

Displacement constraints for interactive modeling and animation of articulated structures

Jean-Dominique Gascuel, Marie-Paule Gascuel

▶ **To cite this version:**

Jean-Dominique Gascuel, Marie-Paule Gascuel. Displacement constraints for interactive modeling and animation of articulated structures. The Visual Computer, 1994, 10 (4), pp.191-204. 10.1007/BF01901286 . inria-00509998

HAL Id: inria-00509998
https://inria.hal.science/inria-00509998

Submitted on 28 Jun 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Displacement constraints for interactive modeling and animation of articulated structures

Jean-Dominique Gascuel* and
Marie-Paule Gascuel*

LIENS, CNRS URA 1327,
Ecole Normale Supérieure, 45 Rue d'Ulm,
F-75230 Paris Cedex 05, France

This paper presents an integrated set of methods for the automatic construction and interactive animation of solid systems that satisfy specified geometric constraints. Displacement contraints enable the user to design articulated bodies with various degrees of freedom in rotation or in translation at hinges and to restrict the scope of the movement at will. The graph of constrained objects may contain closed loops. The animation is achieved by decoupling the free motion of each solid component from the action of the constraints. We do this with iterative tunings in displacements. The method is currently implemented in a dynamically based animation system and takes the physical parameters into account while reestablishing the constraints. In particular, first-order momenta are preserved during this process. The approach would be easy to extend to modeling systems or animation modules without a physical model just by allowing the user to control more parameters.

Key words: Constraints – Modeling – Animation – Simulation – Dynamics

* *Present address*: iMAGIS, IMAG, BP 53, F-38041-Grenoble Cedex 09, France
Correspondence to: J.-D. Gascuel

1 Introduction

Animation systems in computer graphics must provide both modeling tools to define objects and methods to generate their movements and deformations. Purely descriptive animation systems are not always sufficient to animate complex structures (e.g. solids connected by various kind of joints). Various "active models" have been proposed to cope with this problem. Some are based on a set of simplified mechanical laws and most offer an unsurpassed realism. However, there is still no universal method that adapts to the great variety of objects and of motions and that combines efficiency with simplicity of use.
This paper presents a new way of maintaining geometric constraints between solids that can be used for both automatic assembly and interactive animation of articulated structures. At each time step of an animation, the solid components of the structures are first animated as if they were independent. Then the constraints are reestablished by a series of small corrections in displacement. Correction of the displacement, rather than evaluation of the contraint forces, leads to an algorithm which can be applied either to physically based animation systems (the corrections will take physical models into account) or to systems without physical models. The displacement-constraint approach is very easy to implement, provides a simple way of defining complex structures, and produces animations at interactive rates.

1.1 Related work

The animation of articulated solids has drawn a lot of attention in the past few years. [A detailed state-of-the-art can be found in Verroust (1990)]. Featherstone (1983) introduced an animation technique for articulated chains with components that possess one degree of freedom with respect to their parent. These chains are animated by forces and torques chosen by the user. Lathrop (1986) extended this method to tree-like structures and to graphs of components.
Wilhelms and Barsky (1985) proposed an approach adapted to tree-like structures that enables rotations and translations at each joint.
Armstrong and Green (1985) present a more efficient method that is restricted to rotations at hinges. Like the former approach, this does not work if the structure contains closed loops.
Using d'Alembert's virtual work principle, Isaac and Cohen (1987, 1988) animated articulated bod-

ies the components of which can have six degrees of freedom with respect to their parent. Motion is controlled by specifying some of the accelerations (inverse dynamics) and some of the external forces.

Barzel and Barr (1988) use geometric constraints connecting independent solid components to construct complex structures. The basic idea consists in evaluating the forces caused by the action of the constraints. To do this, a system of coupled differential equations (each of them corresponding to a constraint) is numerically integrated over time. The animation sequences that are shown illustrate the use of this method for automatic assembly processes: the successive configurations of the solids obtained during the computations are displayed, and the constraints are satisfied only at the last time step. To produce a full animation of articulated structures with this approach, the numerical integration process should be applied for each frame. The authors admit that their approach, although more general, turns out to be less efficient than the methods of Armstrong and Green (1985) or of Isaac and Cohen (1987, 1988).

More recently, van Overveld (1990) introduces a simple and efficient approach, perhaps less realistic than the previous ones because it uses point dynamics only. Each object is composed of a set of points with masses, connected by distance or angle constraints (so that the mass of an articulated object is divided among the hinges). The basic idea for the animation consists in decoupling the external actions on the system from the internal actions due to constraints. Van Overveld (1991), approximates solids with families of such points, linked by constraints that maintain rigidity. Points shared by two of these solids provide hinges with three rotational degrees of freedom.

1.2 Overview

The displacement-constraints method presented in this paper attempts to find a new compromise between the efficiency and the "realism" of an animation involving articulated structures. In order to model a large variety of objects and of motions easily, we animate solid components connected by geometric constraints. The user is free to choose the number of degrees of freedom in rotation and in translation at hinges and to specify angular or linear constraints on the motion. The graph of constrained objects may contain closed loops. A technique for the automatic construction of valid initial positions is provided with the animation method.

Our basic idea is to decouple the free motion of the solid components from the action of the constraints that are met through *iterative tunings in displacements*. These displacements include small translations and small rotations. To preserve the correctness of motion in an animation system based on solid mechanics, the corrections reestablishing constraints must take the physical parameters of the solids into account. In particular, they must maintain the system first-order momenta.

The correction of displacements rather than the computation of constraint forces is the central innovative aspect of the method. Thanks to this idea, the displacement-constraint approach is not restricted to physically based systems. No integration of constraint forces is required, and the few physical parameters used for constraint processing can be replaced easily by intuitive user's parameters. The method would work in a modeling system, or in an animation module using a non-physical model (for instance, one based on inverse kinematics or on behavioral approaches). In a comprehensive animation system, displacement contraints would be particularly useful to connect physical and non-physical objects.

The remainder of this paper develops as follows: Section 2 presents the basic ideas of our approach and motivates our choices. We give general algorithms for both animation and automatic search of valid initial positions. Section 3 deals with "point to point" constraints between solids that are used to create hinges with three rotational degrees of freedom. Section 4 extends the method to constraints that enable translations at hinges, possibly limited in scope by the user and shows how rotations can be restricted to fixed angular domains. Section 5 gives the details of the implementation and the user's interface and it comments on some results. Section 6 concludes and discusses the work in progress.

2 The displacement-constraint approach

2.1 Basic choices

Our method deals with independent solids linked by geometric constraints, expresses the action of

these constraints by corrections to the displacements, and solves the equations by decoupling the action of the constraints. Let us explain these choices.

2.1.1 For the models

Armstrong and Green (1985) represented articulated solids with tree-like structures, Barzel and Barr (1988) considered sets of independent solid components connected by contraints, and van Oberveld (1991) used only punctual masses.
We have chosen the second approach. Considering solids connected by geometric constraints is a very general way of modeling complex structures (these structures may contain closed loops, and even if they do not, there is no reason why one of the solids should play a particular role, such as the root of a tree). In addition, and contrary to van Overveld (1991), we prefer to use solid mechanics rather than expressing rigidity with an extra constraint between elementary masses.

2.1.2 For the way of expressing the action of constraints

Barzel and Barr (1988) treated geometric constraints by computing the reaction forces maintaining them. Then, these forces were integrated in the equations of motion. Van Overveld (1990, 1991) preferred to correct the momenta of the solids that were integrated to find new positions.
Our approach is different. We still want to find a motion correction that takes constraints into account. Nevertheless, we prefer to express our algorithm directly in terms of *adjusting displacements.* This point of view, which avoids the explicit computation of constraint forces, yields more efficent and more general algorithms:

– In an animation system based on dynamics, our corrections take into account the physical models of the solids. They are somewhat equivalent to the addition of some constraint forces, the values of which could be derived easily at each time step (but they are useless; directly adjusting displacements is more efficient).
– The avoidance of an explicit computation of forces gives a much more general aspect to our constraint processing. In fact, the algorithm would still work in an animation system with no physical model, for instance, in a system based on inverse kinematics, or on behavioral approaches. Simply, our corrections would no longer be based on physical data, but on parameters controlled by the user.

2.1.3 For the way of solving the equations

Two main approaches have been used to maintain constraints between independent components:

– The solution of a global system of coupled differential equations, each one associated with a constraint. This kind of approach was used by Barzel and Barr (1988).
– The decoupling of the action of constraints and of the other external actions. To do this, the components are first animated as if they were independent. Then, the constraints are taken into account by associating a set of corrections with each of them and by using a combination of the corrections with respect to each component. This process is iterated until all the corrections have become very small.

Van Overveld (1990; 1991) claimed that the latter approach avoids the heavy and complex computations associated with the former and that it offers a sufficient physical credibility, provided that the first-order linear and angular momenta are conserved during the constraint processing. We have chosen to experiment with a technique of that kind that can be implemented and tested very easily.

2.2 The animation algorithm for displacement constraints

In the remainder of this paper, all points, vectors and matrices are expressed in worldfixed coordinates.
The general algorithm for animation under constraints is directly derived from our three basic choices. To compute the vew positions and orientations of the solids at time $t+dt$ from a configuration at time t that satisfies the constraints:

1. Compute the new positions and orientations of the solids as if they were independent. In an animation system based on dynamics, this is done by integrating the Euler equations of motion (Goldstein 1983) over time for all the externally applied

forces and toques that are not due to constraints. For each solid:

$$\vec{v}(t+dt)=\vec{v}(t)+\frac{\sum \vec{F}(t)}{m}dt \quad (1)$$

$$\vec{\omega}(t+dt)=\vec{\omega}(t)+J^{-1}(\sum \vec{\mathcal{M}}(t)-\vec{\omega}(t)\wedge J\vec{\omega}(t))\,dt \quad (2)$$

$$\vec{x}(t+dt)=\vec{x}(t)+\vec{v}(t+dt)\,dt \quad (3)$$

$$R(t+dt)=(I+dt\,\tilde{\omega}(t+dt))\,R(t) \quad (4)$$

where m and J are the mass and tensor of inertia of the solid, respectively, $\vec{F}$ represents the external forces and $\vec{\mathcal{M}}$, the external torques with respect to the body's center of mass; $\vec{v}$ and $\vec{\omega}$ are the linear and angular speed vectors, respectively; $\tilde{\omega}$ is the rotation speed matrix "dual" of $\vec{\omega}$ (characterized by: $\forall \vec{a}\,\tilde{\omega}\vec{a}=\vec{\omega}\wedge\vec{a}$); $\vec{x}$ is the translation vector and R, the orientation matrix defining the current configuration of the solid.

This integration scheme for Euler equations of motion uses rotation vectors for small rotations during dt, which is a second-order approximation. If the rotations are larger than a given limit, the system goes back in time, and the time step is reduced. Furthermore, the matrix $R(t+dt)$ must be reorthogonalized just after using Eq. 4.

2. Take the constraints into account by slightly displacing the solids during a series of iterations (these displacements include small rotations and small translations). Because the constraints were satisfied at time t, just before the free motion during dt, the corrections reestablishing a given constraint cannot be larger than the sum of the constrained displacements of the solids during dt, so the formalism of the rotation vectors is also used to define the small rotations due to constraints. Sections 3 and 4, which deal with various kinds of constraints, detail the way we associate small displacements to each of them, and the way these displacements are combined when a solid is subjected to several constraints simultaneously.

Stop when all the resulting corrections are smaller than a specified limit, or when a maximum number of iterations is reached. Limiting the number of iterations avoids deadlocks when the system is over-constrained. When this maximum is reached, the system continues with the animation, thus introducing a configuration that does not exactly satisfy the constraints. A warning is issued, and the user can choose either to continue with the animation (for instance, if the problem is temporary, and the divergences are not too large) or to stop the animation and modify the set of constraints.

3. Adjust the linear and angular speeds of the solids by considering the positions and orientations that they have effectively reached during a time step (the corrections due to constraints must be taken into account in the kinematics of movement, as if we had added constraint forces):

$$\vec{v}(t+dt)=(\vec{x}(t+dt)-(\vec{x}(t))/dt$$

$$\tilde{\omega}(t+dt)=(R(t+dt)\,R^{-1}(t)-I)/dt$$

2.3 Automatic search of valid initial positions

The provision of an easy way to build models is very important in an animation system. In our framework of solids subjected to constraints, it would be very difficult to find valid initial positions of the objects manually. Please note that we really need such positions to start using our animation algorithm. Indeed, if the constraints were too far from being satisfied in the initial configuration, corrections of displacements during a small time interval that are too large would add extremely large values to the initial speeds.

Fortunately, a method for the automatic building of initial positions that satisfies the constraints can be derived from our approach easily. We just iterate our usual treatment of constraints (step 2 of the animation algorithm) from the arbitrary positions given by the user, but without updating the speeds of the solids when a solution is found.

During this process, the displacements computed at each time step must not be too large, because rotations are again expressed with rotation vectors. To keep the integration errors at an acceptable level, the convergence speed must not be too large (we will see in Sect. 3 that this speed can be controlled by the user, through a parameter α). As a larger number of iterations is needed to reach a valid configuration based on arbitrary positions of the solids, the maximum number of iterations is larger than that of an animation process.

3 "Point to point" constraints

This section describes the method of correcting displacements within the framework of the basic "point to point" constraints. Section 4 will extend

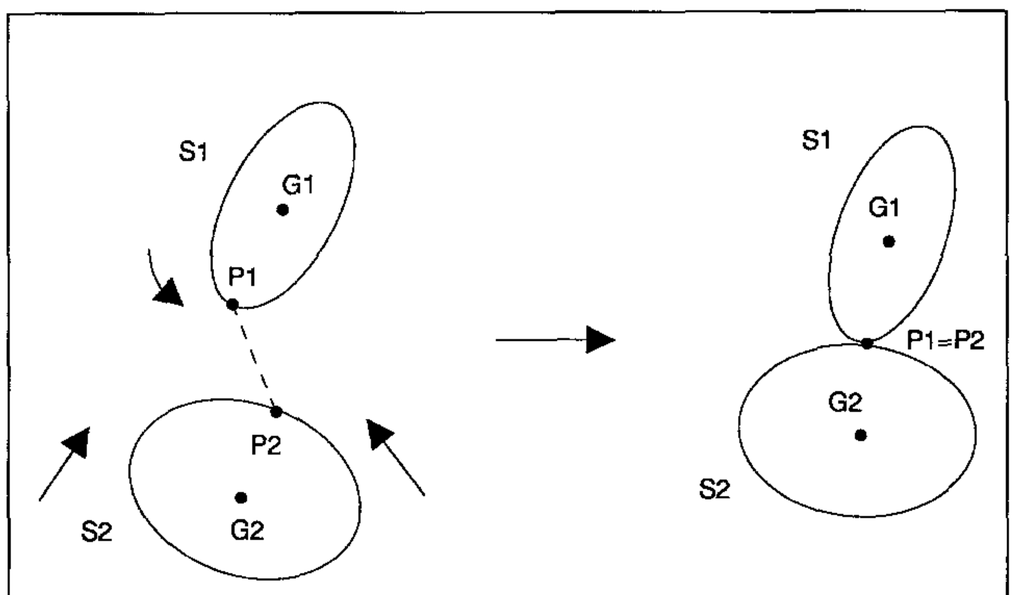


Fig. 1. A "point to point" constraint between two solids

the results to more complex constraints between solids.

With the same terminology as that of Barzel and Barr (1988), a "point to point" constraint between two solids S_1 and S_2 creates a joint with three degrees of freedom in rotation by making two points, P_1 of S_1 and P_2 of S_2, coincide during the overall movement (Fig. 1).

To maintain a set of "point to point" constraints during an animation sequence, two questions must be answered: How can we find reasonable displacements to satisfy a given constraint? How can we combine these displacements for a solid subjected to several constraints simultaneously?

3.1 Displacements associated with a "point to point" constraint

Satisfying a given "point to point" constraint by translating solids is always possible, but used alone, it would produce unrealistic behaviours. In consequence, the displacements we associate with a constraint must include both rotation and translation.

As we said previously, it seems natural to use the physical models of the solids (when we have them) while correcting their displacements. Indeed, the heaviest object must move less than the lightest. In addition, the relative amounts of rotation and translation applied to each solid must correspond to its mass and tensor of inertia.

3.1.1 Finding the right amount of rotation versus translation

To take the physical parameters of the solids into account while defining small rotations and small translations, we compare the action of a "point to point" constraint and the action of a piece of rubber that would pull point P_1 to point P_2 (Fig. 1).

Let J_1 and J_2 be the tensors of inertia of the solids at time t expressed in world-fixed coordinates; m_1 and m_2, their masses; and G_1 and G_2, their centers of mass.

A rubber of stiffness k connecting P_1 to P_2 would produce on S_1 and on S_2 the instantaneous forces and torques:

$$\vec{F}_1 = k(P_2 - P_1) \tag{5}$$

$$\mathcal{M}_{G1}(P_1, \vec{F}_1) = (P_1 - G_1) \wedge \vec{F}_1 \tag{6}$$

$$\vec{F}_2 = -\vec{F}_1 \tag{7}$$

$$\mathcal{M}_{G2}(P_2, \vec{F}_2) = (P_2 - G_2) \wedge \vec{F}_2 \tag{8}$$

If we want to consider the lasting action of this rubber over time, these values for forces and torques should be integrated in Euler equations of motion to determine the resulting displacements. This would lead to complex equations (where parameters P_i, G_i and J_i, although constant in the local coordinate system, would be functions of time because they are expressed in world-fixed coordinates) closely related to those used by Barzel and Barr (1988). (Note that Barzel does not decouple the action of individual constraints while solving the equations.) To generate animations at interactive rates, more approximations must be made.

To determine an expression for the displacements associated with a constraints, we integrate the action of the rubber during a small time interval Δt *that is considered to be small enough to enable the use of finite differences.* This is equivalent to neglecting the motion of the solids during Δt. This approximation is clearly justifed because the displacements that we are looking for will be smaller or equal than the displacements that produced the violation of the constraints during the dt for which the use of a finite difference integration of Euler equations of motion was justified.

The extra terms added by the force (Eq. 5) and the torque (Eq. 6) to the first solid's equation of motion (Eqs. 1 and 2) are:

$$\Delta\vec{v}_1 = \frac{k\Delta t}{m_1}(P_2 - P_1) \tag{9}$$

$$\Delta\vec{\omega}_1 = \Delta t\, J_1^{-1}\, k(P_1 - G_1) \wedge (P_2 - P_1) \tag{10}$$

(similar expressions are valid for the second solid). These extra amounts of linear speed and rotation speed produce the extra displacements:

$$\Delta\vec{R}_1 = J_1^{-1}\, \frac{k\Delta t^2}{2}(P_1 - G_1) \wedge (P_2 - P_1) \tag{11}$$

$$\Delta\vec{x}_1 = \frac{k\Delta t^2}{2m_1}(P_2 - P_1)\,. \tag{12}$$

These expressions provide us with the relative values of the rotation and translation that correspond to the mass and the tensor of inertia of the solid. More precisely, the amount of rotation can be deduced from the amount of translation of the solid:

$$\Delta\vec{R}_1 = m_1\, J_1^{-1}(P_1 - G_1) \wedge \Delta\vec{x}_1 \tag{13}$$

In addition, the rotations and the translations computed with this method for the two solids are derived from the action of the opposite forces and torques (Eqs. 5–8), so that they conserve the first-order linear and angular momenta[5] of the system (S_1, S_2).

3.1.2 Scheme for applying the small rotations and small translations

We want to use Eqs. 11 and 12 to define the small rotations and translations associated with the constraints. Of course, any value can be taken for the rubber's stiffness k, so these expressions can be rewritten:

$$\Delta\vec{R}_1 = \alpha J_1^{-1}(P_1 - G_1) \wedge (P_2 - P_1) \tag{14}$$

$$\Delta\vec{x}_1 = \frac{\alpha}{m1}(P_2 - P_1) \tag{15}$$

where α is a scalar parameter that can be fixed by the user and that is related to the convergence speed and to the correctness of the motion. Indeed, taking a very small value for α during the iterative constraint processing would correspond to a precise integration over time of the action of the rubber between constrained points, but this small value would also lead to a low convergence speed, so a sompromise must be found.

In practice, a different value for α can be chosen for each constraint between a pair of solids, and the rotations and translations are computed and applied in a sequential way. When a single constraint is specified, the constraint processing scheme consists in iterating:

1. Apply the small rotation of Eq. 14
2. Apply the small translation of Eq. 15

until the constraint is verified (Sect. 3.2 will extend this scheme to multiple-constraints situations).

3.1.3 A possible criteria for choosing α:

To solve the simplest situations very quickly, the best method would be to find a computation scheme that satisfies the constraint after the first iteration of the constraint processing, provided that no other constraint is applied to the solids. This gives us a good criterion for choosing α, but obliges us to modify the iterations scheme slightly. Indeed, "point to point" constraints cannot always be satisfied by pure rotation, but are easy to satisfy exactly with an adequate translation of the solids. If we wanted to have:

$$P_1 + \Delta\overrightarrow{x_1} = P_2 + \Delta\overrightarrow{x_2} \tag{16}$$

taking the following value for α would be sufficient:

$$\alpha = \frac{m_1\, m_2}{m_1 + m_2} \tag{17}$$

In practice, we want to combine this translation (which gives an exact result) with a small rotation. To obtain an exact solution for the constraint in one iteration, we use a new computation scheme:

1. Apply the small rotations:

$$\overline{\Delta\vec{R}_1} = \frac{m_1\, m_2}{m_1 + m_2}\, J_1^{-1}(P_1 - G_1) \wedge (P_2 - P_1)$$

$$\overline{\Delta\vec{R}_2} = \frac{m_1\, m_2}{m_1 + m_2}\, J_2^{-1}(P_2 - G_2) \wedge (P_1 - P_2) \tag{18}$$

2. Then, from the positions P_1', P_2' of the points *after rotation*, compute and apply the small translations exactly satisfying the constraint:

---

[5] *Let* $G = m_1 G_1 + m_2 G_2$ be the center of mass of the system (S_1, S_2). The linear momentum of the system is $(m_1 + m_2)\vec{v}_G = m_1\vec{v}_1 + m_2\vec{v}_2$, and the angular momentum is $J\vec{\omega}_G$, where the inertia tensor J is computed with respect to G. These quantities are conserved by the action of internal opposite forces applied on S_1 and S_2, when no external force or torque is acting on the system (Goldstein 1983).

$$\overrightarrow{\Delta x_1} = \frac{m_2}{m_1 + m_2}(P_2' - P_1')$$

$$\overrightarrow{\Delta x_2} = \frac{m_1}{m_1 + m_2}(P_1' - P_2') \tag{19}$$

Note: This algorithm modifies the amounts of translaton versus rotation slightly, because the translation vectors are no longer computed from the same positons of P_1 and P_2 as those of the rotation vectors. Then, although it takes the physical data into account, the second scheme is considered a good heuristic approach rather than a close approximation of the exact motion. Nevertheless, it gives very satisfactory results in practice (motion computed with this method "seems realistic") and leads to very fast convergence rates for constraint processing (for the "monocycle" example of Sect. 5.3, this second scheme for constraint processing converges most of the time with less than six iterations per frame).

3.1.4 Extension to animation systems using no physical data

In an animation system with no physical model, constraints speed can be iteratively maintained with a good convergence by using the second computation scheme, where physical data are replaced by intuitive user's parameters:

- The factor $m_2/(m_1+m_2)$ in the translation of S_1 (Eq. 19) is proportional to the amount of translation of S_1 compared with that of S_2. It can be replaced easily by a user parameter μ_{12} associated with a given constraint, verifying $0<\mu_{12}<1$ and specifying the "ability to move" of one solid relative to the other (the factor for translation of S_2 would be $\mu_{21}=1-\mu_{12}$).
- The expression $m_1 J_1^{-1}$, which can be factorized together with μ_{12} in the S_1 rotation (Eq. 18), controls the amount of rotation of S_1 versus translation. This matricial expression can be replaced with a positive scalar parameter[6] associated with each solid provided that its "ability to rotate" is specified.

[6] Replacing the inertia tensor with a scalar is equivalent to considering a spherical mass distribution.

3.2 Combining displacements due to individual constraints

In practice, several constraints can be applied to a solid simultaneously, so that rotations and translations due to particular constraints must be combined. To find the displacement due to a given set of constraints, we use a linear combination of the particular displacements. Although rotations are not commutative in general, we have computed "small rotations" approximated by rotation vectors. One of the advantages of this formulation is that small rotations can be combined in a commutative way, exactly as for translations: the resulting rotation vector can be defined by a weight sum of rotation vectors due to individual constraints.
At each iteration of the constraint processing:

– We first compute and combine the small rotations.
– Then, we do the same thing for the translations.

Depending on which scheme is used, the translations can be computed from the orientations of the solids either before or after rotation.
Our way of combining rotations and translations must preserve the property of conservation of the first-order momenta obtained when a single. A sum of displacements would work [this solution, which corresponds to summing the constraints forces applied on each solid, was used by van Overveld (1990; 1991)]. Nevertheless, this method would lead to divergences in some cases: suppose that four solids S_0 to S_3(S_1, S_2 and S_3 being located in the same place) are linked by three "point to point" constraints between G_0 and each of G_1, G_2, G_3 (Fig. 2). With this configuration, summing the three translations computed for S_0 (the mass of which is assumed to be half the mass of the other solids) and displacing S_1, S_2 and S_3

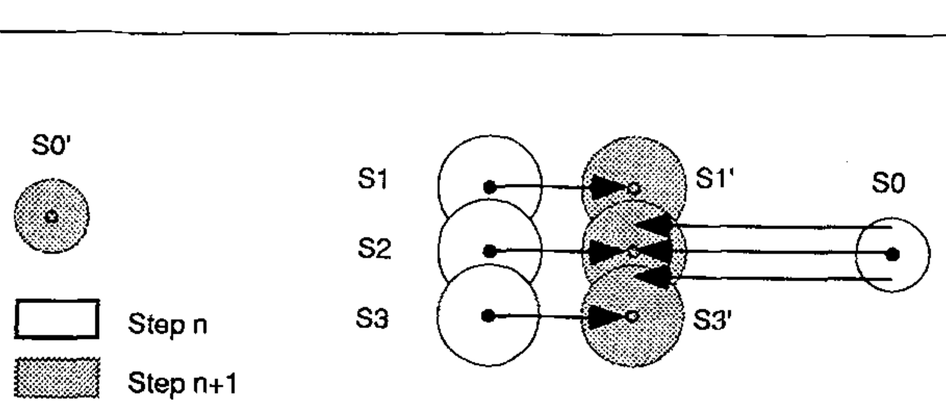


Fig. 2. For these solids, summing the translations associated with each constraint would move them farther away from a solution at each iteration

would move them farther and farther from a solution at each iteration of the constraint processing.

To avoid this problem, we weight the displacements affected by the solids before summing them. In order to conserve the first-order momenta, the same weighting factor must be used for couples of displacements due to the same constraint. If S_i and S_j are two objects linked by a given constraint, we weight the associated displacements with:

$$\frac{1}{\max(n_i, n_j)}$$

where n_i and n_j are the numbers of constraints applied to the solids, respectively.

Let us explain briefly why this factor works by studying more precisely the simple cases where constrained points P_i coincide with the G_i, as in Fig. 2 (in these cases, our algorithm only produces translations of the solids). In the following proof, the value $m_i m_j/(m_i+m_j)$ is used for the parameter α associated with the constraint between S_i and S_j. However, the same would be true at least for any other value smaller than m_i and m_j (the key point is that the coefficients λ_{ij} must be smaller than unity).

Let us call G_i^n the position of G_i after n iterations of the constraint processing, and C_i the set of indices j for which the solid S_j is constrained with S_i (by convention, $i \in C_i$). The weighted sum of translations gives:

$$G_i^{n+1} = \sum_{j \in C_i} \lambda_{ij} G_j^n \tag{20}$$

where:

$$\lambda_{ij} = \frac{m_i}{m_i+m_j} \frac{1}{\max(n_i, n_j)} \quad \text{if } j \in C_i - \{i\}$$
$$\lambda_{ii} = 1 - \sum_{j \in C_i - \{i\}} \lambda_{ij} \tag{21}$$

The coefficients λ_{ij} sum to one, and verify:

$$\forall i, \forall j \in C_i, \quad 0 < \lambda_{ij} < 1 \tag{22}$$

This is quite evident if $i \neq j$. To determine it for λ_{ii}, we just have to note that if $j \neq i$, $\lambda_{ij} < (1/n_i)$, and that n_i is the cardinal of the set of indices $C_i - \{i\}$.

Thus, with our choice for the weighting factors, G_i^{n+1} is a barycenter of the $(G_j^n)_{j \in C_i}$. Because of the strict inequalities of Eq. 22, it lies in the interior of the convex hull of these points, and the situation in Fig. 2 can no longer occur.

Note: A formal convergence proof of the constraint processing would be quite hard to formulate. It would involve both characterizing over-constrained situations (for which no solution can be found) and dealing with constrained systems with an infinity of valid solutions. Even for systems with exactly the right number of constraints, a formal proof taking the corrections in rotation and in translation into account would be considerably more involved. Nevertheless, we can note that the topology of the constraint graph does not appear in the equations nor in the resolution scheme. The partial proof described here works just the same way in a configuration with or without closed loops, and it appears to be the same for a more general proof. In practice, we have applied the displacement-constraint method in various situations with very different topologies for the graph of constrained solids. We had no problems with convergence, even in complex situations such as the one in Fig. 8.

4 Extensions to other constraints

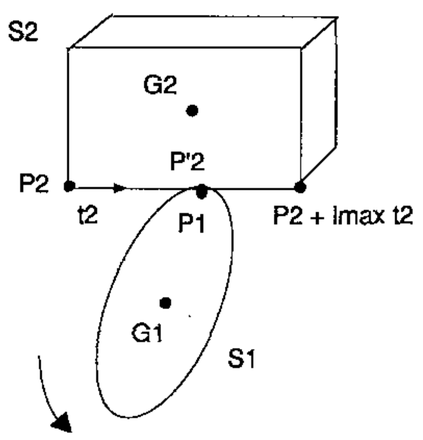
This section presents heuristics that extend the "point to point" constraint method to more complex relationships between connected solids. First of all, we study two constraints that introduce a number of degrees of freedom in translation, and then we present angle and twist constraints, which provide a precise control of the rotations at hinges. Translation and rotation constraints can be used together, so that any number and combination of degrees of freedom can be chosen for each joint.

4.1 Translations at hinges, possibly user-restricted in scope

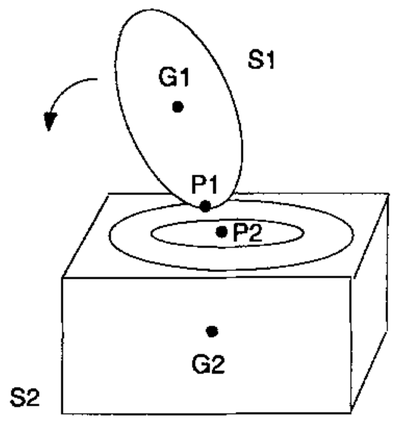
4.1.1 "Point to segment" constraints

These constraints give one degree of freedom in translation at joints, possibly user-restricted to a given interval. The user chooses two points P_1 and P_2 on the solids as before and defines a segment passing through P_2 by a unit vector $\vec{t}_2$ linked to S_2 and by a scalar interval $[\lambda_{\min}, \lambda_{\max}]$ (that can be $[-\infty, +\infty]$ if needed). Then, the constraint is:

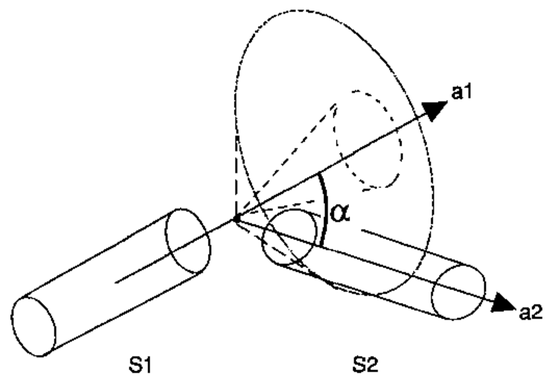
$$P_1 = P_2 + \lambda \vec{t}_2 \quad \text{with} \quad \lambda \in [\lambda_{\min}, \lambda_{\max}].$$



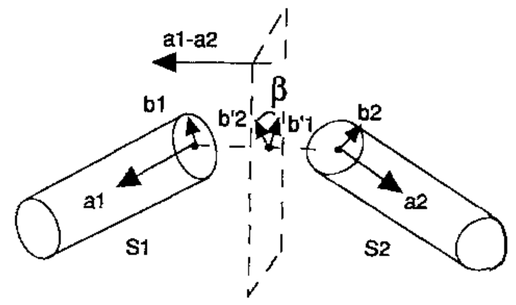
3a



b



4a



b

Fig. 3. a A "point to segment" constraint. **b** A "point to ring" constraint

Fig. 4. a Angle and **b** twist constraints between solids

The displacements of S_1 and S_2 associated with this constraint at each iteration of the constraint processing are computed by replacing P_2 in the equations that give the small rotation and the small translation with the point P_2' of the segment $[P_2+\lambda_{\min}\vec{t}_2, P_2+\lambda_{\max}\vec{t}_2]$ that is the closest to P_1. Thus, S_1 will be free to translate along the permissible segment of S_2 (Fig. 3a).

4.1.2 "Point to ring" constraints

The "point to ring" constraints allow a point P_1 of S_1 to translate in a plane linked to S_2, with a possible limitation to a disc or to a ring-shaped domain. The user defines the plane with $(P_2, \vec{n}_2)$ (normal vector), and gives an interval $[r_{\min}, r_{\max}]$ to limit the distance between P_1 and P_2 (Fig. 3b). If $r_{\min}=0$, the domain is a disc.
To compute the displacements associated with this constraint, P_2 is replaced in Eqs. 18 and 19 with the point P_2' of the ring-shaped domain that is the closest to P_1 (P_2' is the projection of P_1 onto the plane if this projection lies inside the ring; elsewhere, P_2' is the point of the ring that is the closest to the projection).

4.1.3 Other translation constraints

The movement of P_1 with respect to S_2 can be easily constrained to any other area defined with respect to S_2 (and of dimension 1, 2, or 3), as long as an efficient procedure is known to compute the nearest permissible point P_2'. In particular, "point to curve," "point to surface," "point in sphere" constraints can be useful.

4.2 Restricting rotations at hinges

To decrease the number of permissible rotations, or to limit their scopes, the user can combine the translation constraints defining a joint with angle and twist constraints.

4.2.1 Angle constraints

The user defines two unit vectors, $\vec{a}_1$ and $\vec{a}_2$, respectively linked to S_1 and S_2, and specifies an angle constraint:

$$\widehat{\vec{a}_1\,\vec{a}_2} \in [a_{\min}, a_{\max}] \subseteq [0, \pi].$$

If $a_{\min}=a_{\max}$, this forces $\vec{a}_2$ to lie on the surface of a cone of axis $\vec{a}_1$. Otherwise, S_2 still has three degrees of freedom in rotation with respect to S_1, but $\vec{a}_2$ must stay in the difference of two cones, fixed with respect to S_1 (Fig. 4a).
To find the correction of displacements at a joint subjected to an angle constraint, we use the following heuristic: the rotation vectors $\Delta\vec{R}_1$ and $\Delta\vec{R}_2$ are first computed according to the main joint constraint (for instance "point to point," "point to segment," or "point to ring"). Then, if $\widehat{\vec{a}_1\,\vec{a}_2}$ does not lie in the given interval, rotations of the solids in the plane $(\vec{a}_1, \vec{a}_2)$ are added (we use their inertia J_1 and J_2 to conserve angular momenta). The translations are computed in the usual way.

4.2.2 Twist constraints

The user still gives $\vec{a}_1$ and $\vec{a}_2$ of S_1 and S_2 and defines two other vectors $\vec{b}_1$ and $\vec{b}_2$ of the solids, respectively perpendicular to $\vec{a}_1$ and to $\vec{a}_2$. Let $\vec{b}_1'$

and $\vec{b}'_2$ be the projections of $\vec{b}_1$ and $\vec{b}_2$ on the median plane, of normal $(\vec{a}_1 - \vec{a}_2)$ (Fig. 4). The user limits the twist by giving an angular interval $[b_{\min}, b_{\max}]$ for the angle between $\vec{b}'_1$ and $\vec{b}'_2$.
To correct the rotation associated with this joint, the solid S_1 (or S_2) is possibly rotated around $\vec{a}_1$ (or $\vec{a}_2$) according to the twist constraint (again, the inertia of the solids is used to conserve the angular momenta).

4.2.3 Combining angle and twist constraints

Angle and twist constraints often need to be combined, for instance, to define a complex hinge in an articulated body. Then, the two constraints can be specified with respect to the same vectors $\vec{a}_1$ and $\vec{a}_2$. To correct the rotation vectors associated with the joint, rotations in the plane $(\vec{a}_1, \vec{a}_2)$ are added first in order to satisfy the angle constraint. Then rotations around $\vec{a}_i$ are computed according to the twist constraint (this second rotation does not modify the angle $\widehat{\vec{a}_1 \vec{a}_2}$).
Angle and twist constraints can also be very useful in situations where there is no hinge and no translation constraint between the solids. The "monocycle" example of Sect. 5.3 demonstrates this point.

Note: The complete suppression of the rotations between two solids may be useful. This can be done easily by choosing $(b_{\min} = b_{\max})$ and $(a_{\min} = a_{\max} = 0)$ in an angle and twist constraint.

5 Implementation and results

5.1 Animation framework

We have implemented the displacement-constraints method as an extension of the animation system described by Gascuel et al. (1991)[7]. This system is based on solids mechanics and enables one to coat objects with deformable material.
More precisely, a solid is structured in:

- A "kernel" that contains the data needed for animating its local coordinate system. Kernels can be immobile, follow a predefined trajectory, or be dynamically animated by the integration of Euler equations of motion over time.
- The material, rigid or deformable, that constitutes the object (defined in the local coordinate system). A detailed description of our model for deformable material can be found in Gascuel et al. (1991). Its main features are the simple and efficient control of deformations and the automatic response to collisions that it provides.
- The "skin" surface controlled by the deformations of underlying material. A chain of objects can be coated with the same continuous skin, which is useful for solids connected by hinges [we often use bicubic spline surfaces (Fig. 5)].

5.2 User interface

In the current implementation, we use a description language with a set of simple key words to define objects and constraints. A full example description file is given in Appendix A. The syntax for describing the set of constraints connecting a pair of solids is the following:

```
constraint
    object1 name;     (default to world)
    object2 name;
    hinge   P1 P2;    (default to none)
    axial   u1^N {min λ1} {max λ2};
                (default to no translation)
    planar  n1^N {min ρ1} {max ρ2};
                (default to no translation)
    angle   a1 a2 {min α1} {max α2};
                (default to no angle constraint)
    twist   b1 b2 {min β1} {max β2};
                (default to no twist constraint)
end
```

Many situations require constraints that vary over time. Specifying these dynamic constraints can be done easily with our description format by specifying in which time interval each constraint is valid.
Describing contraints with this language has been a good way of experimenting very rapidly with various kinds of constraints. In the future, we plan to implement a graphic interface that will take advantage of the intuitive geometrical meaning of the parameters and options (hinge, axis, plane, ...).

[7] In the previous version of this system, articulated objects were animated with Armstrong and Green's technique (1985). This method was quite efficient, but limited to hinges with three degrees of freedom in rotation, and none in translation. In addition, angle and twist constraints could not be specified.

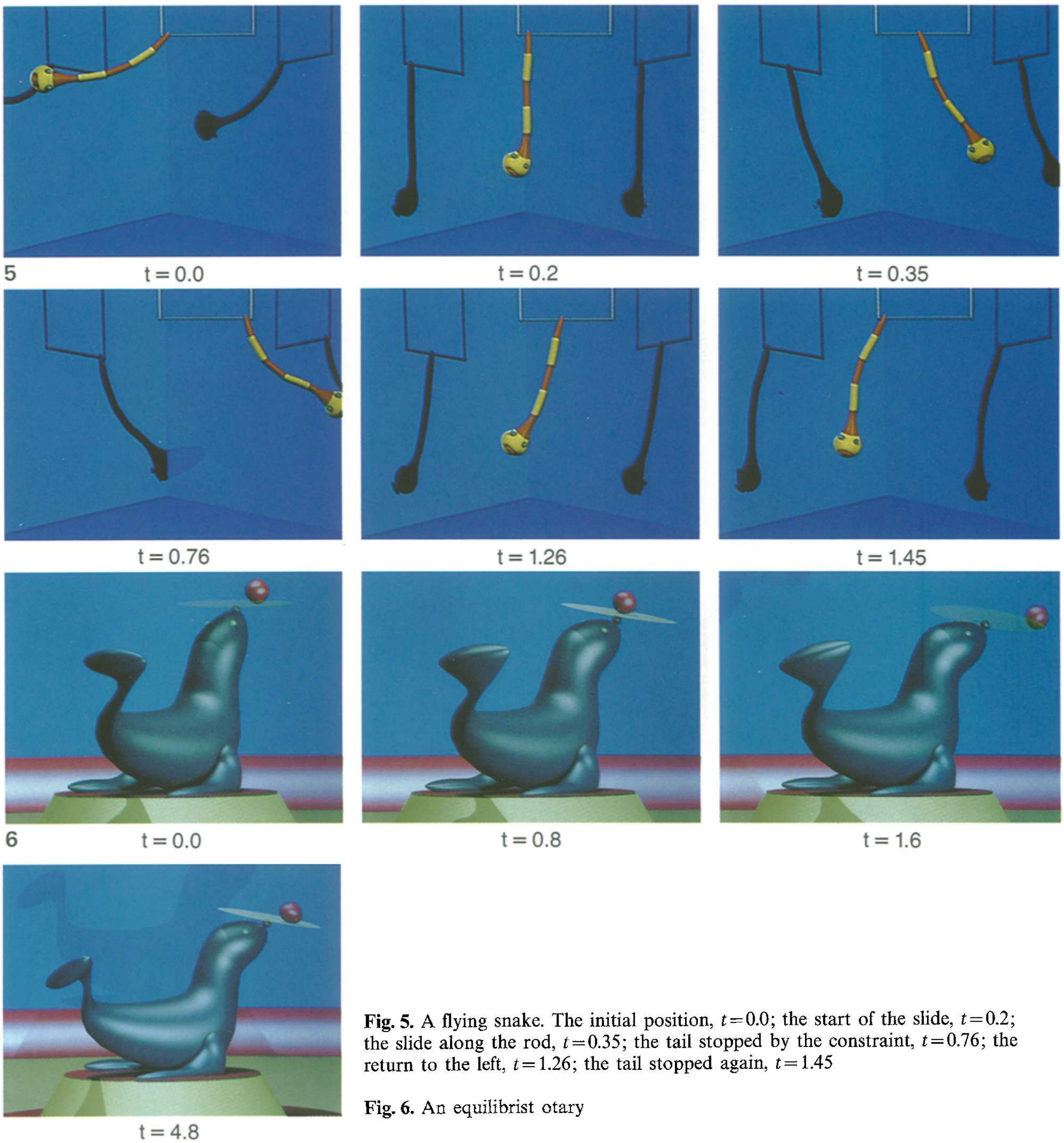


Fig. 5. A flying snake. The initial position, $t=0.0$; the start of the slide, $t=0.2$; the slide along the rod, $t=0.35$; the tail stopped by the constraint, $t=0.76$; the return to the left, $t=1.26$; the tail stopped again, $t=1.45$

Fig. 6. An equilibrist otary

5.3 Samples of animations

5.3.1 Flying snake

The snake animated in Fig. 5 is composed of six deformable solids connected by five "point to point" constraints and covered by the same continuous skin. A "point to segment" constraint allows the tail of the snake to slide along a rigid rod. Please note the complex motion of the snake: the tail slides first to the right because of the combined action of gravity and of the constraints on the snake. Then it is stopped by the "point to segment" constraint, which makes the snake swing. Finally,

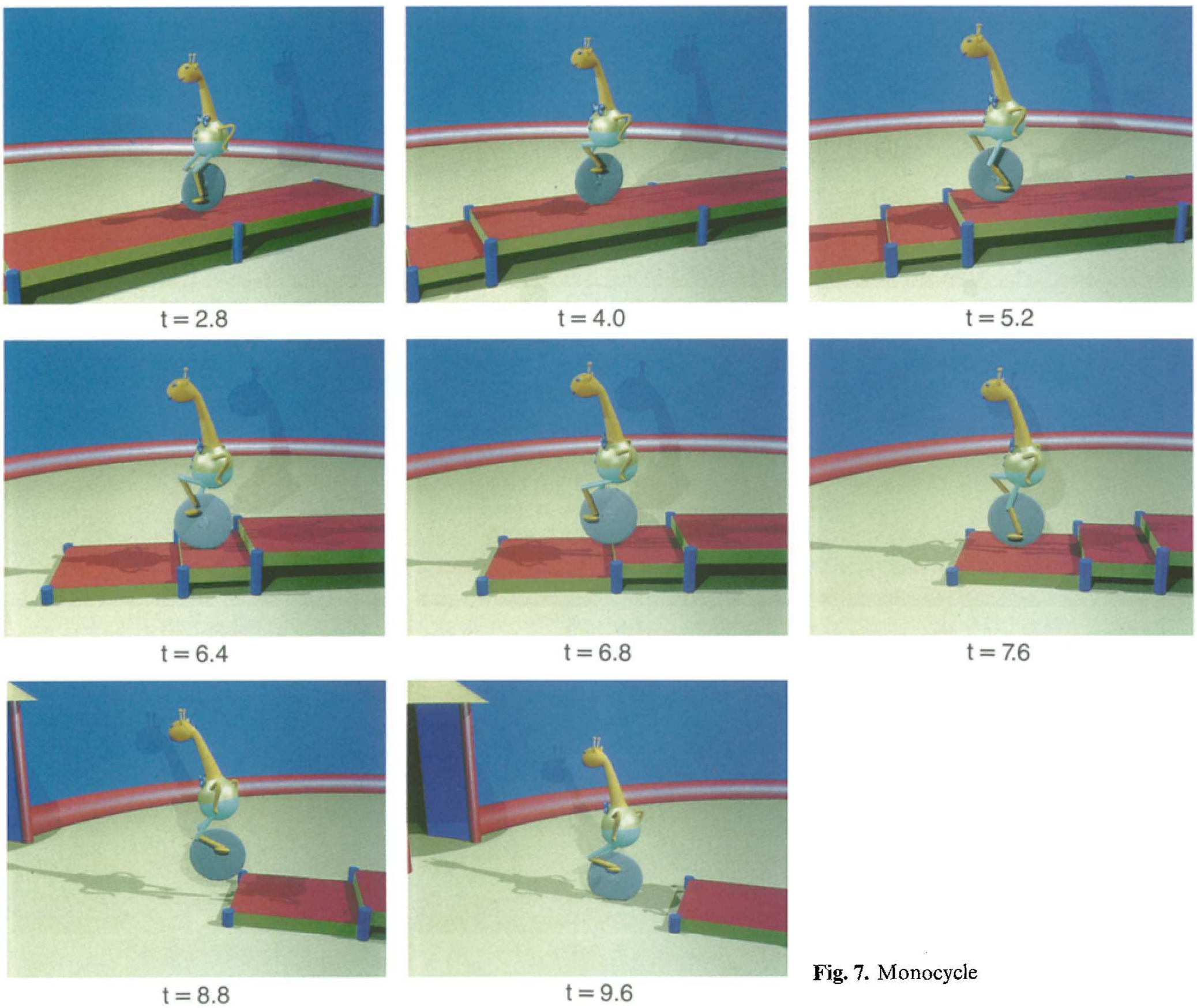


Fig. 7. Monocycle

the snake flies back to the left, which makes the tail slide again.

5.3.2 Equilibrist otary

This animation illustrates the use of a "point to ring" constraint. In this particular case, the ring is a disc: a deformable ball is constrained to stay on a tray held by an otary.
The otary is composed of three deformable components, the head, the body, and the tail, connected by "point to point" angle and twist constraints and covered by a continuous skin. The tray is maintained on the otary's nose by a "point to point" constraint. An angle constraint with the world-fixed coordinate system keeps it almost horizontal. During the animation, various external forces and torques are applied to the otary's head and tail to make the system move.

5.3.3 Monocycle

This animation involves six "point to point" and angle constraints, two "point to point"-only constraints, and three angle-only constraints. The graph of constrained objects, depicted in Fig. 8, includes several closed loops.
The animation sequence shown in Fig. 7 has been generated simply by applying a horizontal force to the character's body. A "point to point" con-

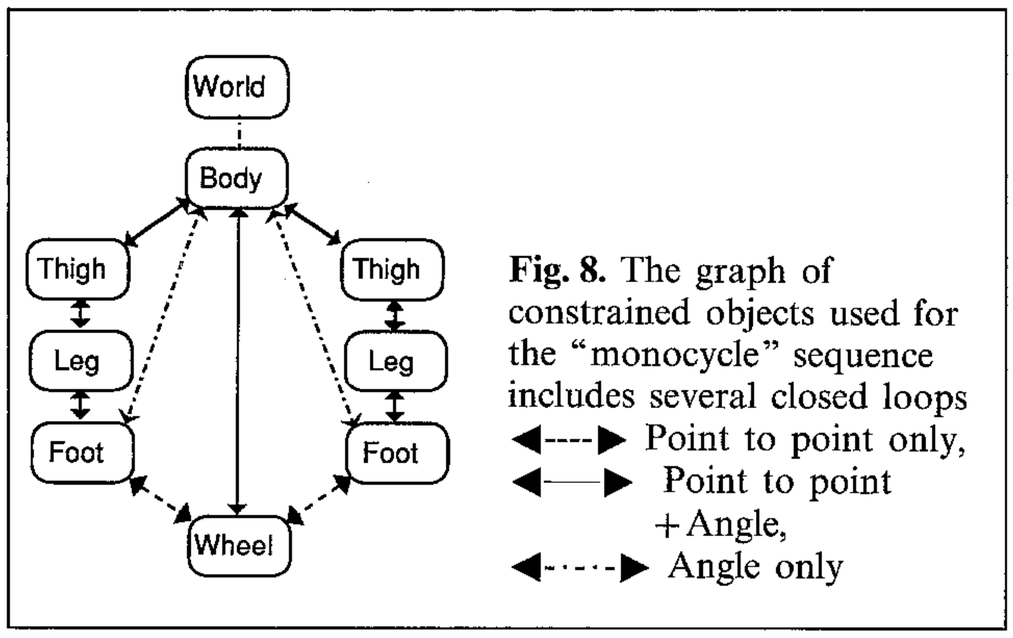


Fig. 8. The graph of constrained objects used for the "monocycle" sequence includes several closed loops

straint between the body and the center of the wheel generates the motion of the wheel, but because of its interactions with the floor, the wheel rolls rather than slides: the wheel is coated with deformable material, and the model used provides an automatic response to collisions and to lasting contacts (Gascuel et al. 1991), so that the rotation of the wheel is automatically generated by friction with the floor. This rotation is transferred to the legs through the action of the constraints. More precisely, the feet are constrained to stay on the wheel, and an angle constraint keeps them almost horizontal. "Point to point" and angle constraints connect the different components of the legs. Finally, an angle constraint between the world-fixed coordinate system and the character's body prevents the monocyclist from falling over the floor.

More complex than the previous ones, this animation is still interactive and did not raise a convergence problem. The full description file used to specify the constraints is given in Appendix A.

6 Conclusion

We have presented an integrated set of methods for building and animating systems of solids connected by geometric constraints. Our techniques, based on iterative corrections of displacements, are easy to implement and enable the simulation a large variety of objects and of motions. Any number of translations or rotations can be allowed at each joint between solids. Each degree of freedom can be limited in scope by the user. The method still works when the graph of constrained objects contains closed loops.

As in Barzel and Barr (1988), an automatic technique for assembling the objects before animation is provided. The construction of animation models, very hard to achieve manually, is significantly simplified. In modeling systems, our technique would facilitate the independent modeling of the different components of a complex scene. This components would be positioned without requiring the use of physical modeling for the solids by geometric constraints.

Displacement constraints could be integrated into various kinds of animation systems (for instance, in systems based on inverse kinematics, or on behavioral approaches). In the present implementation, the method is inserted in a dynamic simulation system, where it provides a good compromise between the efficiency of the computations and the "correctness" of the results: corrections associated with the constraints take the physical data into account, and first-order momenta are conserved during the constraint processing.

We are currently studying methods of extending displacement constraints to "soft constraints" between solids. With soft constraints the user will be able to specify preferential angular or linear domains for the relative positions of objects at hinges without restricting the motion to specific predefined boundaries. The solids will tend to satify the soft constraints if they are not subjected to forces that are too large, and if their accelerations are not too important.

We are interested in the ability of soft constraints to make the behavior of solid systems more flexible and we plan to use them as a compromise in over-constrained situations.

References

Armstrong W, Green M (1985) The dynamics of articulated rigid bodies for purposes of animation. Graph Inerface 85, Montreal, pp 407–415

Barzel R, Barr A (1988) A modeling system based on dynamic constraints. Comput Graph 22:179–188

Featherstone R (1983) The calculation of robot motion using articulated-body inertias. Int J Robotics Res 2:13–30

Gascuel MP, Verroust A, Puech C (1991) A modeling system for complex deformable bodies suited to animation and collision processing. J Visualization Comput Animation, 2:82–91, A shorter version of this paper appeared in Graphics Interface 91

Goldstein H (1983) Classical Mechanics, 2nd edn. Addison Wesley, Reading, Mass

Isaacs PM, Cohen UF (1987) Controlling dynamic simulation with kinematic constraints, behavior functions and inverse dynamics. Comput Graph 21:215–224
Isaacs PM, Cohen UF (1988) Mixed method for complex kinematic constraints in dynamic figure animation. Visual Comput 2:296–305
Lathrop RH (1986) Constrained (closed-loop) robot simulation by local constraint propagation. IEEE Int Conf Robotics Automation San Francisco, pp 689–694
Overveld C van (1990) A technique for motion specification in computer animation. Visual Comput 6:106–116
Overveld C van (1991) An iterative approach to dynamic simulation of 3-D rigid-body motions for real-time interactive computer animation. Visual Comput 7:29–38
Verroust A (1990) Etude de problèmes liés à la définition, la visualisation et l'animation d'objets complexes en informatique graphique. Thèse d'état, Université Paris XI, France
Wilhelms J, Barsky B (1985) Using dynamic analysis to animate articulated bodies as humans and robots. Graph Interface 85, Montreal, pp 97–104

Appendix A: The description file used for the monocycle

This is the description file for all the constraints involved in the monocycle animation sequence. The objects linked by the constraints are the fixed external world, the wheel, the character's body, and the legs which are constituted of `left-thigh`, `right-thigh`, `left-leg`, `right-leg`, `left-foot`, `right-foot`. (in this description *Oy* is the vertical axis).

```
/* Constraint between the wheel and the
character's body */
constraint object1 body; object2 wheel;
hinge 0 –2.0 0 0 0 0;
angle 1 0 0 1 0 0 max 0; end

/* Angle constraint preventing the character
from falling */
constraint object2 body; /* default object1:
world-fixed coordinate system */
angle 0 1 0 0 1 0 max 0.066; end

/* Constraints along the left leg */
constraint object1 body; object2 left-thigh;
hinge 0.4 –0.6 0  0 0 0;
angle 1 0 0 1 0 0 max 0; end

contraint object1 left-thigh;
object2 left-leg; hinge 0 0 1  0 0 0;
angle 1 0 0 1 0 0 max 0; end

constraint object1 left-leg;
object2 left-foot; hinge 0 0 1.4  0 0 –0.1;
angle 1 0 0 1 0 0 max 0; end

constraint object1 wheel; object2 left-foot;
hinge 0.4 0 –0.5 0 0 0; end

constraint object1 body; object2 left-foot;
angle 0 0 1 0 0 1 max .1; end

/* Constraints along the right leg */
constraint object1 body;
object2 right-thigh;
hinge –.4 –0.6 0  0 0 0;
angle 1 0 0 1 0 0 max 0.04; end

constraint object1 right-thigh;
object2 right-leg; hinge 0 0 1  0 0 0;
angle 1 0 0 1 0 0 max 0; end

constraint object1 right-leg;
object2 right-foot; hinge 0 0 1.4  0 0 –0.1;
angle 1 0 0 1 0 0 max 9; end

constraint object1 wheel;
object2 right-foot;
hinge –0.4 0 0.5  0 0 0; end

constraint object1 body; object2 right-foot;
angle 0 0 1 0 0 1 max .1; end
```