DYNAMIC ANIMATION OF N-DIMENSIONAL DEFORMABLE OBJECTS

Yannick Remion, Jean-Michel Nourrit, Olivier Nocent

Laboratoire d'Etudes et de Recherches Informatiques,
Université de Reims Champagne-Ardenne,
rue des crayères, BP 1035, 51687 Reims cedex 2,
France
{ yannick.remion, jm.nourrit }@univ-reims.fr http://www.univ-reims.fr/leri
nocent@leri.univ-reims.fr

ABSTRACT

This paper presents a new, accurate, efficient and unified method for dynamic animation of one, two or three-dimensional deformable objects. The objects are modelled as d-dimensional juxtapositions of d-dimensional patches defined as parametric blending of a common d-dimensional mesh of 3D control points. Animation of the object is achieved by dynamic animation of its control points. This ensures that at each time step the object shape conforms to its patches definitions, and, thus, that every property implied by the nature of the blending functions is verified. Dynamic animation of these continuous models implies no "matter discretising" as the control points are not considered as material points but moreover as the degrees of freedom of the continuous object. A generic (both for blending functions nature and object intrinsic dimension d) mechanical model reflecting this idea is proposed. Then, according to this modelling idea, a convenient generic dynamic animation engine is built from Lagrangian Equations. This engine relies upon an accurate and very efficient linear system. Forces and constraints handling as well as numerical resolution process are then briefly discussed in this scheme.

Keywords: Dynamic animation , Lagrangian equations, spline, parametric surfaces, parametric volumes, deformable objects.

INTRODUCTION

Usual dynamic animation methods for deformable continuous objects often imply some discretising of the objects' matter (see for examples [Lucia86], [Breen92], [Chanc95], [Provo95]). In such schemes, matter is concentrated upon a finite set of material points. We think this is physically unrealistic and would prefer to handle those objects as continuous.

We thus present how continuous deformable objects of intrinsic dimension 1, 2 or 3 can be handled by an unified, accurate and yet efficient dynamic animation engine. This engine is built upon continuous kinematics and mass repartition modelling, and handles both discrete or continuous external strains as well as supplementary constraints. As one can immediately guess, we still lack an efficient tool handling actually continuous internal strains such as elasticity or viscoelasticity. Nevertheless, this work could then be viewed as the foundation of an animation engine dealing accurately and implicitly with actually continuous deformable objects of any intrinsic dimension between 1 to 3.

In a previous study ([Remio99a,99b]) we developed an animation engine for continuous curve-like objects. In order to animate actually continuous

objects, this engine modeled 3D curve-like objects as successions of spline segments, and considered their control points as the degrees of freedom of the continuous objects rather than material points "discretizing" the object. The geometrical properties of the actual modelled object shape and the degree of control exerted upon this shape are tied to the chosen spline segments. These features may have been carefully chosen by the object designer, in which case they are relevant to the modelled object ; the method conveniently ensures that they are preserved over time. The engine, thus, accurately handles generic continuous curve-like objects whose shapes are constrained by the chosen spline segments models. Furthermore, the generated linear system to be solved exhibits interesting properties (axes separation and common constant matrix for the 3 axes).

As we found these properties rather smart, the presently described engine, not surprisingly, is built upon similar assumptions but handles indiscriminately continuous curves, surfaces or volumes. The current paper then begins with an extension of the "succession of spline segments" model to similar 2D or 3D continuous models. In fact a generic model for d-dimensional continuous objects will be proposed that will both emphasise the similarities between the 3 cases and bring a common frame for the development of the underlying dynamic animation engine. Obviously this mechanical model encompasses geometry, kinematics and mass repartition.

The theoretical development of the engine is then proposed. We first choose Lagrangian Equations as the most suited dynamic equations for the defined mechanical system and reshape those equations in computationally efficient formulas. After which forces and constraints incidence in these equations are briefly and generally discussed, and a short presentation of classical numerical resolution methods follows. The validity of the method is then demonstrated on an example.

MECHANICAL MODEL

Geometry

The previous work was developed upon a unified and generic parametric 3D curve model which we called "succession of spline segments". This model is built as a succession of one or more curves defined as functional combinations of a common set of 3D control points with a set of parametric blending functions. It encompasses a great deal of classical spline models both "interpolating" (Hermitian splines [Barnh74] , Catmull-Rom splines [Catmu74] and TCB splines [Kocha84], …) and "approximating" (Bezier splines [Bézie66,77], [Farin90], B-splines [Riese73], [Foley89], β-splines [Barsk81,83] and NURBS [Piegl91], …).

Extension of this modelling scheme to surfaces or volumes is straightforward. Hence, we model objects as juxtapositions of d-dimensional patches (see Fig 1 to 3).

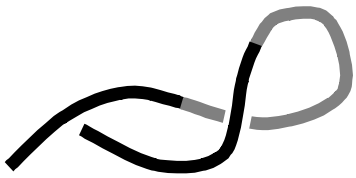

Figure 1 : 1-d model : succession of spline segments

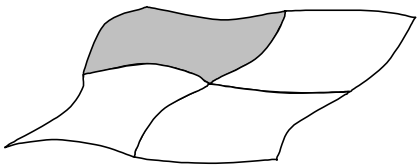

Figure 2 : 2-d model : juxtaposition of 2D patches

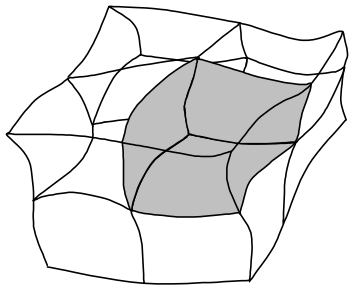

Figure 3 : 3-d model : juxtaposition of 3D patches

These patches are defined (see Fig 4 to 6) as d-dimensional parametric functional combinations of a common set of n 3D control points q_i ($i \in [0,n[$). A point on such an object is referenced by its patch number $j \in [0,np[$ [1] and its parametric position on this patch $p \in [0,1]^d$. Its absolute 3D position is given by the position function of its patch, built upon a set of blending functions $b_i^j(p)$:

$$M_j : \quad [0,1]^d \rightarrow \mathbb{R}^3$$
$$p \mapsto M_j(p) = \sum_{i \in [0,n[} b_i^j(p) q_i$$

[1] Both control points and patches could be numbered in a d-dimensional pattern ; for further convenience in the engine building we number them in a one dimensional way.

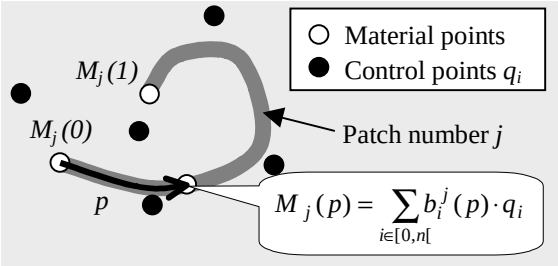

Figure 4 : 1d patch geometrical definition

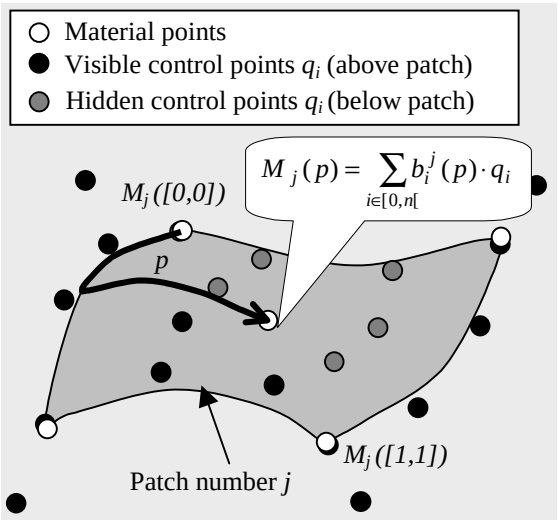

Figure 5 : 2d patch geometrical definition

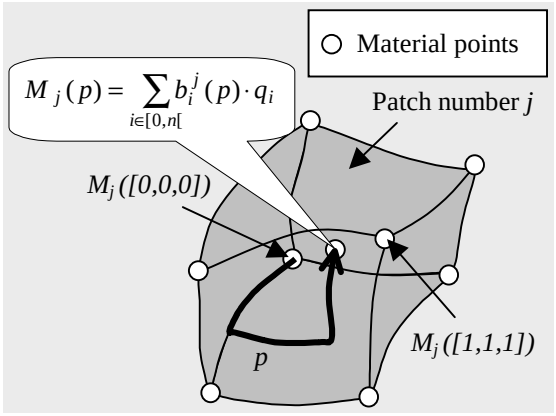

Figure 6 : 3d patch geometrical definition[2]

Mechanical System Choice

Let's now focus on specification of the mechanical system that will be animated.

As previously noted, in order to preserve the geometrical properties over time, we shall animate the control points. Nevertheless, we have already quoted that the concentration of the object mass on a discrete set of 3D points would be unrealistic and even more so if one remembers that some of these control points could not lie on the object.

Hence, similarly to the D-NURBS proposed by D. Terzopoulos and H. Qin in [Terzo94] and [Qin96], we consider the control points as the degrees of freedom of the continuous object. As mass repartition remains continuous, these methods are more accurate than the more straightforward animation of a set of material control points.

Kinematics

The first step in mechanical modelling of an identified system concerns kinematics, and thus expressions of the positions and velocities of the material points. More precisely, this step should yield temporal position functions for each of the material points.

According to the previous decisions (i.e. geometrical modelling and mechanical system choice), at each time step the object shape will be defined as the same parametric combination of the instantaneous positions of the mobile control points. The temporal position functions can easily be expressed on a per patch basis as the following functions of a parametric position p and time t :

$$P_j : \quad [0,1]^d \times \mathbb{R} \rightarrow \mathbb{R}^3$$
$$(p,t) \mapsto P_j(p,t) = \sum_{i\in[0,n[} b_i^j(p) q_i(t)$$

where $b_i^j(p)$ still are the combination functions of the control points q_i used by the patch number j[3].

Therefore, the kinematics of the object are defined by the position functions P_j and their first time derivatives expressing the velocities of the material points :

$$\dot{P}_j(p,t) = \sum_{i\in[0,n[} b_i^j(p)\dot{q}_i(t)$$

As one can see, the kinematics of the whole object are defined by the finite set of the n 3D functions $q_i(t)$. These functions can be seen as the $3n$ degrees of freedom of the continuous mechanical system.

Mass repartition

The second step in the mechanical modelling of the system concerns mass repartition over the object.

[2] Control points are not represented in order to keep the figure clear.

[3] Hence, as expected, a material point is still referenced by its patch number j and its parametric position p on this patch.

In the particular cases where d=1 or 2 (curves or surfaces) we consider that thickness is negligible and so, model mass repartition in the same way for each value of d as *np* mass density functions (one per patch) :

$$\rho_j : \quad [0,1]^d \times \mathbb{R} \rightarrow \mathbb{R}$$
$$(p,t) \mapsto \rho_j(p,t)$$

These mass density functions should not be considered as giving mass density per unit length, surface or volume but rather per unit parametric variation of *p*. Hence a small part *dp* around the material point at parametric position *p* on patch number *j* has a mass $\rho_j(p,t)\,dp$.

As patches geometry can evolve in time, this implies that mass density per unit length, surface or volume can evolve too, even if $\rho_j(p,t)$ is actually invariant in time (this time invariance simply meaning that matter is not gliding along the object and is tied to its parametric position *p*).

ENGINE DEVELOPMENT

Dynamic equations

Kinematics and mass repartition being modelled, we now have to choose the dynamic equations most suited to the nature of the system.

Since the mechanical system is continuous and non rigid with a finite number of degrees of freedom (*3n*), analytic mechanics seem best suited than Euler or Newton/Euler formalisms dealing respectively with particles and solids. We therefore choose to use Lagrangian equations :

$$\frac{d}{dt}\frac{\partial K}{\partial \dot{q}_i^\alpha}(q,\dot{q},t) - \frac{\partial K}{\partial q_i^\alpha}(q,\dot{q},t) = Q_i^\alpha - \frac{\partial E}{\partial q_i^\alpha}$$
$$i \in [0,n[,\ \alpha \in \{x,y,z\}$$

where $K(q,\dot{q},t)$ is the kinetic energy function, $q_i^\alpha(t)$ are the degrees of freedom, *E* the potential energy of the external forces deriving from a potential and Q_i^α the power rating of the other external forces in the virtual movement instilled by q_i^α (see [Germa86]).

The kinetic function can be expressed as :

$$K(q,\dot{q},t) = \tfrac{1}{2}\sum_{j\in[0,np[}\int_{p\in[0,1[^d}\rho_j(p,t)\Big[\sum_{i\in[0,n[} b_i^j(p)\dot{q}_i(t)\Big]^2 dp$$

Then, the left terms of the equations become :

$$\frac{\partial K}{\partial q_i^\alpha} = 0$$

$$\frac{\partial K}{\partial \dot{q}_i^\alpha} = \sum_{j\in[0,np[}\int_{[0,1]^d}\rho_j(p,t)\Big[\sum_{l\in[0,n[} b_l^j(p)\dot{q}_l^\alpha(t)\Big]b_i^j(p)\,dp$$
$$= \sum_{j\in[0,np[}\sum_{l\in[0,n[}\Big[\int_{[0,1]^d}\rho_j(p,t)b_l^j(p)b_i^j(p)\,dp\Big]\dot{q}_l^\alpha(t)$$

Under the additional hypothesis that mass density functions are invariant in time (i.e. matter is not gliding along the object), this last term can be simplified as :

$$\frac{\partial K}{\partial \dot{q}_i^\alpha} = \sum_{l\in[0,n[} M_{il}\,\dot{q}_l^\alpha(t)$$

where M_{il} are the constant following terms :

$$M_{il} = \sum_{j\in[0,np[}\int_{[0,1]^d}\rho_j(p,t)\,b_l^j(p)b_i^j(p)\,dp$$

Lagrangian equations then yield the following linear equations system of the unknowns $\ddot{q}_i(t)$:

$$M\,\ddot{q}^\alpha = W^\alpha \qquad \alpha \in \{x,y,z\}$$

where M is the *n* square symmetric matrix filled with elements M_{il} and W_i^α is the force term :

$$W_i^\alpha = Q_i^\alpha - \frac{\partial E}{\partial q_i^\alpha}.$$

Some important facts should be noted concerning this equations system :

- it is formally accurate (i.e. it implies no methodological numerical error such as those brought by a purely numerical resolution of Lagrangian equations) and nevertheless, requires no symbolic computation.
- it is composed of 3 independent systems, one for each scene axis (x,y,z) .
- these 3 systems use the same matrix M .
- this matrix is constant over time and can be built and inverted once at the beginning of the animation process.

Forces

In order to change the kinematics of the object, forces have to be exerted on it and these forces should be handled as suited in the chosen dynamic equations.

In Lagrangian equations, forces are introduced in the system either by their contribution to the potential energy *E* or by their power ratings in the virtual movements instilled by the degrees of freedom. Let's

show these two cases upon two classical external force fields : respectively gravity and viscosity.

Gravity

Gravity forces obviously derive from the following potential with *z* as the vertical axis :

$$E_{grav} = \sum_{j\in[0,np[} \int_{[0,1]^d} \rho_j(p,t) \sum_{i\in[0,n[} b_i^j(p) q_i^z(t)\, dp$$

and thus their contributions to the right terms of the equations are :

$$W_{grav}{}_i^x = W_{grav}{}_i^y = 0$$

$$W_{grav}{}_i^z = -\frac{\partial E_{grav}}{\partial q_i^z} = -\sum_{j\in[0,np[} \int_{[0,1]^d} \rho_j(p,t)\, b_i^j(p)\, dp$$

Under the usual hypothesis that mass density functions are invariant in time, these terms are invariant in time too and can be computed once at the beginning of the process.

Viscosity

Viscosity forces do not derive from a potential, hence their power ratings in the virtual movements have to be expressed.

The force density exerted upon the material point at parametric position *p* on patch number *j* is :

$$F_j(p,t) = -C\, \dot{P}_j(p,t) = -C \sum_{l\in[0,n[} b_l^j(p) \dot{q}_l(t)$$

where *C* is the viscosity coefficient.

The coordinates of the velocity of this material point in the virtual movement instilled by q_i^α are :

$$\hat{\dot{P}}_j^{i,\alpha}(p,t)^{\alpha'} = \sum_{l\in[0,n[} b_l^j(p) \delta_{li} \delta_{\alpha\alpha'}$$
$$= b_i^j(p)\, \delta_{\alpha\alpha'} \quad \alpha' \in \{x,y,z\}$$

and thus the total contribution of viscosity is :

$$W_{visc}{}_i^\alpha = Q_{visc}{}_i^\alpha = \sum_{j\in[0,np[} \int_{[0,1]^d} F_j(p,t)\, \hat{\dot{P}}_j^{i,\alpha}(p,t)\, dp$$
$$= -C \sum_{l\in[0,n[} \dot{q}_l^\alpha(t) \sum_{j\in[0,np[} \int_{[0,1]^d} b_l^j(p) b_i^j(p)\, dp$$

which can be summarised as

$$W_{visc}{}^\alpha = \Phi\, \dot{q}^\alpha(t)$$

where Φ is a constant *n* square symmetric matrix :

$$\Phi_{il} = -C \sum_{j\in[0,np[} \int_{[0,1]^d} b_l^j(p) b_i^j(p)\, dp\,.$$

Constraints

Control of the dynamics is partly achieved by the exertion of external forces, but in many practical cases supplementary constraints are imposed to the movement. These constraints are a smart way of specifying things like “this point should not move”, “this point should follow a specific trajectory over time”, or many more kinds of movement restrictions.

Constraints are usually handled by one of the following methods [Platt92], [Terzo94] :

- the reduced coordinate method, each constraint actually negates a dof. The system is made smaller and the constraints are always actually realised. The drawback is that this reduction is an arbitrarily complex problem that can hardly be efficiently and automatically resolved.

- the penalisation method, which roughly consists in adding viscous specialised springs aiming to push constraints back to realisation when they derive. This method is computationally efficient since it only adds forces and does not increase the system size, but constraints are only rarely accurately realised. Moreover, this method can lead to inaccurate equilibrium states as the constraints springs responses required to balance the other forces imply that constraints are not realised.

- Lagrangian multipliers, an accurate constraints handling tool provided by the Lagrangian formalism. Using this tool, the equations classically become [Germa86]:

$$\frac{d}{dt}\frac{\partial K}{\partial \dot{q}_i^\alpha} - \frac{\partial K}{\partial q_i^\alpha} = W_i^\alpha + \sum_{c\in[0,nc[} L_{ci}^\alpha\, \lambda_c \qquad i \in [0,n[,\ \alpha \in \{x,y,z\}$$

$$\sum_{\alpha\in\{x,y,z\}} \sum_{l\in[o,n[} L_{cl}^\alpha\, \ddot{q}_l^\alpha = D_c(q,\dot{q}) \qquad c \in [0,nc[$$

  where the second equations set express the *nc* constraints upon the unknowns, and λ_c is the Lagrangian multiplier of the constraint number *c*, a new unknown related to the reaction force needed to keep the constraint realised.

  In this scheme, constraints are accurately realised but the system size is increased by the number of scalar constraints *nc*.

Using Lagrangian multipliers, our system becomes :

$$\begin{bmatrix} M & & & -L^{xT} \\ & M & & -L^{yT} \\ & & M & -L^{zT} \\ L^x & L^y & L^z & 0 \end{bmatrix} \cdot \begin{bmatrix} \ddot{q}^x \\ \ddot{q}^y \\ \ddot{q}^z \\ \Lambda \end{bmatrix} = \begin{bmatrix} W^x \\ W^y \\ W^z \\ D \end{bmatrix}$$

If the L^{α} are time independent the whole matrix can be built and inverted once. Thus the price for accuracy is reduced and Lagrangian multipliers method is practically usable.

If the L^{α} vary over time, the size increase and, worst, the time dependence of the whole matrix could impose the less accurate but more efficient penalisation method.

Numerical resolution

Classically rewritten as a Cauchy's problem the linear system becomes :

$$\dot{\xi} = F(\xi, t) = \begin{bmatrix} \dot{q} \\ f(q, \dot{q}, t) \end{bmatrix}, \qquad \xi = \begin{bmatrix} q \\ \dot{q} \end{bmatrix}$$

where f is the numerical function solving the linear system of the unknows $\ddot{q}$.

The temporal integration of this system is a classical mathematical problem for which a set of well-known solutions exist [Press88], (for example explicit or implicit Euler, prediction/correction, Runge Kutta second or fourth order methods, …). These methods differ according to their precision, to the number of evaluations of F (i.e. resolution of the system) per time step and to their ability to automatically adapt the time step as needed by the actual temporal "regularity" of the movement.

The later property (i.e. ability to adapt time step automatically) is quite relevant since it permits to either slow down resolution in order to properly handle a singularity, or to speed up resolution when movement is regular enough.

According to the precision and efficiency needed, one can choose any of these (or others) methods. Some authors ([Desbr99], [Baraf98]) recommend implicit Euler resolution scheme for its stability, especially for stiff scenes. We haven't yet tested this scheme, but we hope to do so in a near future as it could allow for larger time steps and thus faster simulations.

SOME RESULTS

The example shown below (plates 1 to 5) is an animation of a deformable elastic child's swing. The mechanical system consists in 4 ropes each modelled as 4 1d patches, a sitting board modelled as 3x2x1 3d-patches and a piece of cloth modelled as 2x2 2d patches.

Every involved patch uses separable Catmull-Rom blending functions. Thus the required control points are 4+3 for each rope, (3+3)x(2+3)x(1+3) for the board, and (2+3)x(2+3) for the piece of cloth, for a grand total of 4x7+120+25=173. Hence our mechanical system uses 519 degrees of freedom.

Gravity and viscosity are the sole external forces applied. Internal strains (non-linear elasticity) are roughly modelled by a d-dimensional mesh of non-linear springs attached to material points inside each d-patch.

Constraints encompass 4 ropes ends stationarity (motionless material points) and 8 inter-objects links (2 material points belonging respectively to each object constrained to stick one to the other). We handle those constraints by Lagrangian multipliers.

Numerical resolution is achieved by a fourth order Runge-Kutta method using a fixed time step.

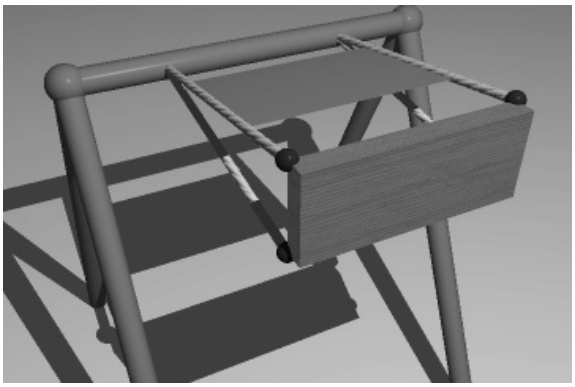

Plate 1 : deformable child's swing starting position[4]

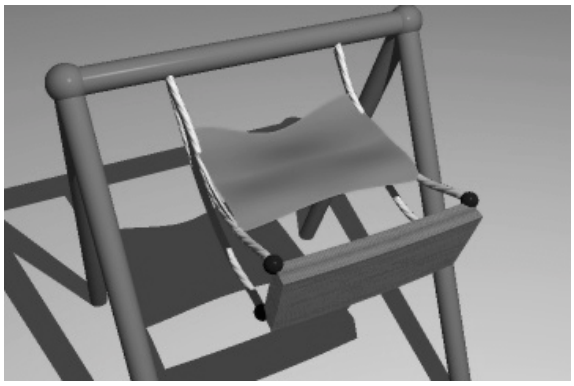

Plate 2 : deformable child's swing at t=0.36 s

[4] In these plates, non material objects have been added : a beam from which the swing is hanging and 4 spheres (one at each rope/board link).

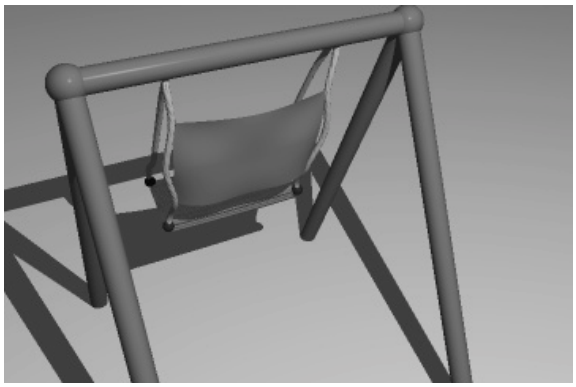

Plate 3 : deformable child's swing at t=0.84 s

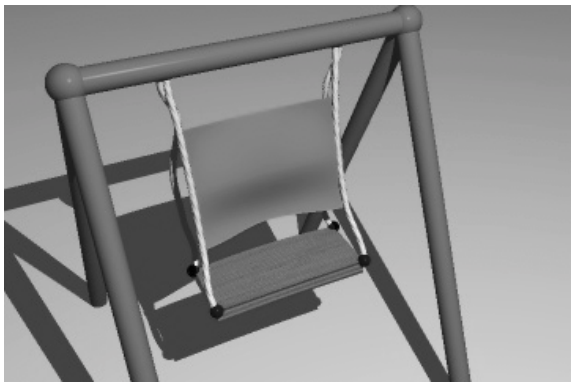

Plate 4 : deformable child's swing at t=2.0 s

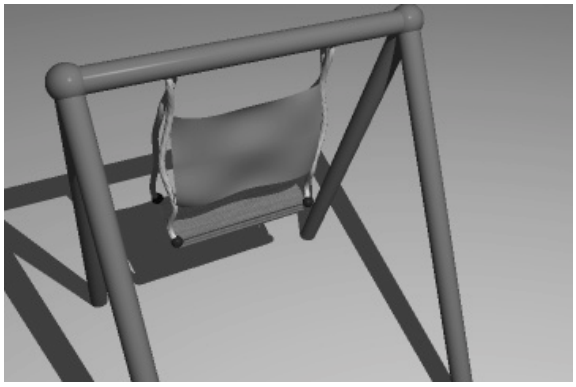

Plate 5 : deformable child's swing at t=3.28 s

A WORD ABOUT IMPLEMENTATION

Starting the implementation of the engine, we decided to keep as close as possible to its unified formal presentation. Hence we had to build a data type for deformable objects that would be generic both for the intrinsic dimension of the object and for the peculiar blending functions used. If the abstraction of blending functions was quite obvious relying upon the function polymorphism, abstraction of the intrinsic dimension was not as obvious. In fact we came to define, in C++, our deformable object class as a template of another type supposed to be one of 3 proposed "dimension types". These 3 "dimension types" share a peculiar feature : each of them encompasses with the same names 2 nested types encoding respectively discrete and continuous positions for the associated dimension. In this scheme the template abstract object type can rely on its "dimension type" parameter for everything related to its intrinsic dimension.

A concrete (usable) class for a peculiar object (with known dimension and blending functions) is then obtained as a class derived from the instantiation of the abstract template with the right "dimension type". Obviously this class has to replace the abstract methods related to actual blending functions.

The abstract template actually implements in its methods the described engine overall design, relying on both its "dimension type" parameter and its abstract methods for any detail specific to intrinsic dimension or actual blending functions used.

CONCLUSION

As expected, extension of the previous work to surfaces and volumes was straightforward. Furthermore, thanks to a unified but similar mechanical model, the theoretical development is so much similar that the resulting unified engine shares the same interesting properties :

- **shape constraining** : the object shape conforms at each time step to its patches definitions ; this can be seen either as a necessary restriction or as a blessing as it implies that the geometrical properties inferred by these peculiar patches are always verified by the modelled object
- **generality** : the underlying geometrical model is generic enough to encompass a great deal of classical parametric 3D curves, surfaces or volumes modelisations
- **accuracy** : the actual continuous objects are handled by accurate numerical equations
- **efficiency** : the linear system to be solved has very interesting properties (axes separation and common constant matrix for the 3 axes)

Obviously, this engine benefits from the unified development frame, and equally animates curves, surfaces or volumes.

This work is a step towards dynamic animation of smooth deformable continuous objects, as it permits, according to the chosen blending functions, a smooth and continuous mass repartition. This has to be completed by a continuous handling of internal strains (implying elasticity, viscoelasticity and so on). This completed engine should then be compared to finite elements methods.

REFERENCES

[Baraf98] Baraff D., Witkin A.. *Large steps in cloth simulation*, SIGGRAPH'98 Proceedings, Computer Graphics, p 43-54, 1998.

[Barnh74] Barnhill R., Riesenfeld R. *Computer aided Geometric design*, Academic Press, 1974.

[Barsk81] Barsky B. *The Beta-spline : a local representation based on shape parameters and fundamental geometric measures*, PhD Thesis, University of Utah, 1981.

[Barsk83] Barsky B., Beatty J. *Local control of bias and tension in Beta-splines*, Computer Graphics, v 17, n 3, p 193-218, 1983.

[Bézie66] Bézier P. *Définition numérique de courbes et surfaces*, Automatisme, v 11, p 625-632 (part 1), v 12, p 17-21 (part 2), 1966.

[Bézie77] Bézier P. *Essai de définition numérique des courbes et surfaces expérimentales*, Thèse de doctorat, Université Paris VI, 1977.

[Breen92] Breen D.E., House D.H., Getto P.H., *A physical-based particle model of woven cloth*, The Visual Computer, Vol. 8, pp. 264-277, 1992.

[Desbr99] Desbrun M., Schröder P., Barr A. *Interactive animation of structured deformable objects*, Graphics Interface'99 proceedings, p 1-8, Kingston, Canada, June 1999.

[Farin90] Farin G. *Curves and surfaces for Computer Aided Geometric Design a practical guide*, Academic Press, second edition, 1990.

[Foley89] Foley J., Van Dam A., Feiner S., Hughes J. *Computer Graphics : principles and practice*, Addison Wesley, 1989.

[Catmu74] Catmull E., Rom R. *A class of local interpolating splines*, in [Barnh74].

[Chanc95] Chanclou B., Luciani A., *Physical models and dynamic simulation of planetary motor vehicles with a great number of degrees of freedom*, Proceedings of IAS 4 Conférence, 99. 465-472, 1995.

[Germa86] Germain P. *Mécanique X*, tome I, Ecole Polytechnique, Ellipses, 1986.

[Kocha84] Kochanek D., Bartels R. *Interpolating splines with local tension , continuity and bias control*, Computer Graphics, v 18, n 3, p 33-41, 1984.

[Lucia86] Luciani A., Cadoz C., *Utilisation de modèles mécaniques et géométriques pour la synthèse et le contrôle d'images animées*, Deuxième colloque Image, CESTA, 1986.

[Piegl91] Piegl L. *On NURBS : a survey*, Computer Graphics and Application, v 11, n 1, p 55-71, 1991.

[Platt92] Platt J. *A generalization of dynamic constraints*, Graphical Models and Image Processing, Vol. 54, No. 6, pp. 516-525, 1992.

[Press88] Press W.H., Flannery B.P., Teukolsky S.A., Vetterling W.T. *Numerical Recipes in C*, Cambridge University Press, 1988.

[Provo95] Provot X., *Deformation constraints in a mass-spring model to describe rigid cloth behaviour*, Proceedings of Graphics Interface, pp. 147-154, 1995.

[Qin96] Qin H., Terzopoulos D. *D-NURBS : A physics-based geometric design framework*, IEEE Transactions on Visualization and Computer Graphics,2(1), March, 1996, pp. 85-96, 1996.

[Remio99a] Remion Y., Nourrit J.M., Gillard D., *Dynamic Animation of Spline Like objects*, WSCG'99 proceedings, Plzen, Czech Republic, pp. 426-432, 1999.

[Remio99b] Remion Y., Nourrit J.M., Gillard D., *A Dynamic Animation Engine For Generic Spline Objects*, Journal of Visualization and Computer Animation (to appear)

[Riese73] Riesenfeld R. *Applications of B-spline Approximation to geometric problems of Computer Aided Design*, PhD Thesis, University of Syracuse, 1973.

[Terzo94] Terzopoulos D., Qin H. *Dynamic NURBS with Geometric Constraints for Interactive Sculpting*, ACM Transactions on Graphics, 13(2), April, 1994, pp. 103-136, 1996.