Automatic animation control of physical systems

Georges Dumont, Bruno Arnaldi, Gérard Hégron

INRIA

UNITÉ DE RECHERCHE
INRIA-RENNES

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tél.:(1) 39 63 55 11

Rapports de Recherche

N° 1170

Programme 6
Robotique, Image et Vision

AUTOMATIC ANIMATION CONTROL OF PHYSICAL SYSTEMS

Georges DUMONT
Bruno ARNALDI
Gérard HEGRON

Février 1990

RR-1170

IRISA

INSTITUT DE RECHERCHE EN INFORMATIQUE
ET SYSTÈMES ALÉATOIRES

Campus Universitaire de Beaulieu
35042 - RENNES CÉDEX
FRANCE
Téléphone : 99 36 20 00
Télex : UNIRISA 950 473 F
Télécopie : 99 38 38 32

Automatic Animation Control of Physical Systems.

Georges DUMONT
Bruno ARNALDI
Gérard HEGRON

INRIA [1] */ IRISA* [2]
Campus de beaulieu
35042 Rennes Cedex
France

Contrôle automatique de l'animation de systèmes physiques.

Publication Interne n°511 - Janvier 1990 - 22 Pages

[1]Institut National de Recherche en Informatique et Automatique.
[2]Institut de Recherche en Informatique et Systèmes Aléatoires.

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE
(L. A. 227)
UNIVERSITÉ DE RENNES 1 I. N. S. A. DE RENNES

INSTITUT NATIONAL DE RECHERCHE
EN INFORMATIQUE ET EN AUTOMATIQUE
(LABORATOIRE DE RENNES)

Résumé

Dans ce rapport, nous présentons la conception d'un système d'animation dans lequel les systèmes de solides rigides et déformables sont animés à partir des informations extraites de leurs modèles géométriques et de la spécification de contraintes cinématiques et dynamiques.

Nous présentons les aspects théoriques, numériques et pratiques de la mise en œuvre du système d'animation. Il est basé sur la dérivation automatique et sous une forme symbolique des équations du mouvement à partir d'un modèle créé de manière interactive. Ce processus sera décrit par l'étude complète d'un exemple. Le contrôle du mouvement est basé sur l'application de la dynamique directe, mais le système permet de plus l'animation d'objets (solides) déformables. La prise en compte de contraintes holonomes et non holonomes pour spécifier un effet désiré, est faite sans avoir a calculer les efforts qui réalisent ces effets. Nous proposons également de détecter et de traiter les chocs qui peuvent mettre en jeu les objets d'une scène. Des résultats expérimentaux illustrent ce problème de contrôle du mouvement. Pour conclure, nous présentons des résultats de simulation scientifique et les discutons.

Abstract

This report presents the design of an extensible animation system in which rigid and deformable multibody systems are animated from their geometric models and the specification of kinematic and dynamic constraints.

Theoretical, numerical and practical aspects of the system implementation are presented. One of its main features is the automatic derivation of the symbolic form of the motion equations from a physical system model created interactively. This process is detailed by means of a simple example. Besides achieving a motion control by the application of direct dynamics, the system provides an animation of deformable objects, an automatic motion control from a specified motion without having to determine the forces required to perform this desired effect, and an object collision detection and response. Experimental results are presented to illustrate each animation control. Finally, the ability of our animation system for scientific simulation is discussed.

Automatic Animation Control of Physical Systems

Georges DUMONT
Bruno ARNALDI
Gérard HEGRON

Publication Interne n° 511
Janvier 1990

1 Introduction

An object animation requires a motion coordination. Traditional animation systems where the animator explicitly controls motions and deformations, become inadequate when the scene contains a lot of objects whose motions are due to actions, reactions, joint constraints and physical properties. For this reason, the use of simulation methods in animation has been considerably investigated recently.

Simulation models especially developed for solving specific natural phenomena have already been proposed by Fournier and Reeves[9, 10, 19] and by Miller[15]. A background on mechanics regarding the animation of rigid solid objects can be found in [1, 14, 25]. Recently, deformable models have been extensively featured in the literature and interesting results are already available. Barr[4] creates deformable objects by means of prescribed deformation jacobian matrices. Terzopoulos[21, 22, 23] uses dynamical deformable models and a finite difference method to model the deformation under prescribed forces. Furthermore, he has proposed physically-based models of objects capable of inelastic deformation. Some authors have developed different constraint approaches for the animation of rigid or flexible bodies [5, 18, 26, 27].

The methods presented in this paper provide a means to incorporate some of these animation techniques within a single coherent system. Our animation system has been implemented whose main originalities with respect to the previous ones are :

- The automatic derivation in a symbolic form of the motion equations from physical objects models created interactively ;
- The animation of rigid and deformable objects under an unified form ;
- The automatic motion control of physical objects from a desired motion taking into account dynamics properties without using inverse dynamics.

Our animation system also handles the object collision detection and response at each time step.

In this paper, we first describe the principles of our system, concerning the user interface and the mechanical formalism. Thereafter the implementation of the system is developed. The automatic writing of symbolic motion equations is especially detailed on a simple example. Then deformation, motion and collision controls are explained and illustrated with experimental simulations. Finally, the ability of our system to perform scientific simulation is discussed.

2 System Principles

2.1 Overview

The object of this work is the animation of any multibody systems using traditional motion control techniques as well as dynamics. As the animator may be neither a computer scientist nor a mechanician but an artist, the animation system has to provide automatic motion computation from the description of the mechanism behavior.

The animator constructs the multibody system and specifies constraints so as to produce the desired motion. This modeling is interactively achieved in three steps : object shape design, object interrelationship description, specification of kinematic and dynamic constraints evolving over the time.

Geometry, material composition and joint coordinate systems of the objects are designed by using a solid modeling system. The animator is provided with a standard joint library. These joints are presented in Table 1.

In a third step, to animate the mechanism, the animator specifies kinematic and dynamic constraints such as :

Table 1: Joint library

Nu.	Joint	Rot	Trans	DOFs
1	Embedding	0	0	0
2	Pin	1	0	1
3	Sliding	0	1	1
4	Twist sliding	1	1	1
5	Cylindrical	1	1	2
6	Plane on plane	1	2	3
7	Ball and socket	3	0	3
8	Cyl. on plane	2	2	4
9	Ball in a cyl.	3	1	4
10	Ball on plane	3	2	5
11	Flying object	3	3	6

- gravity (direction and module);
- springs, dampers, thrusts, forces and torques applied on joint degrees of freedom (DOF), the parameters of which (given in Table 2) may evolve over the time and therefore may be described by a trajectory with respect to time;
- ponctual forces applied on objects;
- nonholonomic constraints as rolling without sliding (for instance, for a wheel);
- for a specialist, hand-written constraint equations if needed.

Table 2: DOF coefficients

Descr.	param.	note
spring	k	rigidity coefficient
	l_0	rest length of the spring
damper	v	damping coefficient
motor	F	force or torque

2.2 Mechanical formalism

The used mechanical formalism is based on the principle of virtual work associated with the LAGRANGE's multipliers [3, 12] or the penalization method. It has been chosen to deal with "general multibody systems"[20, 28] involving holonomic and nonholonomic constraints, open and closed chains, and to derive automatically their motion equations from their geometric, kinematic and dynamic constraint descriptions.

Let $q = (q^i)_{i=1,n}$ be the Lagrangian parameters of one multibody system (S), submitted to p holonomic constraints $f_h(q,t) = 0$, $(h = 1, 2, ..., p)$, and to p' nonholonomic constraints $g_l(q, \dot{q}, t) = 0$, $(l = 1, 2, ..., p')$. These nonholonomic constraints could always be written as :

$$\sum_{i=1}^{n} a_{li}(q,t) \cdot \dot{q}^i + b_l(q,t) = 0$$

where $\dot{q}^i = \frac{dq^i}{dt}$ is the derivative of q^i with respect to time.

The **principle of virtual work** states that there exists at least one reference frame in which, for all virtual displacement of (S) and at any moment, the amount of virtual work that acts upon (S) is zero. It can be expressed by :

$$\delta\mathcal{W}_d + \delta\mathcal{W}_l + \delta\mathcal{W}_j = 0$$

where :
$\delta\mathcal{W}_d$ is the virtual work of given effects,
$\delta\mathcal{W}_l$ is the virtual work of binding effects,
$\delta\mathcal{W}_j$ is the virtual work of inertia effects.

For one system (S) with the generalized coordinates $q = (q^i)_{i=1,n}$, the principle of virtual work can be written as :

$$\mathcal{Q}_i + \mathcal{L}_i + \mathcal{J}_i = 0, \quad i = 1, 2, ..., n$$

where $\mathcal{Q}_i$ (resp. $\mathcal{L}_i$, $\mathcal{J}_i$) is the generalized given (resp. binding, inertia) effect relative to q^i.

Now, let $\mathcal{C}$ be the kinetic energy of (S). By using the LAGRANGE's formula we get :

$$\mathcal{J}_i = -\frac{d}{dt}(\frac{\partial \mathcal{C}}{\partial \dot{q}^i}) + \frac{\partial \mathcal{C}}{\partial q^i}, \quad i = 1, 2, ..., n.$$

The introduction of $\mathcal{L}_i$ can be done in two ways.

If we use the principle of **LAGRANGE's multipliers**, by introducing $\lambda^h\,(h = 1, 2, ..., p)$ and $\mu^l\,(l = 1, 2, ..., p')$, we obtain the following set of equations :

$$\begin{cases} \mathcal{Q}_i - \frac{d}{dt}(\frac{\partial \mathcal{C}}{\partial \dot{q}^i}) + \frac{\partial \mathcal{C}}{\partial q^i} + \lambda^h \frac{\partial f_h}{\partial q^i} + \mu^l a_{li} = 0, & i = 1, 2, ..., n \\ f_h(q, t) = 0, & h = 1, 2, ..., p \\ g_l(q, \dot{q}, t) = 0, & l = 1, 2, ..., p'. \end{cases} \tag{1}$$

This system is a non-linear differential equation system that can be solved by well-known algorithms, the description of which is beyond the scope of this article. Its solving gives, for each time step, the values of the generalized coordinates and those of the LAGRANGE's multipliers which are directly related to the forces exerted by links onto parameters. The system is solved for each of the $n + p + p'$ variables. Therefore, if we do not need to compute these forces, this method induces extra computation. Furthermore, if links are linearly dependent, the system becomes singular.

When there is no need to calculate the constraint forces, we use a **penalization method** to take the links into account. With this method we obtain the system :

$$\mathcal{Q}_i - \frac{d}{dt}(\frac{\partial \mathcal{C}}{\partial \dot{q}^i}) + \frac{\partial \mathcal{C}}{\partial q^i} + k\, f_h \frac{\partial f_h}{\partial q^i} + k\, a_{li}\, g_l = 0, \quad i = 1, 2, ..., n \tag{2}$$

where k is a chosen penalization constant.

This is a non-linear differential equation system of order n which can be solved more quickly than the previous one. It leads to the same results and becomes equivalent to the previous one when k tends towards infinity. No singularity occurs when binding equations are dependent, even if one of these equations is duplicated that remains equivalent to multiply k by 2. Futhermore, penalization method allows to pay no attention to some mechanical inconsistencies : the constraint violations, for instance, are detected on the graphical output. This is a more convenient detection method for an artist.

3 System Implementation

3.1 Overview

The animation system, written in the C programming language, allows to automatically derive and compute the motion equations from any rigid multibody system description. The flow of control of the system is illustrated in Figure 1. The system input is the user description of the mechanism in terms of objects, joints, constraints and initial state (positions, velocities, etc.). The following automatic process can be broken down into three main phases : extraction of mechanical properties of the objects, symbolic derivation of motion equations, and solving of these equations for each time step.

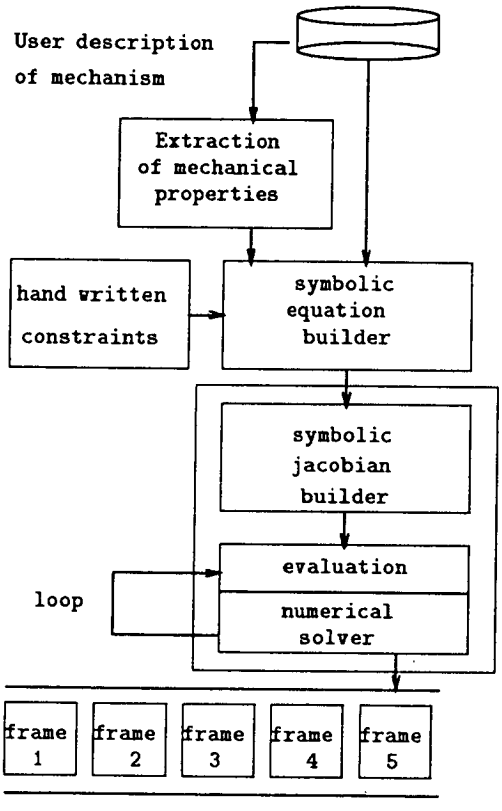


Figure 1: Synoptic of the application

Mechanical properties

The mechanical properties of each object (center of gravity (COG), inertia matrix, principal coordinate system of inertia) are extracted from object geometric model. Inertia and mass are integrals over the volume. A discrete integration is performed by using a parallel ray tracing algorithm from a regular grid mapped on a face of the object bounding box. The ray tracing technique has been chosen owing to its applicability to a large variety of solid models. Along each ray, inner parallelepipeds are computed according to the grid resolution and the intersection points between the ray and the object boundary. Physical properties of the parallelepipeds are calculated, then the object ones using associativity of COGs and additivity of inertia matrices with respect to the global COG.

Formation of symbolic motion equations

The motion equations are then automatically generated under a **symbolic** form from the mechanism data structure (tree) according to the following steps :

- for each object, the COG and the angular velocities are symbolically computed by a tree traversal according to the joints between objects and the relative positioning of local frames. Symbolic names are given to the system parameters during this phase ;
- from the previous results, the whole kinetic energy C of the mechanism is computed as the sum of each object kinetic energy ;
- for each object and each joint, the given work $\mathcal{W}_d$ (gravity, spring, damper, thrust, force and torque works) is evaluated . Its associated given actions $\mathcal{Q}_i$ are derived by performing a symbolic derivation of $\mathcal{W}_d$ with respect to parameters :

$$\mathcal{Q}_i = \frac{\partial \mathcal{W}_d}{\partial q_i} + \frac{\partial \mathcal{W}_d}{\partial \dot{q}_i}$$

- holonomic and nonholonomic constraints are derived from the geometric and kinematic constraint specifications ;
- the last step consists in expressing the motion equations according to Equation 1 or 2.

At this step, a finite difference method is applied to discretize the equation with respect to time. Then the jacobian matrix J, used for numerical resolution, is symbolically computed by using :

$$a_{ij} = \frac{\partial f_i}{\partial q_j}$$

where a_{ij} is the element of J (f_i is the i-th equation and q_j is the j-th parameter). The symbolic computation is performed once only.

Resolution of the motion equations

For each time step, a numerical resolution of the nonlinear system is performed by using a *Newton Raphson* algorithm, as follows :

- evaluation of the symbolic jacobian matrix J ;
- solving of the numerical system : $J\Delta q = f$ where Δq is the unknown incrementation of the parameter vector q and f is the value (evaluated from its symbolic expression) of the equation vector. This computation is achieved by using a LU decomposition of the jacobian matrix.

3.2 Symbolic expressions

In this section, the advantages of a symbolic computation of the motion equations versus a pure numerical solution are discussed. Then the implementation of the symbolic calculus is described.

Why a symbolic way to build equations ?

First, the operations performed under a symbolic form are exact. This is important in case of derivatives, because a reliable numerical derivation is very difficult to perform on a computer. For instance, $\frac{d\sin(x)}{dx}$ will be represented exactly by $cos(x)$ and not by $\frac{sin(x+h)-sin(x)}{h}$.

Second, the equation simplifications, based on the particular form of the modeled system, are easily performed, and the structural knowledge of the mechanism is not lost during the calculation.

The use of a library of symbolic functions offers two main advantages :

- the programming of new features is made easier by the natural way of writing the corresponding routines. Moreover, it is shorter, that reduces the errors of programming. An example of program for the construction of the motion equations which refers to the inertia and given effect terms (see Equation 1), is presented below to illustrate this assertion ;

```
nbequa = 0;
for(i=0;i<nbparam;i++) {
  /* -ddt(d/d(q')) ec + d/d(q) ec */
  interm2 = eq_plus(
              eq_mun(
               deq_dt(
                deq_dqp(ec,ident[i]))),
               deq_dq(ec,ident[i]));
  /* motion equation[i] =   interm2
                          + d/dq W
                          + d/dq' W */
  equa[i] = eq_plus(interm2,
              eq_plus(
               deq_dq(work,ident[i]),
               deq_dqp(work,ident[i])));
  nbequa++;
}
```

- the system offers to the user two different ways of introducing informations : graphical and hand writing. The hand writing of equations under a natural form is appreciated by the specialist who wants to introduce new equations which are not allowed by the graphical interface. An example of a hand-written constraint (where # represents : $= 0$) is given here. These equations are those of the prescribed car trajectory presented in Figure 9.

```
valid*((a+l*cos(theta)-h*sin(theta)-xcercle)*cos(theta+beta1)
+(b+l*sin(theta)+h*cos(theta)-ycercle)*sin(theta+beta1))#

valid*((a+l*cos(theta)+h*sin(theta)-xcercle)*cos(theta+beta2)
+(b+l*sin(theta)-h*cos(theta)-ycercle)*sin(theta+beta2))#
```

Where `valid` allows to activate (or desactivate) the constraint, `a`, `b`, `theta`, `beta1` and `beta2` are DOFs of the car. `l` and `h` are respectively the length and the width of the car. `xcercle` and `ycercle` are parameters of the trajectory.

The data structure

The data structure is based on a binary tree, the leaves of which stores three kinds of entities :

- a constant ;
- a variable which can be modified by external processes (mass, gravity, rigidity and damping coefficients, motor or thrust parameter linked to a trajectory). The variables allow to have a model evolving through time ;
- a DOF of the mechanical system (unknowns, parameters).

The other nodes are operators such as :

- arithmetical operators : $+$, $-$, $*$, $/$;
- functions : $\sin(x)$, $\cos(x)$, $\exp(x)$, $\tan(x)$, $\arctan(x)$, etc... (where x is an expression) ;
- derivatives : $\frac{dx}{dt}$, $\frac{\partial x}{\partial q_i}$ or $\frac{\partial x}{\partial \dot{q}_i}$ where t is the time and q_i is a parameter of the mechanical system.

Such an implementation obviously requires a large memory space as the complexity of the mechanical models grows. But the produced equations are the equations of a physical structured system, and the equations are known to share common terms. For this reason, the replacement of the binary tree structure by a Direct Acyclic Graph is proposed : common equations are shared into the structure by means of pointers. For taking benefit of this implementation two kind of databases are defined : the mechanism database created by propagation along the links of the system (which is a D.A.G by construction) and the equation database. The operations performed during the equation building process are traited by sets of same kind : for instance, time derivatives or derivations with respect to one given parameter are performed once for all the equation system. This allows to keep the D.A.G structure. To this end, a pointer refering to the last calculated derivative is stored within each shared node of the mechanism D.A.G and each new shared node of the equation D.A.G.

3.3 A detailed example

Mechanism Description

In order to illustrate the features of the animation system, an example showing the animation of a simple mechanism, dubbed gyroscope, by direct dynamics is developed. This gyroscope presented in Figure 2 is constituted of five objects : a cylindrical basis, two bars linked by pin joints. On each bar, a parallelipipedic object is linked by a sliding joint.

During the phase of geometric modeling, the animator is provided with a joint library. The description of the joint coordinate systems occurs during the modeling phase. Once the geometric model (including joints) is achieved, the five DOFs of the mechanism are available (see Table 3).

Table 3: List of the gyroscope parameters

name	description
α_0	rot DOF for the object 0
α_1	rot DOF between O_1 and O_0
α_2	rot DOF between O_2 and O_0
l_3	trans DOF between O_3 and O_1
l_4	trans DOF between O_4 and O_2

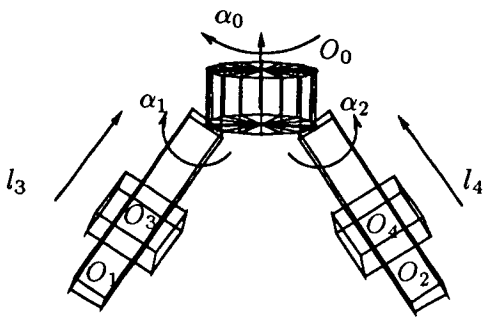


Figure 2: Gyroscope

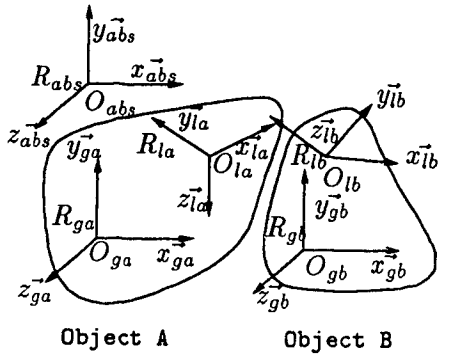


Figure 3: Joint reference frames

Only the relevant DOFs are created during the modeling step. For instance, the introduction of one pin joint (between object 0 and object 1) creates the DOF α_1 and the associated symbolic transformation matrix from object 0 to object 1 :

$$M_{l0->l1} = \begin{pmatrix} \cos(\alpha_1) & -\sin(\alpha_1) & 0.0 & 0.0 \\ \sin(\alpha_1) & \cos(\alpha_1) & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 \end{pmatrix}$$

The objects are parametrized in their inertia reference frame. The joint reference frames R_l are introduced to define the joint location with respect to the local (inertia) reference frames R_g of the considered objects (see Figure 3). For instance, the pin joint, parametrized by α_1, is (partially) defined by its transformation matrix in the local (inertia) reference frame of object 0 :

$$M_{l0->g0} = \begin{pmatrix} -1.0 & 0.0 & 0.0 & 1.0 \\ 0.0 & -1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 \end{pmatrix}$$

Effects as motor, springs, dampers may be applied on the DOF which may be furthermore constrained by thrusts. A summary of these given effects is to be found in Table 4 refering to the equations :

$$\mathcal{W}_{d-spring} = \frac{1}{2} * k * (q - r)^2$$
$$\mathcal{W}_{d-motor} = motor * q$$

Table 4: DOF description

	effect	**coef.**	
α_0	*motor*	*traject.*	
α_1	*thrust*	p= 0.0	$\alpha_1 < p$
α_2	*thrust*	p= 0.0	$\alpha_2 < p$
l_3	*spring*	r= 0.0	k= 100.0
l_4	*spring*	r= 0.0	k= 100.0

The default initial state is defined by the rest position of the system (each DOF is equal to zero) and the external standard given effects. For instance the gravity is represented by the vector : $\vec{g} = (0.0, 9.81, 0.0)$.

The chosen input for this system is a torque on DOF α_0 which obviously allows to generate motion. The inputs (which may vary over the time) are conveniently defined by trajectories as presented in Figure 4.

Output of the mechanism modeler

The mechanical properties, presented in Table 5, are automatically computed by the system. The computation of these properties represents the first step of the animation process.

Table 5: Mechanical properties

O	mass	inertial coef.
0	3.152412	1.048302 1.583837 1.060937
1	4.000000	0.666667 5.666636 5.666636
2	4.000000	0.666666 5.666636 5.666636
3	4.000000	2.666668 1.666639 1.666639
4	4.000000	2.666666 1.666630 1.666630

The next step consists in generating the symbolic expressions of the coordinates of the mass center in accordance with the DOFs. These vectors are derived with respect so as to construct the velocities used later to generate the kinetic energy.

Output of the symbolic calculus

The structural knowledge about the mechanical system (degrees of freedom) allows the symbolic equation generator to calculate the mass center of each object (with respect to the DOF) and the rotational velocities. The translational velocities are the derivatives the coordinates of the mass center with respect to time. A subset of these automatically generated equations (set up by the program) is presented in the following :

- coordinates of the mass center for object 0 :
  $G_0/x = 0, \quad G_0/y = 0.5, \quad G_0/z = 0;$
- rotational velocity vector for object 0 :
  $\Omega_0/x = 0, \quad \Omega_0/y = \dot{\alpha_0}, \quad \Omega_0/z = 0;$
- coordinates of the mass center for object 4 :
  $$\begin{aligned} G_4/x &= \cos(\alpha_0)(-2\sin(\alpha_2) - 1) \\ &\quad - \cos(\alpha_0)\sin(\alpha_2)l_4 \\ G_4/y &= 0.5 + \cos(\alpha_2)(l_4 + 2) \\ G_4/z &= \sin(\alpha_0)\sin(\alpha_2)l_4 \\ &\quad - \sin(\alpha_0)(-2\sin(\alpha_2) - 1); \end{aligned}$$
- rotational velocity vector for object 4 :
  $$\begin{aligned} \Omega_4/x &= \sin(\alpha_2)\dot{\alpha_0} \\ \Omega_4/y &= 0.5(\cos(\alpha_2)\dot{\alpha_0} + \dot{\alpha_2}) \\ &\quad +1.414214(\cos(\alpha_2)\dot{\alpha_0} - \dot{\alpha_2}) \\ \Omega_4/z &= 0.5(\cos(\alpha_2)\dot{\alpha_0} + \dot{\alpha_2}) \\ &\quad -1.414214(\cos(\alpha_2)\dot{\alpha_0} - \dot{\alpha_2}). \end{aligned}$$

Here is one motion equation built by the system (refering to DOF lz_3) :
$-4cos(\alpha_1)^2 lz3\dot{\alpha}_0^2+39.24cos(\alpha_1)-100l_3-4\ddot{l}_3+8\dot{\alpha}_1^2+8\dot{\alpha}_0^2+4l_3\dot{\alpha}_1^2-8cos(\alpha_1)^2\dot{\alpha}_0^2-4sin(\alpha_1)\dot{\alpha}_0^2+4l_3\dot{\alpha}_0^2$

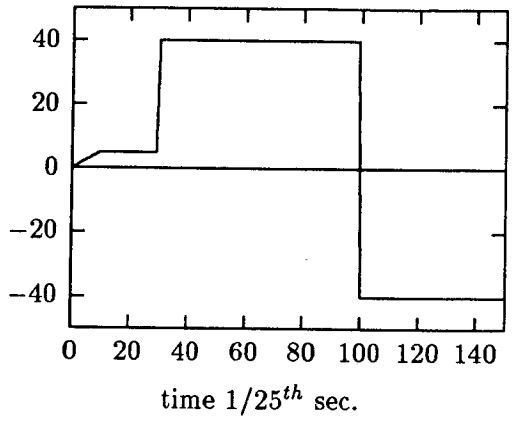


Figure 4: Torque applied on DOF α_0 over the time

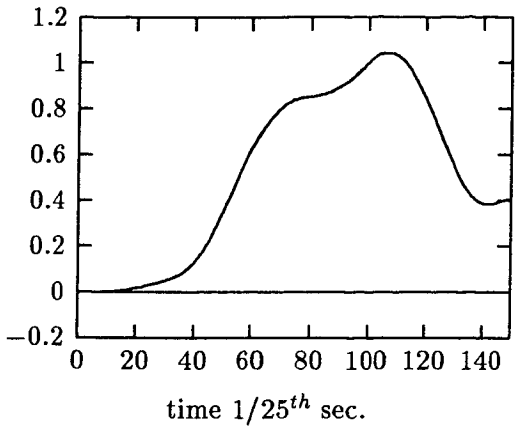


Figure 5: Trajectory for α_1 DOF

Table 6: interest of D.A.G structure

	memory	time for building motion equations	time for 200 frames	number of frames per second
Without D.A.G	131Ko	2.1s	63.4s	3
With D.A.G	46Ko	0.5s	7.2s	> 30

These results have been obtained on a Silicon Graphics Iris 4D/20 Personnal Computer.

Numerical results

The system calculates each parameter value at each time step : for instance, the evolution curve associated with parameter α_1 is presented in Figure 5.

Table 6 illustrates the efficiency of the D.A.G structure with respect to the resolution step.

4 Deformation, Motion and Collision Control

The previous sections describes the animation system abilities for the motion control of any multibody system using direct dynamics. Owing to the chosen mechanical formalism and to the symbolic writing of equations, the system can also handle deformable objects, automatic motion control and object collisions.

4.1 Deformable solids

Two different approaches have been developed for the animation of deformable solids by using mechanical laws.

The first one was based on the finite element method[24, 29]. Our purpose here will not be to describe this method which is well known in the scientific community. Work in this field, using finite difference method, has already been presented by Terzopoulos who furthermore proposed modeling of such system with the use of inelastic deformations. Our implementation was made under a numerical (classical) form[8]. The second one is based on a mesh too, but it uses punctual masses linked by springs and dampers. This method is implemented under a symbolic form into the previous described system. The symbolic equations are derived from the object mesh and the initial state (location of the punctual masses, rest length of the springs) determined by mesh geometry. The involved resolution of these equations is the same as previously. The

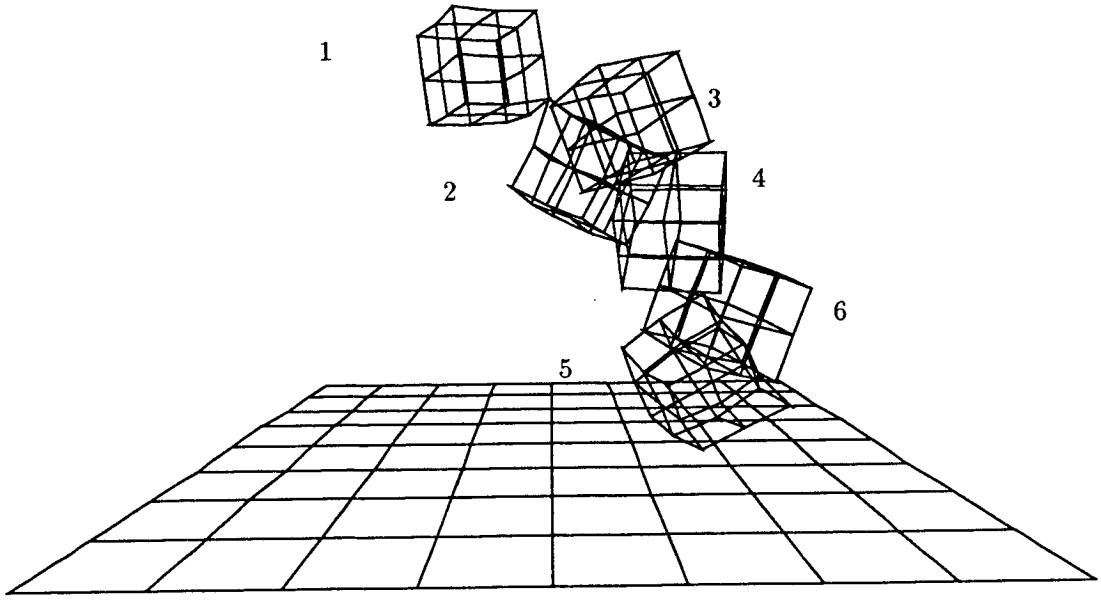


Figure 6: Revolving and fall of a deformable cube

only difference between rigid and deformable models lies in the construction of the symbolic equations from the geometric model.

The revolving and the fall on the ground of an elastically deformable cube illustrates this last approach (see Figure 6). At the beginning, the cube is held at a lower corner with a ball and socket joint. This joint is defined by :

$$\alpha * (x - x_0) = 0$$
$$\alpha * (y - y_0) = 0$$
$$\alpha * (z - z_0) = 0$$

where (x, y, z) are the lower corner global coordinates, (x_0, y_0, z_0) are the initial lower corner global coordinates (coordinates of the joint), α is a variable whose initial value is one.

The cube rotates losing its shape around the point (x_0, y_0, z_0) under the gravity action. Then the cube is dropped by setting the value of α to zero, i.e. the joint does not exist any more. The cube falls down on the ground, bounces and is bent out of shape. The interaction between the cube and the ground is achieved by adding a lower thrust on each cube corner altitude coordinate. The use of symbolic equation writing and of variables whose trajectory is easily specified allows to perform this whole sequence automatically.

This technique can also be applied to deformable surfaces modeled by a mesh.

The animation of deformable solids or surfaces represented by a mesh of punctual masses linked by springs and dampers remains similar to the rigid multibody system ones. The difference lies only in the way of specifying constraints and of deriving symbolic motion equations from these specifications and from object geometry. Moreover, with such deformable object modeling, interactions between rigid and deformable solids are performed in a simple way, because the DOFs are handled independently of the fact they refer to deformable or rigid bodies. This is illustrated in Figure 7, where a rigid sphere bounces on an elastic surface. The non-interpenetration constraints relate the coordinates of the mass center of the sphere (three DOFs) to the surface.

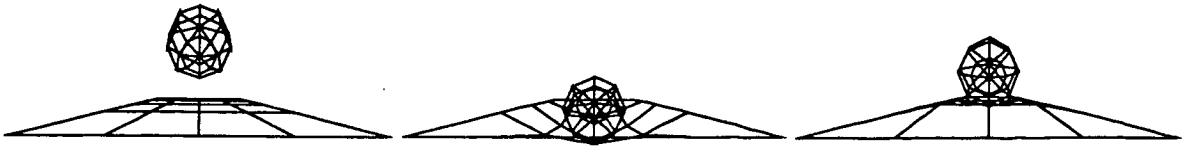


Figure 7: Rigid sphere on an elastic surface

4.2 Automatic Motion Control

The problem of motion control[5] is here presented, which consists essentially in controlling a dynamical system by means of prescribed trajectories. Such a task may always be written as $e = e(\ddot{q}, \dot{q}, q, t)$ which is a constraint, that can be integrated into the motion equations as explained previously. The main difference with robotics, is to avoid the calculation of forces which generate a given effect, as inverse dynamics does. Nevertheless, the effects are taken into account with respect to the dynamic behavior of the system. The main problem is the task modeling which consists in writing the desired effect equations in terms of DOFs only.

In [2], we presented the motion control of an arm whose hand extremity $\vec{X} = (x, y, z)$ follows a 3D spacetime trajectory $\vec{X}_t = (x_t, y_t, z_t)$. The arm is a four link chain (clavicle, upper arm, lower arm, hand) with ball and socket joints (12 DOFs), and thrusts to ensure that motion lies in human capabilities. The task is modeled by three holonomic constraints : $x - x_t = 0$, $y - y_t = 0$ and $z - z_t = 0$ for all t, where x, y and z represent the symbolic expressions of hand extremity location automatically extracted from the root of the arm chain. In this case, task modeling remains obvious.

In order to show automatic control implications onto the user task specification and the task modeling, we present the motion control of a car. The car model presented in Figure 8 is a right parallelepiped suspended on wheels by four coupled spring-absorber. It is animated by exerting a torque mimicing the engine on front wheels. The guiding is assumed to be produced by a torque too, and regulated by a spring-absorber system. The wheels are rolling without sliding on the ground. Such a link can be modelled as : let (O, x, y) be a referential and let us consider a wheel of radius r rolling without sliding on the Ox axis. The contact condition is written as $y_c = r$ (where c is the center of the wheel) and the associated nonholonomic constraint as $r \cdot \dot{\theta} + \dot{x} = 0$. One condition constraining the orientation of the steering wheels is necessary : they roll on concentric circles, that is compatible with the conditions of rolling without sliding.

The nonholonomic constraints are, because of their generality and complexity, of great interest with respect to motion control.

The two-dimensional prescribed trajectory proposed here models the desired motion of car COG. We want to guide the motion without computing the efforts which realize this motion, but by direct writing of the task as a set of constraints, which will be introduced into the motion equations (see Equation 2). These constraint equations are written with respect to the DOFs governing the motion. They express the projection of the trajectory equation onto the space of DOFs.

In the following example, they are written with respect to β_1 and β_2, which are the steering angles expressed in the car reference frame.

When the trajectory is a circle, these constraints are written as follows :

$valid * ((a + l * \cos(\theta) - h * \sin(\theta) - xcircle) * \cos(\theta + \beta_1) + (b + l * \sin(\theta) + h * \cos(\theta) - ycircle) * \sin(\theta + \beta_1)) = 0$

$valid * ((a + l * \cos(\theta) + h * \sin(\theta) - xcircle) * \cos(\theta + \beta_2) + (b + l * \sin(\theta) - h * \cos(\theta) - ycircle) * \sin(\theta + \beta_2)) = 0$

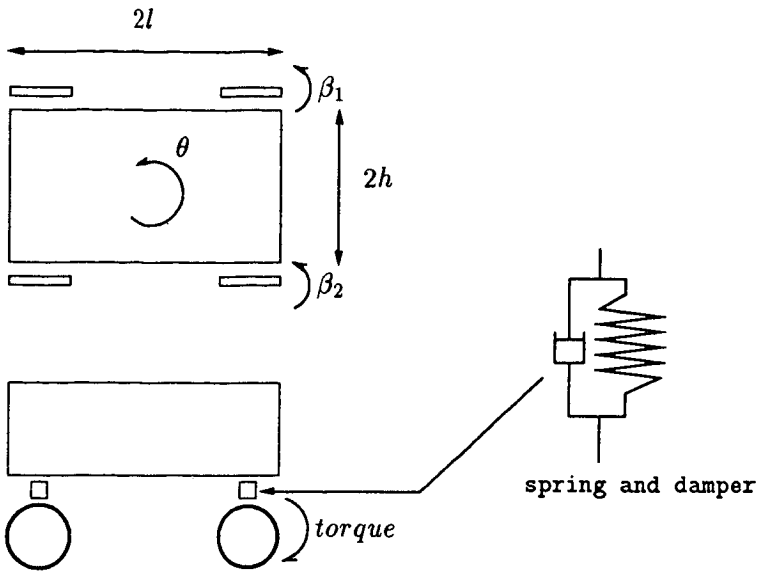


Figure 8: Model of a car

Where a and b are the global coordinates of wheel COG, θ defines the global orientation of the car and, $xcircle$ and $ycircle$ are the coordinates of the circle center.

These equations express that the global velocity of the car COG is tangent at (a, b) to the circle defined by its center $(xcircle, ycircle)$ and its radius R $(= \sqrt{(xcircle - a)^2 + (ycircle - b)^2})$.

The results of motion computation are presentented in Figure 9. The regular curve is the trajectory performed by the car COG under task constraint which remains identical to the prescribed trajectory, i.e the circle. The irregular curve is a variation from the previous one obtained by changing and removing through time either the driving torque, or the rolling without sliding constraint, or the task constraint (circle tracking). These changes are referenced in the Figure 9 under the form "(torque, valid, weight of rolling without sliding constraint)" for each of them.

4.3 Object Interaction

Collision treatment is important in animation systems. Objects have to interact with each other avoiding interpenetration. This problem has already been developed for rigid and deformable objects [11, 13, 16]. Note that a collision physically occurs at a given location and at a given time, and that percussions (actions of collision) are the unknowns.

For solving the contact/collision problem, we have implemented a method based on the principle of virtual work. For one system submitted to one collision from the outer, the variation of momentum is equal to the integration of the external efforts over the duration of collision, which is called percussion action :

$$\Delta(\frac{\partial \mathcal{C}}{\partial \dot{q}_i}) = \mathcal{P}_i$$

Let us recall that the momentum is the derivative of the kinetic energy with respect to the velocities.

A geometrical algorithm detects the collision effects and returns some necessary informations such as the collision plane, the normal vector, as well as the normal velocities of the collision points. When interpenetration is too important (because of the time step) a backward-subdivision of time step is performed so as to detect the spacetime position of the collision points.

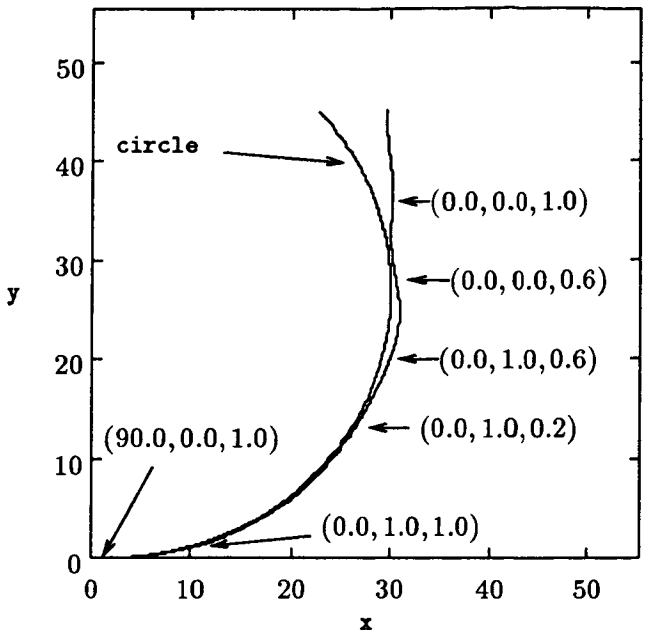


Figure 9: Trajectory of car COG

Then, the collision laws are applied : conservation of momentum, reflexion of relative normal velocities by use of an empirical law : $(V_1 - V_2)^+ = -e(V_1 - V_2)^-$ where V_1 (resp. V_2) is normal velocity of the collision point on first (resp. second) solid and $+$ (resp. $-$) denotes the instant after (resp. before) the collision. The constant e is the coefficient of restitution characterizing the collision. This coefficient depends on the involved objects. If $e = 1$ the collision is elastic and if $e = 0$ the collision is soft.

For each frame (time step t_n), use the recursive procedure defined below. This procedure determines in function of events (as collision) the way of subdividing the calculation time step for treating the events, then induces the event treatment and brings back the initial state compatible for the normal calculation of the next frame (time step t_{n+1}) as if nothing had occured :

```
procedure time_step_management()
begin
 if(not(collision))
  then if(collision_detection)
       then if (treatable)
            then collision_treatement()
            else collision = true
                 subdivide_time()
                 backward_calculation()
                 time_step_management()
       else next_time_step_calculation()
  else if(collision_detection)
       then if (treatable)
            then collision_treatement()
            else subdivide_time()
                 backward_calculation()
                 time_step_management()
       else subdivide_time()
```

```
            forward_calculation()
end
```

The procedure "collision_detection" returns the geometric necessary information to treat the collision. That is : the relative velocities of the objects and the normal vector at contact points. The boolean "treatable" represents a geometric criterium of interpenetration of objects at contact points. The procedure "collision_treatment" performs the treatment based on the principle of virtual work by changing the initial state of the DOFs for the next time step.

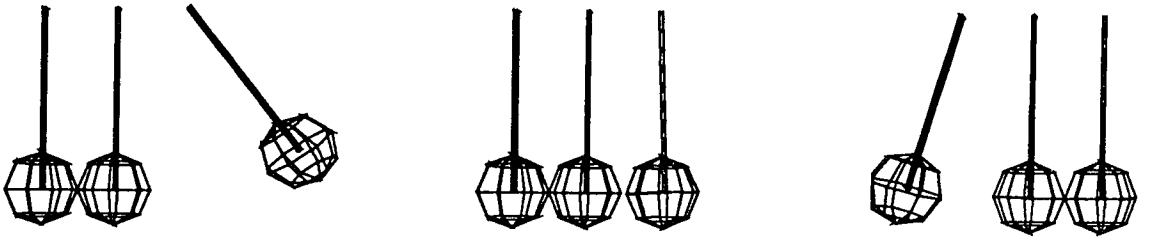


Figure 10: Example of collision

5 From animation to scientific simulation

In previous sections, the use of dynamics for animation was presented. As a general mechanical formalism is used, the ability of our animation system for scientific simulation has been evaluated and compared to ADAMS system [6, 17] using the simulation of a "van in overtaking". The van shown in Figure 11 has ten DOFs which are the six DOFs of frame COG, the rotational DOF for each front wheel around a transversal axis at the extremity of the arm, and two DOFs for the rear axle for pumping (translation along a vertical axis) and for rolling (rotation around a longitudinal axis). Besides the gravity effect, the given effects are suspension forces (see springs and dampers in Figure 11), drift forces applied to each wheels at the contact point according to the drift angle (and to the turning angle for front wheels), vertical forces applied to each wheel center according to the tire stiffness, and propulsion force given in Figure 12. The turning angle is presented in Figure 13.

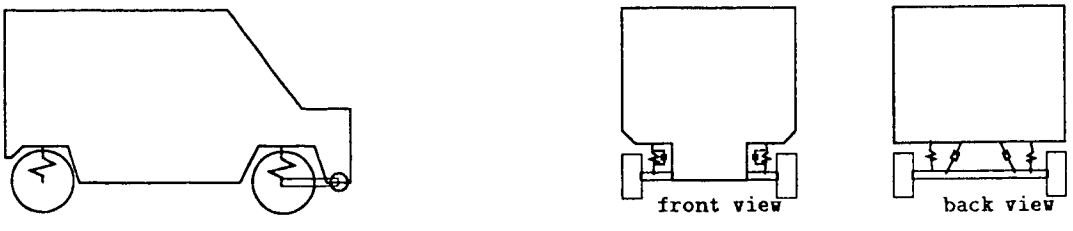


Figure 11: Model of a van

The comparison of DOF accelerations is the most discriminating criterion. Therefore, the numerical results for tranversal acceleration $\ddot{x}$ and longitudinal acceleration $\ddot{y}$ of van COG are

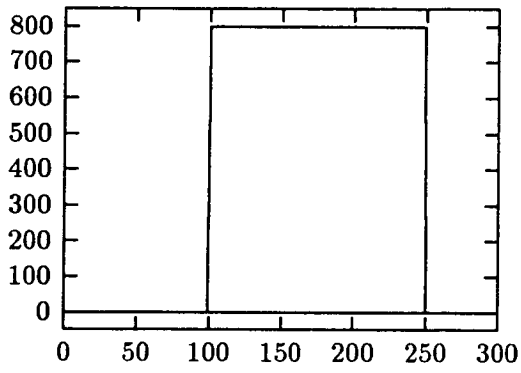


Figure 12: Propulsion force

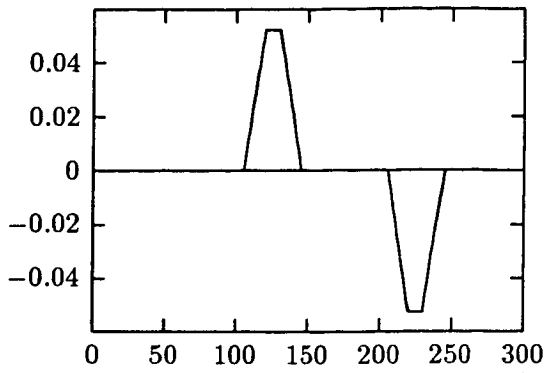
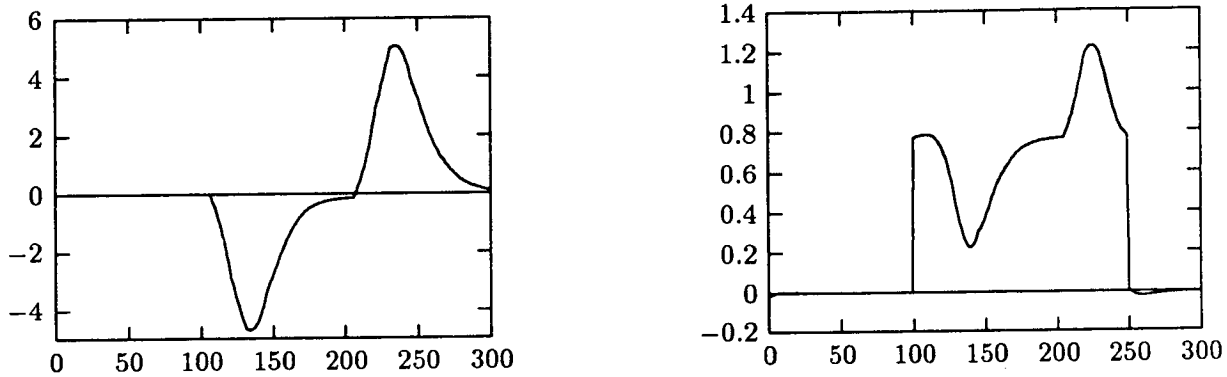
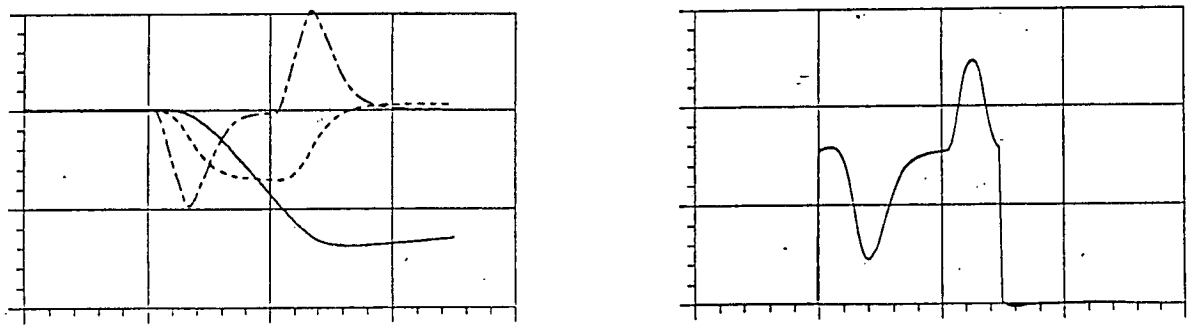


Figure 13: Turning angle

presented in Figure 14. They are equivalent for both systems. At the end of the van in overtaking simulation, the locations (x, y) of the frame COG are the same too (the relative difference is about 10^{-4}).



Our results



ADAMS results

Figure 14: Comparisons between the system and ADAMS

Real-time dynamics simulation plays a major role to study and to experiment physical systems : for instance, in driving simulations[7]. To improve the efficiency of the resolution of the motion equations, the D.A.G. traversal used to evaluate the jacobian matrix from its symbolic representation can be automatically replaced by its equivalent numerical code. Using this, for the "gyroscope" computation leads to 600 frames per second (compare with Table 6). Even if numerical computation provides a considerable improvement in speed with respect to the symbolic version, the implementation of a parallel code remains necessary for complex mechanisms

because the computation time is exponentially dependent on the number of system DOFs. This work is under progress.

6 Futur works and conclusion

We have described an animation system in which the behavior of physical objects is computed from their geometric, kinematic and dynamic models. The system generality is obtained by using a mechanical formalism based on principle of virtual work associated with Lagrange's multipliers or penalisation method :

- animation of general multibody systems with holonomic and nonholonomic constraints, and with open and closed chains ;
- automatic motion control from task modeling with a set of constraints ;
- scientific simulation of physical objects.

The automatic derivation of motion equations in a symbolic form from object models induces the following system facilities :

- implementation of a flexible and friendly user interface providing an artist with a graphical based interface to describe mechanism geometry and behavior constraints (trajectories), with possible intervention for a specialist by means of hand written symbolic equations ;
- exact mathematical operations (as derivation) and numerical stability ;
- event specification : addition or removal of constraints through time by using "variables" ;
- system extensibility : incorporation of collision detection and response in the motion resolution loop, introduction of new kinds of joints and constraints.

The symbolic computation applied to deformable objects modeled as mesh of punctual masses linked by springs and dampers, allows to control the deformation of non rigid solids and surfaces. Furthermore, it allows to control of the interaction between rigid and deformable objects since they are handled in the same way.

The motion control is performed either explicitly by direct dynamics application (Figure 2) and by task modeling (Figure 9) or implicitly by gravity effect and object interaction (Figure 6 and 7). The task modeling differences from a mechanism to another one have been illustrated by the example of an arm along a 3D trajectory and the one of the car following a 2D trajectory. The animator should be provided with an automatic process which would generate the constraint equations from task description (desired motion) and from object model.

Acknowledgements

The van model and ADAMS simulation have been performed by R.V.I. (Renault Véhicules Industriels). Special thanks go to Jean-Luc Corre for illustrations.

Bibliography

[1] W.W. Armstrong and M.W. Green. The dynamics of articulated rigid bodies for purposes of animation. *The Visual Computer*, 1(4):231–240, December 1985.

[2] B. Arnaldi, G. Dumont, G. Hégron, N. Magnenat-Thalmann, and D. Thalmann. Animation control with dynamics. In Proceedings of *Computer Animation'89, State-of-the-art in Computer Animation*, Springer-Verlag, editor, pages 113–123, Computer Graphics International, 1989.

[3] Y. Bamberger. *Mécanique de l'ingénieur 1 : systèmes de corps rigides.* Volume 1, Hermann, 293 rue Lecourbe 75015 Paris, 1981.

[4] A. H. Barr. Global and local deformations of solid primitives. In Proceedings of *SIGGRAPH'84*, pages 21–30, Computer Graphics, July 1984.

[5] R. Barzel and A. H. Barr. A modeling system based on dynamics constraints. In Proceedings of *SIGGRAPH'88*, pages 179–188, Computer Graphics, August 1988.

[6] M. A. Chace. Methods and experience in computer aided design of large-displacement mechanical systems. In *Computer Aided Analysis and Optimisation of Mechanical System Dynamics*, E.J. Haug, editor, pages 233–259, Springer-Verlag, 1984.

[7] R. Deyo, J. A. Briggs, and P. Doenges. Getting graphics in gear : graphics and dynamics in driving simulation. In Proceedings of *SIGGRAPH'88*, pages 317–326, August 1988.

[8] G. Dumont, B. Arnaldi, and G. Hégron. Mechanics of solids for computer animation. In Proceedings of *PIXIM'89*, pages 293–307, September 1989.

[9] A. Fournier, J. Bloomenthal, P. Oppenheimer, W.T. Reeves, and A.R. Smith. The modeling of natural phenomena. *ACM SIGGRAPH'87 Courses Notes*, 17, 1987.

[10] A. Fournier and W.T. Reeves. A simple model of ocean waves. In Proceedings of *SIGGRAPH'86*, page 75, Computer Graphics, August 1986.

[11] M. P. Gascuel. Osea : un nouveau modèle de matière pour traiter les collisions entre objets déformables. In Proceedings of *PIXIM'89*, pages 309–323, September 1989.

[12] P. Germain. *Mécanique.* Volume 1, Ecole Polytechnique, 91128 Palaiseau Cedex, 1986.

[13] J. K. Hahn. Realistic animation of rigid bodies. In Proceedings of *SIGGRAPH'88*, pages 299–308, August 1988.

[14] G. Hégron, B. Arnaldi, and G. Dumont. Toward general animation control. In Proceedings of *C.G.I'88*, Springer-Verlag, editor, pages 54–63, Computer Graphics International, Geneva, May 1988.

[15] G. S. P Miller. The motion dynamics of snakes and worms. In Proceedings of *SIGGRAPH'88*, pages 169–178, August 1988.

[16] M. Moore and J. Wilhelms. Collision detection and response for computer animation. In Proceedings of *SIGGRAPH'88*, pages 289–298, August 1988.

[17] N. Orlandea. *Development and Application of Node-Analogous Sparsity-Oriented Methods for Simulation of Mechanical Dynamic System.* PhD thesis, University of Michigan, 1973.

[18] J. C. Platt and A. H. Barr. Constraints methods for flexible models. In Proceedings of *SIGGRAPH'88*, pages 279–288, August 1988.

[19] W.T. Reeves. Particle systems. a technique for modelling a class of fuzzy objects. In Proceedings of *SIGGRAPH'83*, pages 359–376, Computer Graphics, July 1983.

[20] W.O. Schielen. Computer generation of equations of motion. In *Computer Aided Analysis and Optimisation of Mechanical System Dynamics*, E.J. Haug, editor, pages 183–215, Springer-Verlag, 1984.

[21] D. Terzopoulos and K. Fleischer. Modeling inelastic deformation : viscoelasticity, plasticity, fracture. In Proceedings of *SIGGRAPH'88*, pages 269–278, August 1988.

[22] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. In Proceedings of *SIGGRAPH'87*, pages 205–214, July 1987.

[23] D. Terzopoulos and A. Witkin. Physically based models with rigid and deformable components. In Proceedings of *Graphics Interface'88*, June 1988.

[24] G. Touzot and G. Dhatt. *Une Présentation de la Méthode des Eléments Finis.* Maloine S.A Editeur, 27 rue de l'Ecole de Medecine, 75006 Paris, 1984.

[25] J. Wilhelms and B. Barsky. Using dynamic analysis to animate articulated bodies such as humans and robots. In Proceedings of *Graphics interface'85*, pages 97–104, May 1985.

[26] A. Witkin, K. Fleischer, and A. Barr. Energy constraints on parameterized models. In Proceedings of *SIGGRAPH'87*, pages 225–232, Computer Graphics, July 1987.

[27] A. Witkin and M. Kass. Spacetime constraints. In Proceedings of *SIGGRAPH'88*, pages 159–168, August 1988.

[28] J. Wittenburg. *Dynamics of Systems of Rigid Bodies.* Teubner, Stuttgart, 1977.

[29] O. C. Zienkiewicz. *The Finite Element Method in Engineering Science, third edition.* Mc Graw-Hill, London, 1977.

Liste des publications internes 1990

PI 508 **RAY TRACING ON DISTRIBUTED MEMORY PARALLEL COMPUTERS: STRATEGIES FOR DISTRIBUTING COMPUTATIONS AND DATA.**
Didier BADOUEL, Kadi BOUATOUCH, Thierry PRIOL
Janvier 1990, 16 Pages.

PI 509 **STABILITY ANALYSIS AND IMPROVEMENT OF THE BLOCK GRAM-SCHMIDT ALGORITHM.**
William JALBY, Bernard PHILIPPE
Janvier 1990, 24 Pages.

PI 510 **TESTING FOR THE UNBOUNDEDNESS OF FIFO CHANNELS IN PROGRAMS.**
Thierry JERON
Janvier 1990, 30 Pages.

PI 511 **AUTOMATIC ANIMATION CONTROL OF PHYSICAL SYSTEMS**
Georges DUMONT, Bruno ARNALDI, Gérard HEGRON
Janvier 1990, 22 Pages.

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique