Animation par modèles générateurs : contrôle du mouvement

Alexis Lamouret

UNIVERSITE JOSEPH FOURIER - GRENOBLE I

THESE

présentée par

Alexis LAMOURET

pour obtenir le titre de

Docteur en Informatique

de l'Université Joseph Fourier - Grenoble I

Arrêtés ministériels du 5 juillet 1984
et du 30 mars 1992

Animation par modèles générateurs : Contrôle du mouvement

Soutenue le 14 septembre 1995 devant la commission d'examen

MM.	Bernard ESPIAU	Président
	Eugene FIUME	Rapporteur
	Bruno ARNALDI	Rapporteur
	Marie-Paule GASCUEL	
	Claude PUECH	

Thèse préparée au sein du laboratoire *iMAGIS/IMAG*
iMAGIS est un projet commun entre le *CNRS*, l'*INRIA*, l'*INPG* et l'*UJF*.

Table des matières

Table des figures

Remerciements

En tout premier lieu, je tiens a remercier Rachel Orti, qui m'a aidé à traverser les moments les plus difficiles de la préparation de ce travail, et qui incidemment a été une des rares personnes qui sont parvenues à relire le document en entier!

Je tiens à remercier tout particulièrement Eugene Fiume et Bruno Arnaldi qui ont acceptés d'être rapporteurs de ce travail, et m'ont prodigué des conseils et commentaires d'une grande valeur, ainsi que Bernard Espiau, qui m'a fait l'honneur de présider mon jury.

Marie-Paule Gascuel m'a tout au long de cette thèse apporté ses conseils, son soutien et son enthousiasme. Je l'en remercie profondément, de même que Claude Puech, dont les conseils sur un plan différent m'ont été tout aussi profitables, et enfin Michiel van de Panne, qui m'a apporté une aide considérable dans la réalisation de la derniere partie de cette thèse.

Un grand merci à Dominique Gascuel, grâce à qui l'implantation de mon travail au sein du logiciel *Fabule* s'est effectuée dans les meilleures conditions, et qui est fournisseur attitré à la fois de bons conseils et d'une aide technique toujours efficace.

Je remercie chaleureusement toute l'équipe *iMAGIS*, dans un ordre arbitraire qu'ils ne manqueront pas d'interpréter, George, Nicolas, François, Stéphane, François, Jean-Christophe, Frédéric, Nicolas, Frédo et Mathieu, Cyril, ainsi que Ponpon et Alban, sans oublier tous ceux qui sont passés dans cette équipe et ont contribué à l'ambiance et la bonne humeur qui y ont régné.

Le présent document a été établi en éxécution du contrat N^0 91/815/16 passé par la Direction des Recherches, Études et Techniques - Direction Scientifique - Section Soutien à la recherche.

Introduction générale

Telles qu'elles sont actuellement utilisées dans le domaine de la vidéo, des effets spéciaux et des films de synthèse, les techniques d'animation en synthèse d'images dérivent dans une grande mesure des techniques d'animation traditionnelles, utilisées pour le dessin animé. Ainsi, le processus de création d'une animation commence par la donnée d'un scénario écrit, ensuite transformé en un script plus détaillé, représentant par des dessins les différentes séquences de l'animation, l'angle de vue et autres données visuelles. Ce script est transformé à son tour en un certain nombre d'images-clés, choisies pour leur importance dans le mouvement final; ces images-clés sont enfin complétées par toutes les images intermédiaires nécessaires pour produire l'animation finale.

Ce principe, valable pour le dessin animé, le reste pour l'animation en image de synthèse. Seules les deux dernières étapes diffèrent, et uniquement par leur mode de réalisation. L'ensemble des techniques procédant selon ce principe sont généralement désignées comme les *techniques descriptives*, parce que leur principe est de décrire trés précisément toutes les étapes d'une animation, le travail de l'ordinateur se bornant alors à calculer un certain nombre d'images intermédiaires à celles définies par l'utilisateur, par des techniques plus ou moins simples d'interpolation.

A l'inverse de ces méthodes, de nombreux modèles issus de la recherche récente en synthèse d'images proposent d'automatiser la création d'animations à partir d'un petit nombre de données initiales. Ces modèles sont dits *générateurs*, et parmi eux, nous nous intéresserons plus particulièrement aux modèles qui, pour apporter cet automatisme, font appel aux équations de la dynamique : les *modèles physiques*, ou *dynamiques*.

Chacune de ces deux méthodes, *techniques descriptives* ou *modèles physiques*, présentent un certain nombres de points forts et de points faibles, que nous nous allons nous efforcer de dégager.

Considérons tout d'abord le cas des techniques traditionnellement utilisées dans l'industrie, les techniques descriptives. Comme nous l'avons vu, il s'agit de se baser sur un certain nombre d' « images-clés » représentant des étapes importantes de l'animation, de les spécifier complètement, et ensuite de réaliser toutes les images intermédiaires. Alors que dans le cinéma d'animation le modèle est le dessin en deux dimensions de la scène, en synthèse d'image on utilise une description en trois dimensions des objets représentés.

L'utilisateur spécifie donc les positions et orientations-clés à certains instants de l'animation (ainsi que les formes-clés pour les objets déformables),

qui sont jugées comme pertinentes. Les étapes intermédiaires sont calculées par interpolation.

La qualité essentielle de ce type d'approche est qu'il fournit à l'utilisateur un contrôle total sur l'animation qu'il définit : il décrit exactement ce qu'il souhaite voir se réaliser, et peut donc créer des animations aussi diverses et aussi complexes qu'il peut l'imaginer.

La méthode comporte cependant un certain nombre d'inconvénients majeurs, tous liés au fait qu'il s'agit de méthodes purement descriptives. Tout d'abord, la spécification d'un grand nombre de positions est nécessaire pour obtenir une animation précise et réaliste. Ensuite, les collisions et les interpénétrations entre objets ne sont ni testées, ni à plus forte raison traitées. Par ailleurs, la vitesse des objets doit être entièrement spécifiée par l'utilisateur. Enfin, aucune aide n'est apportée au réalisme, et en particulier à la conformité du mouvement aux lois physiques qui le régissent dans le monde réel. La qualité du mouvement obtenu repose donc de façon considérable sur le savoir-faire du graphiste, et la mise en place de l'animation demande en conséquence une attention et un temps en rapport.

A l'opposé, les méthodes plus récentes tentent d'automatiser la synthèse des mouvements et des déformations. L'utilisateur décrit, en plus de l'aspect extérieur des objets, certaines caractéristiques physiques qui lui sont associées, comme sa masse, sa matrice d'inertie, et sa raideur. La simulation complète est ensuite calculée à partir de ces paramètres.

Les avantages majeurs de cette approche sont qu'elle génère automatiquement un mouvement à partir de peu de paramètres initiaux, qu'elle permet de détecter et de traiter automatiquement les collisions et les contacts prolongés avec d'autres objets, et enfin que le mouvement obtenu possède un certain réalisme a priori, puisque basé sur une simulation physique.

Le problème majeur pour réaliser des animations complexes en utilisant des modèles physiques est de trouver un moyen de donner, au cours du temps, les paramètres nécessaires à l'animation. Si dans le cas des objets inertes, la spécification des paramètres initiaux suffit, en revanche pour des personnages munis de muscles ou des objets munis de moteurs se pose le problème de la spécification au cours du temps des paramètres liés à ces muscles ou à ces moteurs : les forces et les couples qu'ils produisent.

Bien que porteurs d'un grand potentiel, en raison de la simplification du travail apportée par l'automatisation et l'adéquation aux lois physiques, les modèles générateurs ne sont pas à l'heure actuelle largement utilisés dans les logiciels d'animation industriels. La raison principale est due au type d'interface qu'ils offrent : l'utilisateur d'un module d'animation doit fournir pour chaque objet un modèle physique (comprenant des paramètres comme la masse, l'inertie, la raideur etc...), les conditions initiales, et un ensemble de forces extérieures. Cette interface est clairement inappropriée pour un graphiste, qui a une idée précise du scénario qu'il veut voir se dérouler, mais ne dispose d'aucun moyen pour déterminer quelles forces doivent être appliquées aux objets durant l'animation afin d'obtenir le mouvement désiré.

De nombreux exemples dans le monde du cinéma et des effets spéciaux montrent que le réalisme des objets et des créatures animées est un des points essentiels de l'animation par ordinateur. Si certains traits sont détournés de leur base physique, pour obtenir un effet de caricature par exemple, un grand nombre de processus demeurent liés de façon évidente aux lois physiques. Ainsi, pour les contacts entre les objets, même s'ils subissent un traitement particulier, ils est nécessaire de les détecter, et bien souvent suffisant de les traiter de façon automatique.

Ainsi, on peut considérer qu'un des souhaits du graphiste sera d'avoir à sa disposition des outils pour automatiser ces lois, lui permettant de concentrer son attention sur les points vitaux de l'animation. Mais en même temps il souhaite conserver le niveau de contrôle auquel il est habitué, c'est-à-dire la possibilité de réaliser un script décrivant assez précisément les actions à accomplir dans la séquence d'animation. Pour cela, conserver la définition de positions-clés permet de garder ce contrôle, ainsi que l'interface auquel il est habitué. Enfin, durant la création de l'animation, le graphiste souhaite pouvoir voir les résultats du changement de chaque paramètre en un temps le plus court possible, idéalement en temps réel.

Le but de cette thèse est de combler le fossé qui existe habituellement entre la recherche portant sur les modèles physiques, et les techniques traditionnellement utilisées dans l'industrie. Pour cela, nous proposons un outil qui regroupe les qualités décrites dans le paragraphe précédent, à savoir l'introduction des modèles physiques en mème temps que l'utilisation de positions-clés, avec une rapidité d'éxécution permettant l'interactivité.

La première partie présente un état de l'art, mettant en valeur les points forts et les faiblesses des méthodes existantes dans le domaine du contrôle de l'animation, notamment en ce qui concerne la capacité de contrôle et le degré d'automatisme de ces méthodes.

Dans la deuxième partie est développé un modèle de contrôle, basé sur le suivi d'une trajectoire de référence donnée par l'utilisateur. Cette trajectoire est corrigée en fonction des événements survenus durant l'animation et des paramètres physiques des objets. Un outil est ensuite développé, qui permet de décrire une séquence d'animation complète à l'aide d'un scénario, comprenant une succession de rendez-vous temporels, complètement ou partiellement ordonnés.

La dernière partie se propose de réutiliser certains de ces résultats pour résoudre un problème beaucoup plus précis, celui de la marche de créatures articulées, rendu très complexe par l'habitude qu'a tout observateur pour ce type de mouvement, et en particulier pour la marche humaine.

Première partie

État de l'art

Introduction

Comme nous l'avons vu dans l'introduction générale, les modèles physiques proposent une alternative prometteuse aux approches purement descriptives, mais en même temps présentent un certain nombre d'invonvénients, qui les rendent difficiles à utiliser.

Le but de cet état de l'art est de donner, non pas une liste exhaustive des modèles déjà existants, mais plutôt une vue d'ensemble des différents types d'approche, en insistant sur les objectifs de chacune, ce qu'elles apportent au domaine, et quelles sont leurs limitations.

Parmi les premières approches proposées impliquant l'utilisation de lois physiques, un certain nombre de modèles ont cherché à introduire ces lois, non pas comme un moteur d'animation complet, mais comme un ensemble de règles, plus ou moins simplifiées, permettant de guider la conception d'animations plus réalistes. L'objet du premier chapitre de cet état de l'art est de présenter ces modèles.

Les recherches ont parallèlement conduit un nombre croissant d'études à se porter sur l'utilisation des lois de la dynamique comme un outil de simulation, utilisant pleinement le système différentiel d'équations pour produire un mouvement à partir de peu de paramètres initiaux. Après avoir décrit rapidement les systèmes d'intégration numériques qui permettent de résoudre ce système pour des objets rigides isolés, le deuxième chapitre détaille les différentes méthodes proposées pour animer des structures articulées, et pour gérer les interactions de ces structures avec leur environnement.

Enfin, une fois le problème des structures articulées résolu, il reste un écueil majeur à la production d'animations complexes : comment animer, non plus des objets inertes, mais des créatures munies de muscles, des véhicules munis de moteurs, bref des objets actifs? Pour cela, il est nécessaire de développer des techniques de contrôle, permettant en fonction de critères prédéfinis de calculer l'action à exercer au cours du temps pour produire le mouvement voulu. C'est l'objet du chapitre trois.

Notations

Afin de permettre une meilleure compréhension du texte, je me suis efforcé dans cet état de l'art d'uniformiser au maximum les notations utilisées par les différents auteurs présentés, et de réutiliser ces notations dans la présentation

de mon travail. Dans tout le document j'utiliserai les notations suivantes, où le vecteur vitesse angulaire $\vec{\omega}$ est lié à la matrice de vitesse de rotation par la formule : $\forall \vec{a} \in I\!R^3 \quad \tilde{\omega}.\vec{a} = \vec{\omega} \wedge \vec{a}$

Symbole	Description
m	masse
J	matrice d'inertie
F	force
$\mathcal{C}$	couple
v	vecteur vitesse linéaire
ω	vecteur vitesse angulaire
$\tilde{\omega}$	matrice vitesse de rotation
x	vecteur de position
R	matrice d'orientation
δt	pas de temps (variable)
Δt	pas de temps (fixe)

Les notations plus spécifiques, utilisées pour des modèles précis décrits dans l'état de l'art, feront l'objet de définitions complémentaires au moment où elles apparaîtront.

Chapitre 1

De la cinématique à la dynamique

Tout en conservant les préceptes de l'animation traditionnelle, basée sur la définition de positions-clés, certains auteurs ont proposé des modèles permettant de prendre en compte un certain nombre d'aspects de la dynamique, par l'introduction de règles plus ou moins complexes permettant de guider l'utilisateur vers la production d'animations plus réalistes, sans toutefois utiliser ces lois de façon stricte, par l'intégration des équations associées. Les deux modèles présentés ici suivent ce principe, l'un dans le cadre très général de définitions de courbes d'interpolation, et l'autre pour une application beaucoup plus précise, axée sur l'animation de créatures articulées.

1.1 Splines dynamiques

Pintado et Fiume [PF88] proposent un modèle basé sur des splines calculées dynamiquement, afin d'obtenir certains comportements physiques sans pour cela intégrer les équations de la dynamique. A cet effet ils introduisent la notion de *champs d'influence*, agissant sur les objets en mouvement et déterminant leur position future.

Le principe des splines dynamiques est de décrire la trajectoire d'un objet au cours du temps, an calculant sa position future à l'aide de ses positions et vitesses à l'instant courant, et du champ d'influence auquel il est soumis. Afin de faciliter les calculs, les segments de splines sont définis dans la base de polynômes de Hermite $(H_i)_{0\leq i\leq 3}$, tels que :

valeur	H_0	H_1	H_2	H_3	H'_0	H'_1	H'_2	H'_3
$t=0$	1	0	0	0	0	0	1	0
$t=1$	0	1	0	0	0	0	0	1

La position P de l'objet est évaluée pour un ensemble de valeurs discrètes du temps, $(t_i)_{0\leq i\leq n}$, et la trajectoire de l'objet est interpolée entre deux instants t_i et t_{i+1} par un segment de spline décrit dans la base de Hermite.

Soient P et v la position et la vitesse d'un objet soumis à un ensemble de champs d'influences f_k, qui à une position associent un vecteur :

$$\begin{array}{rccl} f_k : & \mathcal{S} \in I\!R^3 & \rightarrow & I\!R^3 \\ & P & \rightarrow & f_k(P) = V_k \end{array}$$

Soient $P_{i,d}$, $P_{i,f}$, $v_{i,d}, v_{i,f}$ les positions et vitesses en début et fin du morceau de spline i, autrement dit les coefficients dans la base de Hermite. Le passage du morceau de spline $i-1$ au morceau de spline i, vérifie :

$$\left\{ \begin{array}{rcl} P_{i,d} & = & P_{i-1,f} \\ v_{i,d} & = & v_{i-1,f} \\ v_{i,f} & = & (1-\mathcal{F})(v_{i-1,f} + \sum \mathcal{A}_k f_k(P_{i,d})) \\ P_{i,f} & = & P_{i,d} + \frac{1}{2}(v_{i,d} + v_{i,f}) \end{array} \right. \tag{1.1}$$

ou les paramètres scalaires $\mathcal{F}$ et $\mathcal{A}_k$ modélisent respectivement le frottement et l'importance du champ f_k. Les deux premières équations sont obtenues par continuité C^1 de la courbe, la troisième définit l'action dans le modèle des champs d'influence, et la dernière exprime la relation position-vitesse. Cette dernière équation pourrait être modifiée pour obtenir une continuité C^2, mais les auteurs jugent les résultats obtenus alors moins satisfaisants.

Avec ces équations on obtient, en particulier, pour un champ d'influence unique et constant $f_0(t) = V_0$, un modèle du champ de pesanteur. D'autres possibilités de champs d'influence sont étudiées permettant d'obtenir des effets plus complexes.

Afin de montrer les capacités du système, et en particulier la possibilité de modéliser des comportements d'interaction entre objets plus complexes que la simple attraction-répulsion, les auteurs proposent de détailler deux types de guidage d'un objet par un autre. Le principe est ici de définir une métrique, scalaire ou vectorielle, entre l'objet cible et l'objet guidé; ensuite, de définir une erreur en fonction de cette métrique, enfin un champ d'influence fonction de cette erreur, exprimé par un contrôleur proportionnel-dérivé, ou proportionnel-integral-dérivé.

Plus concrètement, voyons tout d'abord le cas d'une métrique vectorielle $m(\mathcal{C}, \mathcal{O}, t)$ définie entre un objet-cible $\mathcal{C}$ un objet guidé $\mathcal{O}$ à chaque instant t par $m(\mathcal{C}, \mathcal{O}, t) = P_{\mathcal{O}} - P_{\mathcal{C}}$. Le terme d'erreur s'exprime par :

$$\varepsilon(t) = m(\mathcal{C}, \mathcal{O}, t) - m(\mathcal{C}, \mathcal{O}, 0)$$

et le champ d'influence :

$$f = \alpha(\varepsilon(t) + \beta\frac{\Delta\varepsilon}{\Delta t}) \text{ ou encore } f = \alpha(\varepsilon(t) + \beta\frac{\Delta\varepsilon}{\Delta t} + \gamma\sum\varepsilon)$$

α, β et γ sont des paramètres permettant de régler le comportement de $\mathcal{O}$. Cet ensemble permet de fixer pour but à l'objet guidé de conserver un certain vecteur d'écart entre l'objet et sa cible, pour suivre une crête sans s'en approcher, dans l'exemple décrit par les auteurs.

Le second cas proposé est celui d'une métrique scalaire, définie par la distance euclidienne d entre l'objet guidé et l'objet-cible. Avec ε défini par $\varepsilon = d(P_{\mathcal{C}}, P_{\mathcal{O}}) - d_{ref}$, où d_{ref} est une distance de référence qu'on veut voir maintenue au cours du temps, on définit le champ par :

$$f = \alpha(\varepsilon(t) + \beta\frac{\Delta\varepsilon}{\Delta t})\frac{P_{\mathcal{O}} - P_{\mathcal{C}}}{||P_{\mathcal{O}} - P_{\mathcal{C}}||}$$

On obtient alors un objet qui cherche à rester à une distance constante de l'objet-cible, sans contrainte sur la direction.

Discussion

L'approche proposée ici permet d'utiliser certains aspects des modèles physiques, tout en évitant l'intégration des équations de la dynamique. Au moyen de la définition de champs d'influence, on arrive à modéliser des comportements physiques simples comme la loi de pesanteur, ou des interactions plus complexes entre les objets de la scène.

Cependant le modèle, s'il permet de créer des comportements tels qu'éviter un obstacle, ou même en suivre le contour, ne permet pas de traiter une collision effective. De plus il nécessite la spécification d'un ensemble de lois d'interaction pour chacune des paires d'objets de la scène. Si ce procédé augmente sensiblement la rapidité des calculs, il supprime du même coup l'automatisme du processus, en obligeant l'utilisateur à considérer attentivement les objets qu'il souhaite voir interagir. Enfin, tous les exemples sont réalisés avec des objets modélisés par des points matériels, et il semble difficile de pouvoir étendre de façon simple le modèle à l'animation d'objets articulés complexes, ou d'objets déformables.

1.2 Animation de personnages

L'approche de Girard [Gir87] est d'offrir à l'utilisateur un outil pour la construction interactive de toutes les étapes de la modélisation de la marche de personnages, en utilisant les lois de la dynamique comme des aides au réalisme. Le modèle présenté est prévu pour des bipèdes ou quadrupèdes, ou encore pour des créature ayant n'importe quel nombre de paires de jambes, pourvu que le corps soit rigide.

L'idée de base de Girard est qu'une créature articulée planifie son mouvement à l'avance de façon cinématique, la dynamique intervenant par la suite pour réaliser le plan ainsi déterminé. L'utilisateur va donc spécifier de façon cinématique un certain nombre de données, le système utilisant par ailleurs des physiques et comportementales, issues pour ces dernières de l'observation de données empiriques, pour compléter la spécification du mouvement et le rendre plus naturel.

La construction du modèle se fait de façon progressive. Tout d'abord, l'utilisateur définit le mouvement des membres par un ensemble de positions-clés,

données dans le repère de la hanche. Il peut de plus spécifier certaines des positions dans le repère du monde, de façon à avoir par exemple un pied fixé au sol durant un certain temps. Un graphe éditable de la distance en fonction du temps permet ensuite de régler les vitesses plus finement. Une fois ce graphe obtenu, il peut être globalement étiré ou compressé le long de l'axe du temps pour changer la vitesse d'éxécution du mouvement, sans modifier ses caractéristiques internes. Girard évoque ici la possibilité de résoudre, une fois les positions-clés données, la vitesse et la position en fonction du temps, en minimisant un critère de fluidité du mouvement, mais cette partie n'a pas été implantée.

Un ensemble de trajectoires ainsi définies pour chaque membre constitue une démarche pour l'individu, ce qui permet dans un premier temps à l'utilisateur de créer sa propre bibliothèque de démarches. Une démarche donnée reste paramétrable, dans la mesure ou sa vitesse peut être globalement étiré ou compresséee. Le système peut ensuite, si l'utilisateur souhaite augmenter le naturel du mouvement, procéder pour une démarche donnée, à des altérations visant à vérifier certaines propriétés, telles que la continuité de la courbe des vitesses au cours du temps. Les trajectoires spécifiées dans le repère de la hanche sont ensuite liées au repère du monde, en utilisant la relation fondamentale de la dynamique pour la partie de la marche sans contact avec le sol, et sinon par une loi d'équilibre : une jambe doit être dans une position de référence stable à la moitié de sa période de support. Ainsi, à partir du choix d'une démarche et d'une courbe de vitesses, on peut maintenant obtenir un mouvement complet des jambes dans le repère du monde. Un module permet de passer automatiquement d'une démarche à une autre, en s'appuyant d'une part sur l'adaptation de la durée d'un cycle entre deux démarches, et d'autre part sur la modification progressive des séquences de contacts au sol.

Une fois une démarche définie pour la marche en ligne droite, Girard propose un module de calcul des positions et orientations du corps, permettant de tourner et donc de suivre une trajectoire, spécifiée dans le plan du sol. Tous les éléments sont ainsi réunis pour produire un mouvement complet. L'utilisateur doit spécifier la séquences de démarches que le personnage doit suivre, la trajectoire dans le plan du sol et une courbe de vitesses associée. Le système calcule d'abord l'enchaînement des placements des pieds au sol, et leur position. Il en déduit, à l'aide de la relation fondamentale de la dynamique, et la contrainte de continuité des vitesses, la position horizontale et verticale du corps au cours du temps. Le maintien de l'équilibre sous l'action de la force centrifuge permet de calculer son inclinaison. Une fois la position et l'orientation du corps déterminée à chaque instant, il ne reste plus qu'à ajouter le mouvement des membres issu de la bibliothèque de démarches, pour obtenir le mouvement final.

Discussion

Le modèle proposé par Girard permet à l'utilisateur de choisir entre la spécification totale de son mouvement - tous les modules automatiques peuvent être court-circuités - et la création guidée par un certain nombre de règles

destinées à rendre le mouvement plus naturel. Certaines lois dynamiques sont ici introduites, mais au même titre que les fonctions comportementales, plutôt que comme moteur de simulation. Leur utilisation est sujette à des approximations importantes : en particulier, pour le mouvement sans contact au sol, l'influence des bras et des jambes est négligée. Enfin, l'absence de réelle simulation empêche, ici aussi, de profiter d'autres avantages des modèles physiques, notamment pour l'interaction entre objets.

Chapitre 2

Méthodes de simulation physique

Pour utiliser de façon plus systématique les lois de la physique, et en faire véritablement le moteur de l'animation, le premier pas à franchir est d'élaborer un modèle qui prenne en compte toutes les caractéristiques physiques des objets, et qui produise en fonction de ces caractéristiques une séquence d'animation. Pour cela, la première chose à faire est de transformer les lois de la dynamique, par essence continues, en un schéma d'intégration discret, propre à exprimer l'état du système à un instant en fonction de son état à l'instant précédent.

2.1 Simulation de la dynamique

Cette section rappelle les équations fondamentales de la dynamique du point et du solide, et indique les schémas d'intégration les plus couramment utilisés, en particulier le schéma d'intégration de Newton, utilisé dans cette thèse.

Équations fondamentales du mouvement

Physique du point : Un point matériel est défini par sa position x dans le repère du monde, et sa masse m. Son animation s'effectue un intégrant numériquement au cours du temps la relation fondamentale de la dynamique, qui s'écrit :

$$F = m\ddot{x}$$

où F représente la somme des forces extérieures appliquées au point.

Physique du solide : Un solide est défini par la donnée d'un repère local (matrice d'orientation R et position x par rapport au repère du monde), ainsi que par sa masse m et sa matrice d'inertie J. Son mouvement est régi par le système d'équations :

$$F = m\ddot{x} \tag{2.1}$$
$$\mathcal{C} = J\dot{\omega} + \omega \wedge J\omega \tag{2.2}$$

où $\mathcal{C}$ est la somme des couples appliqués au centre d'inertie du solide, et ω son vecteur vitesse angulaire.

Schémas d'intégration

Pour résoudre les équations du mouvement la plupart des modèles de simulation discrétisent le temps afin d'obtenir une approche discrète, le but final étant de produire une séquences de 25-60 images par seconde et non de résoudre un système différentiel compliqué.

Deux schémas d'intégration des équations du mouvement sont présentés ici.

Schéma d'Euler : Il s'agit d'une intégration du premier ordre des équations du mouvement :

$$\begin{aligned}
\dot{x}(t+\delta t) &= \dot{x}(t) + \frac{F(t)}{m}\,\delta t \\
x(t+\delta t) &= x(t) + \dot{x}(t+\delta t)\,\delta t \\
\omega(t+\delta t) &= \omega(t) + J^{-1}\left[\mathcal{C}(t) - \omega(t)\wedge J\omega(t)\right]\,\delta t \\
R(t+\delta t) &= \left[I + \delta t\,\tilde{\omega}(t+\delta t)\right]\,R(t)
\end{aligned}$$

où $\tilde{\omega}$ est la matrice de vitesse de rotation associée à ω. Cette matrice est caractérisée par : $\forall a \in I\!R^3\ \ \tilde{\omega}.a = \omega \wedge a$. I désigne la matrice identité.

Schéma de Newton : Il est conçu de manière à donner une solution exacte pour des forces et couples qui seraient constants pendant un intervalle de temps δt (voir [PTVF92]).

$$\begin{aligned}
\dot{x}(t+\delta t) &= \dot{x}(t) + \frac{F(t)}{m}\,\delta t \\
x(t+\delta t) &= x(t) + \frac{1}{2}\left[\dot{x}(t) + \dot{x}(t+\delta t)\right]\delta t \\
\omega(t+\delta t) &= \omega(t) + J^{-1}\left(\mathcal{C}(t) - \omega(t)\wedge J\omega(t)\right)\,\delta t \\
R(t+\delta t) &= \left[I + \frac{1}{2}(\tilde{\omega}(t) + \tilde{\omega}(t+\delta t))\,\delta t\right]\,R(t)
\end{aligned}$$

On peut en particulier prouver facilement que les objets en chute libre auront avec ce shéma d'intégration un échantillonnage de la trajectoire discret, mais exact, ce que ne propose pas le premier schéma.

Bien qu'elles soient à la base de la plupart des approches physiques, les équations fondamentales de la dynamique ne suffisent pas à animer des objets articulés. Les modèles présentés ci-dessous apportent diverses solutions à ce problème.

2.2 Simulation de chaînes articulées

Armstrong et Green [AG85] proposent de décrire un système articulé par une structure arborescente. La racine a six degrés de liberté, et chaque fils a trois degrés de liberté en rotation par rapport à son père. La résolution est récursive. Elle se fait en écrivant les équations de la dynamique 2.1 et 2.2 pour chaque objet r composant la structure articulée (qu'on appellera par la suite un lien), et une relation de dépendance entre l'accélération de chaque fils s et celle de son père r, due au fait que le centre du repère s est fixe dans le repère r.
On peut réécrire le système sous la forme :

$$\dot{\omega}_r = K_r a_r + l_r \tag{2.3}$$

$$F_r = M_r a_r + n_r \tag{2.4}$$

où K_r, l_r, M_r et n_r sont des variables calculées en fonction de K_s, l_s, M_s et n_s à l'aide de l'équation :

$$R_s a_s \;=\; \omega_r \wedge \omega_r \wedge x_s \;+\; a_r \;-\; x_s \wedge \dot{\omega}_r \tag{2.5}$$

où x_s désigne le vecteur coordonnées du centre du repère propre de s dans le repère de r.

En remontant les relations de dépendances accélération père-fils 2.5 dans les équations 2.3 et 2.4 on arrive à calculer l'accélération de la racine r_0 en translation et en rotation, puis on redescend pour obtenir celle de chacun des fils jusqu'aux feuilles. On calcule ensuite en fonction de l'accélération à l'instant courant et des positions et vitesses passées les positions et vitesses à l'instant courant, suivant le schéma d'intégration d'Euler décrit plus haut.

Cette résolution, bien que rapide puisque basée sur un parcours double de l'arborescence, pose cependant quelques problèmes. Tout d'abord, les objets doivent nécessairement être descriptibles sous forme d'une arborescence, ce qui exclut toute possibilité de boucle fermée. Ensuite, ils possèdent nécessairement trois degrés de liberté en rotation à chaque articulation, ce qui restreint le type d'objet à animer. Enfin, il s'agit bien ici de simulation et non de contrôle, et aucune aide n'est fournie à l'utilisateur pour exprimer les forces et couples externes à appliquer sur chaque lien pour réaliser un mouvement donné.

Featherstone [Fea83], puis Latrop [Lat86] utilisent une notation spatiale qui englobe les équations du mouvement en translation et en rotation dans un même formalisme. Comme pour Armstrong et Green, on obtient un système de relations entre accélérations des pères et des fils, que l'on résout en remontant depuis les feuilles pour trouver l'accélération du père, puis en redescendant pour obtenir celles des nœuds jusqu'aux feuilles.

2.3 Simulation par contraintes

Barzel et Barr [BB87, BB88], puis Gascuel et Gascuel [GG94] définissent des contraintes géométriques simples entre les liens du système articulé, considérés

comme des objets indépendants. Ces contraintes peuvent être du type point-point (comme indiqué en figure 2.1), mais aussi plus complexes comme point-courbe (chez Barzel et Barr), point-surface (chez Gascuel et Gascuel), angulaires etc...

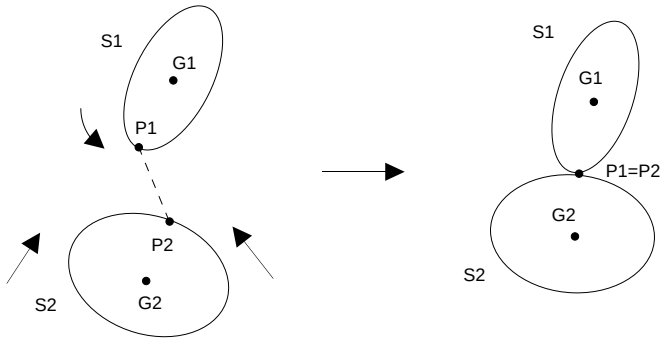


FIG. 2.1 - *Contrainte « point-point »*

Une contrainte « point-point » entre deux solides S_1 et S_2 crée une charnière avec trois degrés de liberté en rotation en faisant coïncider deux points, P_1 de S_1 et P_2 de S_2 pendant toute l'animation.

2.3.1 Contraintes résolues par calcul de forces

Barzel et Barr résolvent les contraintes en calculant les forces qu'il faut exercer sur chaque lien pour qu'elles soient vérifiées. Soit D la mesure de la déviation par rapport à la contrainte ($D = 0$ si la contrainte est vérifiée). $\dot{D}$ représente la variation de D, et $\ddot{D}$ son accélération.

$\ddot{D}$ dépend linéairement de F et C, les forces et couples exercés sur le solide. On minimise D en résolvant l'équation :

$$\ddot{D} + \frac{2}{h}\dot{D} + \frac{1}{h^2}D = 0, \text{ soit } D(t) = D_0 e^{-\frac{t}{h}} \tag{2.6}$$

où h est le pas d'intégration, qui doit être plus petit que le pas de temps utilisé dans l'animation. Ainsi, le terme exponentiel permet de voir D se rapprocher très vite de 0, ce qui signifie que les contraintes ne seront jamais vraiment résolues, mais suffisamment approchées pour un résultat satisfaisant dans la pratique. La connaissance de D, ainsi que des forces et couples externes, permet de déduire les forces et couples de contrainte. On en déduit ensuite les positions, orientations et vitesses de chaque articulation. L'algorithme peut être appliqué pour des contraintes multiples, de différents types (voir en début de paragraphe), mais les auteurs ne précisent pas s'il est utilisable pour les boucles fermées. Outre l'animation, ce modèle permet la construction automatique de positions initiales des objets contraints.

2.3.2 Contraintes résolues par déplacements

Gascuel et Gascuel [GG94] résolvent directement les contraintes par des petits déplacements. Étant donné qu'il s'agit du modèle utilisé comme moteur d'animation associé aux outils de contrôle développés dans cette thèse, nous donnons ici une description plus précise des capacités de ce modèle et des techniques utilisées.

Le but de cette approche est de simuler des structures articulées complexes à l'aide d'une méthode simple, rapide et efficace. Ces structures sont construites à partir de solides indépendants liés par des contraintes géométriques similaires à celles utilisées par Barzel et Barr. L'utilisateur est libre de choisir le nombre de degrés de liberté en rotation et en translation aux charnières, et de spécifier des contraintes linéaires et angulaires sur le mouvement. Le graphe des objets contraints peut contenir un nombre quelconque de boucles fermées. Comme pour Barzel et Barr, ce modèle permet la construction automatique de positions initiales des objets contraints.

En revanche, plutôt que de calculer les forces de maintien des contraintes, les auteurs proposent de régler directement les déplacements des objets, pour éviter d'intégrer ces forces durant la série d'itérations nécessaire pour remplir les contraintes.

A chaque pas de calcul, les solides se déplacent comme s'ils étaient indépendants, suivant le schéma d'intégration de Newton détaillé dans le paragraphe 2.1. Ensuite, les contraintes sont résolues par *des réglages itératifs de déplacements*.

Déplacements associés à une contrainte unique

Présentons tout d'abord la méthode dans le cas d'une contrainte unique entre deux solides, du type point-point (voir figure 2.1). Nous expliquerons par la suite comment combiner les actions de contraintes multiples.

Vérifier une contrainte point-point en translatant un solide sans aucune rotation est toujours possible, mais produirait la plupart du temps un comportement non réaliste. Pour trouver la proportion adéquate entre rotation et translation, en fonction des propriétés physiques (masse, inertie) des objets, les auteurs font une analogie avec un élastique qui aurait maintenu la contrainte durant le pas de temps.

Cela conduit au schéma de calcul suivant, où les proportions entre rotation et translation et les déplacements relatifs des deux objets sont cohérents avec leurs paramètres de masse et d'inertie (en considérant les notations de la figure 2.1) :

1. Appliquer respectivement à S_1 et S_2 les petites rotations données par les vecteurs-rotation:

$$\begin{cases} \overrightarrow{\Delta R_1} = \frac{m_1 m_2}{m_1+m_2} J_1^{-1} \left(\overrightarrow{G_1P_1} \wedge \overrightarrow{P_1P_2}\right) \\ \overrightarrow{\Delta R_2} = \frac{m_1 m_2}{m_1+m_2} J_2^{-1} \left(\overrightarrow{G_2P_2} \wedge \overrightarrow{P_2P_1}\right) \end{cases} \tag{2.7}$$

   où les m_i sont les masses, les J_i les tenseurs d'inertie, et les G_i les centres de masse des deux solides.

2. Ensuite, d'après les positions P'_1, P'_2 des points *après rotation*, calculer et appliquer les petites translations satisfaisant exactement la contrainte:

$$\begin{cases} \overrightarrow{\Delta x_1} = \frac{m_2}{m_1+m_2} \overrightarrow{P'_1 P'_2} \\ \overrightarrow{\Delta x_2} = \frac{m_1}{m_1+m_2} \overrightarrow{P'_2 P'_1} \end{cases} \tag{2.8}$$

Les constantes $\frac{m_1 m_2}{m_1+m_2}$ et $\frac{m_i}{m_1+m_2}$ ont été ici choisies pour avoir, d'une part le rapport entre rotation et translation correspondant à celui d'un élastique liant les deux points, d'autre part pour pouvoir résoudre une contrainte unique en un pas de temps.

La méthode peut être étendue afin d'offrir des degrés de liberté en translation aux charnières, pouvant être limités en rayon d'action. Pour contraindre le mouvement de P_1 dans chaque sous-région définie par rapport à S_2, P_2 est remplacé dans les équations (2.7) et (2.8) par le point P'_2 du domaine choisi qui est le plus proche de P_1. « Point-segment », « point-courbe », « point-surface », « point dans une sphère » sont des exemples de contraintes simples, permettant la modélisation d'objets articulés plus complexes.

Dans la méthode présentée jusqu'à maintenant, chaque contrainte laisse trois degrés de liberté en rotation entre les solides. Restreindre les rotations aux charnières peut être utile, et peut être fait avec le même type d'approche. Une fois qu'une paramétrisation (comme par exemple l'utilisation des quaternions) a été choisie pour les orientations, on peut calculer la plus petite rotation pour ramener la « distance angulaire » entre deux objets à une valeur admise. Cette rotation est répartie entre les deux solides en fonction de leurs inerties respectives sur cet axe particulier.

Combinaison des déplacements dus à des contraintes individuelles

En pratique, plusieurs contraintes peuvent être simultanément appliquées à un solide, en conséquence de quoi il est nécessaire de combiner les rotations et les translations produites par chaque contrainte. Une simple somme des déplacements calculés pour chaque contrainte individuelle conserverait le moment de premier ordre, mais conduirait à des divergences dans certains cas particuliers.

Ce problème est résolu en pondérant les déplacements affectés au solide avant d'en faire la somme. Pour obtenir la conservation du moment de premier ordre, le même poids doit être utilisé pour les couples de déplacements produits par la même contrainte. Si S_i et S_j sont deux objets liés par une contrainte donnée, le déplacement associé est pondéré par $\frac{1}{\max(n_i,n_j)}$, où n_i et n_j sont les nombres de contraintes appliquées respectivement à chaque solide.

Une série d'itérations est nécessaire pour remplir les contraintes dès que plus d'une contrainte est appliquée par solide. Ceci est réalisé par un processus itératif qui s'arrête quand toutes les corrections résultantes sont plus petites qu'une marge spécifiée, ou quand le nombre maximum d'itérations est atteint. Limiter le nombre d'itérations évite les impasses quand le système est surcontraint. Habituellement un petit nombre d'itérations suffit, même quand le graphe des objets contraints contient des boucles fermées.

Correction de la cinématique du mouvement des objets contraints

Les corrections dues aux contraintes doivent être prises en compte dans la cinématique du mouvement, comme si nous avions ajouté des forces de contraintes. Une fois qu'une position des solides vérifiant les contraintes avec une précision satisfaisante a été trouvée, nous ajustons leur vitesse linéaire et angulaire en considérant les positions et orientations qu'ils ont effectivement atteint durant le pas de temps :

$$\begin{aligned} v(t+\delta t) &= 2(x(t+\delta t)-x(t))/\delta t - v(t) \\ \tilde{\omega}(t+\delta t) &= 2\left(R(t+\delta t)R^{-1}(t)-I\right)/\delta t - \tilde{\omega}(t) \end{aligned}$$

Discussion

Les deux approches présentées ci-dessus [BB87, GG94] pour résoudre les contraintes permettent l'animation de structures complexes, l'une par calcul des forces, l'autre par calcul direct des déplacements aux charnières. Des structures contraintes assez générales peuvent être construites et animées avec ces méthodes, en particulier des boucles fermées chez Gascuel et Gascuel. L'algorithme proposé par Barzel et Barr est cependant coûteux en temps de calcul, comme le soulignent les auteurs.

Enfin, ces approches, comme celle d'Armstrong et Green, n'offrent pas de contrôle complet sur l'animation. Si la définition de contraintes permet d'animer des structures articulées, et d'imposer certains types de comportement (comme par exemple rester sur une courbe, ou sur un plan), elles n'apportent en revanche pas de solutions pour suivre un scénario donné : elles permettent essentiellement l'animation de structures inertes, soumises uniquement aux forces de gravité.

2.4 Dynamique inverse

Isaacs et Cohen [IC87, IC88] proposent un modèle qui permet, à partir de la connaissance de certains éléments du mouvement d'un objet articulé, de déduire le reste du mouvement par simulation physique. Pour cela ils combinent la dynamique et la dynamique inverse.

Les équations du mouvement sont décrites par le principe des travaux virtuels, qui permet d'exprimer les équations en fonction des degrés de liberté du système, ou coordonnées généralisées. Le système s'écrit sous la forme :
$[M][\ddot{x}] = [F]$ où M, x et F sont les masses, coordonnées et forces généralisées. La résolution se fait en donnant, pour chaque coordonnée, ou bien $\ddot{x}$ (cas de la dynamique inverse), ou bien F (cas de la dynamique).

Pour faciliter la donnée de ces valeurs au cours du temps, les auteurs introduisent des fonctions comportementales, du type « freiner », ou « suivre une trajectoire donnée par positions clés », par exemple.

Le modèle proposé par Isaacs et Cohen permet d'obtenir des animations d'objets complexes, en utilisant à la fois la dynamique, par le calcul des forces et couples et l'intégration des équations de la dynamique, et la cinématique, avec la définition de trajectoires complètes pour certains éléments de l'animation.

Mais le problème des systèmes descriptifs se retrouve dans ce modèle : pour les éléments de mouvement décrits par l'utilisateur, aucun réalisme a priori n'est fourni, et l'utilisateur doit spécifier, pour chaque équation du système matriciel, ou bien la force, ou bien l'accélération à chaque pas de temps. Enfin, la résolution du système matriciel est assez coûteuse, et nécessite, pour chaque nouvel objet, la réécriture des équations dans le système de coordonnées généralisées.

Chapitre 3

Méthodes de contrôle

Les modèles présentés dans le chapitre précédent proposent des méthodes pour animer des objets articulés de façon physique, grâce aux équations de la dynamique, ce qui permet de ne spécifier que peu de paramètres de l'animation, et d'obtenir pour les objets un comportement réaliste.

Mais les objets animés par ces modèles sont des objets inertes, soumis uniquement à des actions externes telles que la gravité, ou à des lois de comportement simples. Le but des méthodes de contrôle, présentées dans ce chapitre, est de permettre l'animation de personnages munis de muscles, ou d'objets munis de moteurs, en fournissant un moyen automatique de calculer au cours du temps les forces et couples exercés par ces muscles ou moteurs, afin de réaliser une action donnée.

De nombreuses approches basées sur ces principes ont été développées ces dernières années. Elles se distinguent par un certain nombre de caractéristiques :

1. Leur objectif : souhaite-t-on obtenir un contrôle très fin de l'animation, un automatisme complet, ou encore une solution intermédiaire?

2. Le contrôle fourni à l'utilisateur : quels sont les paramètres et définitions que l'utilisateur spécifie, et quels sont ses possibilités d'intervention au cours de l'animation?

3. La technique de résolution du problème : pas à pas au cours du temps ou en une grande étape de calcul?

Les méthodes de contrôle présentées dans ce chapitre sont regroupées en fonction du type d'approche utilisé pour la résolution. On peut distinguer deux grandes classes : les méthodes utilisant des contrôleurs et les méthodes d'optimisation.

1. Les méthodes d'optimisation sont des méthodes globales, qui permettent à l'utilisateur de ne spécifier que le ou les but(s) à atteindre, sous forme de contraintes de positions ou de vitesses. Un système d'optimisation calcule le mouvement en minimisant un critère spécifié par l'utilisateur, tel que l'énergie dépensée au cours de l'animation.

2. Les méthodes utilisant un contrôleur sont basées sur un mécanisme de réponse du système à un état donné. A chaque pas de temps, un module de contrôle analyse l'état du système, et en déduit les forces correspondantes à exercer sur les objets munis de moteurs, permettant de calculer l'état du système au pas de temps suivant.

3.1 Méthodes d'optimisation

Les méthodes d'optimisation sont des méthodes globales, c'est-à-dire qu'elles résolvent le problème en une grande étape de calcul, pour une séquence d'animation complète [BN88, Han93, WK88, Coh92].

Le but de ces travaux est de laisser l'utilisateur définir un scénario pour ses objets, sous forme de contraintes (contraintes aux limites ou à certains instants durant l'animation). Ces contraintes, plus les équations de la dynamique, forment un système d'équations que l'on résout en minimisant un certain critère, généralement un critère d'énergie. Ainsi, le modèle physique de l'objet est utilisé pour interpoler de manière physiquement crédible entre les positions-clés données par l'utilisateur.

Deux approches peuvent être classées dans cette catégorie : les techniques de contrôle optimal [BN88, Han93] et celles d'optimisation sous contraintes [WK88, Coh92].

3.1.1 Le contrôle optimal

Les travaux de Brotman et Netravali [BN88], puis ceux de Hanotaux [Han93], proposent de résoudre le système en le décrivant comme une suite de problèmes avec condition initiale et condition finale, chacun pouvant être représenté par un ensemble d'équations différentielles. La résolution utilise la théorie du contrôle optimal.

Le système se compose de la donnée du vecteur d'état s représentant les degrés de liberté du mobile, d'un vecteur de commande u, et du critère de minimisation K.

Dans le modèle de Brotman et Netravali, les équations différentielles sont linéaires, ce qui a pour conséquence de réduire les possibilités du modèle à des objets isolés. Les contraintes sont alors les positions et vitesses de cet objet aux instants t_0 ... t_n de l'animation, données par l'utilisateur. Le problème est considéré comme une suite de problèmes à deux bornes ($t_{initial} = t_i$, $t_{final} = t_{i+1}$).

Les équations du mouvement sont représentées par un système différentiel linéaire du type :

$$\dot{s}(t) = G(t)s(t) + H(t)u(t) \tag{3.1}$$

où F et G sont déterminées par les équations de la dynamique. Dans le cas décrit dans [BN88], $s(t) = [x(t), \theta(t), \dot{x}(t), \dot{\theta}(t)]$, où x désigne la position et θ l'orientation du solide. La commande u représente les forces et couples du

moteur associé au solide : $u(t) = [0, 0, \frac{1}{m}F, I^{-1}.M]$, où m est la masse du solide, I sa matrice d'inertie, F et M les résultantes des forces et moments.

Étant donnés $s(t_{initial})$ et $s(t_{final})$, il s'agit de déterminer $u(t)$ et $s(t)$ en minimisant un critère du type :

$$K = \int_{t_{initial}}^{t_{final}} [(s^T(t)As(t)) + (u^T(t)Bu(t))]dt \tag{3.2}$$

Le choix de A et B va déterminer la nature du critère de minimisation : $A = 0, B = 1$ correspond à une minimisation de l'énergie dépensée pour réaliser le mouvement;
$A = \begin{bmatrix} 0 & 0 \\ 0 & [10] \end{bmatrix}, B = 1$ correspond à un mouvement plus régulier, puisqu'on attache plus d'importance à la minimisation des vitesses.

Pour résoudre le système les auteurs font appel à la théorie du contrôle optimal [BH75], qui utilise les coefficients de Lagrange pour réécrire la commande sous forme d'une équation de la forme :

$$u(t) = Q(t)s(t) + R(t)s(t_{final}) \tag{3.3}$$

avec $Q(t)$ et $R(t)$ deux matrices dont la résolution dépend de l'intégration numérique depuis t_{final} de plusieurs équations différentielles linéaires complexes. Une fois $Q(t)$ et $R(t)$ connues, on intègre 3.3 de $t_{initial}$ à t_{final} pour trouver les valeurs de u et s.

Cette résolution, en considérant le problème comme une suite de problèmes à deux bornes, provoque une rupture de continuité de la commande à chaque changement d'intervalle. Pour y remédier, les auteurs proposent de reformuler s et u en incluant la commande u dans le vecteur d'état s, et en utilisant $\dot{u}$ au lieu de u. Si cela lisse la commande, en revanche cela augmente énormément l'énergie dépensée au cours du mouvement.

Le travail de Gabriel Hanotaux [Han93] généralise le modèle précédent à des équations non linéaires, ce qui lui permet de l'étendre aux mouvements de rotations et aux solides articulés (il donne l'exemple d'un pendule à deux articulations). Cet apport est permis par une nouvelle approche pour la résolution des équations différentielles considérées. Au lieu de les intégrer numériquement, ce qui devient impossible pour la forme plus générale des équations qu'il obtient, il opère une descente de gradient sur le critère de minimisation K, à partir d'une estimation initiale du contrôle u.

Les équations sont décrites de manière non plus matricielle mais fonctionnelle :

$$\dot{s} = f(s(t), u(t), t) \tag{3.4}$$

$$K = \Phi(s(t_{final})) + \int_{t_{initial}}^{t_{final}} L(s(t), u(t), t)dt \tag{3.5}$$

On utilise de nouveau les multiplicateurs de Lagrange, cette fois pour déterminer une modification de u qui entraîne une diminution de K.

La boucle peut s'écrire comme suit :

- *Initialisation* : Donner une valeur initiale pour u au cours du temps. Intégrer l'équation avec cette valeur pour obtenir les valeurs de s et K.
- Tant que le critère de minimisation n'est pas minimal, déduire de u et s courants une variation δu qui assure $\delta K < 0$.

On considère l'état minimal atteint quand $\delta K = 0$.

Un avantage majeur de cette approche est que, comme toutes les méthodes d'optimisation, elle engendre automatiquement une certaine anticipation dans les mouvements (comme par exemple la prise d'élan pour un saut). D'autre part, la minimisation d'un critère d'énergie permet de sélectionner, parmi les différents mouvements possibles, celui qui répond le mieux au critère spécifié.

Mais ces avantages sont contrebalancés par des limitations assez sérieuses : tout d'abord, le coût. On a vu que le modèle de Brotman et Netravali nécessitait l'intégration numérique, en arrière puis en avant dans le temps, d'un certain nombre d'équations différentielles. Ce modèle est de plus limité par le caractère linéaire des équations qu'il traite, ce qui exclut les objets articulés. L'approche de Hanotaux permet les équations non linéaires et les objets articulés, mais au prix d'un coût encore plus élevé : les intégrations numériques doivent se faire à chaque pas d'un processus itératif. D'autre part, le problème, courant en optimisation, des minima locaux n'est pas résolu : il faut partir d'une solution proche du minimum pour avoir l'assurance de l'atteindre. Enfin, ce modèle ne traite pas les cas de discontinuités, pourtant nécessaires à la simulation des collisions et contacts.

3.1.2 Méthodes d'optimisation sous contraintes

Une deuxième méthode utilisant la minimisation est celle présentée par Witkin et Kass [WK88], et reprise par Cohen [Coh92, LGC94].

Le problème est ici présenté comme un système d'équations discrètes à résoudre, les inconnues étant les paramètres à chaque pas de temps (positions, vitesses, accélérations etc...). L'utilisateur spécifie le mouvement sous forme de contraintes : positions-clés comme dans la méthode précédente, (éventuellement des contacts prolongés), avec l'ajout d'inégalités dans le travail de Cohen. Il fournit également un critère d'optimisation, et les paramètres physiques à associer aux objets. Les équations de la dynamique sont discrétisées et considérées comme des contraintes supplémentaires.

Sur ce système d'équations on utilise les techniques de la théorie de l'optimisation sous contraintes pour obtenir le mouvement qui minimise le critère spécifié. La formulation physique s'appuie sur les équations de Lagrange :

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{s}}\right) - \frac{\partial T}{\partial s} - F = 0 \tag{3.6}$$

où T est une énergie cinétique, F une force généralisée, et s le vecteur de coordonnées généralisées, ce qui permet d'obtenir des variables indépendantes, nécessaires pour la résolution.

Le temps est discrétisé par la méthode des différences finies, ce qui permet d'obtenir les dérivées de s par des équations linéaires :

$$\dot{s}_i = \frac{s_i - s_{i-1}}{h}, \quad \ddot{s}_i = \frac{s_{i+1} - 2s_i + s_{i-1}}{h^2}$$

Les contraintes C sont données sous forme d'équations du type :

$C(s(T), \dot{s}(T), \ddot{s}(T)) = 0$, pour T fixé ou appartenant à un intervalle $[T_0, T_1]$. Après discrétisation, cette équation est remplacée par $C_j(s_{i-1}, s_i, s_{i+1}) = 0$. De même l'équation 3.6 est discrétisée (son expression dépend de l'énergie cinétique de la coordonnée généralisée considérée).
Le critère K à minimiser proposé est de la forme :

$$K = \int_{t_0}^{t_1} \mid F(t) \mid^2 dt \quad \text{soit, après discrétisation :} \quad K = h. \sum_{i=0}^{i=N} \mid F_i \mid^2$$

où h est le pas d'intégration.

Le problème d'optimisation se présente donc comme suit : résoudre le système $C_j(s_{i-1}, s_i, s_{i+1}) = 0$ en minimisant K.

Dans la pratique, un système de calcul symbolique est utilisé pour passer des variables « classiques » aux variables généralisées utilisées dans 3.6. Cependant, l'expression de l'énergie cinétique généralisée doit être spécifiée par l'utilisateur. Le système obtenu (un immense système matriciel, mais composé essentiellement de coefficients nuls) est résolu par programmation séquentielle quadratique [PGW81], rendue possible grâce au calcul de la Hessienne et de la Jacobienne du système. Essentiellement, il s'agit de procéder à une suite d'itérations à partir d'une solution approchée.

Le travail de Cohen [Coh92] porte sur l'amélioration du modèle précédent d'une part par l'introduction de fenêtres « espace-temps », qui permettent de limiter la taille des systèmes à optimiser, d'autre part par l'interpolation par des courbes splines, qui permet de discrétiser le temps en intervalles plus grands. La solution qu'il propose utilise de façon plus importante le traitement symbolique des équations. D'autre part, l'ajout d'inégalités dans les contraintes accroît le champ des possibilités du modèle.

Liu, Gortler et Cohen ont par la suite introduit la hiérarchisation dans ce modèle [LGC94], toujours pour réduire le coût du calcul. En introduisant la possibilité de traiter le problème à différents niveaux de précision, cette approche permet de ne régler l'animation de façon précise que pour les détails importants. Cette hiérarchisation utilise une décomposition en ondelettes pour la description des trajectoires des degrés de liberté, ce qui permet d'obtenir un système mieux conditionné, et une convergence plus rapide de l'algorithme.

Ces modèles présentent à peu près les mêmes avantages que la technique de contrôle optimal décrite plus haut : l'utilisateur spécifie par l'intermédiaire de

positions-clés (une contrainte est une égalité, donc une position-clé) les éléments essentiels de l'animation, et le système s'occupe d'interpoler de manière réaliste. La minimisation d'un critère d'énergie assure une certaine cohérence du résultat, et le principe de l'optimisation permet l'anticipation des mouvements.

Les inconvénients sont également du même ordre : tout d'abord le coût important de la méthode, qui s'accroit de manière dramatique en fonction du temps de la simulation et du nombre de paramètres - bien que l'approche hiérarchique [LGC94] résolve en partie ce problème. Ensuite, la complexité du modèle lui-même, d'autant plus que l'utilisateur doit expliciter lui-même les équations du mouvement en proposant un système de coordonnées généralisées pour son objet, bien que des approches plus récentes résolvent ce problème. De plus, rien n'assure ici non plus qu'on atteint vraiment un minimum global de l'énergie, en particulier lorsque la solution approximative donnée par l'utilisateur est trop éloignée de la solution recherchée. Cette dernière contrainte est en fait très forte, puisqu'elle oblige l'utilisateur à avoir une idée assez précise du résultat à obtenir. Un autre inconvénient lié aux systèmes de minimisation est que les contraintes ne sont jamais exactement vérifiées, et en particulier celles qui dérivent des équations physiques. L'utilisateur devra donc choisir entre une résolution plus précise de ces équations ou accorder la plus grande importance aux positions-clés définies, l'un toujours au détriment de l'autre. Enfin, ce système ne traite pas les collisions, sauf celles qui sont définies explicitement par l'utilisateur comme des contraintes (l'utilisateur spécifie alors les instants extrêmes des contacts, et les positions relatives des objets).

3.2 Utilisation de contrôleurs

A l'opposé des modèles d'optimisation, les modèles utilisant des contrôleurs fonctionnent selon le principe de la dynamique directe. Ils ajoutent au système à simuler une sorte de « boîte noire » permettant de calculer une action à exercer en fonction de l'état courant du système, avant de procéder à l'intégration des équations physiques à ce pas de temps.

Suivant l'application recherchée, on peut distinguer un certain nombre de modèles proposant des approches plus ou moins spécialisées, plus ou moins paramétrables, plus ou moins complexes et plus ou moins automatiques.

Étudions tout d'abord le type d'approche correspondant aux contrôleurs spécialisés.

3.2.1 Contrôleurs spécialisés

Le but de cette section n'est en aucun cas d'établir une liste exhaustive de ce type de modèle, ce qui paraît démesuré, mais plutôt de décrire à travers quelques exemples le type de problème posé et résolu par ce genre d'approches.

Ces modèles traitent souvent l'animation de créatures articulées, pour des mouvements aussi divers que la marche, la course ou le plongeon. Nous présentons ici un tel modèle, ainsi qu'une application plus originale proposant l'étude

de l'interaction de véhicules avec leur environnement.

Animation de véhicules terrestres

Jimenez et Luciani [JLR93, JL93] présentent une utilisation de contrôleurs spécialisés pour l'animation de véhicules terrestres. Il s'agit d'étudier le comportement de véhicules automoteurs ou tractés sur un modèle de sol plus précis et plus réaliste que dans la plupart des approches réalisées jusqu'à présent.

Dans le cas de véhicules tractés, le contrôleur est modélisé par un ressort de rappel reliant le véhicule à une cible guidée par l'opérateur (dans le plan du sol) grâce à un système à retour d'effort. Pour les véhicules automoteurs, les roues du véhicule sont munies de couples d'entraînement, commandés par une vitesse angulaire ou linéaire. Dans le cas de véhicules tri-dimensionnels, un régulateur permet le couplage des deux roues pour respecter la consigne de vitesse.

Une fois le type de contrôle défini, Jimenez et Luciani se penchent sur les interactions du véhicule avec le terrain, modélisé de façon à représenter les phénomènes tels que la friction, le glissement, l'adhérence et le dérapage. Pour cela, les auteurs établissent un « modèle minimal », c'est-à-dire le modèle le plus simple rendant compte du phénomène étudié. Ainsi, aussi bien les interactions dans le sol que dans le véhicule et les interactions sol-véhicule sont modélisées par un ensemble d'interactions entre masses ponctuelles. Le système est jugé satisfaisant quand les interactions générées rendent compte de l'effet modélisé.

Cette approche montre une première utilisation de contrôleurs, dans le cas de véhicules stables évoluant sur un sol accidenté. Le contrôle proposé à l'utilisateur permet de guider interactivement le véhicule, tout en recevant une information sur la difficulté du mouvement. Cette approche est cependant fortement liée à la nature du problème : ici, le véhicule repose toujours sur le sol, et le guidage par opérateur se fait dans le plan du sol. De plus, la stabilité du véhicule permet d'éviter le problème de l'équilibre, qui constitue un des points principaux des approches étudiant les mouvements en trois dimensions, comme le modèle présenté ci-dessous.

Course de créatures articulées

Raibert et Hodgins [RH91] proposent un modèle de course de créatures à une, deux ou quatre pattes. Ce modèle fournit « clé en main » un acteur de synthèse qui, étant données sa configuration (paramétrable dans une certaine mesure), la vitesse et la démarche souhaitée, est capable d'entreprendre un mouvement en ligne droite sur un sol plan.

L'utilisation du contrôle a ici pour but de permettre à l'utilisateur d'éviter la spécification des détails du mouvement à chaque liaison du solide, et à chaque étape du mouvement.

Les deux principaux problèmes rencontrés dans ce type de contrôleur sont d'une part la difficulté à relier une attitude générale (du type trotter pendant deux minutes puis s'arrêter) aux paramètres générateurs du mouvement (quelles

forces appliquer aux muscles des cuisses à chaque instant); d'autre part la gestion de tous les paramètres dynamiques qui entrent en jeu : outre les forces exercées à chaque instant par les muscles, les échanges d'énergie entre le système et le milieu extérieur sont déterminants dans la création du mouvement. Ainsi, en course, le contrôleur doit anticiper le mouvement pour tenir compte de l'énergie cinétique.

Le principe développé dans cette approche est de décomposer le problème du contrôle du saut en plusieurs sous-problèmes résolus indépendamment. Les trois sous-problèmes définis ici sont le changement d'appui, la régulation de la vitesse et le maintien de l'équilibre vertical. Chacun de ces sous-problèmes est résolu par l'utilisation d'un contrôleur affecté à une articulation différente de la créature articulée. Différentes démarches sont proposées à l'utilisateur : galop et course pour un bipède, trot, galop, course et bonds pour un quadrupède, et saut pour un kangourou, dont les pattes sont considérées comme solidaires, et donc modélisées par une seule patte.

Changement d'appui : L'idée développée ici est que les animaux utilisent des structures élastiques dans les membres pour minimiser l'énergie utilisée lors de leur mouvement. Ainsi, lors de la prise d'appui, de 20% à 40% de l'énergie est réutilisée sans recours aux muscles (par rebond sur le sol). Pour le kangourou le lien ressort est la cheville, et pour le bipède et le quadrupède c'est le joint télescopique. Le muscle est représenté par l'équation :

$$f = k(x - x_r) + b\dot{x} \tag{3.7}$$

où k est la constante de raideur et b la constante de frottement. Le contrôle de x_r est utilisé pour initier, moduler et stopper l'oscillation.

Pour le saut vertical pour une jambe sans masse, l'altitude du saut est prédite par la formule :

$$h_{prévu} = (EP_{tension} + EP_{élévation} + EC)/mg \tag{3.8}$$

Le contrôle se fait par ajout ou suppression d'énergie.

Contrôle de la vitesse : Les systèmes munis de jambes agissent comme des pendules inversés. Ils basculent ou accélèrent en fonction de la position choisie pour le prochain point d'appui. On détermine le « point neutre », qui à une vitesse donnée est le point d'appui tel que le mouvement soit symétrique par rapport à ce point d'appui. Si le pied se pose avant, il va accélérer, et s'il se pose après il va ralentir. La détermination du point d'appui, avant ou après le point neutre, permet de régler la vitesse du système, selon la formule :

$$x_{fh,d} = \frac{\dot{x}T_s}{2} + k_{\dot{x}}(\dot{x} - \dot{x_d}) \tag{3.9}$$

où $x_{fh,d}$ est la distance du point d'appui à la projection du centre de gravité, $\dot{x}$ la vitesse, $\dot{x_d}$ la vitesse souhaitée, T_s la durée prévue de la prochaine période,

et $k_{\dot{x}}$ est un gain. Le premier terme est l'estimation de la position du « point neutre », et le second terme est la correction de la vitesse, ou une accélération désirée. Une fois trouvé $x_{fh,d}$, le système en déduit la position des différentes articulations de la jambe.

Maintien de l'équilibre vertical : Le système maintient l'équilibre de la créature par l'orientation du tronc, réglée pour les bipèdes et quadrupèdes par l'articulation de la hanche, et pour le kangourou par le genou. Le couple de contrôle de posture s'exprime par un asservissement linéaire :

$$\tau = -k_p(\phi - \phi_d) - k_v\dot{\phi}$$

où τ est le couple appliqué à l'articulation, ϕ l'angle du corps, ϕ_d l'angle souhaité, $\dot{\phi}$ la vitesse de rotation du corps et k_p et k_v sont des gains.

Tous les processus de contrôle sont considérés comme découplés, et les interactions effectives considérées comme des perturbations. Un automate à états finis est utilisé pour synchroniser les actions de contrôle.

Le modèle présenté ici permet d'obtenir des démarches pour des robots sauteurs qui sont assez proches de robots réels utilisés par les auteurs. Cette approche est prometteuse dans la mesure où elle montre une réalisation de synthèse relativement fidèle à une créature réelle. Malheureusement l'approche semble difficilement généralisable automatiquement à d'autres types de démarches. En effet, pour les modes de course et de saut qui sont simulés, l'acteur de synthèse n'a qu'un point d'appui au sol à un instant donné (pour les quadrupèdes, les deux pattes en contact simultané sont traitées comme une unique « patte virtuelle »), ce qui simplifie le traitement du changement d'appui.

3.2.2 Création interactive de contrôleurs

Plutôt que d'offrir un modèle « clé en main » de contrôleur comme on l'a vu dans les approches précédentes, Wilhelms et Skinner [WS89] proposent de laisser une plus grande liberté à l'utilisateur en lui fournissant un modèle beaucoup plus souple, à charge pour l'utilisateur de construire le contrôleur souhaité. Une interface permet de définir ce contrôleur, en combinant un ensemble de fonctions simples et intuitives.

Le contrôleur se compose d'un ensemble de capteurs en entrée, d'effecteurs en sortie, et d'une fonction de transfert. Les capteurs sont des fonctions qui produisent un signal en fonction de l'état de tout ou partie du système. Dans le cas présent le signal est normalisé sous la forme d'un vecteur direction et d'un scalaire donnant son intensité. La fonction de transfert convertit le signal d'entrée en un signal de sortie, utilisé par les effecteurs pour produire une action (ici une force à exercer sur l'objet contrôlé). Les auteurs proposent plusieurs types de capteurs : premièrement, les capteurs de *distance*, ou au contraire de *proximité*, qui produisent un signal (positif ou négatif) en fonction de la distance de l'objet contrôlé aux autres objets. Ensuite, les capteurs de *qualité*, qui

réagissent à certaines caractéristiques de l'environnement, comme par exemple la couleur des objets. Les effecteurs produisent une force à exercer sur l'objet, dont la direction et le point d'application dépendent de la position et de l'orientation de l'effecteur sur l'objet. La fonction de transfert est composée de *connexions* et de *nœuds*. Les connexions transmettent simplement la sortie des senseurs aux effecteurs, alors que les nœuds peuvent additionner les signaux, ou les amplifier. Les auteurs proposent un certain nombre de nœuds permettant de se rapprocher, de s'éloigner, ou d'éviter un objet donné, ainsi qu'un nœud d'arbitrage, permettant de pondérer les actions de différents nœuds.

Enfin, la construction des contrôleurs se fait au moyen d'une fenêtre de construction, qui permet à l'utilisateur de connecter les différentes composantes décrites ci-dessus pour obtenir le comportement souhaité pour l'objet animé.

Le modèle proposé par Wilhelms et Skinner permet, à l'aide d'une interface rendue relativement intuitive par le caractère explicite des fonctions définies, de modéliser un certain nombre de comportements plus ou moins complexes. Cependant, le modèle de simulation est assez simplifié (les objets animés sont rigides et non articulés), et la gestion des interactions entre les différents nœuds utilisés reste à la charge de l'utilisateur, ce qui peut être prohibitif pour construire des interactions vraiment complexes.

3.2.3 Génération automatique de contrôleurs

Contrairement aux modèles présentés jusqu'à présent ou l'aide de l'utilisateur ou du concepteur était requise pour définir un modèle, les systèmes de génération automatique de contrôleurs permettent d'engendrer automatiquement un ensemble de modes de locomotion pour une créature articulée quelconque. Si l'utilisateur n'a aucun contrôle a priori sur la démarche, ni même sur le type de démarche, en revanche il n'a qu'un minimum de données à fournir au système pour produire un mouvement : la structure des créatures et les paramètres physiques qui lui sont associés.

Le principe de base de ce type de méthodes est de décrire un objet articulé quelconque et de laisser le système trouver une démarche adéquate pour le faire avancer. Pour cela, on décrit un contrôleur générique et on effectue une série de tests en faisant varier les paramètres du contrôleur. Les solutions conservées sont celles qui répondent le mieux à un critère défini par l'utilisateur, comme « parcourir la plus grande distance en un temps donné ».

Plusieurs articles [NM93, vdPF93, vdPKF94b, vdPKF94a, NAC94, NAHR95] illustrent cette approche. Les méthodes diffèrent d'abord par le contrôleur générique choisi, et également par la façon de sélectionner les modèles retenus.

Algorithme génétique

Le principe proposé par Ngo et Marks [NM93] est de définir pour un objet articulé ce qu'il doit faire, en l'occurrence avancer, sans lui dire comment le faire. L'animateur définit la structure physique de la créature, ses moteurs, et les critères d'évaluation du mouvement (ici, la distance parcourue en un

temps donné). A partir de ces données, l'ordinateur doit trouver un mode de déplacement, physiquement réaliste, et qui optimise le critère d'évaluation.

Le modèle est composé d'un module de contrôle (nommé ici « stimulus et réponse »), et d'un module dynamique restreint : l'objet articulé est considéré comme un objet unique et déformable, la gestion des déformations étant purement cinématique. Le contrôleur générique est lié à un ensemble de fonctions stimulus et réponse. La valeur des paramètres de ces fonctions est obtenue par un algorithme génétique, partant d'une population échantillon réduite. Le modèle présenté est en dimension deux.

Le module « stimulus et réponse » est constitué d'un ensemble de fonctions scalaires dépendant des variables d'état du système, et d'un ensemble de réponses qui sont des prescriptions données aux valeurs angulaires des liens de la créature.

Une réponse se présente sous la forme d'un ensemble $(\theta_i^0 \quad 1 \leq i \leq n)$ de valeurs angulaires cibles pour chacune des articulations de la créature et d'une constante de temps τ. L'angle θ_i d'une articulation est calculé à partir de cet ensemble par une équation du type :

$$\tau^2 \ddot{\theta}_i + 2\tau\dot{\theta}_i + (\theta_i - \theta_i^0) = 0 \tag{3.10}$$

A chacun de ces ensembles est associé une fonction stimulus, définie par :

$$W.\left(1 - \max_{j=1}^{V}[\lambda_j(v_j - v_j^0)]^2\right) \qquad \text{avec} \quad W = \sum_{j=1}^{V} \log\left(\frac{\lambda_j}{\lambda_j^{min}}\right) \tag{3.11}$$

où les v_j sont des variables sensitives, v_j^0 et λ_j des paramètres à déterminer par le module de recherche, W un poids, et λ_j^{min} la valeur minimale autorisée de λ_j. Quand la fonction stimulus est positive, on passe à la fonction réponse θ_i suivante.

Les variables sensitives sont de quatre types : angles des articulations, force exercée par les extrémités des liens sur le sol, vitesse verticale du centre de masse, et position verticale du centre de masse. Résoudre le système signifie trouver un ensemble de valeurs des paramètres $(\tau, \theta_i^0, v_{ij}^0, \lambda_{ij})$ (qui constitue un « génôme ») tel que le mouvement obtenu par simulation vérifie le critère de choix. Pour obtenir ces valeurs, on applique un algorithme génétique sur une population initiale choisie au hasard. Une fois cette population évaluée, on va croiser chacun de ces génômes avec le meilleur de ses voisins. La localité des croisements permet d'éviter l'uniformisation de la population. Le croisement consiste à échanger un certain nombre de caractéristiques (ou gènes) entre les deux génômes. Enfin, on applique une mutation au génôme ainsi obtenu, en bruitant l'un des stimulus et en choisissant un autre aléatoirement.

On répète ces opérations jusqu'à obtenir une colonie dominante.

L'approche proposée par Ngo et Marks permet de trouver automatiquement une démarche pour des créatures articulées, sans que l'utilisateur ne présage en rien des possibilités du modèle décrit. Grâce à l'algorithme génétique, la population de départ peut être assez réduite, et conduire pourtant à des modes

de locomotion complexes. Cependant, l'obtention de telles solutions demande beaucoup d'étapes de croisements et mutations, ce qui rend l'algorithme coûteux. Par ailleurs, l'approche bi-dimensionnelle du problème, et la simplicité du critère d'évaluation, ne permettent d'engendrer que des mouvements plans, avançant en ligne droite.

Approche aléatoire

L'approche de van de Panne et Fiume [vdPF93] part du même principe que le modèle décrit ci-dessus : plutôt que d'essayer de modéliser une démarche correspondant à une idée préconçue de l'animateur, celui-ci va simplement décrire la créature à animer (masse, longueur des liens, articulations, position des muscles), et laisser entièrement au système le choix de la démarche.

Le type de contrôle est ici la production de couples aux articulations. La structure du contrôleur générique est un réseau de capteurs binaires et d'effecteurs reliés par un système d'équations non linéaires pondérées. Le choix des paramètres se fait par évaluation d'un grand nombre de valeurs de ces poids, puis par un réglage plus fin des meilleures solutions obtenues. Le critère peut être la distance parcourue par la créature, ou des fonctions plus complexes, qui seront détaillées plus loin. Le modèle proposé est en dimension 2 (plan vertical).

Une créature est composée d'un ensemble de membres liés par des articulations. Sur cet ensemble on place un certain nombre de capteurs et d'effecteurs. Les capteurs sont de quatre types : capteurs de contact, d'angle, de vision et de longueur. Le capteur de contact est activé (réponse 1) quand le point sur lequel il est placé est en contact avec le sol. Le capteur d'angle est positionné sur une articulation. Il est activé si l'angle formé par les deux liens est compris dans un certain intervalle. Le capteur de vision est activé si un certain point, qu'il suit, est dans son cône de visibilité. Enfin, le capteur de longueur répond si une certaine longueur est comprise entre un minimum et un maximum donnés.

Les moteurs sont de deux types : linéaires et angulaires. Les moteurs angulaires sont positionnés aux articulations, alors que les moteurs linéaires sont appliqués entre deux points. Les forces et couples sont donnés respectivement par les formules :

$$F = k_s(x_d - x) - k_d\dot{x} \tag{3.12}$$

$$C = k_s(\theta_d - \theta) - k_d\dot{\theta} \tag{3.13}$$

Les constantes k_s et k_d sont données par l'utilisateur à la création du modèle.

Le réseau capteurs-moteurs est constitué de nœuds d'entrée, de nœuds internes (ou cachés), et de nœuds sortie. Ces nœuds sont liés entre eux par des liaisons unidirectionnelles pondérées (choisies entières dans $[-2, 2]$). Les nœuds d'entrée sont connectés à tous les nœuds cachés et de sortie, et les nœuds cachés et de sortie sont entièrement interconnectés (voir figure 3.1).

Les nœuds d'entrée prennent leurs valeurs de celles des capteurs, les nœuds internes et de sortie somment les poids transmis, et répondent 1 si cette somme

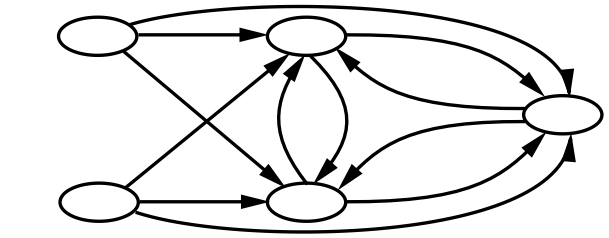


FIG. 3.1 - *Exemple de réseau capteurs-moteurs*

est positive. Les valeurs des nœuds de sortie sont ajoutées dans une variable d'état du moteur : $k_1.dt$ si le nœud répond 0, $k_2.dt$ sinon. Cette variable d'état est maintenue entre 0 et 1. Quand elle atteint 1 le moteur est activé, et quand elle atteint 0 le moteur est éteint. Les constantes k_1 et k_2 donnent un temps de réaction du système, et sont choisies en fonction de la durée voulue du cycle de déplacement.

Le choix qui reste à faire maintenant concerne les valeurs des poids affectés à chaque liaison du réseau. Ce choix est fait par génération aléatoire de nombreux essais, évalués par un critère K, variant suivant le type d'application.

$$K = \int_0^{t_{final}} (\dot{x} + kz^2)dt$$

Dans cette formule K mesure la vitesse horizontale $\dot{x}$ et la hauteur z (avec un coefficient de proportionnalité k), ce qui permet d'obtenir un mouvement avec des bonds importants.

Une autre fonction d'évaluation, permet à la créature de suivre un objet en mouvement :

$$K = \int_0^{t_{final}} (v.X)dt$$

où X est un vecteur pointant vers l'objectif et v la vitesse de la créature. K mesure alors l'écart par rapport à la direction et la distance à l'objet suivi. Les mouvements retenus sont ceux qui ont le plus fort K.

Enfin, une amélioration des solutions retenues est faite, non pas en modifiant les poids des liaisons, mais cette fois en réglant les paramètres choisis précédemment : les constantes k_1 et k_2 du réseau, les bornes données aux capteurs et aux moteurs, et les constantes données aux forces et couples moteurs.

Dans la suite de ses travaux [vdPKF94b, vdPKF94a], van de Panne a simplifié le modèle pour réduire l'espace de recherche pour les contrôleurs. Il a remplacé le réseau capteurs-moteurs par un graphe de positions, qui n'utilise plus de capteurs mais change d'état au bout d'un temps donné, quel que soit l'état du système (voir la figure 3.2).

Un graphe de positions est simplement un ensemble de positions-clés pour la créature articulée dans le repère propre de celle-ci. L'utilisateur spécifie le nombre de positions-clés P_i et le temps T passé entre chacune, et pour chaque position P_i les angles internes θ_{ik}. Le système optimise les positions elles-mêmes

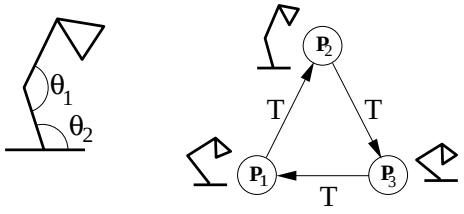


FIG. 3.2 - *Graphe de positions.*
A gauche, la créature articulée d'angles internes θ_1 et θ_2. A droite, le graphe à trois états P_1, P_2 et P_3.

pour obtenir celles qui remplissent le mieux le critère de sélection, ici simplement la plus grande distance parcourue dans un temps donné. Dans [vdPKF94b], le modèle se limite aux graphes cycliques, alors que dans [vdPKF94a] les auteurs étudient les graphes acycliques, et la paramétrisation des démarches par composition de deux graphes de mouvements déjà existants (par exemple l'un de marche et l'autre de course rapide), avec ou sans nouvelle étape d'optimisation. Parmi les mouvements apériodiques étudiés, les auteurs montrent des séquences de redressement après une chute, permettant à la créature d'évoluer de façon autonome dans un environnement plus complexe.

Approche par ondelettes

L'approche de Nougaret, Arnaldi et Crozot [NAC94] est différente dans ses buts : plutôt que d'optimiser un contrôleur pour obtenir un mouvement à partir de contrôleurs choisis au hasard, les auteurs proposent d'utiliser un contrôleur déjà efficace pour avancer le long d'une ligne droite, issu de l'observation de mouvements réels, et d'optimiser les variations par rapport à ce mouvement de référence pour vérifier un but donné par l'utilisateur.

Le contrôleur utilisé combine l'utilisation de la cinématique et de la dynamique. La cinématique est utilisée pour donner la configuration interne de l'objet, c'est-à-dire les positions relatives des liens au cours du temps. Ces positions sont créées à partir de données biologiques ou zoologiques. La dynamique est utilisée pour calculer les interactions de l'animal avec le milieu extérieur, c'est-à-dire le calcul des positions, vitesses et orientations de l'animal au cours du temps.

Les trajectoires des articulations internes de l'objet sont exprimées sous forme d'ondelettes, afin de donner une formulation compacte du mouvement, et de fournir un jeu de paramètres restreints pour l'optimisation. La base d'ondelettes choisie par les auteurs est un ensemble de fonctions obtenues par translation et dilatation d'une fonction de base, ϕ : $\phi_i(t) = \phi(d_i(t - \tau_i))$, avec $\phi(t) = (t^2 - 1)e^{\frac{1}{2}t^2}$. d_i représente la dilatation, et τ_i la translation. Toute fonction f peut s'approximer dans une base finie ϕ_i, $1 \leq i \leq N$ avec une précision ε :

$$f(t) = \sum_{i=0}^{i=N} w_i \phi_i(t) + \varepsilon(t)$$

Les w_i sont les poids de f dans la base ϕ_i. Chaque degré de liberté θ_k de

l'animal est exprimé dans cette base d'ondelettes, avec des poids $w_{i,k}$ fixés pour le mouvement de référence.

Le processus d'optimisation consiste à perturber les coefficients représentatifs de θ_k dans la base : $w_{i,k}, d_{i,k}$ et $\tau_{i,k}$, pour minimiser un critère de coût, dépendant de la proximité des objectifs à atteindre. Les coefficients sont contraints à rester dans une limite raisonnable autour de leur valeur initiale, pour préserver l'aspect naturel du mouvement. L'optimisation est réalisée par programmation séquentielle quadratique.

Par la suite Nougaret [NAHR95] propose de précalculer les optimisations et de les réutiliser en utilisant un réseau de neurones, afin de stocker les données, et de permettre d'interpoler entre les échantillons précalculés de l'animation. La grille d'échantillonnage est choisie aussi lâche que possible, et le réseau de neurones sert à émuler le processus d'optimisation.

Le modèle décrit ici propose d'inclure les connaissances acquises dans d'autres domaines dans un modèle physique, permettant ainsi de gérer automatiquement les interactions avec le milieu extérieur, tout en fournissant un point de départ basé sur l'observation de modes de locomotion réels pour modéliser le mouvement d'animaux. L'ajout du système neuronal permet, de plus, d'engendrer un certain nombre de solutions échantillons, pour retrouver ensuite par interpolation la solution la plus proche du problème considéré.

Cependant, le modèle ne permet pas à l'heure actuelle de gérer les systèmes discontinus, ce qui empêche d'une part de prendre en compte les collisions, d'autre part de modéliser des créatures en contact avec le sol.

Discussion

Nous avons vu dans cette partie un certain nombre de modèles, allant des plus simples aux plus complexes, qui, s'ils utilisent tous les lois de la physique à un degré plus ou moins élevé, présentent une grande variété de résultats, aussi bien par la fidélité aux lois de la physique que le degré d'automatisme, la quantité de travail laissé à l'utilisateur, ou encore le type de résolution utilisé.

Le problème que nous nous sommes proposés de résoudre dans cette thèse est de rendre les modèles générateurs utilisables par un utilisateur « naïf » en lui fournissant un moyen de spécifier de façon relativement précise le scénario qu'il souhaite voir se réaliser, tout en préservant les qualités principales des modèles physiques, c'est-à-dire l'automatisme de la génération du mouvement et l'adéquation aux lois de la dynamique, en particulier pour la gestion des collisions. Voyons, pour les modèles proposés jusqu'à présent, dans quelle mesure ils répondent à notre attente.

Les modèles « mixtes » présentés en début de cet état de l'art permettent la spécification de scénario, dans la mesure où ils s'appuient principalement sur la définition de positions-clés, qui est actuellement l'outil de base dans ce domaine. En revanche, l'utilisation limitée qu'ils font de la dynamique, en particulier l'absence de moteur d'intégration, ne permet pas de traiter de façon physique les interactions entre objets. Si la première approche, limitée rappelons-le à des objets isolés, permet certains types d'interactions basées sur des lois comportementales, la deuxième se limite dans ce cadre à la définition par l'utilisateur des contacts avec le sol.

Les modèles d'animation par contraintes et de dynamique inverse offrent un compromis entre les deux points recherchés, en permettant à l'utilisateur, ou bien de créer des animations de manière purement physique, mais sans contrôle, ou bien de définir pour certains degrés de liberté des trajectoires purement cinématiques, laissant alors de coté la possibilité de les traiter de façon physique, en particulier dans le cas de collisions ou contacts.

Les techniques d'optimisation permettent à l'utilisateur d'engendrer une animation suivant un scénario précis, en spécifiant un ensemble de positions-clés. Les modèles physiques sont utilisés pour trouver une interpolation correcte entre ces positions, grâce à un processus de minimisation. Ainsi, ces approches permettent un certain réalisme tout en conservant à l'utilisateur la possibilité de définir un scénario. Cependant, ces techniques sont coûteuses, au détriment de l'interaction avec l'utilisateur, et demandent un travail technique supplémentaire à l'utilisateur pour mettre au point le modèle. De plus, ce type de modèles

ne gère pas non plus le traitement des collisions.

Les méthodes utilisant des contrôleurs, en revanche, présentent l'avantage d'être compatibles avec la simulation directe, ce qui permet ce traitement. Les contrôleurs spécialisés permettent d'engendrer des mouvements complexes dans le cadre qui leur est consacré, mais sont en général trop spécifiques pour permettre de déduire de leur fonctionnement une quelconque généralisation. Les contrôleurs engendrés automatiquement permettent d'animer des créatures arbitrairement compliquées, avec cependant le handicap d'un coût éventuellement important, surtout en dimension trois), mais leur but est plutôt de créer des créatures autonomes que de suivre un scénario précis.

En définitive, l'ensemble des modèles décrits ici présentent par rapport au but que nous nous sommes fixés ou bien un manque de contrôle au niveau de la définition de la trajectoire, ou bien une insuffisance dans le modèle physique utilisé, tout particulièrement concernant la détection et le traitement des collisions, qui rappelons-le sont nécessaires dans des situations aussi simples que le contact d'une créature avec le sol. C'est pourquoi nous avons développé un modèle tentant de répondre de façon satisfaisante à toutes ces conditions. Ce modèle est présenté dans la partie II de ce document. Signalons qu'il utilise la notion de contrôleur déjà présentée dans cette partie, comme module de contrôle associé à un module de simulation physique.

Deuxième partie

Combiner simulation et contrôle de trajectoire

Introduction

Comme il a été montré dans la partie I, il est difficile de concilier les propriétés des modèles physiques telles que réalisme et automatisme avec une interface facilement utilisable par un non-spécialiste : plus le contrôle offert à l'utilisateur est fin, plus l'interface qui lui est proposée est difficile à manier. En particulier, peu de modèles présentés jusqu'à présent proposent à la fois la détection automatique des collisions (avec les corrections éventuelles qui devraient en résulter) et le suivi d'un scénario évolué. Le but de cette partie est de fournir à l'utilisateur un outil qui lui permette de concilier ces deux qualités.

Le chapitre 4 décrit un modèle de contrôle de trajectoire, permettant de spécifier le mouvement par des positions-clés, mais en conservant la capacité des modèles dynamiques à réagir physiquement aux évènements externes survenant au cours du mouvement.

Dans le chapitre 5, nous montrons un système de définition de scénarios, construit comme une couche de contrôle supplémentaire par-dessus ce modèle, et permettant de définir des événements de synchronisation au cours de l'animation, afin de faciliter la création d'animations plus complexes.

Chapitre 4

Contrôle de trajectoire d'un objet

4.1 Présentation rapide du modèle

Le but du modèle développé ici est de permettre au graphiste de définir un ensemble de positions-clés pour guider le mouvement, sans avoir à traiter les problèmes tels que le réalisme du mouvement ou l'interpénétration des objets. Nous voulons laisser le système corriger ce mouvement grossier durant une simulation directe, réalisée à une vitesse interactive.

Pour y parvenir, les objets sont munis d'effecteurs de puissance limitée, représentant des moteurs ou des muscles. A chaque pas de la simulation les forces et les couples produits par les effecteurs sont calculés par un module qui dérive du contrôle Proportionnel-Dérivé (PD) utilisant la différence entre une position-cible et la position courante de l'objet comme entrée principale. Les cibles sont placées sur des courbes d'interpolation définies d'après les positions-clés données par le graphiste (voir Figure 4.1). Afin de permettre à l'utilisateur de ne spécifier qu'une vitesse moyenne, le mouvement des cibles est étroitement lié au mouvement de l'objet associé. La vitesse moyenne sera suivie dans le cas d'une trajectoire en ligne droite. Par contre, en cas de méandres dans la trajectoire, ou de perturbations dues à des collisions ou des contacts, l'objet sera ralenti le temps de répondre à ces perturbations, puis retrouvera sa vitesse moyenne. On peut retrouver les résultats décrits dans ce chapitre dans [LGG95], paru dans la revue « The Journal of Visualization and Computer Animation ».

La section 4.2 explique les principes de base de cette approche en traitant le cas d'un solide isolé guidé par des positions et orientations-clés définies par le graphiste.

La section 4.3 traite de situations plus complexes. Les objets contrôlés peuvent être connectés à d'autres objets par plusieurs types de charnières. Chaque objet peut être soumis à des forces externes, qui peuvent ou non durer dans le temps.

La section 4.4 présente l'étude des paramètres utilisés et montre comment choisir ceux-ci en fonction de l'effet d'animation voulu.

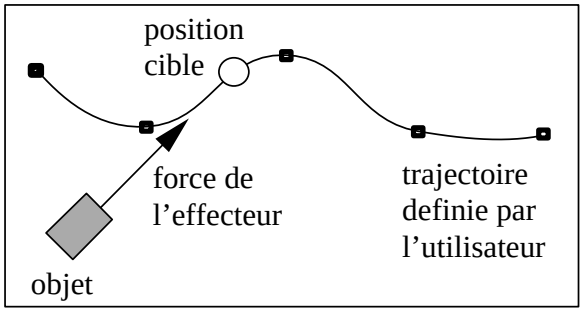


FIG. 4.1 - *Objet muni d'un effecteur guidé le long d'une trajectoire*

4.2 Contrôle de trajectoire d'un objet isolé

Cette section propose l'étude du cas d'un objet simple isolé, guidé par un ensemble de positions-clés définies par l'utilisateur. Les cas plus compliqués sont traités dans les sections ultérieures. Suivant le type de contrôle désiré, l'objet est muni d'un effecteur en translation, en rotation, ou les deux. Chaque effecteur est caractérisé par une puissance maximum : la force ou le couple produit est donc borné.

La méthode utilisée pour le calcul du mouvement ressemble au jeu du chat et de la souris : à chaque pas de temps, l'objet cherche à se rapprocher d'une position-cible située sur la trajectoire définie par l'utilisateur. Mais cette position-cible se déplace en même temps, et sa vitesse dépend du mouvement de l'objet qu'elle guide. Cette approche permet de conserver l'objet à proximité de la trajectoire, quelles que soient les actions externes exercées.

Les deux prochains paragraphes décrivent les points-clés de la méthode : comment calculer l'action des effecteurs à partir de la position-cible, et comment modifier cette position à chaque pas de temps.

4.2.1 Calcul des forces et des couples

Deux types d'effecteurs sont disponibles dans le système. Les effecteurs de translation produisent une force de translation $\vec{F}$, limitée en puissance, et appliquée au centre de masse de l'objet. Cette force vérifie : $||\vec{F}|| < F_{\max}$. Les effecteurs de rotation produisent un couple $\vec{\mathcal{C}}$ vérifiant : $||\vec{\mathcal{C}}|| < \mathcal{C}_{\max}$. Le principe de l'action des effecteurs est de produire des forces et couples pour rapprocher l'objet d'une position-cible $\overrightarrow{x_{\text{cible}}}$ ou orientation-cible R_{cible} associée.

Supposons qu'il n'y ait pas d'action externe. D'après le schéma d'intégration de Newton (détaillée au début de la partie I), la force à exercer sur l'objet pour atteindre la position $\vec{x_{\text{cible}}}$ en un pas de temps δt est :

$$\vec{F}(t) = 2m\,\frac{\overrightarrow{x_{\text{cible}}} - \vec{x}(t)}{\delta t^2} - 2m\,\frac{\vec{v}(t)}{\delta t} \tag{4.1}$$

Malheureusement, cette force est inadaptée au cas présenté, pour plusieurs raisons. Tout d'abord, la vitesse nécessaire pour atteindre la cible en un pas de temps propulserait l'objet loin devant celle-ci au cours des pas de temps suivants. De plus, supprimer entièrement les effets du précédent vecteur-vitesse

supprimerait l'inertie, tant que la puissance maximum n'est pas atteinte. Enfin, la plupart des systèmes de simulation utilisent un pas de temps adaptatif, pour réguler la simulation quand des problèmes comme la forte interpénétration entre objets en collision surviennent. Utiliser le pas de temps courant pour calculer l'action de l'effecteur causerait des accélérations ou décélérations non souhaitées à chaque variation de δt.

L'approche présentée ici est différente. Nous calculons une force telle que l'objet parcoure une fraction donnée α de sa distance à la cible (notée d_{base}, voir figure 4.2) dans un temps de relaxation Δt fixé (pour éviter tout problème lié au pas de temps adaptatif). Un paramètre β est utilisé pour prendre en compte la vitesse de l'objet, sans supprimer entièrement son effet.

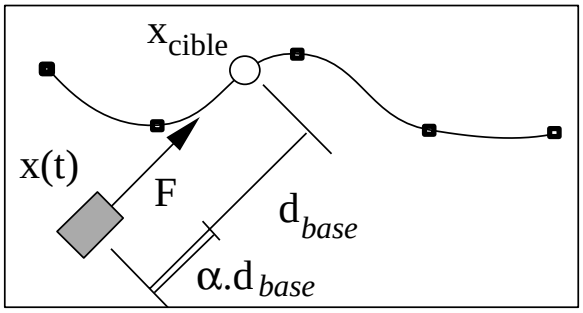


FIG. 4.2 - *Définition de α et d_{base}*

Cela conduit à la formule :

$$\vec{F}(t) = 2m\ \frac{\alpha\left(\overrightarrow{x_{\text{cible}}} - \vec{x}(t)\right)}{\Delta t^2} - 2m\frac{\beta\vec{v}(t)}{\Delta t} \tag{4.2}$$

- Si $\beta = 0$, la vitesse de l'objet est ignorée durant le calcul des forces.
- $\beta = 1$ signifie qu'on corrige complètement l'effet de la vitesse de l'objet, supprimant ainsi l'effet de l'inertie.
- Entre les deux, on peut régler β pour obtenir une trajectoire plus ou moins amortie. La valeur de β affecte donc la vitesse de convergence vers la position-cible.

Une analogie avec la théorie du contrôle, qui sera plus développée dans la section 4.5, donne une valeur théorique pour la valeur critique d'amortissement de β :

$$\beta_{\text{critique}} = \sqrt{2\alpha}$$

On peut démontrer ce résultat dans le cas simple d'une cible immobile, en résolvant l'équation différentielle associée :

$$\ddot{x} \ + \ \frac{2\beta}{\Delta t}\dot{x} \ + \ \frac{2\alpha}{\Delta t^2}x \ = \ 0$$

Le discriminant de cette équation vaut : $\Delta = \frac{1}{\Delta t^2}\left(\beta^2 - 2\alpha\right)$. La résolution de l'équation donne donc pour $\beta > \beta_{\text{critique}}$ une exponentielle décroissante, alors

que pour $\beta < \beta_{\text{critique}}$ un terme en cosinus apparaît, provoquant des oscillations autour de la position d'équilibre. Durant l'expérimentation, il a été vérifié que cette valeur heuristique pour β_{critique} est bonne quel que soit le comportement de la cible, pour des pas de temps suffisamment petits.

Dans la pratique, le graphiste pourrait vouloir, plutôt que de donner directement la valeur de β, régler directement l'atténuation du système. Le graphiste donne ainsi un « coefficient d'élasticité » $B \in [0, 1/\beta_{\text{critique}}]$, avec valeur par défaut à 1, et le programme utilise $\beta = B.\beta_{\text{critique}}$. Quand une forte vitesse de convergence est souhaitée, le choix de B légèrement plus petit que 1, et dépendant du critère de convergence, est recommandé. Par exemple, les tests ont montré que si le critère de convergence est de passer le plus vite possible dans une tranche de $1/10^{eme}$ de la distance initiale, un bon choix est $B = 0.7$.

Le processus est similaire pour un effecteur de rotation. Le calcul de la distance objet-cible se fait ici en utilisant les quaternions (voir annexe B), et en évaluant l'angle entre deux orientations par :

$$d(q_a, q_b) = angle(q_a.q_b^{-1}) = 2\arccos(Re(q_a.q_b^{-1}))$$

Nous calculons le couple nécessaire pour tourner l'objet d'une fraction α vers l'orientation de la cible durant le temps de relaxation Δt, avec une compensation de la vitesse de rotation régulée par un paramètre β. En reprenant l'équation d'Euler pour les orientations on obtient :

$$\vec{\mathcal{C}}(t) = \frac{2.J\left(\alpha\vec{W} - \beta\vec{\omega}(t)\right)}{\Delta t} - \vec{\omega}(t) \wedge J\vec{\omega}(t) \tag{4.3}$$

$$\text{où} \quad \tilde{W} = \frac{R_{\text{cible}}R_{\text{objet}}^{-1} - I}{\Delta t} \tag{4.4}$$

Voyons maintenant comment les positions et orientations-cibles sont calculées à partir des positions-clés définies par l'utilisateur et du mouvement de l'objet associé.

4.2.2 Calcul de la nouvelle position cible

Comme précisé dans l'introduction, le graphiste spécifie grossièrement la trajectoire de l'objet muni d'un effecteur par un ensemble de positions et/ou d'orientations-clés. Ces données sont d'abord interpolées pour obtenir une trajectoire continue pour la cible. Les techniques pour interpoler entre positions et orientations-clés (converties en quaternions) sont détaillés dans l'annexe C.

Les positions/orientations de la cible pour le calcul des forces et/ou couples appliqués par les effecteurs à chaque pas de temps sont représentées par les paramètres courants le long de la courbe spline.

Déplacer la position-cible à une vitesse constante le long de la courbe ne serait pas une bonne solution. En effet, l'objet muni d'un effecteur peut être ralenti par une collision, et ne suivra alors pas le chemin prédéfini avec suffisamment de précision si la cible s'est trop avancée. De plus, la force ou le couple

de rappel de l'effecteur deviendrait trop fort dans ce cas, produisant ainsi des variations non désirées de la vitesse de l'objet.

Nous préférons calculer la position suivante de la cible à partir d'une *distance de base* spécifiée par l'utilisateur. Le système assure que tout point de la trajectoire finale de l'objet sera au plus à cette distance du chemin prédéfini.

Pour cela, le système essaie de toujours garder la cible à la distance de base de l'objet, avec la contrainte que la cible ne revienne jamais en arrière le long de son chemin. Ainsi, si l'objet est repoussé en arrière, sa position-cible associée reste la même pendant quelques pas de temps. Si l'objet avance vite, alors la cible aussi.

Il est intéressant de noter qu'en l'absence de forces extérieures, on peut approximer raisonnablement la vitesse moyenne de l'objet aussi bien que de la cible par la formule $v_{moy} = \alpha.d_{base}/\Delta t$. Ainsi, plutôt que de donner des paramètres peu explicites comme α et β et la distance objet-cible, l'utilisateur peut exprimer ces paramètres en choisissant la *vitesse moyenne souhaitée* v_{moy} dont on vient d'établir la formule, la *marge d'erreur souhaitée* d_{base} et le *coefficient d'amortissement* B, qui ont tous trois un sens intuitif. On verra dans les exemples que ces coefficient correspondent bien au sens qu'on leur attribue, et le chapitre 5 montrera une application de la vitesse moyenne pour la synchronisation.

4.3 Traitement des situations complexes

La méthode décrite jusqu'ici permet la combinaison de simulation physique sur un objet isolé avec le contrôle de trajectoire. La force de l'effecteur décrite ci-dessus est encore suffisante dans le cas où une force externe est appliquée sur l'objet durant quelques pas de temps. Par exemple, quand une collision avec un autre objet est détectée, l'objet contrôlé est dévié de sa trajectoire, mais revient plus près du chemin prédéfini après un certain temps, du fait de l'action de ses effecteurs.

Cependant, la plupart des objets utilisés en animation par ordinateur sont sujets à des types variés d'actions externes pendant une séquence d'animation. Certaines de ces actions résultent en des forces externes continues comme la gravité ou le frottement fluide, alors que certaines autres ne durent que quelques pas de temps comme la plupart des forces de collision. D'autres actions externes qui durent dans le temps peuvent être extrêmement complexes, comme les interactions entre composants voisins dans une structure articulée, à la fois petites et très fluctuantes. De ce fait, l'ensemble des forces externes doit être correctement pris en compte pour le contrôle du mouvement.

Pour garder l'objet associé près du chemin prédéfini, le module de contrôle doit prendre en compte l'effet de ces actions externes dans le calcul des forces et des couples. Typiquement, un objet sujet à une force de gravité constante doit la compenser avec son action d'effecteur, sans quoi il restera loin de la trajectoire cible, comme montré dans la figure 4.3.

Cependant, lorsqu'un graphiste introduit des effets d'animation comme un objet rebondissant sur des obstacles à cause de la gravité et des forces de ré-

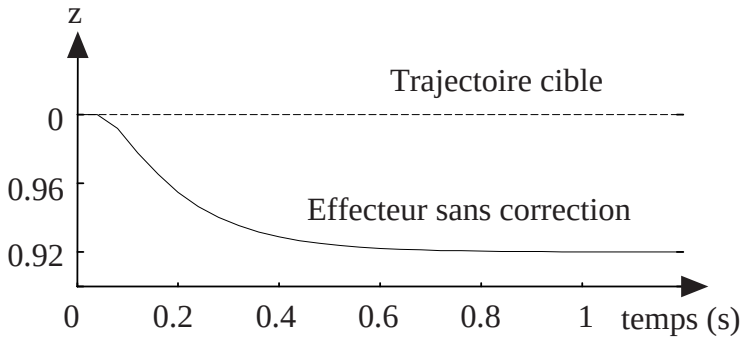


FIG. 4.3 - *Effet de la gravité.*
Balle suivant une cible sur une trajectoire rectiligne suivant l'axe Ox ($\alpha = 0.1, \beta = 0.6$)

ponse, il ne souhaite pas que le module de contrôle cache instantanément ces effets. Un objet ne doit pas réagir trop vite à une action extérieure, mais être toujours dévié par des collisions soudaines. Après une période d'adaptation raisonnable, l'objet doit essayer de compenser l'effet d'une force appliquée qui reste constante dans le temps. En particulier, un objet sujet à la gravité doit être contrôlable.

Tout d'abord, nous devons fournir au module de contrôle un senseur adéquat, capable de percevoir les effets des actions externes. Capter directement ces effets ne serait pas une bonne solution. Du fait de notre approche d'animation des structures complexes, décrite en partie 2.3.2 de l'état de l'art, certaines actions externes peuvent être directement exprimées comme des déplacements plutôt que comme des forces. Nous utilisons donc la position courante de l'objet, son orientation et sa vitesse, comme entrées de l'effecteur.

A chaque pas de temps, le contrôleur a besoin d'approximer la somme des actions externes durant le dernier intervalle de temps. Cela peut être fait en comparant la position courante de l'objet avec sa « position prédite », qu'il essayait d'atteindre au dernier pas de temps. En pratique, nous préférons utiliser les vitesses plutôt que les positions et les orientations pour calculer cette somme. Cela réduit les erreurs numériques, car nous évitons l'approximation résultant de la conversion d'un vecteur rotation en matrice. Les forces externes évaluées sont :

$$\vec{F}_{\text{ext}}(t-\delta t) \;=\; m\frac{\vec{v}(t)-\vec{v}(t-\delta t)}{\delta t} - \vec{F}_{\text{effecteur}}(t-\delta t) \tag{4.5}$$

et les couples :

$$\vec{C}_{\text{ext}}(t-\delta t) = J\left(\frac{\vec{\omega}(t)-\vec{\omega}(t-\delta t)}{\delta t}\right) \;+\; \vec{\omega}(t)\wedge J\vec{\omega}(t) - \vec{C}_{\text{effecteur}}(t-\delta t) \tag{4.6}$$

Lutter contre l'action de ces forces et couples externes pourrait être fait en ajoutant l'opposé de $\vec{F}_{\text{ext}}$ ou de $\vec{C}_{\text{ext}}$ à l'action de l'effecteur. Mais cela ne serait pas une bonne idée pour plusieurs raisons. Comme dit précédemment,

nous ne voulons pas corriger trop vite l'action des forces externes. D'autre part les objets articulés sont souvent sujets à des petits déplacements très fluctuants pour le maintien des contraintes, dus aux forces d'interaction très complexes entre composants voisins.

Utiliser – avec un délai – l'opposé de ces forces produirait un phénomène d'amplification qui conduirait à des oscillations autour de la trajectoire-cible. Une telle situation est représentée dans la figure 4.4 (a), qui représente les oscillations horizontales produites pendant le mouvement contrôlé d'un bras à trois articulations dont l'extrémité doit bouger le long d'une ligne verticale.

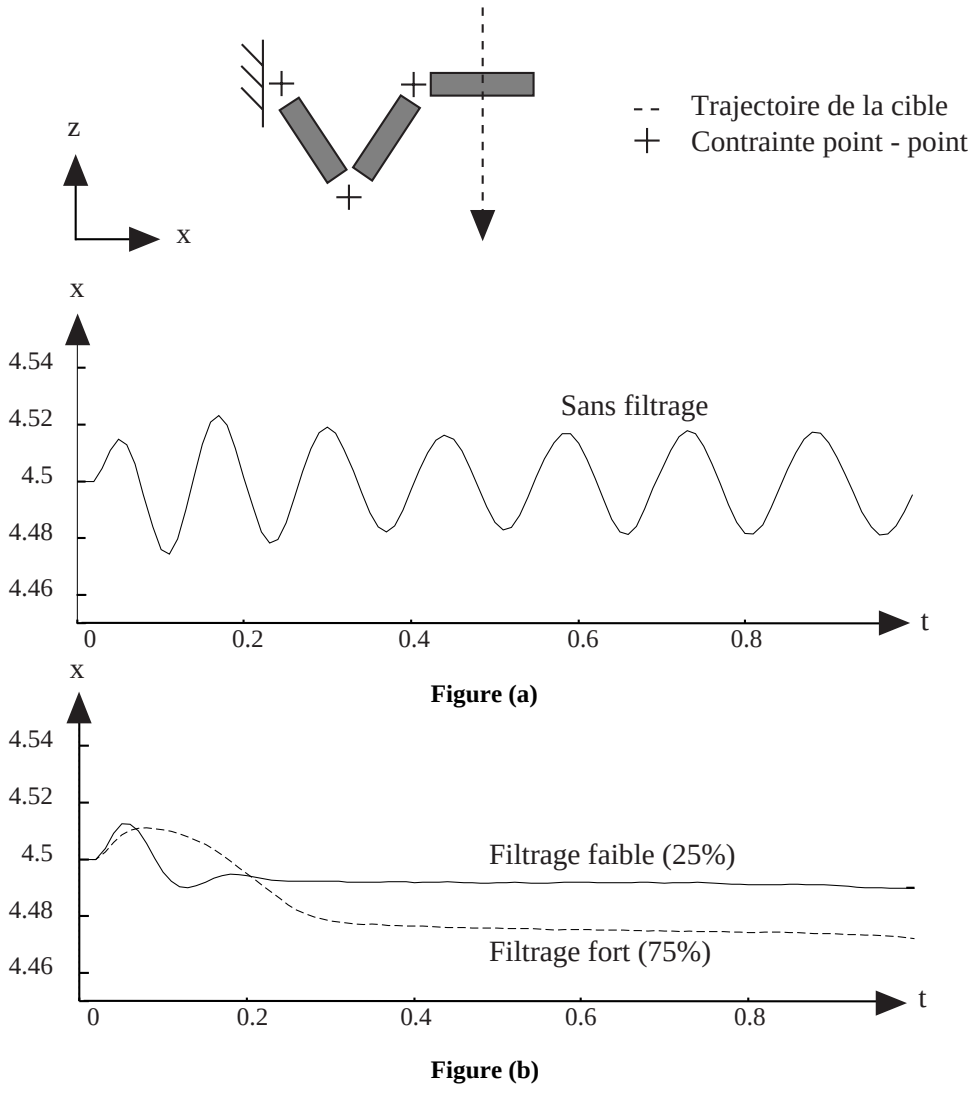


FIG. 4.4 - *Effet du filtrage sur un objet articulé.*
(a) Mouvement sans filtrage des forces des effecteurs : oscillations. (b) Mouvement avec filtrage. Les oscillations disparaissent après un temps d'adaptation.

La solution consiste à utiliser un filtre pour calculer les forces et couples des

effecteurs d'après les approximations des actions externes. Le premier avantage de ce filtre est de produire une variation plus douce des forces des effecteurs (la figure 4.4 (b) montre que les oscillations disparaissent). Cela donne également un paramètre pratique pour réguler le délai entre les actions externes et la réponse du module de contrôle.

Pour offrir un filtrage indépendant du pas de temps adaptatif utilisé durant le processus de simulation, l'utilisateur spécifie un paramètre γ représentant la proportion de filtrage effectué par seconde. La force ou le couple de correction de l'effecteur est ensuite calculé d'après l'approximation des actions externes et des valeurs de corrections précédentes en utilisant le paramètre $\gamma\delta t$ pour le filtre :

$$\vec{F}_{\text{corr}}(t) = -\gamma\delta t\vec{F}_{\text{ext}}(t-\delta t) + (1-\gamma\delta t)\vec{F}_{\text{corr}}(t-\delta t) \tag{4.7}$$
$$\vec{C}_{\text{corr}}(t) = -\gamma\delta t\vec{C}_{\text{ext}}(t-\delta t) + (1-\gamma\delta t)\vec{C}_{\text{corr}}(t-\delta t) \tag{4.8}$$

L'algorithme complet de calcul des forces et des couples des effecteurs peut donc être décrit par le processus suivant :

1. Calcul de la composante principale de l'action de l'effecteur par les équations (4.2) et (4.3).

2. Calcul des forces de compensation des actions externes observées par ajout du terme de correction donné par (4.7) et (4.8), où $\vec{F}_{\text{ext}}$ et $\vec{C}_{\text{ext}}$ sont calculés d'après (4.5) et (4.6).

3. Si nécessaire, tronquer des forces et couples à leur valeur maximum autorisée, pour obtenir des effecteurs de puissance limitée.

4.4 Résultats

4.4.1 Exemples en dimension 2

Ce paragraphe est destiné à montrer l'influence des différents paramètres définis dans les sections précédentes sur l'amortissement, les oscillations, les collisions, et l'influence du pas de temps adaptatif. Ces tests ont été faits sur des exemples simples, afin de mettre en évidence les comportements du modèle à la variation des paramètres, et ses réactions aux interventions externes.

Rappelons que la force principale s'exprime en fonction de α et B par la formule :

$$\vec{F}(t) = 2m\,\frac{\alpha\left(\overrightarrow{x_{\text{cible}}} - \vec{x}(t)\right)}{\Delta t^2} - 2m\frac{\beta\vec{v}(t)}{\Delta t} \quad \text{avec} \quad \beta = B.\beta_{\text{critique}} = B.\sqrt{2\alpha}$$

La distance d_{base} intervient dans cette formule par la relation : $d_{base} = \| \overrightarrow{x_{\text{cible}}} - \vec{x}(t)\|$.

Variations de α, B et d_{base}

Nous montrons ici l'effet de la variation des coefficients associés au calcul de la force ou du couple, α, B et d_{base}.

Voyons tout d'abord l'effet de ces variations pour le mouvement le plus simple possible, à savoir en ligne droite. Nous n'avons pas fait varier d_{base} pour cet exemple, car cela aurait conduit à un simple changement d'échelle. Le mouvement défini est une trajectoire rectiligne uniforme, et l'objet est isolé - pas de forces extérieures -. Il part d'une distance de 1 de la trajectoire, et la distance objet-cible d_{base} est de 0.5 (voir figure 4.5).

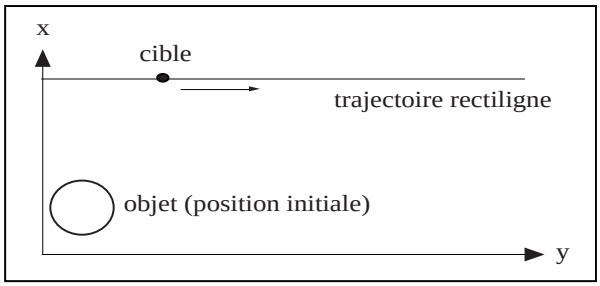


FIG. 4.5 - *Mouvement rectiligne*

Les figures 4.6 et 4.7 montrent l'amortissement de la courbe en fonction du choix de B, pour deux valeurs différentes de α.

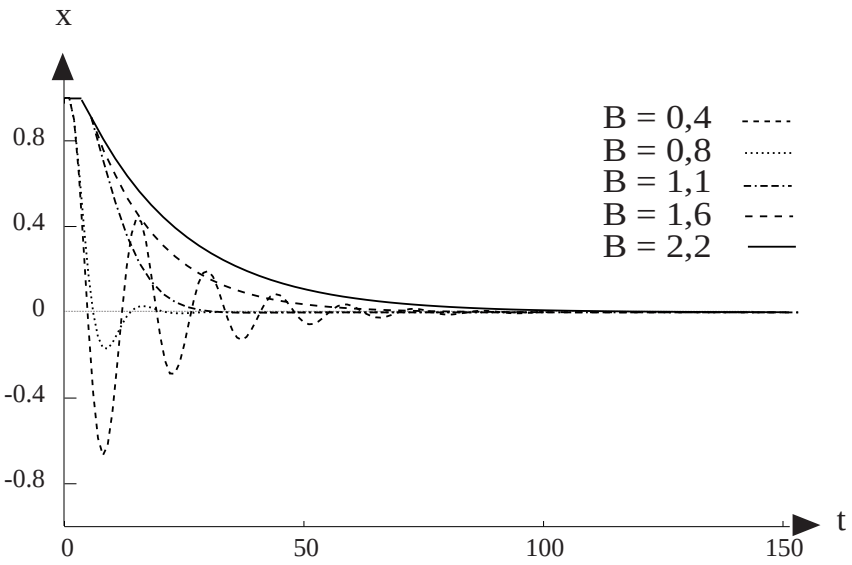


FIG. 4.6 - *Variations de B pour $\alpha = 0.1$.*

On voit apparaître sur la figure 4.6, pour $\alpha = 0.1$, les différentes catégories

de mouvement en fonction du choix de B :

- $0 \leq B < 1$: mouvement sous-amorti, entraînant des oscillations autour de la position d'équilibre;
- $B = 1$: courbe critique, indiquant la limite entre les mouvements sous-amortis et sur-amortis. Il s'agit donc de la courbe de plus forte convergence, parmi celles qui ne dépassent jamais l'objectif;
- $B > 1$: mouvements sur-amortis, de convergence de plus en plus longue quand B augmente.

Sur la figure 4.7 on observe le même type de phénomènes. Cependant, on peut noter que l'ensemble des comportements oscillations/amortissements en fonction de B est décalé, par exemple en notant que la courbe critique est ici aux environs de $B = 1.25$. Cela vient du fait que le résultat théorique qui veut que $\beta_{critique} = \sqrt{2\alpha}$ n'est pas exactement vérifié dans la pratique, en raison des erreurs d'intégration, pour un pas de temps trop grand.

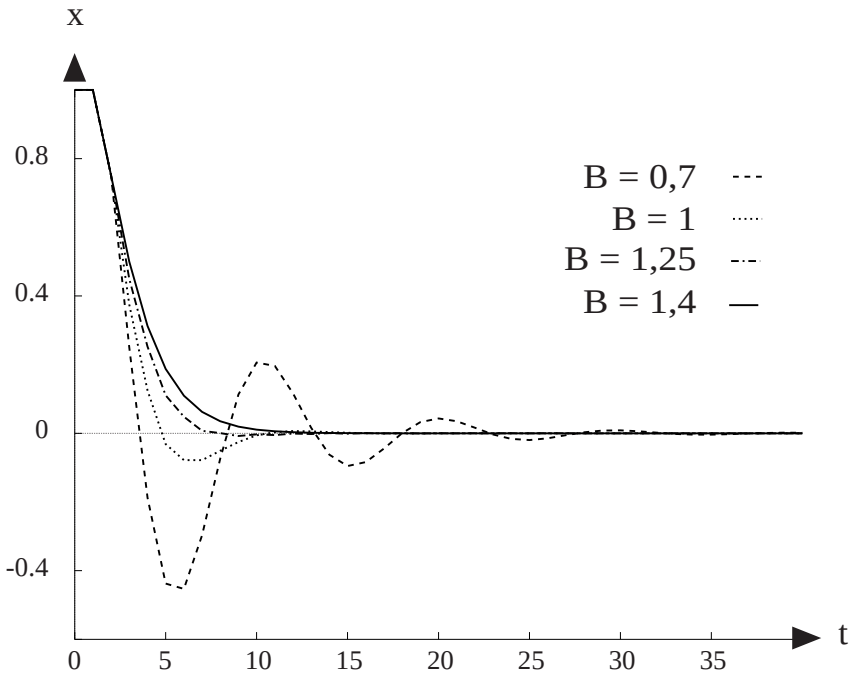


FIG. 4.7 - *Variations de B pour $\alpha = 0.25$.*

Ce résultat se retrouve sur la figure 4.8, qui montre la courbe critique obtenue par mesures, comparée à trois courbes d'interpolation. On observe que si, pour des petites valeurs de α la courbe est parfaitement interpolée par la parabole renversée $y = \sqrt{2x}$, en revanche pour α trop grand les interpolations polynômiales sont plus imprécises, et c'est la courbe exponentielle qui suit le plus fidèlement la courbe critique. Notons que cette figure a été réalisée pour

un pas de temps assez grand, $\delta t = 0.04$. Cependant, en réduisant le pas de temps on obtient moins d'imprécisions et donc une bonne approximation par la parabole. C'est pourquoi, nous avons conservé la valeur de $\beta_{critique}$ conforme à sa valeur théorique.

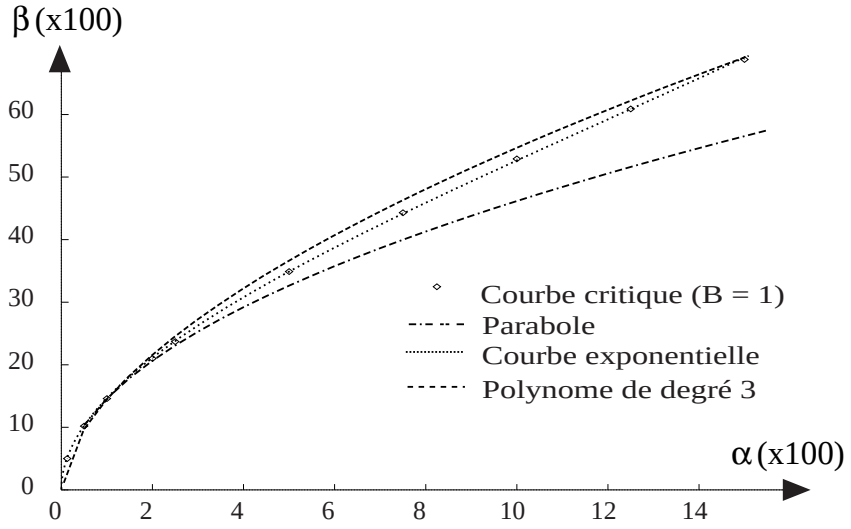


FIG. 4.8 - *Courbe critique et différentes courbes d'interpolation*

L'effet de la variation de d_{base} est montré dans les figures 4.9 et 4.10. Pour cela, nous avons choisi un exemple plus propre à illustrer l'effet de ce paramètre sur la précision avec laquelle la trajectoire est suivie, c'est-à-dire une trajectoire formant un rebond sur le sol (celui-ci étant placé de façon à obtenir un contact pour $x = 0$ sur la courbe). Cette trajectoire est définie, plus ou moins grossièrement, par trois trajectoires, allant d'une parabole à une approximation grossière de celle-ci.

La figure 4.9 montre trois courbes $C1$, $C2$ et $C3$, représentant les splines définies par l'utilisateur, et trois ensembles de points $O1$, $O2$ et $O3$ représentant les trajectoires effectivement suivies. Les paramètres pour cette figure sont $\alpha = 0.1$, $B = 0.4$ et $d_{base} = 0.1$ dans les trois cas, la seule modification étant ici la forme de la spline de départ. On voit sur cette figure l'effet de choisir d_{base} petit devant la taille de la trajectoire (ici, $d_{base} = 0.1$ est de l'ordre de $1/10$ de la taille de la boucle) : celle-ci est finement suivie, quelle que soit sa forme. En revanche, si on choisit d_{base} plus grand ($d_{base} = 0.5$ pour la figure 4.10), on aura un effet de lissage sur la trajectoire effectivement suivie, rendant celle-ci plus conforme au réalisme recherché que la trajectoire initialement spécifiée par l'utilisateur.

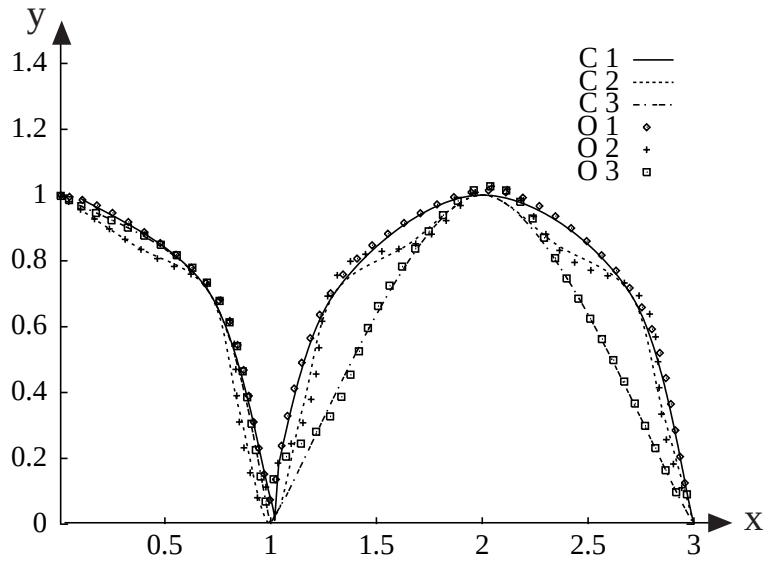


FIG. 4.9 - *Rebond au sol (d_{base} choisie petite)*

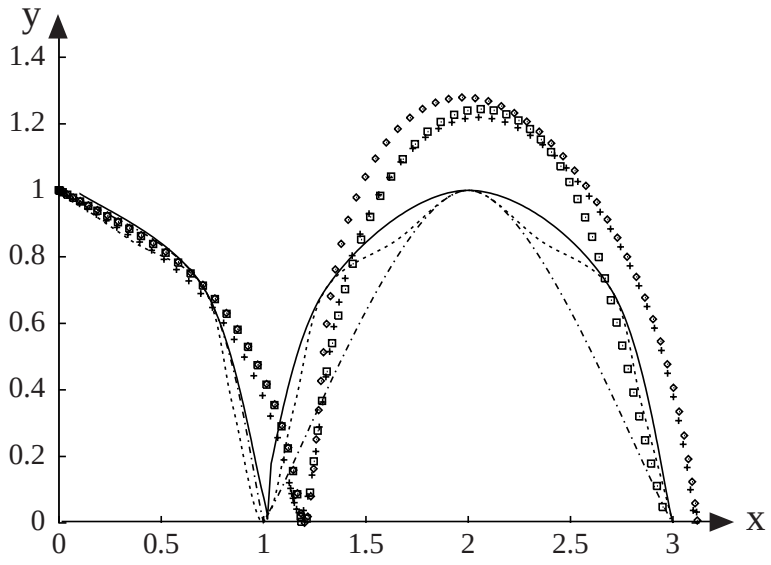


FIG. 4.10 - *Rebond au sol (d_{base} choisie grande)*

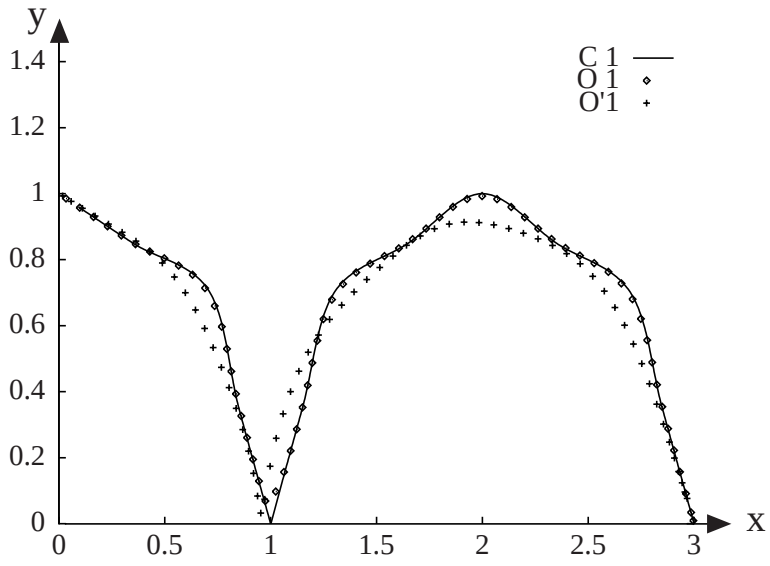


FIG. 4.11 - *Rebond au sol (valeur différente de B)*

La figure 4.11 présente le même type de résultat, avec une valeur différente de B, cette fois sous-amortie. Les ensembles de points $O1$ et $O'1$ montrent la trajectoire suivie par l'objet suivant la cible sur la courbe $C1$, avec $d_{base} = 0.1$ pour $O1$ et $d_{base} = 0.5$ pour $O'1$; $\alpha = 0.1$ et $B = 1$ dans les deux cas. On voit que, si le mouvement est plus amorti, on obtient également le même type de lissage que pour l'exemple précédent pour d_{base} choisi grand, conduisant encore une fois à une courbe plus proche de la parabole.

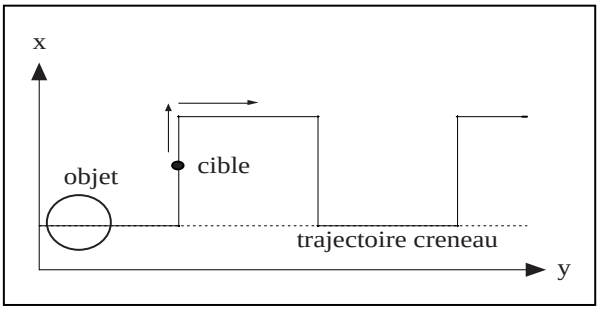


FIG. 4.12 - *Trajectoire utilisateur en forme de créneau*

La figure 4.13 regroupe le type de résultat sur d_{base} et B montré ci-dessus, cette fois-ci sur une trajectoire physiquement irréalisable : un créneau (représenté sur la figure 4.12). On observe les mêmes phénomènes, d'amortissement ou non en fonction de B d'une part, et de lissage en fonction de d_{base} d'autre part. Dans cet exemple, changer la distance d_{base} permet de rester plus ou moins proche de la trajectoire initiale, et donc de produire un mouvement plus ou moins lisse ou anguleux (dans les courbes 2 et 5 on a changé d_{base}, toutes autres

choses étant égales par ailleurs). De la même façon que pour le mouvement rectiligne, la variation de B (entre les courbes 2, 3 et 4) affecte l'amortissemant de la trajectoire finale.

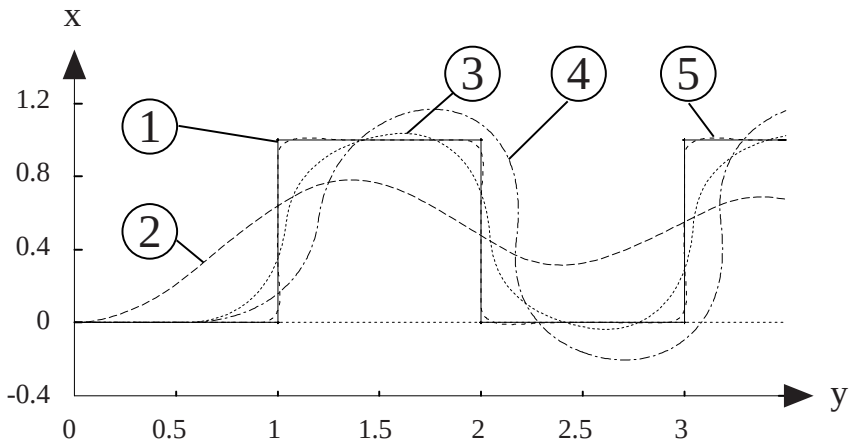


FIG. 4.13 - *Trajectoires effectives pour un créneau. Différentes allures de la courbe suivant les valeurs attribuées aux paramètres α et B, dans le cas d'une trajectoire en forme de créneau*

Déviation par un obstacle

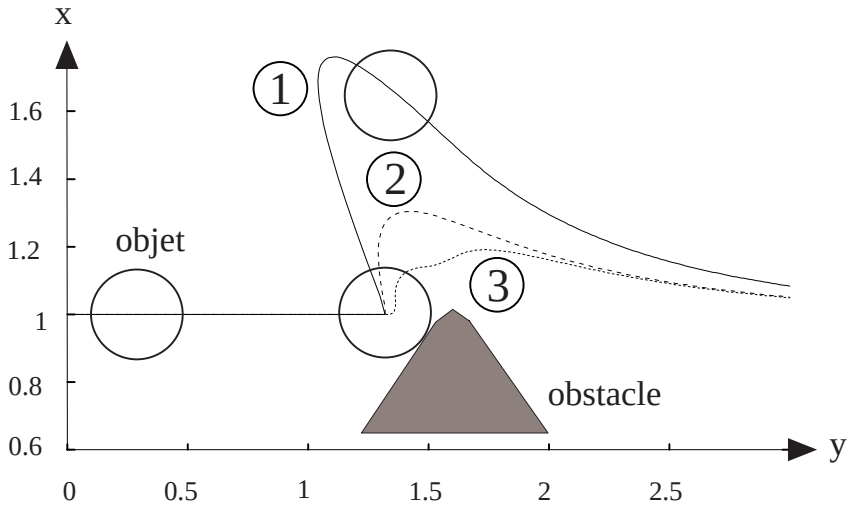


FIG. 4.14 - *Différentes réactions aux collisions suivant les paramètres physiques de l'objet contrôlé*

Il est bien évident cependant que les paramètres physiques donnés aux objets entrent également en compte dans le mouvement obtenu, particulièrement

dans le cas d'interactions. Ceci est illustré dans la figure 4.14, qui montre une trajectoire de nouveau rectiligne, avec ici un obstacle qui empêche l'objet de suivre la trajectoire donnée.

Les courbes 1 et 2 diffèrent par la masse de l'objet (courbes 1 et 3: $m = 1$, courbe 2: $m = 0.4$); et les courbes 1 et 3 par sa raideur (courbes 1 et 2: raideur 2000, courbe 3: raideur 20). Un objet plus mou (raideur moindre) aura tendance à s'écraser plus sur l'obstacle; un objet plus léger à rebondir plus.

Pas de temps adaptatif

Cet exemple illustre un point important du calcul des forces et couples. Dans beaucoup de systèmes de simulation physique, et en particulier dans celui utilisé dans cette thèse, le pas de temps du simulateur n'est pas constant. Cela permet d'utiliser un pas de temps relativement grand quand aucun problème ne survient dans la simulation, et de réduire ce pas de temps quand nécessaire, pendant les collisions ou les contacts prolongés notamment. La figure 4.15 montre de nouveau une balle qui rejoint une trajectoire rectiligne, d'une part avec une simulation à pas de temps constant, d'autre part avec un pas de temps choisi aléatoirement. On voit que, si les trajectoires ne sont pas entièrement confondues, l'erreur obtenue est faible, et en particulier la trajectoire de l'objet est lisse aussi bien pour un pas de temps constant que pour un pas de temps aléatoire. La courbe 1 est obtenue pour $\alpha = 0.1$ et $B = 0.6$, alors que pour la courbe 2 les paramètres sont $\alpha = 0.1$ et $B = 1.4$.

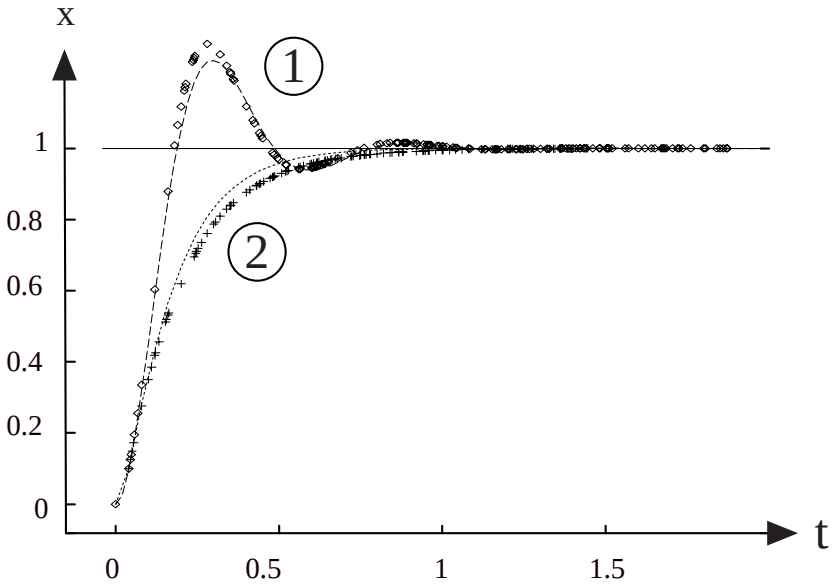


FIG. 4.15 - *Robustesse pour un pas de temps aléatoire*
Les courbes sont à pas de temps constants, alors que les points montrent la simulation pour un pas de temps adaptatif, et cela pour différentes valeurs de B.

4.4.2 Exemples en dimension 3

Film de synthèse « Simply Implicit »

L'animation « Simply Implicit » montre les résultats de notre méthode de contrôle de trajectoire dans le cas d'un objet simple contrôlé en translation : une balle élastique sujette à la gravité et aux interactions (comprenant la réponse élastique aux chocs et le frottement) avec d'autres solides. Cette animation a été conçue pour la présentation du matériau implicite élastique décrit dans [Gas93].

Le script dessiné par le graphiste (Figure 4.16) montre la balle rebondissant sur une étagère, et entrant en collision avec un ensemble de jouets déformables jusqu'à ce qu'elle les chasse tous de l'étagère.

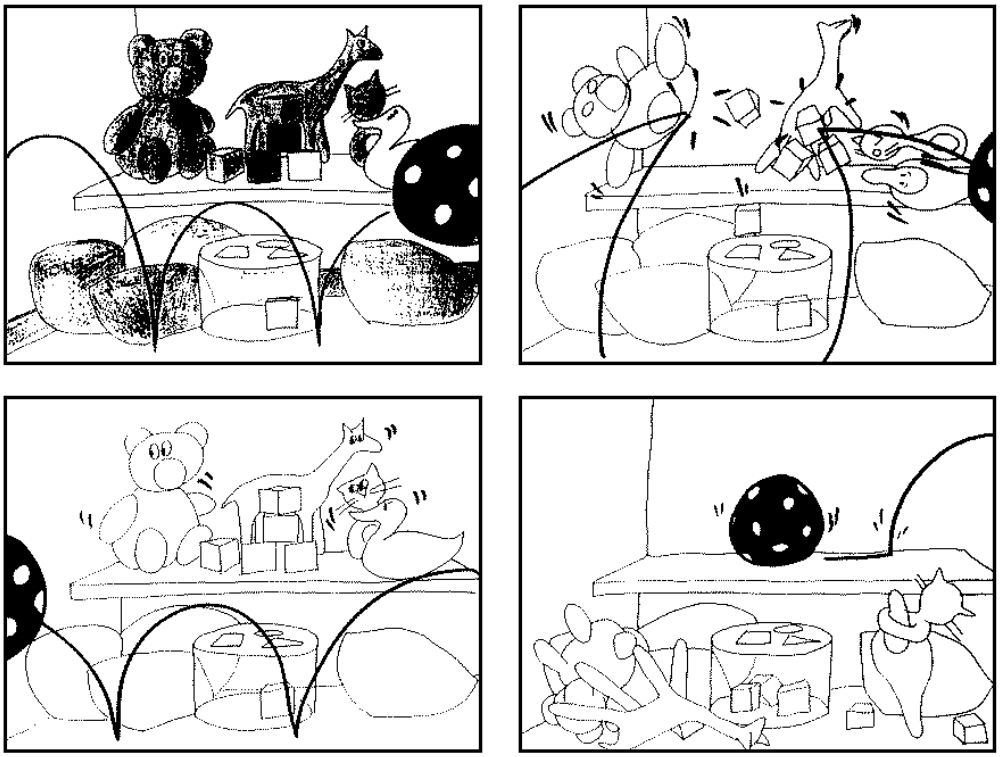
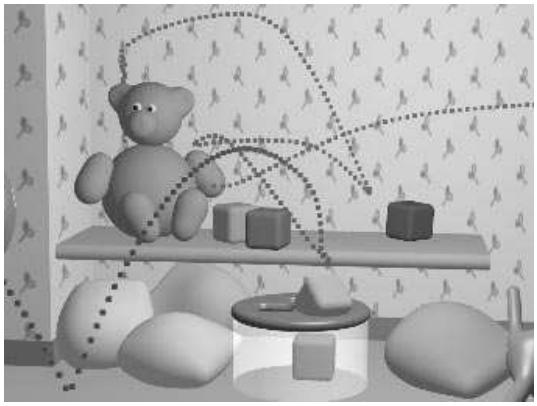
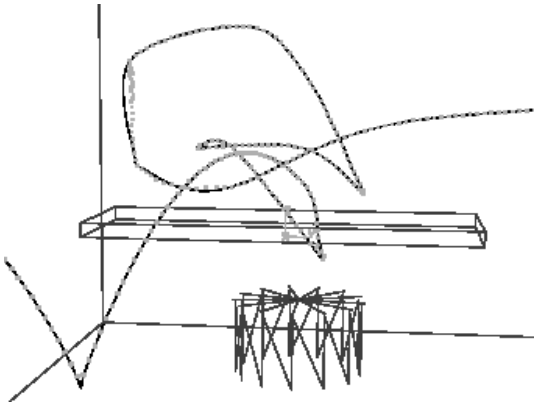


FIG. 4.16 - *Scénario donné par une graphiste, Morgane Furio*

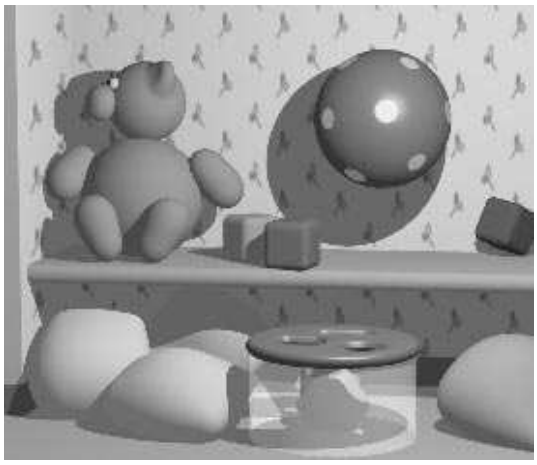
Pour obtenir l'animation finale, le modèle de la balle est muni d'un effecteur en translation. Aucun contrôle de trajectoire n'est appliqué aux autres objets, composés d'objets implicites élastiques, et animés par utilisation directe du module de simulation. Pendant ce processus, le graphiste n'a pas à considérer les propriétés physiques de la balle (masse, inertie, raideur), ni les nombreuses collisions et contacts qui se produisent durant le mouvement. Le système *corrige* automatiquement la trajectoire définie par le graphiste en fonction de tous ces paramètres (voir figure 4.17). En particulier, la rotation de la balle est entièrement causée par l'action des forces de frottement durant les chocs.



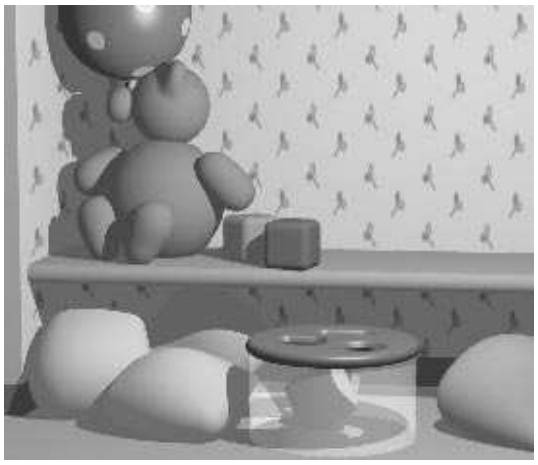
a: Trajectoire de la balle dans l'environnement complet.



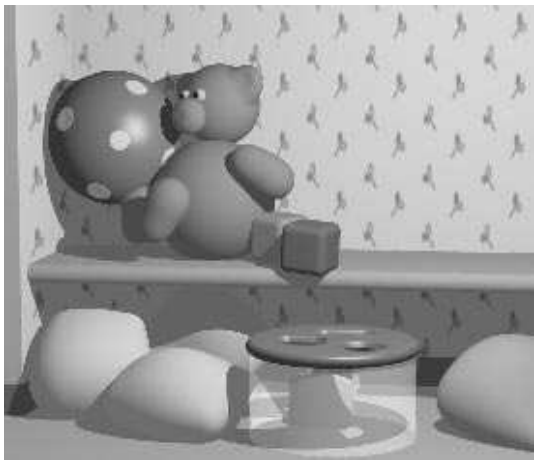
b: Trajectoires de la balle et de sa cible. Les différences entre les points et la courbe font apparaître les corrections de trajectoire dues aux collisions.



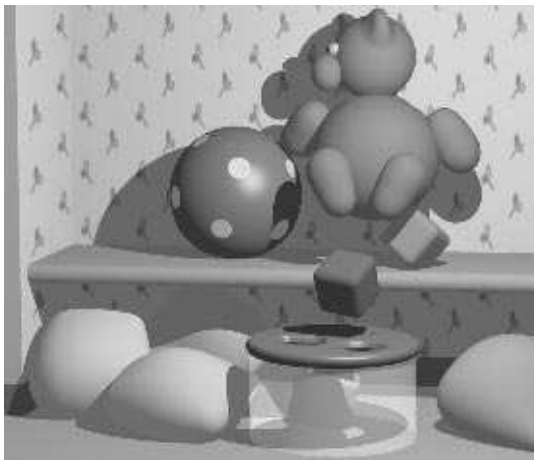
c: La balle entre dans la scène...



d: ...percute le mur et l'ours...



e: ...pousse l'ours de l'étagère...



f: ...et commence à rouler sur l'étagère.

FIG. 4.17 - *Simply Implicit*

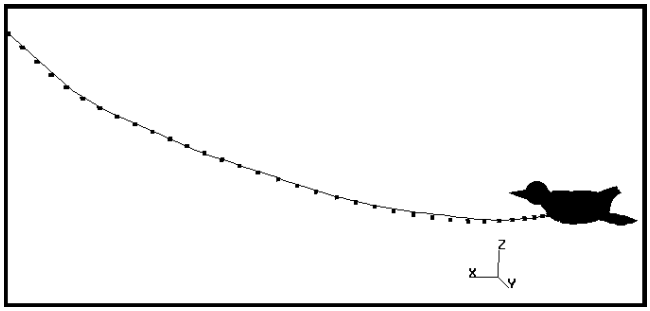
Exemple articulé : Vol d'oiseau

La figure 4.18 montre les résultats pour le mouvement d'un objet articulé contrôlé par quatre effecteurs.

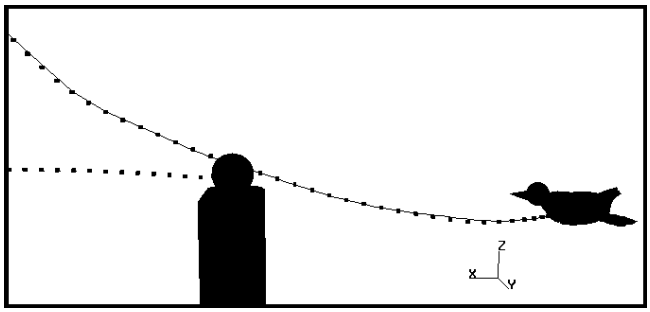
L'objet (un oiseau) est composé d'un corps rigide et de deux ailes rigides, connectées au corps par des charnières avec contraintes d'angle. Le corps est muni d'effecteurs en translation et rotation, et les ailes en rotation seulement. Grâce au module « contraintes de déplacement » assurant le maintien des charnières entre le corps et les ailes, aucun effecteur de translation n'est nécessaire pour les ailes. Des rotations-clés « haut » et « bas » sont données pour les ailes pour simuler le vol de l'oiseau. Les différentes animations montrent comment la trajectoire de l'oiseau est modifiée par différentes collisions. (La figure ne montre que la trajectoire en translation du corps, par souci de clarté).

Exemple articulé II : Serpent

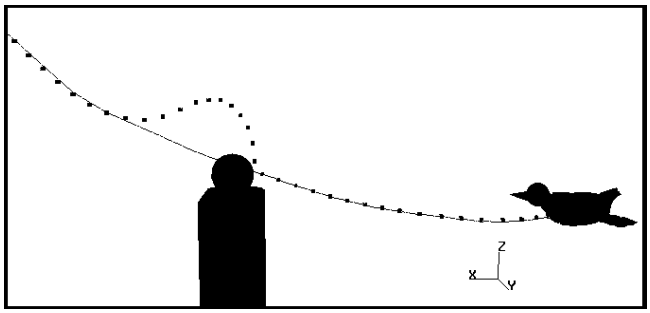
La figure 4.19 montre un serpent composé de neuf liens connectés par des contraintes point-point. Une trajectoire en translation est spécifiée pour un effecteur situé dans la tête, passant à travers un cylindre déformable. On voit le corps du serpent se déplier de sa position initiale, entrer en contact avec le cylindre, et ralentir tout en le tirant.



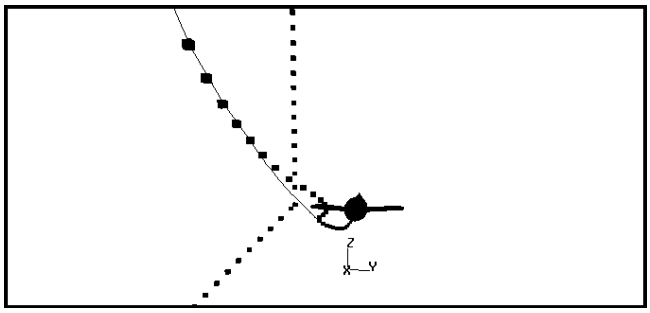
L'oiseau seul : il est légèrement dévié par l'inertie



L'oiseau et une balle légère : seule la balle ressent l'impact



Avec une balle lourde : seul l'oiseau est dévié



Avec une balle tombant : les deux sont déviés

FIG. 4.18 - *Variations du vol de l'oiseau en fonction de différentes actions externes*

Courbes : trajectoire spécifiée par l'utilisateur; pointillés : trajectoire effective.

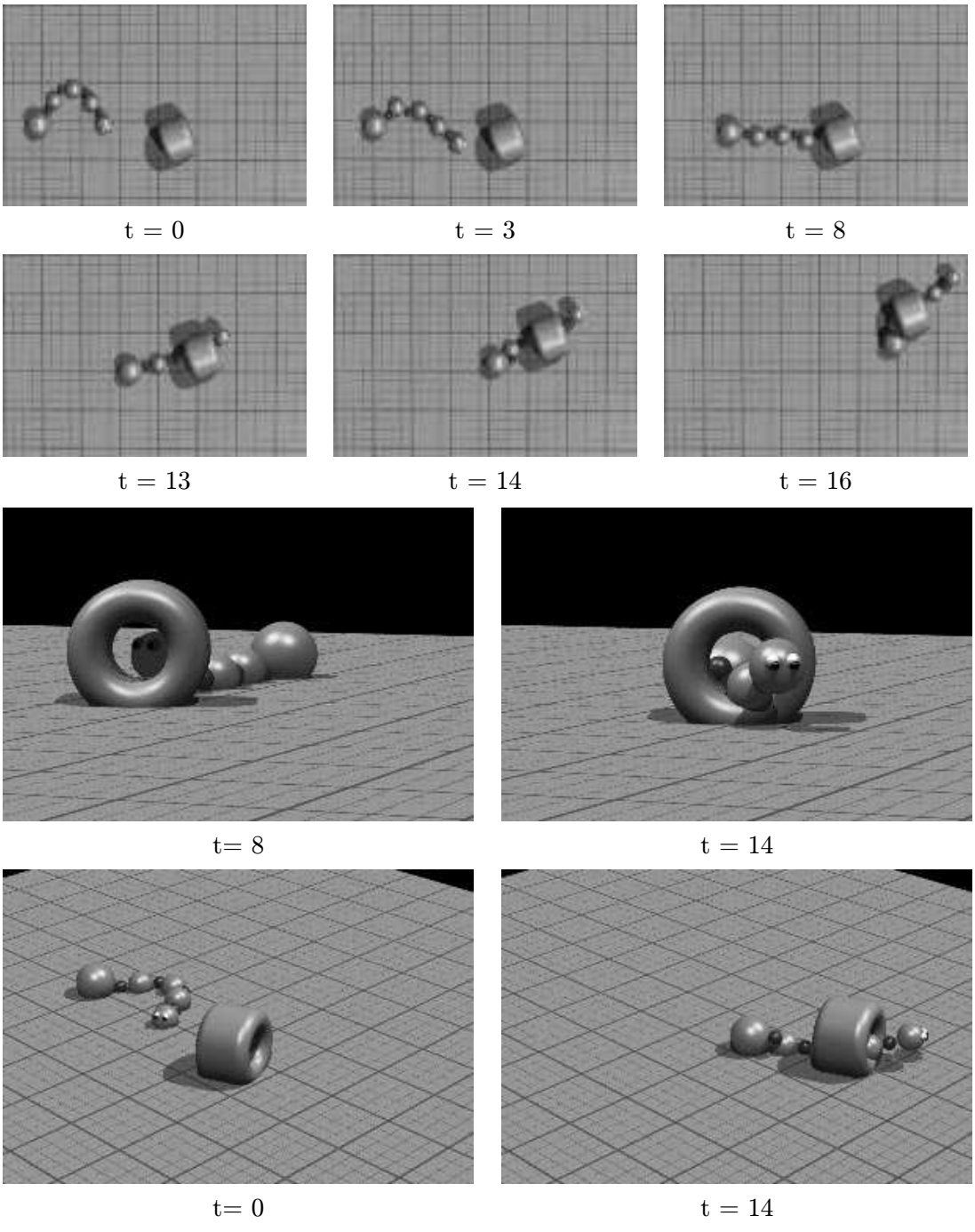


FIG. 4.19 - *Serpent interagissant avec le sol et avec un cylindre déformable*

4.5 Comparaison avec les modèles existants

Nous avons vu dans l'état de l'art un certain nombre de modèles permettant d'utiliser de façon plus ou moins systématique les lois de la dynamique pour améliorer le réalisme et/ou augmenter l'automatisme du mouvement. D'autres modèles, plus classiques, dans le domaine de l'automatique, traitent de problèmes similaires à ceux abordés ici. L'objet de cette section est de revoir à la lumière de notre modèle certains de ces modèles, dont les concepts ou les techniques développés permettent d'établir un parallèle avec le modèle proposé.

Penchons-nous tout d'abord sur les techniques purement descriptives d'interpolation entre positions-clés, que nous avons déjà présentés dans l'introduction, et dont l'annexe C décrit un exemple.

Comme un certain nombre d'autres modèles utilisant la dynamique, celui que nous avons présenté utilise la spécification de positions et orientations clés par l'utilisateur comme point de départ de l'animation. Leur but est à la fois similaire et relativement différent. Similaire, parce qu'il fixe le cadre de l'animation, et permet de définir de façon relativement précise l'enchaînement des événements. Et en même temps différent, parce que dans le cas des techniques d'interpolation, cette définition constitue l'essentiel du travail de création d'une animation, qui n'est ensuite suivi que d'une interpolation automatique assez simple, à base de polynômes de degré réduit, trois le plus souvent. Ce qui signifie pour l'utilisateur la nécessité de contrôler très finement le résultat de ses spécifications, en particulier dans le cas de nombreux objets en interaction.

Dans notre modèle en revanche, le but de la spécification de positions et d'orientations-clés est de donner une ébauche de trajectoire, qui sera ensuite corrigée au cours d'une simulation physique, en fonction d'un petit nombre de paramètres simples et intuitifs, tels que la « marge d'erreur souhaitée », représentée par la distance objet-cible, le « coefficient d'amortissement », et la « vitesse moyenne souhaitée ». Prenons l'exemple de la balle dans la séquence *Simply Implicit* présentée plus haut. Cette séquence, pour être réalisée par un modèle à base de positions-clés, exigerait un nombre d'autant plus conséquent de réglages fins que de nombreux jouets entrent en collisions les uns avec les autres et avec leur environnement tout au long de la séquence. Le même phénomène apparaît pour les déformations entre objets. Or ici nous avons un modèle qui permet de gérer ces déformations, ces collisions, et en même temps de contrôler la trajectoire de l'objet de manière relativement précise. Cette séquence présente donc le cas typique d'animations dans lesquelles le modèle peut être utilisé. Par ailleurs, les exemples en dimension deux ont montré (en particulier les figures 4.9 et 4.10) qu'un choix judicieux des paramètres permet d'obtenir des effets physiques convaincants sans avoir à définir la trajectoire de façon trop précise. De plus, le fait que chaque paramètre a une signification intuitive permet de régler ceux-ci facilement pour obtenir l'effet désiré.

Une autre source d'inspiration pour le modèle présenté est le modèle de contrôleur proportionnel-dérivé (PD) ou proportionnel-intégral-dérivé (PID) [Sev89] qui permettent de calculer une action à exercer sur un effecteur pour

atteindre un objectif donné, en réduisant un terme d'erreur exprimé en fonction de l'objectif.

Voyons tout d'abord le cas du contrôleur PD. Dans le problème qui nous occupe, deux possibilités sont à envisager pour exprimer cette erreur. Elle peut tout d'abord s'exprimer par la formule $\varepsilon = x_{cible} - x$, avec une force donnée par $f = k_p\varepsilon + k_d\dot{\varepsilon}$, ce qui revient à dire que l'erreur à corriger est la distance objet-cible, que notre modèle garde volontairement constante. Une autre façon d'exprimer l'erreur, pour tenir compte de la distance objet-cible constante, est donnée par Pintado et Fiume dans [PF88]. L'erreur s'exprime alors comme la différence entre la distance objet-cible courante, et la distance souhaitée d_{ref}, qui correspondrait au d_{base} de notre modèle. L'expression de la force est toujours $f = k_p\varepsilon + k_d\dot{\varepsilon}$.

Dans les deux cas, l'utilisation d'une telle force dans notre modèle conduirait à une situation de bloquage. La première expression conduirait, avec notre algorithme de progression de la cible, à un objet uniformément accéléré : l'erreur, donc la force f (et l'accélération $\ddot{x}$ en l'absence de forces externes) restant toujours de même norme, proportionnelle à la distance objet-cible. La deuxième conduirait à l'arrêt de l'objet dès que la condition de distance aura été rencontrée une fois.

Ce problème de bloquage s'explique par le fait que le principe de base du contrôleur PD est de réduire une erreur par rapport à un modèle de guidage fixé pour la cible dès le départ, alors nous avons vu que nous ne souhaitons pas ce type de comportement, mais au contraire adapter la vitesse de l'objet en fonction des événements survenus lors de l'animation.

D'une manière analogue, l'expression de la force de correction par filtrage des actions externes évoque le terme intégral d'un contrôleur PID, et en effet leur but est relativement similaire : corriger les actions externes de longue durée. Pour un contrôleur PID lié à une erreur $\varepsilon = x_{cible} - x$, la force vaut :

$$f = k_p\varepsilon + k_d\dot{\varepsilon} + k_i\int\varepsilon$$

alors que dans notre cas, f vaut $f = k_p\varepsilon + k_d\dot{x} + f_{corr}$, avec :

$$\begin{aligned} f_{corr}(t_n) &= \gamma.f_{ext}(t_{n-1}) + (1-\gamma)f_{corr}(t_{n-1}) \\ &= \gamma\left[f_{ext}(t_n) + (1-\gamma)^1.f_{ext}(t_{n-1}) + ... + (1-\gamma)^n.f_{ext}(t_0)\right] \end{aligned}$$

Principalement, l'effet du terme intégral du PID est d'ajouter un terme constant à la force de l'effecteur : dans l'idéal, si l'erreur devient nulle après un temps d'adaptation, la force devient alors égale au terme intégral. Cela permet en effet de corriger l'action de forces constantes, ou exercées durant un grand intervalle de temps, en raison du choix nécessairement petit du coefficient k_i pour la convergence du modèle. Notre approche permet également la correction d'un terme constant ou très durable : si $f_{ext} = cte = f_0$, alors $f_{corr} = \left[1 - (1-\gamma)^{n+1}\right] f_0$ au temps t_n. Mais en même temps nous cherchons à établir un temps de réponse, et à traiter les forces ayant une action à moyen terme, en effaçant plus vite leur influence si elle n'est pas répétée, par les termes

en puissance de $(1-\gamma)$ qui décroissent plus ou moins vite selon le choix du paramètre. En conséquence, notre méthode peut être vue comme un contrôleur-PID généralisé, adapté à notre but spécifique.

Tournons nous maintenant vers un modèle présentant un certain nombre de ressemblances avec notre approche, aussi bien dans les concepts que dans les techniques mises en œuvres : l'approche de Pintado et Fiume, qui proposent un modèle basé sur la dynamique pour apporter un certain réalisme à la spécification de trajectoires par positions-clés, et notamment un module de suivi de cible par un contrôleur PD ou PID, avec une distance objet-cible de référence d_{ref}.

Si la présentation des deux approches met en valeur leurs points communs, il convient d'insister sur leurs différences. Tout d'abord, l'utilisation d'un contrôleur PD ou PID, comme nous venons de le voir, présente des différences avec notre approche, et notamment empêche le déplacement de la cible de s'effectuer en fonction de l'objet. Dans le modèle de Pintado et Fiume, il est nécessaire de fournir une information supplémentaire de mouvement de la cible, qui contraint ce mouvement sans tenir compte des interactions survenues au cours de la simulation.

D'autre part, l'utilisation directe du contrôleur faite par [PF88] pour le calcul des positions et vitesses de l'objet guidé, sans système d'intégration des équations de la dynamique basé sur l'application de forces sur les objets, rend impossible de traiter un des points majeurs de notre approche, qui est de permettre au modèle de réagir aux interactions avec les objets qui l'entourent, et en particulier de modéliser les collisions et contacts entre objets.

Nous avons vu enfin l'utilisation possible des modèles d'optimisation pour améliorer le réalisme d'une séquence définie par un ensemble de positions-clés. On peut voir notre approche comme la recherche d'un compromis entre la précision avec laquelle la trajectoire est suivie d'une part, et le naturel du mouvement d'autre part, engendré par les effets de l'inertie, les interactions entre objets au cours de l'animation et l'action des forces externes. Selon un tel point de vue, on peut envisager d'utiliser un processus d'optimisation pour déterminer, non pas une trajectoire possible, éventuellement paramétrable comme le propose notre modèle, mais la trajectoire qui vérifie le mieux un certain critère lié à ce compromis.

Cette approche, si elle présente l'intérêt évident de permettre à l'utilisateur d'éviter un réglage de paramètres, en donnant tout de suite la « bonne réponse », comporte également certains inconvénients. Outre le fait que l'optimisation est un processus relativement coûteux, et que le traitement des collisions introduit des discontinuités difficiles à traiter par ce type de méthode, le choix d'un critère de minimisation pour une quantité aussi peu quantifiable que le degré de compromis s'avère être une tâche difficile. A l'opposé notre approche propose à l'utilisateur de choisir entre une grande variété d'effets, guidé par un ensemble de paramètres simples et intuitifs.

4.6 Conclusion

La méthode présentée dans ce chapitre devrait grandement simplifier l'utilisation de modèles physiques pour l'animation, en offrant une aide importante à la création d'animations et de déformations réalistes pour des scripts précis. Le but ici n'est pas de calculer l'action de muscles pour effectuer une action donnée, mais plutôt d'accroître le réalisme d'une trajectoire voulue par un graphiste, et qui peut être totalement irréalisable sans aide extérieure. Notre méthode permet aux objets de suivre un script prédéfini qui peut être aussi loin de la dynamique que le souhaite l'utilisateur, en ajoutant les qualités de la dynamique chaque fois que c'est possible.

La spécification de l'animation se fait par la définition de positions et orientations-clés, ce qui permet à l'utilisateur d'avoir un bon contrôle de la séquence, sans qu'il soit nécessaire de dépenser trop de temps à régler attentivement les trajectoires pour améliorer le réalisme, ni pour des questions de détection ou de réponse à des collisions ou de déformations. Pendant un processus de simulation interactive, le système *corrige automatiquement* les trajectoires en fonction des paramètres physiques des objets ainsi que des collisions et contacts détectés durant le mouvement. Les déformations adéquates des objets sont calculées durant le même processus.

La méthode consiste à associer des effecteurs en translation et/ou en rotation aux objets à contrôler. Le module de contrôle utilisé pour calculer l'action des effecteurs à partir d'une trajectoire définie par l'utilisateur dérive d'une version généralisée de contrôleur proportionnel-intégral-dérivé. Les objets contrôlés peuvent prendre en compte les actions externes observées pour réguler leur mouvement. Déviés par une collision soudaine, les objets tendent à compenser les actions externes continues après une période d'adaptation spécifiée par l'utilisateur. La méthode de contrôle fonctionne dans des situations complexes où les objets sont des composants de structures articulées, et est compatible avec la gestion d'un intervalle de temps adaptatif. De plus, le contrôle de trajectoire peut être appliqué sur seulement certains des objets de la scène, tandis que la dynamique pure est utilisée pour les autres. Cela devrait aider le graphiste à concentrer son travail sur les mouvements importants uniquement. Un objet peut être contrôlé en translation mais non en rotation (ou le contraire), laissant le simulateur produire des rotations réalistes d'après les forces de frottement pendant les collisions et les contacts avec les autres objets.

Chapitre 5

Synchronisation et scénarios

5.1 Introduction

Dans le modèle décrit précédemment, nous avons choisi de calculer le planning de l'animation durant la simulation plutôt que de le pré-spécifier. Effectivement, les variations de vitesse doivent dépendre de la complexité de la trajectoire, et des événements tels que collisions et contacts, qui surviennent durant le mouvement. Cependant, la plupart des séquences d'animation produites en image de synthèse nécessitent un travail attentif de synchronisation entre les différents objets de la scène, ou même entre différents composants d'une créature articulée. Par exemple, reproduire un mouvement de marche nécessite une coordination entre les membres inférieurs et supérieurs du personnage.

Le premier outil développé pour satisfaire cette synchronisation est de simplement spécifier la trajectoire d'un objet dans le repère local d'un autre objet. On peut ainsi réaliser des mouvements plus évolués, comme par exemple deux personnes se serrant la main, ou le mouvement d'une jambe par rapport au torse. Nous proposons ensuite un module plus complet pour la réalisation de scénarios, par la définition de contraintes de synchronisation qui lient les mouvements respectifs des objets munis d'effecteurs.

L'utilisateur peut synchroniser différents objets - ou différents composants d'une structure complexe - en spécifiant un graphe d'événements temporels qui lieront ensemble certaines des positions-clés définies pour chacun d'eux. Cela permet la donnée de scénarios complexes, incluant les rendez-vous (les objets adapteront leurs vitesses si l'un d'entre eux est ralenti), et le contrôle de la synchronisation des différents mouvements. Le travail présenté dans ce chapitre a fait l'objet d'une publication à « Graphics Interface '95 » [LG95].

5.2 Définition d'une trajectoire relative à un autre objet

Spécifier un mouvement relatif à d'autres objets peut être plus utile pour certaines applications que de donner une position désirée dans les coordonnées du monde. Par exemple, si deux personnes veulent se serrer la main, l'endroit précis où elles le font n'est pas vraiment important. Ce paragraphe montre

comment définir des positions et orientations-clés dans les repères locaux des objets.

Quand l'utilisateur spécifie l'ensemble des positions-clés définissant la trajectoire désirée d'un objet muni d'un effecteur, il associe à chacunes d'entre elles un repère, local ou du monde. La courbe décrivant la trajectoire de la cible n'est plus alors fixe dans le repère du monde, mais évolue dans le temps en fonction des positions des repères par rapport auxquels elle est définie.

5.2.1 Algorithme

L'algorithme pour calculer le mouvement de la cible est modifié de manière à tenir compte du caractère évolutif de la trajectoire :

1. Pour calculer chaque nouvelle position de la cible, on sélectionne les points de contrôle (les clés) influençant le paramètre cible courant sur la courbe-spline. Nous utilisons ici des splines cubiques par morceaux (annexe C), pour lesquelles chaque portion de courbe dépend de quatre points de contrôle. On ajoute aux points de contrôle un cinquième point de transition, pour permettre à la cible de passer à la portion de courbe suivante.

2. On convertit les coordonnées de ces points de contrôle (si nécessaire) dans le repère du monde, à partir des coordonnées locales de l'instant courant.

   Pour la plupart des applications, le trajectoire ne devrait pas dépendre du mouvement d'un point de contrôle que la cible a déjà dépassé. Le point de contrôle est donc « épinglé » dans sa position courante dans le repère du monde, dès que la cible l'a dépassé.

3. Enfin, quand on a la portion de spline considérée dans le repère du monde, on utilise la méthode de dichotomie habituelle pour trouver le nouveau paramètre-cible, que l'on recherche comme précédemment à une distance donnée de l'objet guidé.

Étant donné que deux objets ne se déplacent pas trop pendant un pas de temps (cette condition est de toute façon requise pour tout système d'animation pas à pas), la trajectoire de la cible reste lisse (voir figure 5.1).

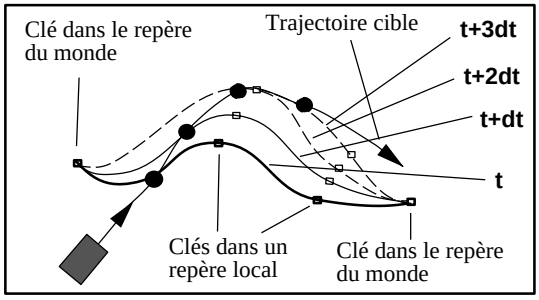


FIG. 5.1 - *Mouvement de la cible pour une trajectoire relative*

5.2.2 Résultats

La figure 5.2 montre deux personnages se serrant la main. La position finale pour chaque main est définie dans le repère local relatif à l'autre main, alors que la position initiale reste la main de départ. La position où les mains se rencontrent est donc calculée automatiquement durant l'animation, et dépend en fait de la force des effecteurs, et de la masse, l'inertie et les positions des articulations de chaque bras. L'animation montre que, même si les positions initiales et finales sont modifiées au cours du temps, la trajectoire est peu modifiée à chaque pas de temps, permettant ainsi une variation fluide entre les trajectoires relatives.

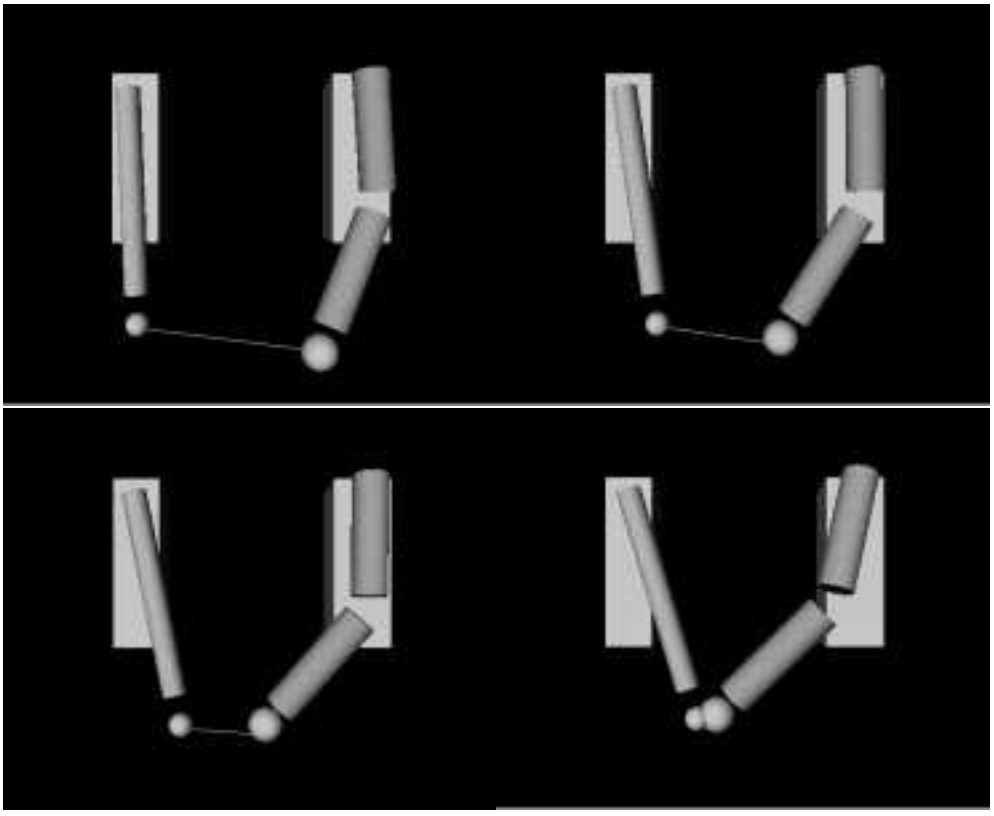


FIG. 5.2 - *Trajectoire relative (I)*

La figure 5.3 montre un bras à trois articulations qui doit renverser une balle se déplaçant sur un wagonnet mobile. La position initiale de la trajectoire est donnée dans le repère fixe, alors que la position de rencontre de la main et de la balle est définie dans le repère local de la balle. Les contacts entre la balle et la main sont automatiquement détectés et traités par la simulation. On observe ici qu'avant le contact main-balle, la trajectoire est modifiée au cours du temps (images un à trois) alors que quand la main retrouve sa position initiale après la rencontre, cette dernière est passée, et donc la position est fixée dans le repère du monde (image 4).

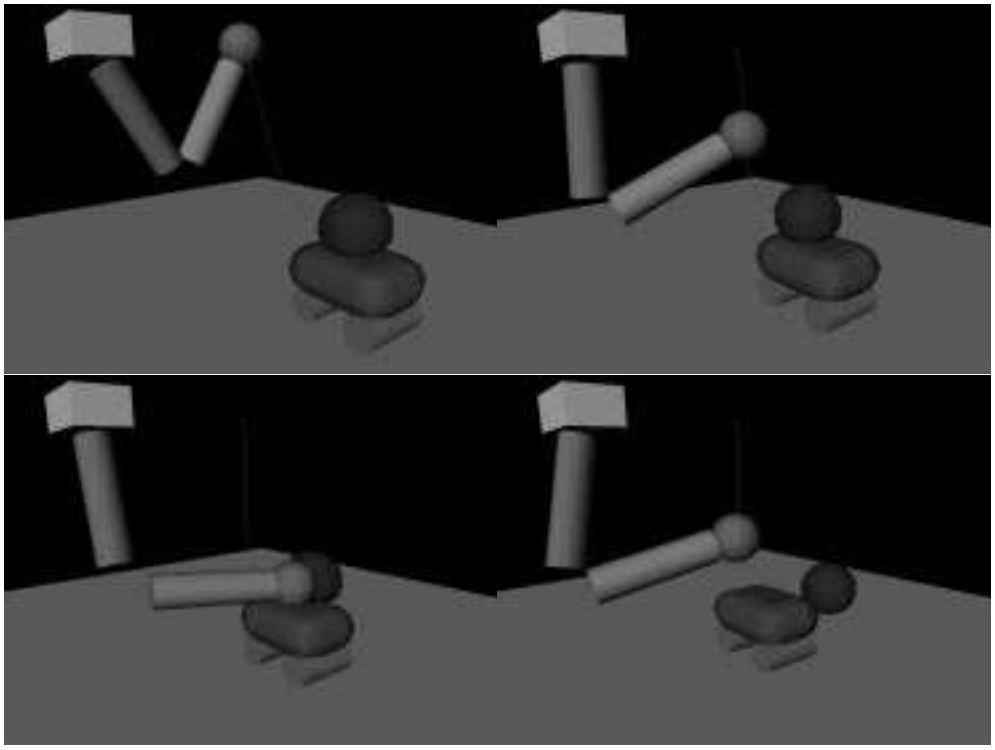


FIG. 5.3 - *Trajectoire relative (II)*

Ce principe sera de nouveau utilisé en troisième partie de ce document, dans le cadre plus complexe de l'animation de personnages. Les positions-clés seront alors des positions en rotation.

5.3 Synchronisation du mouvement

Voyons tout d'abord comment remplir une contrainte de synchronisation seule. Une contrainte de synchronisation est définie en sélectionnant un ensemble de positions (ou orientations) sur les différentes trajectoires des objets définies par l'utilisateur, et en imposant aux objets d'atteindre ces positions tous en même temps. Par exemple, le rendez-vous entre différents personnages à un endroit donné peut être décrit de cette façon. Un autre exemple est un personnage qui doit plier simultanément les genoux et les bras. La synchronisation est alors réalisée entre des positions et des orientations-clés.

Calculer les mouvements sous contrainte de synchronisation est réalisé par la régulation de la vitesse moyenne des différentes cibles par un processus auto-adaptatif.

5.3.1 Régulation de la vitesse des cibles

Comme nous l'avons dit ci-dessus, l'utilisateur spécifie une contrainte de synchronisation au moment où il définit les trajectoires cibles des différents objets, en indiquant qu'un ensemble de positions-cibles clés doivent être atteintes

en même temps. C'est donc les vitesses moyennes des cibles qu'il s'agit d'ajuster. Les cibles doivent adapter leur vitesse suivant les événements (objets déviés par des collisions) qui se produisent durant la simulation, c'est pourquoi nous ne voulons pas prédéfinir ces vitesses en utilisant une reparamétrisation de la spline [SB85, BH89, HP93]. Nous cherchons plutôt un moyen adaptatif de les réguler.

La manière qui semble la plus robuste pour synchroniser un ensemble d'objets consiste à tenir compte de l'objet le plus « en retard » par rapport à l'objectif. Ce retard est calculé sur la cible, en fonction de leur vitesse courante (qui est liée, rappelons-le, à celle de l'objet), et de la distance qui leur reste à parcourir jusqu'au point de rendez-vous.

Un autre point est que nous voulons que la cible retrouve sa vitesse idéale de départ après que la contrainte de synchronisation ait été résolue (les cibles ralenties ne devraient pas le rester, sans quoi la vitesse courante de chaque objet dépendrait de chaque perturbation survenue depuis le début de l'animation). Ainsi, quand aucune réduction de la vitesse de la cible n'a été nécessaire depuis plusieurs pas de temps consécutifs, ou quand la contrainte de synchronisation a été atteinte, la vitesse est augmentée à nouveau jusqu'à la vitesse initialement prévue.

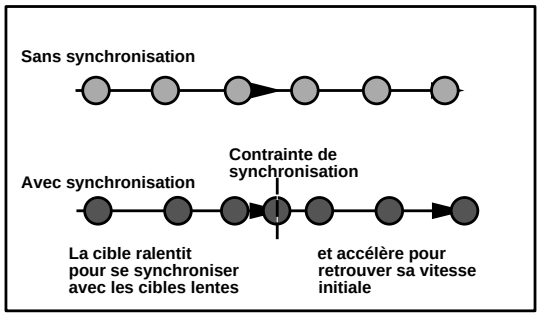


FIG. 5.4 - *Variations de vitesse de la cible dues à la synchronisation*

5.3.2 Choix du paramètre de contrôle

Si aucun événement extérieur (pas de collision, etc) ne survient, la vitesse moyenne d'un objet, et en conséquence de sa cible, est $\alpha.d_{base}/\Delta t$. Pour modifier la vitesse d'une cible, il faut donc changer un ou plusieurs de ces paramètres. De plus, à part les vitesses relatives des objets les uns par rapport aux autres, nous ne voulons modifier aucune des caractéristiques de l'animation : ni l'amortissement autour de la trajectoire (basé sur le rapport α/β^2), ni la précision avec laquelle l'objet suit cette trajectoire (donnée par d_{base}). Les modifications de la vitesse de la cible seront donc réalisées en réglant α, tout en préservant le rapport α/β^2.

5.3.3 Algorithme de synchronisation

Nous recherchons un critère qui nous donne le retard relatif de chaque objet par rapport à une certaine contrainte de synchronisation. Cette contrainte

impose que les cibles des objets reliés par elle doivent atteindre des positions (ou orientations) buts sur leur trajectoires au même moment. En conséquence, nous utilisons la distance relative entre les positions courantes des cibles et leur position but le long des courbes splines pour définir la « proximité » relative des objets contraints. Les vitesses des cibles sont évaluées par comparaison de la distance parcourue par la cible en un pas de temps avec la distance totale qu'il reste à parcourir entre la position courante et la position but.

Dans le système implémenté, l'abscisse curviligne s (définie comme la longueur de la courbe spline entre le premier point et le point courant) est calculée directement grâce à l'utilisation de courbes splines reparamétrées, pour lesquelles le paramètre spline correspond à une approximation de l'abscisse curviligne (cette méthode dérive de [GP90], et est reprise en annexe C.2).

Ainsi le temps nécessaire à une cible c pour atteindre son objectif, d'abscisse curviligne s_{but} est estimé à :

$$t_{estimé} = dt.(s_{but} - s_{courant})/ds$$

où ds est la longueur de courbe parcourue durant le pas de temps dt précédent.

A chaque pas de temps, le module qui règle la vitesse de la cible suit le schéma suivant :

- Vérifier si la cible a modifié sa vitesse récemment. Sinon, elle est autorisée à accélérer d'un certain pourcentage, jusqu'à avoir retrouvé sa vitesse maximale. Ce pourcentage peut être réglé par l'utilisateur, pour régler le temps de retour à la normale.
- Évaluer le temps que chaque cible prendrait à sa vitesse courante pour atteindre son but, et garder le maximum de ces temps comme base de réglage pour les vitesses (dans la mesure où les cibles s'aligneront sur la plus lente) : $t_{souhaité} = max_c(t_{estimé}(c))$.
- Réduire la vitesse de chaque cible pour qu'elle atteigne son but en un temps $t_{souhaité}$: $\alpha_{t\ +\ dt}(c) = (t_{estimé}(c)/t_{souhaité}).\alpha_t(c)$.

On utilise ensuite la méthode présentée au chapitre 4 pour déplacer les cibles, calculer les forces des effecteurs et simuler tous les objets.

5.3.4 Résultats

La figure 5.5 montre une application de cet algorithme. Deux bras articulés, de masse et d'inertie différentes, doivent vérifier une contrainte de synchronisation en se relevant : la position-clé intermédiaire (bras à 45^o) doit être atteinte simultanément par les deux bras. Cette contrainte est matérialisée sur les images par une barre horizontale, joignant les deux points à atteindre simultanément. Les positions des deux cibles au cours du temps sont montrées en figure 5.6. Le paramètre spline, proportionnel à la longueur de la trajectoire, est ici montré plutôt que la position dans l'espace, pour donner une meilleure vue sur la vitesse relative des deux cibles. On voit sur cette figure que le mouvement est

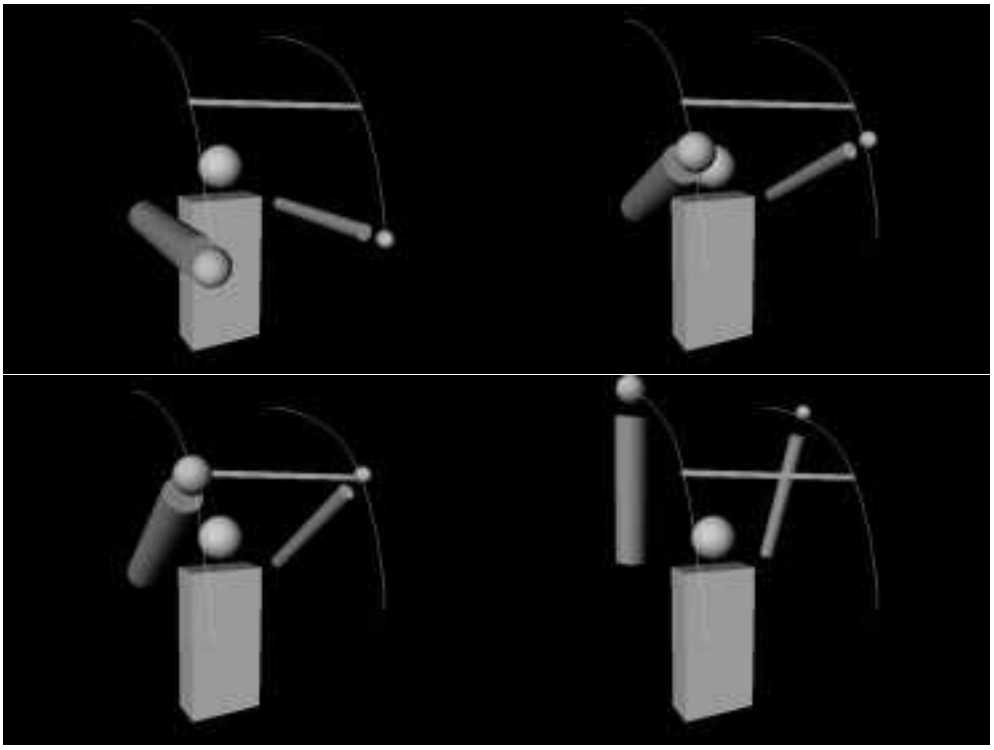


FIG. 5.5 - *Synchronisation simple*

continu pour chaque bras, et qu'alors que le bras « faible » conserve une vitesse constante, le bras « fort » ralentit progressivement pour atteindre le rendez-vous. Après celui-ci, il accélère rapidement pour retrouver sa vitesse initiale, le pourcentage d'augmentation de sa vitesse à chaque pas de temps étant ici relativement élevé.

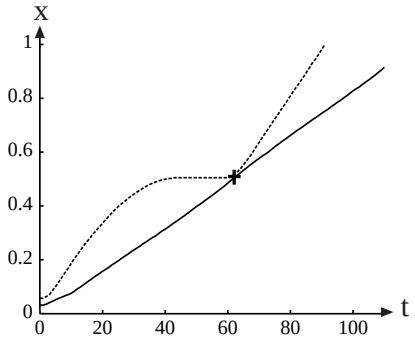


FIG. 5.6 - *Position des deux cibles, en fonction du temps.*
La ligne pointillée désigne le bras « fort », et la ligne pleine le bras « faible». Le rendez-vous est représenté par la croix.

5.4 Scénario d'une séquence d'animation

La méthode de synchronisation présentée ci-dessus peut être utilisée à une plus grande échelle, dans le but d'imposer un scénario donné à une séquence d'animation faisant intervenir un certain nombre d'objets.

La section précédente montre comment organiser la coordination de plusieurs cibles pour une seule contrainte de synchronisation. Plus généralement, si l'utilisateur anime n objets munis d'effecteurs, il peut avoir besoin de définir plusieurs de ces contraintes. Chacun de ces événements relie plusieurs cibles, et un ordre partiel sur les événements est généralement spécifié.

L'ordonnancement des contraintes de synchronisation peut alors être défini comme un réseau de Pétri, comme indiqué sur la figure 5.7. Ce réseau définit les relations de descendance entre certaines des contraintes. A un temps donné, seul un nombre limité de ces contraintes sont actives, celles dont les objectifs des contraintes « parents » dans le graphe sont déjà atteints. Sur la figure, la contrainte C_4 sera activée dès que C_2 et C_3 seront atteints.

Pour implémenter ce processus, les contraintes de synchronisation sont stockées dans une structure où chacune passe par trois stades différents :

- *En attente*, qui signifie que la contrainte n'agit pas sur les cibles qu'elle relie. Cette contrainte n'est pas considérée par le système.
- *Active*, lorsque les cibles sont en train de réaliser la contrainte.
- *Atteinte*, quand la contrainte est réalisée. Les cibles retournent alors à leur état initial (non contraintes).

Dans notre implantation une contrainte *En attente* devient *Active* dès que tous ses parents sont *Atteints*. Sur la figure 5.7, les C_i ($i = 1..4$) représentent les contraintes, les T_j ($j = 1..5$) sont les cibles impliquées pour chacunes d'entre elles, les E_k ($i = 1..3$) sont les arêtes orientées, définies par l'utilisateur qui spécifie la séquence d'événements désirée.

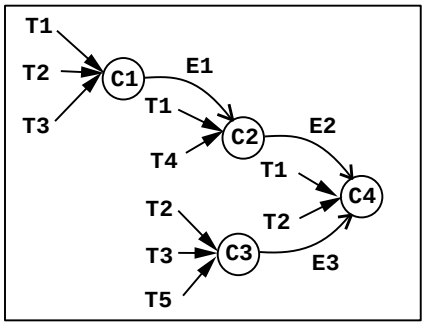


FIG. 5.7 - *Le graphe de contraintes de synchronisation.*

On peut cependant choisir toute autre règle pour l'activation des contraintes. Par exemple il peut être utile de les prendre en compte plus tôt, en décidant l'activation dès qu'un nombre suffisant de leurs parents est en mode *Atteint*, où dès qu'il en reste moins qu'un nombre donné d'*Actives*.

Résultats

Un exemple de ce type d'application est montré en figure 5.9 : trois sphères représentant des personnages simplifiés se synchronisent selon un scénario prédéfini par l'utilisateur : les sphères doivent passer deux par deux sous chacunes des trois arches. Les images présentées montrent les différentes étapes de l'animation : les deux balles claires doivent d'abord se synchroniser sous la première arche, alors que la troisième balle a un mouvement libre. Celle-ci prend ainsi une avance confortable, que la balle la plus claire doit ensuite combler pour passer la deuxième arche. Enfin, la balle sombre doit à son tour combler le retard pris, et sa vitesse sous la dernière arche lui fait achever sa trajectoire en avance. On observe une collision entre deux balles au cours de l'animation, qui va ralentir les deux objets en collision, sans affecter le scénario. Ces résultats sont montrés de façon plus détaillée sur la figure 5.8, montrant les trois parcours des cibles en fonction du temps. On voit ici que les trajectoires sont toujours C^1, ce qui signifie que les vitesses des cibles sont continues. Les croix représentent toujours les points de synchronisation, et on voit que ceux-ci sont atteints.

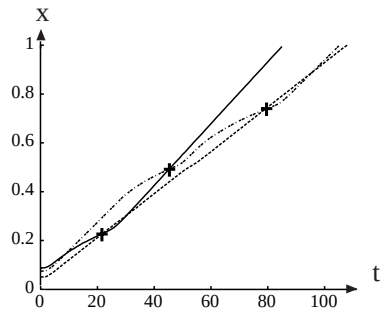


FIG. 5.8 - *Position des trois cibles en fonction du temps*

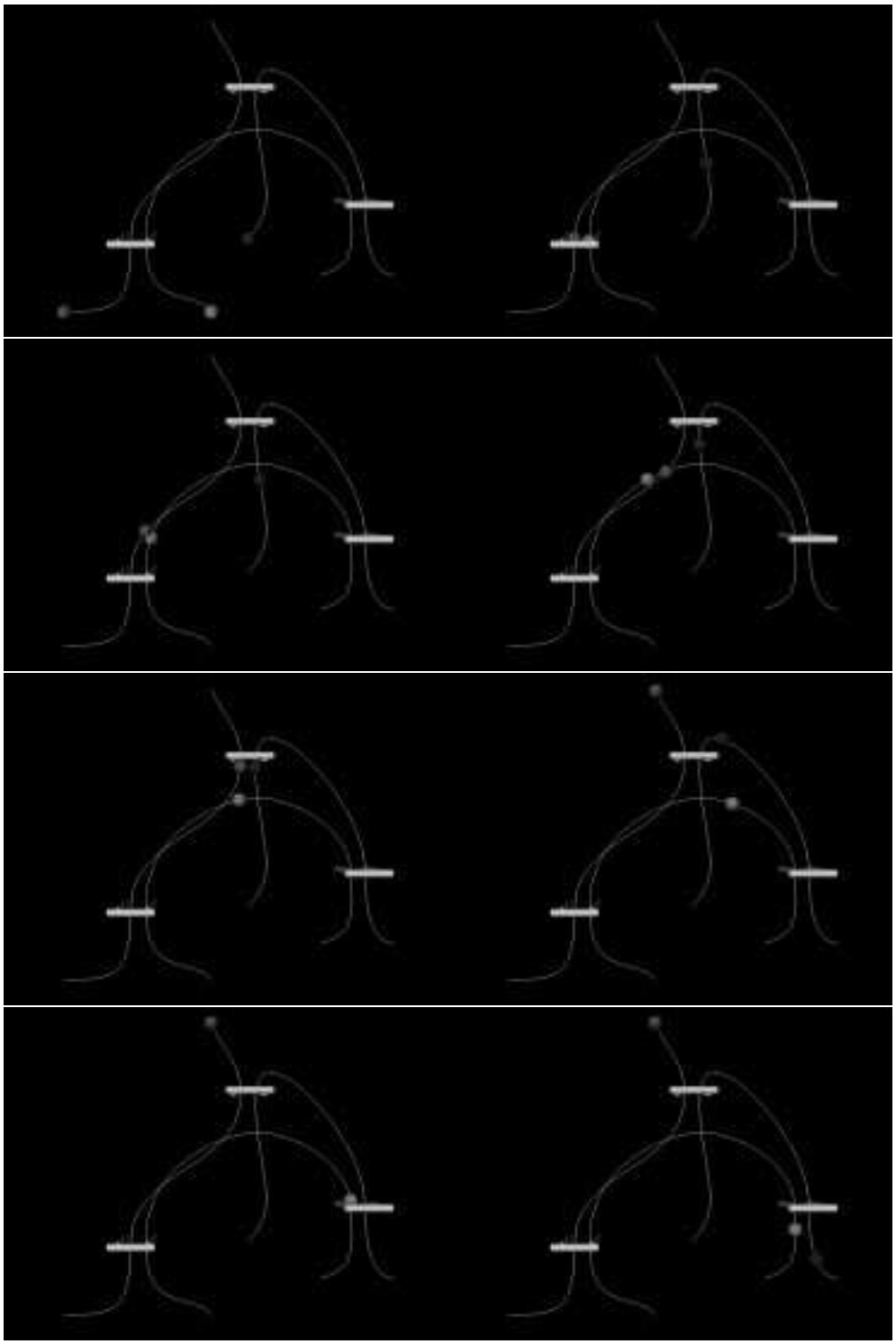


FIG. 5.9 - *Utilisation du graphe de synchronisation*

5.5 Conclusion

Ce chapitre a présenté une méthode qui vient compléter le système de suivi de cible présenté au chapitre précédent, en permettant de décrire des contraintes pour la réalisation d'un scénario. Le graphiste peut ainsi définir des séquences d'animation d'une façon à la fois précise et rapide, en suivant le principe du scénario : il définit ses événements clés et l'ordre dans lequel il veut les voir s'effectuer. Le système est alors capable de réaliser la séquence décrite, en ajustant au besoin les vitesses des objets, et en calculant automatiquement la simulation (détection et traitement automatique des collisions) en fonction de ces éléments.

Le système de contrôle et de synchronisation présenté jusqu'à présent ne résoud cependant pas tous les problèmes. Reprenons l'exemple présenté plus haut de deux personnes se serrant la main. Si, au lieu de commencer l'animation lorsqu'ils sont suffisamment proches comme nous l'avons fait, on la fait commencer lorsqu'ils sont à trente mètres l'un de l'autre, il serait ridicule qu'ils démarrent le protocole consistant à se serrer la main avant le moment ou ils sont suffisamment proches et qui dépend de leur façon d'effectuer ce geste, de leur vitesse ce jour-là et de nombreuses autres variables. Il est évident que la description d'un tel processus doit englober plus qu'un simple calendrier temporel, et doit être basé notamment sur la connaissance du comportement des sujets, domaine à la fois vaste et hors du cadre de notre approche. Ce que nous pouvons proposer en revanche pour aider l'utilisateur à régler lui-même ce genre de problème est un système de fenêtres d'activations qui décident de la zone dans laquelle un guidage peut être enclanché, dans le cas présent par exemple lorsque les deux personnes sont à portée de bras.

D'autre part, malgré tout le savoir-faire de l'utilisateur le modèle risque d'être également insuffisant, par exemple, pour guider la marche d'un personnage de synthèse, dont le mouvement paraîtrait alors très artificiel, en raison notamment de l'habitude qu'a l'oeil humain à reconnaître ce type de mouvement. En revanche, il fournit des outils réutilisables pour cette application, en les couplant avec des techniques d'apprentissage et de génération automatique de contrôleurs.

Troisième partie

Animation de personnages

Introduction

La synthèse de mouvements de marche et de course réalistes est un problème difficile qui a été le centre d'intérêt d'une grande communauté en animation par ordinateur, en biomécanique, et en robotique. Une solution de ce problème est de permettre la construction de comportements de marche pour que les acteurs de synthèse puissent être dirigés avec des primitives de haut niveau, plutôt que des primitives décrivant le mouvement de chaque articulation. La construction de tels comportements autonomes reste un problème non résolu par les techniques de dynamique existantes. Bien que le modèle que nous présentons ne résolve pas tous les problèmes, il fournit une méthode de calcul effective pour apprendre à des acteurs de synthèse à se déplacer.

La source logique d'inspiration quand on utilise un mécanisme de contrôle basé sur les senseurs et les effecteurs pour des acteurs de synthèse (aussi bien humains qu'animaux ou imaginaires) vient des solutions proposées par la nature. Malheureusement, la plupart des recherches dans ce sens conduisent à une description très fidèle et très minutieuse d'un type de marche plutôt qu'à une compréhension globale des mécanismes complexes impliqués. En même temps, il semble raisonnable de penser que les systèmes biologiques de senseurs-effecteurs sont très adaptés à la créature qui les utilise, à son type de déplacement, et à l'environnement externe habituel. En tant que tels, il devraient pouvoir fournir des indices permettant de définir des systèmes de contrôle similaires pour des acteurs de synthèse, mais ils ne contiennent en aucun cas la solution complète [BBZ91].

Le travail présenté dans cette partie est issu d'une collaboration avec le professeur Michiel van de Panne, de l'université de Toronto. Ces travaux ont fait l'objet d'une publication au « 6th Eurographics Workshop on Animation and Simulation » [vdPL95]. Le modèle que nous proposons est un processus en plusieurs étapes pour la synthèse automatique de modes de déplacements équilibrés. Ce processus tire de notre expérience de tous les jours l'utilisation de guidage pour simplifier l'apprentissage. Imaginons un parent aidant un enfant qui apprend à marcher en lui fournissant une main pour le guider. Tout comme cette main qui ne sert qu'à maintenir l'équilibre nous proposons d'introduire une aide extérieure pour apprendre la marche à des acteurs de synthèse. Un autre exemple significatif est celui des « petites roues » sur les bicyclettes, qui offrent un support externe pour simplifier l'apprentissage. Les techniques de rééducation pour les cas de démarches humaines pathologiques utilisent également des mécanismes de guidage [DSP92]. La technique d'optimisation guidée

que nous présentons utilise un couple externe pour forcer la créature à garder une posture droite durant la marche ou la course. Grâce à ce couple de rappel, le problème de l'équilibre est temporairement (et artificiellement) résolu, ce qui diminue la complexité du problème consistant à trouver les actions de contrôle adéquates pour produire le mouvement souhaité.

Une fois que la stratégie de base pour une démarche a été synthétisée, le soutien externe est fortement réduit ou éliminé lors d'une nouvelle optimisation. D'abord, le travail du couple de rappel est réduit en perturbant la stratégie de contrôle initiale pour obtenir un mouvement plus équilibré, ce qui produit un mouvement avec un petit couple résiduel. Ensuite, tout le travail du couple peut être supprimé par tranches. Ceci peut être mieux décrit par un bébé apprenant à faire son premier pas libre hors des bras de ses parents. Lorsque ce premier pas est bien appris, nos parents synthétiques « reculent d'un pas », permettant un second pas libre et ainsi de suite.

Cette partie est divisée en deux chapitres. Le chapitre 1 présente le couple de rappel et son utilisation dans le processus d'optimisation automatique. Le chapitre 2 présente les résultats obtenus pour des créatures synthétiques, en particulier un modèle humain.

Chapitre 6

Optimisation guidée

L'originalité des techniques présentées dans ce chapitre réside dans l'introduction d'un couple de rappel (CDR) qui permet de simplifier le type de problème d'optimisation rencontré en animation. Les techniques d'optimisation de contrôleurs (voir le paragraphe 3.2.3 de l'état de l'art) conduisent dans le cas de mouvements en trois dimensions à un espace de recherche considérable. Grâce à l'introduction d'un processus d'apprentissage, basé sur le CDR, la recherche de solutions est divisée en plusieurs phases, chacune portant sur un espace de recherche beaucoup plus réduit.

Il y a plusieurs facteurs qui influencent le mouvement produit en utilisant les techniques d'optimisation de paramètres. Tout d'abord, la construction d'un acteur synthétique commence avec la construction du corps de la créature et de ses effecteurs. Il reste ensuite trois décisions à prendre concernant la création du mouvement souhaité : le choix des paramètres à optimiser (ou de la représentation du contrôle), du critère de performance, et de la technique d'optimisation. Chacune de ces trois décisions affecte la difficulté du problème résultant.

Nous avons choisi ici de reprendre le modèle de van de Panne et al. développé dans [vdPKF94a,vdPKF94b] qui présente l'intérêt d'être simple à mettre en œuvre, et s'est déjà avéré efficace en dimension 2, pour le coupler avec un processus d'apprentissage. L'approche finale décrite dans ce chapitre a fait l'objet de deux implantations, l'une au sein du module d'animation développé à l'université de Toronto, et une autre au sein du module de contrôle qui fait l'objet de cette thèse. Je décrirai plus en détail cette dernière implantation, évoquant simplement les différences avec celle de Toronto dans le chapitre décrivant les résultats.

6.1 Description d'une créature articulée

Nous avons vu dans la partie II comment animer des objets afin d'obtenir un scénario particulier. Le but recherché dans cette partie est sensiblement différent, dans la mesure où nous voulons animer des créatures articulées de façon automatique, sans avoir à spécifier aucun scénario. Nous montrons ici comment nous concevons ces créatures, et comment le travail décrit précédemment pourra être réutilisé.

Une créature articulée est définie par un certain nombre d'objets - ou liens - reliés par des contraintes de position permettant de modéliser les articulations. Chaque lien est affecté des paramètres physiques usuels : masse, inertie, raideur (il s'agit ici d'objets déformables). Les contraintes utilisées sont des contraintes axiales, qui laissent donc un degré de liberté en rotation (voir figure 6.1).

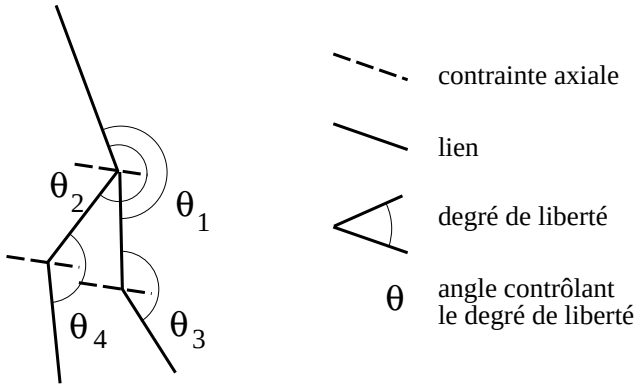


FIG. 6.1 - *Exemple de créature articulée*

Les muscles exercés aux articulations sont modélisés par des couples exercés sur un lien de l'articulation, le guidant vers une cible fixe dans le repère du second lien.

Un contrôleur est défini, comme dans le modèle introduit par van de Panne et al dans [vdPKF94a], par un graphe cyclique de positions-clés (voir figure 6.2). Chaque position-clé P_i est définie par un ensemble de valeurs θ^d, une pour chaque angle contrôlé. L'ensemble Ω_d des positions-clés forme l'espace des paramètres du contrôleur. Pour n positions-clés portant sur m degrés de liberté pour une créature articulée, on a :

$$\Omega_d = [P_1, ..., P_n] = [(\theta_1^{d1}...\theta_1^{dm}), ..., (\theta_n^{d1}...\theta_n^{dm})]$$

On donne un intervalle de temps fixe T pour réaliser un cycle, c'est-à-dire parcourir toutes les positions-clés. Ainsi, à l'instant $t \in [iT/n+kT, (i+1)T/n+kT]$ le contrôleur calculera pour chaque degré de liberté j le couple à appliquer à l'articulation pour atteindre l'angle désiré θ_i^{dj}.

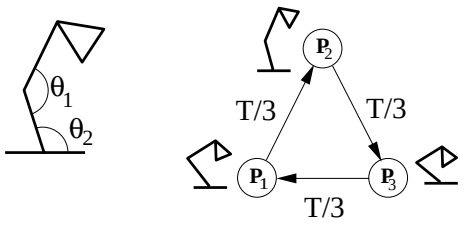


FIG. 6.2 - *Graphe de position cyclique.*
A gauche, la créature articulée d'angles internes θ_1 et θ_2. A droite, le graphe à trois états $P_1 = (\theta_1^{d1}, \theta_2^{d1})$, $P_2 = (\theta_1^{d2}, \theta_2^{d2})$ et $P_3 = (\theta_1^{d3}, \theta_2^{d3})$.

Une fois défini un contrôleur, par l' ensemble de paramètres Ω_d, il reste une

grande marge de manœuvres pour le choix de ces paramètres. Voyons maintenant quel processus d'optimisation nous utilisons pour choisir parmi ces contrôleurs lesquels conviendront le mieux.

Il y a deux facteurs qui peuvent rendre le problème difficile. Le premier est la taille de l'espace de recherche. Les méthodes telles que les algorithmes génétiques et les recuits simulés sont typiquement utilisées pour traiter cette difficulté. Un deuxième facteur, souvent trop peu pris en compte, est la fraction de l'espace de recherche occupé par les solutions utiles, et de même la capacité du critère de performance à guider la recherche de paramètres vers ce sous-ensemble. C'est pourtant le facteur qui rend si difficile la marche équilibrée pour une créature de synthèse en dimension 3. En effet, surtout pour un bipède, les mouvements équilibrés ne constituent qu'une très petite part de l'espace de recherche, rendant le problème extrêmement difficile à résoudre pour tout système d'optimisation.

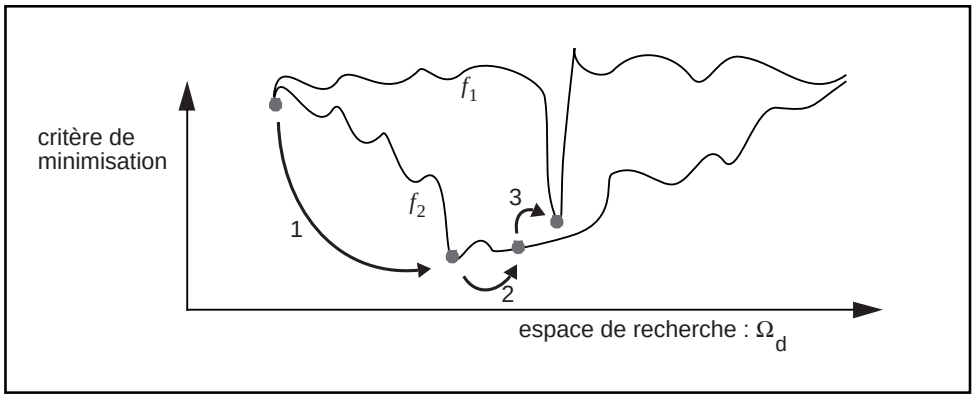


FIG. 6.3 - *Effet de la reformulation du problème d'optimisation*
Le mouvement équilibré souhaité est représenté par le minimum global de f_1. *L'utilisation du couple de rappel change l'équation du problème, représenté par* f_2. *L'effet des trois phases d'optimisation sont montrées schématiquement. La phase* 1 *trouve une solution au problème simplifié, la phase* 2 *trouve un point sur* f_2 *qui minimise le couple de rappel, et la phase* 3 *revient au problème initial sans aide extérieure.*

La difficulté peut être surmontée de deux façons. La première consiste à changer le critère de performance de manière à récompenser les progrès partiels, par exemple récompenser les créatures qui restent debout même si elles n'avancent pas. Cela peut être une tâche difficile, qui risque d'obliger l'utilisateur à apprendre comment construire un critère approprié, ce qui va à l'encontre de notre choix de laisser l'acteur virtuel faire l'apprentissage. La seconde méthode est de modifier le système contrôlé pour augmenter l'espace des solutions de façon à trouver plus rapidement une solution dans ce nouvel espace, puis de le restreindre progressivement pour revenir à l'espace initial (voir figure 6.3). Par exemple, pour apprendre à marcher à un bébé cela pourrait correspondre à commencer avec des pieds très larges et stables, puis réduire progressivement cette taille jusqu'à la normale. La technique que nous proposons évite de modi-

fier la forme des créatures animées en introduisant dans un premier temps un couple externe pour gérer les problèmes d'équilibre. Par la suite, ce couple peut être réduit ou supprimé, mais le problème sera en tout cas simplifié puisqu'on partira d'un contrôleur voisin de la solution finale recherchée.

6.2 Le couple de maintien de l'équilibre

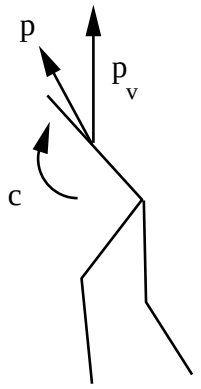


FIG. 6.4 - *Le couple de rappel*
Le couple de rappel C est exercé sur le tronc pour assurer que le vecteur de verticalité p, fixé au tronc, reste toujours proche du vecteur idéal p_v.

Le couple de rappel que nous utilisons est un couple exercé sur le torse d'une créature articulée pour le garder en position proche de la verticale, comme le montre la figure 6.4. L'utilisateur doit spécifier d'une part la « position verticale » désirée au moyen d'un vecteur p_v lié au repère du monde, d'autre part le vecteur position courante p, lié au repère du torse, qui doit rester proche de p_v (p et p_v sont tous les deux exprimés dans le repère du monde, afin de pouvoir les comparer). Cela permet des mouvements avec le torse penché en avant ou sur le coté, par exemple. Le CDR maintient le torse à proximité de la position verticale en appliquant un couple $C = k_p(p \wedge p_v) - k_d\omega$, où k_p et k_d sont des constantes de raideur et d'amortissement, et ω est la vitesse de rotation du torse. Notons que, bien que si le CDR permet le maintien du torse en position verticale, ce sont toujours les jambes qui supportent à chaque instant le poids du corps.

6.3 Étapes de l'apprentissage

Le processus d'optimisation utilisant le CDR est divisé en trois étapes, comme le montre la figure 6.5. Comme pour apprendre la bicyclette à un enfant, le but est d'abord d'apprendre les bases du mouvement avec une aide extérieure, puis de réduire cette aide ou la supprimer quand l'apprentissage avance. Comme on l'a vu plus haut, l'ensemble des paramètres à optimiser

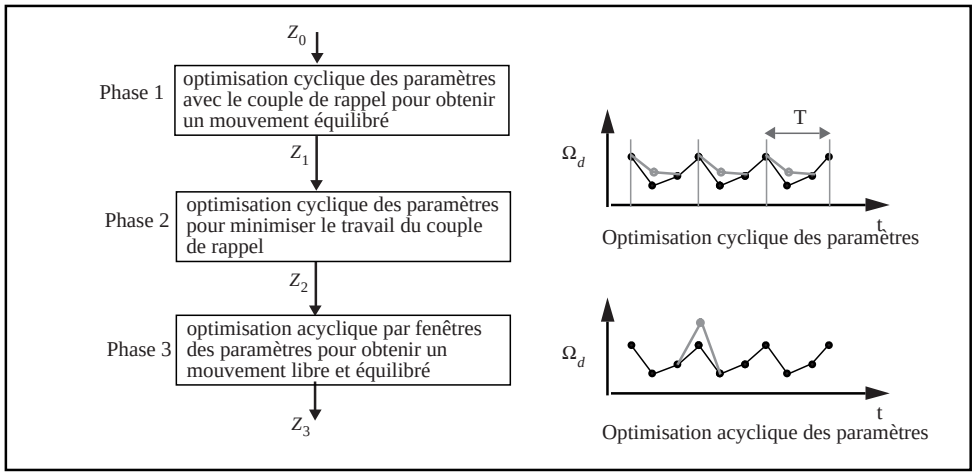


FIG. 6.5 - *Phases de l'optimisation guidée. Chaque phase assure que la phase suivante part d'un point de départ correct pour l'optimisation*

Ω_d contient $n.m$ éléments, où m est le nombre d'effecteurs à contrôler et n le nombre de positions-clés dans un cycle. L'animateur est chargé, comme une part du processus de description des objets, de définir pour chaque articulation θ_{min} et θ_{max}, qui donnent les bornes de l'espace de recherche. Ces bornes correspondent aux limites des articulations pour l'animation de personnages. Notre choix pour l'algorithme d'optimisation est une descente de gradient, encore que de nombreux autres algorithmes pourraient être utilisés. Il s'agit en fait, pour une solution donnée, de faire légèrement varier un des paramètres, choisi aléatoirement, et de voir si le mouvement obtenu est meilleur ou non. S'il est meilleur, le changement est conservé, sinon il est rejeté. En pratique, cette technique semble bien fonctionner avec le problème d'optimisation traité.

Grâce au CDR, la recherche automatique de paramètres pour produire un mouvement de marche basique devient une tâche simple. Les critères de performance utiles pour cette étape de l'optimisation sont liés à la distance parcourue par le modèle durant quelques périodes du mouvement périodique. Dans le cas de la marche humaine, il peut être pratique de construire un critère de performance qui mesure la similarité avec des données existantes sur le sujet. Un critère simple est donné par : $K = \int w_i(\theta_i - \hat{\theta}_i)^2 dt$, où $\hat{\theta}_i$ représente les données expérimentales pour l'articulation i, et w_i donne l'importance relative de chaque articulation dans le critère de performance. Si un balancement périodique du torse est nécessaire, la définition du vecteur de maintien à la verticale peut être changée cycliquement pour en tenir compte.

La deuxième phase de l'optimisation utilise toujours le CDR, mais en minimisant la dépendance du mouvement à son action. A partir des paramètres établis par la phase précédente, Ω_1, le mouvement est optimisé pour minimiser le travail du CDR, défini comme $K = \int(\mathcal{M}.\mathcal{M})dt$, où $\mathcal{M}$ est le moment du couple. Le jeu de paramètres Ω_2 résultant montre un mouvement où le CDR

joue un rôle minime, mais non complètement éliminé. Il s'agit typiquement d'un acteur de synthèse incapable de maintenir son torse en utilisant uniquement ses muscles, ce qui se manifeste par un torse excessivement rigide.

La dernière phase de l'optimisation est cette fois-ci acyclique : il est nécessaire d'optimiser chaque pas libre à la suite.

Le processus est illustré en figure 6.6. Un premier point à noter est que la contrainte de cycle sur les paramètres est supprimée, permettant ainsi des petites perturbations sur le contrôle cyclique prédominant pour tenir compte des actions de correction nécessaires pour maintenir l'équilibre. Deux fenêtres de taille fixe servent à définir la durée des essais de simulation : la fenêtre d'optimisation et la fenêtre d'évaluation. La fenêtre d'optimisation définit le sous-ensemble Ω_w de paramètres pour effectuer l'optimisation. Ce sont les paramètres affectant le mouvement immédiatement avant que l'action du CDR soit restaurée. La fenêtre d'évaluation mesure la quantité de travail K nécessaire pour restaurer une position équilibrée à la fin du passage libre du mouvement. L'optimisation consiste à minimiser K en fonction du sous-ensemble Ω_w. Les fenêtres sont déplacées dans le temps quand K a été suffisamment minimisé dans la fenêtre courante.

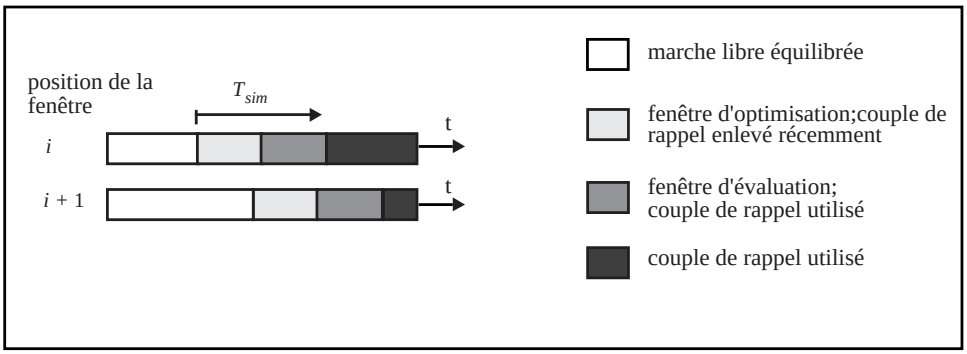


FIG. 6.6 - *Phase 3 : suppression progressive du couple de rappel*

Chapitre 7

Résultats

Nos acteurs de synthèse sont des créatures articulées de 5 et 15 membres, avec 4 et 16 effecteurs respectivement. Les simulations nécessitent de 3 à 20 secondes de temps de calcul pour chaque seconde de simulation, et chaque essai a nécessité de quatre à huit secondes de simulation. Bien que le processus d'optimisation ait été décrit en trois phases, dans la plupart des cas on obtient des résultats satisfaisants en appliquant uniquement les phases un et deux.

7.1 Robot bipède

Nous montrons ici un l'exemple d'une créature bipède composée d'un tronc et de deux jambes, les pieds étant des sphères liées aux mollets. Le modèle comporte donc 4 degrés de libertés internes, aux genous et aux hanches. Les « pieds » sphériques et le sol sont des objets déformables représentés par le modèle d'objets implicites décrit dans [Gas93]. Les pieds ont une surface de contact avec le sol très faible, en raison de leur petite dimension, ce qui rend particulièrement difficile d'arriver à un mouvement équilibré. De plus, cet exemple présente peu de degrés de liberté, ce qui rend d'autant plus difficile le contrôle dans la mesure où peu d'effecteurs peuvent affecter le mouvement.

Nous avons ici un graphe cyclique de quatre positions-clés, pour quatre degrés de liberté, soit 16 paramètres pour l'optimisation. Les positions au départ de l'optimisation sont données dans la figure 7.1. La distance parcourue dans un intervalle de temps donné reste le critère d'optimisation le plus utile trouvé.

L'effet du CDR peut être parfois visible simplement après la phase 1, à cause du mouvement déséquilibré crée, alors que le torse reste cependant toujours droit. La phase 2 de l'optimisation supprime de tels effets indésirables, résultant en des placements de pieds qui conduisent à des mouvements plus équilibrés. Les phases 1 et 2 nécessitent en général 50 à 200 essais pour arriver à des résultats raisonnables. La phase 3 est plus coûteuse, nécessitant à peu près 500 essais pour produire 5 secondes de mouvement libre équilibré. Une fenêtre d'optimisation d'un demi-pas avec une fenêtre d'évaluation d'un pas suffit dans la pratique. La figure 7.2 montre les résultats d'un mouvement de marche après les deux premières phases d'optimisation.

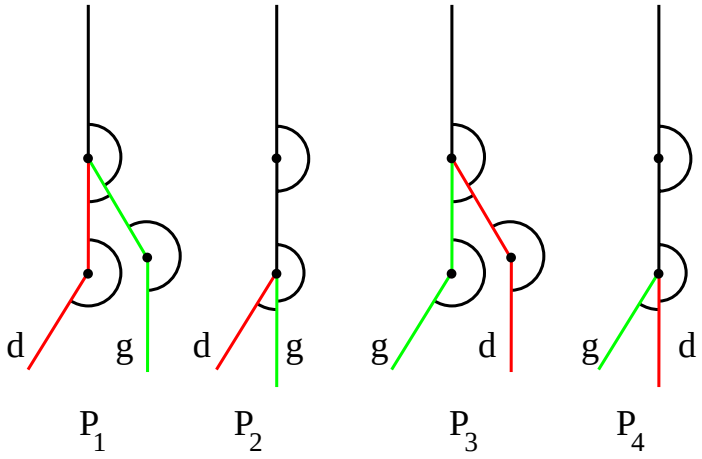


FIG. 7.1 - *Positions de départ de l'optimisation*

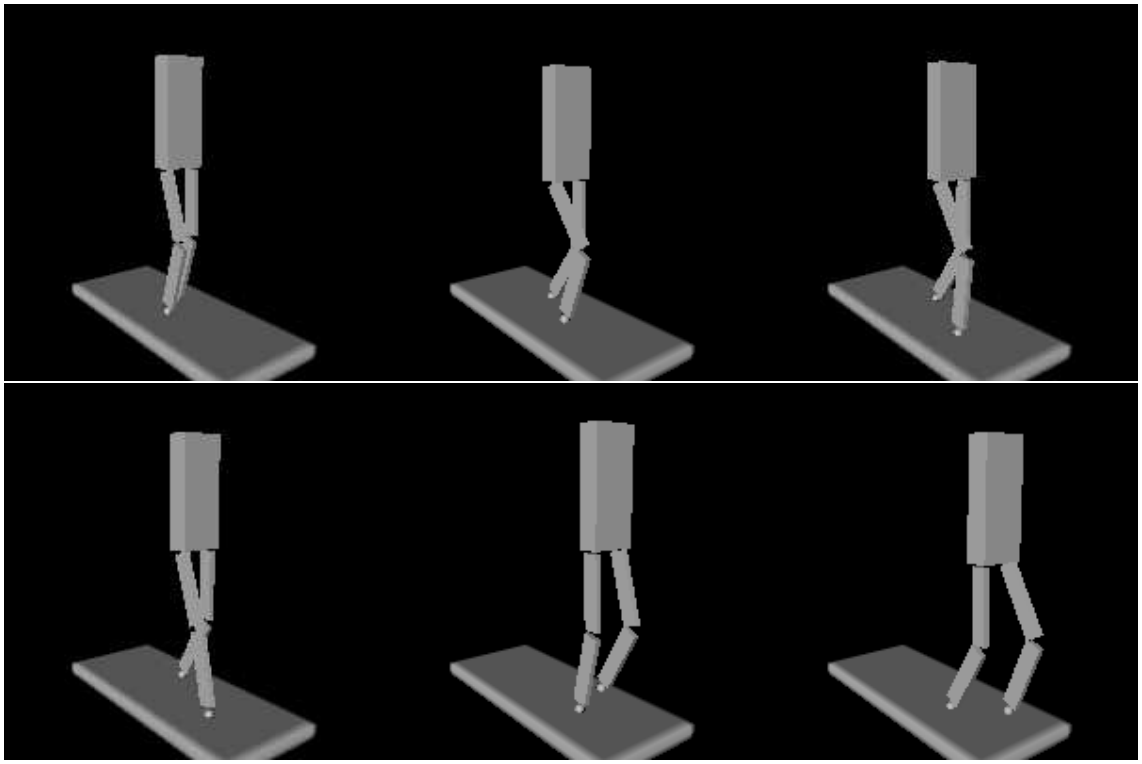


FIG. 7.2 - *Marche d'un robot bipède*

7.2 Modèle humain

Cet exemple a été réalisé par Michiel van de Panne à l'université de Toronto, en utilisant les techniques décrites précédemment. Il est cependant donné ici pour montrer que le modèle fonctionne avec deux modules de simulation distincts, traitant les objets et les collisions de façon différente. Les équations du mouvement utilisées ici sont celles de [HRS91]. Les créatures articulées sont décrites comme une arborescence, et les angles internes sont donc directement les degrés de liberté du système. Les objets sont ici rigides, et les contacts au sol sont modélisés en utilisant un système de ressorts modifié pour permettre la simulation de la friction et du glissement.

L'exemple présente un modèle humain tri-dimensionnel de proportions et de paramètres physiques associés réalistes [NAS78]. Le modèle utilisé est montré en figure 7.4. Bien que le personnage possède ici 16 degrés de liberté, 8 peuvent être traités de manière largement passive, en laissant seulement 8 à régler pour le contrôle. Des conditions de symétrie sont utilisées ainsi que la représentation cyclique de l'ensemble de paramètres pour diminuer de moitié la taille effective de cet ensemble. Une discrétisation cyclique de 8 positions-clés a été utilisée ici.

La marche, la course et les bonds ont été modélisés en utilisant les phases 1 et 2. Les mouvements ont été différentiés en utilisant des ensembles de paramètres initiaux différents en phase 1, et des critères de performance différents. L'ensemble de paramètres initial pour la marche consiste en la définition de 4 positions-clés montrant les jambes se levant de façon lente et alternée. Ces positions-clés initiales faisaient marcher le personnage sur place quand le CDR était appliqué. Le mouvement de course a été initialisé avec un ensemble de positions plus éloignées, utilisant les chevilles pour propulser le corps en l'air, créant un mouvement de saut sur place avec alternement des jambes. Pour le mouvement de bonds les paramètres sont choisis de façon à permettre un sautillement sur place. Les trois mouvements ont été optimisés par rapport à la vitesse et la similarité avec un mouvement humain connu [Inm81] (phase 1), et pour l'équilibre (phase 2). La trajectoire du point milieu du bassin est montrée pour les différents mouvements en figure 7.3.

Les mouvements résultants sont suffisamment convaincants quand il s'agit d'une représentation par lignes (figure 7.5), mais deviennent moins convaincants quand un modèle géométrique est utilisé pour le rendu. Étant donné que nous sommes sans aucun doute plus accoutumés à voir les êtres humains se mouvoir, le mouvement humain est probablement le plus difficile à animer. L'image montrée figure 7.4 montre le modèle « habillé » dans une attitude de course. Il est à noter que l'un des avantages généraux des techniques d'optimisation de paramètres sur les techniques descriptives est qu'elles offrent un traitement unifié pour la création de démarches différentes, comme la marche, la course, ou le saut.

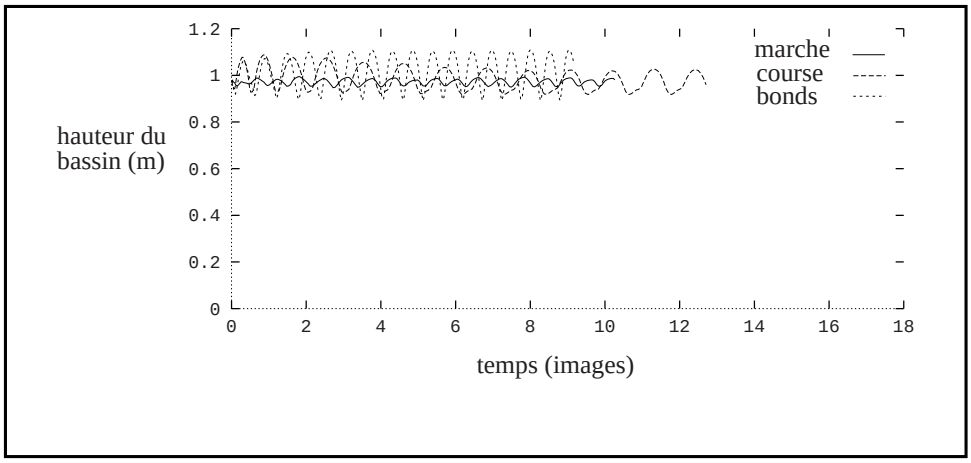


FIG. 7.3 - *Hauteur moyenne du bassin au cours des mouvements de marche, course et bonds.*

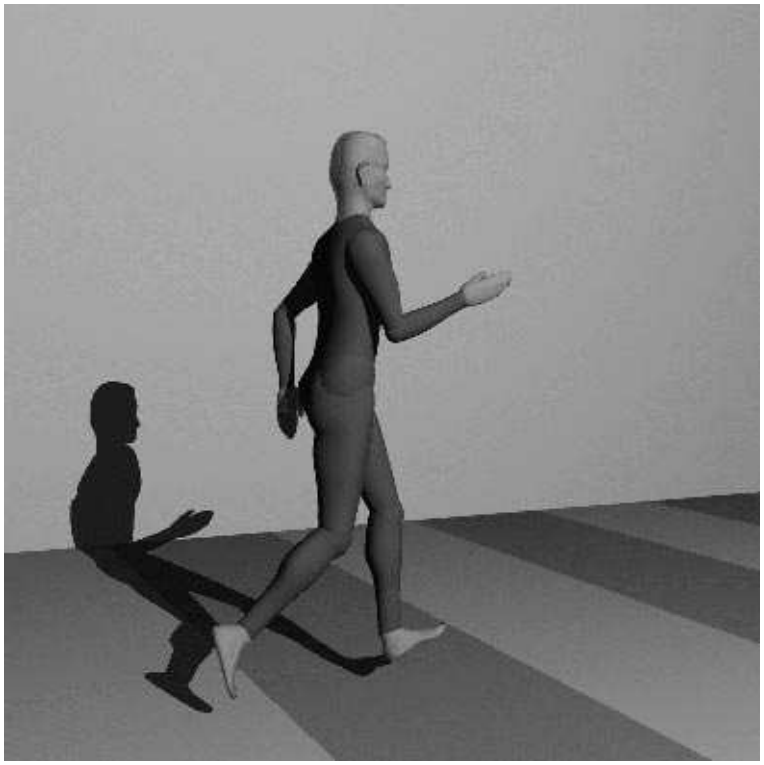


FIG. 7.4 - *Modèle humain de 16 degrés de liberté internes*

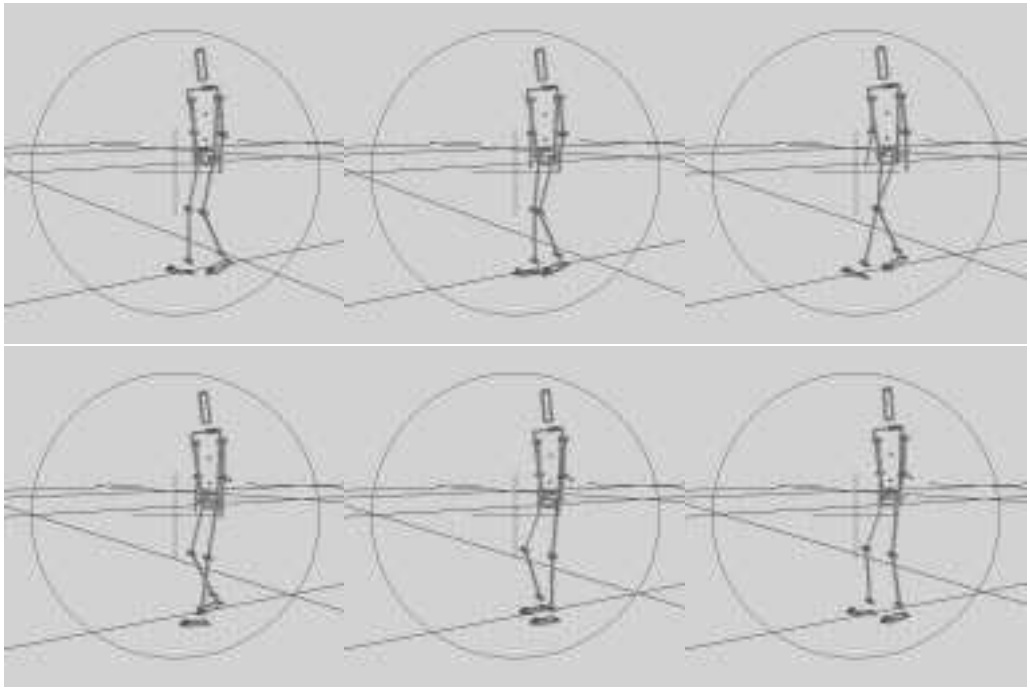


FIG. 7.5 - *Marche d'un personnage humain*

Conclusion

Le modèle que nous avons présenté dans ce chapitre s'appuie sur l'observation de techniques réelles d'apprentissage. En utilisant un couple de rappel qui permet aux créatures animées de maintenir leur équilibre, nous reprenons l'idée couramment utilisée de s'appuyer sur une aide externe dans les premières phases de l'apprentissage, comme pour les vélos munis de roulettes par exemple. Notre système d'optimisation gagne ainsi en rapidité en divisant le processus en plusieurs phases plus simples à réaliser.

L'optimisation proprement dite est décomposée en trois phases. Lors de la première phase, la créature articulée « apprend à avancer » à l'aide d'un couple de rappel, dont la puissance n'est pas limitée, au cours d'un processus cyclique. Pendant la phase deux ce couple est réduit pour permettre une démarche plus équilibrée et faisant un usage beaucoup plus restreint de cette aide externe, lors d'un processus à nouveau cyclique. Enfin, la phase trois supprime totalement l'aide externe en optimisant chaque pas effectué pour obtenir un mouvement libre.

Il est par ailleurs intéressant de garder à l'esprit que l'utilisation des seules phases un et deux de l'algorithme permettent de réaliser des animations suffisamment réalistes, y compris pour des mouvements qui sont en fait physiquement impossibles, ce qui peut arriver quand l'utilisateur n'est pas expert du domaine.

Conclusion générale

Nous avons abordé dans cette thèse le problème du contrôle de l'animation en image de synthèse, et tenté d'apporter des solutions originales permettant de coupler l'utilisation de modèles physiques avec le contrôle du mouvement par suivi de trajectoire.

La première partie passe en revue les différentes techniques d'animation existantes, et fournit les motivations pour ce travail. On a vu en particulier les différents points forts et points faibles des méthodes de contrôle de l'animation par modèles physiques, qu'ils fonctionnent par optimisation, ou par la définition de contrôleurs calculant l'action à exercer sur les objets en fonction de l'état du système. On a observé dans la plupart de ces modèles l'absence de prise en compte des actions externes exercées sur les objets contrôlés, en particulier en cas de collisions ou de contacts prolongés. D'autre part les modèles d'optimisation et de génération automatique de contrôleurs se sont révélés prometteurs mais difficiles à utiliser pour des scènes complexes en raison de l'espace de recherche considérable qu'ils ont à traiter, surtout pour les modèles en dimension 3.

Dans la partie II nous présentons un modèle original de contrôle basé sur le principe des effecteurs, permettant de spécifier une animation à un haut niveau de contrôle. Le principe repose sur la mise en place de trajectoires de guidage spécifiées par l'utilisateur pour les objets devant être guidés, couplées avec un moteur d'animation permettant de prendre en compte, à la fois ces trajectoires et les paramètres physiques des objets, traitant ainsi le problème des collisions en adaptant la trajectoire effectivement suivie par l'objet en fonction des interventions extérieures durant l'animation. La méthode consiste à associer les objets à des effecteurs produisant des forces (ou des couples) permettant de suivre une position (ou une orientation) cible se mouvant sur le chemin défini par l'utilisateur. La cible règle sa vitesse en fonction du déplacement de l'objet et des événements externes (collisions, contacts) qui surviennent durant l'animation.

Nous avons ensuite mis en place un système permettant la définition de scénarios évolués par l'introduction d'événements-clés qui doivent être résolus indépendamment du temps. Ces événements sont présentés sous forme de contraintes temporelles qui relient les cibles de plusieurs objets guidés. La vitesse de progression des cibles, et donc celle des objets, est modifiée en fonction du retard relatif des objets les uns par rapport aux autres, afin que tous les objets se présentent en même temps aux positions définissant la contrainte. Les

événements-clés peuvent être hiérarchisés par un graphe de synchronisation, afin de réaliser des mouvements plus complexes. La résolution des contraintes se fait alors en fonction des contraintes couramment actives, les autres pouvant être en attente de la fin d'un événement.

La dernière partie présente une application plus spécifique, c'est-à-dire l'animation de marche de personnages. Ce travail, réalisé en collaboration avec le professeur van de Panne de l'université de Toronto, utilise certaines des notions développées dans la partie II, en particulier l'idée d'introduire un couple externe - dit de rappel - pour guider la créature animée. La modélisation des muscles utilise également une version simplifiée du module de suivi de trajectoire, puisqu'ici les trajectoires sont réduites à un point.

Nous proposons de simplifier le processus de création de contrôleurs pour l'animation de personnages par l'introduction d'une phase d'apprentissage, guidée par le couple de rappel. Ce couple permet de maintenir l'équilibre d'une créature articulée pendant une première phase d'apprentissage de la marche. Nous réduisons ensuite ce couple par une phase d'optimisation, afin d'obtenir un mouvement visuellement satisfaisant, doté d'un couple de rappel indétectable à la vision de l'animation, et qui reste cependant cyclique, ce qui permet de définir l'animation par peu de paramètres. Une dernière phase d'optimisation permet éventuellement de supprimer tout couple de rappel, permettant ainsi l'obtention d'un mouvement parfaitement libre et équilibré.

Perspectives

Comme nous l'avons vu dans les différentes parties de ce document, les modèles proposés dans cette thèse ne permettent pas de résoudre tous les problèmes, et ouvrent peut-être la voie à autant de questions qu'ils fournissent de réponses. Le but ici n'est pas de résoudre ces problèmes ni forcément apporter des éléments de réponses, mais d'ouvrir la réflexion aux différentes possibilités d'extensions ou d'améliorations des modèles proposés.

Dans le chapitre 4, nous avons vu comment sous certaines réserves l'optimisation des paramètres au cours de l'animation pourrait conduire à un choix plus systématique dans le compromis entre le réalisme de l'animation et la précision du suivi de trajectoire. Si cette optimisation s'avère une tâche ardue, et que le choix des paramètres relativement intuitifs que nous proposons nous semble un bon moyen d'obtenir les effets voulus en un temps raisonnable, l'optimisation présente néanmoins toujours l'attraît d'une automatisation plus complète.

Le chapitre 5 propose une approche à base de contraintes de synchronisation pour permettre la spécification d'animations plus évoluées. Cependant le problème de la synchronisation est beaucoup plus vaste que la simple définition d'événements temporels, qu'ils soient discrets ou continus. Les problèmes impliqués ici sont beaucoup plus vaste que simplement temporels, et impliquent

entre autres des modèles comportementaux pouvant dépendre fortement du cas particulier traité. Si ce sujet semble digne d'être étudié de façon beaucoup plus approfondie, il est tout aussi clair que son étendue ne permette pas de résolution rapide des questions qu'il soulève, et de toutes façons n'entre pas dans le cadre que nous nous sommes fixés.

Enfin, le modèle d'animation de personnages, s'il permet d'obtenir des animations convaincantes, même en conservant une aide extérieure limitée, est en revanche pour une animation sans aucune intervention d'un coût proportionnel à sa durée, ce qui le rend peu séduisant en particulier pour des animations longues. Parmi les différentes méthodes envisageables pour résoudre ce problème on peut citer la possibilité d'ajout de capteurs permettant une évaluation automatique du déséquilibre engendré à chaque pas, et la création une fois pour toutes d'un processus de réajustement automatique au pas suivant, de façon à avoir de nouveau un contrôleur cyclique, et donc utilisable quelle que soit la durée de l'animation. Il est d'autre part souhaitable de développer des critères de sélection permettant de récompenser l'effort d'un modèle pour tourner, afin de pouvoir engendrer automatiquement des contrôleurs de marche capables d'aller n'importe quelle direction. Le couplage d'un tel contrôleur avec un module de suivi de trajectoire inspiré de celui présenté dans le chapitre 4, enfin, permettrait d'obtenir une créature capable de suivre un scénario complet, franchissant ainsi un grand pas dans le domaine de l'animation de personnages.

Quatrième partie

Annexes

Annexe A

Implantation

Tout le travail présenté dans cette thèse a été implanté au sein du logiciel *Fabule*, qui constitue la plate-forme développée et utilisée par le groupe animation de l'équipe *iMAGIS*. Avant de décrire plus en détail l'implantation des modules propres au contrôle, voyons rapidement quelles sont les caractéristiques de ce logiciel.

A.1 Le logiciel Fabule

La philosophie de *Fabule* est d'avoir une architecture ouverte permettant à tous ses utilisateurs d'ajouter ses développements le plus facilement possible. Cela est facilité par la structure même du programme, qui présente les caractéristiques suivantes :

- une architecture modulaire, soutenue par la définition de classes $C++$;
- un langage interprété, *Tcl*, utilisé pour la boucle principale d'animation, pour permettre l'insertion, le test et le prototypage de tout nouveau module;
- l'interface fournie par la librairie graphique *Inventor*, qui facilite la spécification d'animations et la visualisation des différents paramètres du modèle;
- la présence de pages de manuel utilisateur et d'exemples simples et commentés pour chaque module.

L'architecture du logiciel est centrée autour de la notion de classe, chaque classe possédant un certain nombre de caractéristiques propres. Les deux classes primordiales dans cette architecture sont celles qui modélisent les deux aspects d'un objet physique donné :

- *FaSolid*, qui permet de définir le modèle géométrique de l'objet : sa forme, sa couleur, s'il est rigide ou déformable, en un mot sa structure externe;

– *FaGenerator*, qui correspond au modèle dynamique associé, qu'il s'agisse d'un objet inanimé (mur, sol etc...) ou animé comme à peu près tout le reste, et qui peut être alors actif ou inerte. Cet attribut peut être vu comme le repère local associé à l'objet.

A chacune de ces deux classes sont attribués un certain nombre de caractéristiques qui déterminent le comportement des objets au cours de la simulation.

Bien qu'il soit théoriquement possible dans *Fabule* de mélanger n'importe quel types de moteurs d'animation : simulation de modèles physiques, modèles descriptifs ou comportementaux, aucune assurance n'est alors fournie que la séquence ainsi définie est réalisable, et nous nous limiterons ici à la description des modèles utilisant la simulation physique.

La production d'une animation peut se décomposer en les étapes suivantes :

– modélisation géométrique des objets;

– spécification des caractéristiques physiques des objets;

– animation, simulation, et éventuellement réglage du modèle;

– production d'images ou de fichiers destinés au calcul du rendu des images;

– Traitement des images en vue de produire un film.

Voyons plus en détail comment s'organisent les points deux et trois de cette séquence.

A.2 La modélisation

La modélisation d'une séquence d'animation se fait au moyen d'un script écrit en langage *Tcl*, en consiste en la défiintion de tous les objets de la scène et, pour chacun, son mode de comportement : objet immobile ou animé, actif ou inerte, isolé ou partie d'une structure articulée etc... Chacun de ces attributs correspond à une classe attachée au générateur de l'objet, *FaGenerator*.

Dans l'état actuel du logiciel, seulement certains modules sont munis d'une interface permettant la définition de ce script de façon interactive. Il est donc toujours nécessaire d'écrire une partie des informations requises pour la simulation au moyen de l'édition d'un fichier de commandes *Tcl*. Il est bien sûr possible, et quelquefois préférable, d'écrire entièrement ce fichier à la main.

A.3 Le module de simulation

Dans le cas d'objets rigides ou peu déformables, comme tous ceux utilisés dans les exemples présentés dans cette thèse, on peut considérer en première approximation que les caractéristiques physiques de l'objet - masse, raideur, matrice d'inertie ... - restent inchangées au cours du temps. La boucle de simulation peut alors être décomposée en trois étapes :

– Animation des objets isolés;

– Traitement des contraintes;

– Traitement des collisions et contacts entre objets.

L'animation des objets isolés est simplement un module d'intégration des équations de la dynamique, par le schéma de Newton présenté tout au début de l'état de l'art.

Dans le cadre des exemples développés dans cette thèse, le module de résolution des contraintes fait appel au modèle de « contraintes de déplacement » présenté dans [GG94], et décrit dans l'état de l'art.

Enfin, le module de traitement des collisions et contacts entre objets utilise le modèle d'objets déformables définis par le matériau implicite présenté dans [Gas93].

A.4 Implantation du module de contrôle

L'ajout à la structure existante des modèles de contrôle décrits dans cette thèse a consisté à ajouter un module de contrôle dans la boucle de simulation, juste avant les trois étapes de la simulation. Le modèle de base présenté au chapitre 4 est implanté sous la forme de trois classes $FaTarget$, $FaActuator$ et $FaPathControl$, auquel vient s'ajouter une quatrième classe $FaSynchroGoal$ pour la définition de contraintes de synchronisation, et enfin la classe $FaSetOfGoals$ pour la spécification de graphes de synchronisation.

$FaPathControl$ est la classe associée au $FaGenerator$ et c'est elle qui gère l'éxécution des différentes phases du contrôle :

– vérifier l'état du graphe de contraintes ($FaSetOfGoals$);

– vérifier l'état de chaque contrainte de synchronisation et en déduire la vitesse courante de chaque cible ($FaSynchroGoal$);

– déplacer chaque cible à sa nouvelle position ($FaTarget$);

– calculer les forces de l'effecteur ($FaActuator$).

La spécification d'une séquence d'animation se fait comme précédemment, en ajoutant aux définitions les attributs présentés ci-dessus. Une interface complète a été réalisée pour ce module, permettant de tout définir, depuis les positions-clés jusqu'aux événements de synchronisation. Elle se présente sous la forme d'une fenêtre 3D pour le positionnement des positions-clés, et de plusieurs fenêtres de dialogue permettant de régler les paramètres du modèle et de connecter entre eux les différents éléments du $FaPathControl$. En particulier, les paramètres α, β et d_{base} peuvent ou bien être donnés directement ou bien déduits des variables plus intuitives v_{moy}, B et d_{base} déjà présentées.

A.5 Marche de personnages articulés

L'ensemble de la partie III a fait l'objet d'un développement autour d'un exemple particulier, prouvant la validité du modèle. Cet exemple est constitué, en plus des définitions habituelles en *Tcl*, d'un certain nombre de procédures plus évoluées permettant de réaliser les différentes phases de l'algorithme d'optimisation. Elles permettent :

- d'engendrer, une fois le fichier courant défini et les paramètres à optimiser précisés, le fichier particulier à chaque simulation;
- de tester la chute de la créature, auquel cas les paramètres sont modifiés et la simulation relancée;
- d'évaluer les critères d'optimisation utilisés, à savoir la distance parcourue tout d'abord, puis la puissance utilisée, et de déterminer le progrès ou non par rapport au test précédent;
- de s'arrêter au terme d'un nombre fixé de simulations.

Malheureusement, la rédaction de cet exemple ne permet pas de généraliser suffisamment les procédures impliquées pour en faire un module générique, utilisable et intégré dans *Fabule*.

Annexe B

Quaternions

B.1 Définitions et propriétés des quaternions

Les quaternions sont des éléments de $\mathcal{R}^4$. Ils forment un groupe multiplicatif non commutatif. Un quaternion est composé d'une partie réelle et d'une partie imaginaire. On peut le noter $q = (s, \vec{v})$, où $s \in \mathcal{R}$ est la partie réelle, et $\vec{v} \in \mathcal{R}^3$ la partie imaginaire.

Soient $q = (s, \vec{v}), q^{'} = (s^{'}, \vec{v}^{'})$ deux quaternions, et $q_0 = (0, \vec{v})$ un quaternion de partie réelle nulle. On a les propriétés suivantes :

- Somme : $q + q^{'} = (s + s^{'}, \vec{v} + \vec{v}^{'}])$
- Produit : $q.q^{'} = (s.s^{'} - \vec{v}.\vec{v}^{'},\ s.\vec{v}^{'} + s^{'}.\vec{v} + \vec{v} \wedge \vec{v}^{'})$
- Inverse ou conjugué : $q^{-1} = (s, -v)$
- Inverse du produit : $(q.q^{'})^{-1} = q^{-1}.q^{'-1}$
- Norme : $||q||^2 = s^2 + \vec{v}.\vec{v}$
- Exponentielle d'un quaternion de partie réelle nulle [Han93] :
  $exp(q_0) = \left(\cos ||\vec{v}||, \frac{\vec{v}}{||\vec{v}||} \sin ||\vec{v}||\right)$
- Logarithme (dans $\mathcal{R}^3$) [Han93] : $\log q = \frac{\vec{v}}{||\vec{v}||} \arccos(s + k\pi),\ k \in \mathcal{N}$
- Puissance : si $q = (\cos \frac{\theta}{2}, \sin \frac{\theta}{2}.\vec{v})$ alors on a : $q^n = (\cos \frac{n\theta}{2}, \sin \frac{n\theta}{2}.\vec{v})$

Ces propriétés font des quaternions une représentation pratique et compacte pour les rotations, comme on va le voir tout de suite.

B.2 Quaternions et rotations

Le sous-groupe des quaternions unitaires (de norme 1) permet de représenter les rotations de la façon suivante : soit une rotation d'angle θ selon l'axe de vecteur directeur unitaire $\vec{u}$, et R sa matrice. On a, pour un vecteur $\vec{v}$ quelconque :

$$R.\vec{v} = q.(0,\vec{v}).q^{-1}, \quad \text{avec} \quad q = \left(\cos\frac{\theta}{2}, \sin\frac{\theta}{2}.\vec{v}\right)$$

Le produit de deux rotations s'obtient donc par un produit de quaternions, et la rotation résultante est représentée par le quaternion produit :

$$R_q[R_{q'}.\vec{v}] = q.(q'.(0,\vec{v}).q'^{-1}).q^{-1} = (q.q').(0,\vec{v}).(q.q')^{-1} = R_{q.q'}.\vec{v}$$

Voyons maintenant comment on peut faire une interpolation linéaire entre deux rotations à l'aide des quaternions.

Soient q et q' deux quaternions, et u le paramètre d'interpolation, compris entre 0 et 1. On définit la formule d'interpolation linéaire sphérique (*slerp*, pour *s*pherical *l*inear int*erp*olation) par :

$$q_u = slerp(q, q'; u) = \frac{\sin(1-u)\theta}{\sin\theta} q + \frac{\sin u\theta}{\sin\theta} q',$$

où $\cos\theta$ est défini par le produit scalaire des quaternions : $q.q' = \cos\theta$.

Annexe C

Courbes splines

L'utilisateur, pour définir une trajectoire appliquée à un objet, a besoin de techniques permettant, à partir d'un nombre de données limité, de définir un chemin correspondant à ses attentes. Il doit être possible, ou bien de définir un chemin le plus lisse possible, ou bien d'introduire des discontinuités, et cela au moyen de paramètres de contrôle simples.

Généralement une séquence d'animation comporte un certain nombre d'événements clés, qui sont représentatifs des différentes étapes du mouvement. C'est pourquoi les techniques de spécification de trajectoire s'appuient sur la définition de positions-clés, et utilisent des techniques d'interpolation entre ces positions.

L'interpolation linéaire entre positions est à écarter d'emblée, parce qu'elle ne produit pas de trajectoire $\mathcal{C}^1$. De même, l'idée d'interpoler le chemin entre n points par un polynôme de degré $n-1$ (polynôme de Lagrange) est à écarter, en raison du manque de stabilité de ces polynômes. Une gamme de courbes d'interpolations répond en revanche sur de nombreux points aux attentes des utilisateurs : les courbes splines.

C.1 Splines d'interpolation

Les courbes splines cubiques sont des courbes polynômiales de degré trois définies à l'aide d'un certain nombre de points, appelés points de contrôle. Les splines d'interpolation, utilisées dans toute la suite, sont des courbes qui passent par les points de contrôle, alors que les splines d'approximation se contentent d'approximer ces points, sans y passer. Nous utiliserons les notations suivantes :

- Points de contrôle : $P_i = (x_i, y_i, z_i), 0 \leq i \leq m$
- Courbe spline : $Q(u), u \in [0, m]$
- La courbe spline passe par les points de contrôle aux valeurs entières : on a $Q(i) = P_i$.
- Soit $Q_i(u)(0 \leq u \leq 1)$ le $i - eme$ segment de la courbe. On a :
  $Q_i(u) = Q(u + i)$, et donc $Q_i(0) = P_i$ et $Q_i(1) = P_{i+1}$.

On utilise généralement la notation matricielle pour décrire une courbe spline :

$$Q_i(u) = \left[\begin{array}{cccc} u^3 & u^2 & u & 1 \end{array} \right] M \left[\begin{array}{c} P_i \\ P_{i+1} \\ P_{i+2} \\ P_{i+3} \end{array} \right]$$

Les différentes familles de splines sont caractérisées par la matrice M.

C.1.1 Les cardinal-splines

Les cardinal-splines sont définies par les trois équations :

$$\begin{array}{rcl} Q_i(0) & = & P_i \\ Q_i(1) & = & P_{i+1} \\ Q_i^{'}(0) & = & c(P_{i+1} - P_{i-1}) \end{array}$$

soit, sous forme matricielle :

$$M_c = \left[\begin{array}{cccc} -c & 2-c & c-2 & 0 \\ 2c & c-3 & 3-2c & -c \\ -c & 0 & c & 0 \\ 0 & 1 & 0 & 0 \end{array} \right]$$

Les Catmull-Rom splines sont les cardinal-splines pour $c = 0.5$. On a alors : $Q_i^{'}(0) = \frac{1}{2}[P_{i+1} - P_{i-1}]$. La tangente en un point a donc pour pente le segment joignant le point précédent au point suivant.

C.1.2 Splines et polynômes de Hermite

Les cardinal-splines présentées ci-dessus présentent un inconvénient majeur, dû au fait que le coefficient c est choisi une fois pour toutes, autrement dit on ne peut pas modifier les caractéristiques de la courbe en fonction des points de contrôle.

Kochanek et Bartels [KB84] proposent des splines contrôlables par trois paramètres : continuité, tension et raideur. Pour cela, ils utilisent la base des polynômes de Hermite :

$$\begin{array}{rcl} f_1(x) & = & 2x^3 + 3x^2 + 1 \\ f_2(x) & = & -2x^3 + 3x^2 \\ f_3(x) & = & x^3 + 2x^2 + x \\ f_4(x) & = & x^3 - x^2 \end{array}$$

dont la propriété remarquable porte sur les valeurs des polynômes et de leurs dérivées en 0 et 1 :

$$\begin{array}{llll} f_1(0) = 1 & f_i(0) = 0 & \text{pour} & i = 2,3,4 \\ f_2(1) = 1 & f_i(1) = 0 & \text{pour} & i = 1,3,4 \\ f_3^{'}(0) = 1 & f_i^{'}(0) = 0 & \text{pour} & i = 1,2,4 \\ f_4^{'}(1) = 1 & f_i^{'}(1) = 0 & \text{pour} & i = 1,2,3 \end{array}$$

On peut ainsi considérer la courbe $Q_i(u)$ comme une combinaison linéaire de ces polynômes, pondérés par les positions et vecteurs tangents :

$$Q_i(u) = P_i f_i(u) + P_{i+1} f_2(u) + DD_i f_3(u) + DA_i f_4(u)$$

où DD_i et DA_i désignent en fait les tangentes de départ et d'arrivée de la courbe Q_i. Notons que grâce à cette formulation nous n'avons plus nécessairement $DA_i = DD_{i+1}$, ce qui signifie qu'il peut y avoir des discontinuités de la dérivée dans la courbe d'interpolation. L'équation devient sous forme matricielle :

$$Q_i(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_i \\ P_{i+1} \\ DD_i \\ DA_i \end{bmatrix}$$

Kochanek et Bartels expriment ensuite DD_i et DA_i en fonction des P_i au moyen de trois paramètres, nommés tension t, continuité c et biais b :

$$DD_i = \frac{(1-t)(1+c)(1+b)}{2}(P_i - P_{i-1}) + \frac{(1-t)(1-c)(1-b)}{2}(P_{i+1} - P_i)$$

$$DA_i = \frac{(1-t)(1-c)(1+b)}{2}(P_i - P_{i-1}) + \frac{(1-t)(1+c)(1-b)}{2}(P_{i+1} - P_i)$$

avec $t, c, b \in [-\infty, +\infty]$. Il est à noter que pour des valeurs de t, c, b en dehors de $[-1, 1]$ on peut avoir des comportements surprenants, comme des retournements de tangente par exemple. En réglant ces paramètres pour chacun des points de contrôle on peut obtenir à peu près toutes les variations souhaitables sur le chemin (voir figure C.1).

C.1.3 Interpolation d'orientations

L'interpolation entre orientations utilise les quaternions (voir annexe B).

Gascuel et Lyon [GL95] on étendu la technique de Kochanek et Bartel au cas des rotations, en utilisant les logarithmes de quaternions définis par Hanotaux dans [Han93].

Le système matriciel devient alors :

$$Q_i(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \log Q_i \\ \log Q_{i+1} \\ \log DD_i \\ \log DA_i \end{bmatrix}$$

où Q_i et Q_{i+1} sont des quaternions (voir annexe B), avec $Q_i = (s, \vec{v})$:
$\log Q_i = \frac{\vec{v}}{||\vec{v}||}(\arccos(s + k\pi))$. k est choisi de façon à emprunter le plus court chemin.

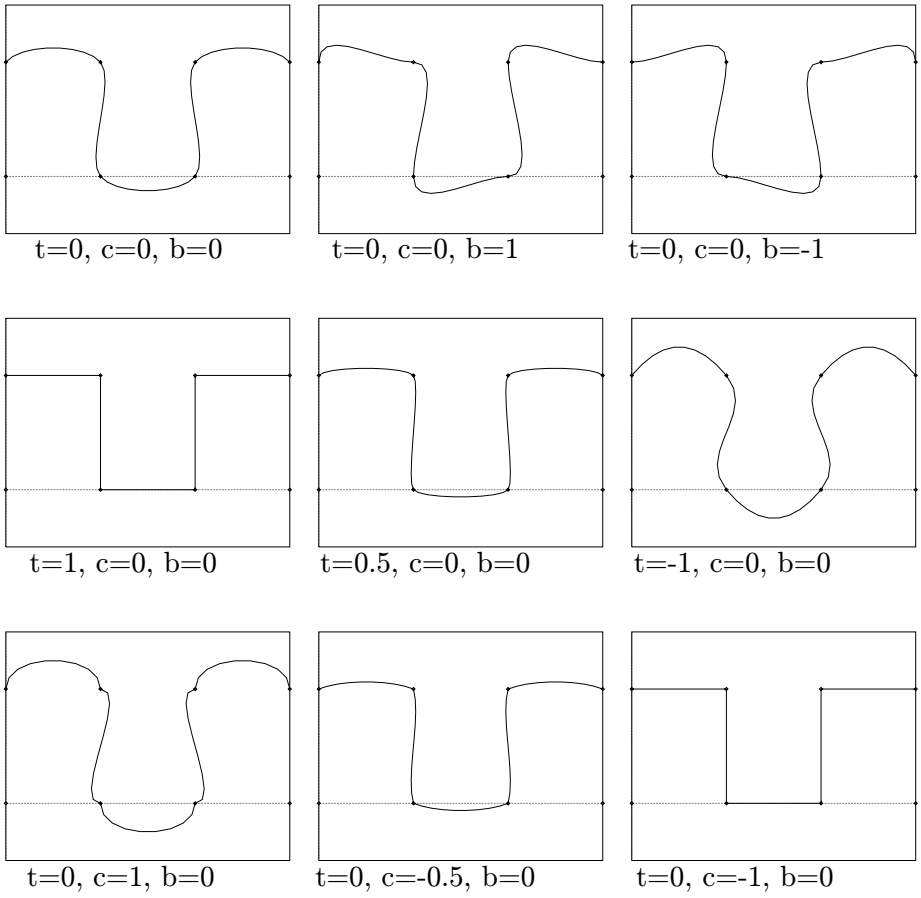


FIG. C.1 - *Aspect de la courbe en fonction de différentes valeurs de continuité, de tension et de biais*

C.2 Splines reparamétrées

Pour calculer une longueur de chemin directement avec le paramètre spline, nous avons besoin de splines reparamétrées, telles que le paramètre soit partout proportionnel à la longueur de la spline. Ce type de spline a été introduit par [GP90], et est décrit dans [GL95]. Le principe utilisé est d'échantillonner la courbe spline pour des valeurs de u équiréparties u_1 ... u_n, avec $du = u_{i+1} - u_i$. Pour chaque échantillon on évalue la distance qui sépare le point associé $P(u)$ sur la courbe au point suivant :

$$dl_i = ||P(u_{i+1}) - P(u_i)|| = \sqrt{dx_i^2 + dy_i^2 + dz_i^2}$$

où dl_i est la variation de l'abscisse curviligne entre u_i et u_{i+1}, dx_i, dy_i et dz_i sont les variations des coordonnées du point P de la spline entre $P(u_i)$ et $P(u_{i+1})$. On dispose ainsi de la longueur totale de la courbe :

$$L = \sum_{i=1}^{n-1} dl_i$$

et de l'abscisse courante l par interpolation linéaire entre deux évaluations. Si $u = u_k + \lambda(u_{k+1} - u_k)$ on a :

$$l(u) = \sum_{i=1}^{k-1} dl_i + \lambda dl_k$$

Une opération similaire est effectuée dans l'espace des quaternions pour les orientations.

Bibliographie

[BB87] R. Barzel and A. Barr. Modeling with dynamic constraints. *State of the Art in Image Synthesis (SIGGRAPH'87 course notes Number 17, Anaheim, Ca)*, 1987.

[BB88] R. Barzel and A. Barr. A modeling system based on dynamic constraints. *Computer Graphics*, 22(4):179–188, August 1988.

[BBZ91] N. I. Badler, B. Barsky, and D. Zeltzer. *Making Them Move*. Morgan Kaufmann Publishers Inc., 1991.

[BH75] A.E. Bryson and Y. Ho. *Applied Optimal Control: Optimization, Estimation, and Control*. Hemisphere Publishing Corp., 1975.

[BH89] R. Bartels and I. Hardtke. Speed adjustement for key-frame interpolation. *Graphics Interface '89*, pages 14–19, May 1989.

[BN88] L. Shapiro Brotman and A.N. Netravali. Motion interpolation by optimal control. *Computer Graphics*, 22(4):309–315, August 1988.

[Coh92] M. Cohen. Interactive spacetime control for animation. *Computer Graphics*, 26(2):293–302, July 1992.

[DSP92] R. Dickstein, Z. Smolinski, and T. Pillar. Self-propelled weight-relieving walker for gait rehabilitation. *Journal of Biomedical Engineering*, 14:351–355, July 1992.

[Gas93] M.P Gascuel. An implicit formulation for precise contact modeling between flexible solids. *Computer Graphics*, pages 313–320, August 1993. Proceedings of SIGGRAPH'93.

[GG94] J.D. Gascuel and M.P. Gascuel. Displacement constraints for interactive modeling and animation of articulated structures. *The Visual Computer*, 10(4):191–204, March 1994.

[Gir87] M. Girard. Interactive design of 3d computer-animated legged animal motion. *Computer Graphics and applications*, 7(6):39–51, June 1987.

[GL95] J.D. Gascuel and C. Lyon. A new set of tools to describe and tune trajectories. In *Computer Animation*, pages 82–89, May 1995.

[GP90] B. Guenter and R. Parent. Computing the arclength of parametric curves. *IEEE Computer Graphics and Applications*, 10(3):21–29, November 1990.

[Han93] G. Hanotaux. Techniques de contrôle du mouvement pour l'animation. *Thèse de doctorat*, École Nationale Supérieure des Mines de Saint-Étienne, Université de Saint-Étienne, April 1993.

[HP93] G. Hanotaux and B. Peroche. Interactive control of interpolation for animation and modeling. *Graphics Interface '93*, pages 201–208, May 1993.

[HRS91] M. G. Hollars, D. E. Rosenthal, and M. A. Sherman. *SD/FAST User's Manual*. Symbolic Dynamics Inc., 1991.

[IC87] P.M. Isaacs and M.F. Cohen. Controlling dynamic simulation with kinematic constraints, behavior functions and inverse dynamics. *Computer Graphics*, 21(4):215–224, July 1987.

[IC88] P.M. Isaacs and M.F. Cohen. Mixed method for complex kinematic constraints in dynamic figure animation. *The Visual Computer*, 2(4):296–305, December 1988.

[Inm81] V. T. Inman. *Human Walking*. Williams and Wilkins, 1981.

[JL93] S. Jimenez and A. Luciani. Animation of interacting objects with collisions and prolonged contacts. *Modeling in Computer Graphics*, pages 129–141, 1993.

[JLR93] S. Jimenez, A. Luciani, and O. Raoult. Physical simulation of land vehicles with obstacle avoidance and various terrain interactions. *The Journal of Visualisation and Computer Animation*, 4:79–94, 1993.

[KB84] D.H.U. Kochanek and R.H. Bartels. Interpolating splines with local tension, continuity, and bias control. *Computer Graphics*, 18(3):33–41, 1984.

[LG95] A. Lamouret and M.P. Gascuel. Scripting interactive physically-based motions with relative paths and synchronization. *Graphics Interface'95*, May 1995.

[LGC94] Z. Liu, S.J. Gortler, and M.F. Cohen. Hierarchical spacetime control. *Computer Graphics*, pages 35–42, July 1994. Proceedings of SIGGRAPH'94.

[LGG95] A. Lamouret, M.P. Gascuel, and J.D. Gascuel. Combining physically-based simulation of colliding objects with trajectory control. *The Journal of Visualization and Computer Animation*, 6(2):71–90, April-June 1995.

[NAC94] J.L. Nougaret, B. Arnaldi, and R. Cozot. Optimal motion control using a wavelet network as a tunable deformation controller. In *Fifth Eurographics Workshop on Animation and Simulation*, September 1994.

[NAHR95] J.L. Nougaret, B. Arnaldi, G. Hégron, and A. Razavi. Quick tuning of a reference locomotion gait. In *Computer Animation*, pages 146–153, May 1995.

[NAS78] NASA. The anthropometry source book. Nasa reference publication 1024, Johnson Space Center, Houston, 1978.

[NM93] J.T Ngo and J. Marks. Spacetime constraints revisited. *Computer Graphics*, August 1993. Proceedings of SIGGRAPH'93.

[PF88] X. Pintado and E. Fiume. Grafields: Field-directed dynamic splines for interactive motion control. *Proceedings of the Eurographics Conference*, pages 43–54, September 1988.

[PGW81] W. Murray P. Gill and M. Wright. *Practical Optimization*. Academic Press, New York, 1981.

[PTVF92] William Press, Saul Teukolsky, William Vetterling, and Brian Flannery. *Numerical Recipes in C, second edition*. Cambridge University Press, New York, USA, 1992.

[RH91] M. Raibert and J. Hodgins. Animation of dynamic legged locomotion. *Computer Graphics*, 25(4):349–358, July 1991.

[SB85] S. Skeletee and N. Badler. Parametric keyframe interpolation incorporating kinetic adjustement and phrasing control. *Computer Graphics*, 19(3):255–262, July 1985.

[Sev89] Y. Sevely. *Systèmes et asservissements linéaires échantillonnés*. Dunod Univertité, Bordas, Paris, France, 1989.

[vdPF93] M. van de Panne and E. Fiume. Sensor-actuator networks. *Computer Graphics*, August 1993. Proceedings of SIGGRAPH'93.

[vdPKF94a] M. van de Panne, R. Kim, and E. Fiume. Synthesising parametrized motions. *Fifth Eurographics Workshop on Animation and Simulation*, September 1994.

[vdPKF94b] M. van de Panne, R. Kim, and E. Fiume. Virtual wind-up toys for animation. *Graphics Interface '94*, May 1994.

[vdPL95] M. van de Panne and A. Lamouret. Guided optimization for balanced locomotion. *6th Eurographics Workshop on Animation and Simulation*, 1995.

[WK88] A. Witkin and M. Kass. Spacetime constraints. *Computer Graphics*, 22(4):159–168, August 1988.

[WS89] J. Wilhelms and R. Skinner. An interactive approach to behavioral control. *Graphics Interface '89*, 1989.