Département Informatique, CNAM
RSX206 - Conception d'Applications Multimédia
2007

Animatique 1

P. Cubaud

plan de l'exposé :

1. Introduction

Utilisation de l'ordinateur dans l'animation :

- dans la création des dessins (images clés)
    - numérisation des images-clé
    - création avec un logiciel de dessin
    - objets complexes synthétisés par programme
- dans la création du mouvement
    - calcul des images intermédiaires
    - calcul des déplacements
- dans le coloriage
    - dessin assisté
    - synthèse d'image
- dans la prise de vue
    - caméra pilotée par ordinateur
    - caméra synthétique
- en post-production (synchronisation image-son, etc.)

Différents niveaux d'intervention de l'ordinateur :

N1) éditeurs graphiques sans prise en compte du temps :
pour les designers

N2) calcul des images intermédiaires et déplacement d'objets
sur une trajectoire : pour les dessinateurs (et les remplacer ?)

N3) gestion des objets de la scène : translation, rotation +
caméra virtuelle(zoom, pan, tilt)

N4) définition d'acteurs : objets possédant une capacité de gestion
de leur animation. Gestion de contraintes entre acteurs

N5) les acteurs deviennent extensibles et apprennent

On s'intéresse ici aux N3 et N4

animatique : spécification et génération de mouvement sur des objets graphiques (3D)

on ne ne traite pas l'animation assistée par ordinateur

on ne présentera pas les systèmes existant (exposé à venir)

une revue des premiers systèmes dans [THA] chap. 4

une première classification [PAR] :
- tech. bas niveau : l'animateur exerce un contrôle fort
- tech haut niveau : contrôle faible, les algos à la place
- des cas particuliers importants : les gaz, la vie, l'humain

Difficulté du sujet :

left, 0 if both (or none) are pressed, and 1 if right is pressed. T
thetical expression will act as a modifier to the first part of the
physics tells us that

```
Velocity=Space/Time
```

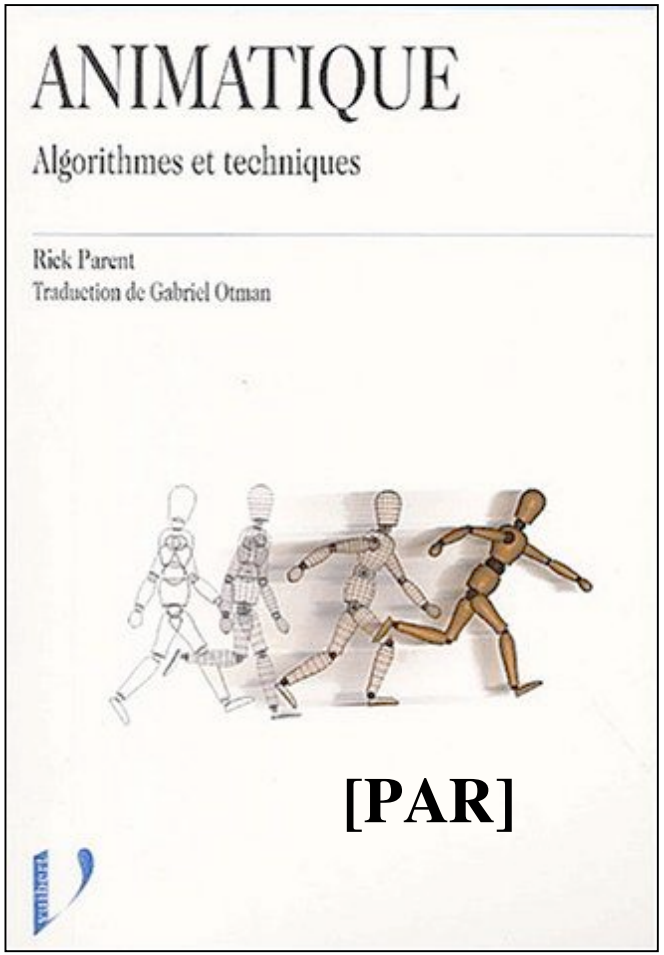
Then, solving for Space we get

```
Space=Velocity*Time
```

which is fundamentally what we are doing here (albeit in an a
multiplying a constant ROTSPEED (which will determine how

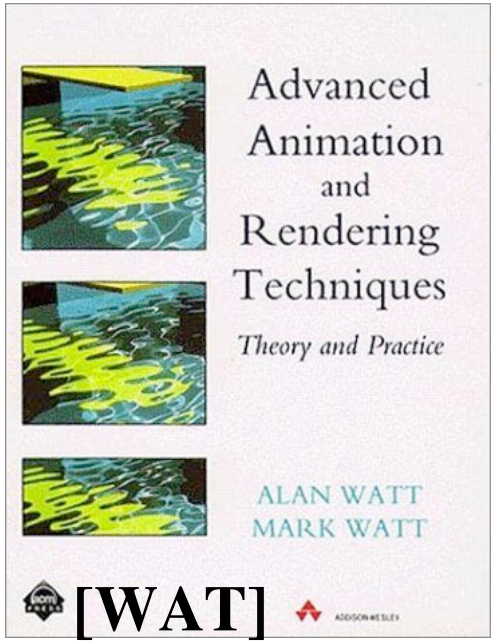
-à la frontière de nombreuses disciplines (mécanique, physique, biophysique, robotique …)
- en informatique :
 calcul numérique, RO, IA, simulation, syst. répartis…
- phénomène complexe => simulateur complexe
=> méthode de développement sophistiquée

Bibliographie



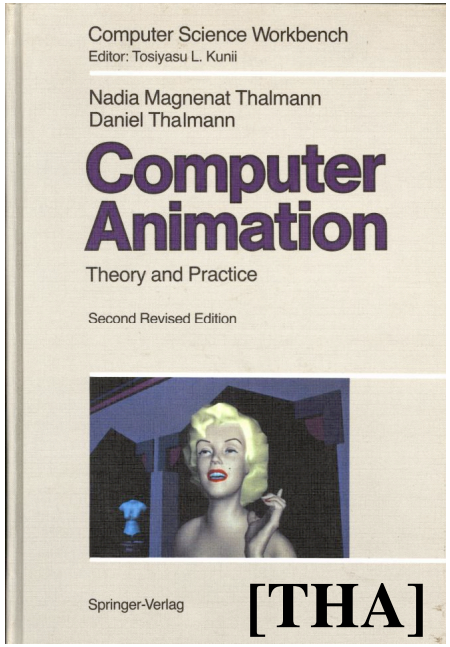
[PAR]

trad. en 2003, 45 euros env.



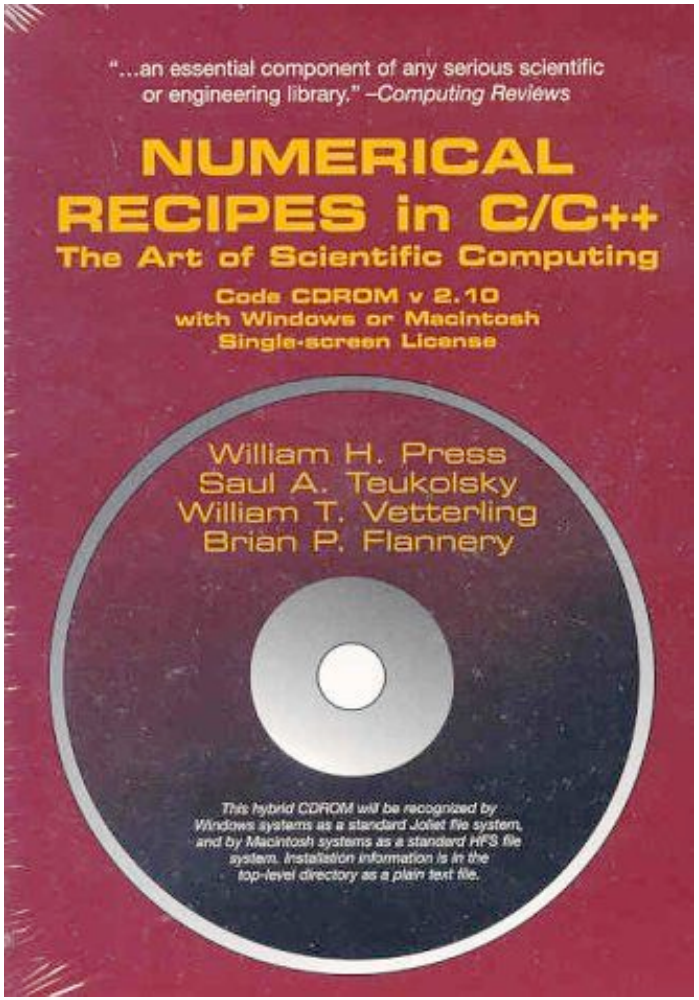
[WAT]

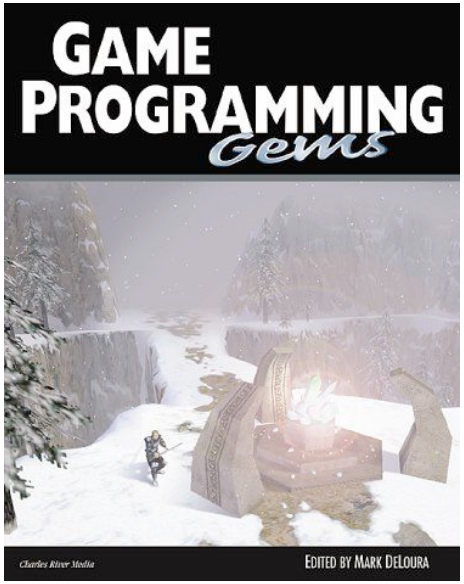
1992, non traduit
(partie IV seulement)

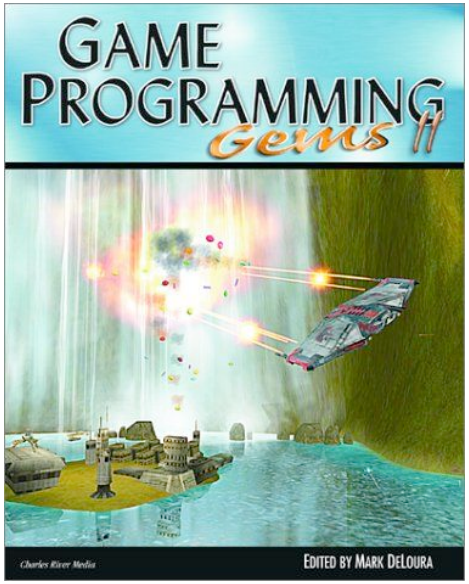


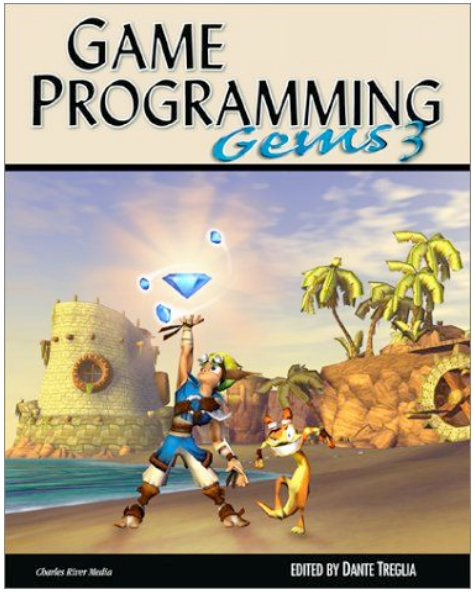
[THA]

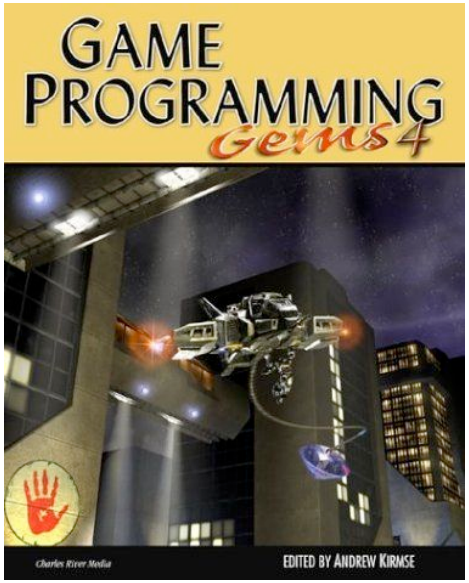
1985 et 90


"...an essential component of any serious scientific or engineering library." –Computing Reviews
NUMERICAL RECIPES in C/C++
The Art of Scientific Computing
Code CDROM v 2.10
with Windows or Macintosh
Single-screen License
William H. Press
Saul A. Teukolsky
William T. Vetterling
Brian P. Flannery
This hybrid CDROM will be recognized by Windows systems as a standard Joliet file system, and by Macintosh systems as a standard HFS file system. Installation information is in the top-level directory as a plain text file.


GAME PROGRAMMING Gems
Charles River Media
EDITED BY MARK DELOURA


GAME PROGRAMMING Gems II
Charles River Media
EDITED BY MARK DELOURA


GAME PROGRAMMING Gems 3
Charles River Media
EDITED BY DANTE TREGLIA


GAME PROGRAMMING Gems 4
Charles River Media
EDITED BY ANDREW KIRMSE

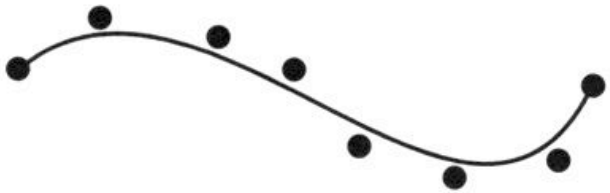
2. Trajectoire

2.1Courbes paramétrées
2.2 Interpolation du déplacement
2.3 Vitesse non constante
2.4 Objet en rotation
2.5 Dynamique

2.1 Rappel sur les courbes paramétrées

$$p\begin{pmatrix} x \\ y \\ z \end{pmatrix} = f\begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} f_x(u) \\ f_y(u) \\ f_z(u) \end{pmatrix}$$ on prend $0 \leq u \leq 1$

Une spline d'interpolation dans laquelle la spline passe par les points de contrôle intérieurs



Une spline d'approximation dans laquelle seuls les points d'extrémité sont interpolés, les points de contrôle intérieurs étant uniquement utilisés pour tracer la courbe

Figure 3.1 **Calcul de l'interpolation et des splines d'approximation**

exemple : les B-splines uniformes :

$$f_i(u) = \begin{pmatrix} u^3 & u^2 & u & 1 \end{pmatrix} \frac{1}{6} \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{pmatrix} \begin{pmatrix} \vec{P}_{i-1} \\ \vec{P}_i \\ \vec{P}_{i+1} \\ \vec{P}_{i+2} \end{pmatrix}$$

Assemblage et continuité

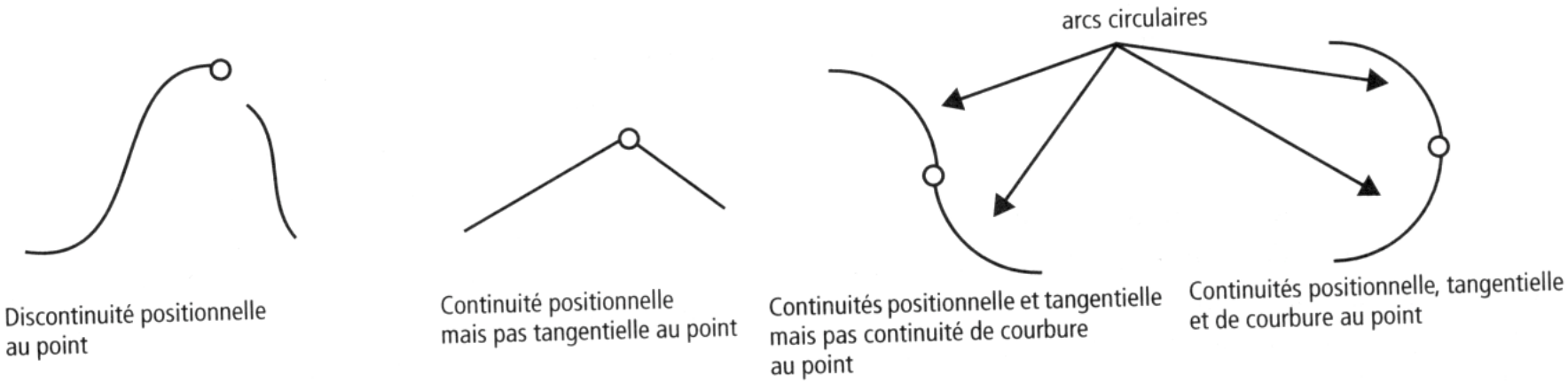


Figure 3.2 **Continuité (au point indiqué par un petit cercle)**

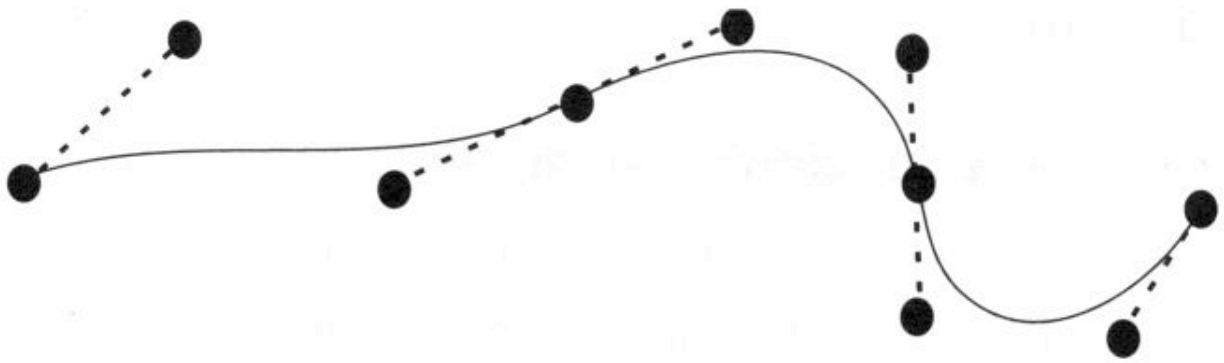


Figure B.35 **Courbe cubique de Bézier composite avec tangentes et points de contrôle colinéaires**

2.2 Interpolation du déplacement sur la courbe

pour produire une vitesse uniforme, on parcoure des arcs de longueur constante

$$X(u) = Au^3 + Bu^2 + Cu + D$$

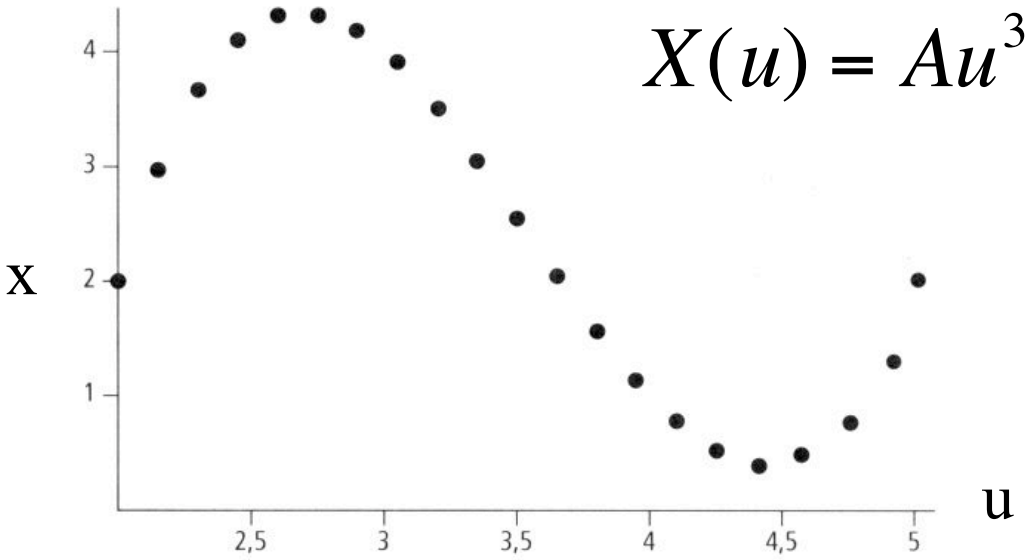


Figure 3.4 **Exemple de points produits par des incréments égaux d'un paramètre d'interpolation pour une courbe cubique typique**

Si on progresse à Δu constant, on n'a pas pour autant Δx constant !

Calcul de la longueur d'arc

-Première méthode : rapide, grossière

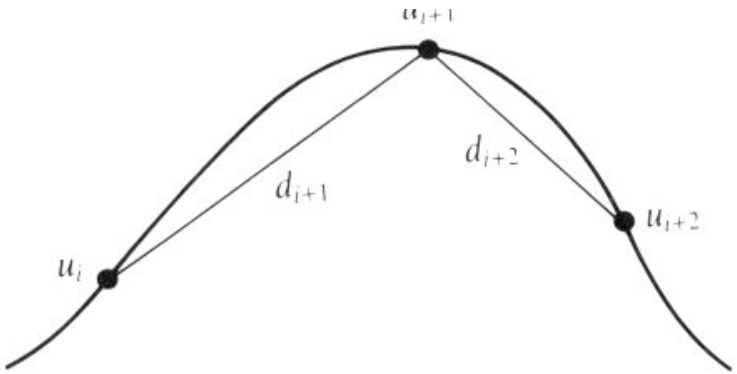


Figure 15.5 Forward differencing and accumulated chord length.

Index	Paramètre	Longueur d'arc (G)
0	0,00	0,000
1	0,05	0,080
2	0,10	0,150
3	0,15	0,230
4	0,20	0,320
5	0,25	0,400
6	0,30	0,500
7	0,35	0,600

$$L = L_i + \frac{u - i\Delta u}{\Delta u}\left(L_{i+1} - L_i\right)$$

Connaissant L, on peut retrouver u

- deuxième méthode : précise, coûteuse (en temps de calcul)

$$L(u_1,u_2) = \int_{u_1}^{u_2} dl(u) = \int_{u_1}^{u_2} \sqrt{\left(\frac{dx}{du}\right)^2 + \left(\frac{dy}{du}\right)^2 + \left(\frac{dz}{du}\right)^2}$$

$$= \int_{u_1}^{u_2} \sqrt{Au^4 + Bu^3 + Cu^2 + Du + E}\,.du$$

avec :

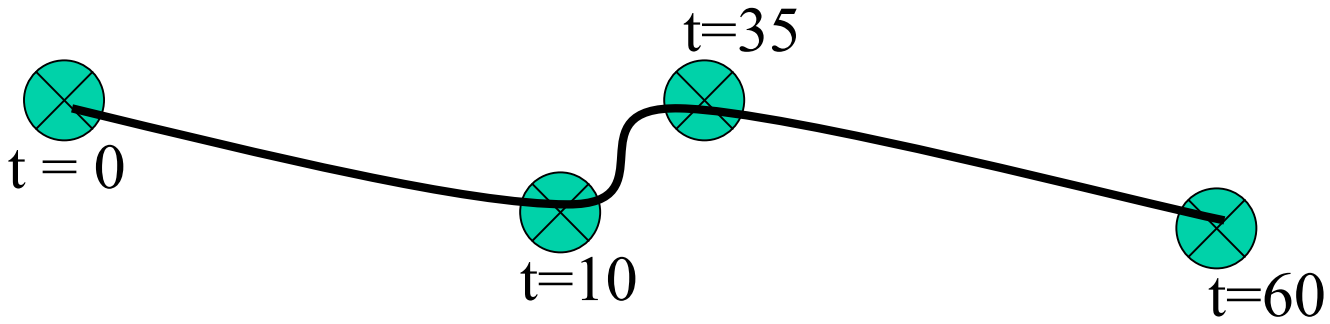
$$A = 9\left(a_x^2 + a_y^2 + a_z^2\right)$$
$$B = 12\left(a_x b_x + a_y b_y + a_z b_z\right)$$
$$C = 6\left(a_x c_x + a_y c_y + a_z c_z\right) + 4\left(b_x^2 + b_y^2 + b_z^2\right)$$
$$D = 4\left(b_x c_x + b_y c_y + b_z c_z\right)$$
$$E = c_x^2 + c_y^2 + c_z^2$$

ne s'intègre pas formellement !

⇒utiliser des méthodes numériques (trapèzes, Chebyshev…)

⇒approches adaptatives : dans certaines zones, la dérivée varie vite qu'ailleurs

2.3 Vitesse non constante

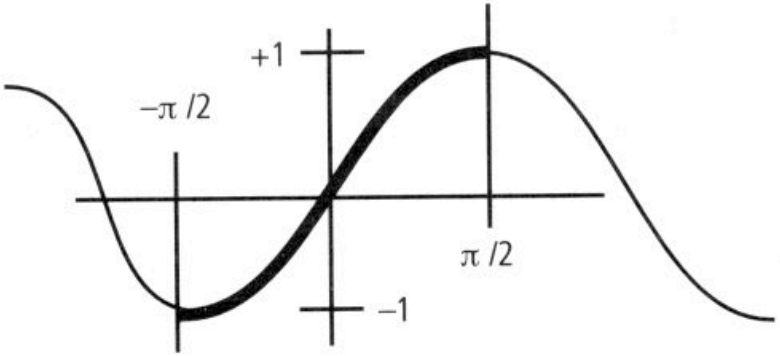


Il faut deux fonctions :
- la courbe d'espace pour la trajectoire P(u)
- la fonction temps-distance S(t)

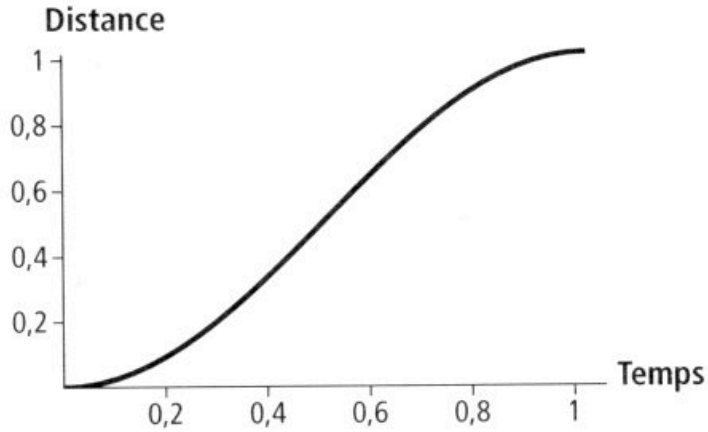
S(t) doit être monotone en t et continue

Pour t donné :
-calculer S(t) = L = distance parcourue à date t
- trouver u pour cette distance d'après la table
- calculer P(u), la position du point

- Le « ease-in / ease-out »



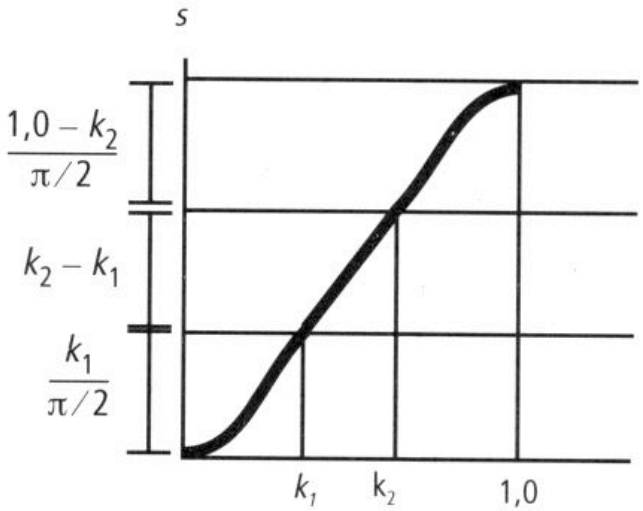
Segment de courbe sinusoïdale à utiliser pour le contrôle de l'effet d'*ease-in/ease-out*



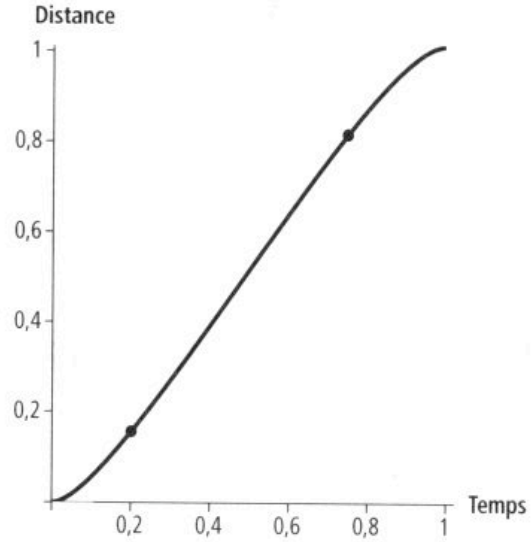
Segment de courbe sinusoïdale mappé vers des valeurs utiles

Figure 3.11 **Utilisation d'un segment de courbe sinusoïdale comme fonction temps-distance pour l'effet *ease-in/ease-out***

$$S(t) = ease(t) = \frac{1 + \sin(\pi t - \pi/2)}{2}$$



Courbe d'ease-in/ease-out telle qu'elle est constituée au départ



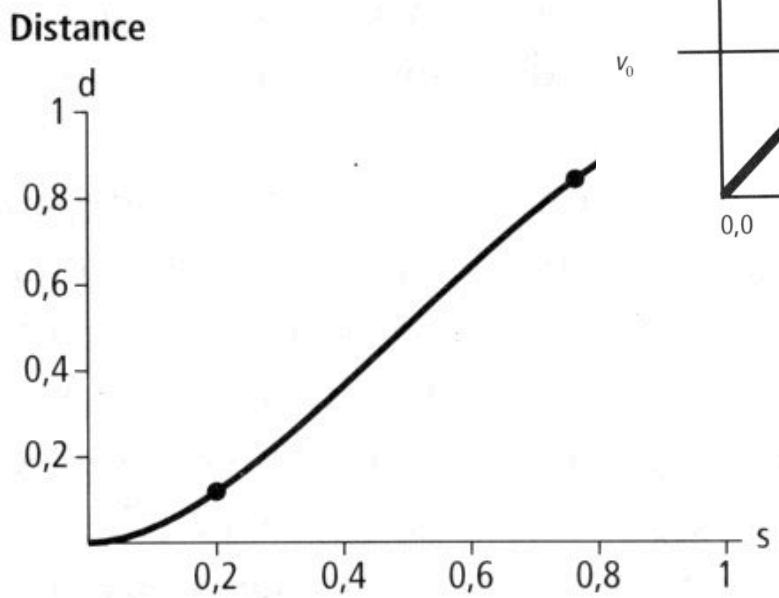
Segments de courbe étalonnés en valeurs utiles avec des points marquant les intersections de segments

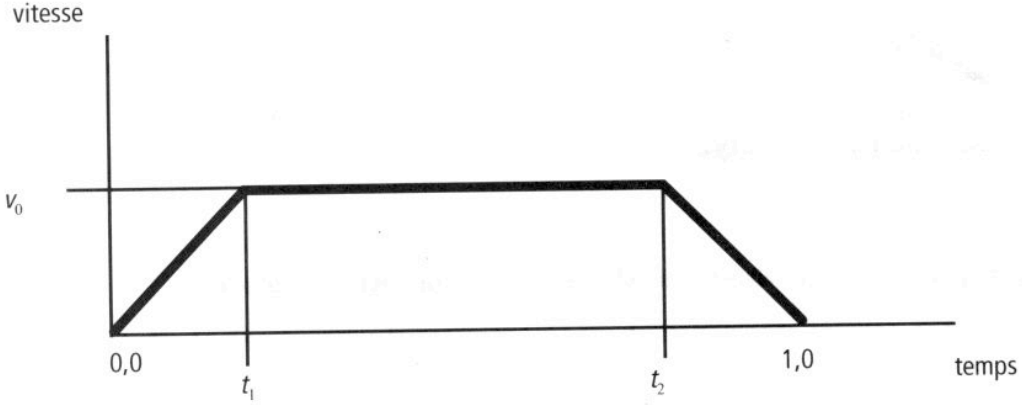
Figure 3.12 **Utilisation de segments sinusoïdaux avec un intervalle intermédiaire à vitesse constante**

$$ease(t) = \begin{cases} = \left(k_1 \cdot \frac{2}{\pi} \cdot \left(\sin\left(\frac{t}{k_1} \cdot \frac{\pi}{2} - \frac{\pi}{2}\right) + 1\right)\right)/f & t \le k_1 \\ = \left(\frac{k_1}{\pi/2} + t - k_1\right)/f & k_1 \le t \le k_2 \\ = \left(\frac{k_1}{\pi/2} + k_2 - k_1 + \left((1-k_2) \cdot \frac{2}{\pi}\right) \sin\left(\frac{t-k_2}{1{,}0-k_2} \cdot \frac{\pi}{2}\right)\right)/f & k_2 \le t \end{cases}$$

quand $f = k_1 \cdot 2/\pi + k_2 - k_1 + (1{,}0 - k_2) \cdot 2/\pi$

- Accélération constante





$$d = v_0 \cdot \frac{t^2}{2 \cdot t_1} \qquad 0{,}0 < t < t_1$$

$$d = v_0 \cdot \frac{t_1}{2} + v_0 \cdot (t - t_1) \qquad t_1 < t < t_2$$

$$d = v_0 \cdot \frac{t_1}{2} + v_0 \cdot (t_2 - t_1) + \left(v_0 - \frac{\left(v_0 \cdot \frac{t - t_2}{1 - t_2} \right)}{2} \right) \cdot (t - t_2) \qquad t_2 < t < 1{,}0$$

-Fonction distance-temps arbitraire

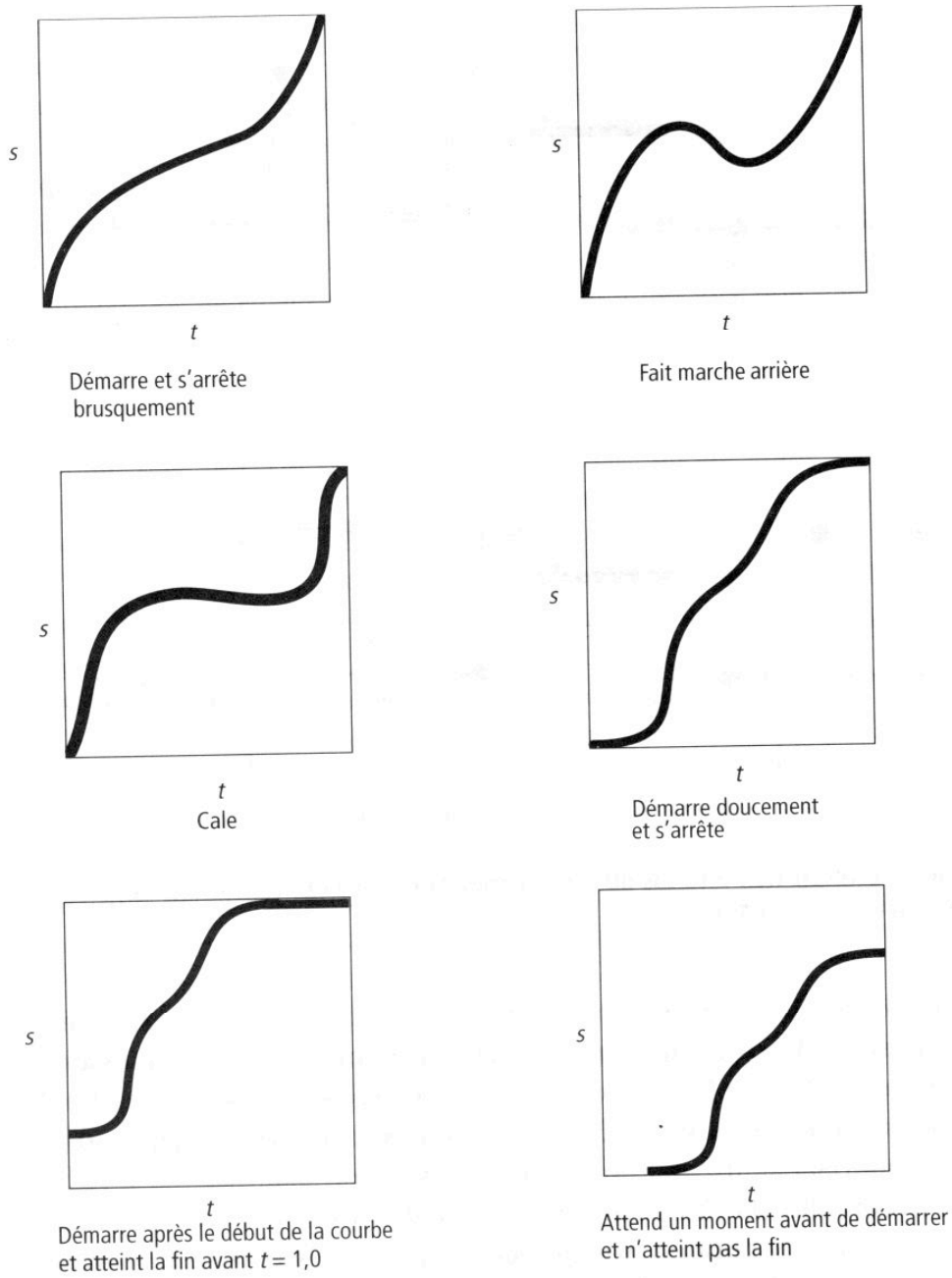
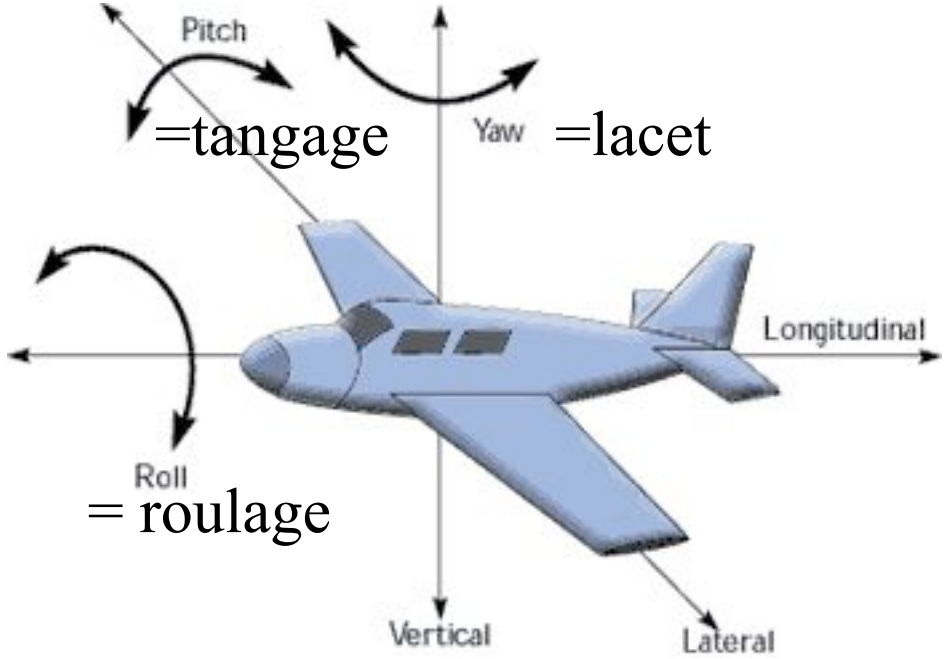


Figure 3.19 **Exemples de fonctions distance-temps**

2.4 Objet en rotation



Angles d'Euler :

Rotations sur les axes :

$$R_Z(\theta) = \begin{pmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad R_X(\theta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$R_Y(\theta) = \begin{pmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

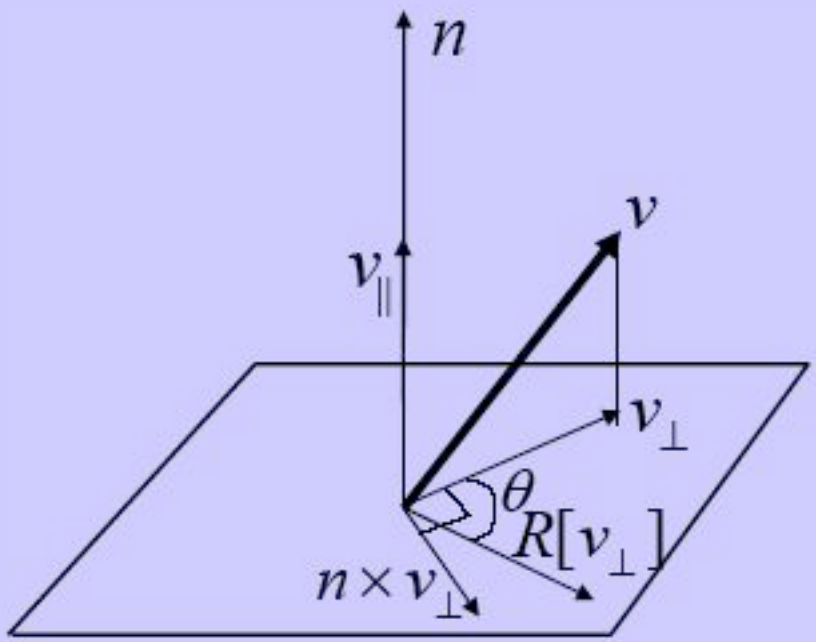
Inutilisable pour l'interpolation d'une orientation à une autre !

exemple d’application : les caméras de simulateur de vol

les caméras de jeux vidéo

« third person » : derrière et en haut le personnage
(ex. de *Tomb raider*)

-Rotation sur un axe quelconque n d'un angle θ



$$v_{\parallel} = (n \cdot v)n \qquad v_{\perp} = v - v_{\parallel}$$

$$R[v_{\perp}] = v_{\perp}\cos\theta + (n \times v_{\perp})\sin\theta$$
$$= v_{\perp}\cos\theta + (n \times v)\sin\theta$$

$$R[v_{\parallel}] = v_{\parallel}$$

$$\begin{aligned}
R[v] &= R[v_{\parallel} + v_{\perp}] \\
&= R[v_{\parallel}] + R[v_{\perp}] \\
&= v_{\parallel} + v_{\perp}\cos\theta \\
&\quad + (n \times v)\sin\theta \\
&= (n \cdot v)n + (v - (n \cdot v)n)\cos\theta \\
&\quad + (n \times v)\sin\theta \\
&= v\cos\theta + n(n \cdot v)(1 - \cos\theta) \\
&\quad + (n \times v)\sin\theta
\end{aligned}$$

Quaternions

- **Definition**: A *quaternion* is a quadruple $q=(s,v)$, where s is a scalar, and v a three-dimensional vector.
- The quaternions form a non-commutative group under the multiplication rule:

$$\begin{aligned} q_1 \cdot q_2 &= (s_1, v_1) \cdot (s_2, v_2) \\ &= (s_1 s_2 - v_1 \cdot v_2, s_1 v_2 + s_2 v_1 + v_1 \times v_2) \end{aligned}$$

- Quaternion multiplication is equivalent to that of

$$q = s + i v_x + j v_y + k v_z$$

$$\text{where } i^2 = j^2 = k^2 = -1, ij = k, ji = -k$$

- The complex number c is the special case $(\mathrm{Re}(c),[\mathrm{Im}(c),0,0])$

Quaternions (cont'd)

The *conjugate* of $q = (s,v)$ is

$$\overline{q} = (s,-v)$$

The *inner product* of $q_1 = (s_1,v_1)$ and $q_2 = (s_2,v_2)$ is

$$< q_1, q_2 >= q_1 \cdot \overline{q}_2 = s_1 s_2 + v_1 \cdot v_2$$

The *norm* of $q = (s,v)$ is

$$|q|^2 =< q, q >= q \cdot \overline{q} = s^2 + v_x^2 + v_y^2 + v_z^2$$

The *inverse* of $q = (s,v)$ is :

$$q^{-1} = \frac{\overline{q}}{\|q\|^2}$$

Corollary :

$$< q_1, q_2 >=< q_2, q_1 >$$

$$\|q_1 \cdot q_2\| = |q_1| \|q_2\|$$

Rotating with Quaternions

- Rotation by θ around normalized direction *n* may be represented by the unit quaternion

$$q=(\cos(\theta/2),\ n\sin(\theta/2))$$

- A regular 3D vector *v* may be represented as the quaternion (*0*,*v*). *v* is rotated by *q* to:

$$R_q[v] = \overline{q} \cdot v \cdot q$$

- Since

$$\begin{aligned} R_q[v] &= (\cos(\theta/2), -n\sin(\theta/2)) \cdot (0, v) \\ &\quad \cdot (\cos(\theta/2), n\sin(\theta/2)) \\ &= (0, v(\cos^2(\theta/2) - \sin^2(\theta/2)) \\ &\quad + 2n(n \cdot v)\sin^2(\theta/2) \\ &\quad + 2(n \times v)\cos(\theta/2)\sin(\theta/2)) \\ &= (0, v\cos\theta + n(n \cdot v)(1-\cos\theta) \\ &\quad + (n \times v)\sin\theta) \end{aligned}$$

Interpolating Two Orientations

- Orientations may be interpolated by interpolating their respective quaternions.

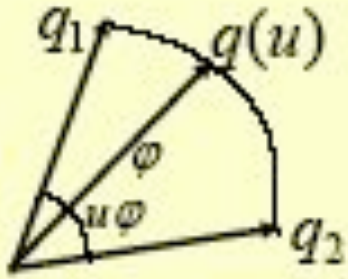
- Quaternions on a 4D sphere may be interpolated along a geodesic (the shortest path between two points on a sphere). This is the arc where the sphere intersects a plane through the two points and the origin.

- For any $u \in [0,1]$, writing $q(u) = \alpha(u)q_1 + \beta(u)q_2$, and solving the following system of equations for $\alpha(u)$ and $\beta(u)$:

$$1 = \|q(u)\|^2 = \alpha(u)^2 + \beta(u)^2 + 2\alpha(u)\beta(u) < q_1, q_2 >$$

$$\cos\varphi = < q_1, q_2 >$$

$$\cos(u\varphi) < q_1, q(u) > = \alpha(u) + \beta(u) < q_1, q_2 >$$



Spherical Linear Interpolation

$$q(u) = \frac{\sin(1-u)\varphi}{\sin\varphi} q_1 + \frac{\sin(u\varphi)}{\sin\varphi} q_2 \quad u \in [0,1]$$

<= SLERP(q1,q2,phi)

Autre exemple d'utilisation :

- convertir (roll, pitch, yaw) en un quaternion q
- convertir la variation d'orientation inter-trame en q'
- calculer la nouvelle orientation q''= q. q'
- convertir q'' en matrice de rotation R
- appliquer R aux points qui définissent l'objet

exemples de code `quat2mat` et `mat2quat` dans [WAT] et [PAR]

manipulation de quaternions possible avec DirectX (pas openGL)

2.5 Dynamique

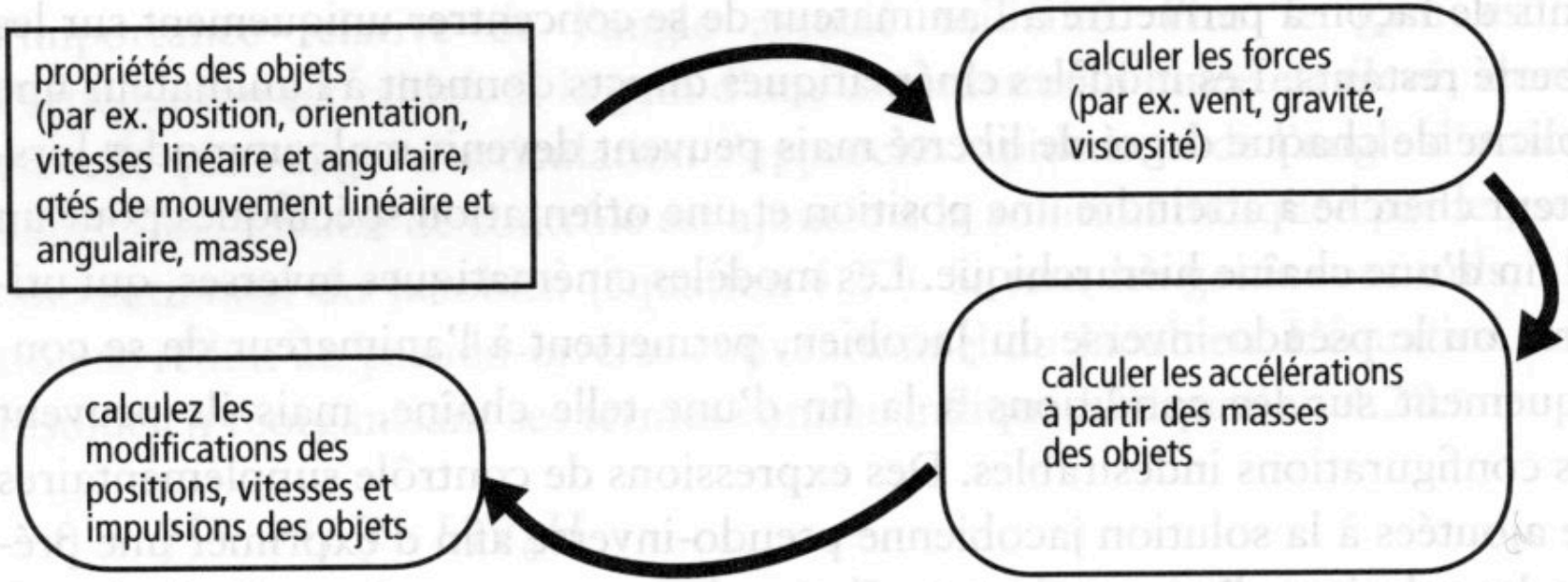
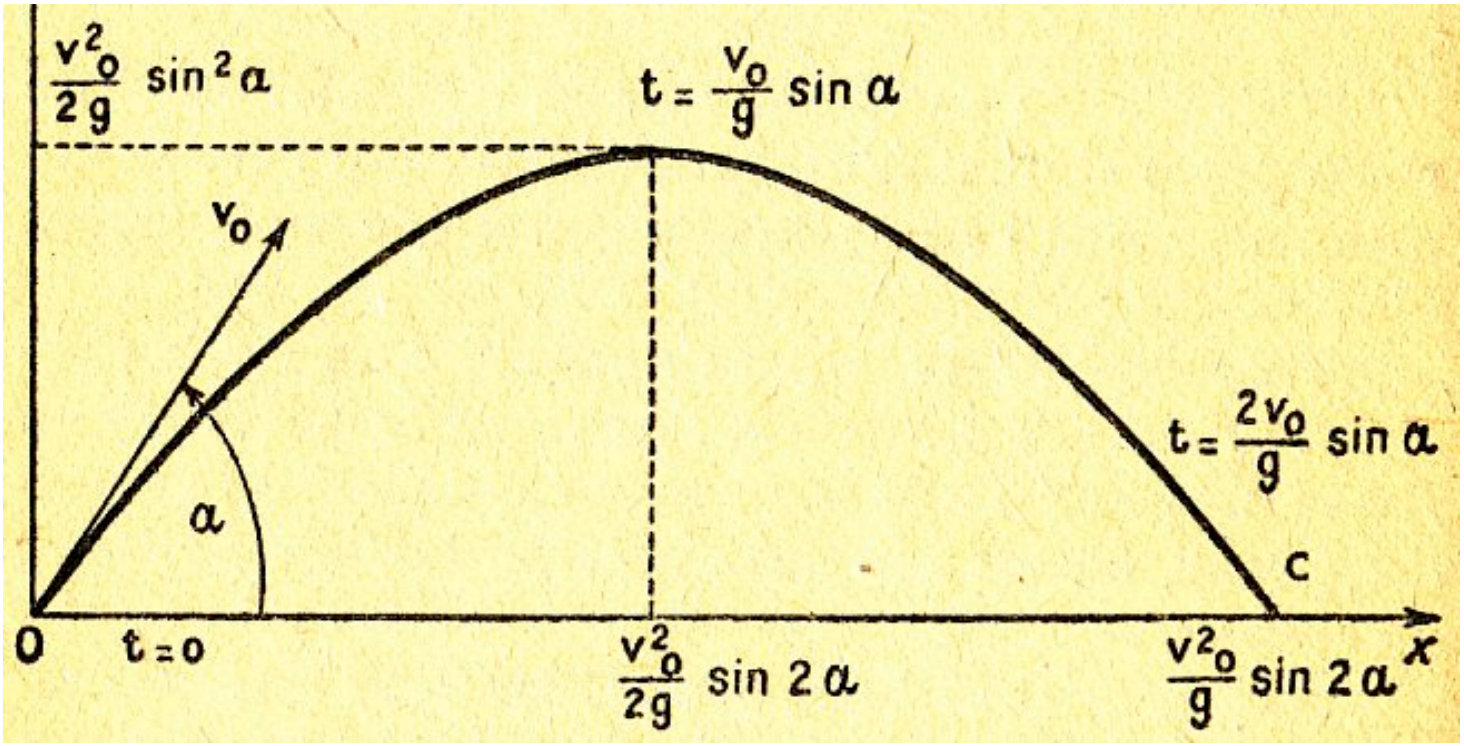


Figure 4.23 **Cycle de mise à jour de la simulation des corps rigides**

Formule fondamentale de la dynamique :

$$\sum \vec{F} = m\vec{a} \quad \text{avec} \quad \vec{a} = \begin{pmatrix} d^2x/dt^2 \\ d^2y/dt^2 \\ d^2z/dt^2 \end{pmatrix} = \begin{pmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{pmatrix}$$

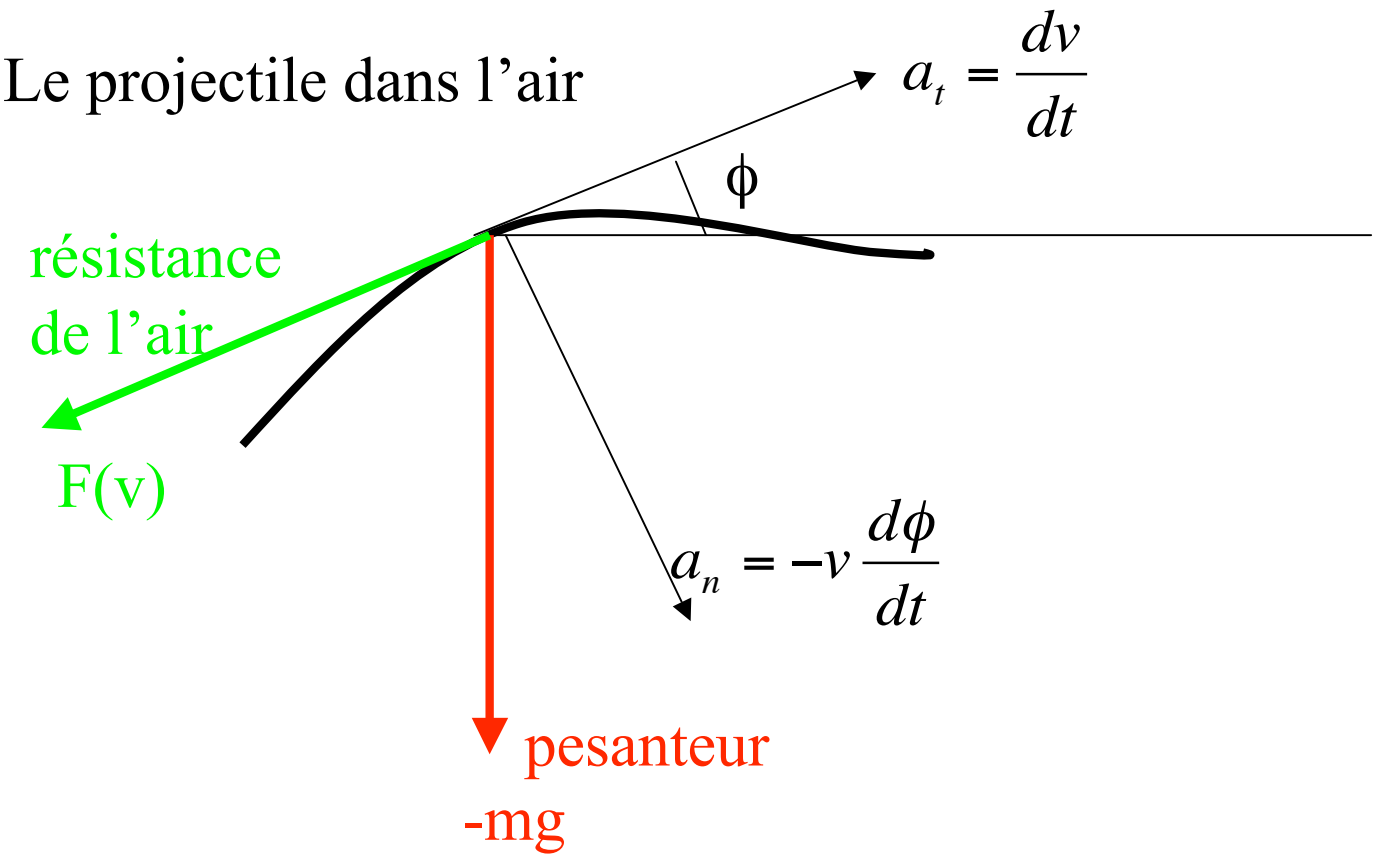
- Exemple d'un projectile dans le vide



$$\sum \vec{F} = \begin{pmatrix} 0 \\ -mg \\ 0 \end{pmatrix} = m\vec{a} = \begin{pmatrix} m\ddot{x} \\ m\ddot{y} \\ m\ddot{z} \end{pmatrix} \Rightarrow \begin{cases} \dot{x} = v_0 \cos\alpha \\ \dot{y} = v_0 \sin\alpha - gt \\ \dot{z} = 0 \end{cases} \Rightarrow \begin{cases} x = v_0 t \cos\alpha \\ y = v_0 t \sin\alpha - \frac{1}{2} g t^2 \\ z = 0 \end{cases}$$

avec g = 9.81 m/s^2

- Le projectile dans l'air



Equation différentielle : $$\frac{dv}{d\phi} = v\left(\tan\phi + \frac{F(v)}{mg\cos\phi}\right)$$

Résolution par intégration numérique : de Euler à Runge-Kutta

3. prise en compte de l'environnement

3.1 détections de collisions
3.2 réponse aux collisions
3.3 contraintes
3.4 les particules
3.5 les groupes (bandes, troupeaux)
3.6 l’autonomie

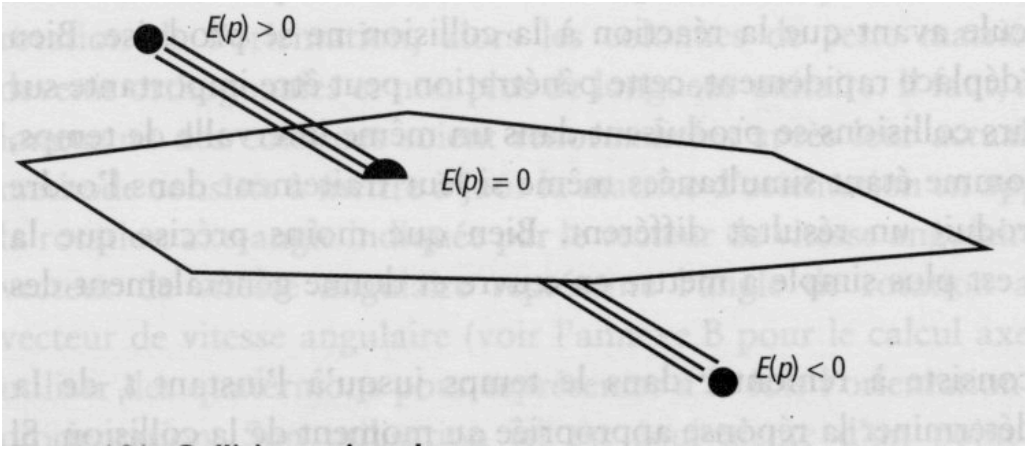
3.1 Détection de collisions

-ex. de la collision point-plan

Equation du plan en 3D : Ax + By + Cz = D
Connaissant 3 points non alignés du plan, on a :

$$\begin{pmatrix} A \\ B \\ C \end{pmatrix} = \overrightarrow{P_0P_1} \otimes \overrightarrow{P_0P_2} \quad \text{et} \quad D = A.P_{0x} + B.P_{0y} + C.P_{0z}$$

on étudie le signe de APx + BPy + CPz - D :



-test sur des polyèdres

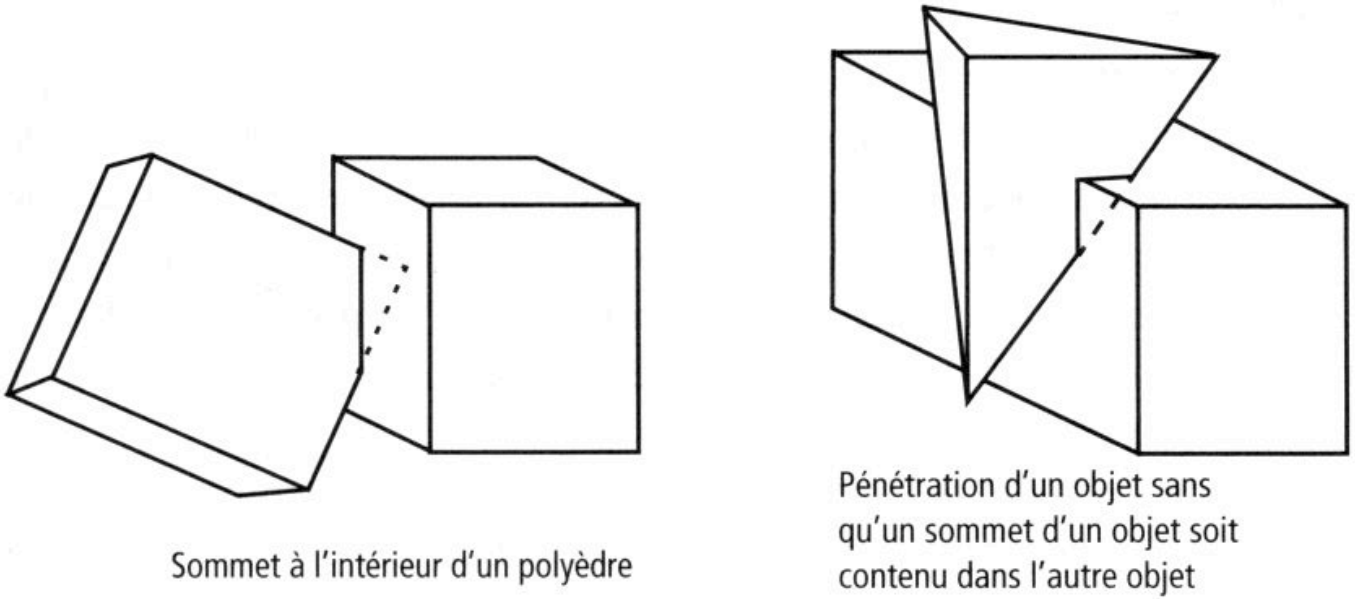


Figure 4.36 **Détection d'intersections de polyèdres**

intersections droite / polygone : coûteux
=> voir cours IMASON « élimination parties cachées »

- Sphères englobantes

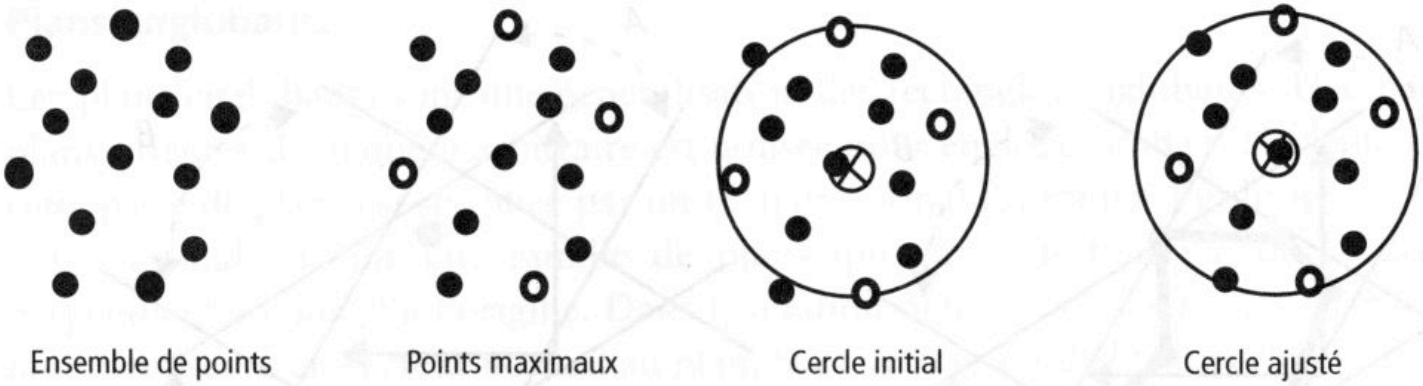
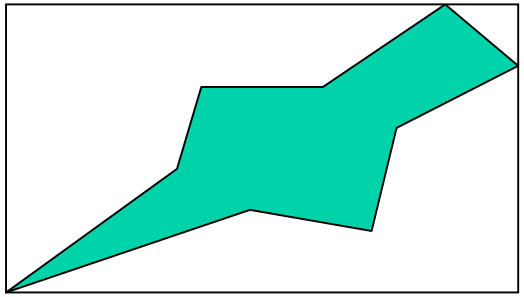


Figure B.20 **Calcul d'un cercle englobant pour un ensemble de points**

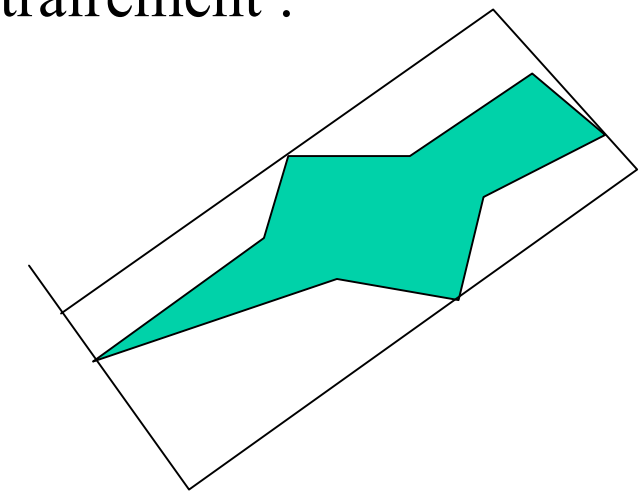
-recherche des 6 valeurs extremes Xmin, Ymin, Zmin, Xmax, etc...
et des 6 points concernés iXmin, iYmin, iZmin, etc…
-trouver la paire de points la plus éloignée des 3 paires : P1, P2
- le centre de la sphère est C = (P1+P2)/2
son diamètre est la plus grande distance Dmax
- passer en revue tous les points P et ajuster la sphère si PC > Dmax

- Autres techniques de volumes englobants

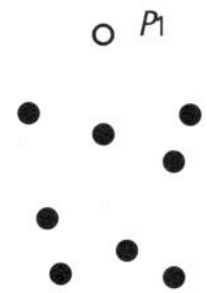
• boîte englobante alignée dans l'axe :
ABB : axis-aligned bounded box



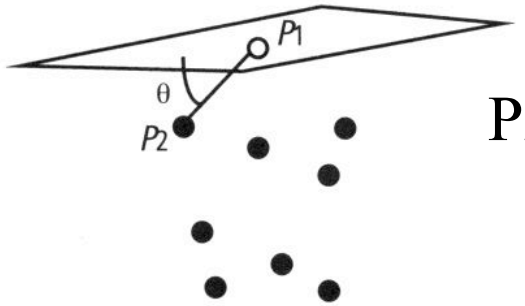
• boîte englobante orientée arbitrairement :
OBB : oriented bounded box



• construction de l'enveloppe convexe (ex. de solution)



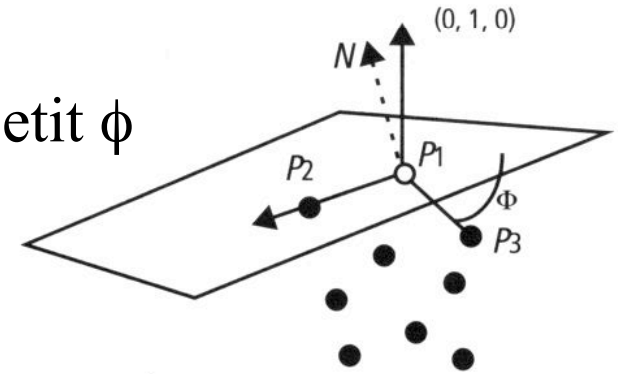
Étape 1 : Trouver le point le plus élevé



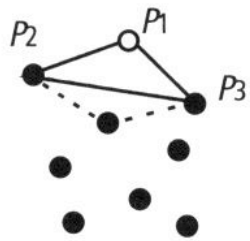
P2: plus petit θ

Étape 2 : Trouver une arête de l'enveloppe convexe

P3: plus petit ϕ



Étape 3 : Trouver un triangle initial



Étape 4 : Construire chaque triangle de l'enveloppe convexe en utilisant une ou deux arêtes existantes de triangles d'enveloppe convexe tracés antérieurement

algo. complet dans [PAR] p. 445

Un état de l'art sur les logiciels de détection de collision

Tangi.Meyer@irisa.fr
Guillermo.Andrade@irisa.fr

Présenté par Jean-Marie Souffez

IRISA

Outils	Méthodes	Entrées	Sorties	Principes
RAPID	OBB-Trees	Soupe de Polygones	Détection / paires intersectantes	2Body
VCollide	Surcouche de Rapid	Soupe de Polygones	Détection / paires intersectantes	Nbody - Cohérence temporelle
ICollide	Diagramme de Voronoi	Polyhèdres convexes	Distance / paires	Nbody - Cohérence temporelle
Solid	ABBTrees	Soupe de Polygones	Point de contact	Nbody - Cohérence temporelle
Quick_CD	K-Dops trees	Soupe de Polygones	Collision entre triangles	2Body
Swift	Voronoi+Multilevel model	Polyhèdres convexes	Distance avant collision	Nbody - Cohérence temporelle
V-Clip	Voronoi	Polyhédres convexes	Distance avant collision	NBody
VPS	Voxel+ Intersections géo.	Nuage de points	Pénétrations points-voxels	2Body
Hcollide	Spatial décomp.+OBBTrees	Polyhèdres	Pénétrations points-surfaces	Cohérence Temporelle
Contact	OBB-Trees	Soupe de Polygones	Point de contact	2Body - détection continue
ColDet	OBB-Trees	Soupe de Polygones	Collision exacte	2Body

3.2 Réaction à la collision

On peut réagir aux collisions trop tard :

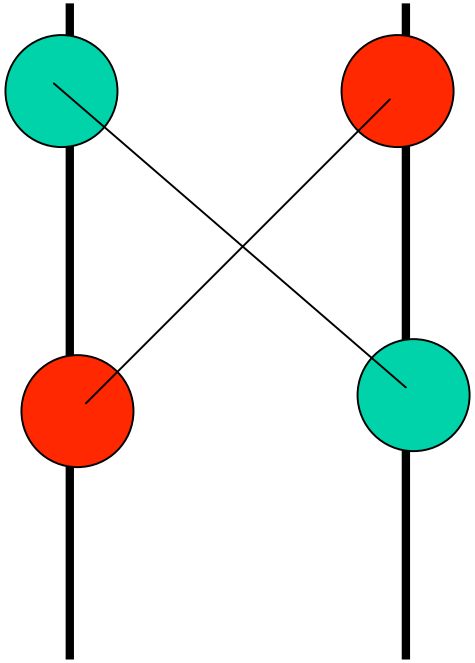


image #n
date T

#n+1
date T+dt

FAUX

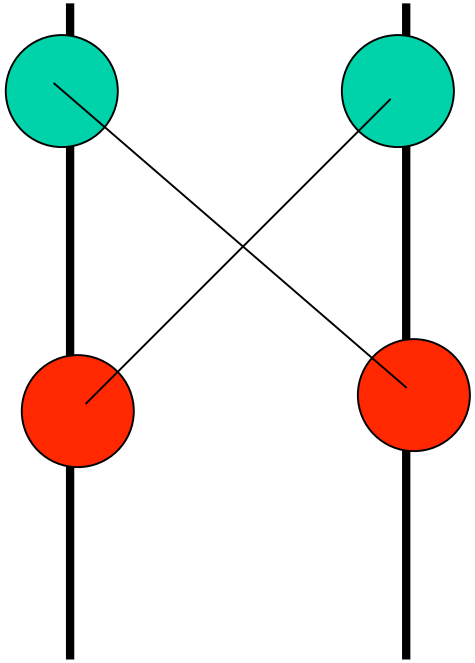


image #n
date T

#n+1
date T+dt

VRAI

• Méthode par pénalisation

- un point P est pénalisé pour avoir pénétré un autre objet O
- la surface de O est considérée fixe suite à la collision
- on crée une liason ressort (raideur k) entre O et P
et une masse à P

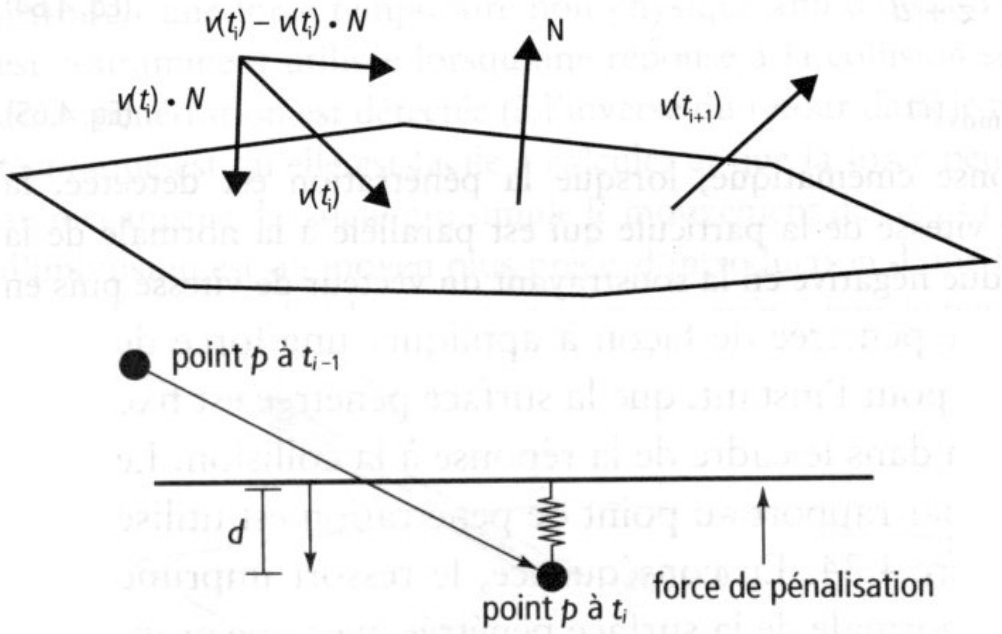


Figure 4.34 **Ressort de pénalisation**

loi de Hooke :
F = -k.d

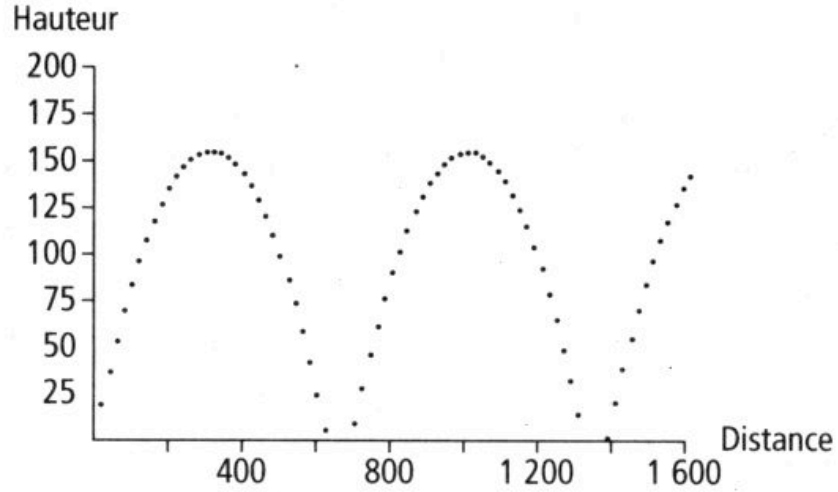


Figure 4.35 **Méthode à pénalisation avec une constante de ressort de 250 et une masse ponctuelle de 10 pour l'exemple de la section 4.3.1**

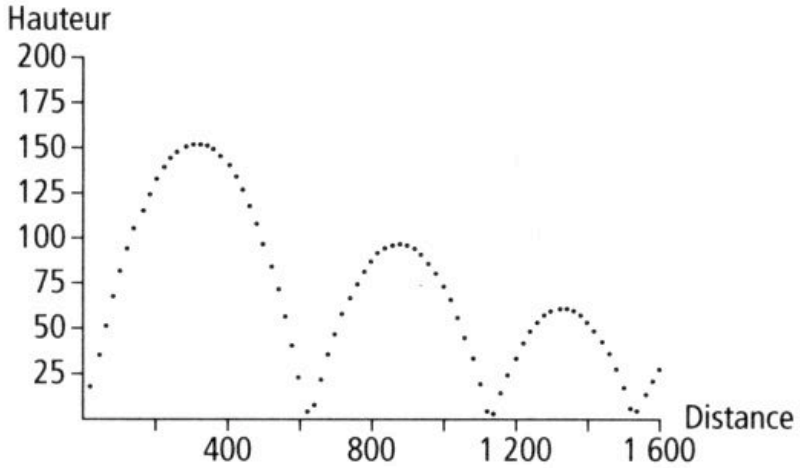
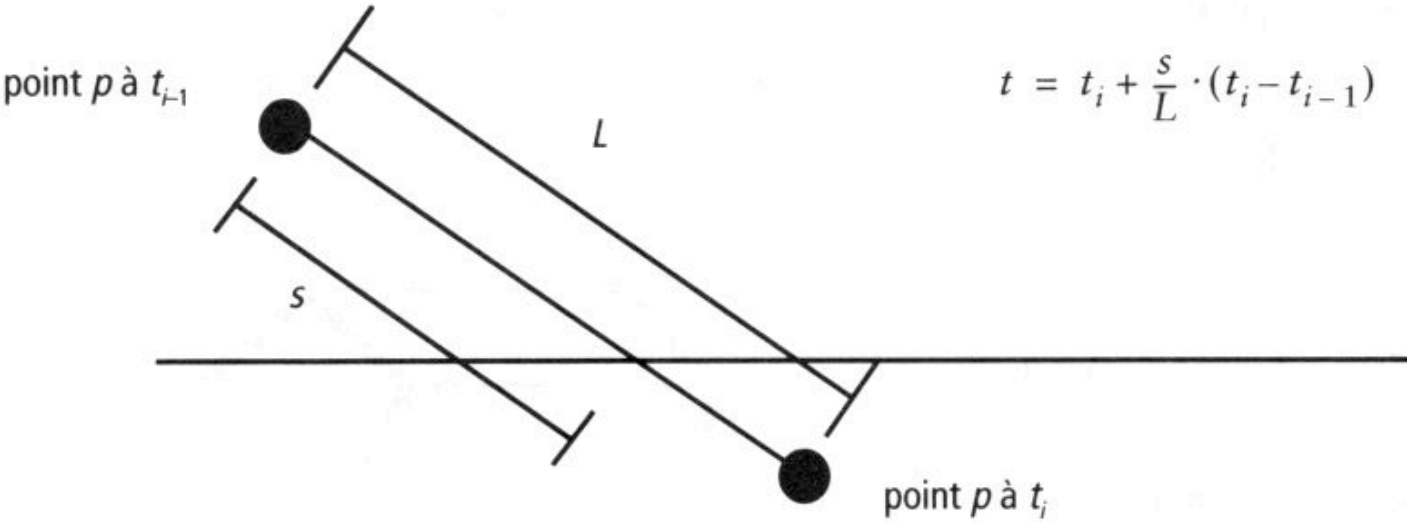


Figure 4.33 **Réponse cinématique à des collisions avec le sol en utilisant un facteur d'atténuation de 0,8 pour l'exemple de la section 4.3.1**

• Méthode par calcul de la force d'impulsion

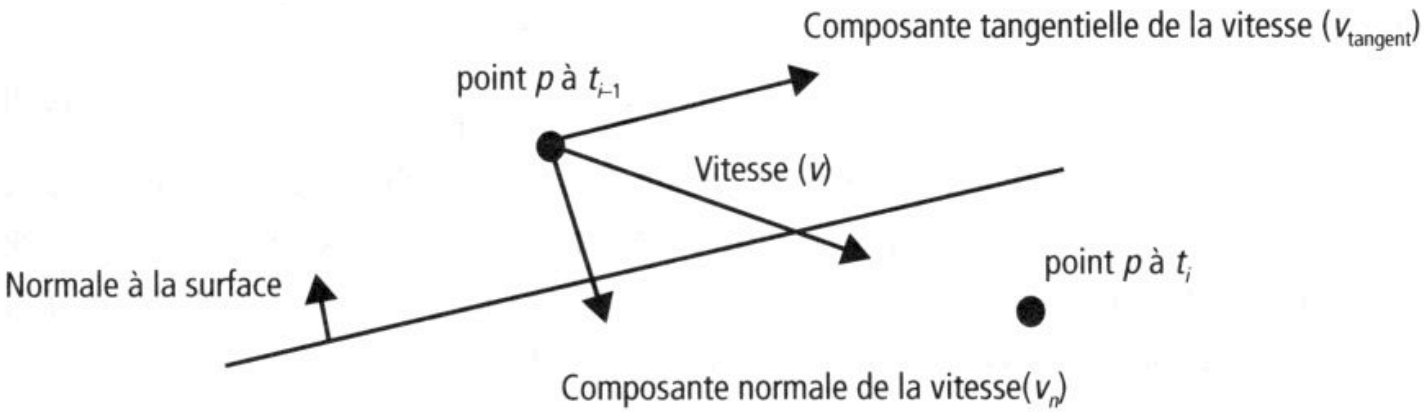
1- remonter le temps au point d'impact



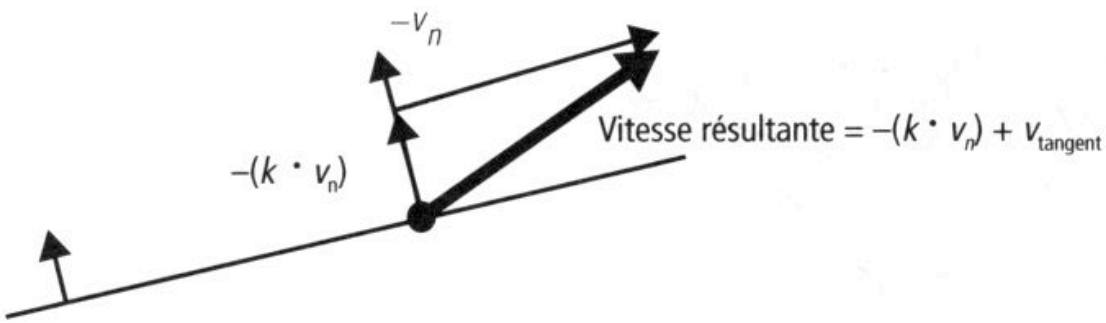
$$t = t_i + \frac{s}{L} \cdot (t_i - t_{i-1})$$

Figure 4.38 **Estimation linéaire de l'instant d'impact, t**

2- calculer les conséquences de l'impact



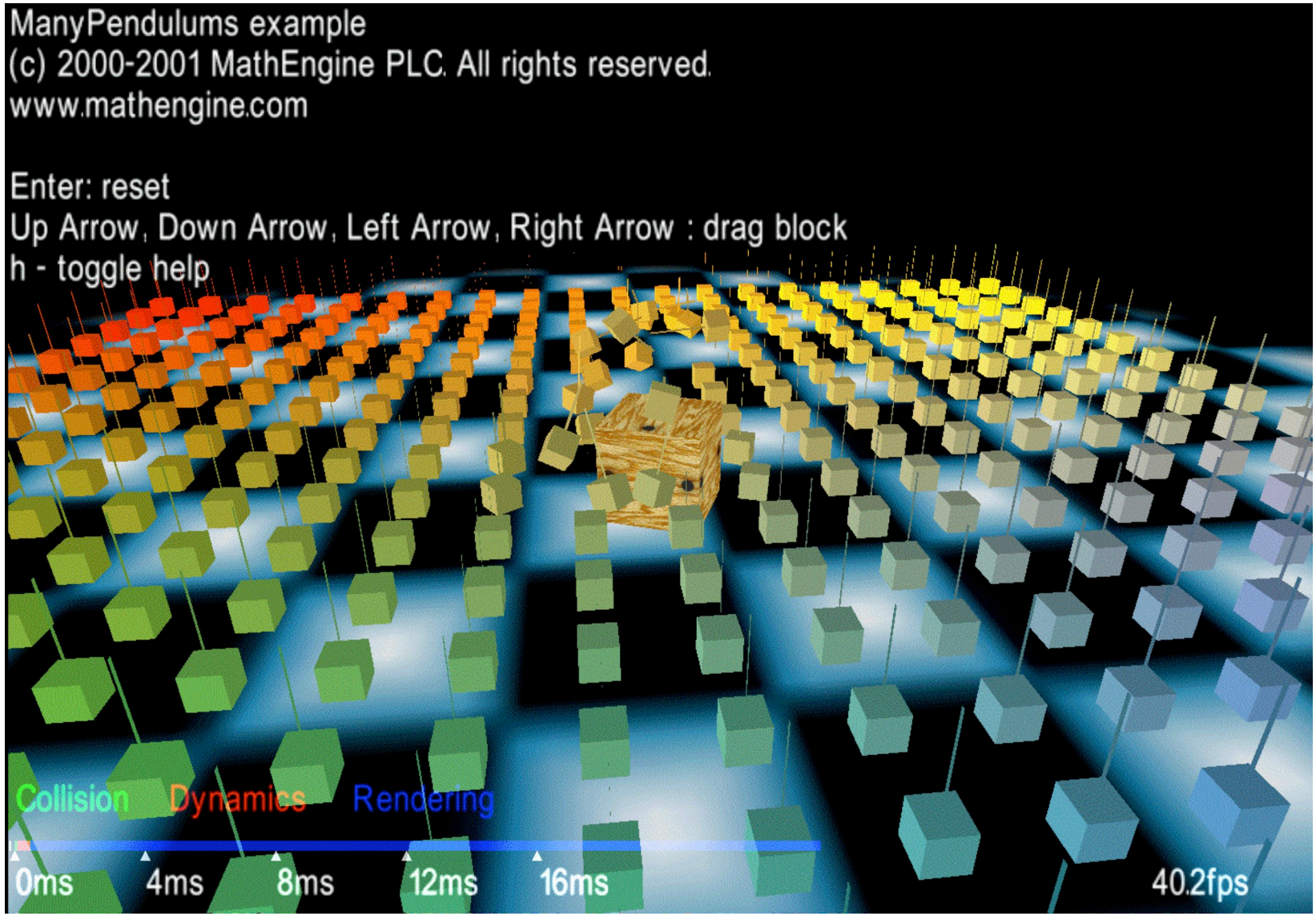
Composantes de la vitesse d'une particule entrant en collision avec un plan

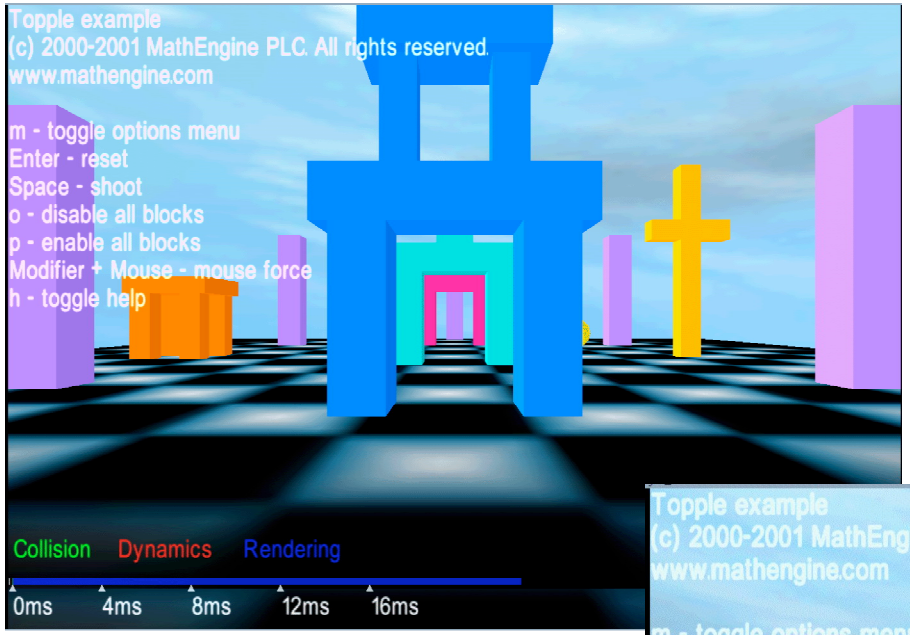


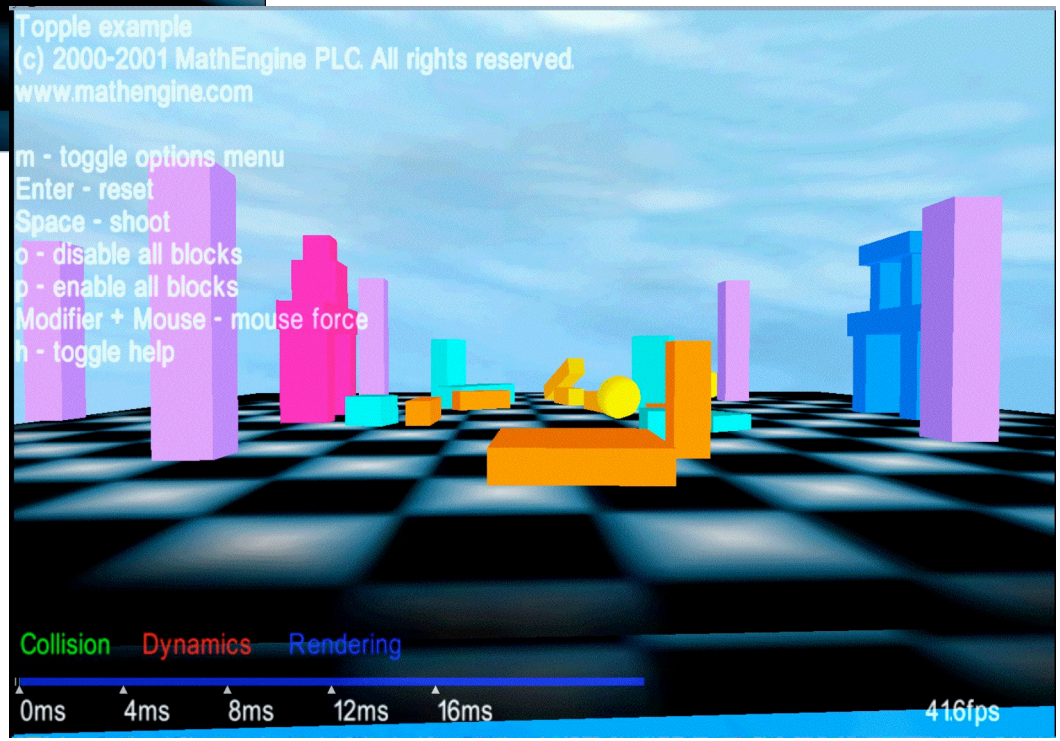
Calcul de la vitesse résultant d'une collision (k est le coefficient de restitution)

Figure 4.39 **Réponse à l'impact entre un point et un plan**

A etudier : Karma (soc. MathEngine) utilisé dans Renderware

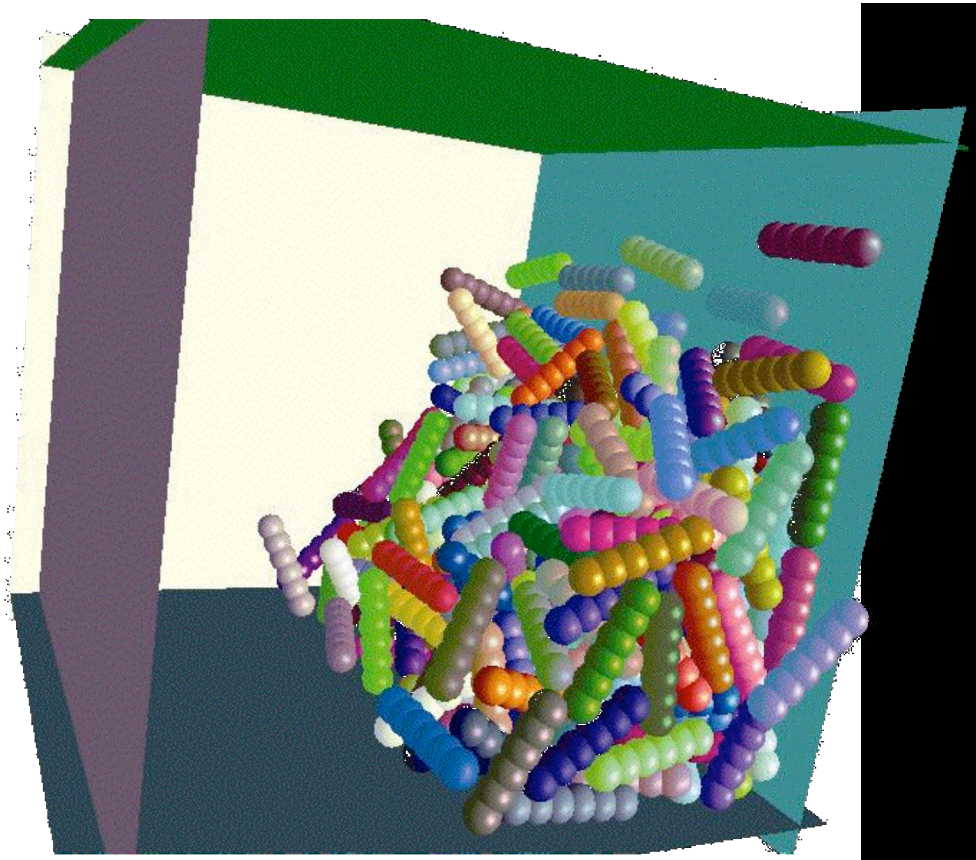


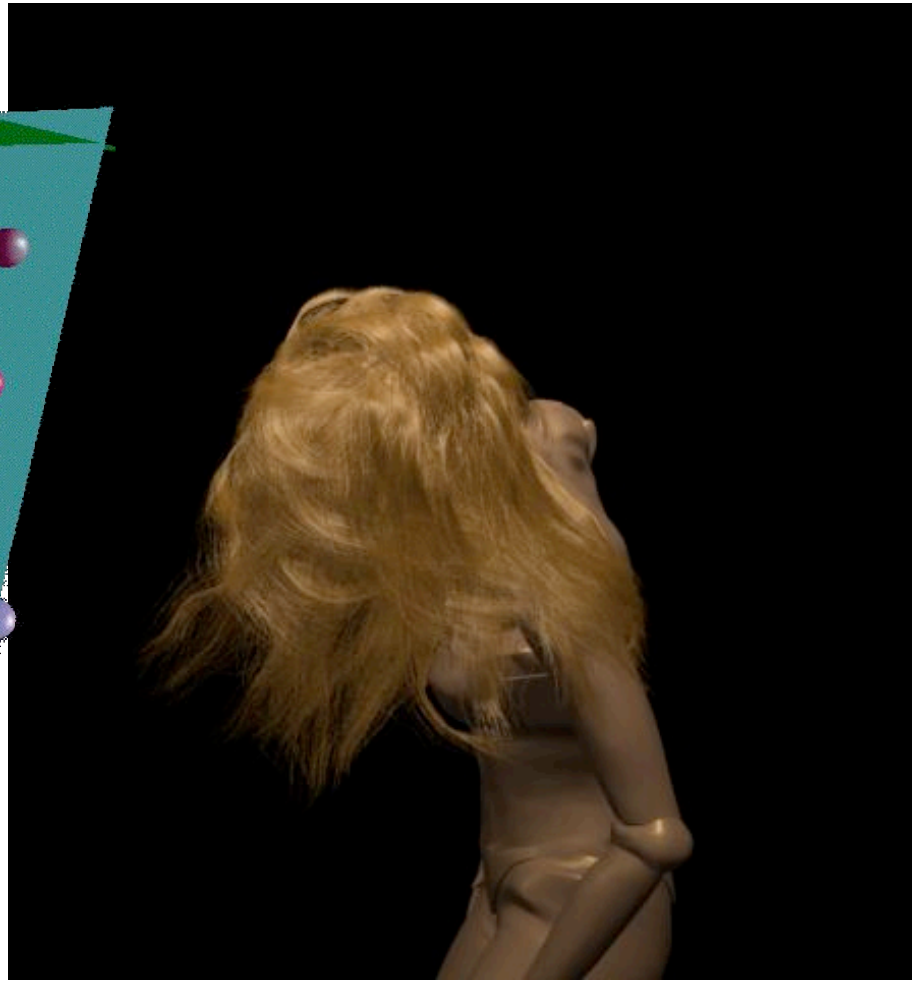

Topple example
(c) 2000-2001 MathEngine PLC. All rights reserved.
www.mathengine.com
m - toggle options menu
Enter - reset
Space - shoot
o - disable all blocks
p - enable all blocks
Modifier + Mouse - mouse force
h - toggle help
Collision
Dynamics
Rendering
0ms
4ms
8ms
12ms
16ms


Topple example
(c) 2000-2001 MathEngine PLC. All rights reserved.
www.mathengine.com
m - toggle options menu
Enter - reset
Space - shoot
o - disable all blocks
p - enable all blocks
Modifier + Mouse - mouse force
h - toggle help
Collision
Dynamics
Rendering
0ms
4ms
8ms
12ms
16ms
416fps

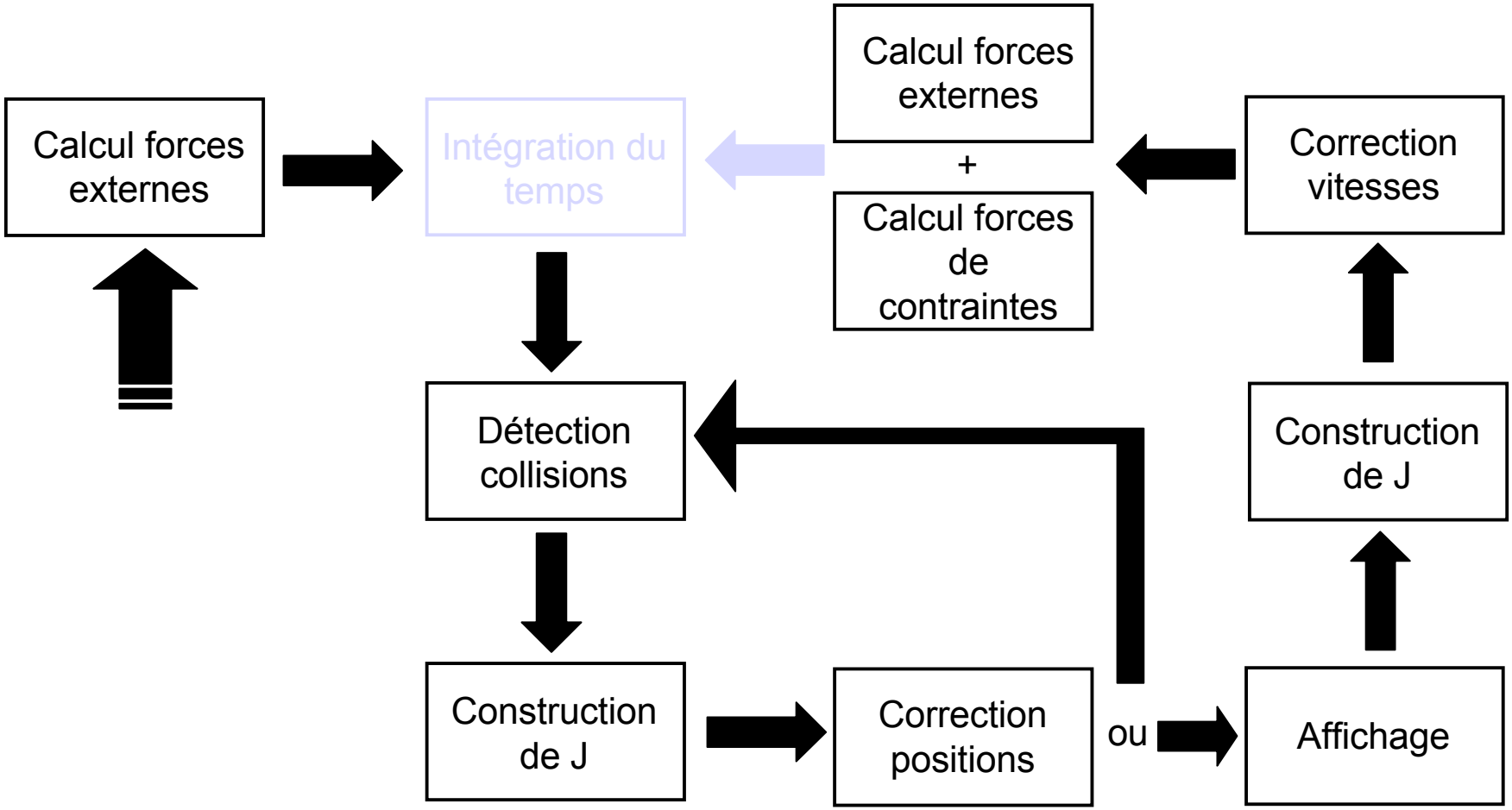
A consulter : présentation de O. GALIZZI (IMAG/iMAGIS)

« Animation de solides en contact par modèle physique »





Boucle de simulation



J : jacobienne des contraintes