation (Figure 2) which is used in the examples throughout this paper. In this notation, circles represent places, rectangles represent transitions, dots represent tokens and arrows represent the arcs, with weights above. By definition, an unlabeled arc has weight 1.

In the PN of Figure 2, only transition *t2* is enabled; *t1* is not enabled because it would require two tokens in *P1* to fire, since *w(P1, t1) = 2*. When *t2* is fired, the tokens in *P2* and *P3* are removed and *P4* receives one token. Note that the number of marks in a PN is not necessarily conserved.

In addition to the basic PN model, several extensions appear in the literature [10]. In this paper we use some extensions, namely, inhibitor arcs, fire priorities to transitions and timed nets. An inhibitor arc connects a place *P* with a transition *t* and enables *t* only if *P* has no tokens. In the graphical notation, inhibitor arcs are represented with a circle on the edge. The basic PN model does not consider the notion of time. One way to include this notion is to establish a waiting time for the tokens in a place before they enable the output transitions [11]. In order to give more consistency to the dynamics of

Graphical Notation



Mathematical Notation

$P = \{P1, P2, P3, P4\}$

$T = \{t1, t2\}$

$F = \{(P1, t1), (P2, t2), (P3, t2), (t1, P4), (t2, P4)\}$

$w(P1, t1) = 2;\ w(P2, t2) = w(P3, t2) = w(t1, P4) = w(t2, P4) = 1$

$M_0 = [1\ 1\ 1\ 0]^T$

Figure 2 PN graphical and mathematical notations.

the PNs we have defined a set of firing priorities associated with the transitions. In the case of having more than one transition enabled, only that with higher priority will fire. If there is more than one transition enabled with the same priority, it is necessary to use a random function to define which one will fire. In the graphical notation, the priority of the transition is indicated by means of a number in parenthesis at the side of the transition label. By definition, when this number is not indicated, the transition has priority equal to 1.

3.2 The Control Method

Before presenting our method, we must define the body model that will be controlled. In this paper, the stick figure model, which consists of a hierarchical set of rigid segments (limbs) connected at spherical joints, will be used. The main advantage of the stick figure model is that the motion specification is easy because it is only necessary to give, for each joint, the values to its three degrees of freedom (DOF). An important point here is that any body model may be animated by moving an underlying skeletal approximation, which does not have to bear any resemblance to the final rendered appearance of the figure. Thus, the motion control problem for figures reduces to that of controlling the movement of an abstract articulated skeleton that, in this case, is a stick figure model.

3.2.1 Global Control

In the global control level we use the PN theory described previously. Two types of places are defined in our model, *condition place* and *action place*. *Condition places* are used to represent conditions of the net control and do not model any movement. *Action places* are used to represent movements that will be executed by an actor. In a lower abstraction level, action places represent the movement of a limb or a set of limbs of the actor. In a higher abstraction level action places may represent, for instance, activities such as "walk" or "jump".

The execution of the movement is modeled by a transition representing its instantaneous starting, with a directed arc to a place representing the movement being executed. An execution time T_i is assigned to each place P_i. A token becomes ready to aid in enabling an output transition of place P_i only T_i time units after P_i received the token. This approach, in which the timed places represent a movement or a net condition, has been chosen for three reasons:

- It preserves the classic PN notion of transitions as instantaneous events.
- It is possible to represent and execute more than one movement concurrently. If we had represented the movements as transitions, only one could fire at an instant, therefore executing only one movement.
- It does not obscure the state of the system (represented by the net marking) during the time that a movement is executing.

3.2.2 Local Control

The local control level represents the effective movements of a stick figure model. For this purpose, we adopted the inverse kinematics method. This method was chosen because it is only necessary to specify discrete positions and motions for end parts. The system then computes the necessary joint angles and orientations for other parts of the body to put the specified parts in the desired position, executing the necessary motions. Trying to animate an articulated figure using, for example, the forward kinematics method is completely intuitive but tedious to do in practice. The motions of the end parts are determined indirectly as the accumulation of all transformations that lead to those end parts. We also do not use the forward dynamics method because it is necessary to know forces and physical laws *a priori* to realistically simulate determined movements. For example, it is not easy to find physical laws and joint torque patterns to predict how the leg will move in a walking biped figure. An alternative to this problem could be to use the inverse dynamics method to analyze the torque and forces required for the given motion, but this method is also very difficult to use. Consequently, we decided to use the inverse kinematics method in the local control level in this work.

We used a tool named IKAN that provides an inverse kinematics algorithm to compute the desired limb posture [3]. This tool uses a combination of analytical and numerical methods to solve generalized inverse kinematics problems including position, orientation, and aiming constraints. The combination of analytical and numerical methods results in faster and more reliable algorithms than conventional inverse Jacobean and optimization-based techniques. This method also allows for the user to interactively explore all possible solutions using an intuitive set of parameters that defines the redundancy of the system.

3.2.3 Global Control over Local Control

Although it is necessary to have a local control method, an advantage of our approach is to abstract the movement technique used by means of action places in the global control level. This allows that the animator chooses the best technique to simulate the desired movement without having to remodel the sequence of the animation, because it was defined in the event domain. Many simulations of the articulated figures movements have implicit discrete behaviors. For example, in a biped figure walk there are events (behavioral rules) that must be used so that the animation looks realistic [8]:

- Both legs may not be out of the ground at the same time.
- The same leg may not be raised more than one time sequentially.

Furthermore, we may also synchronize the arms motions to those of the legs. Thus, the left arm motions may be synchronized with the right leg motions and the right arm motions may be synchronized with that of the left leg, characterizing the classic behavior of this skill.

In this case, using a modeling tool to describe this cyclic sequence of movements facilitates the work of the animator. It is only necessary to construct, in the time domain, a set of movement functions that will be used in the local control level, such as forward limb, backward limb, and so on. It is used the time domain in this level because the events could not change the behavior of these movements. Thus, these movements are organized in the global control level considering the events that cause transitions among them. This provides an ability to partition the motion design problem into smaller problems of determining behavioral rules and of determining events that connect the movements that

compose these rules. Moreover, if the animator wants to change parameters of the movements, e.g., the size and/or velocity of a step, it is only necessary to alter the respective functions in the local control level. In another situation, if the animator wants to remodel the walk such that the arms do not move anymore, it is necessary to remove the action places in the global control level that invoke the functions responsible by the arms movements, without changing the rest of the animation control model.

4 Experimental Results

To illustrate the feasibility of our method, this section presents two examples of movement control in different abstraction levels to biped figures that resemble human beings. Thus, both upper and lower limbs of these figures have the same characteristics, i.e., the same DOF.

In the first example, we model the control among limbs to simulate the walk of a figure. In the second one, in a higher abstraction level, we model the control between two figures to simulate a ball game.

4.1 Biped Figure Walking

In the computer animation field the human walk has been exhaustively studied and, at least conceptually, it is well understood. The walk is a cyclic sequence in which the legs swing forward and backward, providing alternatively support to the body. This sequence is achieved by imposing some "behavioral rules" mentioned in the previous section, which we used in this example.

In order to solve this motion control problem, we define that the global control level is responsible for the coordination among the legs and arms. Thus, the possible movements of each limb are defined using an approach partially based on the model present by Girard and Maciejewski [12]. We use in this work a number of parameters presented in that paper as necessary for describing the gait of a biped figure.

A *gait pattern* describes the sequence of lifting and placing of the feet. The pattern repeats itself as the figure moves; each repetition of the sequence is called the *gait cycle*. The time taken to complete a single gait cycle is the period *P* of the cycle. Moreover, the duration of the leg phases is called *support duration* and *transfer duration*. Hence, we have:

$$P = SupportDuration + TransferDuration \quad (1)$$

During each gait cycle period any given leg will spend a percentage of that time on the ground. This fraction is called the *duty factor* of leg and it is given by:

$$DutyFactor = \frac{SupportDuration}{P} \quad (2)$$

The walk requires that the duty factor of the each leg exceeds 0.5 since, by definition, both feet must be on the ground simultaneously for a percentage of the gait cycle period. Duty factors less than 0.5 would result in running skill, because the entire body would leave the ground for some duration. In addition, we may define the *stroke* as the distance traveled by the body during leg support duration. For our example, the values for stroke and duration will be, respectively, equal to *x* and *n*. Consequently, the velocity of the step will be *x/n*.

The movement rules and parameters, and also the feet and hands trajectories are defined in the local control level and encapsulated in action places in the PN that models the global control level.

We modeled the PN for the global control level (Figure 3) according to previously defined "behavioral rules". Table 1 presents the function defined to each condition place and Table 2 presents the movements that should be executed for each action place. For simplicity and without losing generality in the modeling of the net, it is defined that the duty factor is equal to 0.5. In addition, the arms motions are synchronized with the legs motions. Thus, it is necessary a single action place to execute these motions and consequently a single transition to represent their instantaneous beginnings.

The animation sequence of a biped figure walking in a straight line behaves in the following way. The figure is initially at rest position, such that the arms and legs are parallel to the length of the body. After the figure have walked the number of steps defined in the weight of the arc between place P3 and transition t5, which in our example is equal to 3, it will be again at rest position. Thus, the first and last steps will have half the size and duration of a normal one. In the PN of Figure 3 (result in Figure 4), the first step is modeled by P2 (frames 1 to 10), the last one by P7 or P8 (frames 50 to 60), whereas the entire step is modeled by P4 or P6 (frames 10 to 50). The start of a step, except the last one, adds one token to P3, and its finish enables two output transitions of the place at issue. While there is not a token in P5, the transition enabled with higher priority will fire, indicating that the walk is not finished. Otherwise, the inhibitor arc will disable this transition and only the transition enabled for the last step fires.

An action place in Figure 3 is responsible for a set of limbs movements. However, in a lower control level, we could assign for each action place a single limb movement. Thus, there would be four action places executing concurrently for each step.

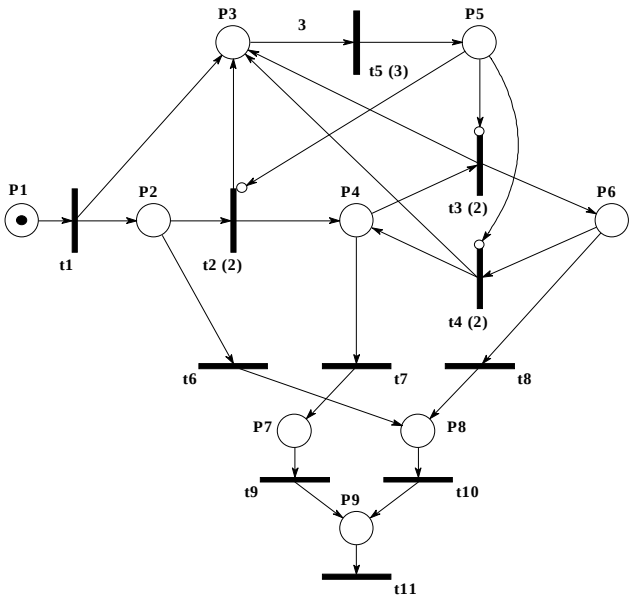


Figure 3 PN model for the global control of the walk of a biped figure.

Condition places	Function
P1	start walk
P3	steps counter
P5	steps counter equals to the number of steps defined
P9	finish walk

Table 1 Condition places.

Action places	Movements	Parameters values
P2, P7	left leg supports the body; right leg transfers forward; left arm swings forward; right arm swings backward	time = n/2 stroke = x/2
P8	left leg transfers forward; right leg supports the body; left arm swings backward; right arm swings forward	
P4	left leg transfers forward; right leg supports the body; left arm swings backward; right arm swings forward	time = n stroke = x
P6	left leg supports the body; right leg transfers forward; left arm swings forward; right arm swings backward	

Table 2 Action places for the movements.

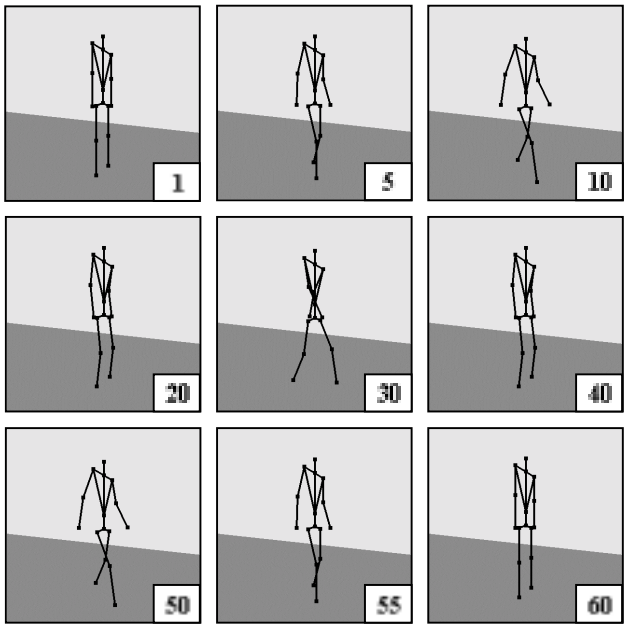


Figure 4 The biped figure walking.

The partitioning could change the movement of a determined limb more easily and also facilitate the reuse of this PN in another situation, removing unnecessary places.

4.2 Biped Figures Playing Ball

This example illustrates the control of articulated figures movements in a higher control level. Hence, the intention is to demonstrate that our approach using PNs is suitable for different abstraction levels. The animation consists of two biped figures walking in a road and playing a ball. Thus, from time to time the actors stop and that with the ball throws it for the other. The walk continues, always alternating the possession of the ball. It is important to observe that the actors do not necessarily walk the same distance or the same number of steps before stopping. Therefore, the interactions among these actors may be described by means of events (behavioral rules) that must be used in the animation control:

- The actor with the ball, after executing his walk, continues only after throwing the ball for the other.
- The actor without the ball, after executing his walk, continues only after grasping the ball.
- The actor with the ball may only throw it if the other one is ready to grasp the ball.

We modeled the PN for the global control level (Figure 5) according to these behavioral rules. Table 3 presents the movements of the actors that should be executed for action places of the net.

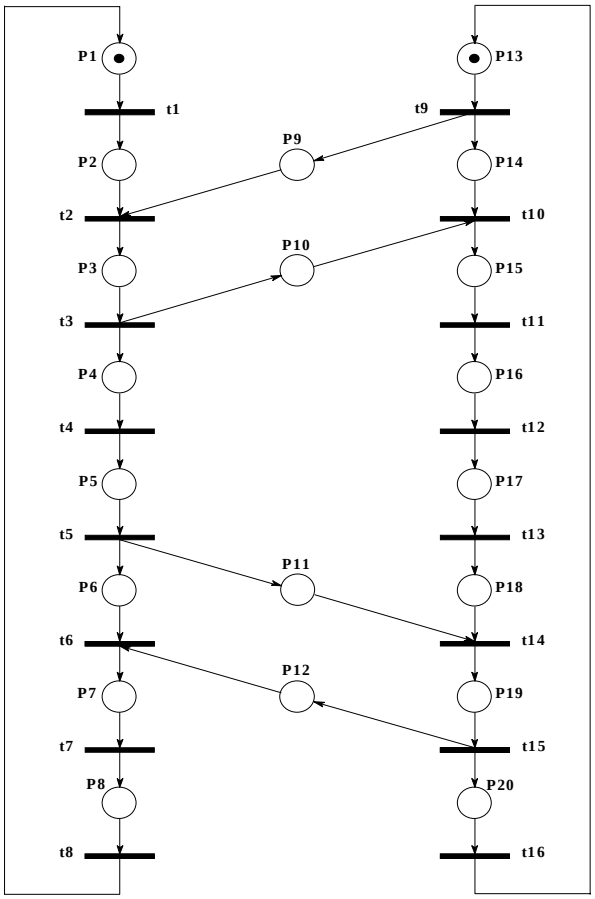


Figure 5 PN model for the global control of the figures playing ball.

Movement	Action place	
	Actor A	Actor B
Walk with the ball	P1	P17
Position to throw the ball	P2	P18
Throw the ball	P3	P19
Position to walk without the ball	P4	P20
Walk without the ball	P5	P13
Position to get the ball	P6	P14
Get the ball	P7	P15
Position to walk with the ball	P8	P16

Table 3 Action places of the actors movements.

The action places P10 and P12 model the ball's trajectory. The condition places P9 and P11 indicate that the actor is ready to grasp the ball. For this model, these places are temporized (they have the same time that the action places P14 and P6, respectively). Thus, the actor will have time to position the body to grasp the ball. In this example the actors could walk infinitely, but it does not occur because we defined a limit of frames for the animation. Figure 6 presents the resulting animation, with actor A on the left side and actor B on the right side of each frame.

This example illustrates not only the reuse of basic PN models (the movement of walking, for example, is modeled by the PN presented in the previous example), but also the capacity of encapsulation offered by PNs. The graph of Figure 5 hides the details of the movements, such as walk or position, enforcing a hierarchical description model.

5 Conclusion

In this paper we used the notions of global and local control to implement a PN-based method for the global control of articulated figures movements that works over an inverse kinematics library (local control). Taking into account the division of movement control strategies into concrete and intentional, we consider our approach as belonging to the concrete level, although it has a higher abstraction level than the direct use of inverse kinematics. The event-based model and the use of PNs, although yet mathematical tools, hide the low level "operations" of the inverse kinematics. For this reason, our approach is a concrete level control method that steps towards the intentional level.

The use of PNs as movement control tool is justified because they are amenable both to simulation and formal verification, since there are numerous tools and techniques available. Moreover, PNs may accommodate models at different abstraction levels, ranging from models closely related to the local control (such as the control of each limb of an articulated figure, as shown in the walking biped example) to higher level models (such as the ball game example, that deals with the interaction among actors).

It is our belief that in order to make use of the full potential of computer animation, it is necessary to provide means for application-oriented users like artists to use physically based control methods. On the other hand, for users like scientists interested in complex simulations, it is important to avoid the trial and error method to compose their animated visualizations. Our PN-based control approach is a step towards both directions, in the sense that it provides a higher abstraction level tool and also a powerful analysis and simulation tool.

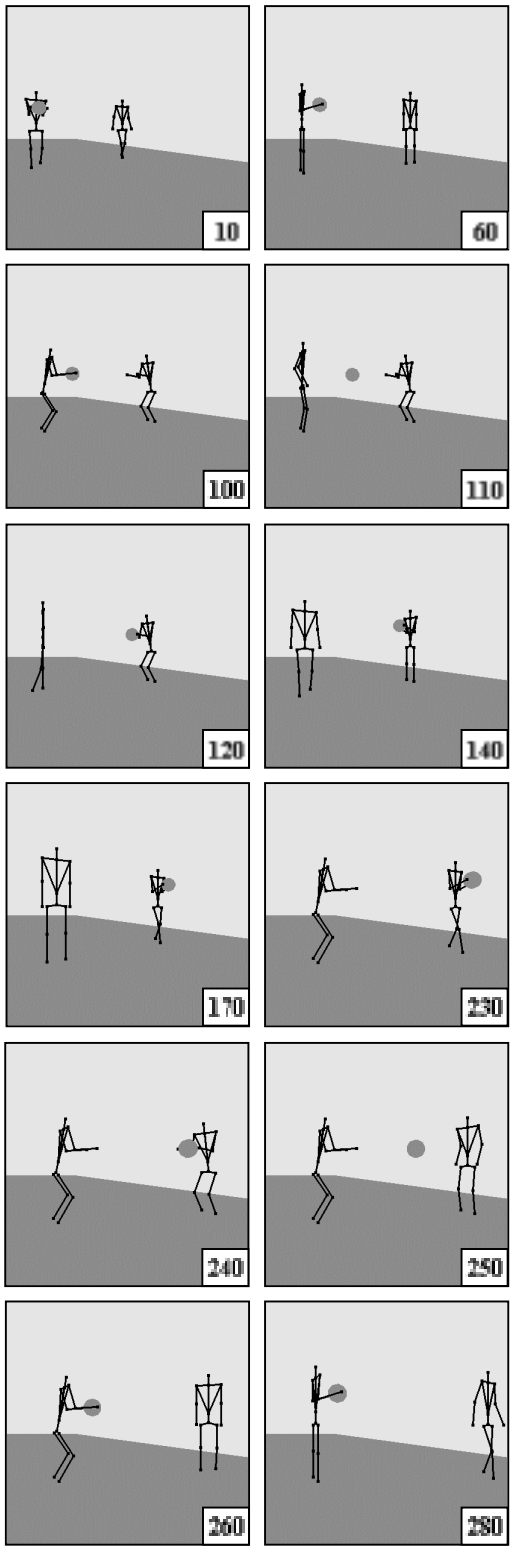


Figure 6 The biped figures playing ball.

Acknowledgements

The first author is sponsored by CAPES. The second author is sponsored by FAPESP, grant number 00/10247-3. Special thanks to Prof. Norman Badler and his research group at the Computer and Information Science Department, University of Pennsylvania, for sending us the IKAN library and related bibliography.

References

[1] L. P. Magalhães, "Modeling and Analysing Computer Animations", *Proc. of XI SIBGRAPI* (1998), 18–19.

[2] J. T. F. de Camargo, L. P. Magalhães and A. B. Raposo, "Local and Global Control in Computer Animation", *Proc. of VIII SIBGRAPI* (1995), 151–157.

[3] D. Tolani, A. Goswami and N. I. Badler, "Real-Time Inverse Kinematics Techniques for Anthropomorphic Limbs", *Graphical Models* 62(5) (2000), 353–388.

[4] R. Tomovic and R. B. McGhee, "A Finite State Approach to the Synthesis of Bioengineering Control Systems", *IEEE Transactions on Human Factors in Electronics* HFE-7(2) (1966), 65–69.

[5] D. Zeltzer, "Motor Control Techniques for Figure Animation", *IEEE Computer Graphics and Applications* 2(9) (1982), 53–59.

[6] P. A. Fishwick and H. A. Porr, "Using Discrete Event Modeling for Effective Computer Animation Control", *Proc. of Winter Simulation Conference* (1991), 1156–1164.

[7] D. Kalra and A. H. Barr, "Modeling with Time and Events in Computer Animation", *Computer Graphics forum* 11(3) (1992), 45–58, (*Proc. of EUROGRAPHICS '92*).

[8] L. P. Magalhães, A. B. Raposo and I. L. M. Ricarte, "Animation Modeling with Petri Nets", *Computers & Graphics* 22(6) (1998), 735–743.

[9] C. A. Petri, "Kommunikation mit Automaten", Schriften des IIM Nr. 3, Bonn: Institute für Instrumentelle Mathematik, 1962.

[10] T. Murata, "Petri Nets: Properties, Analysis and Applications", *Proc. of the IEEE* 77(4) (1989), 541–580.

[11] J. E. Coolahan, Jr. and N. Roussopoulos, "Timing Requirements for Time-Driven Systems Using Augmented Petri Nets", *IEEE Transactions on Software Engineering* SE-9(5) (1983), 603–616.

[12] M. Girard and A. A. Maciejewski, "Computational Modeling for the Computer Animation of Legged Figures", *Computer Graphics* 19(3) (1985), 263–270, (*Proc. of SIGGRAPH '85*).