$N°$d'ordre : 2837

THÈSE

présentée à

L'UNIVERSITÉ BORDEAUX I

ÉCOLE DOCTORALE DE MATHÉMATIQUES ET INFORMATIQUE

par Olivier PARISY

POUR OBTENIR LE GRADE DE

DOCTEUR

SPÉCIALITÉ : Informatique

****

GÉNÉRATION AUTOMATIQUE DE MODÈLES PHYSIQUES À PARTIR DE MODÈLES GÉOMÉTRIQUES ANIMÉS PAR SQUELETTE
Application aux Animats

****

Soutenue le : 2 Juillet 2004

Après avis de :

MM.	Yves	DUTHEN, Professeur, Toulouse	**Rapporteurs**
	Jacques	TISSEAU, Professeur, Brest	

Devant la commission d'examen formée de :

MM.	Jean-Marc	SALOTTI	**Président**
	Pascal	GUITTON	**Rapporteur**
	Yves	DUTHEN	**Examinateur**
	Christophe	SCHLICK	**Examinateur**
	Jacques	TISSEAU	**Examinateur**

- 2004 -

Remerciements

Mes remerciements vont tout d'abord à Emilie, qui a supporté mes doutes et m'a soutenu dans ce travail. Merci à ma famille pour m'avoir toujours appuyé dans mes choix.

Merci à Christophe pour sa confiance tout au long de mes errances et explorations intellectuelles.

Merci à mes relecteurs, Yann et Florence, qui ont chassé les coquilles et amélioré ce document bien plus patiemment que je n'aurais pu le faire.

Et enfin merci à mes amis et collègues, trop nombreux pour les citer, pour m'avoir conseillé et changé les idées tout au long de ces années.

Résumé

L'animation et la simulation informatique de créatures virtuelles sont des domaines ayant bénéficié d'un effort de recherche important de la part de la communauté de l'informatique graphique. Un certain nombre de techniques, telles que l'interpolation de clefs (keyframing) ou la capture de mouvement, sont très largement utilisées mais imposent un travail important à l'animateur du fait des possibilités limitées de réutilisation des mouvements produits dans des projets ultérieurs.

En parallèle à ces techniques "en boucle ouverte", des approches basées sur des outils de plus haut niveau ont vu le jour : simulation physique, algorithmes de locomotion / préhension, systèmes perceptifs, etc. Toutes présentent l'intérêt d'abstraire le mouvement en fournissant à l'animateur des outils de contrôle de l'animation aux paramètres moins nombreux et plus génériques que les techniques traditionnelles.

Cette problématique du contrôle d'un système en fonction de certains objectifs, ainsi que certains des outils décrits trouvent leur pendant dans le domaine de la vie artificielle. Celle-ci s'intéresse en effet à la genèse de phénomènes biologiques via la synthèse de systèmes se comportant de manière similaire aux êtres vivants. Au-delà de la simulation informatique, des implémentations de ces techniques dans le cadre de la robotique ont montré leur intérêt pratique et les possibilités offertes par la genèse de comportements dans un environnement complexe.

Sur la base de l'observation d'un besoin de généricité des outils d'animation en informatique graphique, et de l'intérêt de fournir un environnement complexe dans lequel effectuer des simulations en vie artificielle, nous montrerons l'utilité d'un rapprochement des problématiques de ces communautés via une plate-forme commune, elle-même articulée autour de la notion de simulation physique et la génération aisée de modèles physiques de créatures artificielles.

Nous étudierons ainsi un cadre de travail commun tenant compte des besoins et contraintes de ces disciplines, les difficultés posées par l'utilisation d'une simulation physique et en quoi les outils que nous proposons en facilitent l'accès. Nous montrerons enfin les usages envisageables pour cette plate-forme, et les apports attendus aussi bien dans les domaines de l'informatique graphique que de la vie artificielle.

Summary

The animation and computer simulation of virtual creatures are topics which have benefitted from a strong research impulse from the community of computer graphics. Some techniques, such as keyframing of motion capture, are widely used but impose a great burden on the animator because of the limited re-use possibilities of produced movements in new projects.

Parallel to those open loop techniques, other approaches based on higher level tools have been developped : physical simulation, locomotion / prehension algorithms, perceptive systems, etc. They all have the interest of abstracting the movement by bringing the animator control tools with higher-level parameters, and in a smaller number than traditionnal techniques.

This problem of system control with respect to some goals, and some described tools have counterparts in the artificial life domain. This discipline addresses the issue of the genesis of biological phenomenom by synthetising systems which behave in a way similar to life beings. Beyond computer simulations, robotic implementations of those techniques have shown their practical interest and possibilities offered by behaviours generation in a complex environment.

Basing our work on the observation of a need of genericity for animation tools in computer graphics, and on the interest of a complex environment in which artificial life simulations can take place, we will show the usefulness of bringing together the issues of those communities using a common platform, itself based on a physical simulation and the easy generation of physical models of artificial creatures.

We will hence study a framework taking into account needs and constraints of those disciplines, the issues raised by physical simulation use and how the tools we are proposing make it easier to use. We will finally show how this platform can be used, and the expected contributions either from a computer graphics or artificial life point of view.

Table des matières

Table des figures

Introduction

L'animation en informatique graphique est un domaine d'une grande vitalité, en particulier du fait de ses nombreux usages : génération de trucages, voire de séquences complètes pour le cinéma et la publicité, visualisation scientifique, études en ergonomie, jeux vidéo...

Ce domaine vise traditionnellement à la génération de descriptions informatiques de scènes constituées de modélisations d'éléments de décor variés et d'acteurs au sens large du terme. Cette description est ensuite exploitée par des outils de synthèse d'images qui les utilisent afin de générer les images ou animations proprement dites.

Dans le cadre de notre mémoire de DEA, nous avons eu l'occasion d'étudier plus en détail les outils permettant la génération de ces modèles et leur animation, puis de mettre en œuvre un outil de modélisation original sous forme d'extension à un logiciel de synthèse d'images, Maya [84].

Lors de ce travail, nous avons pu constater qu'en parallèle à cette approche classique de la génération d'images de synthèse, l'étude d'outils permettant une interaction avec les éléments et acteurs visualisés, ou une production interactive d'animations réagissant à des évènements externes a pris une ampleur considérable. Au-delà de l'étude des réalités virtuelles, l'une des raisons de cet engouement est probablement l'augmentation des puissances de calcul, qui a permis l'utilisation de ces techniques dans le domaine des jeux vidéos.

Nous nous sommes ainsi, dans le cadre de cette thèse, intéressés à des outils d'animation de plus haut niveau que ceux habituellement utilisés, donc mettant l'accent sur l'interactivité et l'autonomie de modèles d'êtres vivants plutôt que sur la facilité pour un animateur de manipuler précisément le détail des résultats obtenus.

Afin de présenter ces travaux, nous tenterons dans un premier temps de donner une vue d'ensemble des approches classiques et plus interactives de la génération de modèles d'êtres vivants animés (chapitre 1). Nous généraliserons ensuite notre propos en présentant dans le chapitre 2 une autre approche visant à l'autonomie et traitant ainsi d'outils et de problématiques proches : l'étude des animats et de la robotique autonome.

Le chapitre 3 sera dévolu à une présentation de l'utilité d'un environnement réaliste pour l'étude de méthodes d'animation interactives, puis à la discussion d'une réalisation logicielle permettant sur ces bases une génération aisée de modèles d'êtres vivants animables dans une simulation physique, et cette simulation à proprement parler. Nous proposerons enfin dans le chapitre 4 des possibilités d'évolution et d'exploitation de ces outils, puis nous conclurons dans le chapitre 5.

Un certain nombre d'éléments complémentaires ont été reportés en annexe ; nous y préciserons en particulier les notions physiques utilisées dans nos simulations (annexe A), et nous y aborderons des éléments du comportement animal ayant servi d'inspiration à un certain nombre de travaux présentés.

Chapitre 1

Animation d'Êtres Vivants en Informatique Graphique

1.1 Introduction

De part leur intérêt en images de synthèse, un corpus important de techniques et d'outils ont été développés pour la représentation et l'animation d'êtres vivants, et en particulier d'êtres humains. Après une présentation de techniques classiques en modélisation et animation, nous aborderons des techniques plus spécifiques visant à l'obtention de mouvements et comportements plus réalistes et faciles à contrôler que dans le cadre d'outils génériques. Nous discuterons ensuite de leurs caractéristiques, puis des enjeux et apports d'une animation intégrant une simulation physique pilotée par des contrôleurs adaptés.

1.2 Modélisation

Avant d'animer un être vivant, il est nécessaire de le représenter. Nous détaillerons ici les principales techniques utilisées pour modéliser un être vivant, et la manière dont ces modèles sont animables. Notons que la plupart des études dans ce domaine ont été appliquées aux êtres humains, ceux-ci étant d'un intérêt tout particulier de par les problématiques qu'ils permettent d'étudier et les informations qu'ils peuvent transmettre. Citons ainsi des applications telles que l'interface facilitée avec des systèmes de vente ou d'information [93], ou le dialogue enrichi par la transmission d'émotions.

1.2.1 Modélisation Surfacique

De manière générale, toute animation d'un être vivant est réductible aux techniques traditionnelles en informatique graphique : un ensemble de polygones ou de surfaces paramétriques définissant la surface du modèle est modifié image par image via des outils de déformations génériques, que nous aborderons section 1.2.2.

Représentation Polygonale (Maillages)

Il s'agit ici d'approximer une surface sous forme d'un ensemble de points, ceux-ci étant les sommets d'un ensemble de polygones (souvent des triangles, plus "robustes" algorithmiquement car toujours plans et convexes). Le succès de ce type de description, malgré la perte certaine de qualité qu'elle implique, est justifié par les facilités de manipulation et surtout de visualisation qu'elle permet.

En effet, les intersections entre surfaces se font à complexité constante, et celles avec des droites ont des solutions analytiques. De plus, un certain nombre de techniques permettent d'obtenir un rendu intéressant en temps réel (placage de textures, ombrages), ce qui permet une accélération matérielle de la visualisation.

Cette facilité d'affichage a motivé un certain nombre de recherches qui ont abouti à des algorithmes permettant de convertir les autres représentations géométriques en ce formalisme, ce qui permet d'observer en temps réel une approximation valable de l'objet manipulé.

Par contre, sa structure même empêche la représentation précise de surfaces courbes, ce qui la rend peu adaptée aux activités de modélisation ou de représentation exacte. La famille de surfaces décrite ci-après est le plus souvent utilisée pour cela ; cette représentation sera ensuite éventuellement convertie en maillage si cela s'avère nécessaire.

Représentation par Approximation ou Interpolation

Pour répondre aux limitations de la représentation précédente, une solution consiste à conserver un ensemble de points (nommés dans ce cadre "points de contrôle"), puis à considérer que la surface manipulée vérifie une certaine relation par rapport à ces points. Il en existe de deux types :

– dans le cas d'une approximation, la surface est proche des points de contrôle ;
– dans le cas d'une interpolation, la surface passe par tous les points de contrôle.

De plus, on utilise en général des familles de surfaces ayant la propriété de normalité [12, page 20]. Dans ce cas, la surface ne dépend pas du repère dans lequel ses points de contrôle sont exprimés. Sa forme n'est ainsi pas modifiée par les transformations affines et une interpolation des points de contrôle aboutit à une interpolation des surfaces correspondantes. Notons que cette exigence n'est guère restrictive, puisqu'elle permet un comportement intuitif de la surface et est donc respectée par toutes les familles de surfaces classiquement utilisées.

Dans tous les cas, le type de surface obtenu dépend des fonctions utilisées pour calculer la contribution de chaque point, et dans le cas général de coefficients (les poids) associés à chaque point de contrôle et permettant de modifier leur influence.

Ces surfaces, bien que nécessitant un certain volume de calculs pour être manipulées et étant délicates à visualiser directement, ont des avantages indéniables sur la représentation polygonale et ont été largement étudiées.

1.2.2 Techniques de Déformation Classiques

Nous allons présenter ici rapidement un certain nombre d'outils de déformation ; nous avons détaillé ces techniques dans un article publié en 2000 [84].

Warping, *bending*

Historiquement, certains des premiers outils de déformation ont été proposés par Parent en 1977 [82]. Il s'agissait essentiellement de déformer des surfaces polygonales (*meshes*) via le déplacement de leurs sommets, un concept qui sera largement utilisé par la suite. L'approche de Parent était novatrice en ce qu'il visait l'utilisation de son système par des artistes, et se focalisait donc sur la simplicité d'accès de l'interface et l'utilisation de métaphores.

Les concepts décrits sont les suivants : on dispose de primitives et de volumes prismatiques. Ceux-ci sont ensuite manipulés via des translations, rotations et mis à l'échelle puis combinés par des additions ou des soustractions effectuées via un calcul d'intersection.

Deux outils de déformation sont alors proposés :

– Le *warping* consiste en la sélection d'un point P, puis en son déplacement. Cette action induit le déplacement d'un certain nombre de points au voisinage de P en fonction de leur distance à ce même point, ceci via une fonction de pondération donnant une impression d'élasticité.
– Le *bending* s'effectue en deux étapes : tout d'abord, on définit un squelette (ensemble de segments) sur lequel la géométrie est projetée. Ensuite, les déformations du squelette (en général des rotations) sont répercutées interactivement sur le modèle.

Le *bending* est particulièrement intéressant en ce qu'il préfigure beaucoup d'outils modernes basés sur ce même mécanisme : un calcul de coordonnées locales généralement relativement coûteux mais effectué une fois pour toutes, puis une altération interactive d'un outil se répercutant sur l'objet à déformer.

Déformations axiales

Une autre grande famille de déformations est constituée par les déformations axiales. Leur formulation initiale a été proposée par Barr en 1984 [10], dans une publication largement citée par la suite. Son approche a consisté à définir un certain nombre d'opérateurs associés à un axe pouvant être librement positionné. Son objectif était de déformer des modèles solides tout en conservant les informations sur les tangentes et normales à leur surface, et ceci via des produits matriciels qui offrent une complexité de calcul faible.

Les opérateurs globaux sont les suivants (tous s'effectuent autour d'un axe) :

– *bend* : "pliage" de l'objet ;
– *twist* : "enroulement" de l'objet ;
– *taper* : "effilage" de l'objet.

Ici aussi, une métaphore est utilisée : Barr parle de modeler de l'argile, ou de plier du métal.

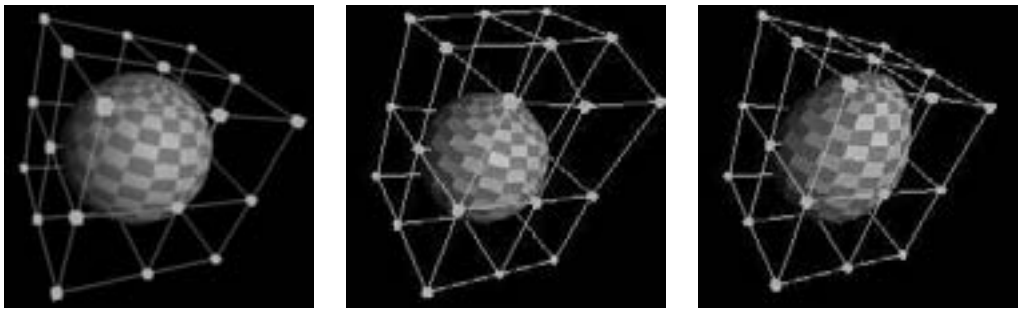
Une méthode analytique générant des déformateurs locaux à partir d'opérations locales est ensuite proposée, utilisant la matrice jacobienne d'une fonction de l'espace.

Les opérateurs de Barr, considérés maintenant comme des outils de base en modélisation au même titre que les unions et intersections, ont été largement utilisés et étendus par la suite, ainsi que nous le détaillons dans [84].

FFD et dérivées

De par son usage propre et le nombre de ses dérivées, la FFD (*free-form deformation*) est probablement le type de déformation ayant rencontré le plus de succès (voir à ce sujet un article de Mikita [69] pour une étude exhaustive). Elle a été proposée initialement par Sederberg et Parry en 1986 [97, 96], et se base sur le mécanisme suivant :

1. un réseau (*lattice*) parallélépipédique est défini par un certain nombre de subdivisions sur les trois axes d'un repère ; ce volume englobe l'objet à déformer ;
2. pour chaque point caractéristique de l'objet, des coordonnées locales au réseau de déformation sont calculées (formellement, il s'agit d'un changement de repère) ;
3. des points du réseau sont ensuite déplacés ;
4. en considérant que l'ensemble des points du réseau définissent un volume de Bézier (par un triple produit tensoriel), les coordonnées globales de chaque point peuvent être recalculées par interpolation .



(a) gel **(b)** déformation locale **(c)** dégel

FIG. 1.1 – Exemple d'utilisation des FFD

L'étape 1 correspond à la définition de l'outil de déformation, l'étape 2 à un "gel" des coordonnées (figure 1.1a), l'étape 3 à la manipulation de l'outil (déformation locale, figure 1.1b) et la 4 au "dégel" des coordonnées (figure 1.1c).

La force de cet outil tient en plusieurs points :
- Il repose sur une métaphore simple, le bloc de plastique : l'objet est coulé dans un plastique transparent, et les déformations du bloc se répercutent sur le modèle.
- L'étape de gel, la plus coûteuse, est effectuée une seule fois alors que les étapes de déformation et de dégel, plus rapides, peuvent être effectuées de manière interactive.
- La déformation d'un petit nombre de points (quelques dizaines en général) permet la déformation de manière continue d'un nombre arbitraire de points : cette déformation est donc de "haut niveau".

Ce modèle est à la base de toute une famille de méthodes qui ont tenté de pallier à ses limitations (difficultés d'obtention d'une déformation précise en particulier) et de le généraliser :

reformulation par splines, généralisation du réseau de déformation, manipulation directe de l'objet à déformer...

Utilisation en Animation

Cette approche par déformateurs est souvent rédhibitoire en pratique de par la grande quantité de manipulations nécessaires, et la difficulté à obtenir un résultat réaliste (rien ne guidant l'animateur dans ses manipulations). Par ailleurs, ces animations ne peuvent être générées dynamiquement par définition, sauf à utiliser des outils d'animation assemblés et paramétrés de manière ad hoc (fastidieux et peu réutilisables). Des outils plus génériques s'avèrent nécessaires.

1.2.3 Squelette Associé à une Peau

Concernant la structure du corps, une technique très utilisée est l'animation par squelette, issue du formalisme de Parent [82] décrit en section 1.2.2. Elle consiste en l'adjonction à la surface définie précédemment (la "peau") d'un squelette composé d'un ensemble d'os organisés sous une forme arborescente. L'animateur spécifie avant d'animer son modèle l'influence de chaque os, donc la zone de peau qu'il déformera lorsqu'il se déplacera.

Plus précisemment, à chaque constituant de la surface (vertex pour un maillage, point de contrôle pour une surface paramétrée) est associé un ou plusieurs os, l'intensité de chacune de ces associations étant précisée par un poids. La combinaison des déplacements de chaque os, pondérée par leurs poids respectifs, permet de calculer les déformations de la surface associée. Un exemple de ce type de déformation est illustré figure 1.2.

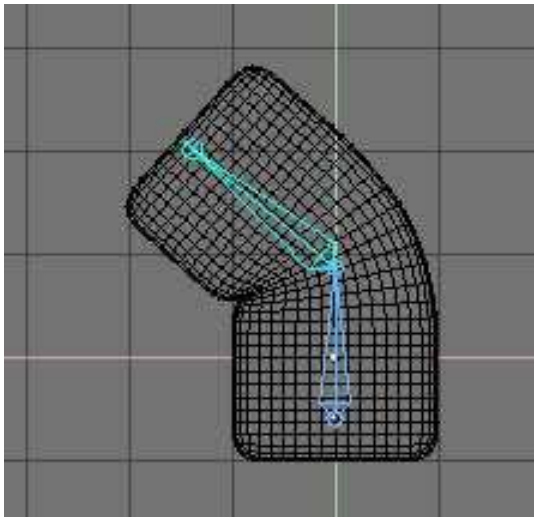


FIG. 1.2 – Déformation par Squelette

L'intérêt de cet outil d'animation est que l'animateur n'a plus maintenant, pour définir une nouvelle image de son animation, qu'à spécifier la "pose" prise par le squelette à ce moment,

le plus souvent via un ensemble de rotations (voir à ce propos la section 1.3.2 consacrée à la cinématique directe, une méthode d'animation des systèmes articulés).

L'animation par squelette peut donc être considérée comme un outil de déformation de surfaces de haut niveau, similaire de ce point de vue aux FFD et autres outils de modélisation. Tout comme ces derniers, elle permet des déformations d'apparence régulière (*smooth-looking*) du fait qu'elle ne se contente pas de déplacer la surface mais la déforme localement.

Ceci limite d'autant les manipulations, et permet l'utilisation de techniques d'animation variées telles la capture de mouvements, les *keyframes* conçus manuellement ou toute autre source de données (voir section 1.3), ce qui simplifie encore la tâche de l'animateur. Il est par ailleurs bien adapté à la structure du corps de beaucoup d'êtres vivants, permettant d'obtenir des résultats relativement réalistes.

Cet outil est, de plus, très largement disponible : exploitable dans les logiciels d'animation majeurs, il est par ailleurs standardisé en VRML, grâce au travail de l'*Humanoid Animation Working Group*. Les données nécessaires à la dynamique inverse (masses, contraintes angulaires, cf. section 1.3.4) y sont en effet exprimables dans un modèle hiérarchique du squelette humain, la norme H-Anim [101] (voir la figure 1.3).

FIG. 1.3 – Hiérarchie Standard H-Anim

La norme MPEG-4, bien qu'elle ne définisse pas directement de modèle de ce genre, est très utilisée puisqu'elle permet la transmission d'informations permettant la description, le

calibrage et l'animation de modèles de corps ou de visages. Une description de la norme est fournie par les documents émis par le comité MPEG [29] ; les travaux récents de Firsova *et al* [38] en sont un exemple d'implémentation complexe (la figure 1.5 en est une illustration).

1.2.4 Déformations Spécifiques

La représentation par squelette fournit des résultats intéressants, mais pas toujours suffisamment réalistes. En effet, un être vivant ne se réduit pas à une surface tendue sur une armature : un ensemble de structures internes, et plus particulièrement des tissus graisseux et musculaires, jouent sous la peau lors des mouvements et la déforment de manière complexe.

Ceci peut être représenté via l'utilisation de déformateurs génériques associés à des zones précises du corps, et indexés au mouvement du squelette proche : on peut par exemple ainsi représenter la contraction et le gonflement d'un biceps en fonction de la flexion du coude. Un exemple d'utilisation de cette technique pour un genou dans le cadre du logiciel Maya [3] est représenté figure 1.4.

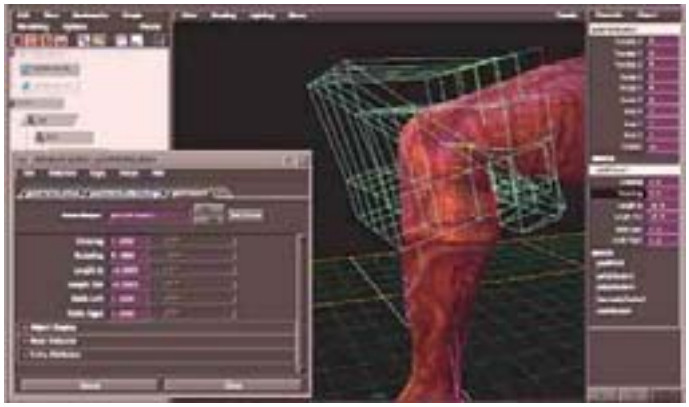


FIG. 1.4 – Déformation Dédiée à une Articulation

Cette technique, très utilisée pour améliorer une animation, impose de consacrer un temps important au paramétrage précis de ces déformations. A contrario, dans le cas du visage en particulier, des outils spécifiques peuvent être développés puis réutilisés. Des travaux de ce type ont été par exemple effectués par S. Kshirsagar [56] dans le cadre de l'équipe MIRALab de Nadia Magnenat-Thalmann.

Bien que reposant sur des bases empiriques (par opposition aux techniques décrites dans la section suivante), des résultats intéressants sont obtenus, et ces techniques permettent souvent une utilisation en temps réel (la figure 1.5 montre ainsi l'application de données capturées sur un visage à des modèles synthétiques animés par cette technique).

Un cas particulier est l'objet de recherches importantes : la représentation et l'animation des lèvres, qui jouent un rôle de premier ordre pour une communication crédible. Toute une série de modèles paramètriques ont ainsi été proposés, parfois même associés à des techniques de rendu spécifiques telles celles proposées par S. King en 2001 [54].

FIG. 1.5 – Animation Faciale

1.2.5 Structures Anatomiques

Les problèmes associés aux déformations ad hoc, et les difficultés d'obtention de résultats réalistes avec ces outils, sont particulièrement flagrants dans le cas de la modélisation d'un visage humain : la recherche de crédibilité impose une modélisation fine du comportement des traits et de la peau, difficile à mettre en place sans prendre en compte l'anatomie musculaire sous-jacente. Ceci peut être effectué via l'utilisation de modèles musculaires géométriques, tels ceux proposés par K. Kähler en 2001 [52, 53] (figure 1.6), ou de modèles physiques tels qu'utilisés par T. Firsova en 2002 dans le cadre de son implémentation de MPEG4 [38].

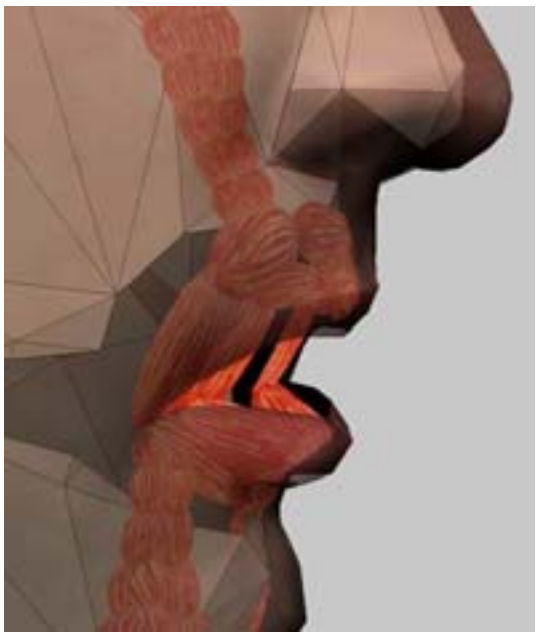


FIG. 1.6 – Muscles Synthétiques

Les techniques précédentes se basent dans les cas les plus réalistes sur des données anatomiques et biomédicales afin de les paramétrer finement. A l'extrême, il est possible de se baser sur ce type d'informations pour reconstruire un visage, voire l'ensemble des tissus constituants le corps. Leurs déformations pourront ensuite être simulées via des algorithmes adaptés tels les éléments finis, appliqués par exemple par R. M. Koch au visage en 1998 ([55], voir figure 1.7) et par Jean-Christophe Nebel aux tissus humains en 2001 [76]

1.3 Génération du Mouvement

Deux approches principales pour la génération du mouvement sont distinguables : son enregistrement (section 1.3.1), pour restitution postérieure, et sa synthèse (sections 1.3.2 et 1.3.4). Dans le second cas, il est possible d'abstraire les mécanismes utilisés via l'utilisation de moteurs d'animation (section 1.3.5).

FIG. 1.7 – Animation d'un Visage Reconstruit

Nous allons présenter ici un certain nombre de ces méthodes. Dans le cadre de la marche humaine, une autre classification et le détail de leurs forces et faiblesses ont été présentés par F. Multon en 1999 [75].

Notons que le plus souvent, du fait de son adéquation avec la structure et les possibilités motrices des vertébrés, nous admettrons le plus souvent ici l'existence d'un modèle d'animation de bas niveau dont la primitive de manipulation est la modification des angles articulaires. Nous nous intéresserons plus particulièrement à la simulation de mouvements et comportements sur cette base.

1.3.1 Acquisition de Mouvements

Une technique permettant simplement la génération de mouvements réalistes consiste en l'acquisition de mouvements depuis un acteur réel. Un certain nombre de méthodes existent pour cela, allant de combinaisons à capteurs variés jusqu'au traitement des images fournies par un certain nombre de caméras (citons par exemple les travaux de B. Stuart *et al* publiés en 2001 [102]). Elles s'avèrent suffisamment précises pour permettre un enregistrement fidèle, et permet d'obtenir ou des angles articulaires, ou les déplacements de points-clefs de l'acteur dans l'espace. Dans les deux cas, la connaissance d'un modèle géométrique de l'acteur permet de passer d'une représentation à l'autre. D. Sturman a proposé un cours fournissant une vue d'ensemble de cette famille d'outils durant Siggraph 1994 [103].

Si ces enregistrements permettent une grande qualité des mouvements, en particulier en terme d'accélération et de vitesse, se pose le problème de leur généralisation. D'une part, le résultat est strictement déterminé par l'enregistrement initial, il est donc nécessaire de disposer d'une banque de données suffisamment importante pour répondre à l'ensemble des besoins de l'animateur. Des techniques d'altération de ces enregistrements et d'interpolations entre eux existent cependant ; elles seront décrites dans la section 1.3.3.

D'autre part, le problème de la généralisation à des modèles géométriques variés se pose : s'il est possible de transposer des mouvements humains à des humanoïdes via une modification appropriée des longueurs des segments du corps pris en compte par le modèle de l'acteur, une extension à un quadrupède par exemple ne peut se faire simplement. De manière générale, de ces mesures est absente toute référence au contexte ou à la génération du mouvement qu'elles représentent.

1.3.2 Synthèse de Trajectoires

Les méthodes décrites ici se caractérisent, tout comme l'acquisition de mouvements, par une manipulation de trajectoires indépendamment de leurs causes. Il est ainsi possible que des mouvements physiquement impossibles soient générés, et tout comme en animation traditionnelle un certain talent est requis pour obtenir des mouvements réalistes.

Keyframes

Le *keyframing* (interpolation de *positions clefs*) permet la spécification d'un certain nombre de points clefs de la trajectoire d'un point. Sur cette base, la spécification des dates et éventuellement des vitesses et tangentes associées à ces points clefs permet une reconstitution de cette trajectoire. Pour cela, on utilise des mécanismes d'*interpolation* entre clefs (le plus souvent des courbes splines).

Leur avantage est la possibilité de manipuler une trajectoire continue via un nombre de paramètres relativement limités. L'extension de cette technique à l'ensemble des points à déplacer pour animer un modèle permet ainsi la spécification des mouvements des membres.

Notons qu'il existe manifestement ici une dépendance entre, par exemple, les trajectoires d'un poignet, d'un coude et d'une épaule, du fait de la rigidité des segments de membres. Une animation complète de modèle par *keyframing* serait donc fastidieuse du fait de la quantité de contraintes à prendre en compte (incontournables sous peine de dislocation du modèle).

Cinématique Directe

Une solution au problème d'intégrité du modèle dans le cadre du *keyframing* consiste en une animation, non plus d'un ensemble de points, mais d'un modèle respectant des contraintes métriques et angulaires. Une structure articulée constituée d'un arbre de segments connectés via des articulations aux caractéristiques précises est utilisée pour cela. Un squelette est ainsi défini ; une fois associé à une surface via des outils de déformation adaptés, il permettra la déformation de cette dernière. L'opération permettant de calculer les déplacements des segments du squelette à partir de ses degrés de liberté articulaires est nommée *cinématique directe* [60, 104]. La figure 1.8 présente ainsi un modèle de bras articulé industriel dont les paramètres de contrôle sont les angles θ_1, θ_2 et θ_3.

Ce mode de contrôle d'un modèle est peu coûteux algorithmiquement, et s'avère au premier abord intuitif. Cependant, le type de mouvements le plus couramment effectué par une structure articulée consiste en un positionnement précis de certains de ses points, généralement ses extrémités (pour des mouvements de locomotion, de préhension...). Le contrôle angulaire s'avère alors malaisé pour un manipulateur humain.

Cinématique Inverse

La technique de cinématique inverse (*Inverse Kinematic*, IK) permet de tirer parti de la cohérence des animations obtenues par cinématique directe tout en disposant d'une précision de positionnement de certains points. Elle permet en effet de calculer l'ensemble des variations

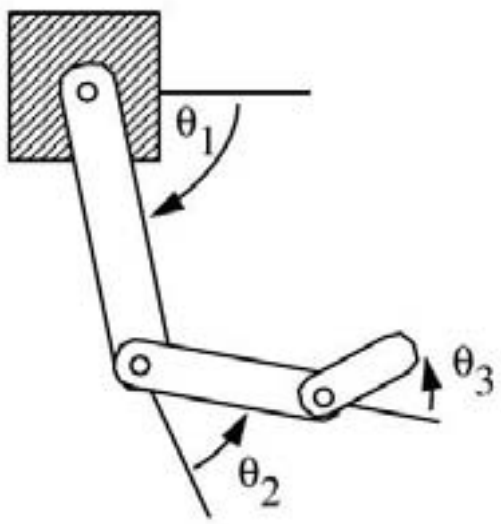


FIG. 1.8 – Cinématique Directe

angulaires d'un modèle articulé générant un mouvement spécifié (par *keyframing par exemple*) de ses extrémités, ou *effecteurs* (voir la figure 1.9 pour un exemple de ce type).

Concrètement, une matrice caractéristique du système, son *jacobien*, est calculée : elle permet de passer pour chaque chaîne de la position de l'effecteur au vecteur d'état (ensemble des angles) correspondant. Une introduction aux outils mathématiques utilisés est disponible dans la synthèse de B. Lingard [60], ou plus récemment celle de F. Multon [75].

Cette technique, issue de la robotique, permet une génération aisée de mouvements complexes via la manipulation de quelques paramètres, tout en conservant une grande précision de déplacement pour les points significatifs ; cela permet par exemple facilement de faire prendre une tasse à un modèle en manipulant la trajectoire de sa main par *keyframing*, le dernier point étant l'anse de la tasse.

Cependant, se pose le problème du réalisme des mouvements obtenus ; une grande maîtrise des vitesses est nécessaire pour que le mouvement ne paraîsse ni trop mécanique, ni trop fluide, et cela nécessite concrètement, comme en modélisation, une bonne maîtrise du comportement du modèle sous-jacent. Ceci est en partie dû aux mécanismes d'interpolation utilisés, qui ne tiennent en général pas compte de la dynamique du mouvement.

Par ailleurs, ce problème est en général mal posé : pour une structure d'au moins deux segments, il n'existe pas une solution unique au problème de positionnement de l'extrémité de la chaîne. Une méthode couramment retenue consiste en l'ajout d'une fonction à minimiser (tâche secondaire) afin de mieux contraindre le problème.

Le coût de cette approche est bien supérieur à celui de l'approche directe, mais reste raisonnable : l'inversion du jacobien mène à une complexité de $\mathcal{O}(3)$. Son usage étant en général limité à une jambe ou un bras, ce surcoût reste négligeable.

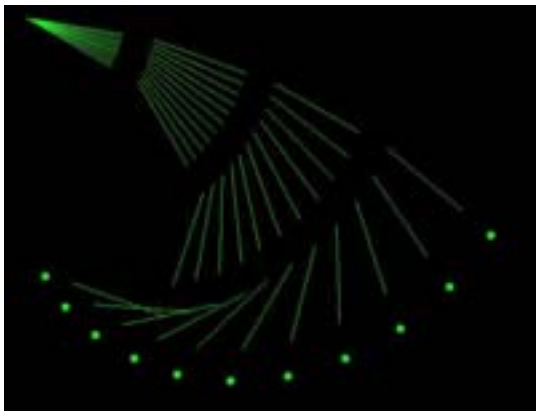


FIG. 1.9 – Cinématique Inverse

1.3.3 Altération de Mouvements

Les approches décrites ici se basent sur un ou plusieurs mouvements déjà spécifiés afin d'en générer de nouveaux ; il est ainsi possible d'obtenir toute une famille de mouvements dérivés à partir d'un ensemble fini initial. Par ailleurs, le fait que ces approches puissent exploiter des mouvements valides physiquement (tels ceux obtenus par acquisition de mouvements) permet souvent d'obtenir des trajectoires plus réalistes que celles synthétisées. Ces techniques sont décrites dans la synthèse de Frank Multon *et al* [75].

Mélange de Mouvements (*blending*)

Le *motion blending* consiste en une interpolation entre les caractéristiques de quelques mouvements extraits d'une base de données. Un exemple classique consiste en la génération de toute une série de démarches à partir d'un cycle de marche énergique et d'une autre démarche plus "triste" : il sera alors possible de choisir l'animation désirée en fonction d'un paramètre "humeur". L'interpolation peut se faire dans le domaine fréquentiel, en se basant sur les harmoniques les plus basses du mouvement, ou dans le domaine spatial. Ce type d'approche peut se généraliser, non plus à deux mais à toute une série de mouvements. Le mouvement désiré se spécifie donc dans un espace de paramètres, d'où une grande liberté et facilité pour l'animateur. Un exemple de travaux de ce type a été proposé par I. S. Lim en 2001 [59] (voir la figure 1.10).

L'avantage de ces techniques est la qualité des mouvements obtenus, de par leur origine. Cependant, la liberté offerte par un grand nombre de paramètres de haut niveau se paie par la taille importante de la base de mouvements nécessaire. De plus, les mécanismes d'interpolation utilisés n'ont *a priori* aucune base physique, d'où un risque de mouvements irréalistes dans le cas de séquences à interpoler éloignées.

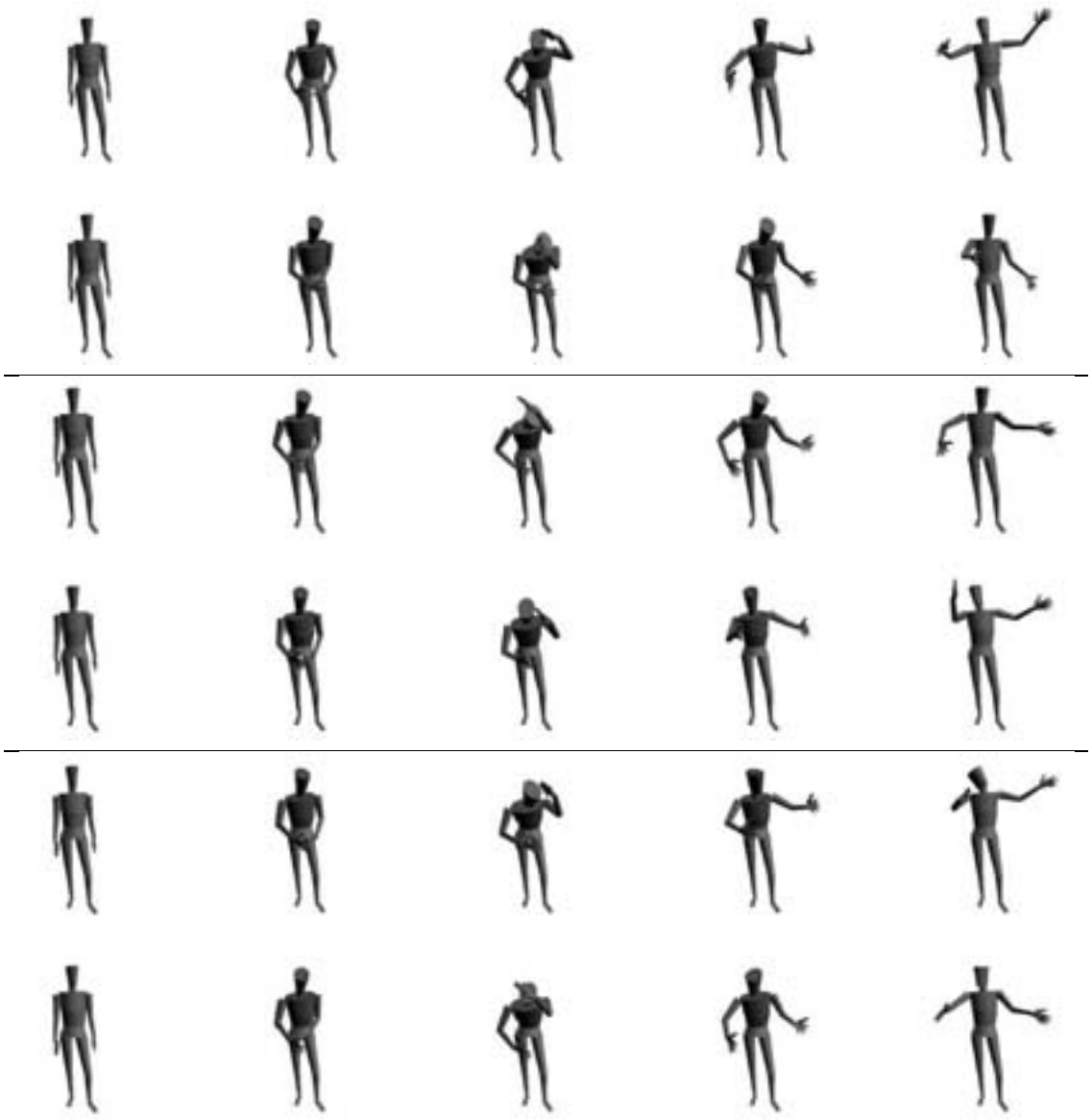


FIG. 1.10 – Altération de Trois Animations

Déformation de Mouvements (*warping*)

Le *motion warping* consiste en une déformation d'un mouvement défini sous forme de trajectoires, qu'il soit capturé ou généré. Tout comme dans le cas précédent, elle peut s'effectuer dans le domaine spatial ou fréquentiel.

Cela peut se faire par spécification de contraintes : les trajectoires sont manipulées comme s'il s'agissait de *keyframes*, par spécification de deux paires position/date et éventuellement de consignes indiquant des décalages dans le temps de points clefs du mouvement. Un schéma d'interpolation permet alors de construire la trajectoire satisfaisant ces contraintes, suivant une technique proposée en 1995 par A. Witkin [114]. Un exemple tiré de cet article est représenté figure 1.11 : la colonne de gauche est l'animation initiale, et celle de droite la même animation déformée afin de franchir un obstacle.

Une autre méthode consiste en la modification globale des trajectoires dans le domaine fréquentiel, via une pondération favorisant certaines bandes de fréquence au détriment d'autres.

Se pose en tout cas la question du réalisme des mouvements obtenus, la dynamique du système ne pouvant être prise en compte de par l'approche purement géométrique retenue.

1.3.4 Approche Physique

Dynamique

La dynamique est l'étude du mouvement des corps en fonction de leurs caractéristiques physiques et des forces et moments qui leur sont appliqués. Il s'agit donc, tout comme la cinématique, d'une approche synthétique du mouvement ; mais ici les causes de ce mouvement sont considérées, d'où des résultats physiquement réalistes si la simulation est suffisamment précise.

Dans le cadre de l'animation, les objets considérés sont essentiellement des solides indéformables ; ils peuvent être libres ou, comme dans le cas de la cinématique, assemblés pour former des structures articulées. Dans ce dernier cas, les déplacements résultants seront calculés à partir des moments aux articulations ou des forces appliquées aux segments via des muscles simulés. Ces derniers ont été entre autres proposés par M. McKenna en 1990 [65], et utilisés plus récemment par l'équipe de D. Terzopoulos [106, 46] (la figure 1.12 représente la structure d'un poisson animé par cette dernière équipe).

Notons que les interactions entre objets (collisions et frottements) doivent être explicitement gérées via des approximations variées. J. Annunziato propose dans [4] une vue d'ensemble des approches possibles.

D'un point de vue mathématique, une introduction au formalisme nécessaire peut être trouvée dans [75]. De manière plus détaillée, McKenna et Zelter[65] décrivent les choix retenus pour leur simulateur, basé sur l'ABM (*Articulated Body Method*) et D. Thalmann [104] fournit une présentation de techniques de simulation courantes.

De manière plus aiguë encore que dans le cas de la cinématique, se pose le problème pour l'animateur de la spécification des moments adéquats pour la génération du mouvement désiré. De plus, la spécification des caractéristiques physiques du modèle dans le cas d'un corps humain devient délicate au-delà d'un certain niveau de précision. Ainsi, si l'on souhaite intégrer

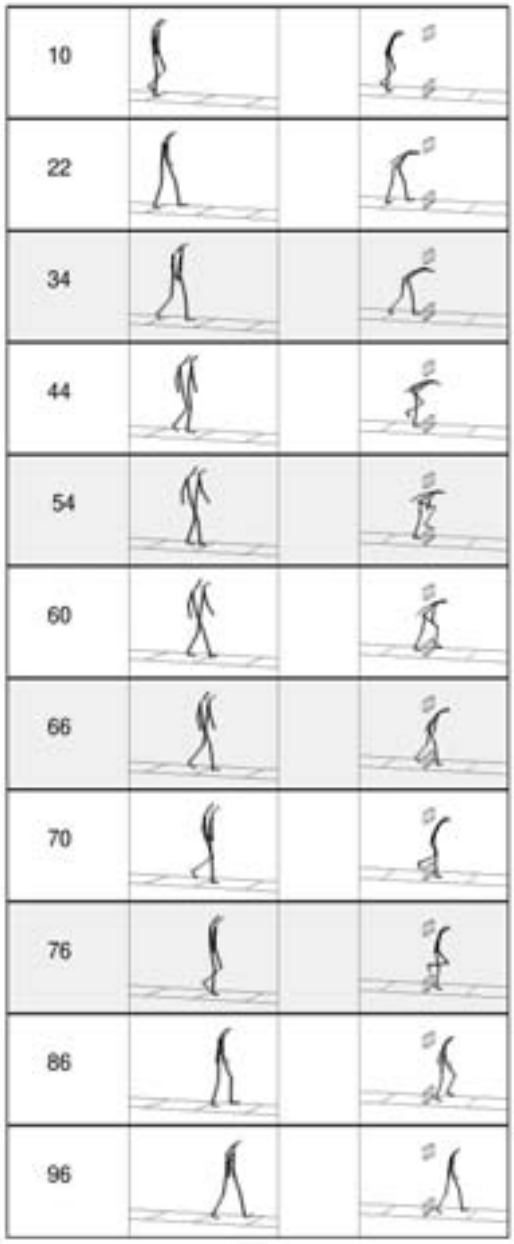


FIG. 1.11 – Warping

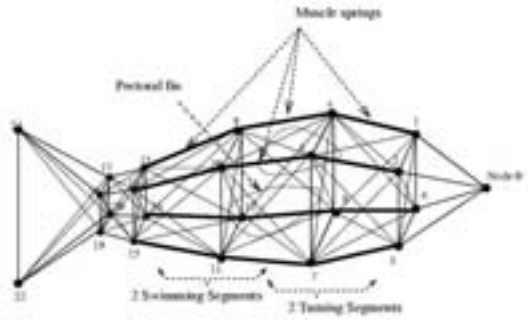


FIG. 1.12 – Structure à Muscles Simulés

l'élasticité des tissus [75], il va s'imposer la mise en place de déformateurs délicats à paramétrer (section 1.2.4) ou de méthodes de simulation du mouvement lourdes numériquement (section 1.2.5).

Dynamique Inverse

Le problème de la dynamique inverse peut être comparé à celui de la cinématique inverse.

Il s'agit ici de calculer les moments à appliquer aux degrés de liberté d'un modèle articulé. Ceux-ci doivent respecter au mieux des consignes exprimées sous forme de contraintes à respecter par certains des points, et éventuellement des articulations (intervalles de rotation limités) de ce modèle.

Un exemple spectaculaire d'application de ces outils est fourni par le projet AnyBody [50] de J. Rasmussen *et al* [88] qui vise à la production de données sur la fatigue musculaire dans le cadre de mouvements usuels. Un squelette est tout d'abord conçu, puis des muscles lui sont rattachés, suivant une approche similaire à celle décrite en sections 1.2.5 ou 1.3.4. L'utilisation d'une méthode de dynamique inverse permet alors, à partir des mouvements recherchés pour le modèle, d'obtenir les forces appliquées aux articulations, donc d'en déduire la contribution de chaque muscle à partir de la description de leurs points d'ancrage sur le squelette et de leurs caractéristiques.

Deux exemples de modèles ainsi animables sont disponibles figure 1.13.

Cette approche permet l'obtention de mouvements physiquement réalistes et précisément spécifiables ; cependant, un certain nombre de problèmes contrebalancent ces atouts. Les algorithmes utilisés sont relativement coûteux ; des problèmes de stabilité numérique se posent, par exemple dans le cas de chaînes fermées (telle celle formée par les deux jambes, le bassin et le sol quand les deux pieds y reposent). Mais surtout, les couples générés peuvent être arbitrairement grands, d'où un manque de réalisme d'un point de vue biomécanique.

Contraintes d'Espace-Temps

Une série d'algorithmes remplissant une fonction comparable à celle de la dynamique inverse est constituée par les *contraintes d'espace-temps* (*spacetime constraints*). Il s'agit ici de

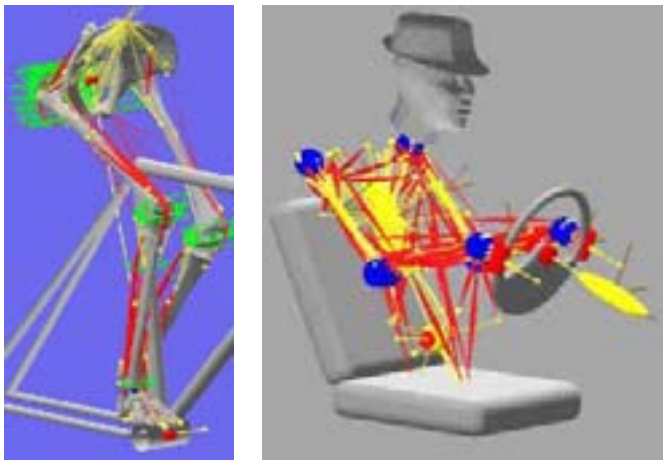


FIG. 1.13 – Modèles Inversibles dans AnyBody

calculer les moments nécessaires non plus à l'obtention d'une série de positions stables (approche image par image), mais plutôt l'ensemble de leurs variations permettant de produire les mouvements correspondants (approche dynamique).

Le problème est posé sous forme d'une série de positions que doit respecter le système au cours du temps (d'où le nom de cette technique), ainsi qu'une fonction à minimiser (en général l'énergie dépensée). Afin de simplifier la résolution, le temps est discrétisé : l'ensemble du mouvement à générer est représenté sous forme d'un ensemble grand mais fini de positions élémentaires (les vecteurs d'état du modèle). La résolution elle-même, recherchant le mouvement résultant en fonction des positions à respecter et des lois de la mécanique, est alors effectuée : on obtient ainsi le mouvement physiquement réaliste approximant au mieux les échantillons de trajectoire initialement fournis.

Les animations obtenues sont de très bonne qualité, mais la quantité d'inconnues à manipuler limite la durée des séquences ainsi simulables. Par ailleurs, ce mécanisme ne permet pas de gérer simplement les collisions, qui doivent être prévues *a priori* et intégrées sous forme de contraintes supplémentaires.

Afin d'alléger les limitations en terme de durée, Cohen [28] a proposé en 1992 un découpage du problème via des "fenêtres d'espace-temps" permettant de limiter pour une simulation donnée les degrés de liberté du modèle à manipuler et la durée de cette simulation. Par combinaison des solutions partielles associées à chaque fenêtre, la simulation globale est alors obtenue.

L'approche de Ngo et Marks exposée en 1993 [77], bien que se voulant proche de ces méthodes, nous a paru plus proche de la génération automatique de contrôleurs ; elle est donc décrite dans la section 1.3.6.

Ces techniques furent critiquées par [46] en 1995, qui leur reproche leur approche de la physique comme étant une contrainte à optimiser, donc dont on peut plus ou moins s'éloigner suivant les besoins. Le fait que le mécanisme de résolution utilisé impose des différentiations symboliques limite de plus leur utilisation à des modèles peu complexes.

En parallèle à ceci, M. van de Panne a proposé en 1990 [109] un algorithme plus souple nommé contrôleur dans l'espace des états (*state-space controller*). Il s'agit ici de spécifier un système physique et un but sous forme d'une position particulière dans l'espace de ses états (un espace abstrait dont le nombre de dimensions élevé permet, grâce à un seul point, de décrire une position, une vitesse et une date). Une structure de données est alors calculée : elle permet pour tout point dans un volume de l'espace des états du système englobant le but de calculer la trajectoire à suivre afin d'atteindre ce but. L'originalité de ce contrôleur est donc que, quelle que soit la position initiale du système, il saura y appliquer les forces et moments nécessaires pour le mener dans la situation désirée.

1.3.5 Moteurs

Les deux techniques décrites dans cette section et la suivante sont proches dans leurs objectifs. Elles se distinguent par leur cadre de travail : cinématique pour les *moteurs*, dynamique pour les *contrôleurs*. De plus, les moteurs sont souvent en boucle ouverte, alors que les contrôleurs exploitent des retours d'information (*feedbacks*). Les usages varient cependant suivant les auteurs, certains moteurs intégrant des contraintes physiques (dans une phase de correction des résultats cinématiques par exemple).

Il s'agit en tout cas de contrôler les degrés de liberté d'un modèle réaliste, donc en trop grand nombre pour être manipulables directement, afin d'obtenir la séquence motrice désirée.

Les moteurs se situent un niveau d'abstraction au-dessus des techniques de synthèse de trajectoires (section 1.3.2). Se basant sur ces dernières, ils proposent la génération de mouvements complets (c'est-à-dire le calcul des positions des effecteurs ou des valeurs angulaires) à partir d'un petit nombre de paramètres.

Leurs représentants les plus courants sont les moteurs de locomotion, qui via des paramètres tels que la longueur de la foulée, le type de démarche (course, avance prudente) voire l'humeur ou l'âge de la personne génèrent l'ensemble des informations nécessaires à l'animation du modèle (se référer par exemple aux travaux de R. Boulic en 1990 pour la marche [36] et à ceux d'A. Bruderlin en 1996 pour la course [25]).

Il existe aussi des moteurs de préhension (*grasping*) permettant d'atteindre puis de saisir un objet : citons les travaux de l'équipe de N.I. Badler en 1996 [34], ou ceux de R. Mas-Sanso et D. Thalmann en 1994 [63] illustrés figure 1.14.

1.3.6 Contrôleurs

Les contrôleurs peuvent être comparés aux moteurs, mais sont plutôt utilisés dans le cadre d'une simulation physique (section 1.3.4). Partant de la constatation que l'animateur n'est en général intéressé que par le mouvement résultant de la simulation, et pas par les moments en présence, les contrôleurs visent à calculer ces derniers afin d'obtenir le mouvement désiré. Ils permettent alors une prise en compte des interactions avec l'environnement (collisions et frictions) du fait d'une utilisation de la dynamique directe. Pour être efficaces, ils fonctionnent souvent en boucle fermée afin de réguler les mouvements, d'où la présence d'une notion de perception dans certains des algorithmes décrits ci-dessous. Une série d'exemples de contrôleurs, en particulier dans le cas de la locomotion humaine, est fournie par F. Multon dans [75].

FIG. 1.14 – Moteurs de Préhension

Conception Manuelle

Un exemple de conception d'un contrôleur pour la marche d'un insecte est décrit par M. McKenna dans [65] (déjà cité). Il s'agit d'une architecture en deux couches, l'une de *contrôle* chargée de fournir des comportements élémentaires via des forces adaptées (fournies par des muscles simulés), et l'autre de *coordination* organisant dans le temps ces comportements élémentaires afin d'obtenir une locomotion à proprement parler.

La couche de coordination est basée sur des oscillateurs construits par observation de la démarche d'insectes réels et générant régulièrement des impulsions ; mais des réflexes peuvent aussi déclencher ou retarder les mouvements élémentaires des pattes, afin d'adapter la démarche à l'environnement. Ainsi, un *step reflex* inhibe l'élévation de la patte lorsqu'elle est proche de son maximum, et un *elevator reflex* la fait monter plus haut qu'habituellement si elle rencontre un obstacle. Il y a ainsi des *feedbacks*, mais qui restent internes à la couche de contrôle. La couche de coordination fonctionne en boucle ouverte, ce qui d'après les auteurs est caractéristique d'une démarche classique (*patterned gait*) mais ne convient pas si une recherche précise d'appuis est rendue nécessaire par la nature du terrain (*free gait*). Notons que l'utilisation de muscles permet une certaine souplesse de mouvement, la locomotion pouvant s'adapter à de faibles variations de terrain alors que dans une approche cinématique, une prise en compte explicite de ses variations serait nécessaire pour recalculer la position des extrémités des pattes. De plus, l'utilisation d'une simulation dynamique permet une interaction avec l'environnement (qui peut appliquer des forces sur le modèle, contraignant ainsi ses déplacements).

Génération Automatique

Notons qu'en parallèle à la construction "manuelle" du contrôleur, certains auteurs proposent de le générer automatiquement. Ceci peut être effectué à partir de rien, via la spécifica-

tion d'une structure articulée et d'une fonction à optimiser (la distance parcourue, par exemple) ainsi que l'a proposé M. van de Panne en 1993 [108]. Cette optimisation du contrôleur peut éventuellement être effectuée en parallèle à celle de la structure physique, comme dans le cas des célèbres travaux présentés par K. Sims en 1994 [99] (ces derniers seront détaillés en section 1.6.3).

Une autre méthode, basée comme celle de Sims sur une optimisation par algorithmes génétiques, est utilisée par Ngo et Marks [77] (déjà cité) pour obtenir des démarches efficaces de la part de structures articulées simulées en deux dimensions. Ils lui associent pour cela un module comportemental basé sur un système de stimulus-réponse générant des moments, qui sera optimisé dans une simulation dynamique par l'algorithme génétique (on se réfèrera à la figure 1.15 pour un exemple de marche apprise dans ce cadre). Malgré la référence explicite aux contraintes d'espace-temps (section 1.3.4), il s'agit bien plutôt d'une génération automatique de contrôleur.

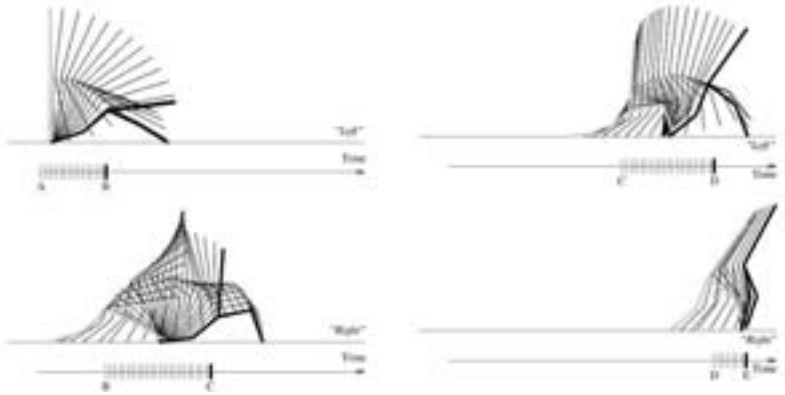


FIG. 1.15 – Apprentissage de la Marche

Si les résultats originaux obtenus peuvent être intéressants, ils ne sont pas aisés à utiliser. Il y a en effet peu de chance qu'ils correspondent à une démarche naturelle, et le fait de ne disposer que d'une fonction à optimiser ne permet pas aisément d'orienter la recherche.

Génération Hiérarchisée

Afin de répondre aux limitations précédentes, R. Grzeszczuk et D. Terzopoulos ont proposé en 1995 [46] une méthode de recherche de contrôleurs en plusieurs étapes qui permet l'obtention de locomotions animales réalistes.

Une fois la structure géométrique et musculaire de l'animal définie par l'animateur, un recuit simulé sur les paramètres d'un série de fonctions contrôlant les muscles permet l'obtention de locomotions optimisant une série de critères (suivi de trajectoire, mouvements coordonnés...). L'un d'entre eux est l'énergie dépensée : il semble qu'une caractéristique commune aux comportements locomoteurs est leur rendement énergétique élevé, ce qui permet à une optimisation incluant ce critère de mener à des mouvements réalistes.

Une étape d'optimisation sur les contrôleurs obtenus, dans le domaine fréquentiel ou spatial, permet ensuite de supprimer les composants des fonctions élémentaires de faible influence (celles-ci étant des splines composées de sommes pondérées de fonctions de base). Notons que dans le domaine fréquentiel, seules les fréquences dont l'amplitude est en-dessous d'un certain seuil sont supprimées : il n'y a donc pas nécessairement suppression des hautes fréquences et du bruit, qui semblent jouer un rôle important dans la crédibilité du résultat [85]. Un exemple des résultats obtenus est illustré figure 1.16.

FIG. 1.16 – Apprentissage de la Locomotion

Les contrôleurs, ramenés à un petit nombre de paramètres, sont alors utilisés comme composants élémentaires d'une seconde étape d'optimisation aboutissant à des tâches plus globales : suivis d'objets (représenté figure 1.17), sauts hors de l'eau... Bien qu'une perception de l'environnement est fournie aux animaux simulés afin qu'ils accomplissent leur tâche, aucun *feedback* proprioceptif n'est utilisé par leurs contrôleurs. Par ailleurs, les mouvements résultants semblent être essentiellement composés de la somme de quelques harmoniques pondérées, éventuellement en opposition de phase ou légèrement décalées.

Utilisation de Réseaux de Neurones

Les mêmes auteurs, afin de limiter le temps utilisé par la simulation physique, proposent *l'émulation* de la simulation du modèle via des réseaux de neurones [47]. L'approximation du comportement physique du modèle semble produire des résultats visuellement convaincants tout en étant beaucoup plus économique en temps de calcul. De plus, ils tirent parti de la

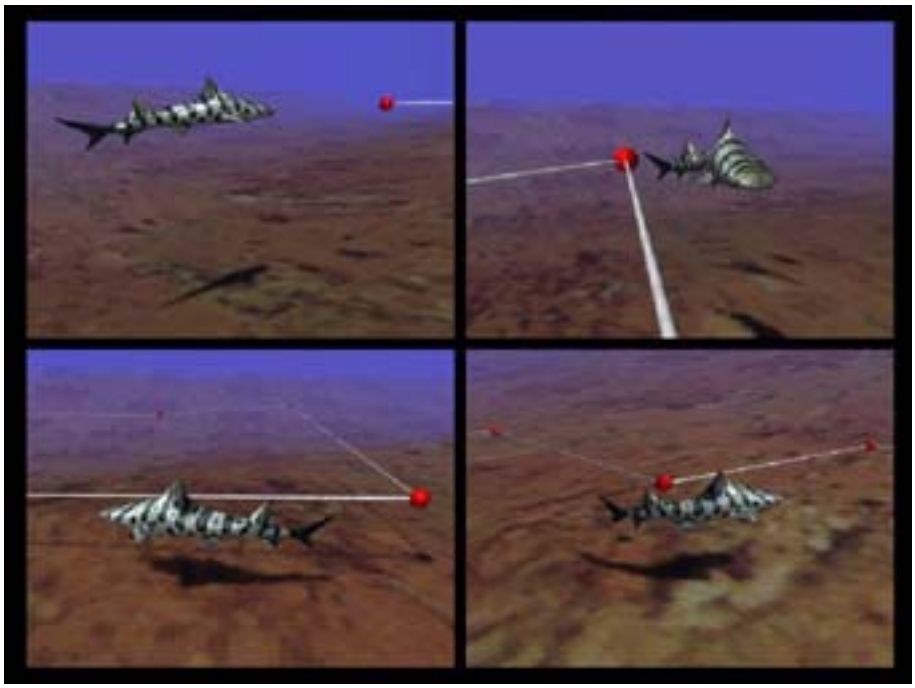


FIG. 1.17 – Comportement de suivi

structure du simulateur pour produire des contrôleurs générant des mouvements répondant à des contraintes d'animation, et ce d'autant plus que du fait des approximations retenues, une exploration par descente de gradient est possible. Finalement, un gain de l'ordre de deux ordres de grandeur est obtenu par rapport à la méthode de l'article de 1995 [46].

1.4 Animation de Niveau Tâche

1.4.1 Objectifs

Cette approche est délicate à classifier, puisque faisant l'interface entre les fonctionnalités motrices et le contrôle de l'entité. Contrairement aux moteurs d'animation, qui calculent à partir d'un petit nombre de paramètres souvent numériques les mouvements à effectuer, les modules de niveau tâche traduisent des consignes symboliques en des informations destinées à des moteurs ou à une utilisation directe.

Le plus souvent, plutôt qu'un langage symbolique, une série de modules spécialisés (locomotion, préhension) est utilisée. Une fois activés, ils peuvent interagir si nécessaire avec les éléments appropriés de l'environnement. Il s'agit donc d'abstraire les comportements moteurs en les décrivant sous une forme masquant les détails géométriques pour se concentrer sur le sens du geste. Dans le cas de la locomotion par exemple, il est, à l'extrême, possible de spécifier un point de destination, un outil de planification (voir ci-après) calculant l'ensemble de la séquence motrice permettant de l'atteindre. Afin que l'ensemble des mouvements soit réaliste,

le système de contrôle doit par ailleurs gérer l'enchaînement entre ceux-ci (*blending*), ainsi que les coordinations avec l'environnement (*feedbacks*) nécessaires à leur précision.

Remarquons qu'il ne s'agit pas d'un modèle décisionnel, puisque les actions à effectuer ne sont pas originaires du contrôle niveau tâche : un animateur ou un processus de sélection des actions à effectuer lui sont donc nécessaires.

Un exemple d'encapsulation dans un système de niveau tâche de comportements de bas niveau et de feedbacks est décrit par N. Badler [34] dans le cas de la préhension.

1.4.2 Techniques

Afin de faire la jonction entre des consignes générales et l'ensemble des actions nécessaires à leur satisfaction, il est possible d'utiliser des algorithmes de planification, généralement issus de la robotique ou de l'intelligence artificielle.

Dans le cas de la locomotion, ces algorithmes se basent sur une représentation discrétisée de l'environnement (sous forme de graphe ou de description symbolique) et produisent un chemin que l'acteur n'a plus qu'à suivre, par exemple en fournissant ces consignes à un moteur de locomotion (cf. section 1.3.5). Tout comme dans le cas de ces moteurs, des attributs peuvent infléchir le résultat (curiosité, flânerie, retard...) ; cela s'avère indispensable pour éviter que tous les humanoïdes aient la même démarche. Ces algorithmes peuvent par ailleurs tenir compte d'informations éventuellement disponibles dans l'environnement.

Cette exploitation d'informations devient indispensable dans le cas des algorithmes de préhension, un autre geste très étudié, qui, par essence, nécessite une perception minimale des objets environnants. Il s'agit ici de coordonner toute une série d'articulations tout en tenant compte des informations fournies par un sens du toucher simulé (détection de collisions).

1.5 Spécification d'Animations

Nous classifierons ici les différentes méthodes d'animation, indépendamment des techniques permettant de les mettre en œuvre. Pour uniformiser la présentation, nous considérerons que toute animation est par essence une boucle perception/décision/action, et formulerons chacune en ces termes.

1.5.1 Boucle Perception-Décision-Action

La notion de boucle perception / décision / action est issue de la psychologie comportementale. Elle n'a ainsi rien de spécifique à l'animation, mais elle sera utilisée régulièrement ici pour caractériser les techniques utilisées. Cette approche considère que l'être humain interagit avec l'environnement via des effecteurs, qui le modifient, et des perceptions issues de capteurs. Un système décisionnel les lie, choisissant les actions à effectuer en fonction des informations fournies par les capteurs et de ses connaissances.

Trois boucles de retour entre l'action et la perception sont alors identifiables :

– La plus directe est due aux actions sur l'environnement, leur résultante étant perçue par les organes des sens. Cela permet des mécanismes de *feed-back* dont la fonction est la convergence vers une situation désirée.

– Le corollaire à ces interactions est l'homéostasie : les actions du corps sur lui-même visent ici à stabiliser certaines variables internes. La régulation de la température corporelle ou de la posture en sont des exemples classiques. Les perceptions sont internes (*intéroception*), ainsi que les actions réflexes résultantes. Un autre exemple est le maintien de posture via la *proprioception* (perception de l'état des tendons et articulations).
– Afin d'améliorer la perception, des *comportements d'acquisition* peuvent être employés (orientation des yeux...) : l'action modifie directement les sensations, sans altération significative de l'environnement.

1.5.2 Scénarios

Un certain nombre d'outils commercialisés permettent la spécification de comportements d'humanoïdes, essentiellement via leurs mouvements. La technique la plus fréquente consiste en la construction d'un scénario qui guidera les mouvements du modèle. Rapportée au niveau locomoteur, cette approche dirige les déplacements de l'humanoïde via une spécification de trajectoires éventuellement associées à des mouvements variés.

Si cette approche offre une grande liberté de manœuvre au concepteur, elle s'avère fastidieuse, en particulier du fait de l'absence de prise en compte des interactions de l'humanoïde avec son environnement. Ainsi, l'évitement des collisions avec des objets ou d'autres humanoïdes doit être géré manuellement, ainsi que les interactions entre humanoïdes ou les manipulations d'objets.

En effet, aucune perception ou prise de décision n'existe au moment du déroulement de l'animation : tout le travail d'évaluation de la situation et de choix des actions à effectuer est fait par l'animateur, celui-ci visant à obtenir un ensemble d'évènements décidés antérieurement. Les types d'actions disponibles peuvent être variés : des primitives de haut niveau permettant une mise en place générale coexistent souvent avec des méthodes de retouche pouvant descendre jusqu'à des modifications manuelles de trajectoires (via une interface graphique le plus souvent).

Le côté fastidieux de ce travail est dû à la dissociation entre les actions (spécifiées via des interfaces variées) et la perception/prise de décision (effectuée durant la visualisation du résultat) : nous sommes en présence d'un mécanisme en boucle fermée, d'où la qualité du résultat final, mais son temps de latence important la rend pénible. Cette méthode de travail est donc finalement proche du développement logiciel, avec des mécanismes de modification et d'évaluation distincts.

Notons que des spécifications de plus haut niveau peuvent être exprimées via des outils d'animation de niveau tâche (cf. section 1.4) ; bien que le gain de temps soit alors considérable, les mouvements n'auront plus la même précision (sauf à les retoucher manuellement comme décrit ci-dessus). Cela reste en tout cas essentiellement un travail de planification d'une animation, qui ne pourra être réutilisée dans un autre contexte sans être remaniée.

1.5.3 Avatars

Plutôt qu'une animation reproductible et décidée de manière préliminaire à sa visualisation, la possibilité pour une personne d'interagir avec un environnement peut être recherchée. Pour

cela, il lui est possible d'y agir via un *avatar*, entité qu'il contrôle ; il dispose par ailleurs d'une vue de l'environnement proche de cette entité, que ce soit via une vue subjective (où l'aspect exploratoire est favorisé, d'où une utilisation pour les visites virtuelles) ou objective (souvent plus pratique si une interaction avec d'autres entités est nécessaire de par la vue d'ensemble qu'elle procure).

Cette approche se distingue de la précédente du fait de l'accent mis sur la perception. L'utilisateur n'est pas considéré comme un animateur, au sens d'un producteur d'une séquence motrice précise, mais comme un acteur à proprement parler, via un sentiment de participation plus ou moins important à la scène se déroulant sous ses yeux. La prise de décision est ainsi complètement reportée hors du système. Du fait de la contrainte d'interactivité, le registre d'actions est limité et de très haut niveau, afin de permettre une prise en main aisée par l'utilisateur. Une approche opposée, mais allant toujours dans le sens de la transparence de l'interface, est l'utilisation d'acquisition de mouvements (cf. section 1.3.1) : le corps est alors directement utilisé comme média d'interaction.

L'animation est plus plaisante du fait d'un temps de latence de la boucle perception / action plus court. La qualité du résultat en souffre cependant : les actions possibles sont moins précises (généralité des commandes de haut niveau, imprécision de l'acquisition de mouvements parfois) et ne peuvent être remaniées. Ce n'est cependant pas un problème, l'accent étant mis sur l'interaction ; si ces techniques sont utilisées pour la réalisation d'animations (pour des films, publicités), une approche classique de prises de vues répétées est utilisée, jusqu'à obtention d'un résultat satisfaisant.

Cette approche est utilisée pour l'exploration ou l'interaction dans des lieux reconstitués ou imaginaires, qualifiés de réalités virtuelles, mais aussi dans un grand nombre de jeux vidéos, où l'interaction peut se faire avec d'autres joueurs ou des acteurs simulés au comportement plus ou moins complexe. Ce dernier aspect sera détaillé dans la section suivante.

1.5.4 Réactivité : Animation Comportementale

Afin de dépasser les limitations en terme de temps de réponse et la réutilisabilité limitée de la spécification par scénario, ainsi que l'obligation de présence d'une personne dans le cas des avatars, il peut être souhaitable d'automatiser l'animation en faisant dépendre le comportement d'une entité d'évènements extérieurs et en particulier des interactions avec les objets et autres humanoïdes l'entourant. Cela va dans le sens d'une simplification du travail nécessaire pour obtenir une animation de qualité ; en effet, non seulement un certain nombre de contraintes et situations sont maintenant gérées automatiquement, via des *feedbacks* précis et rapides, mais de plus cette gestion permet une spécification du comportement via des consignes de plus haut niveau. Il est par exemple possible de nommer des lieux ou des objets, ainsi que des actions de l'humanoïde, puis d'utiliser ce vocabulaire pour décrire les actions à effectuer (*animation par objectifs*). Il est aussi possible d'agir sur les variables motivationnelles de l'acteur, afin d'orienter son comportement dans le sens désiré. Un bilan concernant la notion d'autonomie et sa simulation est disponible dans un article de B. M. Blumberg publié en 1997 [14].

Notons qu'il ne s'agit pas ici d'animation de niveau tâche, dans le sens où il s'agit ici de simuler des mécanismes de prise de décision, non de les interpréter en actes moteurs ; pour ceci, il est tout à fait possible de piloter des modules de niveau tâche par un module décisionnel.

La distinction n'est cependant pas toujours claire, le module décisionnel pouvant directement piloter les capacités motrices (animations de type "Reynolds", détaillées dans la section 1.6.2).

Le corrolaire de cette approche, qualifiée d'*animation comportementale*, est la nécessité de doter l'entité, alors souvent qualifiée d'*agent autonome*, de capacités de perception et de décision. Celles-ci lui permettent, en fonction des évènements extérieurs et de consignes éventuelles, de choisir les actions qu'il doit effectuer. Nous détaillerons ce dernier point dans la section suivante.

1.6 Simulation du Comportement

Les techniques décrites ici sont toutes utilisables dans le cadre de l'animation comportementale. Nous nous intéresserons tout d'abord aux contraintes perceptuelles et d'action des acteurs, en les comparant à celles des avatars. Nous décrirons ensuite les types d'outils de prise de décisions utilisés dans ce cadre.

1.6.1 Perceptions

Les perceptions sont indispensables à un acteur autonome : son comportement n'étant pas déterminé à l'avance, il lui est nécessaire d'utiliser un flux d'informations issu de son environnement afin de décider des comportements appropriés. Les informations les plus couramment disponibles sont la vue (perception à distance) et le toucher (détection de collisions), mais l'ouïe est aussi simulable via des modèles plus ou moins élaborés de propagation du son.

Capteurs

Il est possible de simuler des perceptions réalistes sous forme de capteurs simulés, comparables à ceux utilisés en robotique. Des informations sur la couleur, la distance, les sons sont alors disponibles, généralement sous une forme numérique. Les techniques d'animation réactives (section 1.6.2) tirent parti de ce type d'informations : dans des cas simples, elles reflètent correctement l'état de l'environnemen tout en étant simples à manipuler (se référer par exemple aux travaux de M. van de Panne cités précédemment [108]).

Se pose cependant le problème de l'interprétation de ces données : si elles sont réalistes, elles nécessitent un volume important de traitements pour en extraire les informations significatives à l'exécution de tâches complexes (*reconnaissance*).

Perception Symbolique

Pour contourner les difficultés posées par la reconnaissance, il est possible dans le cas d'un environnement simulé de fournir directement des données plus abstraites. Des informations de haut niveau sont en effet utilisées pour représenter l'environnement[1].

La notion de *perception symbolique* qualifie alors le fait d'extraire des données directement de cette description. Ces données sont qualifiées de symboliques, puisqu'elles sont exprimées

[1]Elles sont par exemple utilisées pour la visualisation de la scène, une vue permettant d'y naviguer n'étant qu'une projection 2D de cette description depuis un point de vue donné.

dans un formalisme de plus haut niveau que des données issues de capteurs. Tous les intermédiaires sont possibles, jusqu'à un accès direct à la base de données de la scène.

Des approches hybrides sont souvent retenues : la scène peut par exemple être projetée sur une "rétine artificielle", d'où le terme de *vision synthétique*. L'acteur dispose, pour chaque point d'une grille, de la couleur, l'intensité lumineuse du point perçu (projection classique), et surtout de sa distance à l'acteur et de l'identification de l'objet auquel ce point appartient. Il est ainsi possible à l'acteur, à partir de ces informations de haut niveau, de se contruire facilement une carte de son environnement sans avoir à identifier des objets ou des distances.

Le réalisme est cependant dans une certaine mesure conservé du fait qu'il s'agit bien d'une perception subjective, contrairement à un accès direct à la base de données globale qui lui fournirait des informations dont il ne peut disposer (du fait d'occlusions ou d'une portée limitée des perceptions).

Un outil permettant la génération de ce type de flux perceptifs a été décrit par C. Bordeux en 1999 [16] (des exemples de capteurs de haut niveau de ce type sont illustrés figure 1.18), et un exemple de système perceptif simple est disponible dans les travaux de X. Tu publiés en 1994 [106]. Un système plus complexe incluant l'apprentissage d'une carte et la planification de déplacements est décrit par Hansrudi Noser *et al* [78].

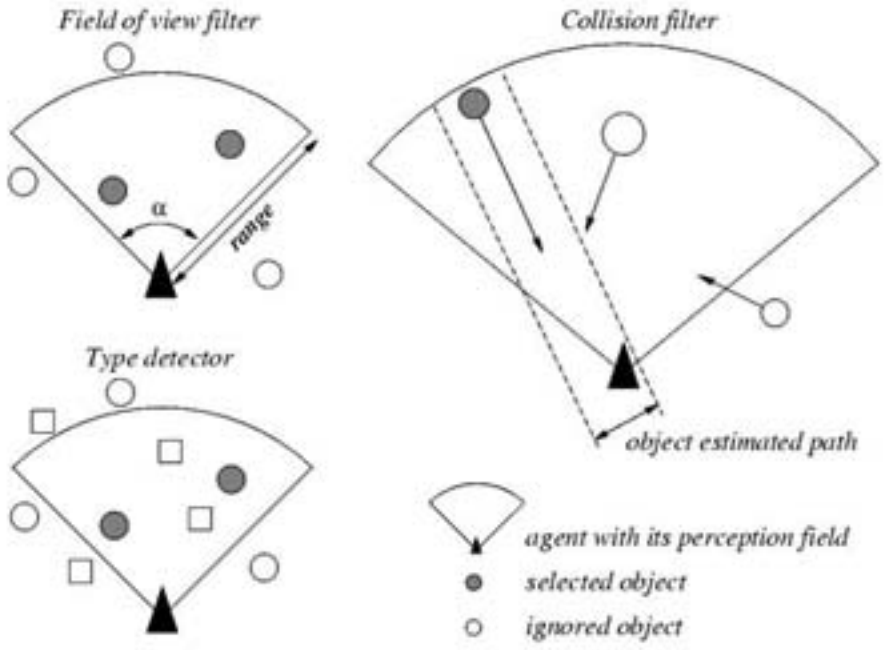


FIG. 1.18 – Capteurs Symboliques

B. M. Blumberg [15] remarque que l'approche par vision synthétique est problablement plus souple que celle par interrogation de base de données, car elle ne nécessite pas des objets rencontrés qu'ils répondent à des requêtes précises ; cette technique est donc probablement plus simple à intégrer à un système de rendu d'environnement traditionnel.

Approche Écologique

Gibson [42, 43] a défini une approche écologique de la psychologie de la perception, mettant l'accent sur les significations de l'environnement pour un agent : un agent s'y comporte comme si des informations, qu'il qualifie d'*affordances*, étaient disponibles dans l'environnement et directement utilisables pour y agir. Il s'agit ici de décrire une adéquation entre le potentiel de l'acteur et les opportunités environnementales permettant l'action. Des stéréotypes d'action sont ainsi associables aux objets, aux êtres et aux lieux.

Cette approche est utilisée en simulation via la définition des éléments perçus sous forme d'objets (dans le sens concret mais aussi informatique du terme) auxquels sont classiquement associés des attributs, mais aussi des méthodes indiquant les opportunités d'actions qu'ils permettent ; libre alors à l'acteur d'exploiter ces possibilités.

Une autre idée importante est l'orientation des perceptions, et du type d'informations extraites en fonction de la tâche en cours. Par exemple, une alternance de lancers de rayons pour capturer des informations spécifiques à une tâche courante, puis l'évaluation des données capturées permet la navigation dans un environnement 3D et l'utilisation des affordances (qui permettent de définir l'intérêt d'un objet en fonction de l'état courant). Ces informations symboliques couplées à un modèle de type attraction / répulsion permettent d'orienter l'acteur en fonction de sa motivation.

Une critique pouvant être faite aux affordances est qu'elles placent une contrainte très forte sur l'autonomie comportementale de l'acteur. Leur présence ne la limite pas en elle-même, si elles sont considérées comme un simple déplacement des capacités de l'acteur dans l'environnement : celui-ci peut choisir ou non d'en tirer parti au même titre que d'une de ses compétences. Cependant, le fait de trop dépendre de ce mécanisme limite les capacités cognitives et adaptatives de l'acteur, du fait qu'elles imposent une interprétation donnée de l'objet. Prenons l'exemple d'un escalier : un agent s'en approchant et désirant accéder à un niveau supérieur peut choisir d'utiliser une affordance "gravir" fournie par celui-ci (une séquence de *keyframes* par exemple). Pour un observateur extérieur, le comportement est adapté dans sa sélection et son exécution (l'affordance ayant été conçue pour cela).

Cependant, reprenons maintenant une situation correspondant à un test d'intelligence courant : plaçons l'acteur précédent dans une pièce contenant une série de boîtes. Il est possible d'empiler les boîtes afin de les escalader ; mais l'affordance "gravir" n'est ici pas présente, aucune caisse prise isolément ne permettant cette ascension. C'est la structure résultant de leur empilement qui le permet, et celle-ci n'est présente que dynamiquement, lorsqu'elle est construite.

Le problème vient donc de ce que les affordances sont une pré-interprétation du monde : le risque est alors d'imposer la même perception de la situation à tous les agents présents, et de ne pas leur fournir de mécanismes de reconnaissance de situations dans lesquelles ils pourraient agir (ce qui permettrait des comportements opportunistes dans des situations similaires à l'exemple précédent). L'usage d'affordances est donc inadapté à des environnements trop complexes (car toutes les actions possibles ne peuvent y être prévues) ou dynamiques (car les opportunités y évoluent, donc ne peuvent plus être précompilées).

1.6.2 Types de Prises de Décision

Approche Réactive

L'approche réactive vise à l'exhibition de comportements intéressants via des techniques simples. L'accent est mis sur une utilisation la plus simple possible des informations perçues, sans tenter de les interpréter ou de les rapporter à un modèle du monde ; par exemple, un ensemble de règles permettant l'évitement de collisions est décrit dans les célèbres travaux de C. W. Reynolds publiés en 1988 [91]. Cette combinaison de mécanismes élémentaires mène à des comportements de vol en groupe (*flocking*) et d'évitements d'obstacles assez spectaculaires (figure 1.19) en ce qu'ils paraissent coordonnés.

FIG. 1.19 – Vol d'Oiseaux

Cette simplicité permet des réactions rapides et une intégration d'un volume de données parfois importantes, mais ne permet finalement que des comportements simples. Cette approche prend sens lorsqu'elle est appliquée à une multitude d'individus appliquant ces règles. Un comportement cohérent du groupe émergera alors de l'interaction de cet ensemble de règles pourtant individuelles (pas de coordination explicite). Celles-ci seront adaptées aux types de comportements attendus et à l'environnement, en prenant par exemple en compte la dynamique du système lorsque celle-ci limite la rapidité d'action de l'individu, tel que décrit par D. Brogan et J. K. Hodgins en 1997 [18].

Des comportements intéressants pouvant être obtenus via l'utilisation de programmes simples, ceux-ci peuvent être générés automatiquement via l'utilisation d'un algorithme d'optimisation, comme dans le cas des travaux de C. W. Reynolds sur la coévolution publiés en 1994 [92] ; des comportements intéressants sont ainsi obtenus (se référer à la section 1.6.3 pour une présentation plus détaillée). L'auteur considère cependant que cette technique serait beaucoup moins efficace dans le cas du contrôle d'entités plus complexes que les "tortues" qu'il utilise.

Approche Ethologique

Lors de la simulation d'animaux, il est possible d'utiliser des modèles comportementaux compatibles avec les mécanismes de sélection de comportements chez les êtres vivants (section E.5). Ainsi, B. M. Blumberg [15] propose d'interagir avec un chien virtuel (figure 1.20) via la manipulation de variables motivationnelles influant sur son comportement, d'où un acteur capable d'autonomie mais pouvant être influencé suivant les vœux de l'animateur (*directability*).

L'auteur propose plus généralement dans cet article des mécanismes d'interaction avec cet animal aux différents niveaux de son architecture (motivationnel, tâche et moteur), ce qui

FIG. 1.20 – Contrôle d'un Animal Virtuel

permet une orientation aisée de l'acteur (figure 1.21). Le niveau motivationnel est de manière originale implémenté suivant des idées inspirées de Minkski [71], c'est-à-dire un ensemble d'agents ("entités orientées-but") en compétition pour le contrôle de la créature. Notons que la décomposition n'est cependant pas complètement comportementale, au sens de la section 2.3.2, certaines fonctionnalités pouvant être partagées entre plusieurs modules.

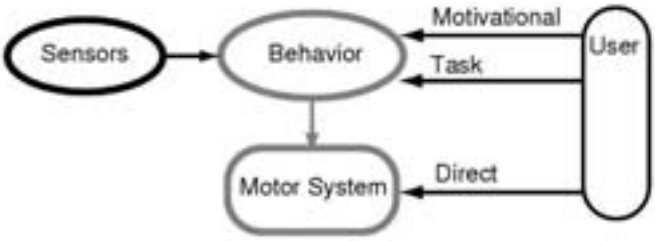


FIG. 1.21 – Contrôle Multi-Niveaux d'un Acteur

Un exemple totalement autonome est fourni dans les travaux de X. Tu ([106], déjà citée en 1.12 et 1.6.1), qui à travers une simulation physique contrôlée par un système de règles et de variables comportementales (au sens de celles décrites en section E.5.5) obtient des comportements d'interaction variés entre des poissons de plusieurs espèces. L'accent est ici mis sur la qualité de la simulation et sur le réalisme du comportement.

Approche Cognitive

Les techniques d'animation de type cognitif visent à une modélisation du comportement humain, tel qu'étudié par les sciences cognitives. Les modèles obtenus présentent l'avantage d'être relativement simples à concevoir et interpréter, car issus de ce que nous apprend l'étude des prises de décisions humaines.

De plus, ils permettent l'intégration de connaissances sur le domaine : dans le cas des milieux urbains, il est possible de constituer des bases de règles caractéristiques de ce type d'environnement, d'où des comportements d'anticipation ou de coordination qui, s'ils sont compatibles avec ceux des autres acteurs (interactions piétons / voitures par exemple) et couplés à des affordances, permettent la simulation de comportements complexes crédibles [105].

Nous ne détaillerons pas ces *comportements cognitifs*, qui mettent l'accent sur la rationalité des actions plutôt que sur leurs modalités d'exécution précises. En effet, ce dernier aspect est crucial en animation physique, pour laquelle une décision de haut niveau de type "se rendre en un lieu" est insuffisante. Il est cependant tout à fait concevable d'interfacer des outils d'animation physique orientés tâches (tels ceux décrits en section 1.3.6) avec des mécanismes cognitifs de prise de décision.

1.6.3 Implémentation de Modèles Décisionnels

Les outils décrits ici permettent l'implémentation des approches décrites dans la section précédente. Certains sont plus appropriés aux modèles réactifs (réseaux de neurones, scripts simples, systèmes à règles) et d'autres aux modèles cognitifs (automates, systèmes d'inférences). Notons que des modèles en couches sont souvent utilisés (motrice, orientée tâche et éventuellement motivationnelle ou planificatrice), chacune pouvant être implémentée dans un formalisme différent comme le propose B. M. Blumberg avec son chien simulé ([15], déjà cité).

La plupart des techniques citées permettent généralement l'expression d'un seul comportement à la fois ; les automates paraissent par contre mieux armés pour la mise en place de parallélismes et des mécanismes de gestion de conflits qu'ils nécessitent.

Scripts

Les scripts désignent, de manière générale, une forme de spécification du comportement sous une forme textuelle. Le niveau d'abstraction peut varier, mais il s'agit en général d'une pratique proche de la programmation via les langages "classiques" (impératifs, éventuellement objets). Un exemple classique de ce type d'outils, intégrant par ailleurs une forme de programmation par acteurs (objets dont le code est exécuté à chaque calcul d'image), a été décrit dès 1982 par C. W. Reynolds [90].

En principe, sous réserve que le langage utilisé soit Turing-puissant, toute autre approche sera en dernière analyse réductible à un script précis. Cette puissance d'expression a cependant comme corrolaire la nécessité d'une spécification manuelle de l'ensemble des comportements désirés. En général, toute abstraction (gestion des collisions par exemple) devra être explicitement développée par l'animateur.

Deux problèmes se posent : un langage informatique n'est pas nécessairement l'outil le plus adapté à l'expression de comportements complexes, en particulier lorsque plusieurs acteurs

sont en interaction, que diverses tâches doivent être effectuées en parallèle, etc. Par ailleurs, un animateur ne peut, pour exploiter pleinement cet outil, faire l'économie d'un apprentissage de la programmation, et ce même si le langage et les bibliothèques associées sont conçus pour lui faciliter les choses. Il se pose ainsi, en marge d'une puissance d'expression théorique, la question de l'adéquation de cette technique au domaine considéré et à ses utilisateurs.

En pratique, il s'avère donc que les scripts sont soit utilisés pour des animations d'objets suivant des règles relativement simples : croissance de plantes, cristaux (on parle alors d'*animation procédurale*), soit pour une construction au-dessus de ces langages généralistes d'outils plus spécialisés mais plus simples d'utilisation. Un exemple de réalisation de ce type est illustré 1.22 : elle a été produite via le logiciel PlantVR [27] écrit par M. Somporn Chuai-Aree [94].

FIG. 1.22 – Génération de Plantes par Script

À titre d'exemple, K. Perlin et A. Goldberg ont décrit en 1996 dans [85] leur plate-forme d'animation, Improv, basée sur une série de scripts utilisant un formalisme proche de l'anglais. Ce formalisme permet de définir le comportement d'un acteur autonome humanoïde en fonction de variables, de conditions sur l'environnement (y compris les autres acteurs ou avatars présents) ou de décisions éventuelles d'un manipulateur, tout ceci via une interface graphique contextuelle spécifiée par l'auteur. Ces scripts sont de haut niveau (la puissance d'expression semble être de l'ordre des automates hiérarchisés), et fournissent des primitives adaptées au domaine, permettant de lancer des scripts en parallèle, de sélectionner des actions avec une certaine probabilité, de les rendre exclusives, etc.

La figure 1.23 présente un exemple d'interaction entre acteurs dans le cadre d'Improv : l'un des personnages raconte une plaisanterie, appréciée par un acteur mais pas par l'autre (qui est contrôlé par un utilisateur).

A contrario, il est possible de définir des langages très simples, ceci syntaxiquement et

FIG. 1.23 – Acteurs Animés dans Improv

du point de vue des opérateurs disponibles, afin de générer des scripts par un processus automatique. Reynolds, dans un article de 1994 [92], utilise par exemple un langage minimal de type LISP pour contrôler des objets 2D de type "tortue" dans le cadre d'un jeu de poursuite. L'originalité de son approche vient d'un processus de compétition entre les programmes permettant d'évaluer leurs performances et de les soumettre à un algorithme de programmation génétique. Des programmes de plus en plus performants sont ainsi générés, la coévolution étant manifeste dans l'apparition de stratégies visant à contrecarrer celles déjà existantes. Exceptée une fonction d'évaluation très générique du résultat des compétitions, aucune influence particulière n'est nécessaire (en particulier, il n'est pas nécessaire de *bootstrapper* le système avec un contrôleur efficace). Un point intéressant est que la notion d'optimalité est ici relative : même des contrôleurs peu efficaces d'un point de vue théorique peuvent être sélectionnés s'ils exploitent des faiblesses de leurs concurrents. Un exemple de comportements de ce type est illustré figure 1.24.

Systèmes à Règles

Les systèmes à règles ne doivent pas être confondus avec les systèmes d'inférence, malgré un vocabulaire proche. Ces systèmes décrivent le comportement sous forme d'un petit nombre de règles, leur utilisation résultant en un comportement global.

Une technique de choix du comportement est l'arbre de décision : une règle simple est associée à chaque nœud, et les feuilles correspondent à des comportements élémentaires (fuite, poursuite...) ; un parcours de cet arbre permet ainsi d'activer le comportement le plus approprié. Cette technique est par exemple utilisée par X. Tu pour simuler le comportement de poissons ([106], déjà cité : voir la figure 1.25).

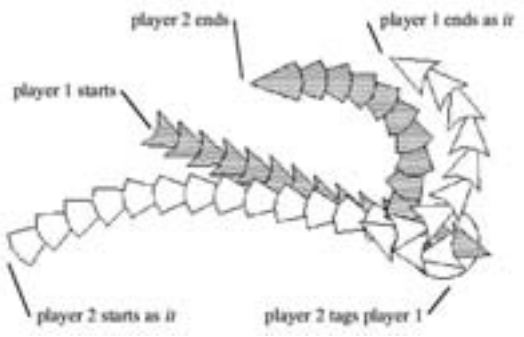


FIG. 1.24 – Comportement de Poursuite Appris

Reynolds ([91], voir la section 1.6.2) fournit un exemple classique d'utilisation de ces techniques en animation comportementale, des règles simples (éviter les collisions, se rapprocher du centre de gravité du groupe...) permettant, par interaction entre un ensemble d'animaux les utilisant, la simulation de nuées d'oiseaux ou de bancs de poissons. Chacun utilise les perceptions qui lui sont propres pour déclencher des comportements simples mais réalistes. L'intégration entre les comportements est effectuée par une simple somme vectorielle.

Cette technique est simple à mettre en place du fait d'un niveau d'abstraction satisfaisant, mais il est délicat d'étendre ces systèmes à des comportements complexes, qui nécessiteraient un nombre important de règles délicates à coordonner.

Réseaux de Calcul

Les techniques décrites ici ne doivent pas être confondues avec les réseaux de neurones artificiels étudiés par le connexionnisme. Il s'agit ici de relier les capteurs à des actionneurs via un réseau de nœuds modifiant et se transmettant l'information. Les comportements générés étant proches des instincts ou réflexes animaux, cette approche est qualifiée de *stimulus-réponse*.

Elle permet l'implémentation de comportements locomoteurs simples mais robustes ; ils manquent cependant d'abstraction, et nécessitent une conception ad hoc des réseaux : le "programme" est entièrement défini par les connexions, les opérations utilisées et les poids des connexions. Modifier des comportements nécessite de recalculer ces poids, même si l'utilisation de techniques d'optimisation permet une forme d'automatisation de ce processus.

▷ **Optimisation de Réseaux** L'article de M. Van de Panne de 1993 [108] décrivant les *Sensor Actuator Networks* (SAN) est original de par son approche synthétique : une fois une créature articulée, ses capteurs et la topologie de son réseau de contrôle définis, des ensembles de poids de connexions sont tirés aléatoirement, et les mouvements obtenus sont évalués afin de retenir les plus intéressants (en fonction de la distance parcourue par exemple). Le résultat obtenu par l'interaction de la simulation physique et d'oscillations entretenues dans le réseau (non-linéaire de par des phénomènes d'hystérésis, de retards et de feedbacks) est une série périodique de mouvements locomoteurs. Ceux-ci n'auraient souvent pas été trouvés via une démarche *top* -

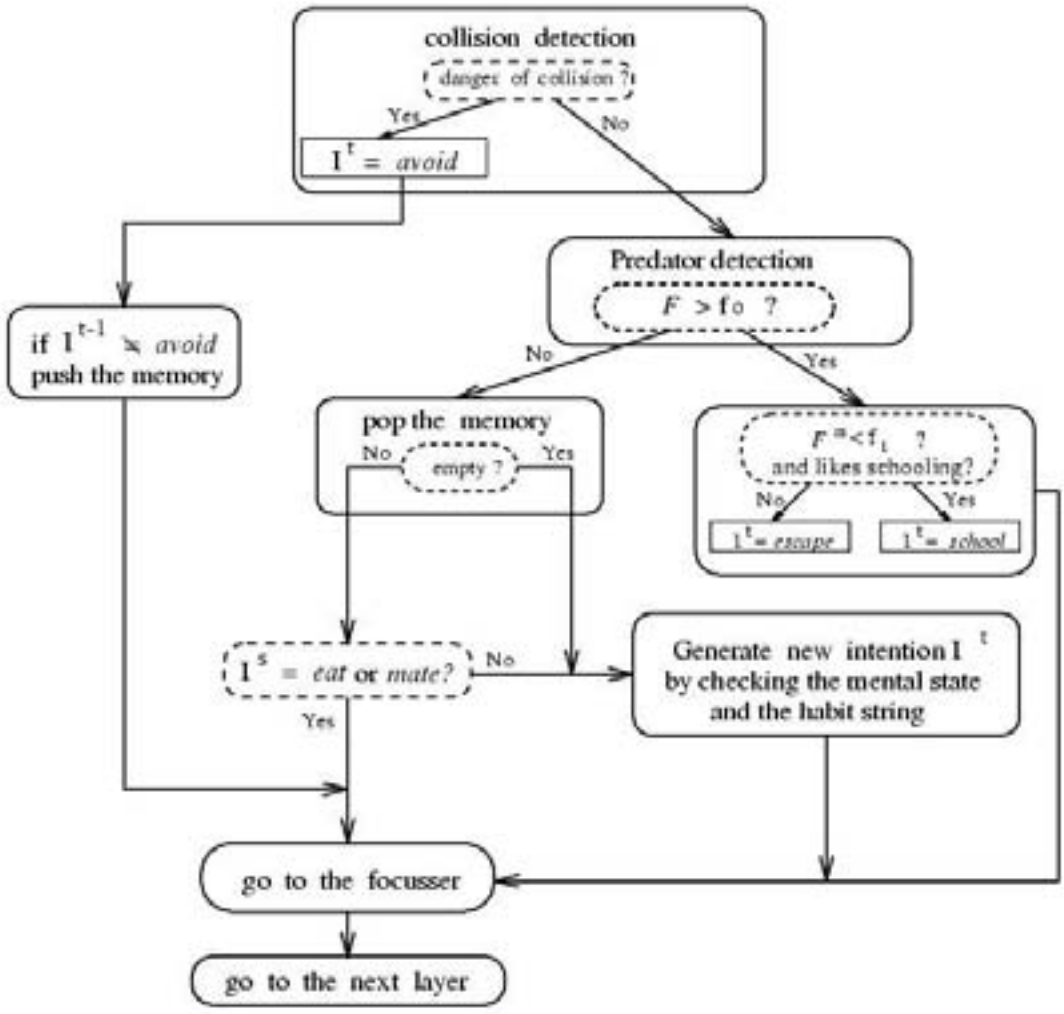


FIG. 1.25 – Règles Comportementales d'un Poisson

bottom, en particulier pour les structures les plus simples.

Notons qu'il ne s'agit pas d'une approche par apprentissage : les poids sont tirés une fois pour toute, mais seul un petit nombre de paramètres caractérisant les capteurs, les effecteurs et certaines caractéristiques du réseau sont ensuite soumis à optimisation afin d'améliorer la "démarche" obtenus. Les auteurs précisent que du fait de la non-linéarité massive du système (indispensable à la qualité des résultats obtenus), les modifications des poids ont trop ou trop peu d'effet pour pouvoir être soumises à une optimisation. Cette technique permet cependant de tirer parti de phénomènes telle la friction, difficile à exploiter par des algorithmes d'optimisation de trajectoires.

Ce système est intéressant par la simplicité des réseaux obtenus d'où découle une certaine généricité (par exemple, des créatures optimisées pour la locomotion sur terrain plat peuvent parcourir des zones légèrement pentues ou irrégulières). Par ailleurs, aucune tentative de reconstitution de l'état du système via les capteurs n'est explicitement effectuée : le mouvement provient d'oscillations entretenues par l'interaction entre le réseau et la structure de la créature. Cependant, la conception du système physique et des capteurs, ainsi que le choix d'une série de constantes doivent être effectués en fonction d'une préconception du type de résultats pouvant être obtenus ; par ailleurs, au-delà d'environ 6 segments le coût de la simulation et la raréfaction des contrôleurs efficaces et esthétiques deviennent problématiques. Les auteurs pensent qu'une coévolution du contrôleur et de la structure physique permettrait d'augmenter graduellement la complexité des systèmes obtenus.

▷ **Optimisation de Réseaux et Structures** Cette approche a été suivie par Karl Sims [99], qui synthétise des créatures mobiles dans des environnements terrestres et aquatiques physiquement réalistes. Ici, il y a effectivement évolution de la structure de la créature en parallèle à celle de son contrôleur ; elle est constituée de segments auxquels sont associés un réseau de contrôle, le tout construit via un processus de développement similaire aux L-systèmes afin de favoriser leur réutilisation dans une même créature (figure 1.26). Une fois évaluées via des critères similaires à ceux de l'article précédent, ces créatures sont soumises à une sélection et un mécanisme de croisement de type algorithme génétique, d'où une optimisation au fil des générations (la figure 1.27 en présente quelques exemples dans le cas d'une optimisation pour la marche).

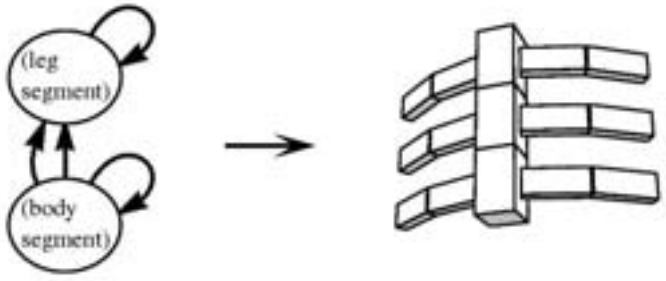


FIG. 1.26 – Génotype et Phénotype

Ici, la physique du système est beaucoup plus lourde à simuler que dans l'exemple précédent ; cependant, des résultats intéressants sont obtenus, et le nombre de segments semble pouvoir atteindre la dizaine ou plus sans difficultés. L'auteur note que, si les réseaux de nœuds obtenus sont complexes et délicats à analyser, cela ne pose pas de problème puisque leur compréhension n'est pas nécessaire (le choix d'une fonction d'évaluation suffit). Il remarque par ailleurs qu'une certaine qualité de la simulation est nécessaire, les créatures tirant parti de la moindre invraisemblance physique (absence de conservation de l'énergie...).

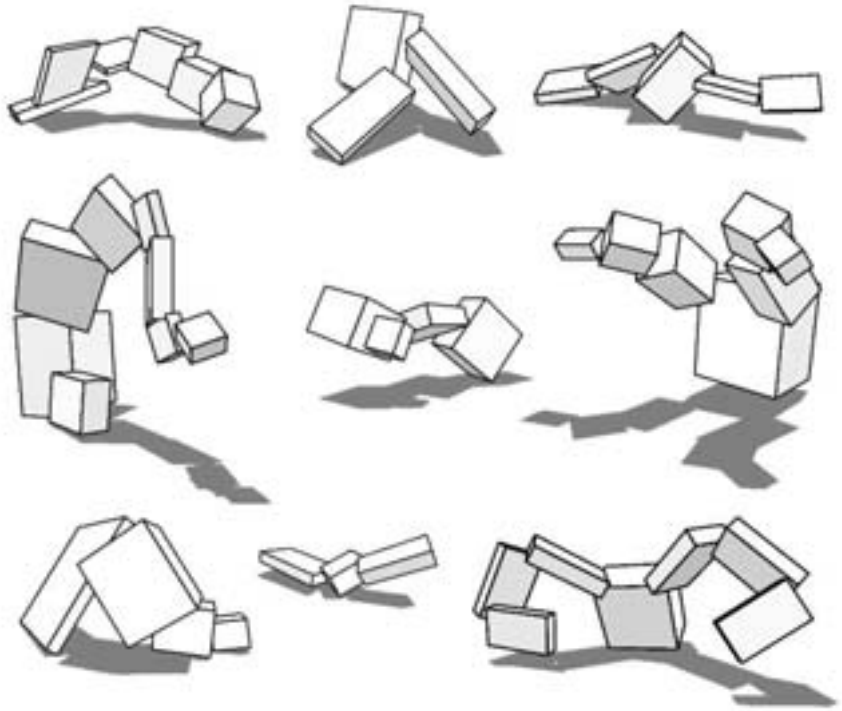


FIG. 1.27 – Créatures Évoluées pour la Marche

Planification (Systèmes Experts)

Si l'hypothèse est faite que le monde est descriptible sous une forme symbolique, et qu'un ensemble de lois le concernant est disponible, des décisions peuvent être prises via un système d'inférences de type "système expert". Il est alors possible de fournir des buts de manière préliminaire ou interactive au système, qui tentera de les atteindre en s'appuyant éventuellement sur des outils d'animation de niveau tâche (cf. section 1.4.2).

Les mécanismes gouvernant l'évolution de l'environnement sont ici considérés comme prédictibles et connus, ce qui permet au système de raisonner sur les conséquences de ses actes, pour ainsi trouver la séquence d'actions menant à un état du monde compatible avec son but. Il suffit alors d'exécuter l'ensemble de ces actions (le *plan*) pour obtenir le résultat désiré.

Cette décomposition en deux tâches, le raisonnement sur le monde puis l'exécution d'un plan, est caractéristique de cette approche dite de *planification* (cf. section 2.2.4).

Un défaut classique des systèmes d'inférences est la difficulté à gérer des comportements multiples, ainsi que, par nature, la prise en compte d'environnements difficilement prévisibles. Ces techniques ne sont pas opportunistes, au sens où elles ne savent pas tirer parti d'une modification de l'environnement au cours de l'exécution d'un plan. Pire, cela invalide celui-ci en général, ce qui nécessite une replanification.

Par ailleurs, ces systèmes sont délicats à étendre : l'ajout d'un trop grand nombre de connaissances mène en général à des incohérences dans la base de règles difficiles à gérer.

Automates

Ce terme décrit une série de techniques présentant l'avantage de permettre facilement l'expression de buts multiples et la gestion de leurs conflits ; de par cette multiplicité, l'opportunisme (au sens de la capacité à déclencher ou modifier un comportement en fonction d'un stimulus inattendu) y est plus facile à implémenter. Nous allons en décrire quelques exemples.

▷ **SCA et réseaux d'automates parallèles** Ce système d'animation, décrit pat B. Webber et N. Badler en 1995 [111], utilise deux niveaux de contrôle pour simuler le comportement d'humanoïdes.

D'une part, une couche réactive basée sur des boucles *sense - control - action* (SCA) assure les comportements de bas niveau et fournit des primitives de niveau tâche (regarder vers, aller à...). Cet ensemble de comportements est représenté par un réseau de nœuds de perception, de contrôle et d'action : les informations circulant entre eux sont des flots de réels, et un comportement peut être altéré via une série de paramètres. L'auteur précise que, contrairement aux architectures de *subsumption* de Brooks (section 2.4.1), des nœuds peuvent être partagés et l'arbitrage peut alors s'effecter non seulement au niveau des effecteurs, mais aussi du contrôle. Le choix d'un flot de réels permet l'utilisation d'algorithmes d'apprentissage non supervisés et d'algorithmes de minimisation au niveau des nœuds de contrôle.

D'autre part, un niveau supérieur utilise des *parallel transition networks* (PaT-Nets). Il s'agit de réseaux d'automates disposant de capacités de passage de messages et de sémaphores. Leurs arcs sont traversés lorsqu'une condition locale au PaT-Net ou environnementale est vérifiée, et chaque état peut être associé à des solveurs externes, à des comportements de bas niveau ou invoquer d'autres PaT-Net (via des messages ou en en instanciant de nouveaux et en attendant leurs résultats).

Ils prennent en charge le processus décisionnel à proprement parler, à savoir le type d'action à déclencher en fonction de l'environnement, le *monitoring* des actions en cours et de contraintes à respecter, leur synchronisation... Il s'agit ainsi d'une structure à deux couches similaire à celle décrite en section 1.5.4.

▷ **Piles d'Automates** Cette technique est utilisée par l'équipe de D. Thalmann (voir par exemple l'article de H. Noser *et al*, 1995 [78]) pour la simulation d'acteurs autonomes en réalité virtuelle. Une pile, initialisée par l'utilisateur avant simulation, contient des actions représentées par des automates hiérarchisables si nécessaire (une action pouvant se décomposer

en sous-actions). Des évènements extérieurs peuvent influer sur le comportement de l'automate en suspendant (dépilant) éventuellement certaines actions au profit d'autres.

▷ **Hiérarchie d'Automates Parallèles** Les *hierarchical concurrent state machines* (HCSM) sont utilisées par J. Cremer pour la simulation du comportement de conducteurs automobiles (il les détaille dans un article de 1995 [30]). Une série d'automates permet la prise en compte d'évènements et de contraintes variés, provenant du milieu et d'attributs caractéristiques au conducteur (humeur, type de conduite..). Une description plus précise de la structuration des automates utilisés est disponible dans un article publié en 1994 [2].

Le modèle *hierarchical parallel transition systems* de S. Donikian (une présentation récente en est disponible [31]) propose un cadre de modélisation du comportement compatible avec le temps-réel et les données psychologiques ; il permet la gestion du temps et le transfert de messages entre automates. Ils ont eux aussi été utilisés pour la simulation de conduites, et permettent l'intégration de l'itinéraire à suivre, du respect de la signalisation et des évènements dus aux autres acteurs (évitement). La gestion des conflits dans cette simulation est effectuée via la définition de priorités entre comportements.

1.7 Discussion

1.7.1 Niveaux de Contrôle

Une classification possible des techniques précédentes consiste à les échelonner du "bas niveau" au "haut niveau". Le premier extrême expose l'essentiel des paramètres d'un modèle d'animation, ce qui permet un contrôle complet des résultats obtenus, mais au prix d'un travail important et fastidieux du fait de la grande quantité de données à manipuler, de la nécessité d'une bonne compréhension de ce mécanisme d'animation et de l'obligation de prendre en compte toute une série de contraintes afin d'obtenir une animation crédible. Un exemple typique en est l'animation par points-clefs.

A l'autre extrême, l'animateur ne dispose que d'un petit nombre de paramètres riches sémantiquement (pouvant, on l'a vu, aller jusqu'à des humeurs par exemple) ; le modèle est simple à manipuler, prend en compte un grand nombre de contraintes, mais ne permet pas un contrôle total des résultats (cas des moteurs de locomotion).

Il est ainsi probablement plus facile, à niveau d'apprentissage égal, d'obtenir plus rapidement des résultats de qualité via des mécanismes d'animation de haut niveau que de bas niveau ; par contre, obtenir exactement le résultat désiré impose l'utilisation, au moins à titre de retouches, de techniques de bas niveau.

Même si la bibliographie précédente reflète une tendance vers le développement d'outils de toujours plus haut niveau, l'utilisation toujours aussi importante par les animateurs de techniques traditionnelles telles que les points clefs montre que le besoin de contrôle précis est toujours d'actualité. Les approches de haut et bas niveau ne sauraient donc être opposées, ou ces dernières considérées comme étant en cours d'obsolescence.

1.7.2 Besoins Génériques d'un Animateur

De la constatation précédente se dégage un certain nombre de besoins des animateurs en informatique graphique. Le plus important est la possibilité de concevoir rapidement (en terme de temps utilisateur) et confortablement une animation. En particulier, les manipulations répétitives (modification "image par image" de paramètres par exemple) doivent être les plus limitées possible. Ce critère est par définition rempli par les outils de haut niveau, mais des techniques plus traditionnelles permettant d'abstraire et d'automatiser le travail sont disponibles dans tous les logiciels d'animation : utilisation de déformateurs géométriques de la famille des FFD, scripts, outils spécifiques à certains types d'objets (simulation de textiles).

De manière complémentaire au point précédent, il est important de remarquer que le contrôle sur l'animation finale prime (dans le sens où l'animateur a un objectif précis), ou plus exactement la possibilité de contrôle. Il ne s'agit pas pour l'animateur de systématiquement manipuler tous les paramètres de son animation manuellement, ce qui irait à l'encontre du point précédent ; cependant, cette option doit toujours être disponible. Pour prendre l'exemple d'une simulation complexe telle que l'animation d'une foule, le fait de disposer d'outils permettant l'obtention aisée d'un résultat en ne spécifiant que quelques critères sera certainement d'un grand secours ; mais si les mouvements de l'un des personnages posent problème (irréalisme du comportement ou inadéquation avec une contrainte scénaristique par exemple), ils doivent pouvoir être édités jusqu'au niveau de détail requis.

Un outil qui abstrairait le modèle d'animation sous-jacent au point de ne plus permettre cette possibilité serait voué à l'abandon, car inexploitable dans un cadre professionnel ; d'où l'intérêt d'une construction d'outils d'animations en tant que surcouches d'outils plus classiques, ce qui permet de revenir à ces derniers si nécessaire. Cette approche permet éventuellement de réutiliser tout ou partie d'une animation déjà conçue (via la modification de paramètres de haut niveau ou l'importation des données de bas niveau générées), ce qui peut être un gain de temps appréciable.

Un autre point essentiel est le réalisme visuel de l'animation. Ceci est particulièrement crucial dans le cas des êtres vivants, que nous sommes habitués à voir bouger et chez lesquels nous percevont facilement toute démarche insolite. Le fait que ce réalisme soit indexé à nos perceptions permet de ne pas requérir d'éventuels outils de simulation une grande exactitude (telle qu'elle serait attendue d'un outil d'ingéniérie par exemple) : l'essentiel est de suffisamment approximer les mécanismes sous-jacents pour obtenir un résultat visuellement convaincant.

1.7.3 Simulation Physique et Informatique Graphique

Considérée comme une forme d'animation de plus haut niveau qu'une manipulation directe des modèles, l'utilisation d'une simulation physique peut répondre à des problèmes difficiles abordés dans les points précédents :

- Recherche de mouvements visuellement réalistes : nous sommes habitués à interagir avec un environnement physique, et sommes donc très sensibles aux mouvements violant les lois de la mécanique.
- Représentation de formes d'interactions du modèle avec l'environnement difficilement scriptables : locomotion sur sol accidenté, réaction aux contacts, et plus globalement

généricité de l'animation (une petite modification de trajectoire introduisant une erreur allant s'amplifiant, toute altération d'une animation soigneusement construite via des outils de bas niveau est délicate).

Cependant, deux points importants doivent être abordés pour une bonne intégration d'une simulation physique dans un outil d'animation. D'une part, il est difficile de construire des modèles physiques d'êtres vivants : les paramètres qui y sont associés ne sont pas tous intuitifs, et nécessitent des connaissances bien spécifiques. A contrario, une large variété de modèles géométriques existent, et les outils pour les concevoir sont bien maîtrisés par les animateurs. Il peut donc être intéressant de disposer d'une méthode permettant la conversion d'un modèle géométrique classique en modèle physique ; nous détaillerons dans le chapitre 3 un outil que nous avons conçu à cet effet. Notons que l'un des à-côtés de cette construction d'un modèle physique à partir du modèle géométrique qu'il servira à animer est que les mouvements obtenus dépendront donc de la géométrie du modèle : l'animateur peut donc dans une certaine mesure manipuler la simulation en jouant par exemple sur la corpulence ou la stature d'un personnage.

Un autre point essentiel est la difficulté de contrôle d'un modèle physique : autant l'animation de collisions et trajectoires simples (la classique simulation d'une boule de bowling percutant des quilles) est aisément accessible, autant la génération d'une locomotion dans une simulation physique par exemple est un challenge. La différence entre ces deux types de simulation est que la seconde nécessite une réinjection constante de forces de faible amplitude dans le modèle afin d'ajuster sa trajectoire, alors que dans le premier cas quelques forces initialement appliquées suffisent. Ce n'est ainsi pas un hasard si actuellement, l'essentiel des simulations physiques utilisées dans des films ou jeux vidéos ne servent qu'à l'animation d'objets inertes ou de corps traînés ou projetés (simulations de type "poupées de son" - *ragdoll* -). L'intérêt d'un contrôleur de haut niveau, permettant l'accomplissement de tâches telles la locomotion ou la préhension, apparaît donc comme une nécessité pour dépasser ces simulations simples, ou pour permettre la génération interactive de mouvements (au sens d'une adaptation à des utilisateurs ou évènements externes). Nous aborderons dans le chapitre 2 un certains nombre d'outils utilisés en robotique pour traiter ces questions, et discuterons dans le chapitre 4 de leur application à notre modèle physique.

Chapitre 2

Animats - Nouvelle IA

2.1 Introduction

Ainsi qu'il a été montré précédemment, l'utilisation d'une simulation physique dans le cadre de l'animation d'êtres vivants permet un réalisme des mouvements et des interactions naturelles avec l'environnement. Se pose cependant le problème du contrôle d'un modèle dans ce cadre, les mouvements n'étant plus directement contrôlables. Il est possible de rester dans un cadre classique en utilisant des contrôleurs convertissant des consignes de mouvement en forces à appliquer au modèle (section 1.3.6), mais leur conception au cas par cas est difficile. Une approche consiste en leur génération via des méthodes d'optimisation (même section), mais leur approche "globale" (maximisation d'une fonction de qualité) rend l'orientation des résultats difficile.

L'intérêt d'une approche comportementale de l'animation a été montré par ailleurs (section 1.5.4) : interactions dépassant les simples collisions, modélisation de comportements de groupes moins fastidieuse, "identification" plus facile du spectateur (les buts apparents du modèle donnant un sens à ses gestes)...

L'exploitation de l'idée d'animation physique d'êtres vivants nécessite donc l'utilisation d'une méthodologie intégrant la possibilité d'apprentissage de tâches (donc de mouvements orientés vers un but apparent) et de leur coordination, et fournissant des critères de modularisation de cet apprentissage afin de le simplifier. Il est bien sûr, en principe, possible d'utiliser les outils d'optimisation déjà décrits sur une fonction de qualité intégrant l'ensemble des comportements désirés, mais sa complexité rendrait une recherche de solutions viables très longue.

Cette notion de complexité d'apprentissage dans le cadre de la génération de contrôleurs orientés tâche, relativement récente en informatique graphique, est *a contrario* une problématique très étudiée dans les domaines de l'intelligence artificielle (IA) et de la robotique ; cette dernière doit de plus, par essence, tenir compte de l'aspect physique des environnements étudiés. Nous nous proposons donc d'aborder une discipline particulièrement en adéquation avec ce que nous avons décrit : l'étude des *animats*.

Pour reprendre la définition donnée dans le descriptif de la sixième conférence internationale sur la simulation des comportements adaptatifs (SAB2000), les animats sont "des systèmes ar-

tificiels - animaux simulés ou robots réels - qui sont largement inspirés des animaux et qui s'avèrent capables de s'adapter, de survivre et de poursuivre leur mission dans des environnements plus ou moins imprévisibles et menaçants".

L'idée de cette approche, qualifiée de *nouvelle IA* ou *IA comportementale*, est de s'inspirer de données issues de la biologie (éthologie en particulier) afin de concevoir des robots et logiciels capables d'adaptabilité et de robustesse dans un environnement variable. L'un des points forts de cette approche est justement, en rupture avec l'IA traditionnelle, de ne pas tenter de résoudre des problèmes relativement abstraits dans un cadre précis. Elle met plutôt l'accent sur le fait que les comportements adaptés ont toujours lieu dans un environnement non statique avec lequel l'interaction s'effectue via des capteurs et des effecteurs. Ceux-ci imposent par ailleurs la nécessité d'un traitement des questions de perception, un environnement réel n'étant pas significatif en soi.

Après une présentation de l'approche "traditionnelle" de la robotique par l'IA, nous détaillerons les points clefs de la notion d'animat. Ceci sera ensuite détaillé à travers l'examen des travaux de Brooks, un chercheur influent dans ce domaine. Nous discuterons ensuite des rapports entre IA et animats, et des apports à attendre de cette approche dans le cadre de l'animation d'êtres vivants.

2.2 IA Symbolique

La nouvelle IA étant en opposition avec les choix faits usuellement lors de l'application de l'intelligence artificielle "classique" aux problèmes rencontrés en robotique, nous exposerons ici les points essentiels à une comparaison. De par cet objectif, le traitement en est très succinct ; on pourra se référer à l'ouvrage introductif de Farreny et Ghallab [37] pour une description plus approfondie des techniques employées, et à la synthèse de Ganascia [41] pour une analyse des différents courants de l'IA, de ses présupposés et de ses acteurs marquants. Plus spécifiquement, une comparaison dans le cadre des réalités virtuelles des approches traditionnelles et comportementales de l'IA a été faite par Mateas dans un article de 1999 [64].

2.2.1 Objectifs

Le but de l'IA est dans l'absolu l'élaboration de machines mimant les capacités cognitives de l'être humain, et plus particulièrement celles considérées comme les plus évoluées et les plus représentatives de la cognition (compréhension du langage et raisonnement). L'ordinateur, de par la souplesse d'expérimentation qu'il permet, s'est imposé comme le support privilégié de cette implémentation.

Malgré cet *a priori* généraliste, l'IA s'est rapidement scindée en une série de sous-domaines de plus en plus spécialisés et pas nécessairement compatibles dans leurs approches. Quelques-unes des branches importantes sont :

– Les systèmes experts, visant à mimer l'expertise d'un spécialiste dans un champ de compétences précis ; leurs importantes (ou supposées telles) retombées économiques ont certainement contribué à attirer l'attention et les efforts importants investis dans cette direction.

– Plus sémantique, le traitement du langage naturel vise à la manipulation de phrases en langage naturel. Des applications en sont la traduction et le dialogue avec un être humain, qui tous deux nécessitent une certaine compréhension du langage (capacité à exploiter les informations présentes dans le texte en fonction d'un corpus de connaissances permettant son interprétation).
– Un autre domaine important est la vision artificielle, et plus généralement la reconnaissance, qui vise à extraire d'un signal ses caractéristiques significatives afin de reconstituer la scène qui en est à l'origine ou de classifier un éventuel objet perçu.

2.2.2 Problèmes et Représentations

L'hypothèse fondamentale de l'IA est la possibilité de coder l'information manipulée dans le système nerveux sous une forme discrète. Deux types de représentations sont principalement utilisés : des graphes ou des symboles, généralement organisés via une logique de prédicats. Notons que tout graphe représentant des relations entre éléments, il est de toute façon possible de le représenter sous forme symbolique.

Une autre approche, mettant l'accent sur l'aspect sémantique et linguistique des traitements humains, utilise des notions de structuration de l'information telles les *frames* de Minsky [70], comparables à des objets, ou les réseaux sémantiques représentant les rapports entre des concepts.

Une part importante des recherches en IA, du fait de l'engouement déjà décrit pour les systèmes experts et plus généralement pour les techniques de démonstration automatique, s'est cependant focalisée sur la manipulation de symboles via des outils logiciels nommés moteurs d'inférence.

2.2.3 Moteurs d'Inférence

Les moteurs d'inférence sont des logiciels mimant la démarche des mathématiciens : leur objectif est de démontrer la validité d'une assertion en se basant sur une connaissance du domaine exprimée sous forme de théorèmes et d'axiomes.

Ce problème se traduit théoriquement en un parcours de l'arbre (ou du graphe) exprimant l'ensemble des réécritures des données du problème via les théorèmes jusqu'à atteindre un ensemble d'axiomes. Cependant, la taille considérable (voir infinie) de cet arbre décourage en pratique toute exploration naïve.

Afin de gérer cette complexité, une procédure de décision de la prochaine règle à appliquer est utilisée. On se base pour cela sur une série d'*heuristiques*, critères censés augmenter les chances de se rapprocher du résultat par rapport à un parcours aléatoire. Elles contiennent en fait une expertise considérée comme caractéristique du travail d'un logicien ou d'un mathématicien, et constituent donc un élément fondamental du système.

Lorsque la base de théorèmes est très riche (des milliers de règles) et est obtenue par *interview* d'un expert dans un domaine, l'ensemble base - moteur est qualifié de *système expert*. L'objectif est alors de stocker le savoir-faire d'un spécialiste (en médecine ou géologie par exemple) et de pouvoir fournir sur demande une analyse compréhensible et justifiable (par examen des règles employées) d'une situation donnée.

2.2.4 IA et Robotique

Approche SMPA

Les techniques de résolution de problèmes fournies par l'IA ont été largement utilisées en robotique. Les données fournies par des capteurs sont alors interprétées par un algorithme de reconnaissance, qui les convertit en un ensemble de symboles nommé *représentation du monde*. Cette représentation construite, un algorithme de résolution de problèmes l'utilise pour planifier la série d'actions à effectuer en fonction d'un but à atteindre. Cette séquence une fois trouvée, le plan est exécuté via l'envoi de commandes, toujours sous forme symbolique, à une série de modules les convertissant en des commandes motrices adaptées. Cette approche est ainsi qualifiée de SMPA, *Sense / Model / Plan / Act*.

Modules Fonctionnels

L'approche SMPA est structurée d'une manière caractéristique des applications de l'IA : la décomposition en modules fonctionnels (mémoire, raisonnement, perception) communicant via un ensemble uniforme de symboles et partageant un modèle du monde centralisé et pertinent (i.e. considéré comme reflétant fidèlement l'environnement compte-tenu des tâches à y effectuer). Cette approche a orienté la recherche dans le sens d'une spécialisation des études, l'unité en étant la conception de modules fonctionnels plutôt que de comportements effectifs.

Les modules étant reliés par le système de symboles, il n'y a le plus souvent pas de recherche d'une connexion à un système de perception ou à de réels actionneurs.

Micro-Mondes, Réalité et Replanification

La constatation précédente a le plus souvent mené à la validation des modules de raisonnement dans des *micro-mondes*, environnements simulés extrêmement simplifiés dans lequel les modulent peuvent directement travailler via des flux d'information purement symboliques. Une variation consiste en une expérimentation de robots dans des conditions suffisamment contrôlées pour que l'imprécision des capteurs et des actionneurs soit négligeable.

Il est cependant accepté dans les rangs mêmes des chercheurs en IA que cette approche est en l'état inadaptée à la réalité : il n'est pas possible de construire une représentation exacte du monde, et cela prend du temps ; il n'est par ailleurs pas possible de prévoir toutes les modifications d'un environnement réaliste. Une technique permettant de conserver une certaine efficacité consiste dans le fait de ne pas se contenter d'appliquer le plan, mais dans le contrôle de son exécution et sa *replanification* éventuelle en cas de fluctuation trop importante de l'environnement, via éventuellement un *feedback* vers le module de raisonnement.

2.3 L'Approche "Animats"

2.3.1 Une Critique de l'IA

Brooks [20] critique les présupposés de l'IA tels qu'exprimés en 2.2, et montre en quoi ils interdisent l'accès à des capacités cognitives réelles, au sens d'effectivement exploitables dans

des simulations ou des environnements réalistes.

Problème de l'Isolation de la Perception

Il note tout d'abord que l'hypothèse essentielle d'une intelligence manipulant des symboles implique la présence d'un système perceptif convertissant les données perçues dans ce formalisme, chaque symbole représentant une entité ou une caractéristique de l'environnement perçu. Cette approche a orienté les chercheurs en reconnaissance vers la construction d'un système de représentations généraliste des informations reconnues, tel que décrit section 2.2.4.

Cela va cependant à l'encontre des observations psychophysiologiques, qui indiquent que la perception n'est pas un processus à sens unique mais est au contraire active, et dépendante de la tâche en cours. Plus généralement, Brooks insiste sur le fait que le problème n'est pas de construire une représentation unifiée et cohérente de l'environnement pour ensuite raisonner sur celle-ci. Il semblerait plutôt que les êtres vivants n'extraient de leurs perceptions que les données qui sont significatives en fonction de leurs objectifs et actions actuels : il n'y a pas de distinction claire entre la perception et le raisonnement, au sens où des modules autonomes se chargeraient pour certains de représenter le monde, et pour d'autres d'utiliser cette représentation. Par ailleurs, les abstractions utilisées sont certainement dépendantes de l'espèce considérée, et celles que l'étude de notre comportement nous fournit ne sont pas nécessairement des indications fiables de la manière dont nous traitons effectivement les signaux sensoriels.

Brooks met ainsi l'accent sur la perception comme problème fondamental [23], alors que l'IA classique s'intéresse plutôt au raisonnement. Il fait par exemple mention du fait qu'aucun programme de reconnaissance ne peut identifier une chaise autrement que sur des prises de vues bien précises ; encore doit-elle être d'un modèle spécifique.

Par ailleurs, les systèmes de raisonnement peuvent résoudre tout un ensemble de problèmes basés sur des situations réelles. Mais ces problèmes sont toujours posés via une entrée symbolique, et jamais par la présentation d'une photographie décrivant la scène. Or, l'abstraction permettant de passer de cette image à la description des concepts pertinents de la scène est l'étape la plus délicate, et celle qui mobilise le plus nos ressources cognitives. Le fait de se baser sur des données symboliques alors même qu'aucun mécanisme robuste de perception n'est disponible est donc un raccourci discutable.

Wilson [113] va dans le même sens en faisant remarquer que dans le cadre des systèmes étudiés par l'IA, le fait de s'intéresser à des compétences isolées dans une approche analytique amène à se concentrer sur des capacités perçues comme fondamentales (algorithmes de recherche et d'extraction) plutôt qu'à des compétences "basiques" telles que la perception, la classification ou l'adaptation. Ce recul par rapport à l'environnement mène à des problèmes largement décrits tels l'ancrage des symboles (*symbol grounding* : problème de la correspondance entre les stimuli physiques et les symboles arbitraires utilisés lors des raisonnements) ou la fragilité (*brittleness*) de ces systèmes (performances satisfaisantes pour une tâche donnée mais difficulté à généraliser ces capacités dans un environnement bruité ou légèrement différent) [5].

Arkin [6] en arrive à condamner ce programme de recherche en lui-même sur la constatation que :

> *The traditional approach to computer vision and the processing of other related sensor modalities has more often than not ignored the fact that perceptual needs are predicated upon the motivational and behavioral requirements of the consuming agent. This has lead to a wealth of litterature on perceptual processing which serves as little more than academic curiosities : i.e., solutions looking for problems. Although much of this work can serve as a useful pastime for researchers and provide for glitzy demonstrations to impress the uninitiated, the resulting contributions over the past 30 or so years have not yet produced robust intelligent robotic perception. [...]*
>
> *The fundamental problem of treating perception as an isolated phenomena is a very serious one and brings into question whether the "general vision" problem is really a problem at all, but rather an artifact of misguided research.*

Utilité d'une Représentation du Monde

En imaginant même qu'une représentation unifiée de l'environnement soit possible, cette activité est si coûteuse en ressources et en temps que ce modèle s'avère obsolète aussitôt créé. Aucune action précise n'est possible dans ce cas pour peu que le milieu soit un tant soit peu fluctuant (problème de la "qualification" des actions, dans lequel le processus de planification est mis en échec par la complexité de l'environnement [5]).

Ce problème a été le plus souvent éludé via l'expérimentation dans des environnements simplifiés à l'extrême (section 2.2.4), mais il est peu probable que ces travaux soient transposables dans le monde réel. La replanification, quant à elle, n'est pas applicable si des comportements réactifs à courte latence (nécessaires par exemple à la locomotion ou à l'évitement de collision) sont étudiés.

Le problème vient de ce qu'en dehors de ces cas simplifiés, une représentation symbolique ne peut décrire avec précision l'environnement : les capteurs et effecteurs sont nécessairement bruités, voire trompeurs, l'environnement évolue trop rapidement pour qu'une éventuelle modélisation en soit utilisable. Par ailleurs, l'idée d'une représentation unique élude le fait que la perception soit dirigée par l'activité en cours, comme le précise la section précédente.

Brooks en arrive à considérer que la notion même de représentation n'est pas une bonne abstraction pour la conception de la plus grande partie d'un système intelligent, et qu'au contraire elle complique les choses [23] ; Arkin [6] ne dit d'ailleurs pas autre chose quand il précise que :

> *In general, there is not point to constructing full scale scene interpretation and three dimensional world reconstruction. Why bother ? All that needs to be done is to identify the necessary perceptual cue to support motor action and the job is done. Much of the inherent difficulty in the general vision problem vanishes, as there is no need to perform such a complex and arduous task.*

Problème de la Décomposition Fonctionnelle

Le problème de la décomposition d'un système cognitif en une série de modules fonctionnels est qu'en pratique, toute recherche dans ce cadre se spécialise en un type de fonctionna-

lité précis (apprentissage, planification, détection de contours...) en effectuant au passage une série d'hypothèses sur ses interfaces rendues nécessaires par le besoin de cerner le domaine concerné. Mais ces hypothèses se font rarement en concertation, et permettent à chacun de restreindre son domaine jusqu'à un niveau où il sera gérable, sans forcément être assuré alors de sa portée. Concrètement, une longue chaîne de modules qui doivent tous être rendus compatibles (ce qui est peut-être un problème aussi difficile que leur conception elle-même) doit être assemblée afin de tester l'efficacité de leur collaboration dans une situation réelle : ceci n'est que rarement effectué en pratique.

Au-delà de cette constatation, Brooks estime que le réductionnisme fonctionnel en IA crée de nouveaux problèmes qui n'ont pas lieu d'être, et dont la résolution n'est donc pas nécessaire à la construction d'une entité artificielle.

2.3.2 Approche de la Nouvelle IA

Fonctionnalités Recherchées

Dans un article de 1991 [23], Brooks cite un certain nombre de caractéristiques que devrait posséder un agent autonome incarné, qu'il appelle une *créature*.

- Elle doit faire face efficacement et rapidement aux modifications d'un environnement dynamique.
- Elle doit être robuste, i.e. des variations mesurées de l'environnement ne doivent pas dramatiquement affecter ses capacités.
- Elle doit être capable de gérer plusieurs buts simultanément et sélectionner celui poursuivi en fonction de l'environnement, d'où un opportunisme et une capacité d'adaptation.
- Elle doit avoir quelque chose à faire dans son environnement, un objectif à poursuivre.

Importance du *Physical Grounding*

Une observation de l'évolution des êtres vivants montre que les premières capacités observées et celles qui ont mis le plus de temps à se développer sont celles nécessaires aux comportements appropriés dans un environnement complexe et seulement partiellement prévisible. Les capacités considérées comme essentielles par l'IA, telles la résolution de problèmes ou le langage, ne sont apparues que beaucoup plus récemment. Cela incite Brooks à penser qu'une réactivité efficace dans un environnement fluctuant est la partie la plus ardue à mettre en place, des capacités de plus haut niveau ne pouvant se construire qu'au-dessus de ces compétences.

Il est tentant d'en déduire que les symboles sont la découverte importante de la vie, puisqu'à la suite de leur apparition les capacités que nous considérons comme de haut niveau ont pu se mettre en place. Cependant, comme le montre la maladresse des robots directement construits sur la base de symboles, ceux-ci n'ont pas de sens s'ils sont utilisés en-dehors d'une référence à des mécanismes robustes, *physically grounded* : ceux-ci sont une base indispensable à la mise en place d'une intelligence.

Contrairement à l'IA classique, où la généricité (capacité à résoudre efficacement toutes les instances d'un problème) est un critère de qualité, il suffit ici de se concentrer sur les situations les plus probables et pas sur une solution parfaite ne faisant que compliquer inutilement le système, donc diminuer sa robustesse globale. De manière générale, ces systèmes ne peuvent être

critiqués sur la base de leur incapacité à traiter des problèmes arbitraires, comme le font par exemple les moteurs d'inférences : cette spécialisation fait leur robustesse, et rien ne permet de restreindre *a priori* la complexité des comportements pouvant être obtenus par le développement incrémental de cette approche.

Paradoxalement, ce refus d'un traitement générique des problèmes mène souvent à une meilleure résistance au bruit et à une adaptabilité aux changements modérés de l'environnement. Par exemple, le fait de n'exploiter que le minimum de signaux requis, en ne cherchant pas à catégoriser ou exploiter les autres [6], permet de le faire aussi souvent que possible[1]. Dans l'approche IA, au contraire, la résolution exacte par le système du problème implique de faire correspondre toute perception avec une instance précise de ce problème. Or, comme nous l'avons vu, cette tâche de catégorisation est difficile à effectuer de manière robuste et rapide en dehors d'un cadre contrôlé. Encore une fois, l'approche classique ne pêche donc pas tant sur l'aspect décisionnel que sur le plan perceptif : la décomposition fonctionnelle isolant cette capacité des problématiques de motivation ou de contrôle dans le cadre d'objectifs précis, elle la rend plus difficile à mettre en place.

Modules Comportementaux

Plutôt que d'utiliser des modules fonctionnels, c'est à dire permettant la perception, la planification, le contrôle, etc., il s'agit ici de définir des modules implémentant un comportement complet, directement reliés aux capteurs et aux actionneurs. Des mécanismes permettent l'arbitrage de l'accès aux actionneurs, telle la *subsumption* (section 2.4.1), mais aucune information ne circule par ailleurs entre les modules. Le plus souvent, l'ajout de comportements ne se fait donc pas par modification ou extension des modules existants, mais par ajout d'un nouveau module.

L'un des avantages de cette approche est de permettre la validation à tout moment des modules implémentés, puisqu'ils n'ont pas à être placés dans une architecture pour être utilisables. Elle est complètement orthogonale à la décomposition fonctionnelle décrite en section 2.2.4.

Absence de Représentation Centralisée

A l'inverse de l'IA, l'un des crédos de la nouvelle IA est que "le monde est son meilleur modèle". Plutôt que de chercher à construire des représentations les plus précises possibles de l'environnement, l'objectif est d'exploiter rapidement ses caractéristiques saillantes, via une connexion entre la perception et l'action ayant le moins de latence possible.

Les données manipulées dans un module donné (cf. section 2.3.2) n'ont de sens que pour celui-ci, et ne sont pas exprimées dans un formalisme global. Chacun peut éventuellement contenir une représentation du monde, mais uniquement sous la forme et avec la précision nécessaire à l'accomplissement de la tâche.

Une représentation de l'environnement existe donc dans le système, mais elle est distribuée et multimodale, sous la forme d'une multitude de traitements directement utilisés pour piloter des actions et jamais fusionnés (s'il doit y avoir fusion, ce sera le plus souvent au moment de la

[1] On retrouve dans cette démarche des caractéristiques de la perception animale telles que décrites par l'éthologie (voir la section E.6.2).

résolution de conflits au niveau des actionneurs [21, 6]). Cette absence de représentation complexe à maintenir simplifie le travail à effectuer et permet des boucles de réaction plus rapides, donc une perception plus fréquente de l'environnement et un comportement plus précis et adéquat. Cela augmente par ailleurs la robustesse du système face à un environnement inconnu, puisqu'il ne dépend pas d'un modèle complexe et donc éventuellement adapté à un environnement trop spécifique mais plutôt de certains de ses *aspects*, significatifs pour les modules les percevant.

Cette approche est complémentaire au *physical grounding* ; elle encourage une construction effective, du bas vers le haut, des modèles développés, ceux-ci ne pouvant facilement être abstraits du système de capteurs et d'effecteurs associés (cette notion est illustrée par les travaux de Nolfi *et al*, section 2.5.2).

Absence de Planification

Tout comme il n'existe pas de représentation centralisée de l'environnement, aucun module n'est non plus en charge de la gestion des buts de la créature. Encore une fois, cette gestion peut être considérée comme distribuée, au sens où chaque module comportemental a ses buts propres : c'est le jeu des inhibitions entre eux et de leur déclenchement lorsque les conditions environnementales nécessaires sont présentes qui mène à un comportement cohérent. Ainsi, la créature n'a aucun contrôle central, et n'est en fait qu'un ensemble de comportements en compétition. Ceci supprime le paradoxe classique d'un centre pilotant l'ensemble du comportement, et est proche des théories de Minsky sur la génération du comportement [71]. La rationalité du comportement global est attribuée par l'observateur, mais n'est pas nécessairement une composante du système.

Une différence avec les robots classiques est que les buts ne sont pas fournis à la créature dynamiquement. Ils sont présents dès sa mise en activité, de par son architecture même : chaque module étant comportemental, le jeu de leurs interactions rend la présence d'un module "motivationnel" inutile. Le monde n'est plus un obstacle à la résolution des buts, interrompant sans cesse l'exécution du plan, mais devient au contraire le support de cette résolution à travers les stimuli qu'il fournit.

Capacités d'Apprentissage

De par l'absence d'une structure de données partagée, on peut s'interroger sur la possibilité pour un système construit suivant ces principes d'apprendre quoi que se soit de son environnement, d'autant que l'absence de fusion des perceptions ne permet pas une réduction de leur complexité.

Brooks remarque à ce propos [24] que la somme d'apprentissage nécessaire à l'adaptabilité du robot n'est pas nécessairement très importante. Par exemple, une fois le robot construit toute une série d'informations est connue concernant ses modalités d'interaction avec le monde (types de capteurs, nombre d'actionneurs...). Ce type de connaissances peut être intégré dans le programme de contrôle, plutôt que de forcer le robot à les découvrir par lui-même.

De même, plutôt qu'un mécanisme global et externe de récompenses, il propose l'utilisation d'une série de fonctions de récompense directement intégrées dans le système. Ainsi, les

animaux utilisent des mécanismes d'évaluation telle la faim pour apprendre à se nourrir à bon escient : les comportements alimentaires ne sont donc pas uniquement optimisés génétiquement en fonction d'une évaluation basée sur la mort de ceux qui ne se nourrissent pas, mais aussi à un niveau individuel. Dans cet esprit, il peut exister des systèmes de récompense basés sur l'état de charge des batteries par exemple, en plus de ceux pour la surface explorée ou tout autre paramètre caractéristique de la tâche pour laquelle le robot a été conçu (les systèmes de classifieurs sont une technique bien adaptée à l'implémentation de ce type de mécanismes).

Il propose finalement une série d'éléments dont l'apprentissage peut être utile à un robot autonome :

- les représentations du monde nécessaires à l'exécution de certaines tâches,
- des caractéristiques des capteurs et effecteurs (calibration),
- les interactions efficaces entre comportements [61],
- de nouveaux modules comportementaux.

Des exemples de traitement de ces problèmes seront détaillés dans la section 2.5.

Wilson [113] insiste plus sur la nécessité de capacités d'apprentissage, qu'il voit comme un outil permettant d'exploiter au mieux les capacités de l'animat et de s'adapter plus facilement aux variations de l'environnement, donc permettant une amélioration de son autonomie. Ceci lui paraît rendu nécessaire par l'impossibilité fréquente d'extraire directement de l'environnement les informations nécessaires à une prise de décision immédiate. Dans un environnement simplifié, dans lequel par exemple un gradient d'odeur permet de retrouver une source de nourriture ou un prédateur, une architecture réactive simple permet de décider de la direction à suivre à tout moment. Mais dans un cadre plus réaliste, les informations peuvent être indirectes (caractéristique de l'environnement associée à un certain type de proie...) et les récompenses retardées (l'animat ne sait qu'*a posteriori* que ses choix ont été pertinents).

Dans ce type de cadre, des outils d'apprentissage robustes tels les systèmes de classifieurs permettent de tirer au mieux parti des informations disponibles. Cette approche *on line* de l'apprentissage est complémentaire à l'utilisation d'outils basés sur la compétition et l'évolution des populations, évitant de "redécouvrir" régulièrement des solutions efficaces.

Il ne s'agit donc pas d'opposer une approche "réactive" qui se concentrerait sur les perceptions et actions au détriment des processus décisionnels (limitant ainsi les comportements à ceux de formes de vie simples) à une approche "planificatrice" cherchant à abstraire (avec les difficultés que nous avons citées) les informations perçues pour les traiter ensuite symboliquement. Au contraire, l'union de ces approches telle que proposée par Arkin [5] permettrait la fusion de données immédiatement perçues et de "perceptions virtuelles" [113] issues des outils d'apprentissage précédents afin d'aller au-delà d'une approche purement réactive de la prise de décisions.

Un exemple de système de ce genre est fourni par les travaux de Ingrand *et al* [49] qui exploitent dans le cadre d'un ensemble de robots coopératifs une architecture permettant l'intégration d'un système réactif et d'un planificateur. Ce système tente de produire des plans en fonction des objectifs qui lui sont fournis. Le superviseur se charge ensuite d'orienter les comportements de la couche réactive et de réévaluer si nécessaire les plans qui lui sont proposés en fonction de leur exécution. La couche réactive est la seule à être reliée aux capteurs

et effecteurs, et n'a donc pas à "attendre" les couches planificatrices comme dans un système SMPA classique pour réagir à l'environnement ; un "micro opportunisme" est donc possible, hors planification. Cependant, la présence des outils de supervision permet une coordination plus aisée des robots dans le cadre de la résolution de tâches complexes.

2.4 Un Exemple : les Travaux de Brooks

L'approche "animats", nous l'avons vu, ne se caractérise pas tant par les problèmes qu'elle se propose de résoudre ou les technologies qu'elle emploie pour y parvenir que par des choix de conception spécifiques. Elle ne saurait donc être réduite au travail d'un groupe de recherche spécifique tant les choix possibles sont vastes.

Cependant, afin d'illustrer plus précisément ce domaine de recherche et quelques résultats qui en sont caractéristiques, nous allons nous baser sur une série d'articles largement cités de Rodney A. Brooks. Certaines des positions qu'il prône sont radicales, et ses choix technologiques et de conception peuvent être discutés ; ils sont cependant révélateurs de la rupture entre les approches comportementalistes et plus traditionnelles du contrôle.

2.4.1 Architectures Proposées

Fidèle à l'idée qu'un modèle ne peut être validé sans référence à son incarnation, Brooks propose une série d'architectures permettant une implémentation effective. Elles sont basées sur des réseaux à topologie fixée d'automates à états finis (il n'existe ni notion d'accès à des données globales, ni réseau de communication établi dynamiquement).

La *subsumption*

L'idée de l'architecture par *subsumption* est de mettre en relation des modules comportementaux via des connexions permettant leur communication ou inhibition. Une présentation plus détaillée de cette architecture, ainsi que quelques exemples de modules utilisés pour les robots décrits section 2.4.2 se trouvent dans [21].

▷ **Structuration** De manière générale, le système est constitué de couches, chacune reliant les perceptions aux actions. Il ne s'agit donc pas d'un modèle à couches au sens classique du terme, dont la complexité et l'abstraction augmenteraient au fur et à mesure qu'elles s'éloigneraient des capteurs et actionneurs. Au contraire, comme décrit section 2.3.2, chaque module implémente un comportement complet. Les relations entre couches se font via des connexions à sens unique permettant l'envoi de message, l'inhibition d'un comportement (le blocage de sa sortie) ou sa suppression (le remplacement de sa sortie par celle du module suppresseur). Ces mécanismes permettent la résolution des conflits entre les commandes destinées aux actionneurs.

Notons qu'aucune couche de bas niveau ne dépend d'une couche de niveau supérieur pour générer un comportement (ce n'est donc pas un modèle hiérarchique) ; l'inverse n'est cependant pas vrai, et une couche supérieure peut utiliser en plus de signaux issus des capteurs ceux transmis entre deux automates éléments d'un niveau inférieur.

L'absence de module de planification à proprement parler (section 2.3.2) est un principe de conception poussé aussi loin que possible : par exemple, si un module doit être déclenché par un autre, plutôt que de les relier directement, ce qui sous-entendrait que le premier considère qu'il a atteint son but (d'où une hypothèse sur l'état du monde), il est préféré un déclenchement indirect du second module via une perception de l'environnement, afin d'être certain que cette hypothèse est vérifiée [21]. Le monde est ainsi utilisé comme un média entre les parties du programme, ce qui augmente sa robustesse.

Cette absence de contrôle ne signifie pas qu'un comportement cohérent émerge forcément du chaos. Brooks précise que contrairement aux processus naturels, il n'est évidemment pas question de se livrer à une multitude d'essais : une conception et un *debugging* minutieux sont nécessaires (ils s'avèrent par ailleurs plutôt aisés de par la démarche incrémentale retenue).

▷ **Constituants** Chaque couche est constituée d'un réseau asynchrone d'automates à états finis éventuellement associés à des éléments de comptage (*augmented finite state machines*, AFSM). Chaque automate dispose d'une série de registres contenant le dernier message reçu sur chaque entrée : la communication se fait par connexion des sorties des automates vers ces tampons. Les AFSM peuvent y exporter le contenu d'un registre, d'un compteur ou le résultat d'un calcul combinatoire basé sur ces valeurs. Ceci excepté, ils ne partagent aucun état ou registre de compteur, dans l'esprit d'une absence de représentation centralisée (section 2.3.2). La communication entre constituants d'un module ne se fait nulle part via des *tokens* auxquels une sémantique valable dans l'ensemble du système serait attribuée, mais uniquement par des flux de nombres, et ceux-ci n'ont de sens que si leur émetteur et leur récepteur sont considérés.

▷ **Implémentation** Les AFSM sont implémentés sous forme de microcontrôleurs ou de simulation sur des processeurs plus complexes ; par ailleurs, on peut obtenir leur spécification par compilation d'un langage de plus haut niveau, le *behavior langage*. Pour de petits réseaux, une compilation sous forme de logique combinatoire ou de consultation de tables (*tables lookup*) est possible ; ce formalisme présente l'avantage de ne pas être Turing-puissant, ce qui permet de donner une limite supérieure à la puissance de calcul nécessaire pour exprimer des comportements simples.

Système d'Hormones

En parallèle à une architecture de *subsumption*, Brooks utilise un système de communication hormonal proche de celui utilisé par le homard [22] afin d'assurer la coordination d'un robot capable de locomotion via des pattes articulées.

Des conditions environnementales ou internes peuvent libérer un certain taux d'hormone simulée, qui décroîtra progressivement par la suite. Il est ainsi possible de modéliser la persistance des comportements.

2.4.2 Résultats

Compte-tenu des présupposés de la nouvelle IA, il est nécessaire de la tester sous une forme incarnée, c'est-à-dire via la construction effective de robots. Ceci a été effectué par l'équipe de

Brooks, qui en a développé un certain nombre. Une caractéristique marquante est l'émergence de comportements, donc d'actions semblant orientées vers des buts, à partir de l'interaction de modules qui ne l'étaient pas.

Systèmes à Répulsion - attraction

Les robots les plus simples sont capables de naviguer dans un environnement réaliste, en évitant les collisions et en se dirigeant vers des endroits intéressants (des lieux distants, ou des objets mobiles). Cette navigation est effectuée via un système essentiellement réactif basé sur l'intégration d'attractions et de répulsions. Un autre modèle est capable de fuir la lumière pour se dissimuler dans des zones sombres via un déplacement en spirale en présence de lumière.

Intégration de Comportements

Une version plus élaborée est capable de se déplacer dans un environnement de bureau et d'y collecter des objets en forme de canette grâce à un bras articulé. De manière intéressante, ses modules comportementaux ne communiquent pas directement : chacun est connecté aux capteurs et à un réseau d'arbitration gérant l'accès aux actionneurs. Cependant, l'action paraît coordonnée comme elle le serait si un planificateur était à l'œuvre. Cette organisation est due au fait que chaque comportement traite les caractéristiques adaptées de l'environnement : lorsque le comportement d'approche du bras a amené la "main" près d'une canette, par exemple, un réflexe de préhension se déclenche sans avoir été piloté par le module précédent du fait de la perception de la présence de l'objet. Ainsi, la cohérence du comportement n'est possible que par son adaptation à l'environnement, et ne peut se comprendre sans lui, d'où l'*embodiement*.

Un avantage de cette structuration est l'opportunisme : chaque comportement se déclenchant lorsque cela est adapté, il est par exemple possible de tendre une canette au robot, qui s'en saisira. Par ailleurs, le problème de la replanification ne se pose plus : si le robot est bousculé pendant sa tentative de préhension, ce comportement cessera faute de stimulation adaptée et celui de recherche reprendra.

Locomotion

Une autre implémentation [19] est celle d'un robot à six pattes ; il est capable grâce à une série de capteurs de se déplacer en s'adaptant au type de terrain rencontré, et de suivre un être humain grâce à des capteurs de chaleur. Les principes de conception décrit précédemment sont poussés à l'extrême ici : il n'y a pas, par exemple, de module gérant la coordination entre les moteurs d'une patte, chacun ayant un comportement indépendant. Le réseau a été construit de manière incrémentale : des couches y étaient régulièrement ajoutées sans modification ou suppression de celles responsables des comportements déjà mis au point. Le système n'est absolument pas hiérarchique (au sens d'une série de systèmes de contrôle de plus en plus abstraits s'échangeant des ordres) et n'utilise aucun modèle cinématique ou système de changement de coordonnées. De manière caractéristique, il est possible de revenir à des comportements antérieurs en inhibant les automates ajoutés ultérieurement.

Le titre de cet article[2] ne se réfère pas à un mécanisme d'évolution automatique, mais à un parallèle entre la démarche utilisée et les mécanismes de l'évolution naturelle, qui agit par modifications successives d'êtres vivants à tout moment viables mais de plus en plus complexes. L'une des idées importantes abordée est qu'il n'est pas nécessaire, pour obtenir des comportements de haut niveau (suivi de personne) à partir d'une série de comportements de bas niveau (équilibre, lever de pattes), de postuler l'existence de "structures qualitativement différentes" ou de "formes uniques d'interconnexion de réseaux" pour intégrer des comportements de plus haut niveau.

Construction de Carte

L'un des robots construits est capable de mémoriser une représentation de l'environnement pour pouvoir ensuite se rendre d'un lieu à un autre. Cela paraît contradictoire avec le mode de conception présenté ci-dessus, du fait de l'absence de partage d'informations qui paraîtrait nécessaire.

Ce robot perçoit l'environnement via 12 capteurs de proximité et une boussole. Il dispose de comportements de bas niveau d'évitement de collision et de suivi de mur, implémentés suivant les mêmes techniques que les robots précédents. Par ailleurs, une couche intermédiaire de comportements organisée en nœuds tente de repérer et de mémoriser des caractéristiques marquantes de l'environnement. Chaque fois qu'une zone marquante est détectée, l'un des nœuds se l'attribue et se connecte aux nœuds adjacents. Ainsi, un graphe de l'environnement est dynamiquement construit (notons à ce propos que le câblage est conçu pour être linéaire en fonction du nombre de sites).

Ensuite, lorsque le robot est proche de ce qu'un nœud croit être son lieu associé, sont niveau d'activation augmente : le robot dispose donc non seulement d'une carte mais aussi d'une perception du lieu où il se trouve, et ceci de manière totalement distribuée (les nœuds ne partageant pas d'informations autres que les perceptions). Brooks précise que cette représentation de l'environnement est proche de ce qui a pu être observé dans l'hippocampe des rats.

Lorqu'il est demandé au robot de se rendre à l'un des sites, une activation se propage depuis le nœud associé vers les nœuds adjacents, ce qui permet au robot de savoir à tout moment s'il est proche du lieu recherché et quelle direction suivre. Aucun processus de planification n'est à l'œuvre pour cela, et aucune représentation explicite du chemin à suivre n'existe. Il est intéressant de remarquer que l'ajout de nœuds augmente la précision de repérage en permettant une représentation plus fine de l'environnement.

2.4.3 Bilan

Brooks envisage l'IA et la nouvelle IA comme complémentaires dans leurs approches [20]. En effet, l'IA traditionnelle effectue des raisonnements complexes dans des environnements appauvris, et espère les voir se généraliser de manière robuste dans des environnements complexes. *A contrario*, la nouvelle IA produit des comportements moins complexes mais robustes et s'exécutant dans un environnement réel, donc bruité et complexe, l'espoir étant de pouvoir

[2] *A Robot that Walks ; Emergent Behaviors from a carefully Evolved Network*

les généraliser à des tâches plus complexes. Il ne nie d'ailleurs pas l'intérêt de l'étude d'une intégration des deux approches.

Brooks propose par ailleurs un certain nombre de problématiques posées par la nouvelle IA [20, 23] :

- Comment plusieurs dizaines de modules comportementaux peuvent-ils être combinés d'une manière leur permettant toujours de coopérer ?
- Comment intégrer des sources de perception multiples lorsque leur fusion semble indispensable ?
- Comment automatiser la construction d'interfaces entre modules comportementaux, afin de générer des systèmes plus complexes ?
- Comment automatiser la construction de modules de génération de comportements ?
- Combien de couches peut-on construire en utilisant l'architecture de *subsumption* avant que les interactions ne deviennent trop complexes ?
- Quelle complexité de comportement peut être atteinte sans recours à une représentation centralisée ?
- Des fonctions complexes tel l'apprentissage peuvent-elles être simulées par un réseau fixe d'automates ?

Il faut noter que les thèses de Brooks sont loin de faire l'unanimité. Werner [112] critique par exemple la tendance qu'a la communauté des systèmes à agents réactifs à rejeter toute idée de planification, de structuration du système (*design*) ou d'incorporation de règles complexes, au profit de l'idée que de l'interaction d'agents simples émergera nécessairement une solution efficace au problème posé.

Il vise en particulier Brooks, en faisant remarquer que ses systèmes sont conçus et debuggés de manière très minutieuse, donc intègrent implicitement une somme de connaissances importantes, et plus techniquement en remarquant que la proclamation d'une décomposition purement comportementale dans le cadre de l'architecture de *subsumption* ne tient pas. L'ajout de certains comportements semble en effet imposer la présence de couches de médiation entre eux, qui elles-mêmes sont difficilement descriptibles en des termes autres que fonctionnels. Plus généralement, il remarque un manque de clarté dans la notion de couches et dans les rapports entre elles.

2.5 Apprentissage et Animats

Ainsi qu'il a été montré précédemment, l'approche de Brooks des animats ne laisse pas une grande place à l'apprentissage, lui préférant une conception minutieuse des modules comportementaux et la reléguant le plus souvent à des tâches d'adaptation limitées.

Cette conception des choses n'est cependant pas intrinsèque à la notion d'animat : un certain nombre d'auteurs considèrent au contraire la notion d'adaptabilité (et pas seulement d'adaptation, telle qu'obtenue par *design*) comme primordiale. Elle permet en effet en principe une plus grande robustesse des comportements existant face à un environnement se modifiant (en particulier dans un cadre coévolutif), voire l'enrichissement du registre comportemental.

Pour illustrer cette tendance, nous allons présenter dans les sections suivantes les travaux de deux équipes développant des animats dotés de capacités adaptatives.

2.5.1 ALECSYS

Dorigo et Colombetti [33, 32] étudient l'utilisation d'outils d'apprentissage par renforcement pour la génération de comportements. Ils utilisent pour cela une procédure qu'ils qualifient de *shaping* : après avoir défini une série d'architectures et une tâche plus ou moins complexe (réactive ou nécessitant une certaine mémoire), ils fournissent à l'agent (simulé dans un premier temps, puis transposé dans un robot) des retours sur ses performances lui permettant d'apprendre les types de comportements attendus et la manière de les coordonner.

Nous allons détailler certains des aspects de ces travaux.

Tâches

Les auteurs font la distinction entre des tâches de type *stimulus-réponse*, connexions directes entre les capteurs et les effecteurs, et les tâches *dynamiques* nécessitant l'existence d'états internes à l'agent. L'essentiel de celles qu'ils ont étudiées sont du premier type : s'approcher d'un objet (fixe ou mobile) ou fuir. L'idée est cependant que l'apprentissage de ces tâches et de leur coordination permet en principe, dans une approche *bottom-up* caractéristique des animats, la génération de modèles comportementaux complexes.

Les coordinations de comportements attendues des animats peuvent être du type :

- Somme indépendante : les comportements se déroulent indépendamment, en parallèle.
- Combinaison : les réponses à des stimulus sont combinées afin de produire un comportement cohérent (fuite d'un prédateur tout en évitant un obstacle par exemple).
- Suppression : un comportement a priorité sur un autre.
- Séquence : les comportements sont enchaînés.

Ces modalités de coordination sont combinables entre elles.

Outil de Contrôle

Les robots sont contrôlés par ALECSYS, un système de classifieurs enrichi d'un algorithme génétique. Cet outil est conçu pour être implémentable efficacement sur un réseau de machines, cette parallélisation permettant d'accélérer l'apprentissage.

Il s'agit d'un ensemble de classifieurs pouvant apprendre des tâches élémentaires ou de coordination (l'architecture retenue par le concepteur du système décidant du rôle de ces classifieurs). Les messages fournis aux classifieurs sont issus des capteurs ; ils activent un ou plusieurs classifieurs en fonction de leur adéquation au stimulus courant et de critères probabilistes. Les messages ainsi générés sont des *propositions d'actions* ; si nécessaire, ils sont arbitrés par d'éventuels classifieurs de coordination, puis l'action résultante est jugée par un mécanisme de critique, dont l'évaluation est fournie au système de classifieurs afin qu'elle se répercute sur les règles utilisées.

Afin de permettre l'apprentissage de tâches plus complexes que des associations stimulus / réaction (dans ce cas, le suivi d'un objet pouvant être temporairement en occlusion), un mécanisme de mémoire est fourni aux animats lors de certaines expérimentations. Il s'agit plus

précisément d'une forme de rémanence sensorielle : aux capteurs est associée, en plus des messages indiquant leur état instantané, une autre série de messages décrivant leur évolution récente. Cette notion rejoint finalement celle de fusion entre les stimuli externes et des "perceptions virtuelles" telles que décrites par Wilson [113] (cf. section 2.3.2).

Architectures Proposées

En principe, un comportement complexe peut être appris par un seul système de classifieurs. Mais l'apprentissage est dans ce cas ralenti, l'espace de recherche pour la création de règles adaptées et l'assignement des crédits étant dans ce cas trop vaste. Les auteurs proposent donc de tenter de le factoriser en un ensemble d'espaces plus petits, via l'utilisation d'un ensemble de classifieurs coordonnés. L'architecture de cette hiérarchie est conçue avant l'expérimentation, le système d'apprentissage ayant ensuite la responsabilité de structurer les comportements dans ce cadre.

Trois types d'architectures ont été retenues :

- Monolithique : le contrôle est assuré par un seul système de classifieurs, exploitant les messages issus des capteurs et dont les productions sont directement utilisées par les actionneurs. C'est un schéma de contrôle classique, mais qui ne permet aucune simplification de l'espace d'apprentissage.
- Plate : un ensemble de systèmes de classifieurs est utilisé ; tous ont accès aux perceptions et peuvent envoyer des messages aux actionneurs. Un arbitrage doit être prévu (priorité entre les classifieurs par exemple) en cas de décisions contradictoires. Ici, la coordination de tâches s'excluant mutuellement peut être partagée entre plusieurs classifieurs, ce qui simplifie l'apprentissage.
- Hiérarchique : tous les classifieurs n'obtiennent pas leurs messages en entrée depuis les capteurs : certains utilisent pour cela des messages issus d'autres classifieurs. Ils peuvent aussi être partitionnés en couches, les plus basses étant connectées aux capteurs et proposant des comportements élémentaires, et les plus hautes coordonnant ces comportements.

Dispositif d'Apprentissage

Le dispositif d'apprentissage retenu repose sur une notion de *critique* : l'animat reçoit régulièrement des retours sur ses actions, qui agissent comme des suggestions lui permettant d'adapter ses comportements. Cette approche est un intermédiaire entre deux techniques. L'apprentissage par l'exemple, d'une part, consiste en une présentation régulière de la "sortie" désirée à l'animat. Cette approche ne serait guère adaptée ici, le comportement correct pouvant difficilement être exhibé. D'autre part, l'apprentissage par renforcement consiste en un retour (positif ou négatif) seulement lorsqu'un objectif ou un fait marquant est atteint. Les auteurs considèrent que cette approche, bien que réaliste, mène à des vitesses d'apprentissage trop faibles.

Ils ont donc choisi d'utiliser un *entraîneur*, qui examine les actions de l'animat dans son environnement et les critique régulièrement. Afin de gagner du temps, en particulier lors des simulations, ils utilisent un *programme de renforcement* (*reinforcement program*, RP). Ce programme peut être considéré comme une description de haut niveau des comportements atten-

dus : à charge pour le processus d'apprentissage de le convertir en un programme de contrôle effectif correspondant à cette spécification.

Les auteurs parlent de *situated translator* pour décrire ce processus : c'est l'interaction du robot avec l'environnement, commentée par le RP, qui permet la sélection d'un contrôleur efficace. L'accent mis sur le *physical grounding* caractéristique de l'approche animat se retrouve donc ici. Contrairement à d'autres travaux, cependant, les informations utilisées par le RP ne sont pas nécessairement issues du robot ; il doit en effet observer non seulement le robot, mais aussi son environnement sans être forcément limité par les détails d'implémentation du robot. Cette plus grande abstraction évite d'avoir à écrire un RP trop complexe et difficile à réutiliser, qui ne serait finalement qu'un programme de contrôle déguisé pour un robot précis (ce qui diminuerait d'autant l'intérêt de ce dispositif d'apprentissage). Expérimentalement, cette approche de "récompense du résultat" (ce qui est effectivement observable) par opposition à la "récompense de l'intention" (les ordres moteurs du robot) s'avère mener à une plus grande résistance au bruit ou à l'altération du robot, l'impact de ses actions sur l'environnement étant pris en compte par le superviseur [32].

Un autre paramètre est à considérer : l'ordre d'apprentissage des tâches, et son impact sur les systèmes de classifieurs. Cette *shaping policy* peut être de type *holistique* (l'ensemble des systèmes de classifieurs est considéré comme un système unique, donc entraîné globalement), ou *modulaire* (seul un sous-ensemble des composants est entraîné à un moment donné).

Supports

Les expérimentations se font dans un premier temps dans une simulation, dont l'intérêt est essentiellement une facilité de manipulation et la possibilité d'un apprentissage accéléré. Elle représente un modèle aussi exact que possible des robots que nous avons décrits. Un environnement et des robots bien plus complexes pourraient être simulés, mais cela irait à l'encontre de l'objectif des auteurs, la transposition des résultats d'apprentissage dans des robots réels.

Le résultat de cette phase initiale est ensuite utilisé pour contrôler un robot, l'Autono-Mouse, apparemment développé de manière ad hoc par les auteurs. Il s'agit d'une plate-forme à roues dotée de capteurs photosensibles à des positions variées, d'un microphone, d'un sonar et de capteurs de contact sur certaines versions (figure 2.1). Les capacités de calcul du processeur embarqué servent essentiellement à l'interfaçage avec les capteurs et actionneurs et à la communication avec une unité centrale, celle-ci se chargeant du contrôle proprement dit.

Résultats

Les résultats obtenus confirment l'importance de l'adéquation entre l'architecture retenue pour un animat et les tâches qui lui sont confiées. Si une structuration plate permet un apprentissage efficace de tâches exclusives, une organisation hiérarchique permet de tenir compte d'interactions plus riches entre les tâches, telles que décrites section 2.5.1 (c'est en particulier le cas de la suppression). Réciproquement, certaines tâches s'avèrent impossibles à apprendre si elles sont incompatibles avec l'architecture utilisée. De manière générale, une compréhension minimale de la structuration requise pour un animat accélère grandement son apprentissage.

FIG. 2.1 – Plate-Forme AutonoMouse

L'influence de la *shaping policy* est elle aussi importante. La plus efficace, dans le cas de structurations hiérarchiques, semble être l'entraînement des couches inférieures, puis des couches de coordination, et enfin un apprentissage holistique afin d'affiner leur coopération. Ce résultat était prévisible, puisqu'il permet une décomposition des tâches d'apprentissage tenant compte de la structure d'apprentissage sous-jacente (et donc, si celle-ci est adaptée en vertu des remarques précédentes, aux tâches à apprendre). De manière intéressante, un parallèle peut être fait entre ce résultat et la méthodologie de conception de Brooks : mise en place de comportements élémentaires, travail sur leur coordination via la *subsumption* et apprentissage modéré final.

Enfin, la transposition des systèmes de classifieurs dans un robot une fois leur apprentissage initial effectué en simulation est prometteuse. Après des résultats initiaux corrects, montrant par là le réalisme suffisant de la simulation pour les tâches considérées, une phase d'apprentissage modérée (beaucoup plus rapide que lors de l'apprentissage initial en simulation) permet une adaptation fine de l'animat aux spécificités de son système sensori-moteur et de son environnement.

Discussion

L'intérêt de cette approche est multiple :

- L'utilisation d'architectures multiples fournit des résultats sur la problématique de l'adéquation entre la structure de l'agent et la tâche qu'il doit effectuer, telle que la pose R. C. Arkin. Dans le cas des architectures hiérarchiques, les auteurs avancent que le fait de laisser au robot l'apprentissage de la coordination de ses comportements a en outre l'intérêt de permettre l'apparition de structures qu'une analyse modulaire classique aurait pu laisser de côté. Cependant, le fait que l'architecture de contrôle soit déterminée initialement par l'expérimentateur ne permet pas de confirmer cette hypothèse .
- Les résultats montrent l'intérêt de l'approche *robot shaping* des auteurs. Ils considèrent en effet que les interactions entre un agent et son environnement sont le plus souvent trop complexes à analyser, ce qui limite donc l'approche *smart design* (étude de ces interactions afin de les orienter dans un sens exploitable via l'écriture d'un contrôleur explicite) à des tâches simples. Ils défendent donc l'idée qu'une période d'apprentissage est nécessaire afin de mettre en place les comportements du fait des interactions avec l'environnement. Ils ne rejettent cependant pas l'utilité d'un *design* initial de qualité, rejoignant Brooks en ce sens.
- Ces expérimentations ne se cantonnent pas à des simulations ; ces dernières ont simplement pour objectif d'accélérer l'apprentissage initial. Elles se situent ainsi dans la mouvance des *embedded agents* : l'objectif est de voir les agents réaliser leurs tâches dans un environnement réel, en temps réel.

Il est important de remarquer que, même si une hiérarchie peut exister entre les systèmes de classifieurs, il ne s'agit pas d'une architecture à couches d'abstraction croissante (décomposition fonctionnelle) telle qu'utilisée en IA traditionnelle. Les modules de plus bas niveau ne sont pas en effet orientés perception ou action, mais sont directement comportementaux (donc autonomes). Les modules chargés de la coordination décident du comportement le plus adapté à un moment donné, mais n'ont pas à "interpréter" la situation ou à planifier les actions.

Nous allons maintenant présenter un second exemple de travaux en robotique adaptative.

2.5.2 Nolfi *et al*

L'équipe de Stephano Nolfi s'est intéressée au problème de la transposition d'un comportement appris dans un simulateur vers un robot réel [68]. Contrairement aux travaux de la section précédente, l'accent a été mis sur une modélisation détaillée des capteurs et actionneurs du robot et sur l'adaptation à leurs spécificités plutôt que sur l'apprentissage de comportements complexes.

Support

L'objectif de ces travaux est le contrôle d'une plate-forme classique en robotique, le Khepera. Ce robot développé à l'E.P.F.L. est une structure circulaire d'un rayon de 5cm. Il est équipé de deux roues pouvant être pilotées finement et d'une série de capteurs de proximité à infra-rouge. Il est piloté par un micro-contrôleur embarqué prenant en charge les tâches d'entrées-sorties locales et de communication avec un ordinateur hôte.

Tâche

L'environnement est composé d'une arène rectangulaire au centre de laquelle sont disposés trois obstacles circulaires. L'ensemble de ces éléments est d'une couleur claire, ce qui simplifie le travail des capteurs de proximité. L'objectif assigné à ce robot est de se déplacer rapidement, si possible en ligne droite, tout en se tenant autant que possible à distance des obstacles. Il s'agit d'une tâche classique d'évitement d'objets fixes modulant une trajectoire, déjà étudiée par d'autres auteurs.

Problématique

Les auteurs s'intéressent à la question de l'adéquation de comportements appris en simulation pour le contrôle de robots réels. Les simulations sont d'une grande aide, et s'avèrent même indispensables en robotique évolutive dès lors que des tâches nécessitant une longue période d'apprentissage sont étudiées.

Se pose cependant la question de la validité des résultats obtenus. Par essence, cette approche s'oppose en effet au *physical grounding*, avec les contraintes que cela implique : difficultés de simulation des dynamiques de l'environnement pouvant mener à des programmes de contrôle inefficaces en pratique, et plus généralement à l'investissement dans la résolution de problèmes inexistant lors de l'usage d'un robot physique (cf. section 2.3.2).

Les auteurs proposent d'approcher ce problème via une amélioration du modèle du robot. En effet, contrairement aux simulations où les actions des robots aboutissent toujours aux résultats désirés et où les capteurs renvoient des informations exactes, les robots réels sont confrontés à des capteurs bruités, aux réponses variées pour un même modèle (un facteur très important de dispersion des mesures), et à des actionneurs imprécis.

Cette imprécision est prise en compte via l'intégration du bruit au modèle perceptif et actionnel du robot, et le calibrage de ce modèle en fonction du robot effectivement utilisé.

Système de Contrôle Adaptatif

De par sa robustesse au bruit, un réseau de neurones artificiels est utilisé pour l'apprentissage du comportement attendu. Puisqu'il s'agit d'une tâche stimulus-réponse simple, la topologie du réseau est minimale : il s'agit d'un *feed-forward* disposant de huit neurones activés par les capteurs et reliés à deux neurones dont l'activation pilote les moteurs.

L'utilisation d'un algorithme d'apprentissage supervisé classique est ici difficile : il est possible de calculer une note évaluant globalement les performances du robot, mais pas nécessairement de commenter ses activités motrices à chaque instant. Du fait de la globalité de ce retour, les auteurs ont donc choisi d'utiliser un algorithme génétique. Ici, les gènes codent directement pour les poids du réseau, qui n'évoluent pas durant la "vie" du robot. L'encodage est une simple bijection, et seules des mutations de probabilité élevée sont employées pour faire évoluer le génotype.

Dispositif Expérimental

La grande dispersion des résultats pour deux capteurs se trouvant dans une situation identique ne permet pas de se contenter d'un modèle générique de capteur : il faut le calibrer en caractérisant le composant précis employé. Pour cela, une série de mesures pour tous les capteurs infrarouge est dans un premier temps effectuée en plaçant le robot à des positions précises dans l'environnement et en le faisant tourner de 360° (figure 2.2). Les données obtenues permettent de modéliser la réponse de chaque capteur aux différents éléments de l'environnement, réponse qui sera exploitée par le simulateur pour fournir des mesures au contrôleur dans le cadre de la simulation. Cette approche présente l'avantage de générer un modèle tenant compte des spécificités des capteurs, du robot et de l'environnement, donc nécessairement plus précis qu'un modèle mathématique générique.

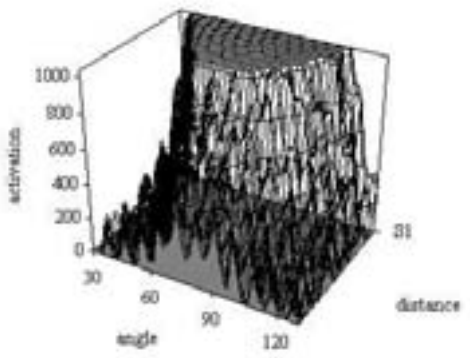


FIG. 2.2 – Modélisation d'un Capteur

Les capacités motrices du robot sont simples à modéliser grâce à l'utilisation de moteurs pas à pas et à sa légèreté, qui rend l'impact de la friction et de l'inertie limité. L'ensemble des combinaisons d'ordres moteurs possibles est envoyé au robot, et les déplacements résultants sont mesurés.

Cette modélisation faite, le robot est positionné à un endroit aléatoire de la simulation, et ses performances sont mesurées. L'ensemble des résultats pour une génération donnée de contrôleurs permet l'extraction des meilleurs individus, qui subissent un processus de mutation suivant l'algorithme génétique puis sont testés à nouveau.

Résultats

Les résultats obtenus par les auteurs montrent qu'un modèle précis de la dynamique robot/environnement peut effectivement être obtenu par échantillonnage du monde via les capteurs et actionneurs du robot. En particulier, bruiter de manière adaptée les données acquises par les capteurs simulés réduit significativement la chute de performances observée lors de la transposition du contrôleur dans le robot. Par ailleurs, un résultat robuste est obtenu si le processus évolutif est poursuivi pendant quelques générations sur le robot physique.

Discussion

Les auteurs admettent la simplicité du dispositif expérimental et de la tâche retenus, et s'inquiètent en particulier des difficultés d'échantillonnage des capteurs en présence d'un environnement plus complexe. Dans ce cadre, ces résultats montrent la possibilité d'utiliser de manière crédible un environnement simulé pour l'apprentissage de comportements. Il est par ailleurs possible de limiter l'impact de leur transposition sur les performances via la prise en compte des caractéristiques spécifiques aux interactions robot/environnement, ce qui est un compromis intéressant entre une simulation purement basée sur un modèle conçu a priori ou un *physical grounding* strict dans lequel tout serait appris directement. Brooks est rejoint dans l'idée que les connaissances disponibles doivent être intégrées à l'animat lors de la conception, et qu'une phase d'*adaptation à soi* est nécessaire pour une adéquation parfaite entre le contrôleur et le robot qui l'embarque.

2.6 Discussion

2.6.1 Qu'est-ce qu'un Animat ?

Afin de pouvoir utiliser l'approche animat dans le cadre de l'animation en informatique graphique, nous allons chercher à caractériser plus précisément cette "nouvelle IA".

Importance de la Robotique

De par ce souci d'adéquation avec un environnement réaliste, beaucoup de ces travaux ont lieu dans le cadre de la recherche en robotique, et bénéficient donc d'une implémentation physique (contrôle d'un robot). Notons cependant que des environnements réels peuvent être

arbitrairement simplifiés jusqu'à permettre des mesures non ambiguës des capteurs et leur interprétation directe. Réciproquement, une simulation d'environnement peut être d'une richesse suffisante pour imposer le développement de contrôleurs de mouvements complexes. L'application d'un outil dans un cadre robotique ne saurait donc être le seul critère permettant de faire d'un robot un animat.

Le point important est qu'un animat est conçu pour effectuer une tâche de manière robuste dans un environnement qui n'a pas nécessairement été conçu pour l'accueillir.

Outils Employés

Si, comme nous le verrons par la suite, des chercheurs dans le cadre de la "nouvelle IA" critiquent les méthodes plus traditionnelles de gestion de tâches (planification via un moteur d'inférences), cela ne signifie pas que ces outils soient en eux-mêmes inexploitables par cette approche du problème du contrôle. Réciproquement, l'usage de réseaux de neurones, de classifieurs ou de toute autre technologie de cet ordre ne suffit pas à faire du système qui les embarque un animat.

Cette approche est en effet plutôt caractérisée par des choix architecturaux guidés par un certain nombre de principes :

- Refus de l'utilisation d'un représentation interne du monde, telle qu'extraite via des "modules perceptifs" et exploitée par un ensemble de "modules décisionnels" dans l'approche SMPA. Il s'agit plus précisément d'éviter le recours à une représentation centralisée, construite indépendamment des tâches en court et exploitée par tout module ayant besoin d'informations. Cependant, le fait de disposer d'un modèle partiel est accepté par Arkin s'il reste local à un module comportemental donné et n'est donc pas partagé [6], et dans une certaine mesure accepté aussi par Brooks même s'il considère de manière générale que "l'environnement est sa meilleure représentation".
- Du point précédent découle que l'accent n'est plus mis sur la notion de traitement et d'exploitation d'un flux d'informations, classique en IA (découpage fonctionnel : abstraction des perceptions dans des modules de reconnaissance, puis exploitation de ces symboles abstraits dans des modules de planification et d'exécution), mais sur l'idée de comportements (traitements d'éléments spécifiques perçus en vue d'une tâche précise). L'effort se décale donc du traitement d'un flux continu de perceptions vers la coordination et l'articulation de comportements.

Rien dans ces conceptions ne condamne les outils de l'IA symbolique, bien qu'ils soient initialement adaptés à la vision "planificatrice" de la robotique. Arkin va d'ailleurs plus loin en proposant une unification des approches planificatrices et comportementalistes (qu'il qualifie de "réactives"). Les modules de comportements seraient ainsi chargés de la bonne exploitation des informations disponibles dans un but précis (et permettraient donc les opportunismes), alors que les modules planificateurs faciliteraient la coordination de ces comportements et l'exploitation d'une connaissance du monde et d'une mémoire permettant d'aller au-delà des perceptions immédiates (indispensables dans tout environnement un tant soit peu réaliste, souvent non-Markovien).

Apprentissage

Des capacités d'apprentissage sont souvent attribuées aux animats. Bien que cela puisse être un facteur de flexibilité face à un environnement évolutif, donc de robustesse, il ne s'agit cependant pas d'une caractéristique qui leur soit intrinsèque.

Elle est en effet associée à des problèmes délicats, tel que le choix de quoi et quand apprendre. Globalement, elle ne présente un réel intérêt que si l'environnement est effectivement évolutif, et prend en particulier tout son sens dans le cas où celui-ci comprend d'autres animats eux-mêmes en situation d'apprentissage. Un mécanisme de coévolution peut alors se mettre en place, et devenir un important facteur d'enrichissement des comportements sous certaines conditions.

A contrario, seules de faibles capacités d'apprentissage sont nécessaires si l'environnement est stable ; il permet alors, par exemple, un calibrage des mécanismes de contrôle, que celui-ci soit rendu nécessaire par leur importation depuis un environnement simulé (ALECSYS) ou par leur *design* a priori (travaux de Brooks).

Dans tout les cas, un animat n'a pas tant pour but d'acquérir plus d'informations sur son environnement (cartographie, apprentissage de classes d'objets...) que de sélectionner les stimuli les plus pertinents et les meilleures stratégies pour mener à bien ses objectifs. L'accent est donc plutôt mis sur l'enrichissement et la bonne coordination du registre comportemental en fonction de l'environnement que sur la construction d'une représentation du monde exploitée abstraitement.

2.6.2 Synthèse : Animation Physique et Animats

Enjeu

L'animation d'êtres vivants peut tirer parti des apports de la nouvelle IA de deux points de vue :

- dans un premier temps, la génération de contrôleurs permettant l'apprentissage (au sens large) de tâches difficiles dans un environnement physique (locomotion, préhension) ;
- par la suite, l'exploitation de ces savoir-faire pour des tâches plus complexes, éventuellement en interaction ou collaboration avec d'autres agents (prédateur-proie, fourmis déplaçant des objets...)

De plus, l'accent mis par l'approche animat sur le comportement permet d'aller au-delà des utilisations classiques de l'animation (conception et précalcul d'une séquence, ou exploitation pour la représentation d'un avatar) : l'interactivité qu'elle permet (ne serait-ce qu'à travers la capacité des agents à réagir) peut s'avérer intéressante dans le cadre d'un jeu vidéo ou d'une réalité virtuelle.

Intérêt pour la communauté "Animats"

Au-delà de l'intérêt pour l'informatique graphique de la nouvelle IA en tant qu'ensemble d'outils pouvant répondre à ses besoins en contrôle, ce rapprochement peut être réciproquement fructueux pour cette dernière.

L'approche animat, de par ses présupposés, ne peut en effet se restreindre à des environnements trop simples ou statiques, où l'apprentissage est limité et les tâches irréalistes. Un environnement changeant et complexe nous paraît justement pouvoir lui être fourni par l'intermédiaire des problèmes se posant en animation d'êtres vivants :

- La variabilité des situations peut être fournie par les interactions entre agents (effets de groupes, interactions réactives classiques, mais aussi coopération ou actions "sociales"), ainsi que par la non-répétitivité des tâches à résoudre (en fonction des variations des besoins de "l'animateur" et d'éventuels autres intervenants).
- Le fait d'utiliser une simulation physique mène à une plus grande richesse de l'environnement : un contrôle précis est nécessaire (par exemple, la locomotion n'est pas une compétence triviale par rapport à un micro-monde symbolique), et des interactions minimales entre les agents sont alors automatiques (nécessité par exemple d'évitement pour réaliser des tâches, même non coopératives).

Le fait d'utiliser des outils d'apprentissage plutôt que des algorithmes conçus au cas par cas fourni un cadre de test exigeant pour ces mécanismes.

Les résultats obtenus peuvent d'ailleurs s'avérer intéressant au-delà des problématiques de l'animation du fait du plus grand réalisme de l'environnement, via par exemple une transposition en robotique.

Problématique

La simulation dans le cadre d'un environnement crédible pose le problème de la génération d'un modèle physique complexe ; la communauté de la nouvelle IA peut être plus familière avec ces questions, via son utilisation d'environnements simulés, mais ces derniers ne tiennent en général pas compte des caractéristiques nécessaires à un réalisme visuel suffisant (frottements, collisions complexes...) et les modèles simulés sont souvent des robots à locomotion simplifiée (roues, faibles vitesses).

Il serait donc intéressant de pouvoir utiliser un modèle issu de l'informatique graphique, donc aisément disponible ou réalisable, afin de générer le modèle physique correspondant. Une large variété de géométries et de structures serait ainsi disponible, permettant d'étendre le champ des expérimentations. Par ailleurs, ce type d'outil, à travers les possibilités de visualisation qu'il permet, simplifierait le traitement des résultat.

C'est cette problématique de la génération aisée d'un modèle physique que nous nous proposons de traiter dans le chapitre suivant.

Chapitre 3

Mise en Œuvre

Ainsi que nous l'avons décrit [83], notre objectif est d'animer physiquement un modèle de créature virtuelle tel qu'on peut classiquement en utiliser en informatique graphique. Cela peut être effectué dans le cadre de logiciels d'animation classique, beaucoup permettant l'association à un modèle géométrique existant d'un modèle physique. Mais ce dernier doit être soigneusement conçu à cet effet, ce qui impose des compétences radicalement différentes de celles habituellement nécessaires à un animateur.

Nous allons présenter ici un ensemble d'algorithmes que nous avons conçus et implémentés afin de permettre la génération d'un modèle physique à partir d'un modèle issu de l'informatique graphique. Cette opération est effectuée de manière totalement automatisée, supprimant ainsi toute charge de travail (*conversion burden*) de la part de l'animateur.

Nous présenterons d'autre part les adaptations que nous avons effectuées sur un simulateur physique afin qu'il puisse animer les modèles générés par l'outil précédent.

3.1 Motivation

3.1.1 Intérêt d'un Cadre Commun

Les communautés de l'animation et de la simulation du comportement ont été amenées à échanger un certain nombre de techniques. Nous pensons qu'elles pourraient mutuellement bénéficier d'un rapprochement de ce type via un outil commun conçu autour de la notion de simulation physique, telle que décrite en section 1.3.4.

La notion de comportement est de plus en plus mise en avant en animation ; elle s'éloigne de mouvements préconçus pour s'intéresser à des notions d'interaction et d'autonomie. Ces concepts sont à la base des travaux en robotique comportementale et animats ; l'informatique graphique peut espérer en tirer des outils de contrôle plus génériques et réutilisables que les techniques ad hoc, non adaptatives le plus couramment utilisées.

Réciproquement, les algorithmes d'apprentissage sont souvent cantonnés à des situations et des environnements simplifiés. Bien que ce type de protocoles soit nécessaire pour valider un certain nombre d'idées, la notion de robotique comportementale, par exemple, impose l'immersion dans des situations plus complexes. De ce point de vue, les mondes plus riches

permis par l'informatique graphique et la simulation physique peuvent servir de "challenge" pour des algorithmes adaptatifs, ceci sans les contraintes posées par la construction de robots par exemple.

Il ne s'agit donc pas seulement d'intégrer une simulation physique à des outils existants. Pour l'exploiter pleinement, chaque discipline a tout intérêt à tirer parti des outils de l'autre : variété des modèles et disponibilité d'outils de visualisation de l'informatique graphique d'une part, algorithmes adaptatifs et approche comportementale des animats d'autre part, et ce dans un cadre de travail commun.

3.1.2 Une Convergence de Besoins ?

La notion de boucle de contrôle, fondamentale dans l'approche animat et l'étude des agents autonomes en général, se retrouve de plus en plus en informatique graphique (cf. section 1.3.6). La mise en place d'un outil commun aux deux disciplines intégrant une simulation physique, des méthodes de contrôle et des outils de visualisation faciliterait l'étude et l'exploitation de ce concept.

Dans ce cadre, l'une des difficultés commune aux deux disciplines est la génération du modèle physique de l'objet ou de l'être vivant à simuler. La communauté de l'informatique graphique utilise souvent des modèles ad hoc, et celle de la robotique dispose d'un certain nombre de modèles représentant les robots qu'elle utilise le plus souvent. Il manque cependant dans les deux cas la possibilité d'une production aisée de modèles de ce type, sans connaissances particulières.

Ces besoins communs étant identifiés, il est important de faire le point sur les objectifs et les priorités de ces disciplines, afin de mieux spécifier le type d'outils pouvant leur bénéficier.

Enjeux en Informatique Graphique

L'objectif essentiel est une augmentation de la qualité visuelle des animations finalement générées : les approximations non décelables visuellement ne sont donc pas problématiques, en particulier si elles permettent d'accélérer la simulation. Ce dernier point est important, car il simplifie le travail de l'animateur en lui permettant de retoucher plus fréquemment les paramètres de la simulation afin de se rapprocher des résultats désirés. La problématique est la même qu'en rendu d'images, et, tout comme dans ce domaine, un animateur acceptera de consacrer le temps nécessaire à l'obtention de la qualité requise.

Une fois sorti du cadre des simulations les plus simples telles qu'utilisées actuellement dans les jeux vidéos par exemple (collisions, *ragdolls*), des contrôleurs de mouvements sont souvent requis du fait de la difficulté d'un pilotage direct d'une simulation physique sur une structure complexe. Il est important qu'ils soient "scriptables", afin de pouvoir orienter de manière suffisamment précise la simulation. Un animateur acceptera de ne pas contrôler dans le détail une simulation (cas des effets de particules, de foules), mais il est important qu'il puisse paramétrer finement les paramètres de cette simulation en fonction de ses besoins.

Du point de vue de l'animation, l'intérêt de la génération de contrôleurs pour des tâches usuelles telles que la locomotion sur terrain plat ou la préhension d'objets est limitée : ces tâches ont en effet été très étudiées et des algorithmes réalistes peuvent les simuler, dans un

cadre physique ou non. Cependant, d'autres domaines demandent encore un travail important de la part de l'animateur, et gagneraient à être automatisés. Citons par exemple la locomotion sur sol accidenté, pour laquelle une prise en compte des détails du terrain est nécessaire, ou l'adaptation de la démarche du corps à sa stature, qui prend tout son sens dans le cadre d'une animation physique.

Enjeux en Termes d'Animats et Robotique Adaptative

L'objectif essentiel du point de vue de l'étude des animats est de rendre plus complexe et réaliste l'environnement dans lequel les simulations sont effectuées, donc de poser des problèmes plus intéressants et exploitables aux contrôleurs. Tout comme en informatique graphique, une simulation de qualité est nécessaire, mais l'importance de sa précision est variable en fonction des applications envisagées (elle doit être particulièrement précise si l'objectif est une transposition à la robotique, telle celle décrite en section 2.5.1, même si des ajustements s'avèreront toujours nécessaires).

De ce point de vue, l'informatique graphique présente l'avantage de fournir une large variété de modèles d'environnements et d'agents, ainsi que les outils nécessaires pour visualiser et exploiter les animations résultantes. Une différence importante entre ces approches est cependant que dans le cas des animats, la génération de contrôleurs efficaces prime de manière générale sur leur facilité de pilotage : l'essentiel est l'accomplissement de la tâche prévue. De plus, encore plus qu'en informatique graphique, la rapidité de calcul est essentielle : du fait des outils utilisés pour l'adaptation des contrôleurs, un grand nombre de simulations doit être effectué pour l'apprentissage d'une tâche donnée.

Sur la base de ces comparaisons, nous allons tenter de dégager les principales caractéristiques d'un cadre de travail qui pourrait bénéficier aux deux communautés.

3.1.3 Caractéristiques Recherchées

Citons ici nos principaux choix et compromis :

- La simulation physique retenue doit être rapide, un outil précis tel qu'utilisé en ingéniérie n'est donc pas requis : il serait superflu en informatique graphique, et néfaste en simulation comportementale du fait du surcoût en temps.
- Du fait des contraintes en temps de calcul précédentes, il serait souhaitable de pouvoir transposer et distribuer facilement la simulation sur un ensemble de serveurs.
- L'outil d'animation retenu doit être d'un type courant et le plus générique possible, afin que des modèles de ce type soient largement disponibles et que les résultats obtenus soient faciles à exploiter. Il doit d'autre part bien se prêter à la conversion vers le modèle d'objets physiques utilisé.
- Dans les deux cas, la disponibilité d'un outil de prévisualisation des résultats obtenus est utile, mais sa qualité n'est pas primordiale : ce n'est de toute façon pas l'objectif en simulation comportementale, et l'approche informatique graphique implique seulement la possibilité d'exporter les résultats vers des outils professionnels (ce que le point précédent assure). Ceux-ci assureront finalement le rendu de haute qualité.

3.2 Outils Retenus

Nous allons introduire ici les outils d'informatique graphique et de simulation physique effectivement retenus compte-tenu des critères précédents. L'architecture de la plate-forme construite à partir de ces outils sera ensuite décrite.

3.2.1 Animation par Squelette

L'animation par squelette, un modèle d'animation largement utilisé et déjà décrit section 1.2.3, a été retenue dans le cadre de notre travail. Rappelons qu'elle est basée sur deux types d'objets : une *peau*, constituée d'une surface texturée classique, et une arborescence d'*os* ou *liens* similaire à celles utilisées en animation cinématique (section 1.3.2). Une influence est par ailleurs spécifiée par l'animateur pour chaque os, ce qui permet de définir comment celui-ci déformera la portion de peau à laquelle il est lié. Plus spécifiquement, un poids est associé à chaque point de contrôle de la surface pour chaque os le contrôlant.

Modèle Retenu

Plusieurs approches étaient possibles pour l'implémentation d'un système d'animation par squelette.

▷ **Interfaçage avec un logiciel 3D :** les logiciels de synthèse d'images incluent tous les fonctionnalités nécessaires à une animation par squelette, et sont souvent extensibles par le biais de *plug-ins* et de langages de script dédiés (nous avons ainsi pu ajouter un modèle de déformation inédit dans le logiciel Maya, par exemple [84]). Cependant, la spécificité des interfaces logicielles à utiliser limite d'autant les possibilités de migration du code écrit à leur intention. Par ailleurs, le fait de requérir un logiciel spécifique, fut-il aisément disponible comme l'est Blender [13] (sur lequel nous avons envisagé un portage de nos méthodes) restreint malgré tout son audience. Enfin, l'interface d'un logiciel 3D ne nous paraît pas être la plus adaptée à l'approche par optimisation, intensive en calculs. Nous préférons donc pour l'instant envisager l'interfaçage avec des logiciels d'animation sous la forme d'une exportation de séquences de mouvements finales depuis nos outils.

▷ **Écriture du code :** il est naturellement possible de réimplémenter les algorithmes d'animation par squelette, ceux-ci n'étant guère complexes. Mais au-delà du temps ainsi perdu, se pose la question du choix d'un modèle d'animation spécifique, les variations possibles étant importantes. Par ailleurs, ces choix sont compliqués par la nécessité de prévoir une importation de modèles existants afin d'en disposer en quantité et variété suffisantes.

▷ **Réutilisation de code :** Pour pallier aux contraintes précédentes, nous avons choisi d'adapter un code d'animation fourni dans le *System Development Kit* associé au moteur de jeu Half-Life et fourni gracieusement par la société Valve [107]. À l'intérêt d'un code d'animation par squelette et de rendu immédiatement disponible s'est ajouté le fait que les méthodes nécessaires au chargement du format de fichier retenu par cette société pour ses modèles sont disponibles.

Il est ainsi possible de tirer parti d'un grand nombre de modèles et d'animations d'origines variées ; ce sont ceux-ci que nous retrouverons dans la suite de ce chapitre (l'un d'entre eux est illustré figure 3.1 à titre d'exemple).

FIG. 3.1 – Modèle Animé par le SDK *Half-Life*

Afin de simplifier la présentation de nos algorithmes, nous allons décrire plus en détail le modèle d'animation utilisé par ce code. Il est cependant important de garder à l'esprit que nos travaux sont adaptables à une large variété d'autres modèles d'animation par squelette sans modifications majeures ; ils sont en effet dans une large part indépendants des spécificités du processus de pondération et de déformation.

A chaque os est associée la liste de ses enfants (os plus éloignés de la racine lui étant rattachés) ; sa pose est spécifiée en utilisant un couple constitué d'un quaternion (pour l'orientation) et d'un vecteur cartésien (pour le positionnement), qui suffisent à définir un changement de repère. Si q et p désignent respectivement les composantes rotationnelles et positionnelles d'un os donné, sa matrice de transformation associée est alors :

$$M_b = M_p.M_q$$

Le plus souvent, afin de limiter les variations de volume du modèle, la composante positionnelle de l'articulation s'avère fixée. La peau est définie comme un maillage de triangles texturés, celui-ci étant animé au niveau de ses sommets. A chacun d'entre eux est associé un os unique ; nous sommes dans le cas spécifique où tous les os ont un poids de 0 ou 1 ; et où il n'existe qu'un os de poids 1 pour un sommet donné. La figure 3.2 est un exemple d'un modèle de ce type, codé par couleurs afin de représenter l'influence de chaque os.

La déformation du maillage est effectuée via une traversée préfixe du squelette : la transformation locale associée à chaque os est calculée puis composée avec la transformation vers le repère global de son parent, et la transformation résultante est appliquée aux coordonnées de l'ensemble des sommets qu'il contrôle.

Un cas particulier est l'os racine, dont par définition le changement de repère s'applique à tous les os fils. Tout sommet étant influencé par au moins un os, l'os racine ne permet donc pas de déformations du maillage, mais uniquement de positionner et d'orienter le modèle dans l'espace global.

La figure 3.3 représente un exemple en deux dimensions de modèle animé par squelette. Il est composé de quatre os nommés $b0$ à $b3$; O est l'origine du repère. Chaque os est représenté

FIG. 3.2 – Influence des Os

par un triangle pointant vers son parent, et le maillage par les pointillés externes.

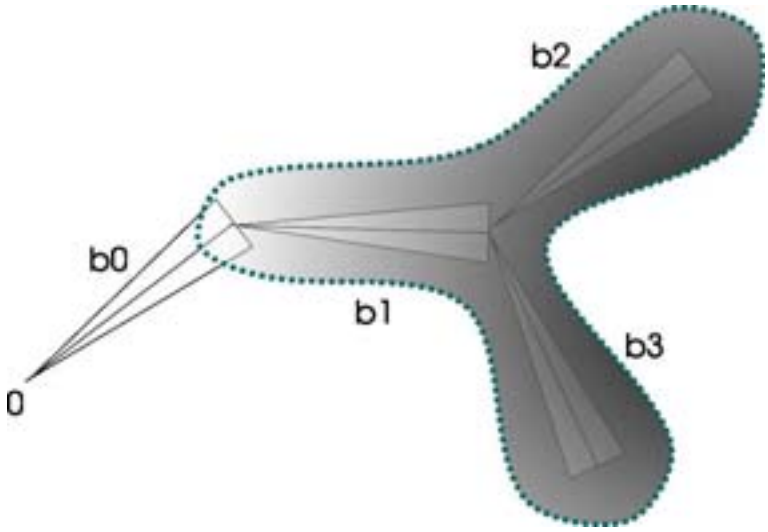


FIG. 3.3 – Animation par Squelette

Une Approche par Feedback

Cet outil d'animation par squelette est utilisable de manières variées, et, on l'a vu en section 1.2.3, s'avère beaucoup plus simple de ce point de vue qu'un outil de bas niveau. Mais, ainsi que nous l'avons décrit précedemment, nous nous intéressons plus particulièrement à son contrôle indirect via un modèle physique. Ce processus a deux aspects : nous devons tout d'abord construire un modèle physique qui soit une approximation aussi bonne que possible du modèle animé par squelette. Le comportement de ce modèle physique sera ensuite simulé,

et fournira en retour les postures articulaires nécessaires à l'animation du modèle graphique initial.

3.2.2 Modèle Physique Utilisé : les ARB

Jusqu'à présent, nous avons simplement considéré un modèle physique comme une approximation pouvant rendre compte des déformations d'une peau mue par un mécanisme d'animation par squelette. En pratique, une large variété d'outils de simulation permettent la représentation d'objets physiques et de leurs comportements.

Les plus génériques d'entre eux sont les simulateurs de corps souples (*soft bodies*). Ils permettent la prise en compte des déformations des objets issues de leurs interactions, mais au prix d'un coût en calculs très important. De plus, même si nous nous permettions cet investissement de ressources, notre modèle d'animation par squelette ne serait pas suffisamment générique pour rendre compte de ces déformations.

Nous avons ainsi choisi d'utiliser un outil de simulation plus spécifique, les corps rigides articulés (*Articulated Rigid Bodies*, ARB). Il est important de remarquer que, ainsi que nous l'avons décrit précédemment [83], les choix d'un modèle physique et d'animation sont profondément liés. Le couple retenu est ainsi une tentative de compromis entre la qualité de l'animation et sa crédibilité, d'une part, et les coûts en calculs associés, d'autre part, suivant les critères de la section 3.1.3.

Le modèle des Articulated Rigid Bodies (ARB) est un intermédiaire entre les modèles souples (*soft models*) et la simulation de solides indéformables. Il permet de représenter les mouvements d'un ensemble de corps rigides en eux-mêmes, mais articulés les uns avec les autres. Du fait de sa capacité à représenter efficacement des objets réels intéressants (tels que les outils mécaniques ou les robots industriels), un effort de recherche important a été investi dans la simulation efficace de cette classe de modèles. Dans la plupart des simulateurs existant, les corps indéformables sont organisés suivant une structure arborescente ; celle-ci est souvent généralisée afin d'inclure des structures closes de type DAG.

De par ces propriétés, il semble raisonnable de considérer la mise en correspondance d'un modèle squeletal et d'un ARB. Mais autant la structure du squelette est similaire aux liens entre corps rigides, autant la continuité de la peau déformée est incompatible avec la nature discrète des composants d'un ARB. Ainsi, un algorithme permettant d'approximer correctement cette surface via un ensemble de corps indéformables s'avère nécessaire.

Un outil de génération d'un modèle physique ARB à partir d'un modèle graphique animé par squelette sera proposé en section 3.5. Mais tout d'abord, la section suivante va détailler l'architecture d'animation et de simulation dans laquelle cet outil prend place.

3.3 Architecture Logicielle

Le schéma représenté figure 3.4 détaille l'ensemble des modules que comporte notre plateforme, et leurs interactions. A la lumière des éléments précédents, ils peuvent être subdivisés

en quatre sous-ensembles.

– La partie gauche permet la conversion d'un modèle géométrique animé par squelette en un modèle physiquement animable. Les différents sous-modules associés seront décrits en section 3.5, et constituent l'essentiel de notre développement algorithmique et logiciel.

– Ces données sont ensuite utilisées dans la partie au centre droit du schéma, qui se charge de la simulation à proprement parler des mouvements du modèle sous la double influence de la simulation dynamique et des forces appliquées au modèle via le système de contrôle. Il sera décrit plus en détails dans la section 3.4, et se base essentiellement sur un simulateur physique tiers, ODE.

– Au fur et à mesure de cette simulation, les paramètres caractéristiques du modèle (les positions et orientations des articulations) sont extractibles de la simulation afin de fournir, lorsque c'est nécessaire, un retour visuel via le modèle géométrique initial (zone supérieure droite de la figure), suivant le mécanisme décrit section 3.2.1. Ceci s'effectue dans la section supérieure droite du diagramme, détaillée sections 3.6 et 3.2.1. Le logiciel nécessaire, ainsi qu'il a été indiqué précédemment, est adapté d'un SDK librement disponible.

– Enfin, un mécanisme de contrôle peut injecter des forces dans le modèle en fonction d'informations issues de l'environnement, ceci afin de l'orienter vers une tâche précise. Les utilisations possibles de ce mécanisme, représenté dans la partie inférieure droite de la figure, seront décrites dans le chapitre 4.

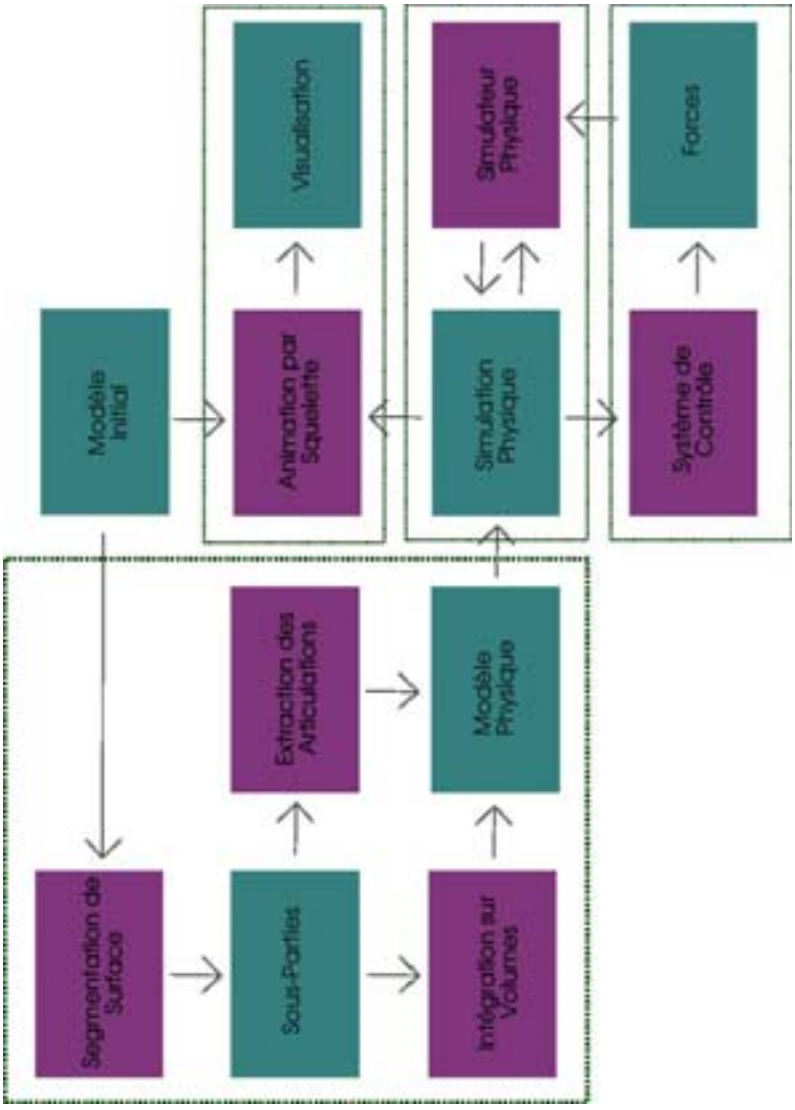


FIG. 3.4 – Modules Principaux et Flux de Données dans notre Modèle

3.4 Implémentation d'une Simulation Physique

Indépendamment de toute considération informatique, l'utilisation d'une simulation physique d'ARB impose la maîtrise d'un certain nombre de concepts issus de la mécanique des solides tels les matrices d'inertie (section A.3.5, les couples et forces internes et externes à un modèle (sections A.1.6 et A.3.8), et la notion d'intégration dans le temps (ces éléments sont détaillés dans l'annexe A).

Cependant, ces connaissances ne suffisent pas à l'implémentation logicielle d'une simulation physique de ce type. Tout d'abord, les limitations qualitatives (nécessité de discrétisation du temps) et quantitatives (précision finie des calculs) propres aux simulations informatiques doivent être intégrées au modèle physique afin d'obtenir une simulation d'une précision et d'une stabilité numériques convenables. Par ailleurs, les algorithmes intégrant ces contraintes doivent conserver une complexité en terme de temps et d'espace mémoire raisonnable compte-tenu de l'application envisagée.

L'ensemble de ces points s'avère délicat à prendre en compte, d'autant qu'ils sont essentiellement contradictoires. Ainsi, et de par l'intérêt d'un simulateur physique efficace, un travail de recherche important a été consenti dans ce domaine. Même muni de ce corpus de travaux, la seule implémentation d'un simulateur physique, qui se fait toujours en fonction d'un compromis spécifique entre les contraintes précédentes, s'avère être un travail considérable. Les simulateurs librement disponibles s'avèrent en général être le résultat publié d'un travail de thèse sur le sujet.

Dans ce cadre, il n'était pas question pour nous de réimplémenter un outil de ce type ; nous nous sommes donc tournés vers des outils de simulation tels ceux précédemment décrits. Un critère important était la disponibilité du code source de ces outils. Même si, ainsi que nous l'avons décrit, la modification des algorithmes au cœur de la simulation n'est pas aisément envisageable sans leur étude approfondie, l'utilisation efficace de ces outils impose de pouvoir étudier et éventuellement remanier les interfaces logicielles utilisées.

Ainsi que nous avons pu le constater, le choix d'une implémentation logicielle spécifique, au-delà des considérations de stabilité, précision et performance, a un impact important sur l'ensemble des outils l'exploitant, en particulier de par une implémentation différente des mêmes concepts (corps rigides simulés, articulations, collisions en particulier). Nous allons décrire ici les deux simulateurs que nous avons eu l'occasion d'utiliser.

3.4.1 Dynamechs

Dynamechs [67] a été notre premier choix de simulateur. Issu des travaux de Scott McMillan dans le cadre de sa thèse [66], il vise à une simulation efficace de structures articulées utilisables en robotique. Pour cela, les matrices utilisées pour la simulation ont été optimisées pour le cas spécifique d'articulations décrites suivant la convention de Denavit-Hartenberg Modifiée (MDH) ; on dispose ainsi du type de liens décrits en section 3.5.4 (les méthodes qui y sont décrites ont d'ailleurs été initialement développées pour l'interfaçage de nos modèles avec Dynamechs).

L'environnement est essentiellement constitué d'un maillage représentant le sol ; aucune interaction entre modèles simulés n'est disponible, bien que le modèle de forces retenu soit

prévu pour être étendu, et permette donc en théorie l'implémentation de ces interactions. Le seul type de représentation géométrique de modèles rigides simulés est le maillage. Une simulation dans le cadre de Dynamechs, utilisant l'outil de visualisation par défaut, est représentée figure 3.5.

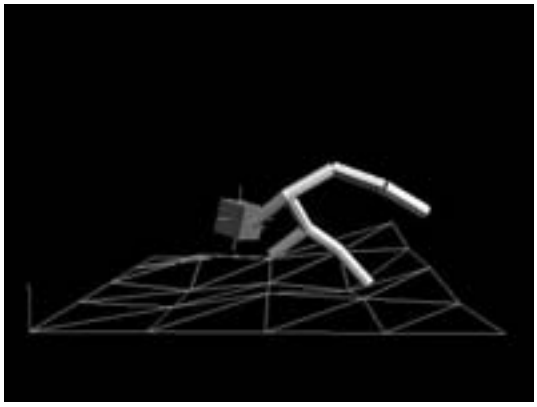


FIG. 3.5 – ARB dans Dynamechs

Le point précédent présente l'avantage de permettre la simulation d'une large gamme de corps, et de manipulations facilitées par l'important volume de méthodes disponibles en géométrie algorithmique. Il est dommage que l'outil de visualisation disponible n'ait pas été écrit indépendamment du code de simulation lui-même, la seule option possible étant sa suppression lors de la compilation. Il a ainsi été nécessaire de remanier le code source afin d'obtenir une meilleure modularité entre la simulation et sa visualisation.

Le modèle de collision retenu s'avère minimaliste : il consiste en une prise en compte des interactions entre le maillage du sol et celui du modèle uniquement au niveau des sommets de ce dernier. Cela permet de se ramener au cas de points matériels, mais montre ses limites dès lors que le sol n'est plus rigoureusement plan : toute zone convexe de celui-ci peut pénétrer assez profondément le modèle avant qu'une collision soit détectée, ce qui mène à la génération de forces de contact de trop grande amplitude.

La présence de forces d'ordres de grandeur très différents dans une même simulation pose des problèmes à tous les simulateurs. Malheureusement, la robustesse de Dynamechs s'avère déjà limitée dans le cadre d'ordres de grandeur raisonnables, et est encore exacerbée par cette gestion des collisions. En pratique, il s'est donc avéré difficile de simuler des modèles relativement complexes, sauf à utiliser des méthodes d'intégration élaborées qui ralentissaient drastiquement la simulation. L'ensemble des simulations stables que nous avons pu étudier portait sur des segments élémentaires simples (un exemple de modèle de ce type, tiré du projet SIGEL [98], est représenté figure 3.6), mais une géométrie plus complexe telle qu'issue d'un modèle arbitraire ne semblait pas facilement simulable.

Ces considérations de stabilité et de performances, difficiles à contourner malgré un travail important sur les paramètres de la simulation, nous ont finalement amenés à envisager l'utilisation d'un autre simulateur.

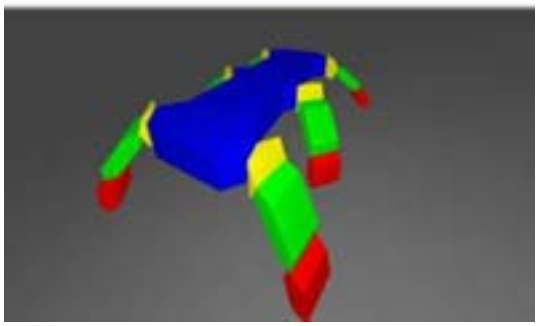


FIG. 3.6 – Robot Marcheur dans Dynamechs

3.4.2 ODE

ODE est un simulateur physique développé par Russell Smith [100]. Il est de conception plus récente que Dynamechs : bien que mettant l'accent sur la stabilité de la simulation, il s'avère en effet largement plus rapide que celui-ci (possibilité de vitesse de simulation plus rapide que le temps réel pour des modèles de complexité moyenne, alors que Dynamechs descend à quelques pour cent du temps réel en cas de collisions complexes). Au-delà de ces caractéristiques, il présente l'avantage d'une plus grande communauté d'utilisateurs et d'une maintenance et évolution active de son code source, ce qui permet une prise en main plus rapide et limite les risques d'être bloqué durablement par une erreur de programmation ou un choix d'implémentation problématique.

Des choix différents ont été retenus dans le cadre de l'implémentation de ODE : le sol est ici réduit à sa plus simple approximation, étant constitué d'un simple plan. Une implémentation similaire à celle de Dynamechs, permettant l'utilisation d'un maillage, est en court d'intégration dans le simulateur mais s'avère poser pour l'instant des problèmes de stabilité. Ce choix présente cependant l'avantage de permettre l'utilisation du modèle de collision sol/corps simple décrit ci-dessus, cette fois-ci sans risque pour la stabilité de la simulation.

Par ailleurs, des collisions réalistes entre les objets simulés sont disponibles, de par une modélisation des forces de contact et éventuellement des frottements (sections A.1.6 et A.3.8). Ceci est rendu possible par la restriction des variétés de corps simulés à des primitives simples : sphères, parallélépipèdes rectangles, cylindres et leurs combinaisons (une simulation typique est présentée figure 3.7).

Les articulations disponibles sont du même type que celles de Dynamechs ; elles présentent cependant l'avantage d'être plus faciles à paramétrer, leurs coordonnées étant simplement fournies sous forme d'un couple quaternion / vecteur de position. Ceci, et la disponibilité d'articulations de type *ball-and-socket*, ne rendent cependant pas les méthodes de conversions articulaires décrits section 3.5.4 nécessairement obsolètes. En effet, bien qu'un modèle puisse en première approximation utiliser des articulations génériques à trois degrés de liberté, la prise en compte des contraintes articulaires propres à chaque modèle peut mener à la composition

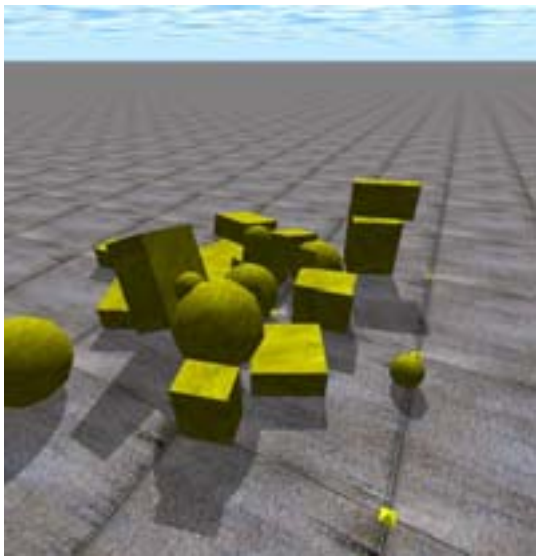


FIG. 3.7 – Primitives Simulées dans ODE

d'articulations plus spécifiques. Ce point sera détaillé en section 4.2.1.

3.4.3 Adaptations du Simulateur Retenu

Nous avons donc finalement fait le compromis de travailler dans le cadre de ODE, abandonnant ainsi temporairement la possibilité d'un sol complexe (bien que des primitives immobiles puissent y être intégrées). Une autre contrainte, plus problématique par rapport au modèle physique initialement retenu, est l'absence de possibilité de simulation de corps autres que des volumes primitifs.

Pour y pallier, nous avons exploité l'architecture extensible d'ODE afin d'y implémenter l'utilisation de maillages en tant que primitive simulable. L'intégration à proprement parler d'une nouvelle primitive géométrique n'est pas problématique : il s'agit essentiellement d'interfacer une structure de données la stockant (nous avons ici réutilisé les outils décrits en section 3.5.1), et éventuellement de mettre à jour le code de visualisation optionnellement utilisable avec ODE. Le simulateur physique à proprement parler n'est en fait pas affecté par le type de primitives manipulées, puisque leurs mouvements propres sont essentiellement déterminés par leur matrice d'inertie (que nous savons calculer, ainsi qu'il sera détaillé en section 3.5.3).

Cependant, un autre élément important doit être mis en place afin d'obtenir une simulation complète : la gestion des collisions entre les maillages et leur entourage. Ce type de code étant complexe à mettre en place et dépendant de l'usage attendu du simulateur, ODE en fournit une implémentation ad hoc pour chaque type de primitive mais permet facilement leur remplacement ou extension. En nous basant sur le code utilisé pour la collision des primitives existantes, il a été relativement aisé d'implémenter des méthodes de collisions réalistes pour les contacts maillage/plan et maillage/sphère, ce qui permet une simulation minimale (sol plan). Les autocollisions (maillage contre maillage), beaucoup plus intéressantes (elles permettent d'envisager

des interactions entre modèles et la présence d'un sol accidenté) sont, elles, beaucoup plus difficiles à mettre en place.

En effet, des bibliothèques de fonctions très efficaces existent pour la détection de pénétrations entre maillages ; dans le cadre de nos travaux, nous avons ainsi retenu pour ses performances V-Clip [72] écrit par B. Mirtich (qui le décrit dans [74]). Cependant, la détection d'un contact entre deux maillages n'est que la première étape d'une détection de collision effective : il faut ensuite générer un ensemble de points de contact et une profondeur de pénétration, ce qui s'avère difficile à obtenir de manière robuste et efficace. Des tentatives de généralisation à partir des méthodes utilisées pour les collisions entre parallélépipèdes ont été tentées, mais sans résultats probants dans le cas général. Nous discuterons dans la section 4.2.2 d'un certain nombre de travaux et possibilités à ce propos.

3.5 Génération d'un Modèle Physique

La génération d'un ARB approximant le modèle initial pourrait en première analyse être aisément résolu en associant à chaque portion de squelette un modèle rigide simple, tel qu'un cylindre, qui tiendrait lieu d'approximation grossière de chaque membre ou segment du corps. Cette technique est par exemple utilisée pour représenter approximativement les robots manipulateurs.

Mais ce choix fait, il n'y aura aucune garantie du point de vue de la qualité ou de la crédibilité des mouvements résultants. En effet, le modèle physique ne disposant pour chaque segment que d'une matrice d'inertie basée sur une approximation du segment initial, il ne réagira pas à l'identique aux forces qui lui sont appliquées. Mais surtout, il interagira de manière complètement différente avec le sol ou d'autres objets, du fait de collisions ne se faisant encore une fois que sur des modèles simplifiés de ses segments. Le risque est donc de se retrouver précisément avec le type d'artefacts que la simulation physique vise à éviter, à savoir des interactions avec l'environnement irréalistes (pieds pénétrant le sol durant la marche par exemple).

Par conséquent, l'objectif de notre algorithme de segmentation de surface est d'associer à chaque os du modèle initial un maillage approximant la surface qui l'environne. L'union de ces maillages devra être égale au volume initialement enclos par la surface, et leurs intersections deux à deux devra être vide afin de conserver les propriétés du modèle d'origine. Après un traitement idoine de ces maillages leur assurant des propriétés géométriques convenables, leurs propriétés physiques seront finalement calculées, ainsi que les caractéristiques des liens les articulant entre eux.

L'ensemble de ce processus est effectué via une série de passes, chacune obtenant ses données de la précédente (les images utilisées sont issues de notre implémentation de ces algorithmes). Nous les détaillerons une à une après une description des classes de bases qui leur sont nécessaires ; un exemple d'application de ces opérations est par ailleurs fourni à la suite de leur présentation, en figure 3.13.

3.5.1 Classes de Base

Les algorithmes décrits ci-après se basent sur des structures géométriques disposant d'opérations bien précises, implémentées sous forme de classes. Ces classes et les méthodes qui leurs sont rattachées vont être décrites, ainsi que les choix d'implémentation retenus.

Squelette

Le squelette est, on l'a vu, l'outil de base permettant d'animer des maillages dans le modèle géométrique retenu. Une abstraction disposant des opérations nécessaires à nos algorithmes est utilisée ici (cela présente l'avantage de ne pas dépendre des spécificités d'un type de modèle donné).

Le composant de base d'un squelette est l'*articulation*, souvent nommée *os* (ces deux termes seront utilisés par la suite). Une articulation est ici entièrement définie par un identifiant unique et un changement de repère. On utilise pour le premier un index entre 1 et nbBones [1] (le nombre d'os). Le second est ici défini via une composante en translation (un vecteur cartésien, ici nommé `position`) et une en rotation (un quaternion).

Le squelette est alors défini comme un arbre d'os, chacun pouvant avoir un ou plusieurs enfants donnés par `children` (et, réciproquement, le parent peut être retrouvé via l'opération `parent`). Un os spécifique est distingué comme étant la racine de l'arbre ; il est renvoyé par la méthode `root`.

À partir de cette représentation, on peut définir la notion de profondeur d'un os, caractérisant sa distance à la racine. Une fonction utilitaire `maxDepthBone` est alors définie pour les algorithmes suivants : elle détermine parmi les (au plus trois) os contrôlant les sommets du triangle passé en paramètre celui de plus grande profondeur.

Les opérations suivantes sont en pratique les méthodes d'un objet squelette. Pour alléger l'écriture, cependant, et ce squelette étant unique pour un modèle donné, on le négligera le plus souvent dans l'exposé des algorithme. Une notation telle que `parent(`*bone*`)` signifiera donc en pratique skeleton.`parent(`*bone*`)`.

Maillages

L'objet traité par ces algorithmes est le maillage polygonal (*mesh*), et plus précisément triangulaire. Ces derniers sont les plus communément utilisés en informatique graphique du fait du caractère planaire des triangles et de la relative simplicité des algorithmes les manipulant en comparaison de ceux s'appliquant aux polygones.

Opérations de Base L'implémentation de ces maillages est ici faite via une classe abstraite nommée `GeoMesh`. Celle-ci peut être considérée comme composée d'un série de sommets (*vertex*), d'une part, et d'un ensemble de triangles (faces) d'autre part. Les opérations de base sur les *GeoMesh* sont l'ajout d'un triangle à un maillage existant (`addTriangle`) et la fusion de deux maillages m_1 et m_2 en un nouveau dont les triangles sont l'union de ceux des triangles

[1] L'implémentation utilise un nombre entre 0 et nbBones -1.

de m_1 et m_2 (`merge`). Un maillage particulier, *le maillage vide* (sans triangles), est dénoté par `emptyMesh`.

Un maillage n'est pas nécessairement connexe ; on peut par exemple obtenir un maillage disjoint via l'union de deux maillages situés à distance suffisante l'un de l'autre. L'opération `extractDisconnectedParts` renvoie alors un vecteur comprenant l'ensemble des composantes connexes du maillage initial. Par définition, leur union est égale au maillage original et elles sont disjointes deux à deux.

Comparaison de Sommets Un point important en géométrie algorithmique concerne le test d'"égalité" entre deux points, qui revient à se demander "deux points sont-ils au même endroit" ? Cette opération aisée mathématiquement pose problème dans toute implémentation informatique du fait de la précision limitée de la représentation des coordonnées. La solution usuelle consiste à décider d'un seuil, petite valeur positive ϵ. La distance entre deux points à tester est alors calculée, et les points sont réputés égaux si cette distance est inférieure à ϵ. Le défaut de cette méthode est dans le calibrage de ce seuil : une valeur trop faible verra des points initialement confondus comme séparés, et un seuil trop élevé fusionnera des points distants. Pour peu que les coordonnées manipulées aient une amplitude importante (via l'union de maillages de tailles très différentes par exemple), et que les maillages soient transformés plusieurs fois via des transformations rigides (changement de repères), deux conditions rencontrées dans nos algorithmes, il n'est pas évident qu'un seuil adapté puisse être déterminé.

Nous avons préféré associer à chaque sommet du maillage un entier unique tenant lieu d'identifiant. Lorsqu'un sommet est dupliqué (dans le cadre d'une opération de découpe par exemple), l'identifiant est lui-même partagé entre ses copies. Il est ainsi possible, indépendamment des imprécisions futures sur les coordonnées, de s'assurer par simple comparaison des identifiants de deux sommets s'ils sont confondus ou non. Cette méthode n'est pas aussi générale que celle du seuil : les sommets à comparer doivent être dans le même repère, et seuls des sommets de même sources sont comparables. Cela suffit cependant à nos besoins et évite toute calibration problématique d'un ϵ.

Ce mécanisme est exploité par une méthode `sew` dont le but est de fusionner les sommets de même position dans un maillage. Ainsi, si un maillage est séparé en deux sous-parties p_1 et p_2, `sew(`*`merge`*`(`p_1,p_2`))` lui sera équivalent. L'avantage de cette approche par comparaison robuste de sommets est qu'elle permet de séparer un maillage en sous-parties, de leur faire subir des transformations, puis de les refusionner sans impact des imprécisions de calculs par simple comparaison des identifiants.

Opérations de Reconditionnement Les maillages sont utilisés en informatique graphique pour décrire des surfaces. Celles-ci ne sont cependant qu'un sous-ensemble de ce qu'un maillage peut effectivement représenter. L'exemple le plus flagrant est représenté par la "soupe de polygones" (*polygon soup*), dans laquelle les intersections entre triangles sont quelconques. En eux-mêmes, ces objets n'ont qu'un intérêt très limité.

Il est donc intéressant de restreindre les maillages à une classe d'objets plus réduite en leur imposant des propriétés plus précises. D'une part, on peut contraindre les intersections

entre triangles à ne se faire que le long d'une de leurs arêtes, qui leur est alors commune. Une deuxième contrainte consiste à imposer une connexité du maillage : il doit alors exister entre tout couple de triangles une chaîne de triangles liés deux à deux par un sommet commun.

Ces deux propriétés sont communément vérifiées par les maillages exploités en informatique graphique, et en particulier par ceux générés lors des phases intermédiaires de l'algorithme d'extraction de maillages décrit par la suite. Contraindre plus les maillages permet cependant d'y appliquer des algorithmes utiles telles les fonctions intérieur/extérieur (présence d'un point dans le maillage) ou d'intégration sur leur volume, deux exemples d'outils nécessaires aux simulations physiques décrites en section 3.4.

La propriété requise par ces dernières opérations est l'*orientabilité*. Il s'agit d'une propriété de la surface ne dépendant pas de sa représentation triangulaire, mais plus généralement de sa topologie. La plupart des surfaces "raisonnables" sont orientables, mais ce n'est par exemple pas le cas du ruban de Mœbius. L'orientabilité est une contrainte forte ne s'appliquant elle-même qu'à une variété particulière de maillages, les *manifolds*. Ceux-ci ont pour propriété que toute intersection entre triangles se fait le long d'un côté (comme décrit ci-dessus), mais de plus ces côtés ne peuvent être partagés qu'entre au plus deux triangles. La fonction `isManifold()` teste cette propriété.

Un *manifold* est alors dit *orientable* si ses triangles peuvent être orientés de manière cohérente. Chaque triangle est lui-même orienté en spécifiant un ordonnancement cyclique de ses sommets. Afin que deux triangles partageant un côté soient orientés de manière cohérente, les deux sommets formant ce côté doivent se trouver dans des ordres inverses dans les ordonnancements de sommets respectifs aux deux triangles.

Soit par exemple la figure 3.8 représentant deux triangles $T0$ et $T1$ partageant un côté commun. Ces triangles seront orientés de manière cohérente si $T0$ est représenté par la liste de sommets $(S0, S1, S2)$ et $T1$ par $(S0, S2, S3)$. Cet ordre de parcours étant trigonométrique, on voit par convention la face externe de ces triangles par rapport au maillage.

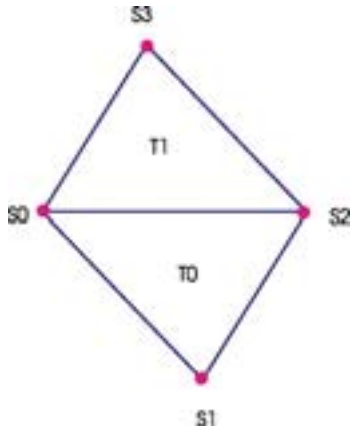


FIG. 3.8 – Orientation Cohérente de Triangles

Avant de décrire l'algorithme utilisé pour orienter le maillage, notons qu'une propriété cruciale ici est que tout *manifolds* clos est orientable s'il ne s'intersecte pas lui-même. Cette dernière contrainte est pathologique de notre point de vue, nous ne cherchons donc pas à y remédier (la surface sera alors remplacée par son enveloppe convexe, ainsi qu'il est décrit plus loin). Ceci admis, s'assurer que les surfaces manipulées sont des *manifolds* clos nous assure donc de l'orientabilité du maillage par l'algorithme précédent.

Une surface dont tous les triangles partagent des côtés à leurs intersections peut ne pas être un *manifold* si ce partage se fait entre deux triangles ; il faut alors supprimer les triangles surnuméraires. Le choix de ce triangle est un problème difficile dans le cas général ; la fonctions utilisée pour cela est `keepManifoldFaces()`. Elle parcourt l'ensemble des côtés d'un maillage et, si l'un d'entre eux est partagé par plus de deux triangles, ceux en surnombre sont supprimés. Les triangles à ôter sont tirés d'un ensemble passé en paramètre. On verra dans la section 3.5.2 la pertinence de cette opération dans le cadre de nos algorithmes.

Si un *manifold* a pu être obtenu, il faut ensuite s'assurer qu'il est clos. Ce test est effectué par la fonction `isClosed()`, qui se contente de vérifier si certains triangles ont moins de trois voisins (ou, ce qui est équivalent, si certains côtés ne sont pas partagés). La fonction `close()` tente quant à elle de clore un maillage. Elle construit pour cela la liste des contours du maillage (listes cycliques de côtés adjacents n'appartenant qu'à un triangle), puis tente de les trianguler. Cela peut cependant échouer si ces contours sont trop complexes (trop concaves et difficiles à approximer par une projection planaire, par exemple).

Si le *manifold* a pu être clos, une technique simple permet de l'orienter : il est garanti que l'algorithme trouvera cette orientation si elle existe et, réciproquement, son échec assure que le maillage n'est pas orientable. L'algorithme 1 décrit cette technique, et l'algorithme 2 la fonction récursive à sa base ; il est implémenté via une méthode `orientFaces()`.

Algorithm 1: orientFaces

```
Data: manifold : le manifold à orienter
Result: oriented : un booléen indiquant si le maillage a été orienté
begin
    forall triangle ∈ manifold do
        unmark(triangle)
    face ← pickAnyTriangle(manifold)
    oriented ← orientRec(face)
end
```

Cet algorithme nécessite de pouvoir marquer temporairement les triangles du maillage : on utilise pour cela les méthodes `mark()` et `unmark()`, `marked()` et `unmarked()` permettant de tester l'état d'un triangle. La méthode `pickAnyTriangle()` choisit un triangle quelconque du maillage afin de lancer l'algorithme ; `shared()` indique si une arête est partagée par plusieurs triangles, alors que `neighbour()` permet d'obtenir le voisin d'un triangle du point de vue de cette arête.

Les méthodes `notCompatible()` et `orientCoherently()` permettent respectivement de tester la compatibilité des orientations de deux triangles vis-à vis d'un sommet commun, et de les orienter si cela est possible en se basant sur le critère d'ordonnancement cyclique décrit ci-dessus.

Algorithm 2: orientRec

```
Data: triangle : un triangle du manifold à orienter
Result: oriented : un booléen indiquant si le maillage est localement orientable
begin
    if unmarked(triangle) then
        mark(triangle)
        forall Côté edge ∈ triangle do
            if shared(edge) then
                neig ← neighbour(triangle,edge)
                if marked(neig) then
                    if notCompatible(triangle,neig,edge) then
                        oriented ← false
                        stop
                orientCoherently(triangle,neig)
                oriented ← orientRec(neig)
    oriented ← true
end
```

Si un maillage s'avère non orientable, un algorithme très classique en informatique graphique permet de générer l'*enveloppe convexe* (*convex hull*) associé à un nuage de points. On peut alors, en négligeant les triangles du maillage, se baser sur l'ensemble de ses sommets et y appliquer cet algorithme. Les algorithmes de génération de *convex hulls* sont un des piliers de la géométrie algorithmique : leur robustesse est garantie, et il en existe des implémentations très efficaces. Nous utilisons *qhull* [9], une version largement éprouvée de cet algorithme ; il est désigné par `qhull` ici.

L'avantage de cette méthode est qu'elle génère à coup sûr un *manifold* clos et sans auto-intersection, donc orientable (cf. section 3.5.1). Cette robustesse se paye cependant par définition par une perte des détails concaves de la surface, ce qui ajoute à l'approximation du modèle. Ce raccourci s'est cependant avéré convenable pour une large gamme de modèles, leurs segments individuels étant le plus souvent essentiellement convexes.

Splittable

Les `Splittable` sont une autre implémentation des maillages disposant d'une opération de découpage le long d'un plan nommée `splitAlongPlane`. Celle-ci prend en paramètre un Splittable et un plan ; celui-ci définissant deux sous-espaces. Elle renvoie alors trois Splittable représentant les surfaces présentes dans chacun des sous-espaces, ainsi que la surface plane enclose dans le contour défini par l'intersection du plan et du Splittable initial. La surface d'intersection et l'un des maillages résultant seront bien sûr vides si le Splittable est entièrement compris dans l'un des demi-plans.

`splitAlongPlane` se base elle-même sur une méthode de découpage des triangles le long d'un plan, créant si nécessaire de nouveaux triangles et arêtes, et classifiant au fur et à mesure les arêtes se trouvant sur le plan afin de définir la courbe d'intersection. Celle-ci est ensuite triangulée afin d'obtenir la surface d'intersection.

On peut librement effectuer des conversions de Splittable à GeoMesh et réciproquement ; celles-ci sont représentées par les constructeurs `GeoMesh()` et `Splittable()`.

Implémentation des Structures Géométriques

Dans une implémentation telle que celle décrite ci-dessus, les structures de données utilisées pour représenter les maillages sont essentielles. Elles conditionnent en effet leur compacité mémoire et les opérations pouvant leur être facilement appliquées.

En parallèle, une étude minutieuse de l'implémentation de ces opérations doit être effectuée afin de s'assurer de leur stabilité numérique ; c'est le domaine de la *géométrie algorithmique* [87].

Plutôt que d'implémenter tout ceci naïvement, nous nous sommes basés sur une bibliothèque C de manipulation de maillages nommée GTS [86], justement axée sur la précision et l'efficacité des manipulations de maillages dans un cadre scientifique. Au-delà d'une encapsulation objet, plus adaptée à notre implémentation C++, notre adaptation essentielle a consisté en l'ajout d'une méthode efficace de découpe le long d'un plan dans le cas des Splittable, et d'un mécanisme de comparaison de sommets (section 3.5.1) facilitant l'opération d'union utilisée lors de l'étape de raffinement des maillages (section 3.5.2).

3.5.2 Algorithme d'Extraction

Les algorithmes permettant l'extraction de maillages à partir d'un modèle géométrique sont décrits ici. Notons que lors de leur implémentation, un modèle objet a été retenu. Cependant, les méthodes exposées ici étant toutes associées à une classe `HLModelExporterHCP` ou dans sa classe mère, `HLModelExporter`, elles sont présentées sous forme fonctionnelle pour alléger l'écriture.

Collections de Données

Par souci de simplification, l'essentiel des collections de données présentées ici sont des vecteurs (collections extensibles et indexées, telles que les `vector` de la bibliothèque C++

STL) alors qu'en pratique des ensembles (arbres équilibrés) ou listes peuvent être utilisés pour diminuer la complexité de certaines séquences. Lorsque cela est pertinent, ces points seront précisés lors du commentaire des algorithmes.

Les principales opérations disponibles sur ces vecteurs sont :
- l'accès aléatoire, dénoté par l'opérateur "crochets"
- l'ajout d'un élément en fin de vecteur, représenté par l'opérateur +=
- `card` : le cardinal d'un ensemble
- `first` : premier élément d'un vecteur / liste

Un deuxième type de structure de donnée est le *mapping*, ou table associative, souvent implémenté sous forme d'une table de hachage ou d'un arbre équilibré. Toutes celles utilisées ici sont homogènes : elles associent un entier à un entier, ceux-ci étant des identifiants d'os, de sommets ou de faces le plus souvent (pour ces derniers, voir la section 3.5.1 ; les os sont tout simplement numérotés de 1 à nbBones ici).

Fonctions Auxiliaires

Une série de fonctions auxiliaires sont utilisées pour simplifier l'implémentation des algorithmes. Citons ici `swap`, qui échange deux valeurs, et surtout `cutByBonePlane`.

Cette dernière fonction teste si un maillage passé en paramètre contient des sommets issus d'une intersection avec le plan de coupe associé à un os donné. Pour cela, elle exploite un *mapping* vIDToPlane contenant une association pour tout sommet construit lors d'un appel à `splitAlongPlane` entre l'identifiant de ce sommet et l'os associé au plan de coupe utilisé.

Algorithme de Haut Niveau

L'algorithme 3 permet le découpage d'un modèle géométrique en un ensemble de maillages (un par articulation, ou *os*) aptes à être utilisés dans une simulation physique (*manifolds* clos). Essentiellement, il s'agit d'une méthode de haut niveau appelant des algorithmes de plus bas niveau, décrits par la suite.

Dans un premier temps, les triangles sont extraits du modèle géométrique afin de construire les maillages initiaux (1). Un découpage hiérarchique des maillages (2) est ensuite effectué ; ainsi, en (3), les sous-parties obtenues se trouvant dans splitted sont redistribuées dans subParts (un vecteur de `GeoList`, elles-mêmes des vecteurs de `GeoMesh`).

Elles sont alors finalement reconditionnées en (4) afin d'obtenir des *manifolds* orientables (cf. section 3.5.1), puis exportées sous forme de coordonnées locales aux articulations, plus adaptées à l'animation par squelette.

Algorithm 3: exportMeshesHierarchicalPlaneSplit

```
Data: model : le modèle géométrique
Result: localMeshRefined : un vecteur de GeoSets (eux-mêmes vecteurs de
        GeoMesh)
begin
1   (localMeshUnrefined, vIDToBone) ← buildMeshes(model)

2   splitted ← ∅
    foreach unrefined ∈ localMeshUnrefined do
        splitted +=Splittable(unrefined)
    (fIDToInter, vIDToPlane) ← recursiveCut(root(model),splitted)

3   subParts ← redistributeSubParts(splitted,vIDToPlane,vIDToBone)
4   localMeshRefined ← rebuildMeshes(subParts,fIDToInter)
    globalToLocalCoordinates(localMeshRefined)
end
```

Extraction Initiale de Maillages

L'algorithme 4 est chargé de l'importation du maillage initial à partir du modèle géométrique, puis d'une première répartition de ses triangles entre une série de maillages (un par os du squelette initial). Compte-tenu du fait que chaque triangle n'est ajouté qu'à un mesh local (point 1), l'union de la surface de ces maillages est exactement la surface originale ; cela restera un invariant durant les opérations ultérieures.

Le maillage associé à chaque os est défini comme l'ensemble des triangles ayant au moins un sommet contrôlé par cet os (cf. section 3.2.1). Si un triangle est contrôlé par plus d'un os, il est alloué à celui ayant la plus grande profondeur dans l'arborescence du squelette, ce qui empiriquement donne les meilleurs résultats compte-tenu des opérations suivantes.

Algorithm 4: buildMeshes

```
Result: localMesh : un vecteur de GeoMesh
Result: vIDToBone : un mapping entier → entier
begin
    (bTriangles,globalVertices,vIDToBone) ← importMeshes(model)
    foreach os ∈ model do
        localMesh +=emptyMesh
    foreach triangle ∈ bTriangles do
        maxd ← maxDepthBone(triangle)
1       localMesh [maxd].addTriangle(triangle)
end
```

Cette méthode se base sur l'algorithme 5, chargé de l'extraction des triangles proprement

dite. Son implémentation n'est pas fournie, puisqu'elle dépend du type de modèle géométrique sous-jacent. Précisons simplement que, ainsi qu'il a été exposé en section 3.5.1, à chaque élément de son paramètre globalVertices est associé un identifiant unique. Par souci de simplification, ces sommets et leurs coordonnées seront considérés par la suite comme rattachés aux sommets des triangles (ainsi, le fait d'ajouter un triangle à un GeoMesh ajoutera simultanément à ce dernier les sommets associés au triangle), bien que la structure de données sous-jacente traite les triangles et sommets indépendamment pour des raisons d'efficacité.

Algorithm 5: importMeshes

Data: model : le modèle à importer
Result: bTriangles : une liste de triangles
Result: globalVertices : une liste de sommets
Result: vIDToBone : *mapping* (identifiant sommet) → (os le contrôlant)

Découpage des Maillages

Dans un modèle d'animation géométrique par squelette, des triangles sont souvent sous l'influence de plusieurs articulations (dans le sens où tous leurs sommets ne sont pas sous le contrôle du même os). C'est justement cette propriété qui permet l'obtention de déformations du maillage au niveau des articulations du modèle (dans le cas contraire, les modèles seraient composés de blocs rigides articulés entre eux, tels les ARBs décrits en section3.2.2).

Pour obtenir des maillages approximant plus précisément l'influence de chaque os que l'algorithme précédent ne le permet, l'algorithme 7 va donc être chargé de découper les maillages précédemment obtenus le long de *plans de coupe* associés aux articulations. Le calcul de l'orientation de ces plans de coupes est confié à l'algorithme 6.

Pour un os donné, le plan associé passe par l'origine du repère local à ce dernier, et est orienté en fonction de sa composante rotationnelle et de celle de ses enfants. Plus spécifiquement, pour un os donné B d'enfants C_i et de parent P la normale à ce plan est (en utilisant le même système de coordonnées pour tous les vecteurs) :

$$\vec{N} = Nm(\vec{p_B} - \vec{p_P}) + Nm(\sum_i Nm(\vec{p_{C_i}} - \vec{p_B}))$$

($Nm()$ étant l'opération de normalisation des vecteurs). Ainsi, si un os n'a pas d'enfant son plan de coupe sera orthogonal au vecteur le reliant à son parent ; dans le cas contraire, il sera médiant au vecteur précédent et à l'ensemble des enfants de l'os. Ce procédé donne de manière générale de bons résultats, cette orientation séparant l'espace entre l'os et son parent de manière conforme à l'intuition.

Algorithm 6: hierarchicalExportCutplane

```
Data: childId : identifiant de l'os fils
Data: fatherId : identification de l'os père
Result: posChild : vecteur position de childId
Result: posFather : vecteur position de posFather
Result: cutPlane : plan de coupe construit
begin
    posChild ← position(childId)
    posFather ← position(fatherId)
    fatherChild ← posChild- posFather
    cSum ← 0⃗
    foreach subChild ∈ children(childId) do
        cSum ← cSum + renorm(position(subChild)-posChild)
    normal ← renorm(fatherChild) + renorm(cSum)
    cutPlane ← plan(posChild, normal)
end
```

À titre d'exemple, la figure 3.9 montre l'ensemble des plans de coupe (représentés sous forme de disques gris) calculés pour le modèle déjà représenté précédemment.

Décrivons l'algorithme de découpe lui-même. Celui-ci devant s'appliquer à l'ensemble du squelette en respectant la hiérarchie des articulations, il est écrit sous forme récursive (un parcours postfixe partant de l'os racine est effectué). Plutôt que de renvoyer des résultats, il travaille par effets de bord sur une partie des paramètres qui lui sont passés, ainsi, il construit progressivement :

- splitted : un vecteur de `Splittable`; chacun de ses éléments contient le maillage actuellement associé à l'os de même identifiant ;
- fIDToInter : un *mapping* des identifiants de faces représentant les intersections avec des plans vers l'identifiant d'os associé ;
- vIDToPlane : un *mapping* des identifiants de sommets appartenant aux faces de fIDToInter vers l'identifiant d'os associé.

Commentons le fonctionnement de l'algorithme. Tout d'abord, afin d'obtenir un parcours postfixe, les os fils de l'articulation courant sont traités via un appel récursif (1). On vérifie ensuite à partir du (2) si des parties de maillage n'ont pas été attribuées à tort à l'os courant, ce qui peut arriver lorsqu'elle ont été héritées d'un autre fils. Ainsi, sur la figure 3.10, si l'os $b2$ est traité avant l'os $b5$ du fait du sens de parcours, l'ensemble du maillage situé à droite du plan de coupe de $b2$ (ligne pointillée) sera attribué à son fils $b3$, y compris la zone inférieure pourtant manifestement influencée par $b5$.

FIG. 3.9 – Plans de Coupe

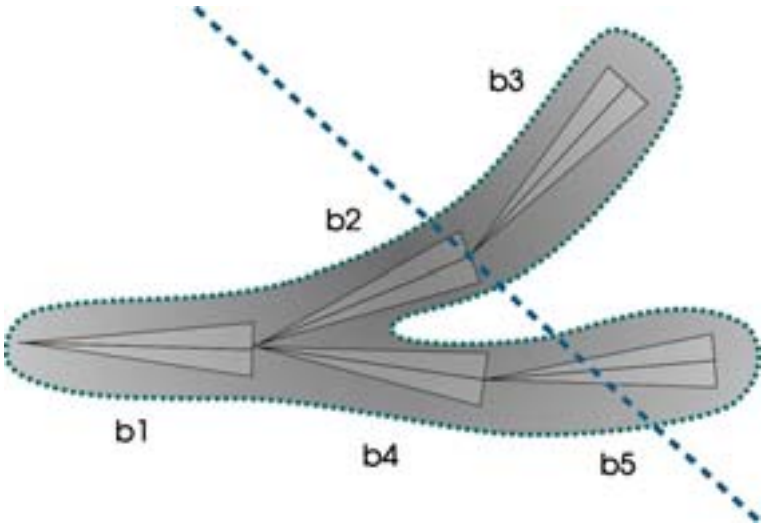


FIG. 3.10 – Erreur d'Attribution de Sous-Partie

Cette vérification d'appartenance est effectuée en redécoupant le maillage courant suivant les plans de coupe associés aux fils[2] (3) et, si le demi-espace associé au fils contient effectivement une portion de maillage, on la lui réattribue (4).

L'utilisation de `swap()` provient du fait que `splitAlongPlane()` renvoie en premier le maillage se trouvant dans le demi-espace négatif (*i.e.* tel que pour tout sommet s du maillage, cutPlane.`getPos(`s`)` [3] est négatif), et en second celui dans l'espace positif. En testant dans quel demi-espace se trouve effectivement l'articulation, on peut attribuer correctement les maillages obtenus à l'os et à son fils.

On traite enfin l'os lui-même (5). On définit et utilise pour cela un plan de coupe séparant au mieux l'os et ses fils via l'algorithme 6 (6). Une caractéristique intéressante de cette opération est qu'elle est la seule impliquant un traitement coûteux des triangles. Cette opération de découpe d'un triangle le long d'un plan s'avère de plus calculable d'une manière bien plus robuste et efficace qu'une transformation CSG générique (c'est la fonction de la classe `Splittable` décrite en section 3.5.1).

On met ensuite à jour les structures de données vIDToPlane et fIDToInter servant à identifier les éléments de zones d'intersection dans les algorithmes ultérieurs (7). Enfin, les sous-parties générées sont attribuées à leurs articulations respectives (8) : la sous-partie se trouvant dans le demi-espace comprenant le parent lui est ajoutée, l'autre étant conservée par l'os (figure 3.10).

Raffinement par Tri des Maillages

L'algorithme décrit ci-dessus permet de définir des zones de l'espace correspondant aux domaines d'influence approximés des os. Cependant, une structure ou pose particulière du squelette peut mener à un maillage concave, auquel cas des parties de ce maillage non directement contrôlées par l'os peuvent se retrouver dans la zone qui lui est attribuée. Par exemple, dans la situation représentée figure 3.11, une zone appartenant à l'os $b1$ sera attribuée par erreur à son fils $b2$ lors de la découpe par le plan $p1$.

Il peut aussi se poser des problèmes d'attributions erronées de mailllages entre les fils d'un même os, tels que décrits précédemment (figure 3.10).

L'algorithme 8 vise à redistribuer les sous-parties mal attribuées en se basant sur des considérations de connexité et, en dernier ressort, sur la prise en compte de l'influence des os. Ceci est effectué via la recherche de points communs entre une portion "orpheline" (associée à tort par un os mais contrôlée par celui-ci) et les portions associées aux os voisins. Pour cela, un mécanisme d'identifiants (décrit en section 3.5.1) est exploité.

Par souci de lisibilité, une partie de cet algorithme relativement long a été extraite sous forme d'une sous-fonction `redistributeWithChildren()` (algorithme 9).

[2] Pour des raisons d'efficacité, ils sont en pratique cachés dans un vecteur indexé par les identifiants d'os ; ils ont en effet déjà été calculés lors d'appels précédents à recursiveCut.

[3] `getPos` renvoie pour un point (vecteur cartésien) et un plan donné le demi-espace dans lequel se trouve le point (soit une valeur positive, négative ou nulle si le point est élément du plan).

Algorithm 7: recursiveCut

Data: bId : l'os à partir duquel découper
Data: splitted : un vecteur de `Splittable` (effet de bord)
Data: fIDToInter : un *mapping* des faces d'intersections vers l'os associé (effet de bord)
Data: vIDToPlane : un *mapping* des sommets d'intersections vers l'os associé (effet de bord)

begin

1 **foreach** *cId* ∈ `children(`bId`)` **do**
  `recursiveCut(`*cId,splitted,fIDToInter,vIDToPlane*`)`
2 bPos ← `position(`*bId*`)`
bMesh ← splitted [bId]
foreach *cId* ∈ `children(`bId`)` **do**
3 cutPlane ← plan de coupe associé à cId
  (bPart,cPart,inter) ← `splitAlongPlane(`*bMesh, cutPlane*`)`
  if *cutPlane.*`getPos(`bPos`)` > 0 **then**
    `swap(`*bPart, cPart*`)`
  if *cPart* $\neq \emptyset$ **then**
4 bMesh ← `sew(merge(`bPart, inter`))`
    splitted [bId] ← bMesh
    splitted [cId] ← `sew(merge(`cPart, splitted [cId], inter`))`
5 **if** *bId* $\neq$ `root()` **then**
  fId ← `parent(`*bId*`)`
  toSplit ← `sew(merge(`splitted [bId], splitted [fId]`))`
6 (cutPlane,cPos,fPos) ← `hierarchicalExportCutplane(`*bId,fId*`)`
  (fPart,cPart,inter) ← `splitAlongPlane(`*toSplit, cutPlane*`)`
  if *cutPlane.*`getPos(`fPos`)` > 0 **then**
    `swap(`*fPart, cPart*`)`
7 **foreach** *Sommet locv* ∈ *inter* **do**
    vIDToPlane +=(locv → bId)
  foreach *Face locf* ∈ *inter* **do**
    fIDToInter +=(locf → bId)
8 splitted [bId] ← `sew(merge(`cPart, inter`))`
  splitted [fId] ← `sew(merge(`fPart, inter`))`

end

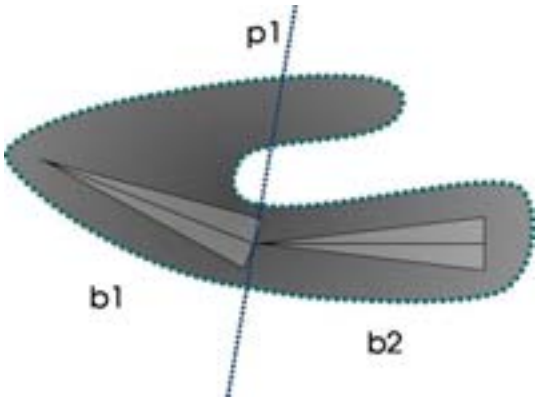


FIG. 3.11 – Problèmes de Coupe

L'algorithme considère donc un à un les maillages générés à l'étape précédente (1) ; après avoir converti chaque Splittable en GeoMesh (2), on écarte le cas particulier de l'os racine (celui-ci n'étant pas à l'origine de découpages).

Les parties disjointes du maillage sont ensuite traitées une à une. Un cas particulier peut alors se présenter (3) : si l'os est une feuille du squelette, on ne dispose pas d'un plan de coupe pour effectuer des tests d'appartenance aux zones (les os sans fils ne subissant pas de découpage). Pour prendre une décision, on exploite alors l'influence de l'os : s'il influence la partie à traiter (*i.e.* s'il existe une relation entre un de ses sommets et l'os dans vIDToBone), on la lui attribue, sinon elle est transférée à son parent.

Le cas général, traité par l'algorithme 9, est cependant celui où l'os traité est un noeud non feuille du squelette. Afin de déterminer l'appartenance de la sous-partie disjointe à traiter, cet algorithme construit tout d'abord une liste des os dont le plan associé a participé à la découpe de cette partie (1). La fonction `cutByBonePlane()` (section 3.5.2) peut efficacement effectuer ce test grâce à vIDToPlane(), construit lors de la découpe récursive (section 8).

Ceci fait, quatre cas sont distingués, en fonction de la présence ou non d'éléments dans la liste précédente, et de l'intersection ou pas du plan associé à l'os courant avec la sous-partie :

- (2) : Elle est en intersection avec les plans associés à cet os et au moins un de ses fils. Elle est attribuée à l'os courant.
- (3) : Elle est uniquement en intersection avec le plan associé à l'os courant. On l'attribue à son parent.
- (4) : Elle est uniquement en intersection avec le plan associé à l'un de ses fils. Il ne devrait alors pas y avoir plus d'un enfant dans la liste (dans le cas contraire, un avertissement est émis). La sous-partie est attribuée à cet enfant.
- (5) : Elle n'est en intersection avec aucun plan, qu'il soit associé à l'os courant ou à ses fils.

Dans ce dernier cas, on commence par construire la liste[4] des os contrôlant les sommets de cette sous-partie (6). Idéalement, un seul os est présent dans cette liste (7) ; on lui attribue alors la sous-partie. Si ce n'est pas le cas, la situation est plus complexe : on est souvent en présence d'un os sans maillage associé, servant d'intermédiaire dans une articulation par exemple.

En dernier ressort, l'ensemble du squelette est parcouru afin de construire la liste des os ayant participé à la découpe de cette partie (8). Une heuristique efficace consiste alors à utiliser le même test que précédemment (9) : si cette partie ne semble être en relation qu'avec un os, on la lui attribue (10). Dans le cas contraire, elle reste en place (11).

Finalement, les maillages sont définis comme l'union de ces sous-parties, et plongés dans le repère local à l'os auquel ils sont associés. Le résultat pour le modèle précédent est présenté figure 3.12 (chaque couleur étant associée à un os). On remarquera les discontinuités de coloration de ce modèle par rapport au modèle d'origine (figure 3.2) : on dispose bien d'une approximation discrétisée des zones influencées par les os.

Algorithm 8: redistributeSubParts

```
   Data: splitted : les maillages à reconstruire
   Data: vIDToPlane : les sommets générés lors d'une coupe
   Data: vIDToBone : l'os contrôlant chaque sommet
   Result: subParts : les maillages finaux (vecteur de GeoList)
   begin
       subParts ← ∅
       for bone ← 1 to nbBones do
           subParts +=GeoList Vide
1      foreach sit ∈ splitted do
           bone ← os associé à sit
2          mesh ← GeoMesh(sit)
           if bone = root() then
               subParts [bone ] +=mesh
           else
               localSubParts ← extractDisconnectedParts(mesh)
               foreach part ∈ localSubParts do
                   if children(bone) = ∅ then
3                      if bone influence part then
                           subParts [bone ] +=part
                       subParts [parent(bone)] +=part
                   else
                       redistributeWithChildren()
   end
```

[4]Pour des raisons d'efficacité, on utilise en pratique un ensemble ici (arbre binaire ou table de hachage)

Algorithm 9: redistributeWithChidren

```
Data: bones
begin
1   cuttingChild ← ∅
    foreach e ∈ children(bone) do
        if cutByBonePlane(part,e,vIDToPlane) then
            cuttingChild +=e
    switch (cutByBonePlane(part,bone,vIDToPlane), cuttingChild ≠ ∅) do
2       case (true,true)
            subParts [bone ] +=part
3       case (true,false)
            subParts [parent(bone)] +=part
4       case (false,true)
            subParts [first (cuttingChild)] +=part
        case (false,false)
5           bones ← ∅
            foreach vertex v ∈ part do
6               if vIDToBone (v) ∉ bones then
                    bones +=vIDToBone (v)
            if bones = { cBone } then
7               subParts [cBone ] +=part
            else
8               cuttingBones ← ∅
                for k ← 1 to nbBones do
                    if cutByBonePlane(part,k,vIDToPlane) then
                        cuttingBones +=k
9               if cuttingBones = { cBone } then
10                  subParts [cBone ] +=part
                else
11                  subParts [bone ] +=part
end
```

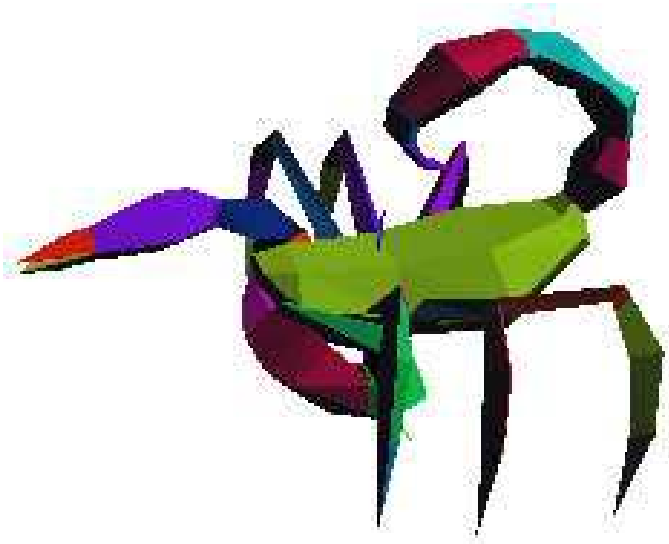


FIG. 3.12 – Modèle Final

Reconditionnement Topologique

L'algorithme précédent construit une série de maillages associés à chaque os d'une manière visuellement plausible. Cela ne signifie cependant pas qu'ils aient les propriétés requises par la simulation physique (être orientables et clos).

L'algorithme 10 vise précisément à générer dans localMeshRefined un ensemble de GeoMesh *manifolds* à partir de ceux de subParts, en utilisant les propriétés et méthodes décrites en section 3.5.1.

Dans toute sa généralité, ce problème de reconditionnement de surface est un domaine de recherche actif ; les algorithmes efficaces s'avèrent souvent complexes et leurs implémentations correctes ne sont pas toujours disponibles librement.

Notre approche de ce problème est médiane. Nous tentons dans un premier temps de pallier autant que possible aux dégénérescences décrites, via l'application d'une série d'algorithmes simples les détectant et y apportant des solutions ad hoc : cela s'avère généralement suffisant. Ensuite, si ces outils s'avèrent insuffisants (du fait de faces non orientables par exemple), nous utilisons une méthode consistant en la génération de l'ensemble des points associés aux sommets du maillage dégénéré, puis en l'extraction de l'enveloppe convexe (*convex hull*) qui leur est associée.

Tout d'abord, une reconstruction exacte est donc tentée. Pour chaque os, l'union de tous les maillages qui lui sont associés est construite, puis on en extrait les sous-parties disjointes (1). On cherche à convertir chacune d'entre elles en un ensemble de *manifolds* (2). Pour cela, on supprime tout d'abord les faces inutiles internes au maillage.

Algorithm 10: rebuildMeshes

```
   Data: subParts : les maillages à reconstruire
   Data: fIDToInter
   Result: localMeshRefined : les maillages construits (manifolds clos)
   begin
       for bone ← 1 to nbBones do
           unionMesh ← emptyMesh
           foreach part ∈ subParts [bone ] do
               unionMesh ← merge(unionMesh,part)
1          usubParts ← extractDisconnectedParts(unionMesh)
           finalParts ← ∅
2          foreach uSub ∈ usubParts do
3              edgesDuplicated ← keepManifoldFaces(uSub, fIDToInter)
               if !edgesDuplicated then
                   finalParts +=uSub
               else
                   subSubParts ← extractDisconnectedParts(uSub)
                   if card(subSubParts) = 1 then
4                      Echec
                   else
                       foreach sub ∈ subSubParts do
5                          sew(sub)
                           finalParts +=sub
6          finalMesh ← emptyMesh
           foreach mesh ∈ finalParts do
               worked ← generateManifold(mesh)
               if worked then
                   finalMesh ← merge(finalMesh,mesh)
           localMeshRefined +=finalMesh
   end
```

Imaginons en effet que deux zones du maillage soient découpées à tort par l'algorithme 7, puis réunies à nouveau par l'algorithme 8. Une surface d'intersection ayant été ajoutée entretemps aux deux sous-parties, leur union comporte maintenant une "cloison" interne. Cela implique l'existence d'au moins trois sommets partagés par au moins trois triangles, et modifie donc la topologie du maillage (qui ne peut par définition plus être un *manifold*).

La fonction `keepManifoldFaces` vise précisément à supprimer ces triangles surnuméraires. Ceci est efficacement effectué en parcourant l'ensemble des sommets et, lorsqu'ils sont partagés par plus de deux triangles, à supprimer ceux se trouvant dans la liste des faces d'intersection fIDToInter (générée par l'algorithme 7).

Cette fonction est donc appliquée au maillage (3). Si aucun triangle n'a été supprimé, le maillage généré n'est pas convertible en *manifold* par cette méthode et est donc stocké pour la phase suivante. Si par contre des modifications du maillage ont bien eu lieu, on cherche le nombre de sous-parties disjointes effectivement générées. S'il y en a plusieurs, on les recoud (opération `sew`) une à une (5) ; le cas contraire génère un échec (4).

Ceci fait, les maillages générés n'ont plus de sommets partagés par plus de deux triangles, mais peuvent encore en avoir n'appartenant qu'à un triangle (présence de trous ou bords). Or par définition, les sommets d'un *manifold* clos, donc orientable, sont partagés par deux triangles exactement. On utilise alors l'algorithme 11, afin de générer une version close, puis orientée du maillage (6). Cet algorithme se base sur les fonctions applicables aux maillages décrites en section 3.5.1. Il commence par tenter de clore le maillage (1) ; si cette opération réussit, on teste si un *manifold* a été généré (2). Si c'est le cas, on l'oriente (3). Dans le cas contraire, les méthodes basiques ayant échoué, on utilise un algorithme de *Quick Hull* (4).

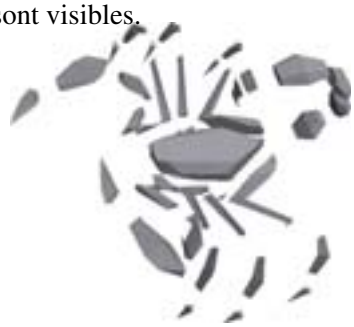
L'union des *manifolds* ayant pu être générés est finalement construite.

Algorithm 11: generateManifold

```
   Data: mesh : le maillage à traiter (modifié par effet de bord)
   Result: gotManifold : un manifold a-t'il été généré ?
1  couldClose ← false
   if !isClosed(mesh) then
       couldClose ← close(mesh)
   if couldClose then
2      if isManifold(mesh) then
3          couldOrient ←orientFaces(mesh)
           if couldOrient then
               return true
4      qhull(mesh)
       orientable ←orientFaces(mesh)
       return orientable
```

Données initiales : le modèle initial est représenté par un maillage unique dont les sommets sont associés aux os (ici, ces associations sont représentées par des couleurs).

Découpe initiale : le maillage est découpé de manière récursive depuis les extrémités des membres jusqu'à la base du squelette. Un certain nombre de sous-parties découpées à tort sont visibles.



Tri des sous-parties : les zones réparties de manière erronée sont associées à l'os le plus approprié. Chaque os est maintenant lié à un ensemble cohérent de maillages.

Reconditionnement topologique : les sous-parties sont réassemblées et le maillage résultant est traité afin d'obtenir un *manifold* orientable.

FIG. 3.13 – Génération de Modèle Physique

3.5.3 Paramètres Physiques

L'algorithme précédent permet l'obtention de maillages associés à chaque os ; ils forment la base de notre simulation, représentant la surface des corps rigides composant notre modèle physique, mais cette information seule n'est pas suffisante pour la définir.

Il est en effet nécessaire de fournir au simulateur les paramètres physiques associés à chacun de ces corps, ceci afin de pouvoir calculer leur réaction aux forces qu'ils subissent. Ces paramètres sont de trois types : la masse, le centre de gravité et la matrice d'inertie. S'ils sont notés respectivement m, G et M_I, on a plus précisément :

$$m = \int_V \rho(M) dv$$

$$G = \frac{1}{m} \int_V \rho(M) M dv$$

$$M_I = \begin{pmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{pmatrix} \text{ avec } \begin{cases} I_{xx} = \int_V (y^2 + z^2) dm \\ I_{yy} = \int_V (x^2 + z^2) dm \\ I_{zz} = \int_V (x^2 + y^2) dm \\ I_{xy} = I_{yx} = -\int_V xy dm \\ I_{yz} = I_{zy} = -\int_V yz dm \\ I_{xz} = I_{zx} = -\int_V xz dm \end{cases}$$

(ces propriétés sont détaillées en sections A.3.4 et A.3.5).

Nous n'avons dans un premier temps pas trouvé d'algorithme pour le calcul de ces intégrales sur nos maillages. Les outils de simulation librement disponibles utilisent le plus souvent des corps basiques (cylindres, parallélépipèdes rectangles) dont les intégrales sont analytiquement connues.

Nous avons ainsi choisi d'implémenter et de tester un certain nombre d'algorithmes simples, en nous basant sur la constatation que les physiciens approximent souvent des corps complexes comme l'union de corps plus simples. Nous avons donc tenté de décomposer nos maillages de manière similaire, suivant deux approches : la subdivision et l'échantillonnage.

Approche par Subdivision

Dans ces algorithmes, nous avons utilisé des schémas de subdivision afin d'approximer un maillage via un ensemble de parallélépipèdes. L'idée de base est de subdiviser la boîte englobante de l'objet le long de son bord le plus long, et d'utiliser des tests spécifiques pour classifier les demi-boîtes résultantes. Si elles sont en intersection avec le maillage, elles devront elles-mêmes être subdivisées. Sinon, elles seront ou abandonnées (si elles sont à l'extérieur du maillage), ou retenues comme éléments d'intégration. Ainsi, à chaque itération, l'ensemble de boîtes retenues constitue une approximation plus fine du volume défini par le maillage initial.

Deux techniques ont été utilisées pour le test d'intersection. La première utilise un test efficace d'intersection boîte/maillage utilisant une structure hiérarchique de type AABB (*Axis Aligned Bounding Box* [110]), alors que le second effectue des manipulations géométriques : à chaque subdivision de boîte, le maillage associé est lui aussi coupé selon le plan médian (une

opération efficace déjà utilisée lors de la génération des maillages), et chacune des sous-parties résultantes est alors associée à la sous-boîte correspondante. Le test d'intersection consiste simplement en une vérification du fait que le maillage inclus dans une boîte donnée n'est pas nul (aucune face).

Le problème inhérent à ces approches géométriques exactes (dans le sens où elles ne fournissent pas de résultats inexacts concertant les boîtes internes et externes au maillage) est qu'elles ne convergent pas nécessairement de manière rapide. Ainsi, le coût en temps associé aux calculs d'intersection et le coût en mémoire lié à la conservation d'une liste d'éléments en cours de traitement limite grandement leur intérêt.

Approche par Echantillonnage

Une autre technique consiste en l'utilisation de méthodes d'échantillonnage ne nécessitant qu'une quantité de mémoire constante, dans l'espoir que les gains en terme de coût en temps et en mémoire compenseraient leur approximation par la possibilité d'utiliser un échantillonnage plus fin. Cette approche n'est pas de type subdivision : elle consiste dans le plongement du maillage dans une grille en trois dimensions définissant un ensemble de boîtes. Ces dernières ne sont pas intersectées avec le maillage ; seuls leurs sommets sont situés par rapport au maillage, en utilisant simplement sa fonction intérieur/extérieur (pour laquelle un *manifold* est requis). Cela permet la classification des boîtes associées en trois sous-ensembles : intérieurs, extérieurs et "en surface". Un calcul à la volée des intégrales recherchées sur les boîtes intérieures évite d'avoir à les stocker, et évite une augmentation du coût mémoire suivant la précision de la grille.

La limitation de cette approche est que, comme toute méthode par échantillonnage, elle peut souffrir d'artefacts. Par exemple, le fait de savoir que tous les sommets d'une boîte sont à l'extérieur ou à l'intérieur du maillage ne permet pas de conclure quant à leurs positions respectives (ainsi, sur la figure 3.14, tous les sommets de la boîtes sont bien hors du maillage, et il y a pourtant bien intersection).

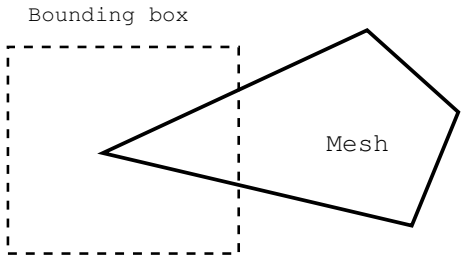


FIG. 3.14 – Problèmes d'Echantillonnage

Se pose donc le problème que, contrairement aux méthodes précédentes, aucune métrique (tel que le volume de l'ensemble des boîtes "en surface") décroissant en fonction de la finesse

de l'échantillonnage n'est disponible[5]. Ainsi, l'évaluation de la qualité du résultat ou de la taille de grille nécessaire à l'obtention d'une précision donnée est délicate.

Approche de Type Delaunay 3D

Ces méthodes d'intégration naïves donnent en pratique des résultats limités sur des maillages complexes, et s'avèrent coûteuses à utiliser. Par ailleurs, il n'est pas évident de définir à partir de quel taux d'approximation les valeurs physiques calculées sont acceptables. Nous avons donc finalement développé un algorithme totalement différent basé sur l'idée que les boîtes ne sont pas une manière naturelle de décomposer un maillage dans le cas général : du point de vue de la géométrie algorithmique, les *simplex* (ici, des tétraèdres) sont bien mieux adaptés. Notre problème devient alors une généralisation dans l'espace de la triangulation de Delaunay, pour laquelle un grand nombre d'algorithmes robustes existent. Ceux-ci retournant un ensemble de tétraèdres dont l'union est le volume original et dont les intersections deux à deux sont vides, la somme des intégrales associées est bien égale à la somme des intégrales sur le maillage initial.

De plus, nous allons montrer que les intégrales auxquelles nous nous intéressons sont facilement et efficacement calculables de manière exacte sur des tétraèdres.

Simplifications En faisant l'hypothèse que la masse volumique ρ du corps est uniforme (constante sur son volume), on peut poser :

$$\begin{array}{rcl} m & = & \rho \int_V dv \text{ (calculable géométriquement)} \\ x_G & = & \frac{1}{V} \int_V x dv \\ y_G & = & \frac{1}{V} \int_V y dv \\ z_G & = & \frac{1}{V} \int_V z dv \\ I_{xx} & = & \rho \int_V (y^2 + z^2) dv \\ I_{yy} & = & \rho \int_V (x^2 + z^2) dv \\ I_{zz} & = & \rho \int_V (x^2 + y^2) dv \\ I_{xy} & = & I_{yx} = -\rho \int_V xy dv \\ I_{yz} & = & I_{zy} = -\rho \int_V yz dv \\ I_{xz} & = & I_{zx} = -\rho \int_V xz dv \end{array}$$

Le problème se ramène donc au calcul de :

$$\begin{array}{ccc} \int_V x dv & \int_V y dv & \int_V z dv \\ \int_V x^2 dv & \int_V y^2 dv & \int_V z^2 dv \\ \int_V xy dv & \int_V yz dv & \int_V xz dv \end{array}$$

Ou, plus généralement, au calcul d'intégrales de type $\int_V f(x, y, z) dv$.

[5]Le théorème de Shannon indique qu'une subdivision au moins deux fois plus fine que le plus petit détail à restituer donnerait de bons résultats, mais cela n'a guère de sens dans le cas qui nous intéresse.

Integration sur un Tétraèdre Générique Dans le cas particulier où trois côtés d'un tétraèdre forment une base orthogonale, le calcul des intégrales est aisément effectué en les décomposant le long des axes de cette base. Cette méthode ne peut pas être appliquée dans le cas général ; nous utiliserons donc une propriété de l'intégration spécifiant que si $g = (g_1, g_2, g_3)$ est un difféomorphisme associant à chaque point p d'un volume V un point $g(p) = (g_1(p_x), g_2(p_y), g_3(p_z)$ de V', alors on a :

$$\int_{V'} f(x,y,z)dv = \int_V f \circ g(u,v,w).\left|J\right|.du.dv.dw$$

Où $J = \frac{D(g_1,g_2,g_3)}{D(u,v,w)}$ est le jacobien de g et M_J sa matrice jacobienne :

$$M_J = \begin{pmatrix} \frac{\delta g_1}{\delta u} & \frac{\delta g_1}{\delta v} & \frac{\delta g_1}{\delta w} \\ \frac{\delta g_2}{\delta u} & \frac{\delta g_2}{\delta v} & \frac{\delta g_2}{\delta w} \\ \frac{\delta g_3}{\delta u} & \frac{\delta g_3}{\delta v} & \frac{\delta g_3}{\delta w} \end{pmatrix}$$

Il peut être montré qu'une matrice M_{T_0T} associant de manière bijective à tout point du tétraèdre T_0 défini par $(O, \vec{i}, \vec{j}, \vec{k})$ un point du tétraèdre T défini par ses quatre sommets (S_0, S_1, S_2, S_3) existe toujours. Le calcul matriciel montre que si T n'est pas dégénéré (donc s'il est de volume non nul), on a :

$$M_{T_0T} = \begin{pmatrix} S_1x - S_0x & S_2x - S_0x & S_3x - S_0x & S_0x \\ S_1y - S_0y & S_2y - S_0y & S_3y - S_0y & S_0y \\ S_1z - S_0z & S_2z - S_0z & S_3z - S_0z & S_0z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Ainsi, g peut être exprimé sous forme d'un produit matriciel avec M_{T_0T}, et on montre alors facilement que $|M_J| = |M_{T_0T}|$. Sachant que M_{T_0T} est inversible par définition, $|M_J|$ ne peut pas être nul, donc g est un difféomorphisme.

En utilisant la propriété associée, on obtient :

$$\int_T f(x,y,z)dv = |M_{T_0T}| \int_{T_0} f(M_{T_0T}P)dv$$

Ce résultat permet d'exprimer toutes les intégrales précédentes comme :

$$|M_{T_0T}| \int_{T_0}(c_0.x^2 + c_1.y^2 + c_2.z^2 + c_3.x.y + c_4.x.z + c_5.y.z \\ +c_6.x + c_7.y + c_8.z + c_9).dv = \int_{T_0} h(x,y,z).dv$$

Où c_i est un ensemble de coefficients dépendant de m_{ij}. Via des intégrales itérées et des considérations géométriques, on montre finalement que :

$$\int_{T_0} h(x,y,z).dv = \frac{c_0 + c_1 + c_2}{60} + \frac{c_3 + c_4 + c_5}{120} + \frac{c_6 + c_7 + c_8}{24} + \frac{c_9}{6}$$

L'ensemble de ces raisonnements, ainsi que l'expression des paramètres c_i, est détaillé en annexe C, (section C.1).

Intégration dans VClip : Comparaison

Lors de notre étude des possibilités d'extension du simulateur physique ODE (section 3.4.3), nous nous sommes intéressés à VClip, un outil de détection des intersections entre maillages développé par Brian Mirtich [74]. Or cet outil s'avère comprendre une fonction de calcul des paramètres physiques de ces maillages, elle-même basée sur un article du même auteur [73].

L'algorithme utilisé se base sur les mêmes hypothèses que le nôtre : l'objet à modéliser est représenté par une union disjointe de polyèdres de densité constante. Les quantités à calculer sont les mêmes, et passent aussi par la réduction du problème au calcul d'un ensemble d'intégrales simplifiées. Cependant, l'approche utilisée pour leur calcul diffère : l'auteur se base sur le *théorème de divergence* pour réduire chacune de ces intégrales volumiques en une somme d'intégrales sur les faces du polyèdre. Ces intégrales surfaciques sont elles-mêmes évaluées via une projection planaire de la surface du polyèdre, celles-ci pouvant alors se ramener à une somme d'intégrales sur les côtés de ses faces (*théorème de Green*).

Ce travail présente des intérêts pratiques importants. Il est d'une part conçu de manière à minimiser autant que possible les erreurs de calculs (stabilité numérique, importante en informatique graphique). Et, d'autre part, il est efficace : un parcours des côtés suffit, d'où une complexité linéaire, et des factorisations bien choisies permettent de limiter le nombre de calculs flottants.

Nous avons naturellement comparé cette méthode d'intégration à celle que nous avons développée, en terme de précision et de rapidité de calcul. Pour cela, nous nous sommes basés sur deux volumes dont les intégrales volumiques sont connues : une sphère et un cylindre. Ils présentent l'avantage de ne pas pouvoir être représentés exactement par un maillage ; il est donc possible de faire varier le nombre de triangles utilisés pour les approximer afin de varier les conditions d'utilisation des algorithmes.

Par construction, notre algorithme d'intégration par tétraèdres et celui de Mirtich donnent des résultats aussi exacts que possible, contrairement à l'algorithme d'intégration par boîtes "naïf" initialement testé. Expérimentalement, ce dernier donne des résultats d'une précision comparable aux deux autres lorsqu'un seuil de tolérance de 10^{-4} lui est imposé.

Ces trois algorithmes sont comparés dans les tables 3.2 et 3.1, où `Box` indique la méthode d'intégration par raffinement successif d'une approximation par boîtes, `Tetr` notre algorithme d'intégration par tétraèdres et `VClip` la méthode proposée par Mirtich.

Volume	Cylindre		Sphère	
Triangles	60	600	1520	159200
Box	0.0134553	1.71523e-06	0.0024689	6.74735e-08
Tetr	0.0134859	1.4177e-06	0.00244059	2.48023e-07
VClip	0.0134859	1.4177e-06	0.00244059	2.48023e-07

TAB. 3.1 – Précision de Calcul des Intégrateurs

La table 3.2 indique leurs durées d'exécution ; il s'agit à titre indicatif de moyennes de mesures en millisecondes. La table 3.1 contient la précision des résultats calculés. Celle-ci est évaluée via l'erreur quadratique entre la matrice d'inertie calculée et sa valeur théorique. Notons que ces algorithmes travaillant sur des approximations polygonales des solides considérés, l'existence d'une erreur résiduelle (bien que diminuant avec la précision de la représentation) est normale.

Volume	Cylindre		Sphère	
Triangles	60	600	1520	159200
Box	34302	678710	82809	112018
Tetr	21	72	123	17260
VClip	5	204	45	8956

TAB. 3.2 – Durée de Calcul des Intégrateurs

Les résultats en terme de précision sont très satisfaisants, en particulier la similarité entre les erreurs effectuées par l'algorithme à tétraèdres et celui de Mirtich. Cette similarité est confirmée par l'évaluation de l'erreur quadratique entre les résultats de ces deux algorithmes : sur l'ensemble des mesures effectuées, celle-ci ne dépasse pas les 10^{-27}, soit en pratique une égalité des paramètres physiques calculés.

Les temps de calcul s'avèrent relativement prévisibles. L'algorithme basé sur un raffinement successif de maillage nécessite, cela avait déjà été constaté, un temps de convergence important. Cela est confirmé par ces mesures, qui montrent bien que cette méthode est inutilisable en pratique, y compris dans les cas les plus simples.

La comparaison entre les deux autres algorithmes met en avant l'intérêt des optimisations développées par Mirtich : de manière générale, son implémentation est entre deux et quatre fois plus rapide que la nôtre. Notons que, compte-tenu de l'identité des résultats obtenus, cet écart n'est pas si important si l'on considère que seule la qualité des calculs a été recherchée dans le développement de notre algorithme. Par ailleurs, deux points mitigent de manière intéressante ces résultats :

- L'implémentation de Mirtich effectue l'ensemble de ses calculs en mémoire, alors que la nôtre invoque un programme externe pour effectuer la décomposition en tétraèdres du maillage à intégrer. De plus, la communication avec ce programme se fait par l'intermédiaire de fichiers de données. On peut donc s'attendre à un impact important sur les temps d'exécution si une bibliothèque de décomposition est directement exploitée par notre implémentation.
- Dans des cas marginaux mais reproductibles, notre algorithme est presque trois fois plus rapide que celui de Mirtich. On peut supposer que ce cas particulier (cylindre à faible résolution) est particulièrement aisé à convertir en tétraèdres via un algorithme de Delaunay.

Globalement, l'algorithme de Mirtich est donc à favoriser par rapport au nôtre, en particulier de part les garanties qu'il fournit en terme de stabilité numérique. Cependant, l'approche par intégration sur tétraèdres semble encore permettre des développements intéressants. Rappelons enfin que les calculs de paramètres physiques étant effectués avant les simulations à proprement parler, leur précision est plus importante que leur vitesse d'exécution.

3.5.4 Extraction des Liens

Une fois calculée l'approximation par un corps rigide du volume du modèle influencé par un os, la dernière étape nécessaire à la génération de l'ARB correspondant est leur interconnexion. La conversion entre les os d'animation et les connexions physiques pourrait être une simple bijection, mais ce n'est pas systématiquement le cas.

En théorie, un os donné dispose de six degrés de liberté ; en pratique, la composante positionnelle de l'os est généralement fixée (cf. section 3.2.1), il ne dispose donc au plus que de trois degrés de liberté rotationnels. Ceci peut être modélisé par une articulation de type *ball and socket* classique, mais même si ce type de lien était disponible dans un outil de simulation donné, il n'est pas nécessairement désirable de l'utiliser pour des raisons de contraintes articulaires exposées ultérieurement (cf. section 4.2.1).

Tout d'abord, il est important de remarquer que beaucoup de simulateurs efficaces d'ARB représentent leurs modèles en utilisant les conventions issues de la robotique Denavit-Hartenberg ou Denavit-Hartenberg Modifiée (MDH). Ces systèmes de coordonnées n'associent qu'un degré de liberté (DoF) à chaque lien, et définissent leur pose en utilisant un ensemble spécifique de quatre paramètres (figure 3.15).

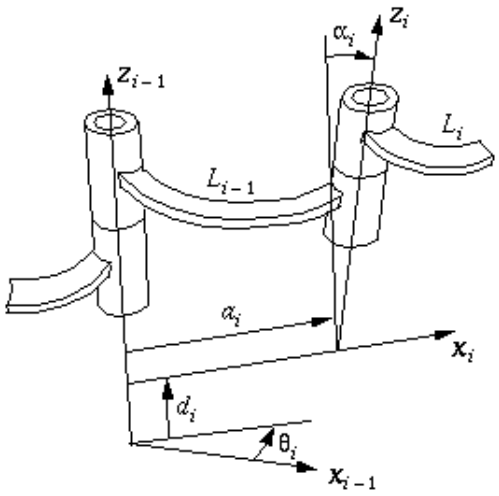


FIG. 3.15 – Paramètres de Denavit-Hartenberg

Ceci mène à des algorithmes de simulation très efficaces, mais au détriment de la possibilité de représenter des liens arbitraires. Il est possible de représenter ces derniers en combinant plusieurs liens à un degré de liberté. Ainsi, une méthode permettant de générer les paramètres MDH pour trois liens à un degré de liberté afin qu'ils puissent représenter le lien à trois degrés de liberté peut s'avérer utile pour exploiter certains simulateurs. Nous allons fournir ici un ensemble de paramètres ayant cette propriété.

Mais tout d'abord, passons en revue les liens élémentaires que nous allons considérer.

Liens à Un Degré de Liberté :

Il s'agit de liens prismatiques (translationnels) ou rotationnels. Dans le cadre des MDH, ils sont définis par deux paramètres de distance a et d et deux angles α et θ. d est variable pour un lien translationnel ; θ l'est dans le cas d'un rotationnel. Dans tous les cas, la transformation associée peut être exprimée comme :

$$T = R_x(\alpha).T_{xz}(a,d).R_z(\theta) = \begin{pmatrix} \cos\theta & \sin\theta & 0 & a \\ -\cos\alpha.\sin\theta & \cos\alpha.\cos\theta & \sin\alpha & d.\sin\alpha \\ \sin\alpha.\sin\theta & -\cos\theta.\sin\alpha & \cos\alpha & d.\cos\alpha \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

En pratique, la composition de trois liens rotationnels n'est pas suffisante pour exprimer des poses articulaires dans toute leur généralité, puisque cette combinaison mène à une valeur constante de la composante y. Pour y remédier, un paramètre translationnel est nécessaire.

Liens de type *Z Screw*

Ces liens n'ont pas de degré de liberté (tous leurs paramètres sont des constantes par rapport au temps) ; ils sont utilisés pour résoudre des problèmes positionnels tel celui décrit ci-dessus. Ces liens sont spécifiés via deux paramètres, une distance d et un angle θ. La transformation associée est :

$$T = T_z(d).R_z(\theta) = \begin{pmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & d \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Composition retenue :

Pour les raisons exposées précédemment, nous utiliserons un lien sans degré de liberté que sera le parent de trois liens successifs à un degré de liberté rotationnel. Leurs paramètres associés seront numérotés de 0 à 3, 0 étant le lien sans degré de liberté et 3 le lien à un degré de liberté feuille. Les paramètres variables seront donc θ_1, θ_2 et θ_3 ; les autres paramètres sont des constantes devant être calculées. Les raisonnements intermédiaires sont détaillés en section C.2 ; indiquons simplement ici que l'ensemble de valeurs finalement obtenu est :

$$\begin{cases} a_2 = d_2 = a_3 = d_3 = 0 \\ \theta_0 = 0 \\ \alpha_1 = -\pi/2 \\ x = a_1 \\ y = d_1 \\ z = d_0 \\ \alpha_2 = \alpha_3 = \pi/2 \end{cases}$$

Leur principal intérêt est qu'elles permettent l'utilisation de fonctions simples pour le calcul de $(\theta_1, \theta_2, \theta_3)$. En effet, si $q = (x, y, z, w)$ est le quaternion représentant la composante rotationnelle que nous souhaitons exprimer et si nous restreignons θ_2 à l'intervalle $[0, \pi]$, nous obtenons :

$$\begin{array}{rcl} \cos\theta_1 & = & c/\sqrt{1-f^2} \\ \sin\theta_1 & = & -i/\sqrt{1-f^2} \\ \theta_2 & = & \cos^{-1}(-f) \\ \cos\theta_3 & = & d/\sqrt{1-f^2} \\ \sin\theta_3 & = & -e/\sqrt{1-f^2} \end{array} \quad \text{avec} \quad \begin{array}{rcl} c & = & 2xz + 2wy \\ d & = & 2xy + 2wz \\ e & = & 1 - 2x^2 - 2z^2 \\ f & = & 2yz - 2wx \\ g & = & 2xz - 2wy \\ h & = & 2yz + 2wx \\ i & = & 1 - 2x^2 - 2y^2 \end{array}$$

Si $|f| = 1$, nous atteignons une singularité de type *gimbal lock*. Dans ce cas, une infinité de solutions existe (deux des paramètres sont liés) : l'une d'entre elles est :

$$\begin{cases} \theta_1 = 0 \\ \theta_2 = \pi \text{ si } f = 1, 0 \text{ sinon} \\ \theta_3 = atan2(g, h) \end{cases}$$

Utilisation de ce Modèle :

Il est maintenant aisé de construire l'ensemble de liens ARB représentant un os donné : il suffit pour cela de lier quatre d'entre eux ainsi que nous l'avons décrit et de les initialiser avec leurs paramètres MDH tels que calculés ci-dessus. Reste un problème : nous ne disposons que d'un corps rigide associé à l'os d'origine, alors que chacun des quatre liens en nécessite un. Ceci est aisément résolu en allouant le corps rigide au lien le plus profond : ainsi, le corps aura autant de degrés de liberté et la même position que l'os original. Les trois autres liens sont quant à eux initialisés avec les paramètres d'une primitive simple, telle qu'une sphère, choisie pour représenter une masse négligeable par rapport au corps rigide lui-même. Cette technique est souvent utilisée dans les simulateurs physiques, et les modifications résultantes du comportement physique global s'avèrent négligeables du fait du rapport entre les masses.

Dans le cas de ODE, des articulations à trois degrés de liberté sont directement disponibles. Cependant, les contraintes articulaires ne sont spécifiables que selon des angles d'Euler préétablis, d'où la nécessité de recourir à une transformation telle que celle décrite ci-dessus pour

pouvoir calculer les limites de ces angles. Notons que ce modèle de contraintes est très frustre dans le cas général, du fait en particulier des effets de *gimbal lock* inévitables avec une spécification de rotations via des angles d'Euler. Des méthodes permettant des contraintes plus élaborées seront proposées en section 4.2.1, dans le chapitre suivant.

3.5.5 Exportation du Modèle

L'ensemble de cette exportation une fois réalisée, Une instance de `PhysicalModel` est construite : il s'agit d'une classe basée sur un arbre de `PhysicalNode` permettant de stocker l'ensemble des données générées par les algorithmes précédents (*manifolds* clos, paramètres physiques et articulations).

Cet objet est finalement transcrit dans un fichier XML, qui présente l'intérêt d'une facilité de manipulation de par son format standardisé et d'une certaine robustesse due à une grammaire bien spécifiée. Le DTD correspondant et un exemple de fichier de ce type se trouvent en annexe D.

3.6 Visualisation

Ainsi qu'il a été précisé en section 3.1.3, la visualisation des résultats obtenus ne nécessite pas une grande qualité de rendu. Ils sont donc affichables interactivement via une bibliothèque de fonctions 3D (OpenGL dans notre implémentation, pour des raisons de portabilité) et le code de rendu associé au SDK décrit en section 3.2.1. Notons que, du fait de la rapidité du simulateur retenu, une visualisation en temps réel des animations physiques est possible, bien que le coût en calcul reste important.

L'exportation d'animations à fins de rendu n'a pas été implémentée ; elle nécessite simplement le choix d'un modèle de description de scènes suffisamment riche pour pouvoir exprimer l'animation par *keyframes* appliquée à un mécanisme d'animation par squelette. Mais dans un cadre expérimental, la visualisation directe reste la plus utile dans le sens où elle permet d'observer rapidement, voire interactivement le résultat d'une simulation.

3.7 Bilan

Nous avons décrit un cadre de travail permettant à un animateur d'obtenir des mouvements physiques en se basant sur une approximation physique polyarticulée d'un modèle continu animé par squelette issu de l'informatique graphique. Ceci est effectué de manière aisée puisqu'automatique ; cependant, ce résultat en lui-même se s'avère pas nécessairement suffisant.

En effet, les mouvements sont maintenant réalistes, mais difficiles à spécifier : l'animateur doit maintenant raisonner en termes de forces injectées à son modèles. Cette conception des choses est intuitive dans certaines situations (modélisation de collisions, simulations de type poupées de son - *ragdolls* -), mais dans la plupart des situations nous retrouverons le même problème que celui que nous avons résolu dans le cas de la conversion de modèle : des compétences relativement peu répandues du point de vue de l'informatique graphique sont requises pour pouvoir tirer le meilleur parti de cet outil.

Ceci mène à l'idée qu'un mécanisme de contrôle devrait être ajouté au sommet de la simulation physique : en utilisant son état courant et un objectif, il calculerait les forces nécessaires afin d'obtenir les mouvements désirés. Des contrôleurs en boucle fermée de ce type ont déjà été décrits dans les chapitres précédents.

Nous décrirons dans le chapitre suivant les utilisations possibles de cette plate-forme, ainsi que ses possibilités d'extensions.

Chapitre 4

Perspectives

4.1 Introduction

À la lumière des éléments bibliographiques proposés dans les chapitres 1 et 2, nous allons proposer un certain nombre d'extensions et d'applications possibles de la plate-forme décrite au chapitre précédent. Certaines d'entre elles peuvent être développées à court terme, alors que d'autres imposent un travail d'étude plus important ; nous tenterons autant que possible de préciser cet aspect au fur et à mesure de notre présentation.

4.2 Extensions de la Plate-Forme

Une première direction de recherche consiste en un dépassement des limitations actuelles de la plate-forme de simulation à proprement parler, ainsi qu'elles ont été identifiées durant le chapitre précédent.

4.2.1 Contraintes Articulaires

Une articulation réelle utilise rarement l'ensemble de ses degrés de liberté ; elle est le plus souvent restreinte à une partie de leurs intervalles. La prise en compte de ce type de contraintes n'est pas aussi simple dans un simulateur physique que, par exemple, dans un système à points clefs (*keyframes*), du fait que les mouvements des articulations sont une conséquence de la simulation, et non un paramètre directement manipulable. Les contraintes sur les liens doivent donc être modélisées comme des forces s'y appliquant lorsqu'elles s'approchent de leurs limites rotationnelles.

Le modèle effectivement utilisé pour définir ces forces de retour est spécifique à chaque simulateur, pour autant qu'il les implémente. Mais si c'est le cas, il est intéressant de tenter de calculer des angles limites pour nos liens. Par exemple, lorsque des données enregistrées sont disponibles (par exemple produites via de la capture de mouvements ou des points clefs), elles peuvent être converties vers des paramètres de liens (un ensemble par articulation et par point-clef). Ensuite, en prenant le minimum et le maximum des données générées pour chaque degré de liberté du modèle, des limites grossières peuvent être calculées. Ce type de technique

est applicable à un modèle de décomposition des articulations via des composantes à un degré de liberté, tel que décrit en section 3.5.4.

En pratique, les essais que nous avons pu mener à partir de cette approche s'avèrent peu concluants : du fait de la généricité des repères retenus pour la représentation des degrés de liberté angulaires (MDH ou Euler[1] le plus souvent), les intervalles angulaires sont généralement trop importants. Des discontinuités propres à certains modèles peuvent encore compliquer ce problème : dans le cadre des angles d'Euler par exemple, l'angle central balaye ainsi souvent l'ensemble de son intervalle théorique (180 degrés).

Les deux approches, plus génériques, que nous proposons dans les sections suivantes permettraient d'obtenir de meilleurs résultats.

Optimisation des Bases

L'étude d'échantillons de mouvements disponibles est envisageable, ceci afin de trouver dans le cadre d'une articulation décomposée en trois articulations à un degré de liberté les bases privilégiées permettant une représentation optimale des mouvements possibles (intervalles minimums balayés par chaque articulation). Précisons ces notions.

Soit qc_i le quaternion représentant une orientation de l'articulation composite et $qe_i = (\omega_j, \vec{v}_j)$ l'orientation d'une articulation élémentaire ($\vec{v}_j$ étant son axe de rotation et ω_j son angle de rotation autour de cet axe), on peut poser :

$$qc_i = qe_1 \cdot qe_2 \cdot qe_3$$

Soient $\vec{v}_1$, $\vec{v}_2$ et $\vec{v}_3$ fixés : on admet alors l'existence de fonctions f_j telles que pour tout qc_i, $f_j(qc_i) = \omega_j$. si i parcourt l'ensemble des échantillons d'orientations disponibles, trois intervalles $[min_j, max_j]$ peuvent être définis à partir des valeurs minimales et maximales calculées pour chaque ω_j et chaque pose.

Le problème initial se pose alors comme suit : pour un ensemble de qc_i, on cherche $\vec{v}_1$, $\vec{v}_2$ et $\vec{v}_3$ tels qu'une fonction similaire à :

$$s = \prod_{j=1}^{3} (max_j - min_j)^2$$

(erreur quadratique) soit minimisée.

Une fois un résultat de ce type obtenu, une optimisation supplémentaire consisterait en une limitation du nombre de degrés de liberté de l'articulation composite : il est en effet probable que, si les mouvements de celle-ci sont restreints à un ou deux degrés de liberté, une partie des intervalles de mouvement des articulations élémentaires soit restreinte à une pose fixe (aux erreurs d'échantillonnage près).

Le problème de cette méthode est que son traitement mathématique est complexe, puisqu'il est sous-contraint. De plus, il est d'une nature non linéaire de par les interactions entre les rotations élémentaires. Ces difficultés peuvent être contournées en se basant sur la fonction

[1] cf. annexe B

d'erreur décrite ci-dessus pour tenter une résolution par optimisation. Il serait aisé par exemple d'encoder les trois vecteurs recherchés (neufs réels) dans le cadre d'un algorithme génétique, et d'utiliser la fonction d'erreur pour les évaluer.

Mais pour obtenir un algorithme robuste, il reste malgré tout à valider mathématiquement un certain nombre de points, et en particulier le fait que la décomposition de toute orientation en trois quaternions soit toujours possible (existence des fonctions f_j) : si le théorème de rotation d'Euler[2] nous assure de l'existence d'une solution pour une orientation et des axes adéquats, rien n'indique que des axes fixés permettent cette représentation pour toute famille d'orientations, même représentant des mouvements réalistes.

Approche par Modèle de Contraintes

La formulation par décomposition en rotations élémentaires s'avérant problématique, une deuxième approche consisterait en une approche de plus haut niveau du problème : un modèle générique des contraintes à imposer à l'articulation peut être généré, puis utilisé pour calculer des forces de retour sur une articulation à trois degrés de liberté générique en fonction de sa position actuelle.

Un exemple de modèle de ce type est fourni par L. Herda *et al* dans un article publié en 2002 [58]. A partir d'un ensemble d'orientations valides pour l'articulation, une surface implicite sur la sphère est générée. Par interpolation, celle-ci est ensuite utilisée pour calculer les forces de retour à appliquer en fonction de la position de l'orientation courante sur la surface implicite générée. Cette approche présente par ailleurs l'avantage de prendre en compte le fait que les contraintes effectives dans les différentes directions ne sont pas indépendantes pour des articulations d'êtres vivants [8].

Implémentation de ces Méthodes

La première méthode implique l'utilisation d'une articulation composite conçue à partir de trois articulations à un degré de liberté rotationnel. Ce modèle est disponible dans les deux simulateurs physiques que nous avons rencontrés ; se pose cependant, au moins pour Dynamechs, un problème dû au fait que les articulations ne sont pas paramétrables de manière générique, mais uniquement en fonction des paramètres de Denavitt-Hartenberg. Ainsi, des articulations sans degré de liberté devront le plus souvent être insérées pour un positionnement correct des axes de rotation élémentaires, ce qui compliquera encore l'articulation composite.

ODE est de ce point de vue favorisé car il dispose d'une interface spécifique, le AMotor, permettant une fois associée à une articulation à trois degrés de liberté de lui imposer des contraintes. Les deux classes effectivement implémentées permettent l'utilisation de contraintes autour des angles d'Euler ou d'axe générique (ainsi, la première méthode proposée pourrait s'utiliser efficacement avec ce mécanisme, en "virtualisant" les trois sous-articulations). Il est possible d'implémenter d'autres types de contraintes, mais le formalisme mathématique est très spécifiquement lié au modèle de simulation sous-jacent, ce qui le rend délicat à aborder.

[2] "Une rotation arbitraire peut être décrite par seulement trois paramètres" (issu de `http://mathworld.wolfram.com`)

A contrario, la seconde méthode proposée utilise un modèle extérieur à toute représentation de l'articulation ce qui permet son adaptation à toute simulation fournissant des articulations à trois degrés de liberté rotationnels, même si celles-ci n'intègrent aucun système de contraintes propres. Le seul élément nécessaire à l'interfaçage sera la possibilité d'injecter dynamiquement des forces de retour au niveau de l'articulation, possibilité que l'on peut attendre de tout simulateur physique.

Discussion

Que ce soit d'un point de vue théorique ou en terme d'implémentation, nous avons vu que l'approche par modèle de contraintes de plus haut niveau que l'analyse purement articulaire s'avère probablement plus prometteuse car plus générale ; c'est dans cette direction que nos efforts vont s'orienter. Notons que cette adaptation n'est cependant pas directe. En effet, l'approche décrite dans la section 4.2.1 est purement géométrique : à partir d'une position actuelle invalide, la position valide la plus proche est retournée par l'algorithme.

Or le calcul de forces de retour à partir de ce vecteur mouvement désiré pose le problème de son calibrage précis : si elle est trop faible, la compensation sera trop lente alors que dans le cas contraire, des oscillations risquent de se développer.

Il s'agit dans le cas général d'un problème de contrôle classique (retour d'un système dans une configuration optimale le plus rapidement possible). Des formalismes tel le contrôle PID (*Proportional, Integral, Derivative*) permettent traditionnellement de résoudre ce problème de contrôle. Une référence classique dans ce domaine est l'ouvrage[3] de Doyle *et al* [35] ; un bilan sur la génération automatique de contrôleurs PID est par ailleurs proposé par K. J. Astrom *et al* dans [7].

Compte-tenu du fait que le système à contrôler est relativement simple et devrait peu s'éloigner des positions valides s'il est régulièrement contrôlé, nous envisageons dans un premier temps d'appliquer des techniques de régulation de type amortisseur-ressort telles celles décrites en section 4.3.1. Si le résultat s'avère instable, des outils tels que ceux décrits ci-dessus pourront être utilisés pour un meilleur interfaçage des contraintes géométriques et du système physique sous-jacent.

4.2.2 Simulation des Collisions

Ainsi que nous l'avons décrit en section 3.4.3, l'ajout de maillages dans ODE implique, pour une représentation efficace des mouvements, une prise en compte des interactions entre ceux-ci et non pas seulement la prise en compte de collisions avec des solides simples telle que nous l'avons implémentée.

Or, dans le cas général, et même en se restreignant à des solides convexes, le fait de disposer de bibliothèques détectant efficacement les collisions ne suffit pas. Le problème se pose en effet de déduire les caractéristiques précises des forces de contact mises en jeu à partir d'informations purement géométriques sur une zone de collision (on se référera à la section A.3.8 pour une description des bases physiques de ce processus).

[3] Maintenant disponible en ligne : `http://www.control.utoronto.ca/people/profs/francis/dft.pdf`

Nous allons proposer deux approches, par sphères hiérarchiques et par modélisation de contact, permettant de prendre en compte ce problème.

Modèle par Sphères Hiérarchiques

Il est possible dans un premier temps de tenter de tirer parti d'algorithmes bien maîtrisés et des primitives de collision déjà disponibles dans ODE pour obtenir un mécanisme approximant les résultats recherchés.

L'un des candidats à cette utilisation est la sphère : les collisions avec ce type de primitive sont très efficacement calculées, et ces réactions sont tout à fait réalistes dans les simulations existantes. Par ailleurs, il s'agit d'une primitive géométrique facile à manipuler et à combiner (union) dans ODE.

Une possibilité consisterait alors en la génération d'une approximation du maillage sous forme d'un ensemble de sphères ; les structures de ce type sont qualifiées de *sphere trees*. Des algorithmes pour les générer automatiquement à partir d'un volume ont été proposés initialement par J. O'Rourke et N. Badler en 1979 [79], et leur utilisation en détection de collisions a été proposée par I. J. Palmer et R. L. Grimsdale en 1995 [81].

Un algorithme moderne de construction de ce type de structure est proposé dans un article publié en 2002 par G. Bradshaw et C. O'Sullivan [17].

Une fois un algorithme de ce type utilisé pour générer l'arbre de sphères, son importation dans ODE ne pose pas de difficultés particulières (notons que les collisions et la simulation physique étant gérées via des objets différents, cela n'aura pas d'impact sur les matrices d'inerties et autres paramètres physiques utilisés). Il faudra par contre déterminer un compromis raisonnable pour le nombre de sphères utilisées : l'augmenter a un impact évident sur la précision des collisions modélisées, mais au détriment de la vitesse de simulation, qui est largement fonction du nombre de forces (donc de collisions élémentaires) à prendre en compte.

Meilleure Approximation

En parallèle aux approximations précédentes, un certain nombre de travaux visent à une simulation efficace des collisions entre maillages, et plus précisément des phénomènes ayant lieu au niveau de leur zone de contact (*contact modelling*) ; citons par exemple l'article publié par Y. Kim *et al* en 2002 [115]. Insistons sur le fait que ce domaine de recherche s'interface étroitement avec la détection d'intersections, plus géométrique, mais ne doit pas être confondu avec celle-ci (qui ne se soucie que de détecter et éventuellement caractériser les zones de contact entre maillages, indépendamment de toute considération physique). Une compilation des travaux à ce propos est par exemple proposée par Carol O'Sullivan *et al* dans [80] (non publié).

Des travaux particulièrement prometteurs pour cela semblent être ceux de G. van den Bergen [44] (2001, non publié), basés sur l'algorithme GJK proposé par E. Gilbert *et al* en 1988 [51]. Ils mettent en effet l'accent sur le calcul de la profondeur de pénétration de deux objets, l'une des caractéristiques du contact rarement prise en compte par les outils de détection de collision et essentielle à ODE. Par ailleurs, ceci est effectué via un formalisme mathématique générique, ce qui permet d'espérer une adaptation à une représentation surfacique aussi

complexe que les maillages. Enfin, une implémentation logicielle (SOLID) de cet algorithme est disponible.

Il n'est pas évident que ces travaux puissent être adaptés directement : une simulation réaliste des collisions dans ODE nécessite la génération de plusieurs points de contact, or SOLID ne semble pouvoir fournir que le plus profond. Mais le formalisme mathématique utilisé permet d'envisager une généralisation de cet algorithme afin d'obtenir les données qui nous sont nécessaires.

Discussion

Pour évaluer les approches précédentes, il convient de distinguer deux situations. Un premier cas d'utilisation des collisions entre maillages est la représentation des auto-collisions au sein d'un modèle : elles ont pour but d'éviter toute interpénétration d'éléments du modèle. Notons qu'une grande précision à ce niveau n'est pas requise, pour au moins deux raisons :

- les contraintes articulaires, telles que proposés en section 4.2.1, permettraient dans une large part une limitation de poses irréalistes (articulations pliant à 180 degrés par exemple) aboutissant à des positionnements pathologiques des parties du corps.
- Les inter-pénétrations, au-delà du fait qu'elles lui permettent de prendre des poses inhabituelles, sont internes au modèle et n'ont donc a priori pas d'influence notable sur ses interactions avec l'environnement et les autres modèles.

Ainsi, l'évitement d'inter-pénétrations s'avère finalement avoir surtout un intérêt visuel, dans le sens où il évite des situations irréalistes ; mais l'impact sur la qualité de la simulation d'une imprécision à ce niveau est probablement limité. Par contre, des interpénétrations de faible amplitude sont susceptibles d'avoir lieu fréquemment, entre pattes par exemple ; il est donc nécessaire de pouvoir en tenir compte efficacement.

Pour ces raisons, l'utilisation d'un modèle de collisions par sphères hiérarchisées est donc probablement un bon candidat à l'évitement des auto-collisions.

Par contre, tout l'intérêt de la simulation physique est dans une simulation réaliste des collisions entre les modèles et leur environnement ; en particulier, un modèle de collision et de friction précis et réaliste est nécessaire à une génération satisfaisante de mouvements locomoteurs. Ainsi, une approche telle celle décrite section 4.2.2 s'avère requise, en particulier dans un cadre d'apprentissage automatique de la locomotion : les expérimentations de Sims décrite en section 1.3.6 ont montré les problèmes inhérents à une simulation physique trop approximée.

L'adaptation d'algorithmes de collision réalistes au moteur de simulation que nous avons retenu n'est probablement pas une tâche réalisable sur le court terme : cependant, la disponibilité de primitives de collisions efficaces et réalistes avec les plans et les sphères permettent déjà des expérimentations dans un environnement comportant quelques obstacles.

4.2.3 Simulations Distribuées

L'essentiel des ressources en calcul utilisées par un outil d'animation, tel que nous l'avons décrit, est consacré aux simulations physiques. Pouvoir accélérer cette simulation est intéressant pour au moins deux raisons :

– Il est ainsi possible de simuler des scènes plus complexes tout en conservant une vitesse d'animation raisonnable dans le cadre d'une approche interactive (conception d'une animation, jeux et réalités virtuelles).
– Dans le cadre d'une approche par optimisation (cf. section 4.3), la nécessité d'un grand nombre de simulations avant l'obtention d'un résultat probant rend leur rapidité cruciale.

De ce point de vue, un système distribué de simulation tel celui décrit par T. Funkhouser en 1995 (système client serveur RING [39]) ou par K. Perlin en 1996 (Improv [85], déjà décrit en section 1.6.3) serait intéressant pour plusieurs raisons.

Notons tout d'abord que le problème de la simulation d'un ARB donné n'est probablement pas évident à distribuer dynamiquement sans des remaniements profonds des algorithmes sous-jacents. Par contre, le fait de répartir les ARB et éléments de l'environnement sur différents points de calcul ne pose pas de difficultés particulières. Dans ce cas, le problème devient alors celui d'une affectation efficace des objets à simuler afin de répartir la charge de calcul, et surtout de limiter le trafic réseau imposé par la prise en compte des interactions entre des objets situés sur des sites distants (la gestion des collisions étant effectuée indépendamment de la simulation, cela peut être mis en place sans remaniement de ses algorithmes).

Ceci peut s'effectuer dynamiquement en affectant un coût aux communications réseau, et en permettant la migration (au sens où cela est employé pour les agents autonomes ou le code mobile) des objets simulés. Un tentative de minimisation de leurs couplages pourra exploiter cette possibilité de migration pour rapprocher sur un même serveur deux objets en interaction forte (l'utilisation d'un coût associé à la migration permettra d'éviter un regroupement de tous les objets sur un même simulateur). Il est remarquable qu'ici, des indices géométriques (distance entre objets, trajectoires) permettent de prévoir dans une certaine mesure les évolutions de charge : contrairement à d'autres problèmes de distribution, il est donc possible de les anticiper, et de tenter d'y pallier en migrant les objets avant leur collision effective.

Dans une approche écologique de la génération de contrôleurs (abordée en section 4.3), ce mécanisme de migration présente un autre intérêt : en associant des pénalités importantes aux migrations, et en fournissant un espace de simulation plus vaste, il serait possible de permettre une évolution en parallèle de créatures variées en laissant le temps à des solutions provisoirement moins efficaces de se développer. Ce type de séparation géographique améliorerait la diversité globale, donc éviterait une convergence trop rapide vers des solutions temporairement plus efficaces ; cette idée d'utilisation de *pools* temporairement isolés d'individus, classique en optimisation par algorithmes génétiques, est dans notre contexte justifiée par les expérimentations de J. Ngo dans ses travaux de 1993 sur les contraintes d'espace-temps [77], ou encore par les expérimentations développées en 1994 par C. Reynolds autour de l'apprentissage de stratégies permettant de remporter un jeu de poursuite [92],

4.3 Utilisations Envisagées

En parallèle à l'amélioration de la qualité de simulation proposée dans la section précédente, toute une série de développements peut être initiée par-dessus celle-ci. L'essentiel des

applications que nous allons proposer s'enchaînent séquentiellement, chacune tirant parti des possibilités offertes par la précédente.

4.3.1 Tenue de Pose ; Suivi de Consigne

Actuellement, aucune force autre que celles issues des collisions avec l'environnement n'est appliquée à nos modèles : ils chutent ainsi simplement sur le sol, sans contraintes particulières. Ainsi, la figure 4.1 montre un modèle comparable à une raie avant simulation, et la figure 4.2 le montre au repos, après stabilisation du fait des forces de frottement.

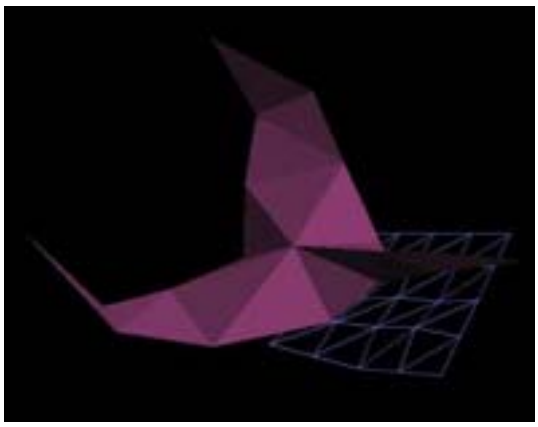


FIG. 4.1 – Modèle Physique Initial

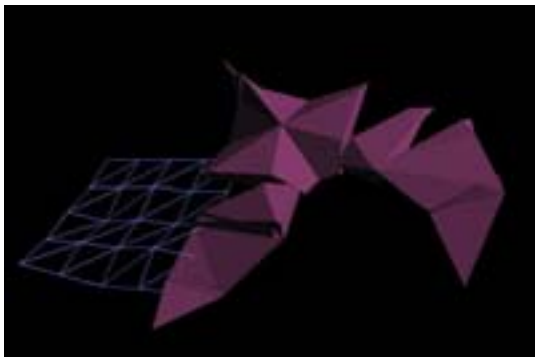


FIG. 4.2 – Modèle au Repos

Formulation du Problème

Nous aimerions générer un contrôleur qui, associé à ce modèle, serait capable de maintenir sa pose initiale via l'injection de couples contre-balançant les forces de collisions externes.

Ce problème est intéressant en soit, puisqu'il implique nécessairement un contrôleur en boucle fermée adapté aux spécificités du modèle simulé. C'est aussi une base utile à des tâches plus complexes telle que l'atteinte de poses [28, 46, 109], pouvant servir de base à des contrôleurs de locomotion.

Essentiellement, nous recherchons l'ensemble des fonction qui, pour l'état d'un lien et une pose attendue donnés, calcule le couple permettant de l'atteindre en un temps minimal.

Approche par Optimisation

Deux techniques sont utilisables pour la résolution de ce problème. Tout d'abord, un ensemble de fonctions paramétrées constituant des solutions potentielles à ce problème peut être généré, puis des outils d'optimisation peuvent être utilisés pour définir les valeurs convenables de leurs paramètres. Une autre approche consiste en la définition d'une famille générique de fonction, puis en l'utilisation d'outils de programmations génétiques pour explorer cet ensemble de fonctions et y trouver celles résolvant ce problème.

De meilleurs résultats peuvent potentiellement être attendus de la seconde méthode, du fait que moins d'hypothèses y sont faites sur la forme d'une solution potentielle (une plus grande variété d'entre elles peut donc être explorée). Cependant, cela implique un coût en calculs beaucoup plus important. Dans un premier temps, nous nous proposons donc de retenir un ensemble de fonctions connues comme étant une solution efficace à ce problème, et de n'utiliser des outils d'optimisation que pour les paramétrer correctement.

Modèle Choisi

Un outil de résolution classique de ce problème est le couplage "amortisseur-ressort", dans lequel la composante "ressort" produit une force ramenant la masse (ici le segment de membre considéré) vers sa position de repos alors que l'amortisseur dissipe l'énergie, évitant des oscillations indésirables. Il peut être exprimé de manière générique comme :

$$F = C.x' + K.x$$

F étant la force de retour (à appliquer au segment), C la constante d'amortissement, K la constante du ressort et x la position de la masse. Dans le cas *critique*, où $C = 2.\sqrt{KM}$ (M étant la masse du corps), cette configuration dissipe l'énergie et revient dans sa position de repos le plus efficacement. Notre problème est donc le calcul, pour chaque lien, des paramètres C et K optimaux ; le fait qu'ils puissent être calculés en utilisant la propriété précédente est une métrique utile pour évaluer l'efficacité de l'algorithme d'optimisation utilisé.

Nous prévoyons l'utilisation d'algorithmes génétiques (tels qu'initialement décrits par Holland [48] et Goldberg [45]) pour calculer ces paramètres ; un certain nombre de travaux déjà décrits dans le chapitre 1 montre leur capacité à générer des résultats significatifs lorsqu'ils sont utilisés pour le contrôle de modèles physiques [108, 99, 46].

Des algorithmes génétiques ajustant les paramètres de la solution sont bien connus dans ce cas (un ensemble de valeurs réelles), mais une fonction d'adéquation (*fitness*) permettant

d'évaluer l'efficacité d'un contrôleur donné est requise. Nous avons l'intention d'utiliser une fonction quadratique classique dans ce but :

$$f = \int_T \sum_{i=1}^{n} (p_i(t) - p_i(0)).dt$$

n étant le nombre de segments, et $p_i(t)$ la position du i^{eme} lien au temps t.

Discussion

En l'état, ce modèle permettrait déjà de tester les possibilités de stabilisation du modèle physique à travers un contrôleur minimal. Sa simplicité vient du fait qu'il ne prend pas en compte les degrés de liberté des articulations, puisqu'il ne permet qu'une seule pose.

Mais l'adjonction à cet outil de méthodes similaires à celles décrites en section 4.2.1 permettrait d'obtenir un véritable système de contrôle articulaire, permettant la prise en compte de contraintes. En n'appliquant plus le mécanisme de retour que lors de l'atteinte de positions invalides, il serait possible d'obtenir un modèle de type *ragdoll* (pas de tenue de posture) mais respectant des limites articulaires.

Des résultats plus élaborés peuvent être obtenus en combinant les idées de respect de contraintes articulaires et de pose par défaut. Soit une orientation articulaire (pose) q, considérons en effet l'algorithme suivant :

- Si l'articulation est dans une pose invalide, la ramener à sa plus proche pose valide (algorithme de la section 4.2.1).
- Sinon, la ramener à q.

Une orientation constante nous ramène au cas précédent. Par contre, faire varier q permet de considérer cette pose de référence comme une consigne qui sera respectée dans la limite des poses valides pour le modèle. Ainsi, on disposerait pour chaque articulation d'une forme de dynamique inverse économique (car elle n'est pas calculée sur une pose globale à respecter pour l'ensemble du modèle, mais localement à chaque articulation) permettant de le contrôler grâce à des informations purement géométriques.

4.3.2 Locomotion Élémentaire

En se basant sur les méthodes exposées précédemment, qui permettent une abstraction du contrôle moteur au niveau des articulations élémentaires, des méthodes telles celles décrites en section 1.3.6 peuvent être appliquées pour un apprentissage d'une locomotion élémentaire, en boucle ouverte. Le critère d'optimisation serait ici la distance parcourue par le modèle, indépendamment de toute consigne dynamique lors de la tâche.

Notons qu'il ne s'agit pas encore *stricto sensu* de générer un contrôleur, mais plutôt un moteur (au sens de la section 1.3.5) : il s'agit simplement d'enchaîner des mouvements, pour l'instant sans prise en compte de l'environnement (seules des informations proprioceptives sont exploitées).

Spécifier la tâche attendue et une méthode de résolution ne suffit pas : il faut de plus décider de l'outil utilisé pour implémenter le contrôleur à proprement parler. L'ensemble des mécanismes décrits en section 1.6.3 est exploitable, bien que certains soient plus adaptés aux méthodes par optimisation.

Sur la base des travaux décrits en section 1.3.6 et sur un certain nombre d'éléments d'éthologie (en particulier l'optimisation de l'innée, section E.2.7 et la mise en place des stéréotypes, sections E.6 et E.6.4), nous proposons de partir d'une série de "briques de comportement" relativement stéréotypées puis d'optimiser leur coordination pour la tâche requise.

Il est ainsi possible d'espérer des comportements plus souples du fait de l'aspect adaptatif (qui courra durant toute la durée de la simulation), mais aussi plus riche qu'une approche strictement exploratoire partant de rien (*from scratch*) du fait de briques initiales pouvant être riches (oscillateurs, intégrateurs, automates déterministes...).

Il est à noter que ces briques ne doivent pas requérir la circulation d'informations de trop haut niveau entre elles ; de manière générale, on peut souhaiter éviter la simulation de comportements trop cognitifs, d'où probablement un évitement de modèles humains au profit de comportements animaux (noter que le plus souvent, le symbolisme ne leur est pas nécessaire).

4.3.3 Locomotion en Boucle Fermée

Une fois un contrôleur rendu disponible par la méthode précédente, une locomotion en boucle fermée est abordable. Il s'agirait alors d'adapter les mouvements à l'environnement afin d'obtenir une locomotion plus robuste. Deux points peuvent être considérés :

- Prise en compte des collisions avec l'environnement : une collision avec un objet de petite dimension ne doit pas empêcher la locomotion, celui-ci doit simplement être évité (à la manière du *step reflex*) décrit en section 1.3.6.
- Suivi de consigne : la locomotion doit pouvoir s'effectuer dans une direction précise, afin de rejoindre un point particulier. Si cette consigne évolue dans le temps, il s'agit alors d'un suivi de trajectoire (*tracking*).

Cette idée d'apprentissage progressif de tâches de complexité croissante est inspirée des travaux de Terzopoulos en informatique graphique (section 1.3.6) et des techniques de *robot shaping* popularisées par Dorigo et Colombetti (section 2.5.1).

Notons qu'elle est la première à impliquer l'exploitation d'informations externes au modèle, et implique donc la mise en place d'un mécanisme de perception minimale (mise à disponibilité des informations issues des méthodes de détection de collision, et possibilité d'obtention de la position relative de la consigne). Par ailleurs, il devient intéressant à ce niveau de disposer d'un modèle de collision plus élaboré afin de pouvoir complexifier l'environnement, ce qui donne tout son intérêt à cette locomotion adaptative.

4.3.4 Coévolution : système Prédateurs / proies

Nous allons présenter ici un projet plus ambitieux, car poussant plus loin la métaphore biologique pour aborder à proprement parler les notions de comportement et d'interaction.

Évolution Naturelle et Artificielle

Permettons-nous une comparaison entre l'évolution biologique et les algorithmes génétiques, afin de mettre en perspective les propositions précédentes.

L'approche "algorithmes génétiques" est finalement assez contrainte dans son usage habituel, du fait de la nécessité d'une fonction d'évaluation. Celle-ci contient implicitement, sinon les moyens, du moins les fonctions du résultat (il existe d'ailleurs une idée de convergence). Par opposition, la vie n'a pas d'autre fonction que de se perpétuer, c'est ce qui la définit. Le génotype, se développant dans un environnement complexe, code alors pour l'ensemble de l'individu, y compris (contrairement à son équivalent artificiel) ses mécanismes reproductifs. Par ailleurs, la notion d'espèce et leur compétition en tant que phénotypes s'ajoute à la seule nécessité d'adaptation à un environnement. Notons d'ailleurs que les éléments pertinents de l'environnement (en terme de menaces, de potentiel de reproduction, de nutrition) sont en général eux-mêmes biologiques. Des concepts aussi élémentaires que la prédation ou la reproduction ont ainsi été "inventés" par l'évolution, et sont des phénomènes existant uniquement dans le champ de la biologie : rien dans l'environnement pré-biotique ne permettait de prévoir ce type d'interactions, issues de la coévolution des espèces.

Notons ainsi que par coévolution, les capacités de certaines espèces peuvent en modifier d'autres, à qui elles sont pourtant complètement étrangères. Ainsi, les phénomènes de coloration vive de proies impropres à la consommation et les mimétismes associés n'auraient aucun sens sans l'existence de capacités associatives chez leurs prédateurs : c'est donc ces capacités qui influencent leurs proies, alors que celles-ci en sont elle-même dépourvues.

Intérêt d'un Environnement Complexe

En première analyse, il est possible d'opposer les algorithmes génétiques, simples optimisateurs de fonctions posées par un utilisateur, et l'évolution biologique, créatrice de nouveauté. La distinction n'est cependant qu'apparente, cette notion de création étant probablement inappropriée. En effet, Eric Werner par exemple [112] soutient une position suivant laquelle les mécanismes d'optimisation (qu'ils soient naturels ou artificiels) ne créent pas de la complexité à proprement parler, mais l'extraient plutôt de l'environnement.

Ainsi, la différence entre l'évolution naturelle et artificielle ne tiendrait pas tant dans leurs mécanismes propres que dans la complexité des environnements dans lesquels elles travaillent. Cela va évidemment dans le sens de la nécessité d'un environnement riche pour l'obtention de comportements complexes. Mais on peut aussi en déduire l'intérêt de la présence simultanée dans l'environnement d'un certain nombre de créatures, non plus en compétition indépendamment les unes des autres pour un même optimum statiquement défini mais en interaction dans la recherche de cet optimum. Pour cela, la tâche à accomplir doit nécessairement inclure les autres créatures, celles-ci suivant leurs objectifs propres ayant alors intérêt à y collaborer ou au contraire à s'y opposer. Elles deviennent ainsi une composante à part entière de l'environnement, qui devient lui-même fluctuant et adaptatif. Cela rejoint les constatations de la section précédente, à savoir que l'évolution génère son propre cadre.

Projet

Pour l'obtention de comportements plus complexes, et pour montrer l'intérêt d'une approche par optimisation par rapport à des approches planificatrices plus classiques, un cadre intéressant serait un environnement suffisamment fluctuant pour que les approches trop basées sur la planification échouent (impossibilité à replanifier) ou soient trop lentes (nécessité de replanification constante) ; il faudrait cependant conserver la nécessité d'une organisation des comportements, afin qu'une approche purement réactive soit elle aussi peu efficace.

Un ensemble de facteurs permettant ceci serait un environnement à topologie complexe (difficultés de locomotion et de perception), la nécessité d'interagir avec d'autres individus aux comportements plus ou moins complexes et éventuellement l'ajout d'évènements stochastiques (nécessité d'opportunisme).

Un cadre de travail possédant ces caractéristiques nous paraît être la simulation d'un écosystème dans un environnement réaliste : la nécessité pour les individus de se nourrir les oblige à interagir entre eux et avec l'environnement, et l'utilisation d'une simulation physique rend les déplacements complexes.

▷ **Environnement** Il s'agirait d'une simulation physique simplifiée mais incluant une dynamique. Un sol plus ou moins accidenté et des plantes seront présents, d'où une pression forte sur les capacités locomotrices des créatures.

Les détails complexes tels la gestion des frottements, des collisions doivent être présents mais ne pas donner lieu à trop de coûts, l'objectif n'étant pas une crédibilité absolue mais une possibilité de déroulement rapide (pouvant être améliorée par le recours à des techniques telle que la distribution des simulations décrite en section 4.2.3).

▷ **Physiologie des Créatures** En parallèle aux capacités de locomotion déjà décrites en section 4.3.3, une physiologie minimale sera simulée, au moins via une variable d'énergie globale, et éventuellement d'autres pour la fatigue par exemple (inverse de l'énergie disponible immédiatement). Le métabolisme basal sera calculé en fonction de la masse (donc du volume) de l'animal, afin de ne pas favoriser nécessairement les grosses créatures. Tous ces paramètres varieront en fonction de l'ingestion de nourriture et de l'énergie dépensée pour la locomotion.

La nutrition sera possible via l'ingestion de plantes et d'autres créatures. Il serait intéressant de leur permettre d'être omnivores, mais d'optimiser les apports et dépenses énergétiques afin que les spécialisations soient rentables.

▷ **Perceptions et Comportements** Les perceptions ne doivent pas être trop symboliques, pour obtenir un certain réalisme ; mais des données purement quantitatives sont complexes à exploiter.

Nous proposons l'utilisation d'une vision synthétique de relativement bas niveau associée à un mécanisme de signature des éléments perçus, correspondant à la reconnaissance dans la nature de textures, colorations, vitesses, etc. d'où un vecteur perceptif de valeurs essentiellement numériques. Ces signatures seraient non arbitraires (contrairement à un système de perception symbolique), et caractéristiques de l'élément perçu. Pour éviter une perception absolue, il se-

rait intéressant que la précision de la signature dépende de la proximité de l'objet perçu (au plus proche, on dispose de modalités de perception tactiles ou chimiques supplémentaires).

La créature doit, pour être efficace, apprendre à donner un sens à ces signatures (encore une fois, non significatives en elles-mêmes) en les associant à des comportements adaptés. Cette approche est intéressante car il n'est pas nécessaire de gérer des problèmes de reconnaissance de bas niveau, et chaque créature peut interpréter différemment une signature en fonction de ses expériences, de son métabolisme... Il n'y a donc effectivement plus de signes dans l'environnement, contrairement à une perception informée.

▷ **Apprentissage** Plus généralement, il serait intéressant d'intégrer ces apprentissages associatifs dans un mécanisme d'intégration de nouveaux comportements : si une situation s'avère intéressante, il serait utile de la ré-exploiter à l'avenir. Un exemple : un prédateur n'a aucun intérêt pour les plantes *a priori*. Cependant, s'il capture une proie près d'une variété végétale donnée, il peut associer les deux et, à l'avenir, quêter préférentiellement près de ces plantes. Ce type d'apprentissage doit se renforcer en fonction des expériences, mais pouvoir se mettre en place dès les premières corrélations.

Un outil intéressant pour cela pourrait être un mécanisme d'apprentissage de type classifieurs (tels qu'utilisés par Sanza [95]) permettant d'attribuer des poids à des associations stimulus-réponse, couplé éventuellement à un mécanisme permettant la création de nouvelles règles en cas d'expériences intéressantes.

Ces renforcements peuvent aussi permettre la création de comportements sociaux, tels des chasses en commun ou des regroupements de type Reynolds. Pour renforcer cela, il serait intéressant de fournir un mécanisme de marquage persistant (phéromones) ou instantané (sons), simplement sous forme d'effecteurs et de capteurs, mais sans leur donner de sémantique spécifique : encore une fois, leur exploitation (et leur détournement par les concurrents) doit être optimisée par les créatures.

Cette approche de la génération du comportement va résolument dans le sens de Brooks : pas de planification ni de génération de modèle du monde *a priori* nécessaires, pas de modules fonctionnels à proprement parler ; mais elle tient aussi compte des caractéristiques d'un environnement simulé (plus d'informations dans les perceptions). La transposition de son approche dans un environnement non réel, tout en tenant compte de sa philosophie, nous semble être un test original de ses idées.

▷ **Reproduction** Pour optimiser les comportements au cours du temps (il ne peut être attendu de chaque individu qu'il apprenne un répertoire intéressant au cours d'une existence), il serait utile de mettre en place un mécanisme lamarkien de transmission de ces agents, s'apparentant aux mécanismes de transmission culturelle de la section E.2.6. Notons que nous ne faisons pas d'hypothèses sur la manière dont ils sont implémentés : nous n'imposons pas en particulier qu'un mécanisme simple de type réseau de calcul soit lui-même altéré par une méthode d'apprentissage durant la vie de l'individu, ou soumis à des mutations pour en tester de nouveaux (l'espace à explorer est trop grand, et l'objectif n'est pas une crédibilité biologique absolue).

De plus, l'avantage d'une reproduction est de permettre un *bootstrapping* du système via l'introduction de quelques comportements élémentaires permettant la survie des premières gé-

nérations. Les comportements moteurs décrits ci-dessus devraient eux aussi pouvoir y être codés.

▷ **Végétaux** Les plantes permettront l'injection d'énergie dans le système. Leur simulation devra être la plus simple possible, mais certaines caractéristiques seraient intéressantes : croissance et reproduction, afin d'obliger les herbivores à gérer leurs stocks ; développement en fonction de la lumière, afin de limiter la densité de végétaux ; valeur alimentaire dépendant de la taille. Tout ceci serait un élément de variabilité de l'environnement indépendamment des actions des créatures.

Chapitre 5

Conclusion

Après une présentation des problématiques et outils de l'informatique graphique, nous avons montré comment ceux-ci pouvaient être rapprochés de ceux de la robotique adaptative et de la génération de comportements autonomes. Cela nous a amenés à analyser et développer une plate-forme de simulation basée sur la notion de simulation physique et à étudier les difficultés associées à cette dernière. Il est ainsi possible de générer, à partir d'un modèle géométrique continu animé par squelette tels qu'ils sont classiquement utilisés en informatique graphique, une approximation physique polyarticulée. Celle-ci permet en retour d'animer le modèle graphique initial via une simulation réaliste. Finalement, nous avons proposé un ensemble de développements possibles à partir de ce cadre de travail.

Cet enchaînement de propositions mène à une simulation de comportements à proprement parler, au sens de mouvements coordonnés en fonction de perceptions et d'objectifs propres. Le type d'interaction entre créatures y est cependant minimal, se résumant essentiellement à des comportements relativement réactifs de poursuite ou d'évitement.

Dépasser ce niveau impliquerait d'assigner aux créatures des tâches plus complexes, requérant une coopération et potentiellement une communication entre elles pour être menées à bien. Il est alors possible d'utiliser une notion que nous avons soigneusement évité d'employer jusqu'à présent, celle de système multi-agents (SMA). Pour nous, celle-ci impose en effet la notion de communication, qui n'arrive que tardivement dans nos perspectives ; il serait par conséquent fallacieux d'associer nos travaux à ce domaine de recherche à leur stade actuel de développement.

L'approche que nous avons retenue est d'ailleurs finalement inverse de celle traditionnellement utilisée en SMA : plutôt que de se focaliser sur la communication et la coopération dans le cadre d'un environnement simplifié, nous partons de considérations d'animation réaliste de mouvements élémentaires pour en arriver à ces problématiques. Ce faisant, nous nous sommes progressivement éloignés des spécificités de l'informatique graphique pour en arriver à des questions de contrôle autonome caractéristiques des animats, progression qui, notre bibliographie l'a mis en avant, se retrouve dans l'évolution de cette discipline.

Ce projet de recherche est bien trop vaste pour être atteignable dans le cadre d'un doctorat ; au-delà de l'ampleur des développements possibles, tels qu'esquissés au cours du chapitre précédent, son aspect pluridisciplinaire impose en effet une familiarisation avec un nombre

important de concepts et d'outils. Dans cette optique, nous espérons que ce document, au-delà du travail d'exposition et d'éclaircissement de nos idées qu'il nous a permis de mener, pourra être un point de départ utile pour quiconque désirerait se lancer efficacement dans des travaux similaires.

Annexe A

Éléments de Mécanique

Introduction

Ceci est la compilation d'un ensemble de notes prises dans des ouvrages de mécanique de niveau DEUG / écoles préparatoires [1, 11, 89]. Afin de limiter au maximum les prérequis tout en allégeant le corps du textes, les notions mathématiques essentielles utilisées ici (repères, torseurs) ont été reportées en annexe B. De même, les notions d'énergie et de travail n'ont pas été abordées ici : elles ne sont en effet pas indispensables à une description des mouvements, et permettent de faire l'économie des opérateurs différentiels.

Deux présentations sont possibles : par type d'objet considéré (point matériel, ensemble de points et solide) ou par classe de lois (cinétique, dynamique). La première approche a été retenue, au prix peut-être d'une certaine redondance.

Ce document aborde essentiellement la mécanique des solides indéformables ; un modèle plus proche des solides réels et des fluides est donné par les *milieux continus*, non traités ici.

A.1 Mécanique du Point

A.1.1 Trajectoire d'un Point

Dans un référentiel donné, la courbe décrite par un point mobile est nommée trajectoire. Soit P ce point, on peut décrire sa position $P(t)$ au cours du temps via un système de coordonnées.

A.1.2 Vecteur Vitesse d'un Point

Le vecteur vitesse du point $P(t)$ par rapport au repère R, à la date t, est la dérivée par rapport à t, pour un observateur lié au repère R, du vecteur position $\vec{OP}(t)$ (où O est un point fixe de R).

$$\vec{V}(P/R) = \left[\frac{d\vec{OP}(t)}{dt} \right]_R$$

Sa norme s'exprime en $m.s^{-1}$. Exprimons cette dérivée en coordonnées cartésiennes :

$$\vec{V} = \dot{x}\vec{x} + \dot{y}\vec{y} + \dot{z}\vec{z}$$

A.1.3 Vecteur Accélération

Le vecteur accélération du point $P(t)$, par rapport au repère R, à la date t, est la dérivée par rapport à t, pour un observateur lié au repère R, du vecteur vitesse $\vec{V}(P/R)$.

$$\vec{\Gamma}(P/R) = \left[\frac{d\vec{V}(t)}{dt} \right]_R$$

Sa norme s'exprime en $m.s^{-2}$. En coordonnées cartésiennes, on obtient :

$$\vec{\Gamma} = \ddot{x}\vec{x} + \ddot{y}\vec{y} + \ddot{z}\vec{z}$$

A.1.4 Relation de Changement de Base de Dérivation

La relation entre la dérivée d'un vecteur $\vec{U}(t)$ dans deux repères R et R_1 est :

$$\left[\frac{d\vec{U}(t)}{dt} \right]_{R_1} = \left[\frac{d\vec{U}(t)}{dt} \right]_R + \vec{\Omega}(R/R_1) \wedge \vec{U}(t)$$

Où $\vec{\Omega}(R/R_1)$, aussi noté $\vec{\omega_e}$, est un vecteur libre appelé *vecteur vitesse instantanée de rotation* de la base de R par rapport à la base de R_1 ; sa norme s'exprime en $rad.s^{-1}$.

Dans le cas (usuel) d'une rotation autour d'un axe (orienté par un vecteur unitaire $\vec{u}$), on pose $\vec{\Omega}(R/R_1) = \omega\vec{u}$, où ω est la vitesse angulaire de rotation.

Relation entre les Vecteurs Vitesse d'un Point

Soit P un point en mouvement dans deux repères R et R_1 d'origines respectives O et O_1. La relation entre ses vecteurs vitesse dans ces deux repères est :

$$\vec{V}(P/R_1) = \vec{V}(P/R) + \vec{V}(P \in R/R_1)$$

Ou plus simplement

$$\vec{v_a} = \vec{v_r} + \vec{v_e}$$

$\vec{v_a}$ est la *vitesse absolue*, $\vec{v_r}$ la *vitesse relative* et $\vec{v_e}$ la *vitesse d'entraînement.*

On a la relation

$$\vec{v_e} = \frac{d\vec{O_1O}}{dt} + \vec{\omega_e} \wedge \vec{OP}$$

Dans le cas particulier de la rotation d'un point autour d'un axe portant un vecteur unitaire $\vec{u}$, on a

$$\vec{v_e} = \vec{\omega_e} \wedge \vec{OP} = \omega(\vec{u} \wedge \vec{OP})$$

Relation entre les Vecteurs Accélération d'un Point

Soit un point P en mouvement par rapport à deux repères R et R_1 d'origines respectives O et O_1. La relation entre ses vecteurs accélération est :

$$\vec{\Gamma}(P/R_1) = \vec{\Gamma}(P/R) + \vec{\Gamma}(P \in R/R_1) + 2\vec{\Omega}(R/R_1) \wedge \vec{V}(P/R)$$

Ou plus simplement

$$\vec{a_a} = \vec{a_r} + \vec{a_e} + \vec{a_c}$$

$\vec{a_a}$ est l'*accélération absolue*, $\vec{a_r}$ l'*accélération relative*, $\vec{a_e}$ le *vecteur accélération d'entraînement* et $\vec{a_c} = 2\vec{\omega_e} \wedge \vec{v_r}$ le *vecteur accélération de Coriolis*.

On a la relation

$$\vec{a_e} = \frac{d^2\vec{O_1O}}{dt^2} + \frac{d\vec{\omega_e}}{dt} \wedge \vec{OP} + \vec{\omega_e} \wedge \frac{\vec{OP}}{dt}$$

Notons que si on a $R = (O, \vec{e_x}, \vec{e_y}, \vec{e_z})$ (vecteurs unitaires), on a pour chaque vecteur

$$\left(\frac{d\vec{e_i}}{dt}\right)_{R_1} = \vec{\omega_e} \wedge \vec{e_i}$$

A.1.5 Point Matériel

Un point matériel est un point de l'espace auquel est affectée une masse, grandeur positive caractéristique de son inertie. Ce concept représente bien les particules chargées et le centre de gravité G des systèmes et solides.

Éléments Cinétiques

Soit dans le référentiel R_1, d'origine O_1, un point A de masse m, de vitesse $\vec{v}$ et d'accélération $\vec{a}$. Si de plus O est un point de R, on définit pour A les éléments cinétiques suivants :

la quantité de mouvement	$\vec{p} = m\vec{v}$
le moment cinétique par rapport à O	$\vec{\sigma_O} = \vec{OA} \wedge \vec{p}$
la quantité d'accélération	$\vec{S} = m\vec{a}$
le moment dynamique par rapport à O	$\vec{\delta_O} = \vec{OA} \wedge \vec{S}$

$\vec{\delta_O}$ est parfois noté K_O.

A.1.6 Notion de Force

Modélisation

Les actions mécaniques s'exerçant sur un point peuvent être représentées par une grandeur vectorielle nommée force (éventuellement résultante de plusieurs termes). Plus précisément, il s'agit d'un vecteur lié, c'est-à-dire d'un couple $(P, \vec{F})$.

Réaction d'un Support - Notion de Liaison

Un point glissant le long d'une surface est soumis à une force de contact (*réaction du support*) ; ce type de force est nommé liaison. Si ce glissement se fait sans frottement, la réaction est portée par la normale à la surface en ce point.

A.1.7 Principe des Actions Réciproques

Dans le cas des forces s'exerçant entre deux points matériels, on a (*principe des actions réciproques* ou *principe de l'action et de la réaction*) :

$$\vec{F_{AB}} = -\vec{F_{BA}}$$

Et $\vec{F_{AB}}$ et $\vec{F_{BA}}$ sont portées par AB.

A.1.8 Référentiels Galiléens - Principe d'Inertie

Le *principe d'inertie* stipule que, dans un *référentiel galiléen*, un point isolé a un mouvement rectiligne uniforme. Tout référentiel en translation rectiligne uniforme par rapport à un référentiel galiléen est lui-même galiléen.

Les lois de la mécanique sont invariantes vis-à-vis de la transformation de Galilée, c'est-à-dire du passage d'un référentiel galiléen à un autre.

A.1.9 Principe Fondamental de la Dynamique du Point

Dans un Repère Galiléen

Soit un point de quantité de mouvement $\vec{p}$ auquel on applique des forces de résultante $\vec{f}$. On a alors

$$\vec{f} = \frac{d\vec{p}}{dt}$$

Dans le cas usuel où la masse est constante, on a (notations évidentes) :

$$\vec{f} = m\vec{a}$$

Cas d'un Repère non Galiléen - Forces d'Inertie

On a alors (notation du A.1.4)

$$\vec{f} = \frac{d\vec{p}}{dt} + \vec{f_i} \text{ avec } \vec{f_i} = -m\vec{a_e} - m\vec{a_c}$$

Où $\vec{f_i}$ représente les forces d'inertie, $-m\vec{a_e}$ la force d'inertie d'entraînement et $-m\vec{a_c}$ la force d'inertie de Coriolis.

A.1.10 Théorème du Moment Cinétique

Moment d'une Force - Théorème du Moment Cinétique

Soit un point P de moment cinétique $\vec{\sigma_O}$ dans R, repère galiléen d'origine O, et soit $\vec{f}$ la résultante des forces s'exerçant sur P.

Si O est fixe, on définit le moment en O de $\vec{f}$ par

$$\mathcal{M} = \vec{OP} \wedge \vec{f}$$

On a alors le *théorème du moment cinétique* :

$$\frac{d\vec{\sigma_O}}{dt} = \mathcal{M}(O)$$

Choc de Particules

On s'intéressera ici aux vitesses après le chocs.

La conservation de la quantité de mouvement est toujours valide :

$$\vec{p_1} + \vec{p_2} = \vec{p_1}' + \vec{p_2}'$$

Cette équation étant insuffisante pour caractériser le mouvement des particules après le choc, on fait en général des hypothèses sur la nature du choc.

Un choc est dit parfaitement élastique si l'énergie cinétique est conservée, et parfaitement mou si les deux points restent solidaires après le choc. Un paramètre e est utilisé pour classifier les intermédiaires ; avec u et u' les vitesses relatives des particules avant et après le choc, on pose :

$$u' = -eu$$

Pour $e = 0$, le choc est parfaitement mou, et pour $e = 1$ il est parfaitement élastique.

A.2 Mécanique des Systèmes de Points

A.2.1 Masse du Système

Soit un système de n points de masses respectives m_i. m désigne alors la masse totale du système de points, définie par :

$$m = \sum_i m_i$$

A.2.2 Éléments Cinétiques

Barycentre (Centre d'Inertie)

Pour un système de n points A_i de masse m_i, le barycentre G est défini par :

$$\vec{OG} = \frac{1}{m}\sum_i m_i \vec{OA_i} \Leftrightarrow \sum_i m_i \vec{GA_i} = \vec{0}$$

Le référentiel barycentrique (noté R^*) a pour origine G et des axes parallèles à des directions fixes. Une grandeur g définie dans ce référentiel est notée g^*.

Torseur Cinétique

Soit un système de points A_i de quantités de mouvement respectives $\vec{p_i}$ dans un référentiel R_1. Le torseur cinétique $[P]$ est alors défini par ses composantes en un point O :

$$\vec{P} = \sum_i \vec{p_i}$$
$$\vec{\sigma_O} = \sum_i \vec{OA_i} \wedge \vec{p_i}$$

$\vec{P}$ est la *résultante cinétique* et $\vec{\sigma_O}$ le *moment cinétique en O* de ce système dans R_1. On a :

$$\vec{\sigma_{O'}} = \vec{\sigma_O} + \vec{P} \wedge \vec{OO'}$$
$$\vec{P} = m\vec{v_G}$$
$$\vec{P^*} = \vec{0}$$

Torseur Dynamique

De même, on définit les composantes du torseur $[S]$:

$$\vec{S} = \sum_i \vec{S_i}$$
$$\vec{\delta_O} = \sum_i \vec{OA_i} \wedge \vec{S_i}$$

$\vec{S}$ est la *résultante dynamique* et $\vec{\delta_O}$ le *moment dynamique en O*. On a :

$$\vec{\delta_{O'}} = \vec{\delta_O} + \vec{S} \wedge \vec{OO'}$$
$$\vec{S} = m\vec{a_G} = \frac{d\vec{P}}{dt}$$

Rapport entre les Torseurs Cinétiques et Dynamiques

On a dans tous les cas :

$$\begin{aligned} \vec{S} &= \tfrac{d\vec{P}}{dt} \\ \vec{\delta_A} &= \left[\tfrac{d\vec{\sigma_A}}{dt}\right]_{R_1} + m\vec{v_A} \wedge \vec{v_G} \end{aligned}$$

Donc si A est fixe dans R_1 ou a une vitesse parallèle à celle de G :

$$\vec{\delta_A} = \frac{d\vec{\sigma_A}}{dt}$$

Moments Cinétiques et Dynamiques par Rapport à un Axe

Soit un axe Δ, O un de ses points et $\vec{u}$ un vecteur unitaire porté par cet axe. Le moment cinétique d'un système par rapport à Δ est caractérisé par l'expression :

$$\sigma_\Delta = \vec{u} \cdot \vec{\sigma_O}$$

De même pour le moment dynamique par rapport à Δ (cf. B.3.2).

Premier Théorème de Koenig

Il permet le calcul de $\vec{\sigma_O}$, $\vec{\delta_O}$ dans R_1 en étudiant dans R_1 (galiléen) le mouvement de G et en utilisant les grandeurs $\vec{\sigma_G}^*$, $\vec{\delta_G}^*$ et ${E_c}^*$.

$$\begin{aligned} \vec{\sigma_O} &= \vec{OG} \wedge \vec{p_G} + \vec{\sigma_G}^* \\ \vec{\delta_O} &= \vec{OG} \wedge \vec{S_G} + \vec{\delta_G}^* \end{aligned}$$

Ces résultats sont valable que O soit fixe ou non dans R_1.

Le second théorème de Koenig, non détaillé ici, permet le calcul de l'énergie cinétique.

A.2.3 Notion d'Action Mécanique

Actions Mécaniques Intérieures et Extérieures

Soit un ensemble de points A_i : les actions mécaniques intérieures (ou forces intérieures) sont les forces d'interaction entre les points du système. Le principe des actions réciproques (cf. section A.1.7) s'applique : si $\vec{f_{ij}}$ est la force que A_i applique sur A_j on a

$$\vec{f_{ij}} = -\vec{f_{ji}}$$

et ces deux forces sont portées par la droite (A_iA_j).

Les actions mécaniques extérieures (ou forces extérieures) sont les forces appliquées à ces points dont la source est à l'extérieur du système.

Formalisation : Premier Principe de la Statique

Toute action mécanique est entièrement caractérisée, d'un point de vue mécanique, par un torseur. Si un corps (S) subit de la part d'un ensemble matériel (E) une action mécanique représentée par un système de n forces $(P_i, \vec{f_i})$, le torseur représentant l'action mécanique de (E) sur (S) s'écrit (éléments de réduction en A) :

$$\{\mathcal{T}\} = \begin{Bmatrix} \vec{R} \\ \vec{\mathcal{M}}_A \end{Bmatrix}_A$$

Avec :

$$\begin{cases} \vec{R} &= \sum_i \vec{f_i} \\ \vec{\mathcal{M}_A} &= \sum_i \vec{AP_i} \wedge \vec{f_i} \end{cases}$$

$\{\mathcal{T}\}$ (aussi noté $[F]$) est nommé *torseur d'action mécanique* de (E) sur (S), $\vec{R}$ *résultante générale* et $\vec{\mathcal{M}_A}$ *moment résultant au point A* de l'action mécanique de (E) sur (S).

On a évidemment :

$$\mathcal{M}_{A'} = \mathcal{M}_A + \vec{R} \wedge \vec{AA'}$$

A.2.4 Théorème de la Résultante Cinétique

Soit un système de points de résultante cinétique $\vec{P}$ dans un référentiel galiléen et $\vec{R_{ext}}$ la résultante des forces extérieures au système. On a alors :

$$\frac{d\vec{P}}{dt} = \vec{R_{ext}}$$

Ce résultat est aussi nommé *théorème du centre d'inertie*.

A.2.5 Théorème du Moment Cinétique / Dynamique

Avec les hypothèses précédentes, ces deux théorèmes se traduisent par les relations :

$$\frac{d\vec{\sigma_O}}{dt} = \mathcal{M}_{(O)ext}$$
$$\vec{\delta_O} = \mathcal{M}_{(O)ext}$$

Le théorème du moment cinétique est valable avec O fixe dans R_1, de vitesse nulle à l'instant d'application du théorème ou de vitesse parallèle à celle de G. Mais son utilisation dans le cas d'un point mobile est de toute façon déconseillée. Notons que le théorème du moment dynamique n'a pas ces limitations.

Théorème du Moment Cinétique (Référentiel Barycentrique)

Que le référentiel barycentrique soit ou non galiléen, on a dans tous les cas :

$$\frac{d\vec{\sigma^*_G}}{dt} = \mathcal{M}_{(G)ext}$$

A.3 Mécanique des Solides

A.3.1 Notion de Solide

Il s'agit d'un corps idéalisé dont la géométrie est indépendante des actions mécaniques qu'il subit : les distances entre ses points restent constantes.

Ainsi, la position d'un solide dans un référentiel dépend au plus de six scalaires (degrés de liberté), la position de son barycentre et trois angles, par exemple.

A.3.2 Champ des Vecteurs Vitesse des Points d'un Solide

Soit un solide (S) en mouvement par rapport à un repère $R(O, \vec{x}, \vec{y}, \vec{z})$. Soient A et B deux points de (S), et un repère lié à (S) noté lui aussi (S).

On a alors la relation :

$$\vec{V}(B/R) = \vec{V}(A/R) + \vec{\Omega}(S/R) \wedge \vec{AB}$$

Ou de manière plus concise :

$$\vec{v_B} = \vec{v_A} + \vec{\omega} \wedge \vec{AB}$$

Les vecteurs vitesses des points d'un solide vérifient ainsi la relation de changement de point du moment d'un torseur, ainsi que nous le verrons en A.3.2.

Vecteur Vitesse Instantané de Rotation

$\vec{\omega}$ est un vecteur dépendant du temps appelé vitesse instantanée de rotation du solide (S).

▷ **Solide en rotation autour d'un axe** Si le solide est en rotation autour d'un axe Δ de vecteur unitaire $\vec{u}$, et si la vitesse angulaire de cette rotation est $\omega = \frac{d\theta}{dt}$, on a :

$$\vec{\omega} = \omega\vec{u}$$

▷ **Cas général** $\vec{\omega}$ est habituellement calculé par décomposition en rotations autour d'axes fixés dans un référentiel R. Soit $\vec{u_i}$ les vecteurs unitaires portés par ces axes et $\omega_i = \frac{d\theta_i}{dt}$ les vitesses angulaires correspondantes, on a alors :

$$\vec{\omega}(S/R) = \sum_i \omega\vec{u_i}$$

▷ **Relation entre les vecteurs rotation d'un solide** Soit un solide (S) en mouvement par rapport à deux repères R et R_1. Ses vecteurs rotation sont lié par la relation :

$$\vec{\Omega}(S/R_1) = \vec{\Omega}(S/R) + \vec{\Omega}(R/R_1)$$

Vecteur Vitesse de Glissement

Le vecteur vitesse de glissement au point P du solide (S_2) par rapport au solide (S_1) est le vecteur : $\vec{V}(P \in S_2/S_1)$. Ce vecteur est parallèle au plan tangent en P à (S_1) et (S_2).

Si (S_2) roule sans glisser sur (S_1), $\vec{V}(P \in S_2/S_1) = \vec{0}$.

Torseur Cinématique

Le champ des vecteurs vitesse des points du solide (S) dans son mouvement par rapport au repère R est représenté, au point A ($A \in S$), par le torseur suivant (*torseur cinématique* ou *torseur des vitesses*) :

$$\{\mathcal{V}(S/R)\} = \left\{ \begin{array}{c} \vec{\Omega}(S/R) \\ \vec{V}(A/R) \end{array} \right\}_A$$

▷ **Axe Instantané de Rotation** Il est possible de montrer que l'ensemble des points de vitesse colinéaire à $\vec{\omega}$ est une droite colinéaire à $\vec{\omega}$. Ceci définit l'axe central du torseur cinématique, nommé aussi *axe instantané de rotation* ou *axe instantané de rotation-glissement*.

▷ **Torseurs Cinématiques Particulier**

- Un *couple* est un torseur cinématique à résultante nulle. Il définit une translation.
- Un *torseur à résultante* est un torseur cinématique à moment nul. Il définit un mouvement de rotation instantanée.

▷ **Composition des Torseurs Cinématiques** Soit un solide (S) en mouvement par rapport à deux repères R et R_1. La relation de composition des torseurs cinématiques s'écrit alors :

$$\{\mathcal{V}(S/R_0)\} = \{\mathcal{V}(S/R)\} + \{\mathcal{V}(R/R_0)\}$$

A.3.3 Champ des Vecteurs Accélération des Points d'un Solide

Pour deux points A et B d'un solide (S), on a la relation :

$$\vec{\Gamma}(B/R) = \vec{\Gamma}(A/R) + \left[\frac{d}{dt}\vec{\Omega}(S/R)\right]_R \wedge \vec{AB} + \vec{\Omega}(S/R) \wedge [\vec{\Omega}(S/R) \wedge \vec{AB}]$$

Le champ des vecteurs accélération des points d'un solide n'est pas représentable par un torseur.

A.3.4 Définition des Éléments Cinétiques

Le découpage du solide (S) en éléments de volume $d\tau$, de centre P et de masse $dm = \rho d\tau$ (où ρ est la masse volumique) permet de se ramener au cas des systèmes de points.

Masse du Solide

La masse m du solide est alors définie par :

$$m = \iiint_{(S)} dm$$

Centre d'Inertie (Barycentre)

Pour tout point A, la position du *centre d'inertie* (ou *centre de gravité*) G du solide (E) est donnée par :

$$\vec{AG} = \frac{1}{m}\int_{P\in E} \vec{AP} dm$$

Notons que si l'on considère une partition de E en n éléments $E_i(m_i, G_i)$, on a pour tout A (cf. section A.2.2) :

$$\vec{AG} = \frac{1}{m}\sum_{i=1}^{n} m_i \vec{AG}_i$$

Torseur Cinétique

Soit un solide (S) de masse m, de centre d'inertie G en mouvement par rapport à un repère R. Son *torseur cinétique* en A est :

$$\{\mathcal{C}(E/R)\} = \left\{ \begin{array}{l} \vec{P} = \displaystyle\int_{P\in S} \vec{v_P} dm = m\vec{v_G} \\ \vec{\sigma_A} = \displaystyle\int_{P\in S} \vec{AP} \wedge \vec{v_P} dm \end{array} \right\}_A$$

La résultante générale du torseur cinétique est nommée *résultante cinétique* (ou *quantité de mouvement*) et son moment résultant *moment cinétique*.

Torseur Dynamique

Soit un solide (S) de masse m, de centre d'inertie G en mouvement par rapport à un repère R. Son *torseur dynamique* en A est :

$$\{\mathcal{D}(E/R)\} = \left\{ \begin{array}{l} \vec{S} = \displaystyle\int_{P\in S} \vec{a_P} dm = m\vec{a_G} \\ \vec{\delta_A} = \displaystyle\int_{P\in S} \vec{AP} \wedge \vec{a_P} dm \end{array} \right\}_A$$

La résultante générale du torseur dynamique est nommée *résultante dynamique* et son moment résultant *moment dynamique*.

Notons que ces définitions des torseurs dynamiques et cinétiques sont valables pour un système quelconque ; de plus, les résultats de sous-sections du A.2.2 restent valables, en particulier les théorèmes de Koenig.

A.3.5 Solide en Mouvement autour d'un Point Fixe

Champ des Vitesses

Soit un solide (S) dont un point est fixe dans R_1 (G par exemple si R_1 est le référentiel barycentrique). La relation du A.3.2 devient :

$$\vec{v}(M/R_1) = \vec{\omega}(S/R_1) \wedge \vec{OM}$$

Opérateur d'Inertie

L'*opérateur d'inertie* d'un solide (S) au point (O) est l'opérateur qui, à tout vecteur $\vec{u}$, fait correspondre le vecteur :

$$\vec{j_o}(S, \vec{u}) = \int_{P \in S} \vec{OP} \wedge (\vec{u} \wedge \vec{O})dm$$

Matrice d'Inertie

L'opérateur du A.3.5 est linéaire, donc représentable par une matrice. La *matrice d'inertie* du solide (S) au point O, relativement à la base $(\vec{x}, \vec{y}, \vec{z})$ est :

$$[I_O] = \begin{bmatrix} A & -F & -E \\ -F & B & -D \\ -E & -D & C \end{bmatrix}_{(\vec{x},\vec{y},\vec{z})} = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yx} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{bmatrix}$$

avec

$$\begin{array}{ll} A = \int_{P \in S}(y^2 + z^2)dm & B = \int_{P \in S}(z^2 + x^2)dm \\ C = \int_{P \in S}(x^2 + y^2)dm & D = \int_{P \in S} yzdm \\ E = \int_{P \in S} zxdm & F = \int_{P \in S} xydm \end{array}$$

où A, B et C sont respectivement les moments d'inertie de (S) par rapport aux axes $(O, \vec{x})$, $(O, \vec{y})$ et $(O, \vec{z})$ et D, E et F respectivement les produits d'inertie de (S) par rapport aux axes $\{(O, \vec{y}), (O, \vec{z})\}$, $\{(O, \vec{z}), (O, \vec{x})\}$ et $\{(O, \vec{x}), (O, \vec{y})\}$.

Nous décrivons en section 3.5.3 des algorithmes de calcul de cette matrice du point de vue de l'informatique graphique, ainsi que du poids et du centre de gravité d'un solide, décrits par des intégrales similaires.

Si le point O et la base $(\vec{x}, \vec{y}, \vec{z})$ sont fixes par rapport au solide (S), les quantités A, B, C, D, E et F sont constantes au cours du temps.

Tenseur ou Opérateur d'Inertie

La matrice $[I_O]$ est la matrice d'inertie en O : ses éléments sont les composantes en (O) du tenseur (ou opérateur d'inertie) sur les axes choisis (se référer à la seconde expression de $[I_0]$ en A.3.5).

Pour un solide, il existe toujours trois directions orthogonales deux à deux telles que la matrice I_O soit diagonale. Ces axes correspondent aux *directions principales* du solide. Le tenseur calculé par rapport à ces directions est le *tenseur principal*.

Dans ce cas, les composantes I_{xx}, I_{yy} et I_{zz} sont respectivement nommées A, B et C. Ces *moments d'inertie principaux* représentent les moments d'inertie du solide par rapport aux axes principaux.

Si l'origine O des axes est en en G, le tenseur est dit *central*.

Si le solide tourne autour d'une de ses directions principales d'inertie, alors $\vec{\sigma_O}$ et $\vec{\omega}$ sont colinéaires.

Théorème de Huyghens

Soit (S) un solide de masse m et de centre d'inertie G. Posons $\vec{OG} = a\vec{x} + b\vec{y} + c\vec{z}$ dans la base $(\vec{x}, \vec{y}, \vec{z})$ considérée. On a alors :

▷ **Relation entre les moments d'inertie aux points O et G :**

$$\begin{aligned} A &= A_G + m(b^2 + c^2) \\ B &= B_G + m(c^2 + a^2) \\ C &= C_G + m(a^2 + b^2) \end{aligned}$$

▷ **Relation entre les produits d'inertie aux points O et G :**

$$\begin{aligned} D &= D_G + mbc \\ E &= E_G + mca \\ F &= F_G + mab \end{aligned}$$

▷ **Moment cinétique par rapport à O**

$$[\vec{\sigma_O}] = [I_O][\vec{\omega}]$$

▷ **Autre formulation** Soit un solide (S) en mouvement par rapport à un repère $R(O, \vec{x}, \vec{y}, \vec{z})$, A un point de (S) et $R_1(A, \vec{x_1}, \vec{y_1}, \vec{z_1})$ un repère lié à (S).

Son *moment cinétique* au point A est :

$$\vec{\sigma_A}(S/R) = m\vec{AG} \wedge \vec{V}(A \in S/R) + \vec{J_A}(S, \vec{\Omega}(S/R))$$

Moment Dynamique

Soit (S) en mouvement autour de O fixe. Considérons le repère d'origine O et d'axes liés au solide et dirigés suivant ses directions principales. Soit alors p, q et r les composantes de $\vec{\omega}$ sur ces axes, les composantes de $\vec{\delta_O}$ sur ces axes sont alors :

$$\begin{aligned}\delta_{O_x} &= A\dot{p} + (C-B)qr \\ \delta_{O_y} &= B\dot{q} + (A-C)rp \\ \delta_{O_z} &= C\dot{r} + (B-A)pq\end{aligned}$$

A.3.6 Solide en Rotation autour d'un Axe Fixe

Ceci est un cas particulier du A.3.5.

Moment d'Inertie par rapport à l'Axe

Soit $\Delta(O,\vec{i})$ l'axe de rotation, et r la distance de dm à Δ. On a alors

$$J_\Delta = \iiint_{(S)} r^2 dm$$

Ce moment d'inertie est un scalaire s'exprimant en $kg.m^2$. Il est aussi calculable par projection :

$$I(S/\Delta) = \vec{i}.\vec{J_O}(S,\vec{i})$$

Éléments Cinétiques

Soit ω la vitesse angulaire de rotation autour de Δ.

$$\begin{aligned}\sigma_\Delta &= J_\Delta \omega \\ \delta_\Delta &= J_\Delta \tfrac{\omega}{dt} \\ E_c &= \tfrac{1}{2} J_\Delta \omega^2\end{aligned}$$

Notons qu'en général, σ_O n'est pas colinéaire à Δ ; le scalaire σ_Δ n'est que la projection de σ_O sur Δ (cf. section B.3.2). Le calcul de σ_O nécessite l'utilisation de l'opérateur d'inertie du A.3.5.

Théorème d'Huygens

Soit a la distance entre G et Δ, et J_G le moment d'inertie par rapport à un axe passant par G et parallèle à Δ.

$$J_\Delta = J_G + ma^2$$

Pour une direction d'axe donné, le moment d'inertie est donc minimal lorsque l'axe passe par G.

A.3.7 Mouvement Général d'un Solide

Le mouvement général d'un solide peut se décomposer en une translation de son centre d'inertie G et une rotation du solide dans le référentiel barycentrique (où G est fixe).

Le premier théorème de Koenig mène à :

$$\vec{\sigma_O} = \vec{OG} \wedge m\vec{v_G} + \vec{\sigma_G^*}$$

avec

$$[\sigma_G^*] = [J_G][\omega]$$

De même,

$$\vec{\delta_O} = \vec{OG} \wedge m\vec{a_G} + \vec{\delta_G^*}$$

où $\vec{\delta_G^*}$ peut être calculé avec une méthode analogue à celle du A.3.5.

A.3.8 Actions Mécaniques s'Exerçant sur un Solide

Classification

Les actions mécaniques s'exerçant sur un solide peuvent être séparées en deux catégories : les actions intérieures, c'est-à-dire ayant lieu entre les différentes parties du solide, et les actions extérieures. Celles-ci se subdivisent à leur tour en actions mécaniques dues à un champ de force extérieur et celles issues du contact du solide avec un support, nommées *liaisons*. Ces dernières sont *a priori* des inconnues.

Modélisation

Exceptées les liaisons, on peut en général modéliser les actions mécaniques par une distribution volumique de forces : chaque élément de volume $d\tau$ entourant le point M subit alors une force :

$$d\vec{f} = \varphi(\vec{M})d\tau$$

Les éléments de réduction du torseur représentant ce système de forces élémentaires peuvent alors être exprimés en un point P (appartenant ou non au solide) :

$$\begin{aligned} \vec{R} &= \iiint_{(S)} \varphi(\vec{M})d\tau \\ \mathcal{M}_{(P)} &= \iiint_{(S)} P\vec{M} \wedge \varphi(\vec{M})d\tau \end{aligned}$$

Cette schématisation n'est pas absolument générale : elle ne s'accorde pas au cas d'un solide constitué par un diélectrique polarisé ou un milieu aimanté plongé dans un champ, par exemple. En effet, celui-ci subit dans ce cas en chaque point une action mécanique caractérisée non seulement par une force mais aussi par un couple (moment).

Notons que dans le cas des forces de pression, le théorème d'Archimède permet le calcul direct d'une résultante et d'un moment résultant.

Liaisons

▷ **Liaisons : cas d'un contact ponctuel** Nous considérons ici le contact entre un solide (S) et un autre solide (S_0) appelé support. Les actions mécaniques exercées par (S_0) sur (S) sont caractérisées au point de contact I par un torseur de résultante $\vec{R}$ (la *réaction*) et de moment $\mathcal{M}_I$.

$\vec{R}$ est une discrétisation de la notion de *densité surfacique des forces de contact* de $(S1)$ sur $(S2)$ en P. Elle est homogène à une force divisée par une surface, généralement exprimée en mégapascals ($1N/mm^2 = 1MPa$).

Notons que $\mathcal{M}(I) \neq \vec{0}$ en général : en effet, le contact n'est jamais rigoureusement ponctuel (l'action mécanique de contact s'effectue suivant une surface).

▷ **Interprétation des composantes de $\vec{R}$ et $\mathcal{M}(I)$** Soit (P) le plan tangent commun à (S) et (S_0) en I. Les éléments de réduction du torseur associé à la liaison sont décomposables en leurs composantes normales et tangentielles :

$$\vec{R} = \vec{N} + \vec{T}$$
$$\mathcal{M}(I) = \mathcal{M}_n(I) + \mathcal{M}_t(I)$$

Ces composantes ont les significations physiques suivantes :
– $\vec{N}$ s'oppose à l'interpénétration : il conditionne l'écrasement local des surfaces,
– $\vec{T}$ s'oppose au glissement relatif des solides,
– $\mathcal{M}_n(I)$ s'oppose au pivotement relatif des solides,
– $\mathcal{M}_t(I)$ s'oppose au roulement des solides l'un sur l'autre.

$\vec{N}$ est une discrétisation de la *densité surfacique normale*, ou *pression*, et $\vec{T}$ de la *densité surfacique tangencielle* des forces de contact de S sur S_0 (c'est la force minimale à exercer pour faire glisser (S)).

▷ **Lois de frottement : lois de Coulomb**

Loi relative au glissement Soient f le coefficient de frottement statique et f_c le coefficient de frottement cinétique. f est petit lorsque le frottement est faible (ordre de grandeur : 0.1 à 0.6 pour des contacts usuels, considéré comme nul si le frottement est négligeable).

Il n'y a pas glissement si $\frac{\|\vec{T}\|}{\|\vec{N}\|} < f$. S'il y a glissement, on a $\frac{\|\vec{T}\|}{\|\vec{N}\|} = f_c \leq f$ et $\vec{T}$ est alors colinéaire et de sens opposé à la vitesse de glissement (cf. section A.3.2) ; c'est la *loi de frottement solide*.

Cette inégalité peut être interprétée en introduisant l'*angle de frottement* φ défini par $f = \tan\varphi$. La condition signifie alors qu'il n'y a pas glissement tant que $\vec{R}$ reste dans le cône de demi-angle au sommet φ (*cône de frottement*).

Loi relative au pivotement Soient λ le paramètre statique de pivotement et λ_c le paramètre cinétique de pivotement ; ils sont homogènes à des longueurs.

Il n'y a pas pivotement si $\frac{\|\mathcal{M}_n\|}{\|\vec{N}\|} < \lambda$. S'il y a pivotement, on a $\frac{\|\mathcal{M}_n\|}{\|\vec{N}\|} = \lambda_c \leq \lambda$.

Loi relative au roulement Soient δ le paramètre statique de roulement et δ_c le paramètre cinétique de roulement (homogènes à des longueurs).

Il n'y a pas roulement si $\frac{\|\mathcal{M}_t\|}{\|\vec{N}\|} < \delta$. S'il y a roulement, on a $\frac{\|\mathcal{M}_t\|}{\|\vec{N}\|} = \delta_c \leq \delta$.

Remarque sur les liaisons Les liaisons (actions de contact) sont *a priori* des inconnues ; or les lois de Coulomb ne donnent pas systématiquement le bon nombre de relations à leur propos. Dans des cas simples, par contre, on arrive à une formulation complète.

Notons cependant que même dans ce cas, on peut ne pas pouvoir décrire les phases de mouvement par une équation différentielle unique, d'où une résolution difficile. Ceci motive les simplifications souvent effectuées.

Simplifications Les frottements de pivotement et de roulement sont souvent négligés. L'action de (S_0) sur (S) peut alors être modélisée par une force $\vec{F} = \vec{R}$ appliquée en I. *Dans ce cas particulier*, on a alors :

$$\mathcal{M}(I) = \vec{0}$$

▷ **Torseur statique d'une liaison** Soit une liaison (L_i) entre deux solides (S_1) et (S_2), et un repère $R(O, \vec{x}, \vec{y}, \vec{z})$ centré sur (L_i). Le torseur d'action mécanique de (S_1) sur (S_2) est alors défini par :

$$\left\{\mathcal{T}(S_1 \xrightarrow{L_i} S_2)\right\} = \underset{O}{\left\{\begin{array}{c} \vec{R}(S_1 \xrightarrow{L_i} S_2) \\ \vec{M}_O(S_1 \xrightarrow{L_i} S_2) \end{array}\right\}}$$

Pour simplifier, on adopte l'écriture :

$$\left\{\mathcal{T}(S_1 \xrightarrow{L_i} S_2)\right\} = \{\mathcal{T}_i\}$$

et ce torseur est nommé *torseur statique* de la liaison (L_i).

▷ **Torseur cinématique d'une liaison** De même qu'en A.3.8, on peut définir le *torseur cinématique* d'une liaison (L_i), noté $\{\mathcal{V}_i\}$ et équivalent au torseur cinématique du mouvement de (S_2) par rapport à (S_1) que la liaison (L_i) autorise.

Principe des Actions Réciproques (ou Mutuelles)

Soient deux systèmes (S_1) et (S_2) en interaction. Le torseur associé aux actions mécaniques que (S_1) exerce sur (S_2) est noté $[A_{12}]$. Alors, quelle que soit la nature de (S_1) et (S_2), on a :

$$[A_{12}] = -[A_{21}]$$

Par conséquent, les actions mécaniques intérieures à un système quelconque sont représentées par le torseur nul (il suffit pour cela de considérer (S_1) et (S_2) comme une partition d'un ensemble matériel en équilibre).

A.3.9 Équilibre d'un Solide : Statique

Un ensemble matériel (E) est en équilibre par rapport à un repère R si, au cours du temps, chaque point de (E) conserve une position fixe par rapport au repère R.

La condition nécessaire à l'équilibre de (S) dans un référentiel galiléen (R) est que le torseur associé aux actions mécaniques extérieures (y compris les liaisons), exprimé en un point quelconque, soit nul. Cela peut se formuler ainsi (*principe fondamental de la statique*) : il existe au moins un repère, appelé repère galiléen, tel que pour tout sous-ensemble matériel (e) de l'ensemble matériel (E) en équilibre par rapport à ce repère, le torseur associé aux actions mécaniques extérieures à (e) soit nul.

Si l'on note $\overline{e}$ l'extérieur de e, cela se formalise en $\exists R_g$ tel que $\forall(e) \subset (E)$, (e) est en équilibre par rapport à R_g :

$$\{\mathcal{T}(\overline{e} \rightarrow e)\} = \left\{\vec{0}\right\}$$

En décomposant cette égalité en deux sous-égalité sur la résultante générale et le moment résultant de ce torseur, on obtient les deux théorèmes généraux de la statique : le *théorème de la résultante statique* et le *théorème du moment statique*.

Pour que ces conditions soient suffisantes, il faut de plus que le solide soit initialement au repos et que les lois de frottement soient satisfaites (cf. section A.3.8).

A.3.10 Théorèmes Généraux de la Mécanique des Systèmes

Introduction

Le schéma usuel d'application des théorèmes généraux est le suivant : on identifie un terme caractéristique de la cinétique du système (composante du torseur cinétique, énergie cinétique) avec un terme issu de l'étude des actions mécaniques s'exerçant sur le système (résultante, moment résultant, travail). Des équations différentielles caractérisant le mouvement sont ainsi obtenues.

La méthodologie consiste en la définition d'un référentiel, d'où découle la classification des actions mécaniques en intérieures et extérieures (dont font partie les forces d'inertie).

Le nombre de degrés de liberté du système fournit alors le nombre d'équations à écrire. Celles-ci s'obtiennent via le théorème de la résultante cinétique, du moment cinétique, le principe fondamental de la dynamique (équivalent aux deux théorèmes précédents) et le théorème de l'énergie cinétique.

Théorème de la Résultante Cinétique

Il est aussi nommé *théorème du centre d'inertie*. Il stipule que dans un référentiel galiléen, le mouvement du centre d'inertie d'un système est le même que celui d'un point ayant pour

masse la masse totale du système, et auquel on appliquerait la résultante des forces extérieures au système.

$$\frac{d\vec{P}}{dt} = \vec{R_{ext}} \Leftrightarrow m\frac{d\vec{v_G}}{dt} = \vec{R_{ext}}$$

(m : masse totale du système ; la première égalité étant valable pour un point fixe dans le référentiel.)

Théorème du Moment Cinétique

Soit O un point fixe d'un référentiel galiléen. On a :

$$\frac{d\vec{\sigma_O}}{dt} = \mathcal{M}(O)_{ext}$$

Ce théorème est encore valable si O est un point mobile dont la vitesse est parallèle à celle de G dans le référentiel considéré.

Ce théorème est valable dans le référentiel barycentrique, qu'il soit ou non galiléen, et ce sans intervention des forces d'inertie :

$$\frac{\vec{\sigma_G^*}}{dt} = \mathcal{M}(G)_{ext}$$

Formulation par Torseur

Les théorèmes de la résultante cinétique et du moment cinétique peuvent s'exprimer sous la forme de l'égalité suivante :

$$[S] = [f_{ext}]$$

Lois de Conservation pour un Système Isolé

Si un système est isolé, le torseur associé aux actions mécaniques qui lui sont extérieures est nul par définition. Des théorèmes de la résultante et du moment cinétique, on déduit :

$$\begin{cases} \vec{P} = \vec{C_{te}} \\ \vec{\sigma_O} = \vec{C_t}e \end{cases}$$

Où O est un point fixe dans un référentiel galiléen ou le barycentre du système.

Principe Fondamental de la Dynamique

Il existe au moins un repère R_g, appelé repère galiléen, et au moins une chronologie, appelée chronologie galiléenne, tels que pour tout sous-ensemble matériel (e) d'un ensemble matériel (E), le torseur dynamique de (e) dans son mouvement par rapport à R_g soit égal au torseur des actions mécaniques extérieures à (e).

Soit

$$\forall(e) \subset (E) : \{\mathcal{D}(e/R_g)\} = \{\mathcal{T}(\overline{e} \to e)\}$$

Le *théorème de la résultante dynamique* et le *théorème du moment dynamique* sont tirés de cette égalité.

Notons que tout repère en translation rectiligne uniforme par rapport à un repère galiléen est aussi galiléen.

▷ **Théorème du moment dynamique** Il peut aussi être tiré de A.3.10 :

$$\vec{\delta_O} = \mathcal{M}(O)_{ext}$$

Ici, O n'est pas nécessairement fixe : aucun problème ne se pose si on a calculé directement $\vec{\delta_O}$ via le théorème de Koenig (cf. section A.2.2).

Principe Fondamental de la Dynamique (Repère non Galiléen)

Le principe fondamental de la dynamique s'applique relativement à tout repère, à condition d'ajouter au torseur des actions mécaniques extérieures le torseur des effets d'inertie d'entraînement et le torseur des effets d'inertie de Coriolis.

$$\{\mathcal{D}(e/R)\} = \{\mathcal{T}(\overline{e} \to e)\} + \{\mathcal{D}_{ie}(e, R/R_g)\} + \{\mathcal{D}_{ic}(e, R/R_g)\}$$

Avec

$$\{\mathcal{D}_{ie}(e, R/R_g)\} = \left\{ \begin{array}{c} -\int_{P\in e} \vec{\Gamma}(P \in R/R_g)dm \\ -\int_{P\in e} \vec{AP} \wedge \vec{\Gamma}(P \in R/R_g)dm \end{array} \right\}_A$$

$$\{\mathcal{D}_{ic}(e, R/R_g)\} = \left\{ \begin{array}{c} -\int_{P\in e} 2\vec{\Omega}(R/R_g) \wedge \vec{V}(P/R)dm \\ -\int_{P\in e} \vec{AP} \wedge [2\vec{\Omega}(R/R_g) \wedge \vec{V}(P/R)]dm \end{array} \right\}_A$$

Théorème des Actions Réciproques (Actions Mutuelles)

L'action mécanique du sous-ensemble matériel (e_2) sur le sous-ensemble matériel (e_1) (ceux-ci partitionnant un ensemble matériel (E)) est opposée à l'action mécanique de (e_1) sur (e_2).

$$\{\mathcal{T}(e_1 \to e_2)\} = -\{\mathcal{T}(e_2 \to e_1)\}$$

Il a déjà été démontré dans le cas particulier de la statique (se reporter à A.3.8). Ce principe est ici un théorème car il découle du théorème de la résultante cinétique et du théorème du moment cinétique, postulés dans le cas général des systèmes mécaniques.

Annexe B

Rappels Mathématiques

Ces éléments mathématiques sont nécessaires à la lecture de l'annexe A, mais peuvent aussi être utilisés dans le corps du texte (angles d'Euler en particulier).

B.1 Système de Coordonnées

Le système de coordonnées utilisé sera le repère cartésien : notons cependant que d'autres sont souvent utilisés afin de simplifier les calculs (repères cylindriques ou sphériques). On utilisera plus précisément le repère R d'origine O et de base $(\vec{x}, \vec{y}, \vec{z})$ (ces axes sont représentés en gras).

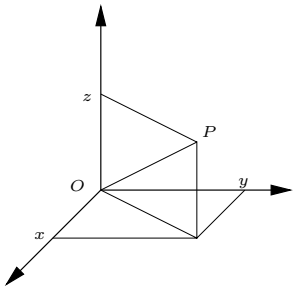


FIG. B.1 – Coordonnées cartésiennes

Les coordonnées cartésiennes x, y, z de P sont alors les projections orthogonales de $\vec{OP}$ sur la base du repère R.

Si nécessaire, les orientations pourront être représentées via des angles d'Euler.

B.2 Angles d'Euler

Trois paramètres indépendants suffisent pour orienter la base $(\vec{x_1}, \vec{y_1}, \vec{z_1})$ d'un repère R_1 par rapport à la base $(\vec{x}, \vec{y}, \vec{z})$ d'un repère R. On utilise habituellement pour cela les angles d'Euler.

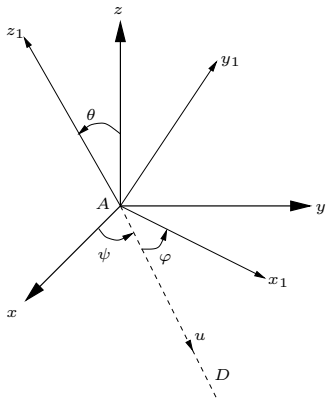


FIG. B.2 – Angles d'Euler

Soit D la droite d'intersubsection des plans $(A, \vec{x}, \vec{y})$ et $(A, \vec{x_1}, \vec{y_1})$ (axe nodal, ou ligne des nœuds) et $\vec{u}$ un vecteur unitaire de direction D. Les angles d'Euler sont alors :

$\psi = (\vec{x}, \vec{u})$: angle orienté par $\vec{z}$ (angle de précession)
$\theta = (\vec{z}, \vec{z_1})$: angle orienté par $\vec{u}$ (angle de nutation)
$\varphi = (\vec{u}, \vec{x_1})$: angle orienté par $\vec{z_1}$ (angle de rotation propre)

On peut considérer que ces trois angles correspondent à trois rotations planes successives, ce qui définit deux bases intermédiaires :

$$(\vec{x}, \vec{y}, \vec{z}) \xrightarrow{Rot(\vec{z},\psi)} (\vec{u}, \vec{v}, \vec{z}) \xrightarrow{Rot(\vec{u},\theta)} (\vec{u}, \vec{w}, \vec{z_1}) \xrightarrow{Rot(\vec{z_1},\varphi)} (\vec{x_1}, \vec{y_1}, \vec{z_1})$$

B.2.1 Repérage d'un solide mobile

Soit un solide mobile autour d'un point fixe. On peut le repérer via les angles d'Euler :

$$\vec{\omega} = \dot{\psi}\vec{z} + \dot{\theta}\vec{u} + \dot{\varphi}\vec{z_1}$$

Ou encore

$$\vec{\omega} = p\vec{x_1} + q\vec{y_1} + r\vec{z_1}$$

avec

$$\begin{aligned} p &= \dot{\psi}\sin\theta\sin\varphi + \dot{\theta}\cos\varphi \\ q &= \dot{\psi}\sin\theta\cos\varphi - \dot{\theta}\sin\varphi \\ r &= \dot{\phi}\cos\theta + \dot{\varphi} \end{aligned}$$

B.3 Torseurs

B.3.1 Définition par ses composantes

Un torseur est un objet géométrique $\mathcal{T}$ pouvant être défini par un point O et deux vecteurs : sa résultante $\vec{R}$ et son moment en O $\vec{M}_O$. Il est alors noté :

$$\{\mathcal{T}\} = \left\{ \begin{array}{c} \vec{R} \\ \vec{M}_O \end{array} \right\}_O$$

$\mathcal{T}$ définit un champ vectoriel vérifiant pour tous points A, B de l'espace la propriété :

$$\vec{M_B} = \vec{M_A} + \vec{R} \wedge \vec{AB}$$

B.3.2 Moment par rapport à un axe

Soit Δ un axe défini par $(O, \vec{u})$ et $\mathcal{M}$ le moment résultant attaché à un système de vecteurs. Le moment de ce système de vecteurs par rapport à Δ est alors par définition le scalaire :

$$\mathcal{M}_\Delta = \vec{u} \cdot \mathcal{M}_{(O)}$$

Ce moment ne dépend pas du choix du point O.

B.3.3 Équiprojectivité

Du fait que le champ des moments d'un torseur soit équiprojectif, on a pour tous points A, B de (S) la relation scalaire suivante :

$$\vec{AB}.\vec{V}(A/R) = \vec{AB}.\vec{V}(B/R)$$

B.3.4 Règles de calcul

Soient deux torseurs exprimés au même point A :

$$\{\mathcal{T}_1\} = \left\{ \begin{array}{c} \vec{R}(\mathcal{T}_1) \\ \vec{M}_A(\mathcal{T}_1) \end{array} \right\}_A \text{ et } \{\mathcal{T}_2\} = \left\{ \begin{array}{c} \vec{R}(\mathcal{T}_2) \\ \vec{M}_A(\mathcal{T}_2) \end{array} \right\}_A$$

▷ **Addition de torseurs** On appelle *somme des torseurs* $\{\mathcal{T}_1\}$ et $\{\mathcal{T}_2\}$ le torseur :

$$\{\mathcal{T}_1\} + \{\mathcal{T}_2\} = \left\{ \begin{array}{c} \vec{R}(\mathcal{T}_1) + \vec{R}(\mathcal{T}_2) \\ \vec{M}_A(\mathcal{T}_1) + \vec{M}_A(\mathcal{T}_2) \end{array} \right\}_A$$

▷ **Produit de torseurs** On appelle *produit des torseurs* $\{\mathcal{T}_1\}$ et $\{\mathcal{T}_2\}$ le scalaire :

$$\{\mathcal{T}_1\} \cdot \{\mathcal{T}_2\} = \vec{R}(\mathcal{T}_1) \cdot \vec{M}_A(\mathcal{T}_2) + \vec{R}(\mathcal{T}_2) \cdot \vec{M}_A(\mathcal{T}_1)$$

Ce produit est indépendant du point choisi pour exprimer les torseurs.

Annexe C

Développements Mathématiques

Cette annexe vise à détailler les calculs aboutissant aux résultats proposés au chapitre 3. Nous présenterons dans un premier temps les calculs de coefficients nécessaires à l'intégration par tétraèdres (section 3.5.3), puis la démonstration de la méthode de résolution associée à la génération d'articulations complexes (section 3.5.4).

C.1 Intégration par Tétraèdre

Nous détaillons ici les calculs et raisonnements présentés en section 3.5.3. Ceux-ci visent au calcul d'intégrales de type $\int_V f(x,y,z)dv$ sur des tétraèdres.

C.1.1 Integration sur un Tétraèdre Générique

Dans le cas particulier où trois côtés d'un tétraèdre forment une base orthogonale, le calcul des intégrales est aisément effectué en les décomposant le long des axes de cette base. Cette méthode ne peut pas être appliquée dans le cas général ; nous allons donc exploiter la propriété de changement de variable suivante :

Théorème 1 *Soit $g = (g_1, g_2, g_3)$ un difféomorphisme associant à tout point p d'un volume V un point $g(p) = (g_1(p_x), g_2(p_y), g_3(p_z)$ de V'. On a alors :*

$$\int_{V'} f(x,y,z)dv = \int_V f \circ g(u,v,w).\left|J\right|.du.dv.dw$$

Où $J = \frac{D(g_1,g_2,g_3)}{D(u,v,w)}$ est le jacobien de g, sa matrice jacobienne étant définie par :

$$M_J = \begin{pmatrix} \frac{\delta g_1}{\delta u} & \frac{\delta g_1}{\delta v} & \frac{\delta g_1}{\delta w} \\ \frac{\delta g_2}{\delta u} & \frac{\delta g_2}{\delta v} & \frac{\delta g_2}{\delta w} \\ \frac{\delta g_3}{\delta u} & \frac{\delta g_3}{\delta v} & \frac{\delta g_3}{\delta w} \end{pmatrix}$$

Nous allons déterminer la fonction g à utiliser afin de ramener un tétraèdre quelconque dans le cas particulier précédent, ce qui nous permettra ensuite d'utiliser cette propriété.

C.1.2 Changement de repère

Montrons qu'il existe une matrice M_{T_0T} associant à tout point du tétraèdre T_0 défini par $(O, \vec{i}, \vec{j}, \vec{k})$ un point du tétraèdre T défini par ses quatre sommets (S_0, S_1, S_2, S_3).

Cas général

Soit 4 points $P0$, $P1$, $P2$, $P3$. Peut-on trouver une matrice (homogène) M les transformant respectivement en $Q0$, $Q1$, $Q2$, $Q3$? Si les L_i sont des vecteurs à 4 composantes $[x, y, z, t]$, on pose :

$$M = \begin{pmatrix} & [L_0] & \\ & [L_1] & \\ & [L_2] & \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Et $M.P_0 = Q_0$ équivaut alors à :

$$\begin{aligned} L_0.P_0 &= Q_0x \\ L_1.P_0 &= Q_0y \\ L_2.P_0 &= Q_0z \end{aligned}$$

De même pour les trois autres couples de points ; on a donc un système de 12 inconnues (on connaît la dernière ligne de la matrice) et 12 équations. On peut le regrouper en trois sous-systèmes, un par L_i ; pour L_0 on a :

$$\begin{aligned} L_0.P_0 &= Q_0x \\ L_0.P_1 &= Q_1x \\ L_0.P_2 &= Q_2x \\ L_0.P_3 &= Q_3x \end{aligned}$$

Soient 4 équations et inconnues. En passant aux composantes :

$$\begin{aligned} L_0x.P_0x + L_0y.P_0y + L_0z.P_0z + LOt &= Q_0x \\ L_0x.P_1x + L_0y.P_1y + L_0z.P_1z + L1t &= Q_1x \\ L_0x.P_2x + L_0y.P_2y + L_0z.P_2z + L2t &= Q_2x \\ L_0x.P_3x + L_0y.P_3y + L_0z.P_3z + L3t &= Q_3x \end{aligned}$$

La matrice carrée associée au système est :

$$C = \begin{pmatrix} [P_0] & 1 \\ [P_1] & 1 \\ [P_2] & 1 \\ [P_3] & 1 \end{pmatrix}$$

Et on a $C.L_O = [Q_i x]$

Résolution du système

Ce système simple est résolvable par inversion de matrice. En effet, si C est régulière[1], on a $L_O = C^{-1}.[Q_i x]$. Si $D = C^{-1}$, on a ainsi :

$$\begin{aligned} L_0 &= D.[Q_i x] \\ L_1 &= D.[Q_i y] \\ L_2 &= D.[Q_i z] \end{aligned}$$

Cas de M_{T_0T}

On cherche maintenant la matrice de passage de T_0 à T définie par (S_0, S_1, S_2, S_3). On a :

$$C = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \quad \text{d'où} \quad D = C^{-1} = \begin{pmatrix} -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

D'où on déduit :

$$M_{T_0T} = \begin{pmatrix} S_1x - S_0x & S_2x - S_0x & S_3x - S_0x & S_0x \\ S_1y - S_0y & S_2y - S_0y & S_3y - S_0y & S_0y \\ S_1z - S_0z & S_2z - S_0z & S_3z - S_0z & S_0z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

On a alors :

$$\begin{aligned} |M_{T_0T}| &= (S_1x - S_0x) \cdot \begin{vmatrix} S_2y - S_0y & S_3y - S_0y \\ S_2z - S_0z & S_3z - S_0z \end{vmatrix} \\ &- (S_2x - S_0x) \cdot \begin{vmatrix} S_1y - S_0y & S_3y - S_0y \\ S_1z - S_0z & S_3z - S_0z \end{vmatrix} \\ &+ (S_3x - S_0x) \cdot \begin{vmatrix} S_1y - S_0y & S_2y - S_0y \\ S_1z - S_0z & S_2z - S_0z \end{vmatrix} \end{aligned}$$

[1] *i.e.* de déterminant non nul ; dans le cas contraire, les P_i ne forment pas une base et le tétraèdre associé est de volume nul.

C.1.3 Intégrale générale

La fonction g recherchée précédemment est donc exprimable sous forme de la matrice $M_{T_0T} = (m_{ij})$. Calculons maintenant sa matrice jacobienne associée :

$$\begin{aligned} g1(x,y,z) &= m_{11}.x + m_{12}.y + m_{13}.z + m_{14} \\ g2(x,y,z) &= m_{21}.x + m_{22}.y + m_{23}.z + m_{24} \\ g3(x,y,z) &= m_{31}.x + m_{32}.y + m_{33}.z + m_{34} \end{aligned}$$

$$M_J = \begin{pmatrix} \frac{\delta g_1}{\delta u} & \frac{\delta g_1}{\delta v} & \frac{\delta g_1}{\delta w} \\ \frac{\delta g_2}{\delta u} & \frac{\delta g_2}{\delta v} & \frac{\delta g_2}{\delta w} \\ \frac{\delta g_3}{\delta u} & \frac{\delta g_3}{\delta v} & \frac{\delta g_3}{\delta w} \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{pmatrix}$$

Le calcul montre que $|M_J| = |M_{T_0T}|$, or M_{T_0T} est inversible par définition (matrice de changement de repère) donc $|M_J|$ ne s'annule pas et g est un difféomorphisme. En utilisant le théorème 1, on en arrive donc à :

Théorème 2

$$\int_T f(x,y,z)dv = |M_{T_0T}| \int_{T_0} f(M_{T_0T}P)dv$$

C.1.4 Calcul des Paramètres Physiques d'un Tétraèdre

On peut maintenant utiliser le théorème précédent pour calculer les intégrales de la section 3.5.3 (on pose $d = |M_{T_0T}|$) :

$$\begin{aligned} \textstyle\int_T x.dv &= d.\textstyle\int_{T_0}(m_{11}.x + m_{12}.y + m_{13}.z + m_{14}).dv \\ \textstyle\int_T y.dv &= d.\textstyle\int_{T_0}(m_{21}.x + m_{22}.y + m_{23}.z + m_{24}).dv \\ \textstyle\int_T z.dv &= d.\textstyle\int_{T_0}(m_{31}.x + m_{32}.y + m_{33}.z + m_{34}).dv \\ \textstyle\int_T x^2.dv &= d.\textstyle\int_{T_0}(m_{11}.x + m_{12}.y + m_{13}.z + m_{14})^2.dv \\ \textstyle\int_T y^2.dv &= d.\textstyle\int_{T_0}(m_{21}.x + m_{22}.y + m_{23}.z + m_{24})^2.dv \\ \textstyle\int_T z^2.dv &= d.\textstyle\int_{T_0}(m_{31}.x + m_{32}.y + m_{33}.z + m_{34})^2.dv \\ \textstyle\int_T xy.dv &= d.\textstyle\int_{T_0}(m_{11}.x + m_{12}.y + m_{13}.z + m_{14}).(m_{21}.x + m_{22}.y + m_{23}.z + m_{24}).dv \\ \textstyle\int_T xz.dv &= d.\textstyle\int_{T_0}(m_{11}.x + m_{12}.y + m_{13}.z + m_{14}).(m_{31}.x + m_{32}.y + m_{33}.z + m_{34}).dv \\ \textstyle\int_T yz.dv &= d.\textstyle\int_{T_0}(m_{21}.x + m_{22}.y + m_{23}.z + m_{24}).(m_{31}.x + m_{32}.y + m_{33}.z + m_{34}).dv \end{aligned}$$

Dans le cas général, il faut donc calculer :

$$\int_{T_0}(c_0.x^2+c_1.y^2+c_2.z^2+c_3.x.y+c_4.x.z+c_5.y.z+c_6.x+c_7.y+c_8.z+c_9).dv = \int_{T_0} h(x,y,z).dv$$

Où les c_i sont des coefficients dépendant de m_{ij}. Par développement, on a :

	x	y	z	x^2	y^2	z^2	xy	xz	yz
c_0	0	0	0	m_{11}^2	m_{21}^2	m_{31}^2	$m_{11}.m_{21}$	$m_{11}.m_{31}$	$m_{21}.m_{31}$
c_1	0	0	0	m_{12}^2	m_{22}^2	m_{32}^2	$m_{12}.m_{22}$	$m_{12}.m_{32}$	$m_{22}.m_{32}$
c_2	0	0	0	m_{13}^2	m_{23}^2	m_{33}^2	$m_{13}.m_{23}$	$m_{13}.m_{33}$	$m_{23}.m_{33}$
c_3	0	0	0	$2.m_{11}.m_{12}$	$2.m_{21}.m_{22}$	$2.m_{31}.m_{32}$	$m_{11}.m_{22} + m_{12}.m_{21}$	$m_{11}.m_{32} + m_{12}.m_{31}$	$m_{21}.m_{32} + m_{22}.m_{31}$
c_4	0	0	0	$2.m_{11}.m_{13}$	$2.m_{21}.m_{23}$	$2.m_{31}.m_{33}$	$m_{11}.m_{23} + m_{13}.m_{21}$	$m_{11}.m_{33} + m_{13}.m_{31}$	$m_{21}.m_{33} + m_{23}.m_{31}$
c_5	0	0	0	$2.m_{12}.m_{13}$	$2.m_{22}.m_{23}$	$2.m_{32}.m_{33}$	$m_{12}.m_{23} + m_{13}.m_{22}$	$m_{12}.m_{33} + m_{13}.m_{32}$	$m_{22}.m_{33} + m_{23}.m_{32}$
c_6	m_{11}	m_{21}	m_{31}	$2.m_{11}.m_{14}$	$2.m_{21}.m_{24}$	$2.m_{31}.m_{34}$	$m_{11}.m_{24} + m_{14}.m_{21}$	$m_{11}.m_{34} + m_{14}.m_{31}$	$m_{21}.m_{34} + m_{24}.m_{31}$
c_7	m_{12}	m_{22}	m_{32}	$2.m_{12}.m_{14}$	$2.m_{22}.m_{24}$	$2.m_{32}.m_{34}$	$m_{12}.m_{24} + m_{14}.m_{22}$	$m_{12}.m_{34} + m_{14}.m_{32}$	$m_{22}.m_{34} + m_{24}.m_{32}$
c_8	m_{13}	m_{23}	m_{33}	$2.m_{13}.m_{14}$	$2.m_{23}.m_{24}$	$2.m_{33}.m_{34}$	$m_{13}.m_{24} + m_{14}.m_{23}$	$m_{13}.m_{34} + m_{14}.m_{33}$	$m_{23}.m_{34} + m_{24}.m_{33}$
c_9	m_{14}	m_{24}	m_{34}	m_{14}^2	m_{24}^2	m_{34}^2	$m_{14}.m_{24}$	$m_{14}.m_{34}$	$m_{24}.m_{34}$

C.1.5 Résolution par intégrales itérées

En utilisant les intégrales itérées et des considérations géométriques, on montre que :

$$\int_{T_0} h(x,y,z).dv = \int_0^1 \int_0^{1-x} \int_0^{1-x-y} h(x,y,z).dz.dy.dx$$

Et tous calculs faits, on obtient :

$$\int_{T_0} h(x,y,z).dv = \frac{c_0 + c_1 + c_2}{60} + \frac{c_3 + c_4 + c_5}{120} + \frac{c_6 + c_7 + c_8}{24} + \frac{c_9}{6}$$

Les résultats précédents permettent le calcul des intégrales recherchées en utilisant un outil de calcul algébrique (les calculs sont simples, mais les résultats intermédiaires sont longs). Soit finalement :

x	$\frac{m_{11}+m_{12}+m_{13}}{24} + \frac{m_{14}}{6}$
y	$\frac{m_{21}+m_{22}+m_{23}}{24} + \frac{m_{24}}{6}$
z	$\frac{m_{31}+m_{32}+m_{33}}{24} + \frac{m_{34}}{6}$
x^2	$\frac{m_{11}^2+m_{12}^2+m_{13}^2+m_{11}.m_{12}+m_{11}.m_{13}+m_{12}.m_{13}}{60} + m_{14}.\frac{m_{11}+m_{12}+m_{13}}{12} + \frac{m_{14}^2}{6}$
y^2	$\frac{m_{21}^2+m_{22}^2+m_{23}^2+m_{21}.m_{22}+m_{21}.m_{23}+m_{22}.m_{23}}{60} + m_{24}.\frac{m_{21}+m_{22}+m_{23}}{12} + \frac{m_{24}^2}{6}$
z^2	$\frac{m_{31}^2+m_{32}^2+m_{33}^2+m_{31}.m_{32}+m_{31}.m_{33}+m_{32}.m_{33}}{60} + m_{34}.\frac{m_{31}+m_{32}+m_{33}}{12} + \frac{m_{34}^2}{6}$
xy	$\frac{m_{11}.(2.m_{21}+m_{22}+m_{23})+m_{12}.(m_{21}+2.m_{22}+m_{23})+m_{13}.(m_{21}+m_{22}+2.m_{23})}{120} + \frac{m_{14}.(m_{21}+m_{22}+m_{23})+m_{24}.(m_{11}+m_{12}+m_{13})}{24} + \frac{m_{14}.m_{24}}{6}$
xz	$\frac{m_{11}.(2.m_{31}+m_{32}+m_{33})+m_{12}.(m_{31}+2.m_{32}+m_{33})+m_{13}.(m_{31}+m_{32}+2.m_{33})}{120} + \frac{m_{14}.(m_{31}+m_{32}+m_{33})+m_{34}.(m_{11}+m_{12}+m_{13})}{24} + \frac{m_{14}.m_{34}}{6}$
yz	$\frac{m_{21}.(2.m_{31}+m_{32}+m_{33})+m_{22}.(m_{31}+2.m_{32}+m_{33})+m_{23}.(m_{31}+m_{32}+2.m_{33})}{120} + \frac{m_{24}.(m_{31}+m_{32}+m_{33})+m_{34}.(m_{21}+m_{22}+m_{23})}{24} + \frac{m_{24}.m_{34}}{6}$

C.2 Composition de Liens

Une méthode de représentation d'une articulation à trois degrés de liberté à partir de la composition de liens plus simples a été présentée en section 3.5.4. Nous allons détailler ici les calculs sous-jacents.

C.2.1 Synthèse de lien

On cherche à synthétiser un lien équivalent à un *Ball and Socket* en combinant des liens plus simples. Trois degrés de liberté rotationnels sont nécessaires, on combinera donc trois liens rotationnels à un degré de liberté (section 3.5.4). L'examen des transformations associées montre que cela ne nous fournit pas de contrôle sur un paramètre translationnel constant en y ; on ajoute donc en amont un lien sans degré de liberté (section 3.5.4) pour tenter d'y remédier. Par la suite, les paramètres associés aux différents liens seront numérotés de 0 à 3, 0 désignant le lien sans degré de liberté. On va donc étudier une composition de liens à un degré de liberté ayant les mêmes propriétés, mais dont les limites sont plus faciles à exprimer.

Les variables associées sont les angles θ_1, θ_2 et θ_3 ; les autres paramètres sont des constantes à déterminer.

Aspect Positionnel

Cette articulation doit avoir un facteur translationnel quelconque, mais constant : cela peut permettre de poser une première série de contraintes sur ce modèle. Compte-tenu des degrés de liberté disponibles, la translation résultante doit être déterminée par les quatre premières transformations ; il ne doit par contre pas y avoir de contribution des transformations suivantes du fait des trois degrés de liberté en rotation qu'ils fournissent, ce qui ne mènerait plus à une translation quelconque.

On peut donc poser $a_2 = d_2 = a_3 = d_3 = 0$ et chercher les paramètres d_0, θ_0, α_1, a_1 et θ_1 à associer à une translation arbitraire. La transformation associée est :

$$
\begin{aligned}
T &= T_z(d_0).R_z(\theta_0).R_x(\alpha_1).T_{xz}(a_1, d_1) \\
&= \begin{pmatrix}
\cos\theta_0 & -\cos\alpha_1.\sin\theta_0 & \sin\alpha_1.\sin\theta_0 & a_1.\cos\theta_0 + d_1.\sin\alpha_1.\sin\theta_0 \\
\sin\theta_0 & \cos\alpha_1.\cos\theta_0 & -\sin\alpha_1.\cos\theta_0 & -d_1.\cos\theta_0.\sin\alpha_1 + a_1.\sin\theta_0 \\
0 & \sin\alpha_1 & \cos\alpha_1 & d_0 + d_1.\cos\alpha_1 \\
0 & 0 & 0 & 1
\end{pmatrix}
\end{aligned}
$$

Le système à résoudre est donc :

$$
\begin{cases}
x &= a_1.\cos\theta_0 + d_1.\sin\alpha_1.\sin\theta_0 \\
y &= -d_1.\cos\theta_0.\sin\alpha_1 + a_1.\sin\theta_0 \\
z &= d_0 + d_1.\cos\alpha_1
\end{cases}
$$

Il existe une infinité de solutions ; on peut par exemple poser $\theta_0 = 0$ et $\alpha_1 = -\pi/2$, d'où on tire $x = a_1$, $y = d_1$ et $z = d_0$.

Aspect Rotationnel

Compte-tenu des éléments précédents, la portion rotationnelle de cette transformation est :

$$R = R_x(-\pi/2).R_z(\theta_1).R_x(\alpha_2).R_z(\theta_2).R_x(\alpha_3).R_z(\theta_3)$$

Il faut déterminer α_2 et α_3 afin que la rotation aie trois degrés de liberté et que les fonctions permettant de calculer $(\theta_1, \theta_2, \theta_3)$ pour une position donnée restent simples.

La matrice de transformation associée est trop complexe pour pouvoir être traitée directement, et il n'y a pas de correspondance simple entre les transformations de l'articulation à 3 degrés de liberté déjà décrite et celle-ci. Pour simplifier les choses, posons $\alpha_2 = \alpha_3 = \pi/2$. La matrice de transformation devient alors (coefficients dans la section suivante) :

$$R_L = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$$

Résolution Générale

Soit maintenant une rotation arbitraire représentée par un quaternion $q = (x, y, z, w)$. Peut-on toujours calculer $(\theta_1, \theta_2, \theta_3)$ tels que $R_L = R_q$?

Cela revient à poser le système suivant :

$$R_L = \begin{pmatrix} \cos\theta_3 * \cos\theta_1 * \cos\theta_2 + \sin\theta_3 * \sin\theta_1 &=& 1 - 2y^2 - 2z^2 &=& a \\ -\sin\theta_3 * \cos\theta_1 * \cos\theta_2 + \cos\theta_3 * \sin\theta_1 &=& 2xy - 2wz &=& b \\ \cos\theta_1 * \sin\theta_2 &=& 2xz + 2wy &=& c \\ \cos\theta_3 * \sin\theta_2 &=& 2xy + 2wz &=& d \\ -\sin\theta_3 * \sin\theta_2 &=& 1 - 2x^2 - 2z^2 &=& e \\ -\cos\theta_2 &=& 2yz - 2wx &=& f \\ -\cos\theta_3 * \cos\theta_2 * \sin\theta_1 + \sin\theta_3 * \cos\theta_1 &=& 2xz - 2wy &=& g \\ \sin\theta_3 * \cos\theta_2 * \sin\theta_1 + \cos\theta_3 * \cos\theta_1 &=& 2yz + 2wx &=& h \\ -\sin\theta_1 * \sin\theta_2 &=& 1 - 2x^2 - 2y^2 &=& i \end{pmatrix}$$

Restriction Angulaire

On borne θ_2 dans l'intervalle $[0, \pi]$ afin de simplifier la résolution. Dans ce cas, on tire immédiatement du système $\theta_2 = \cos^{-1}(-f)$ et $\sin\theta_2 = \sqrt{1 - f^2}$. D'où on déduit :

$$\begin{cases} \cos\theta_1 &= c/\sqrt{1-f^2} \\ \sin\theta_1 &= -i/\sqrt{1-f^2} \\ \cos\theta_3 &= d/\sqrt{1-f^2} \\ \sin\theta_3 &= -e/\sqrt{1-f^2} \end{cases}$$

Cas du *Gimbal Lock*

Pour $|f| = 1$, on se retrouve dans l'équivalent d'une situation de *gimbal lock* ; en effet, le système devient dans ce cas :

$$\begin{cases} \cos\theta_3 * \cos\theta_2 * \cos\theta_1 + \sin\theta_3 * \sin\theta_1 &= a \\ -\sin\theta_3 * \cos\theta_2 * \cos\theta_1 + \cos\theta_3 * \sin\theta_1 &= b \\ -\cos\theta_2 &= f \\ -\cos\theta_3 * \cos\theta_2 * \sin\theta_1 + \sin\theta_3 * \cos\theta_1 &= g \\ \sin\theta_3 * \cos\theta_2 * \sin\theta_1 + \cos\theta_3 * \cos\theta_1 &= h \end{cases}$$

▷ **Cas** $f = 1$

$$\begin{cases} -\cos\theta_3 * \cos\theta_1 + \sin\theta_3 * \sin\theta_1 &= a &= -h \\ \sin\theta_3 * \cos\theta_1 + \cos\theta_3 * \sin\theta_1 &= b &= g \\ -\cos\theta_2 &= -1 \end{cases}$$

Posons $\theta_1 = 0$; alors $\cos\theta_3 = h$ et $\sin\theta_3 = g$.

▷ **Cas** $f = -1$

$$\begin{cases} \cos\theta_3 * \cos\theta_1 + \sin\theta_3 * \sin\theta_1 &= a &= h \\ -\sin\theta_3 * \cos\theta_1 + \cos\theta_3 * \sin\theta_1 &= b &= -g \\ \cos\theta_2 &= 1 \end{cases}$$

Posons $\theta_1 = 0$; alors $\cos\theta_3 = h$ et $\sin\theta_3 = g$.

▷ **Cas général** Si $|f| = 1$, une solution est donc :

$$\begin{cases} \theta_1 &= 0 \\ \theta_2 &= \pi \text{ si } f = 1, 0 \text{ sinon} \\ \theta_3 &= atan2(g, h) \end{cases}$$

Continuité aux limites On voit que, quelle que soit la méthode de résolution choisie, il existe une singularité pour $\theta_2 = 0$ ou $\theta_2 = \pi$, ce qui est incontournable à partir du moment où une rotation dans l'espace est représentée par la composition de trois rotations (théorème d'Euler). Dans ces deux cas, les axes de rotation associés à θ_1 et θ_3 sont alignés. Le fait de contraindre θ_2 dans $[0, \pi]$ simplifie les choses, dans le sens où on ne peut plus "franchir" cette singularité ; on peut cependant l'atteindre.

Annexe D

Représentation des Données

Cette annexe vise à présenter le format de données utilisé pour décrire les modèles physiques produits par les algorithmes décrits au chapitre 3. Par soucis de simplicité de traitement, un format XML est utilisé. Nous fournirons tout d'abord le DTD définissant la grammaire de ce format XML, puis un exemple des données générées lors de l'exportation d'un modèle simple.

D.1 DTD des Modèles Physiques

Ceci est le DTD XML utilisé pour décrire les données exportées par notre outil d'extraction (section 3.5). Ce format de données permet de stocker à la fois un ensemble arborescent de maillages (nœuds `geomesh`) associés à leurs paramètres physiques (les nœuds `rigidbodyparams`), ainsi qu'un ensemble éventuel de poses permettant de stocker une animation (nœuds `poses`).

Une extension de ce format pourrait être l'intégration du modèle géométrique initial, afin de disposer dans un même fichier du modèle à animer (excepté peut-être ses textures, difficiles à représenter efficacement en XML) et de ses propriétés physiques. Se poserait alors le problème de la définition d'une description aussi générale que possible afin de pouvoir représenter des types de modèles variés. Cela paraît réaliste puisque, on l'a vu en section 3.2.1, les systèmes d'animation par squelette restent basés sur des mécanismes communs (hiérarchie des articulations, influence pondérée des os).

```
<!ELEMENT physicalmodel (physicalnode)>
<!ATTLIST physicalmodel rho CDATA #REQUIRED>

<!ELEMENT physicalnode (physicalmesh, poses, children)>
<!ATTLIST physicalnode id CDATA #REQUIRED>

<!ELEMENT physicalmesh (geomesh,spinshift,rigidbodyparams)>

<!ELEMENT poses (pose)*>
```

```
<!ELEMENT children (physicalnode*)>

<!ELEMENT geomesh (vertex, faces)>
<!ATTLIST geomesh name CDATA #REQUIRED>
<!ELEMENT vertex (cartesianvector)*>
<!ELEMENT faces  (triangle)*>


<!ELEMENT rigidbodyparams (mass, inertia, cgpos)>

<!ELEMENT inertia (matrix33)>

<!ELEMENT matrix33 (ml3, ml3, ml3)>
<!ELEMENT ml3 EMPTY>
<!ATTLIST ml3 c0 CDATA #REQUIRED>
<!ATTLIST ml3 c1 CDATA #REQUIRED>
<!ATTLIST ml3 c2 CDATA #REQUIRED>

<!ELEMENT cgpos (cartesianvector)>

<!ELEMENT pose (position, rotation)>
<!ELEMENT spinshift (position, rotation)>

<!ELEMENT position (cartesianvector)>
<!ELEMENT rotation (quaternion)>


<!ELEMENT cartesianvector EMPTY>
<!ATTLIST cartesianvector x CDATA #REQUIRED>
<!ATTLIST cartesianvector y CDATA #REQUIRED>
<!ATTLIST cartesianvector z CDATA #REQUIRED>

<!ELEMENT quaternion EMPTY>
<!ATTLIST quaternion x CDATA #REQUIRED>
<!ATTLIST quaternion y CDATA #REQUIRED>
<!ATTLIST quaternion z CDATA #REQUIRED>
<!ATTLIST quaternion w CDATA #REQUIRED>

<!ELEMENT triangle EMPTY>
<!ATTLIST triangle idx0 CDATA #REQUIRED>
<!ATTLIST triangle idx1 CDATA #REQUIRED>
<!ATTLIST triangle idx2 CDATA #REQUIRED>

<!ELEMENT mass EMPTY>
<!ATTLIST mass m CDATA #REQUIRED>
```

D.2 Exemple de Fichier Généré

Prenons l'exemple du modèle géométrique très simple (une flagelle) représenté en (a), figure D.1. Son squelette ne comporte que deux os, dont la composante positionnelle est représentée par un point bleu et l'orientation par un ensemble d'axes. L'os racine se trouve sur la gauche et, comme on l'a vu en section 3.2.1, il ne déforme pas le modèle mais permet son positionnement. L'os se trouvant à sa droite est son unique fils, et permet une flexion du modèle en son milieu ; l'influence de chaque os est représentée en (b).

(a) Modèle Géométrique **(b)** Influence des Os

FIG. D.1 – Modèle Géométrique Simple

FIG. D.2 – Modèle Physique Extrait

L'extraction se fait ici très simplement par découpe le long d'un unique plan. le modèle physique obtenu est représenté en figure D.2 ; les parties générées sont colorées différemment afin de les distinguer. On trouvera en fin d'annexe le fichier XML décrivant ce modèle physique. On y retrouve la définition des deux sous-parties rigides, d'index 0 et 1 (propriété `id` de `physicalnode`). Leur sont associés les maillages nommés `Refined #0` et `Refined #1` (composés de `vertex` et de `triangles`, ainsi que les paramètres physiques associés (élément `rigidbodyparams` et ses fils). Afin d'alléger la représentation, le nombre d'éléments de listes et la précision des réels ont été diminués.

Cette description de modèle est neutre vis à vis des articulations : leurs poses sont représentées via un couple vecteur/quaternion associé à chaque partie, et l'ensemble des positions valides disponibles pour chaque articulation sont stockées sous forme de liste (élément `poses`). Ce choix permet de déléguer au simulateur physique la génération des articulations et de leurs contraintes articulaires, celles-ci étant très dépendantes de l'outil de simulation utilisé.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE physicalmodel SYSTEM "physicalmodel-neutral.dtd">

<physicalmodel rho="1">
  <physicalnode id="0">
    <physicalmesh>
      <geomesh name="Refined #0">
        <vertex>
          <cartesianvector x="-8.714" y="0.362" z="1.090" />
          <cartesianvector x="-8.737" y="-0.949" z="-0.049" />
          (...)
          <cartesianvector x="-8.696" y="1.421" z="-0.117" />
          <cartesianvector x="-0.704" y="1.305" z="-0.018" />
        </vertex>
        <faces>
          <triangle idx0="1" idx1="0" idx2="2" />
          <triangle idx0="3" idx1="1" idx2="4" />
          (...)
          <triangle idx0="18" idx1="10" idx2="20" />
          <triangle idx0="10" idx1="12" idx2="20" />
        </faces>
      </geomesh>
      <spinshift>
        <position><cartesianvector x="1.741" y="-0.188" z="0" /></position>
        <rotation><quaternion x="0.007" y="0.042" z="0.025" w="0.998" /></rotation>
      </spinshift>
    <rigidbodyparams>
      <mass m="35.8773" />
      <inertia>
        <matrix33>
```

```
            <ml3 c0="30.5212" c1="44.5518" c2="-3.41845" />
            <ml3 c0="44.5518" c1="906.285" c2="0.800282" />
            <ml3 c0="-3.41845" c1="0.800282" c2="912.749" />
          </matrix33>
        </inertia>
        <cgpos><cartesianvector x="-4.510" y="0.273" z="-0.019" /></cgpos>
      </rigidbodyparams>
    </physicalmesh>
    <poses>
      <spinshift>
        <position><cartesianvector x="1.741" y="-0.188" z="0" /></position>
        <rotation><quaternion x="0.001" y="-0.024" z="-0.038" w="0.998" /></rotation>
      </spinshift>
      <spinshift>
        <position><cartesianvector x="1.741" y="-0.188" z="0" /></position>
        <rotation><quaternion x="0.003" y="-0.048" z="-0.076" w="0.995" /></rotation>
      </spinshift>
      (...)
      <spinshift>
        <position><cartesianvector x="1.741" y="-0.188" z="0" /></position>
        <rotation><quaternion x="8.460e-11" y="-1.254e-09" z="-4.142e-06" w="0.999" /></rotation>
      </spinshift>
      <spinshift>
        <position><cartesianvector x="1.741" y="-0.188" z="0" /></position>
        <rotation><quaternion x="0.001" y="-0.024" z="-0.038" w="0.998" /></rotation>
      </spinshift>
    </poses>
    <children>
      <physicalnode id="1">
        <physicalmesh>
```

```
<geomesh name="Refined #1">
  <vertex>
    <cartesianvector x="0.210" y="0.233" z="1.065" />
    <cartesianvector x="0.253" y="0.133" z="1.143" />
    (...)
    <cartesianvector x="0.234" y="-1.072" z="-0.080" />
    <cartesianvector x="-0.065" y="-0.421" z="-0.665" />
  </vertex>
  <faces>
    <triangle idx0="1" idx1="0" idx2="2" />
    <triangle idx0="0" idx1="3" idx2="2" />
    (...)
    <triangle idx0="12" idx1="13" idx2="9" />
    <triangle idx0="13" idx1="11" idx2="7" />
  </faces>
</geomesh>
<spinshift>
  <position><cartesianvector x="-8.718" y="0.150" z="0" /></position>
  <rotation><quaternion x="-0.002" y="-0.119" z="-0.122" w="0.985" /></rotation>
</spinshift>
<rigidbodyparams>
  <mass m="6.6689" />
  <inertia>
    <matrix33>
      <ml3 c0="2.04277" c1="1.28966" c2="-0.135643" />
      <ml3 c0="1.28966" c1="31.4179" c2="0.0314539" />
      <ml3 c0="-0.135643" c1="0.0314539" c2="31.4972" />
    </matrix33>
  </inertia>
  <cgpos><cartesianvector x="-1.695" y="0.089" z="0.013" /></cgpos>
```

```
            </rigidbodyparams>
          </physicalmesh>
          <poses>
            <spinshift>
              <position><cartesianvector x="-8.718" y="0.150" z="0" /></position>
              <rotation><quaternion x="0.010" y="-0.040" z="-0.048" w="0.997" /></rotation>
            </spinshift>
            <spinshift>
              <position><cartesianvector x="-8.718" y="0.150" z="0" /></position>
              <rotation><quaternion x="0.022" y="-0.082" z="-0.100" w="0.991" /></rotation>
            </spinshift>
            (...)
            <spinshift>
              <position><cartesianvector x="-8.718" y="0.150" z="0" /></position>
              <rotation><quaternion x="5.464e-10" y="-2.091e-09" z="-2.531e-09" w="1" /></rotation>
            </spinshift>
            <spinshift>
              <position><cartesianvector x="-8.718" y="0.150" z="0" /></position>
              <rotation><quaternion x="0.010" y="-0.040" z="-0.048" w="0.997" /></rotation>
            </spinshift>
          </poses>
          <children>
          </children>
        </physicalnode>
      </children>
    </physicalnode>
</physicalmodel>
```

Annexe E

Éléments d'Éthologie

E.1 Introduction

Ce chapitre est une synthèse de [40] et de [26, chapitre 50], ainsi que de [62]. Nous nous y intéressons à l'étude du comportement animal et en particulier des intérêts qu'il présente en terme d'adaptation à son environnement. Nous présentons ces éléments sous forme d'une annexe car ils ne sont pas directement exploités dans nos travaux ; leur intérêt est de fournir des possibilités de comparaison et de mise en perspective de la démarche "Animats" (chapitre 2), des courants récents associés en informatique graphique (section 1.3.6 et 1.4 en particulier) ou des perspectives que nous proposons (chapitre 4).

Un comportement animal peut être défini comme une séquence d'actions (généralement motrices) résultant de l'intégration d'informations pertinentes de l'environnement, qualifiées de *stimuli*, et de paramètres internes à l'animal. En dehors des cas les plus simples (taxies et stéréotypes, cf. section E.3.1 et E.6), on observe de plus une influence des stimuli perçus (expérience) sur les comportements futurs (mémorisation), indépendamment de toute réaction immédiate.

Après une présentation de quelques comportements typiques, nous aborderons donc les caractéristiques de la perception animale (section E.4) et des processus de sélection d'une action appropriée, pour finir par la notion d'apprentissage, modification ou acquisition de comportements du fait de l'expérience.

Nous aborderons ainsi les différents types de questionnements proposés par Tinbergen (1963, cité par [62]) pour tout comportement :

- ses causes (les stimuli et mécanismes internes en jeu),
- sa valeur adaptative,
- son évolution (*phylogenèse*),
- son développement durant la vie de l'animal (*ontogenèse*).

E.2 Origine et Acquisition des Comportements

E.2.1 Causes Immédiates, Causes Ultimes

Deux sujets d'études peuvent être abordés devant un comportement immédiat :

- la raison pour laquelle l'animal a ce comportement (son pourquoi) : il s'agit ici d'une recherche de sa *cause ultime*. Une réponse moderne est donnée par l'écologie comportementale, qui expliquera en quoi ce comportement augmente l'adaptabilité de l'animal, ce qui a favorisé sa sélection.
- les mécanismes à la base de ce comportement (son comment) : il s'agit ici de trouver sa *cause immédiate*, ou cause première. Une réponse sera ici fournie par la physiologie, et plus généralement les sciences analytiques.

Ceci peut être modélisé par un cycle : le génotype code pour un phénotype, qui est le substrat des comportements (ce sont les causes immédiates). Ces mêmes comportements, suivant leur adaptabilité, permettront la perpétuation du génotype correspondant (cause ultime).

Les causes immédiates ont un intérêt en elles-mêmes, en ce qu'elles bornent les comportements exprimables par l'animal. Elles peuvent être un sujet d'étude indépendamment de leur dimension adaptative, comme dans le cas des sciences cognitives.

E.2.2 Influence Génétique

La possibilité d'une spécification d'un ensemble de comportements via le patrimoine génétique est particulièrement claire chez les insectes, capables de comportements adaptés et complexes dès leur éclosion. Une dépendance tranchée entre des gènes et certains comportements peut souvent être exhibée, via une répartition discontinue de leur transmission (telle que prévue par les lois mendéliennes). Dans des cas plus complexes, on observe une émergence du comportement suite à l'interaction entre une série de gènes (la variation entre individus est alors continue).

Chez des espèces disposant d'une certaine souplesse de comportement, des situations plus complexes apparaissent. Une prédisposition génétique est souvent observée, par exemple dans le cas de l'acquisition d'un type de chant particulier chez les oiseaux ou de la construction du nid, qui sera optimisée par l'expérience (voir section E.2.7).

E.2.3 Phylogénie

Afin d'observer l'évolution des comportements, il est utile d'étudier les variations de ceux-ci entre espèces apparentées. L'un des phénomènes exhibés a été la *ritualisation*, suivant laquelle un comportement a été détourné d'une fonctionnalité initiale pour acquérir un rôle de transmission d'information. Ainsi, on observe des éléments significatifs en terme de registre comportemental dans les comportements de cour chez les oiseaux.

D'une espèce à l'autre, on observe par exemple des variations dans la fréquence de certains comportements, ou d'autres paramètres quantitatifs : cela permet par exemple, par sélection artificielle, de modifier en quelques générations le chant de crickets mâles.

Notons que des comportements apparemment gratuits (au sens où ils ne permettent pas une adaptation à des contraintes environnementales) peuvent être sélectionnés du fait de leur

influence sur les stratégies de sélection de partenaires, par exemple.

E.2.4 Stratégies Stables Evolutivement

La notion de stratégie stable évolutivement (*evolutionarily stable strategie*) est due à Maynard Smith (1982, d'après [62]). Dans ce cadre, une stratégie est un comportement animal, et une ESS est telle que, une fois adoptée par la majorité de la population, elle est incontournable, c'est-à-dire qu'aucun comportement n'est alors meilleur. Cette notion permet d'expliquer entre autre les stratégies d'attaque ou de fuite, celles-ci étant très dépendantes de la stratégie adverse ; le fait qu'une même stratégie soit retenue par tous permet alors de l'optimiser.

E.2.5 Maturation

Il a été décrit que les comportements stéréotypés peuvent être améliorés avec le temps ; il est cependant important de noter qu'il ne s'agit pas nécessairement d'un apprentissage. Dans le cas de l'"apprentissage" du vol par les oiseaux, par exemple, des mouvements caractéristiques sont observés chez les oisillons. Cependant, l'acquisition du vol est surtout due à la mise en place des structures neurales et musculaires indispensables. En effet, un oiseau immobilisé durant tout son développement s'envole malgré tout normalement, sans expérience préalable, une fois l'âge caractéristique atteint. Ce type de situations est désignée par le terme de *maturation*.

Notons que dès son plus jeune âge, un animal doit être fonctionnel, même si toutes ses capacités ne se sont pas encore développées. Souvent, les comportements évoluent durant son développement pour tenir compte de ses nouvelles aptitudes, de l'évolution de sa taille, de modifications éventuelles de son environnement... Dans le cas des insectes se métamorphosant, l'animal possède deux jeux complets de gènes contrôlant deux programmes de développement en interaction avec deux environnements différents.

E.2.6 Transmission Culturelle

En parallèle à une transmission génétique des comportements, présentant l'avantage d'une possibilité d'optimisation en parallèle à la physiologie de l'animal du fait des mécanismes de sélection, un comportement peut être appris des parents ou du groupe. C'est aussi une forme de transmission, puisque l'animal une fois adulte pourra lui aussi servir de modèle à des jeunes.

Ce type d'acquisition nécessite une capacité importante à modifier son comportement ; il n'est donc pas étonnant de retrouver des exemples fréquents de ce type chez les primates, pour l'utilisation d'outils en particulier. Mais d'autres classes de vertébrés peuvent aussi exhiber cette capacité, les oiseaux en particulier : transmission du chant chez de nombreuses espèces, perçage de bouteilles de lait chez la mésange. A ce propos, cette transmission culturelle ne se fait pas uniquement par imitation, les oiseaux pouvant apprendre ce comportement par la simple observation de capsules percées. Un autre type de transmission culturelle est la reconnaissance du danger chez les oiseaux ou mammifères, orientée par les cris d'alarme des adultes.

Ce type d'acquisition est un outil efficace pour l'adaptation des animaux, leur permettant une modification des comportements alimentaires ou de fuite en fonction de leur environnement, et ce de manière plus efficace que les essais et erreurs individuels et plus rapide qu'une

évolution génétique.

E.2.7 Controverse Inné-Acquis

La confrontation de l'observation des comportements stéréotypés (adaptés uniquement via des mécanismes évolutifs) par les éthologistes et l'étude des capacités d'apprentissage par la psychologie animale a mené à une controverse concernant l'importance prédominante de l'inné ou de l'acquis dans le comportement animal. Les positions se sont assouplies, de par la constatation que les comportements stéréotypés complexes, bien qu'en germe dans le comportement du jeune, nécessitent une pratique pour être optimalement maîtrisés, et du fait que les apprentissages les plus complexes se basent, en dernière analyse, sur les capacités offertes par un système nerveux d'origine génétique.

Interation, et non Opposition

Notons cependant à ce propos que l'expression du génotype en phénotype ne saurait être considérée comme directe, et que l'influence de l'environnement, que ce soit en terme de matière et d'énergie (développement physique correct) ou d'information (structuration du système nerveux) est capitale. C'est donc bien du jeu des interactions entre un génotype et un environnement (au sens large) auquel il est adapté qu'émerge l'individu. Un comportement ne saurait donc être caractérisé sans référence à l'histoire de l'animal ; même les plus stéréotypés nécessitent que l'animal se développe dans un environnement convenable, et sont souvent modifiables durant la vie de l'animal (en cas d'altération d'un sens par exemple).

Pour prendre un exemple extrême [26, p. 1166], "l'utilisation d'une langue en particulier constitue une variation comportementale entièrement apprise d'une capacité innée de l'Humain".

Tout comme les instincts peuvent s'adapter à l'environnement, les apprentissages peuvent s'avérer aussi stéréotypés que des comportements innés : Lorenz a montré comment, une fois un itinéraire appris, les musaraignes d'eau ne le modifient plus, continuant par exemple à contourner un obstacle qui n'existe plus.

Les instincts ne peuvent être définis comme des comportements exhibés sans expérience préalable, l'immense majorité d'entre eux ne pouvant se développer dans l'isolation complète. Par exemple, les systèmes sensoriels ne peuvent souvent se développer normalement sans l'écoute de sons spécifiques à l'espèce ou la vision d'un environnement varié.

Ainsi, la présence de stéréotypes chez tous les individus d'une espèce ne saurait nous renseigner sur leur seule origine génétique : le fait que tous se soient développés dans le même environnement est un paramètre fondamental.

Optimisation de l'Inné

Des capacités d'apprentissage peuvent permettre de moduler et d'adapter des stéréotypes. Un exemple en est donné par l'hybridation de deux espèces de perruches utilisant des techniques différentes pour la construction du nid, ce qui aboutit à des individus dans un premier temps inefficaces, car combinant les deux techniques. Ils apprennent cependant par la suite à

sélectionner l'une des conduites motrices, et retrouvent une certaine productivité bien qu'elle sera toujours parasitée par des artefacts originaires de l'autre stratégie [26, p. 1160].

E.3 Types de Comportements

Nous allons décrire un certain nombre d'activités caractéristiques des animaux ; les éléments sous-tendant ces comportements (perception, apprentissage...) seront détaillés dans les sections suivantes.

E.3.1 Locomotion

La possibilité de se déplacer est une des caractéristiques marquantes du règne animal. Via la coordination de séquences motrices adaptées, elle permet à l'animal toute une série de comportements tels la recherche d'aliments, de partenaires ou d'abris.

Dans les cas les plus simples, telles la cinèse ou les taxies, il s'agit essentiellement pour l'animal de rejoindre le milieu qui lui est le plus favorable ou de s'y maintenir (cause ultime) via une modulation de l'activité en réaction à des stimuli locaux (cause immédiate). Cependant, un rapport beaucoup plus complexe à l'espace peut exister, ainsi que les capacités d'orientation en témoignent. Les sections suivantes détaillent ces processus.

Cinèses

La *cinèse* est l'une des formes les plus élémentaires de réaction à son environnement. Elle consiste à moduler son degré d'activité en fonction d'une variable environnementale, mais sans orientation particulière (cause immédiate).

Les cloportes, par exemple, voient leur activité (locomotrice en particulier) augmenter en milieu sec, pour se limiter en milieu humide. Ainsi, l'animal aura tendance à quitter les zones sèches et à rester dans les zones humides, qui lui sont plus favorables (cause ultime).

Tropismes, Taxies

Une forme de réaction simple à son environnement est le *tropisme* végétal, dans lequel la croissance d'une plante est orientée par un paramètre physique tel la luminosité ou la gravité. Son équivalent animal est la *taxie* ; elle est surtout observable chez les organismes les plus primitifs, et dans certains comportements des insectes (fuite des zones éclairées), bien qu'elle puisse faire partie du registre comportemental d'animaux plus élaborés (les truites s'orientent par exemple toujours vers l'amont, afin de ne pas être entraînée par le courant). Notons la plus grande complexité de ces comportements par rapport aux cinèses, dans lesquelles le gradient du stimulus n'est pas pris en compte.

Orientation dans l'Espace

Des animaux, et les oiseaux en particulier sont capables d'utiliser toute une série d'informations pour retourner à leur nid (*homing*) ou se déplacer entre deux sites (butinage, migrations).

Ils peuvent pour cela se baser sur le champ magnétique terrestre ou leur olfaction (saumons, pigeons voyageurs) : il y a alors mémorisation (comparable au phénomène d'empreinte dans le cas des saumons, retournant pondre à leur lieu de naissance). Par ailleurs, certains peuvent utiliser la direction du soleil même par temps couvert, en utilisant la polarisation de sa lumière (abeilles). Enfin, les oiseaux migrateurs se basant sur la position du soleil, lorsqu'il est visible, semblent utiliser une horloge interne pour décaler leur position en fonction de l'heure (ce qui est rendu nécessaire par la trajectoire apparente du soleil).

Notons que dans le cas du *homing*, la connaissance d'une direction absolue n'est pas d'une grande utilité. Il y a probablement intégration des déplacements de l'animal lors de son déplacement initial, ce qui lui permet de partir approximativement dans la bonne direction une fois relâché : le retour de l'animal à son nid n'est jamais direct (et moins probable hors de la période de soins aux jeunes : il dépend donc de sa motivation).

Il est en tout cas possible de classifier ces comportements suivant trois degrés de complexité :

- le *pilotage* consiste à se rendre d'un repère familier à un autre, et est réservé à de courts trajets dans des environnements contrastés compte-tenu des capacités sensorielles de l'animal ;
- l'*orientation* consiste en un repérage de son orientation absolue, que ce soit grâce au soleil, au champ magnétique terrestre...
- la *navigation* consiste en une capacité d'orientation couplée à la capacité de repérer sa position (via des points de repère, tout comme dans le cas du pilotage).

La mémorisation d'éléments de l'environnement n'est pas réservée aux oiseaux et mammifères. Par exemple, certaines femelles de guêpes (*philantus triangulum*) creusent un nid dans le sol afin de protéger leurs larves. Tinbergen a fait l'expérience de placer durant cette étape une série de cônes de pin autour de l'entrée. Lorsque la guêpe quitte le nid juste terminé, elle effectue au-dessus des vols concentriques, puis s'éloigne afin de chasser. Dans l'intervalle, l'expérimentateur déplace le cercle de cônes : de retour avec une proie, la femelle se dirige alors directement vers le centre du cercle, alors que le nid ne s'y trouve plus. Cette espèce est donc capable de mémoriser certaines caractéristiques de son environnement afin de s'y orienter.

E.3.2 Alimentation

La nécessité de s'alimenter est un exemple d'interaction avec le milieu (niches écologiques disponibles) aboutissant à la sélection d'une morphologie et d'un comportement adapté.

Il est possible de distinguer entre les animaux très spécialisés de ce point de vue, ce qui leur permet un excellent rendement sur le petit nombre de sources alimentaires qu'ils exploitent, et les animaux plus généralistes, voire omnivores moins rentables mais pouvant s'adapter à la disparition d'un type d'aliment.

Dans tous les cas, l'animal dispose d'une *image d'appétence* guidant sa recherche alimentaire ; dans le cas des omnivores, elle sera fixée sur le type d'aliment le plus fréquemment rencontré, d'où une efficacité d'exploitation comparable à celle d'un spécialiste (morphologie exceptée), mais sera modifiée si nécessaire en fonction des approvisionnements disponibles.

La section E.5.5 détaille les stratégies mises en place dans le cadre du comportement alimentaire.

E.3.3 Communication

La communication entre les animaux se définit comme un échange intentionnel d'information entre un *émetteur* et un *récepteur*. Il s'agit en fait de stimuli fournis par d'autres animaux. Lorsque des comportements ont été sélectionnés spécifiquement pour déclencher des réponses adaptées chez d'autres animaux, on parle de *signaux*. Tout comme dans le cas des stimuli, il est possible de considérer qu'il y a eu communication lorsque le comportement du récepteur est altéré suite à cet échange.

Définition

Il est difficile de définir simplement la communication : elle est certainement caractérisée par un échange d'informations entre animaux, mais si on l'étend à toute situation de ce type, il faudra considérer que la fuite d'une proie devant son prédateur est un cas de communication. A l'inverse, restreindre cette définition aux échanges internes à une espèce ôte la possibilité d'y intégrer l'affichage de la toxicité de certains insectes via leur couleur, ce qui est pourtant la communication manifeste d'une information à un prédateur éventuel. Burghardt (1970, cité par [62]) propose à ce propos l'exemple des phéromones utilisées par les fourmis pour marquer l'itinéraire entre des sources de nourriture et le nid. Il s'agit certainement d'une communication entre fourmis ; cependant, un petit serpent utilise ces traces pour trouver le nid et dévorer leurs larves. On ne peut dire ici que les fourmis ont communiqué avec le serpent, cependant.

Il y a finalement communication lorsque les deux animaux considérés sont mutuellement adaptés pour cela : les problèmes occasionnels dus à l'exploitation de ces signaux par des prédateurs sont alors compensés par les avantages usuels de ces échanges (fleurs et insectes, par exemple).

Médias

Les médias permettant la communication entre animaux sont variés : odeurs (production de *phéromones*), sons, messages visuels..., leur sélection dépendant de leur mode de vie et de leur environnement spécifiques, ainsi que de leurs modalités sensorielles : un animal s'orientant beaucoup visuellement aura tendance à communiquer aussi par ce biais. Ainsi, les signaux ne sont généralement pas spécifiquement évolués, mais sont issus de comportements existant par ailleurs dans le registre de l'espèce.

Fonctions

Les messages peuvent avoir des effets variés chez les membres de la même espèce :

– Déclenchement ou pilotage de comportements, dans le cas par exemple de l'attraction de mâles par une phéromone ; ce sont alors de simples informations sur l'état actuel de leur émetteur. Souvent, l'effet est inné et comparable aux taxies (cf. section E.3.1), par exemple dans le cas de la modulation du déplacement des fourmis les guidant vers des sources de nourriture. Il peut cependant varier suivant son récepteur : le chant des crickets attire les femelles mais éloigne les rivaux.

– Transmission d'information, dans le cas du langage des abeilles : via des mouvements codifiés, l'abeille ayant découvert une zone intéressante peut en indiquer la proximité (danse en cercle), voire la direction et la distance si elle est plus éloignée (danse en 8). Cette information est souvent couplée à des régurgitations de nectar stimulant les spectatrices et précisant le type de source à rechercher. De même, les singes vervet, mais aussi certains écureuils ou les poules, communiquent non seulement la présence d'un danger, mais aussi le type de prédateur perçu, ce qui modifie leur comportement de manière adaptée (observation du ciel ou du sol). Notons qu'il s'agit dans les deux cas d'une communication symbolique, dans le sens où le codage utilisé pour transférer l'information est arbitraire.

Dans certains cas, les signaux peuvent modifier le sens d'actes ultérieurs ; on parle alors de *métacommunication*. Ils sont préliminaires aux jeux. Les lions ou les chiens peuvent ainsi indiquer aux jeunes qu'ils sont disposés à jouer via une posture spécifique. Les singes utilisent une mimique spécifique dans le même but.

Apprentissage : le Chant des Oiseaux

Dans le cas des oiseaux et mammifères, la communication sonore est souvent apprise par les jeunes, et ce par imitation des sons de l'environnement. Le fait de s'entendre leur permet une maîtrise progressive des modulations propres à leur espèce. Le chant des oiseaux et la parole humaine sont deux des exemples les plus marquants de cette capacité à utiliser des sons complexes. Une interaction est constatée entre des capacités et composantes innées et une influence de l'environnement et de l'exercice, indispensables à leur complexification et leur maîtrise. Dans le cas du chant, les jeunes doivent dans un premier temps mémoriser les chants d'adultes, suivant un mécanisme similaire à une empreinte (une période critique est discernable), puis les pratiquer afin de les maîtriser (il ne s'agit pas d'un phénomène de maturation, l'animal devant s'entendre pour que son chant soit normal). Une prédisposition innée à acquérir le chant de son espèce est souvent constatée, excepté dans le cas où des interactions sociales sont possibles avec un mâle d'une autre espèce, cette expérience ayant alors plus de poids que la tendance innée à ne pas le reproduire. La durée de la période critique est de plus allongée dans le cas de la présence d'un adulte par rapport à la simple exposition à des enregistrements de son chant.

D'autres oiseaux peuvent apprendre à chanter dans l'isolement total ; d'autres enfin peuvent reproduire tout son auquel ils sont exposés, qu'il s'agisse ou non d'un chant d'oiseau. A l'extrême, certains même peuvent apprendre à chanter sans possibilité de s'entendre. La variété de ces modalités d'acquisition est considérée comme une indication du fait que les mécanismes d'apprentissage du chant forment un système flexible, sensible aux pressions de sélection.

Communication et Société

Lorsque la communication est utilisée à des fins sociales, elle est fortement influencée par la position sociale relative des communiquants. De manière générale, plus un individu est haut dans la hiérarchie et plus les signaux qu'il utilisera seront minimaux ; cela suffira cependant en général, ses dominés y étant très réceptifs. Ainsi, une observation ne suffit pas dans ce cas et

une connaissance de l'histoire du groupe est nécessaire pour comprendre les interactions entre ses membres.

E.3.4 Reproduction et Soins aux Jeunes

Ces comportements impliquent une interaction entre animaux de la même espèce ; ils sont caractérisés par un entrelacement d'interactions entre leur état interne et des stimuli variés, nécessaires à une coordination adaptée des individus.

Comportement Reproducteur

Pour beaucoup d'espèces, les interactions entre individus sont limitées au maximum, pour des raisons de compétition pour les ressources en particulier. Pour aller à l'encontre de ce comportement d'évitement, des rituels de *parade nuptiale* sont souvent utilisés. Il s'agit d'un enchaînement d'actions stéréotypées (cf. section E.6), déclenchées mutuellement par les partenaires, d'où une séquence globale coordonnée confirmant les dispositions respectives des animaux. Cet enchaînement n'est pas nécessairement restreint à un couple : il peut être l'occasion d'une sélection de partenaire (*sélection sexuelle*), en général par la femelle.

Instinct Maternel

Ce comportement est déclenché par une interaction complexe de stimuli propres à la mère (état physiologique) et externes (odeur des petits, contact avec les oeufs, présence du mâle) ; ceux-ci induisent des modifications précises des hormones maternelles, ce qui influe sur ses réactions.

Les comportements s'enchaînent ainsi logiquement : un premier état hormonal mène à des comportements de cour, ce qui mènera à une fécondation éventuelle qui modifiera l'état interne des parents, d'où par la suite des comportements de couvaison par exemple. Cette double dépendance entre les états internes et des stimuli externes se retrouve chez les oiseaux et les mammifères tout particulièrement.

En tout cas, dans le cas des soins aux jeunes prodigués par les rates par exemple, un strict déterminisme biologique ne suffit pas à expliquer leur attitude ; l'expérience les rend adaptées, puis plus efficaces. De plus, ces comportements peuvent être constatés hors de toute influence hormonale, envers des jeunes d'autres femelles par exemple. Une influence du rang de la mère sur la durée des différentes étapes de soin au jeune est par ailleurs constatée chez les primates, par exemple.

E.4 Perceptions

E.4.1 Récepteurs

Les récepteurs concernés par un stimulus donné s'avèrent souvent d'une grande spécialisation. Les vairons paniquent par exemple lorsque le sang d'un de leur congénère est présent dans l'eau où ils se trouvent ; ils réagissent beaucoup moins à celui d'une autre espèce. De

même, le mâle de la mite est extrêmement sensible aux odeurs produites par la femelle, mais relativement peu aux autres substances.

E.4.2 Traitement de l'Information

Dans le cas de l'attirance des crapauds pour les petits objets et de leur crainte des grands, il est possible de montrer une activité de cellules nerveuses ganglionnaires corrélée à la perception de ce type de stimuli. Certaines d'entre elles se répartissent des zones de la rétine, ne s'activant que si celle-ci est stimulée, et étant inhibées si des zones proches sont actives. Un objet suffisamment petit pour que son image soit entièrement projetée dans le centre d'une de ces zones provoquera une excitation maximale de la cellule associée, alors que s'il est plus grand le signal sera beaucoup plus faible de par l'inhibition. Des classes distinctes de cellules existant, chacune caractérisée par l'angle de sa zone active, le crapaud dispose ainsi d'un système de détection simple des tailles : celles à petit angle détectent les proies, alors que celles à grand angle détectent les prédateurs.

Notons qu'il serait incorrect d'en conclure que l'on a trouvé chez le crapaud un neurone associé au concept ou symbole de proie, et l'autre à celui de prédateur : les cellules ganglionnaires rétiniennes n'extraient que des caractéristiques simples de l'environnement, à charge pour le cerveau de les interpréter [57]. L'étude de ce type de corrélations entre les stimuli et leur base neurophysiologique est effectuée par la neuroéthologie.

E.4.3 Stimulus

Un stimulus est finalement défini comme une information détecté dans le milieu et ayant une incidence sur le comportement de l'animal, que ce soit directement (réaction) ou par une modification de ses tendances (crainte par exemple), c'est-à-dire une altération des comportements à venir, pas nécessairement détectable dès la perception du stimulus. En tout cas, une réaction immédiate à un stimulus n'est effective que dans des situations simples : on ne saurait en conclure que les animaux se contentent systématiquement de répondre à des stimuli de ce type. Ils peuvent intégrer des informations variées, et y réagir de manière complexe (voir à ce propos la section suivante).

Les stimuli immédiats (*trigger*) sont ainsi souvent distingués de ceux d'excitation et d'orientation ; cependant, cela n'est pas toujours évident en pratique, un stimulus pouvant présenter plusieurs de ces caractéristiques.

E.5 Prises de Décisions

E.5.1 Définitions

Une fois un stimulus ou un signal perçu, un animal va par définition y réagir. Il peut s'agir d'un comportement stéréotypé, auquel cas l'association stimulus-réaction sera immédiate ; mais souvent, la réaction est plutôt une intégration de toute une série de stimuli antérieurs. La *prise de décision* (*decision-making*) qualifie ainsi les processus expliquant les causalités sti-

mulus - action, et n'implique donc aucune conscience de la situation contrairement à son usage habituel.

Le comportement sélectionné à un moment donné doit être adapté à la situation et aux besoins de l'animal ; ces facteurs internes sont qualifié de *motivation*, encore une fois sans que cela doive être interprété de manière anthropocentrique.

Une variabilité dans les réactions des animaux aux stimuli peut souvent être constatée. Elle peut être déterminée génétiquement, comme on peut l'observer en croisant des populations, ou due à l'expérience individuelle de l'animal. Ces deux facteurs ne sont d'ailleurs pas incompatibles, l'apprentissage pouvant par définition modifier des tendances innées.

E.5.2 Complexité

Le problème de la prise de décision est complexe, car les conflits entre motivations sont fréquents : par exemple, un moineau hésitera à s'approcher d'une source de nourriture si elle est proche d'un prédateur. Cependant, ce choix est variable suivant la situation : il sera beaucoup plus enclin à venir se nourrir si d'autres de ses congénères sont déjà présents, profitant ainsi de la protection du groupe. Ceux-ci émettent à cet effet un cri de ralliement spécifique, lui-même modulé par des facteurs variés. Il sera par exemple beaucoup plus fréquent lorsqu'il fait froid, la nécessité de profiter de toutes les sources de nourriture disponibles requérant le plus de sécurité possible. De même, un moineau criera beaucoup plus si la source de nourriture est divisée que si elle n'est pas partageable. Cet exemple montre bien l'ensemble des paramètres à intégrer : on ne saurait réduire le comportement de l'oiseau à une simple réponse à la présence de nourriture.

E.5.3 Notion d'Inhibition

Le concept d'*inhibition* est fondamental dans l'étude du comportement animal. Il est présent à de multiples niveaux : nous l'avons déjà rencontré en neurologie, où un neurone peut inhiber l'activité d'un autre. Au niveau comportemental, des mécanismes similaires sont à l'oeuvre entre comportements concurrents : un seul pouvant être effectué simultanément, l'exécution de l'un inhibe l'autre. À un niveau intermédiaire, toute une série de mécanismes d'inhibition existe au niveau moteur ; ainsi, non seulement les muscles antagonistes s'inhibent-ils mutuellement lors de la flexion ou de l'extension d'un membre, ce qui améliore la précision du mouvement, mais les muscles de membres opposés font aussi de même, ce qui facilite la locomotion (Sherrington, d'après [62]). La manière dont des comportements et réflexes simples se coordonnent pour mettre en place un comportement complexe telle la locomotion, ainsi que précisé en E.6.4, est ici apparente.

Sherrington décrit les comportements comme étant en compétition, et voit les mécanismes d'inhibition comme une nécessité pour le passage de l'un à l'autre. Il est ainsi possible de considérer que les systèmes gérant des comportements complexes tels la nutrition, le repos, etc. sont en compétition pour le contrôle de la musculature de l'animal ; ceux-ci étant incompatibles car ne pouvant avoir lieu simultanément (puisqu'ils mobilisent les mêmes groupes moteurs), des mécanismes de sélection sont donc bien nécessaires.

Types d'Inhibitions

▷ **Tous Sauf Un** Dans ce cas, une seule activité est menée, toutes les autres étant suspendues. C'est le cas par exemple de l'immobilisation d'un oiseau face à un prédateur. Un autre exemple est celui de certaines limaces de mer (*pleurobranchaea*) dont le comportement de fuite inhibe toute autre activité, mais aussi dont le comportement alimentaire est inhibé par la présence de ses oeufs (afin qu'elle ne s'en nourrisse pas). Dans ce cas simple, il est possible de construire un graphe représentant la hiérarchisation de l'ensemble des comportements de l'animal [62, p 78].

De même, une série de neurones géants nommés LGIs (*lateral giant interneurons*) peuvent être exhibés dans les segments de l'abdomen de langouste. Comme on peut le voir en les stimulant artificiellement, ils ont pour fonction de déclencher le comportement de fuite (une contraction intense de l'abdomen), en parallèle à l'inhibition de l'ensemble des comportements moteurs pouvant interférer avec celui-ci. Dans ce cas extrême, il semble même qu'ils gèrent l'intégration des stimuli concernés, donc la décision de déclencher ou non ce comportement.

▷ **Réciprocité** Encore une fois dans le cas de *pleurobranchaea*, une composante du réflexe de fuite consiste en la rétraction de son appendice buccal en cas de contact ; ceci inhibe par ailleurs tout comportement alimentaire. *A contrario*, le réflexe de rétraction est inhibé lors de la prise de nourriture : ainsi, ces deux comportements s'inhibent mutuellement.

▷ **A Priorités Variables** La plupart des animaux ne se contentent pas d'exécuter une seule action tant que son but n'est pas atteint : cela supprime toute possibilité d'opportunisme, c'est-à-dire d'adaptation aux fluctuations de l'environnement. Ainsi, une évolution des priorités au cours du temps est souvent observée plutôt qu'une hiérarchie figée. Il existe par exemple une inhibition réciproque entre les comportements de cour et de respiration chez le mâle de la salamandre, d'où des allers et retours entre la femelle et la surface.

Un avantage de ces inhibitions est une certaine permanence des comportements sur le court terme, qui évite à l'animal d'"osciller" entre, par exemple, un point d'eau et une source de nourriture, ce qui est une dépense d'énergie et une exposition aux prédateurs inutiles. Au contraire, l'animal aura tendance à continuer à boire un certain temps avant de se nourrir, par exemple, même si cela se fait plus souplement que dans le cas d'une inhibition complète des autres comportements.

E.5.4 Résolution Retardée

Un animal ne peut pas toujours exécuter immédiatement une action le motivant. D'une part, son objectif peut être inaccessible lors de sa décision (cas d'une nourriture inaccessible, par exemple). L'animal est alors "contrarié" (*thwarted*). D'autre part, il peut manquer d'information pour prendre une décision, comme lors de l'évaluation d'un rival pendant laquelle il peut hésiter entre la fuite et l'affrontement. Dans ces situations, l'exécution effective d'une action peut n'avoir lieu que quelques temps après qu'elle ait été décidée.

Un *effet de rebond* est souvent observé dans ce cas : le seuil de déclenchement du comportement restreint est alors beaucoup plus bas, et son intensité plus importante quand il est enfin

exécuté.

E.5.5 Modélisations

Il est intéressant de tenter de prédire ou d'expliquer les réactions des animaux aux stimuli extérieurs ; pour cela, un certain nombre de modèles ont été élaborés. Nous présenterons tout d'abord une série de notions utiles à ce type d'explications, puis quelques modèles à proprement parler. Notons en tout cas que le succès d'un modèle nous informe sur les *principes* de fonctionnement du système décisionnel d'un animal, mais absolument pas sur les *moyens* mis en oeuvre pour cela.

Concepts

Bien qu'en dernière analyse tout comportement soit réductible à une série d'interactions au niveau neural, hormonal et musculaire, la complexité de la physiologie de la majorité des animaux est telle que seuls les comportements les plus basiques sont descriptibles ainsi. En pratique, il n'est donc pas possible d'avoir recours à des explications de ce type, et l'on utilise souvent des notions abstraites, mais opératoires. C'est le cas des concepts de but et de motivation.

▷ **But** Un comportement est rarement atomique, mais se décompose plutôt en une série de tâches plus simples. Le choix du passage de l'une à l'autre au moment le plus adapté est nécessaire à un accomplissement efficace de la tâche globale. L'idée de *but* est souvent utilisée pour expliquer les comportements ; il se définit comme la situation terminant une séquence d'actions. Ce peut être un paramètre simple, comme la profondeur du nid chez les guêpes construisant des nids souterrains. Dans le cas de comportements complexes, on peut souvent définir un but global, et le diviser en sous-buts expliquant la séquence de tâches effectuées. Notons qu'il est peu utile de se demander si effectivement l'animal a un but, au sens où nous l'entendons, ou s'il fait "comme si". L'intérêt de l'explication par but est sa concision, surtout dans le cas où l'animal peut utiliser une variété de comportements pour atteindre son objectif, qui lui est par contre unique.

▷ **Motivation** La notion de motivation désigne une variable qui, en interaction avec d'autres dans un modèle décrivant leur évolution, détermine les comportements sélectionnés par l'animal. Elles peuvent aussi modifier son seuil de sensibilité à un ensemble de stimuli fonctionnellement liés : ainsi, une appétence augmentée influera sur tous les comportements en rapport avec l'alimentation. Il n'est en général pas possible de rapprocher ces variables de mécanismes physiologiques, mais ce sont cependant des outils d'explication puissants. Les modifications motivationnelles sont distinguées de la maturation et de l'apprentissage : contrairement à ces derniers, elles ont généralement lieu sur de courtes périodes de temps et sont réversibles.

Un exemple en est donné par Maynard Smith et Riechert (1984, d'après [62, p. 82]), qui ont ainsi représenté le comportement d'une araignée du désert via deux variables, la peur et l'agression. L'évolution de ces variables est corrélée aux caractéristiques de l'animal et de ses opposants, ainsi qu'au contexte (qui est sur son territoire en particulier). Une araignée combat

alors si son agressivité est largement supérieure à sa peur ; elle fuit sinon. Ce modèle pourtant simple rend bien compte de la réalité.

Des comportements fonctionnellement liés peuvent aussi parfois être partitionnés, puis corrélés à des variables (souvent antagonistes). Ainsi, la prédominance d'une variable augmentera la probabilité qu'un comportement de son ensemble associé soit effectué.

Notons que modéliser le comportement via une série de variables indépendantes et concurrentes n'est pas nécessairement suffisant : par exemple, certains stimuli provoquent une augmentation de la sensibilité à une variété de stimuli, et sont ainsi interprétables comme une augmentation de la "motivation générale". La question de la spécialisation ou de la généralité de la motivation n'est pas tranchée, mais il est clair que des interactions ont lieu entre ces variables.

Modèle de Lorentz

Lorenz a proposé un modèle hydromécanique de l'activité instinctive (modèle de double quantification). L'état motivationnel y est représenté comme un liquide s'accumulant dans un réservoir, et l'intensité du stimulus comme le taux d'ouverture d'une vanne le vidant ce qui, au-delà d'un certain débit, déclenche l'action. Ainsi, une faible motivation couplée à un fort stimulus par exemple peuvent malgré tout déclencher une réaction. Ce modèle ne reflétant qu'approximativement la réalité, il n'est plus accepté aujourd'hui (se référer à [62, p. 86] pour plus de détails). En particulier, il est à boucle ouverte, alors que la plupart des comportements animaux sont régulés par des *feedbacks*.

Modèles Homéostatiques

Ces modèles, contrairement à celui de Lorentz, rendent compte du fait que les motivations d'un animal sont influencées par ses actions. Ils stipulent que les comportements déclenchés par l'animal le sont pour rapprocher certaines de ses variables internes de valeurs d'équilibre, suivant un *feedback* négatif classique en cybernétique (boucle fermée). Cette variable peut être mesurable, dans le cas par exemple de la température chez les homéothermes, ou être plus abstraite et se rapprocher du concept de motivation, dans le cas des comportements sexuels ou d'agression. Notons que dans ce dernier cas, certains considèrent qu'il ne s'agit plus d'homéostasie, alors que d'autres n'y voient pas de différence de nature, l'état interne de l'animal déclenchant dans tous les cas le comportement (McFarland, 1971, d'après [62]).

Ces modèles correspondent bien à la réalité. Cependant, il serait faux de considérer que l'homéostasie se réduit à des comportements déclenchés lorsque des variables quittent leur valeur d'équilibre. Dans le cas de la régulation du taux de fluide dans le sang, par exemple, l'une des difficultés est que le retour dû à l'ingestion de liquide n'est pas immédiat : l'animal ne peut se contenter de boire jusqu'à ce que ses taux soient rétablies, ce qui l'amènerait à ingérer de bien trop grandes quantités de liquide.

Dans ce cas, en plus des détecteurs de déshydratation déclenchant la recherche de liquide, l'ingestion s'avère stoppée par d'autres détecteurs situés entre autres dans la bouche et l'œsophage, et donc déclenchés bien avant que les premiers effets physiologiques soient détectables. De plus, les animaux peuvent aussi anticiper une déshydratation, ce qui les amène à boire pen-

dant qu'ils s'alimentent, et ce alors même qu'aucune influence sur les fluides internes n'est perceptible durant cette ingestion, ce que Mc Farland (1971, d'après [62]) qualifie de *feed-forward*.

Ici encore, le comportement de l'animal n'est pas déterminé directement par un ou quelques facteurs, mais par une intégration de toute une série de paramètres pertinents. Par exemple, le comportement alimentaire n'est pas dirigé uniquement par les besoins mais aussi par la disponibilité de la nourriture, la présence de prédateurs, de congénères... L'observation d'animaux ayant subit des lésions cérébrales confirme le fait que la notion de faim par exemple est en fait l'intégration d'une série de variables physiologiques, toutes n'évoluant pas de la même manière.

Recherche d'Optimalité

La notion de *stratégie d'alimentation optimale* se réfère au fait que les animaux optimisent leur comportement alimentaire (choix de l'image d'appétence en particulier) en fonction d'un gain global, prenant en compte la valeur énergétique de l'aliment (taille et type de la proie, par exemple) mais aussi des besoins nutritionnels spécifiques, l'énergie nécessaire à son repérage, à sa capture et à son ingestion, l'exposition aux prédateurs, le coût en terme de temps...

La théorie d'exploration optimale (*optimal foraging theory*) stipule ainsi que les animaux choisissent leurs sites d'exploration et leurs proies afin de maximiser leur gain énergétique global. Cette théorie est caractéristique du fait que les animaux se comporte "comme si" ils utilisaient ce modèle à gain maximal (se référer à [62, p 75]). Ce type de situation est courant dans le monde animal, et ne signifie aucunement que l'animal ait une quelconque "conscience" des quantités sous-jacentes, tout comme nous n'avons pas besoin de connaître les lois de la mécanique pour rattraper un objet au vol.

Notons que cette capacité n'est pas forcément essentiellement innée, mais peut être aussi optimisée durant la vie de l'animal. Par exemple, il a été décrit par Hunter (1980, cité par [62]), dans le cas du choix des zones à fouiller par les mésanges, des capacités de prises de décision dépendant d'une certaine expérience du type de nourriture à espérer suivant les situations, donc d'un apprentissage et d'une capacité de prédiction. Ce type de modification de l'efficacité d'un comportement n'est cependant pas à confondre avec des phénomènes de maturation des organes sensoriels ou du système nerveux (cf. section E.2.5), comme c'est le cas pour la sélection des proies chez les daphnies par exemple.

En tout cas, le comportement des animaux ne les amène pas toujours, dans le cas de l'OFT par exemple, à un rendement optimal. D'une part, on observe une dépendance entre le rendement et la disponibilité de la nourriture : en hiver, l'animal tendra autant que possible à une efficacité maximale, alors qu'en été, lorsque la nourriture est plus facile à se procurer, c'est beaucoup moins le cas. Par ailleurs, d'autres motivations que la nutrition peuvent être en concurrence : déshydratation, fuite d'un prédateur... L'animal alors sera beaucoup moins réceptif aux stimuli alimentaires.

Approche Moderne

Le comportement animal a été analysé, via le modèle psychohydraulique de Lorenz (cf. section E.5.5), comme étant constitué d'une phase de recherche, ou *comportement appétitif*, et d'une phase d'accomplissement du but, ou *acte consommatoire*. Ce modèle est cependant critiquable, et on utilise plutôt aujourd'hui la notion plus générale de séquence de décisions articulant une séquence d'actions. Cela permet de plus de se passer de la notion de but, l'accent étant mis sur l'explication des circonstances amenant à la sélection d'un comportement plutôt que d'un autre.

Notons par ailleurs qu'un modèle tel l'OFT nous montre que la sélection de comportements n'est pas seulement dépendante de la nutrition mais d'un ensemble de facteurs, tous favorisant la survie, donc le potentiel reproductif de l'animal. Il n'est ainsi pas possible de considérer que l'animal n'a qu'un but à un moment donné, éventuellement décomposé en une série de sous-buts : il dispose à tout moment de toute une série de comportements potentiels concurrents, expressibles sur des échelles de temps variées et exprimés lorsque les conditions le permettent, compte-tenu d'une pondération de toutes les potentialités (qui auront tendance à s'inhiber mutuellement).

E.6 Stéréotypes

E.6.1 Définition

L'instinct, ou *comportement stéréotypé*, est défini comme un comportement inné (c'est-à-dire pouvant être produit sans avoir été précédemment observé) au déroulement invariable ou très peu ajustable. Notons dès maintenant que si son origine est purement génétique, l'expérience peut cependant le modifier, via une meilleure efficacité dans l'exécution de la séquence le plus souvent (cf. section E.2.7). Il est provoqué par un stimulus externe, le *déclencheur* ou *stimulus signal*, et se déroule sans possibilité d'interruption par la suite (quels que soient les stimuli perçus après déclenchement).

Il s'agit en général d'une séquence motrice ; elle est spécifique à une espèce et un stimulus précis. C'est un savoir-faire, dans le sens où il est efficace dans le cadre de vie habituel de l'animal. Il permet ainsi à des animaux à courte durée de vie, tels beaucoup d'insectes adultes, de mener à bien leurs tâches (reproductives en particulier) ce qu'ils ne pourraient faire par essais et erreurs en un temps si limité (construction de nids chez les guêpes non sociales).

E.6.2 Stimulus Déclencheur

Ainsi que nous l'avons décrit, un stimulus est à la base de tout comportement stéréotypé. Il peut être émis par une autre espèce, ce qui permet des comportements d'évitement de prédateur par exemple (repliement des ailes chez certains papillons de nuits en réaction aux ultra-sons caractéristiques des chauve-souris). Le choix d'un type de comportement peut ne tenir qu'à peu de chose : le comportement de soin chez la femelle dinde est déclenché par les pépiements de ses petits ; mais si elle est sourde, elle les tuera, leur présence déclenchant un comportement de prédation chez elle.

Caractéristiques

Le signe est souvent plus précis que la situation ne le laisserait croire : chez les oisillons de goéland, le comportement de quémande n'est pas déclenché par la présence de l'adulte uniquement mais aussi par la simple présentation d'un objet allongé muni d'une tache rouge, mimant le bec. De la même manière, le comportement d'agression du mâle épinoche est déclenché exclusivement par les objets portant une tache rouge (tout comme lui-même) ; il reste cependant neutre à une reproduction ressemblante d'un mâle si elle n'en est pas munie. Notons ici la pertinence de l'étude des causes immédiates (cf. section E.2.1) : le type de perception permis par les organes des sens est en correspondance directe avec les stimuli pertinents présents dans l'environnement de l'animal (perception quasi-exclusive des objets en mouvement par la rétine de la grenouille). Le stimulus peut aussi avoir une valeur quantitative : les parents d'oisillons les nourrissent plus s'ils ouvrent plus le bec, ce qui est caractéristique chez eux d'une faim plus intense. Il est souvent possible dans ce cas d'exhiber un *stimulus supranormal*, plus efficace (provoquant une réaction plus intense) que son équivalent naturel.

L'aspect spécifique du stimulus ne fait pas perdre trop d'efficacité à l'animal dans son milieu (bien qu'il ouvre la voie à des comportements de mimétisme par exemple), et permet une grande économie de traitement par rapport à une perception plus globale de la situation. Le fait que des leurres puissent être présentés par l'éthologiste ne remet pas en cause la pertinence du comportement : si ces perturbations étaient régulières dans l'environnement habituel de l'animal, une discrimination plus importante sur les stimulus serait sélectionnée. Globalement, l'animal extrait de l'environnement le minimum d'information qui lui est nécessaire pour agir de manière adéquate.

Exemples

Un exemple classique, étudié par Konrad Lorenz, est le comportement de roulage dans le nid d'un oeuf perdu par l'oie cendrée. Même s'il lui échappe en cours de route, elle continuera ses mouvements de bec jusqu'au nid. Elle aura le même comportement si on place un objet quelconque près de son nid, et le couvera ensuite longuement s'il est arrondi.

Cette situation est caractéristique de par son stimulus déclencheur précis, et par le déroulement de la séquence jusqu'à son terme une fois initiée, sans plus aucune référence à son objet initial (l'instinct est "sans référence à son résultat") : l'animal ne comprend pas sa signification.

Autre exemple : un chien tend sa gorge à son vainqueur pour faire cesser le combat, ce qui inhibe l'agressivité de ce dernier. Le stimulus est ici un geste (on parle de *stimuli-signe*), et sa résultante est une modification comportementale (l'altération motrice n'en est qu'une conséquence). La signification de ce signe est ici non acquise.

E.6.3 Altération

Les stéréotypes sont modifiables par sélection artificielle : par exemple, les poules tuent dans la nature tout poussin ne présentant pas un motif de taches spécifique sur le dos. Ce comportement a été assoupli chez l'espèce domestiquée par des croisements orientés (comparables à l'effet d'une pression de sélection, seuls les individus présentant le comportement désiré pouvant se reproduire).

Dans le cas de la réaction au bec à tache rouge chez le goéland, ce comportement s'avère appris : les jeunes goélands réagissent instinctivement à des objets variés en les picorant, mais orienteront par la suite leur intérêt vers les modèles réalistes du bec parental. S'ils sont élevés par des individus d'une autre espèce, cependant, ils seront attirés par les caractéristiques de leur bec.

E.6.4 Etude et Fonction

Les réflexes sont considérés comme les formes les plus simples de comportement. Cependant, on ne saurait les distinguer clairement des comportements complexes, en particulier du fait de l'intégration d'un grand nombre de réflexes dans ces derniers (comme dans le cas de la marche). On en arrive ainsi à la notion d'*unité de comportement* chère aux éthologistes. Ceux-ci déterminent en général les actions estimées caractéristiques d'un comportement complexe, puis les abstraient via la notion de *pattern* (qui remplacent un enregistrement exhaustif du moindre mouvement de l'animal), ensembles de mouvements élémentaires coordonnés (par exemple "se tourner", "avaler"). Ces mouvements sont analysés via des *éthogrammes* : il s'agit de caractériser une action en tant que séquence de mouvements précis. Les éthologistes ont initialement qualifié ces actions de schémas d'action fixés (*fixed action patterns*), insistant par là sur leur immuabilité. Ce n'est cependant pas toujours le cas, même pour des attitudes élémentaires. Par ailleurs, tous les comportements ne sauraient être décomposés en réflexes élémentaires ; le niveau de description choisi doit toujours dépendre de la complexité du comportement observé pour être significatif.

Certains *pattern* pourtant apparemment devenus inutiles sont encore activables à l'âge adulte (réflexes associés à l'éclosion chez le poussin) ; la conservation des centres de contrôle associés est probablement due à la réutilisation de certains de ces *patterns* dans la locomotion.

Un grand nombre de réflexes ont des objectifs *homéostatiques* : il s'agit de conserver une température constante, une position d'équilibre... C'est un *feedback*, dans le sens où un signal s'éloignant d'une norme va en retour déclencher une action qui le ramènera dans une zone de tolérance. Encore une fois, ce type de régulation peut déclencher aussi bien des comportements simples (respiration) que complexes (chasse). Par opposition à ces comportements en *boucle fermée*, beaucoup une fois déclenchés ne voient plus leur déroulement être altéré par aucun stimuli ; ils sont alors en *boucle ouverte*. C'est le cas de mouvements rapides tel le mouvement de capture de la mante religieuse.

Les comportements stéréotypés ne sauraient être considérés comme des aptitudes inférieures par rapport à celles qui seront décrites par la suite : ils sont d'une grande robustesse et efficacité dans l'environnement habituel de l'animal (gain de rapidité, simplicité de la physiologie sous-jacente), et peuvent sous-tendre l'ensemble de ses interactions, y compris avec les membres de son espèce (préparation de nid, cour, soins aux jeunes...). Ils se retrouvent à tous les niveaux de l'échelle évolutive, y compris chez le nourrisson (sourire face à un visage stylisé, "saisie" suite à un stimulus tactile...).

E.7 Apprentissage

L'acquisition ou la modification d'un comportement du fait de l'expérience individuelle est qualifiée d'apprentissage. En revenant à la définition initiale du comportement, celle-ci s'ajoute aux facteurs internes pour déterminer la réaction à des stimuli donnés.

D'un point de vue adaptatif, on peut considérer que l'apprentissage ou la cognition (traitée dans la section suivante) vont dans le sens d'une plus grande autonomie de l'animal par rapport à son environnement, puisque s'ajoutent à ses réactions immédiates la possibilité d'anticiper sur des évènements à venir.

E.7.1 Habituation et Sensibilisation

l'*habituation* est une forme primitive d'adaptation à son environnement, qui consiste en l'inhibition de la réaction à des stimuli lorsqu'ils se répètent ou ne fournissent pas de gain en retour. Elle est réversible après un certain temps sans exposition au stimulus, où la sensibilité redevient normale.

Les exemples de ce type sont nombreux : inhibition de la contraction de l'hydre si elle se trouve dans des courants trop agités, de la réaction aux signaux d'alerte chez beaucoup d'animaux s'ils ne sont pas suivis d'une attaque. Elle ne doit pas être confondue avec l'*adaptation sensorielle*, suivant laquelle les organes des sens s'adaptent à un stimulus répétitif ; ce phénomène est beaucoup moins persistant que l'habituation, et n'implique généralement que les organes sensoriels, non le système nerveux.

Un mécanisme symétrique, la sensibilisation, survient lorsque l'animal vient d'être stimulé, positivement ou négativement. Des rats réagissent bien plus fortement à des stimuli normalement sans importance s'ils viennent de recevoir une décharge électrique ; de même pour des vers marins venant d'être nourris. Il s'agit probablement d'un préliminaire à l'apprentissage : si la corrélation entre le stimulus excitant et ceux plus neutres se répète, il y aura possibilité de mémorisation (voir à ce propos l'apprentissage associatif, cf. section E.7.2) ; si la répartition est aléatoire, un phénomène d'habituation aura lieu.

L'habituation permet une adaptation aux stimuli d'un environnement donné ; s'il est certainement utile de réagir vivement à un évènement inhabituel, le fait de le faire pour tout stimuli neutre est une perte d'énergie, et empêche une réponse efficace à des stimuli effectivement associés à des dangers ou avantages.

E.7.2 Apprentissage Associatif

Conditionnement Classique (Pavlovien)

Le *conditionnement classique*, une forme d'*apprentissage associatif*, a été exhibé chez les mammifères et les oiseaux par le physiologiste Ivan Pavlov. Il se base sur une réaction innée, association d'une *réaction inconditionnée* (RI) à un *stimulus inconditionné* (SI). Par ailleurs, un second stimulus ne déclenchant pas initialement la RI est utilisé (c'est un *stimulus neutre*, SN), et n'ayant pas de réponse associée trop forte par ailleurs.

Si le SN est proche dans le temps du SI, il finit lui aussi par déclencher la RI, alors qualifiée de *réaction conditionnée* ou *réflexe conditionné* (RC) : le SN devient un *stimulus conditionné*

(SC).

Dans l'exemple historique, le sujet d'expérimentation est le chien : le SI est de la nourriture, et la RI la salivation. Le SN est matérialisé par une clochette, ne déclenchant pas *a priori* de réaction de salivation chez l'animal.

Cette association est intéressante à deux titres :

– elle établit une correspondance entre des stimulus déclenchant des zones distinctes du cerveau (son et perception gustative) ;
– elle associe un stimulus à une réponse qui y serait tout à fait étrangère en milieu naturel.

Il est intéressant de constater que l'association n'est pas stricte ; l'animal est capable de *généralisation*. Dans le cas du chien, l'animal salive d'autant plus intensément que la fréquence des sons qu'on lui fait entendre est proche de celle de la clochette initiale. Pour Pavlov, le conditionnement est la base de tout apprentissage en ce qu'il permet l'extension des réactions comportementale à de nouveaux stimulus. Notons cependant que la RC n'est pas identique à la RI : il n'y a donc pas de transfert simple entre le SI et le SC.

En parallèle à cette création d'associations, il est aussi possible d'inhiber une RC (jamais complètement, cependant) si l'expérimentateur effectue une association avec un nouveau stimulus ou si le SI est complètement supprimé. Si un comportement est renforcé pour un certain stimulus, et inhibé pour des stimuli proches, on obtient une *discrimination conditionnée*, symétrique à la généralisation (voir à ce propos E.7.3). Dans tous les cas, un renforcement ponctuel reste nécessaire : si toute récompense est supprimée, on observe une *extinction expérimentale*. Ceci n'est cependant pas un retour à l'état initial de l'animal avant conditionnement : il est en effet facile de réactualiser la réaction (*recouvrement spontané*) en ne stimulant pas l'animal pendant quelques temps ou en lui fournissant un nouveau stimulus.

Ces capacités peuvent être exploitées par d'autres animaux, par exemple dans le cas de certains insectes. En parallèle à un goût pénible ou une toxicité les rendant impropres à la consommation, ils sont vivement colorés. Ainsi, après quelques expériences désagréables, un prédateur éventuel les associera aux couleurs et ne tentera plus de chasser ce type de proie. La généralisation ayant lieu permet par ailleurs à d'autres espèces pourtant comestibles d'être protégées par mimétisme.

Notons que les animaux ne se contentent pas de réagir à des stimuli : on constate qu'ils se mettent dans des situations ou effectuent des actions telles que la probabilité d'observer ces stimuli augmente. Les chiens de Pavlov, après quelques temps, se dirigeait ainsi directement vers la zone d'expérimentation dès qu'ils étaient introduits dans le laboratoire, et y manifestaient une grande agitation. Ce type d'initiative est caractéristique des apprentissages par *essais et erreurs*, décrits dans la section suivante.

Conditionnement Opérant (Skinnerien)

Le *conditionnement opérant*, autre forme d'apprentissage associatif, a été découvert par Burrhus Frederic Skinner. Ici, l'accent est mis sur la modification des fréquences de réponses comportementales.

Un exemple classique d'expérimentation en ce sens est la "boîte de Skinner", dans laquelle un rongeur, s'il appuie sur une pédale, obtient une récompense (nourriture). Cet évènement est qualifié d'*agent renforçateur*. Après les manipulations initialement aléatoires de la pédale suivies des premières récompenses, la fréquence d'appui s'avère augmenter rapidement. L'action de ces récompenses sur les réactions de l'animal est qualifiée de *renforcement*, ou *conditionnement opérant*. Inversement, une inhibition de comportement peut être obtenue si sa présence est suivie de conséquences négatives (c'est ainsi que les rats apprennent à éviter les aliments empoisonnés, par exemple). Ceci appuie la théorie de l'effet de Edward Lee Thorndike, suivant laquelle la fréquence d'un comportement varie comme son utilité, et lui vaut sa qualification d'*apprentissage par essais et erreurs*.

Un mécanisme d'apprentissage similaire à celui observé par Pavlov peut être observé dans ce dispositif : l'animal associe à l'un de ses comportements un gain ou une punition, ce qui l'amènera par la suite à le répéter ou à l'éviter. Notons qu'un *renforcement secondaire* peut être utilisé, un stimulus neutre faisant le pont entre un comportement et sa récompense ; si ce stimulus est toujours associé aux moments de récompense, il pourra en tenir lieu (lors de dressages par exemple, où une récompense immédiate n'est pas toujours possible).

Une critique faite à cet apprentissage est que l'animal agit peut-être plus par analogie que par réelle association. En effet, on constate qu'il n'appuie que rarement clairement sur la pédale : en général, il la lèche, la mordille, et manifeste ainsi toute une série de comportements associés à la nourriture. La notion de "pression de pédale" est étrangère au rat : il apprend en fait essentiellement à déplacer, à modifier sa réponse (mais pas sa nature). De même, les pigeons n'actionnent pas du bec un levier de la même manière suivant qu'ils en attendent de la nourriture ou de l'eau. Ainsi, le type de réaction que l'on peut attendre d'un animal est toujours dépendant de son mode de vie, et certaines associations ne peuvent être apprises car trop éloignée de ce référentiel : les rats peuvent apprendre une aversion pour des aliments sur la base de leur goût, mais pas sur celle de stimuli visuels par exemple. Par contre, les oiseaux en sont capables (évitement de chenilles vivement colorées), car ils utilisent beaucoup plus les stimuli visuels. Il est même possible d'observer, dans certains cas, des artefacts comportementaux voire des réponses complètes allant à l'encontre du système de récompense.

En tout cas, cette forme d'apprentissage est très présente dans la nature : elle permet en particulier aux animaux d'apprendre à choisir leur nourriture, sa saveur pour un individu d'une espèce donnée étant caractéristique de sa valeur nutritive. Ce type de perception étant innée, les gènes influencent donc l'apprentissage dans le milieu naturel. Le fait que certaines mésanges percent le bouchon des bouteilles de lait pour se nourrir de la crème peut être interprété dans ce cadre : elles ont appris qu'elles étaient récompensées si elles picoraient ces capsules (ceci appartenant au registre comportemental de l'espèce par ailleurs, ce qui explique les essais initiaux).

E.7.3 Autres Modalités d'Apprentissage

Nous décrivons ici des expériences plus complexes d'apprentissages, considérées comme révélatrices de capacités plus développées que celles requises par l'apprentissage associatif.

Discriminatif Simple

La technique du conditionnement Pavlovien permet de tester les capacités sensorielles et cognitives de discrimination entre deux stimuli chez un animal. Soient deux stimuli de même nature (formes géométriques, couleurs...) SC1 et SC2 auxquels l'animal apprend à associer respectivement une réaction conditionnée RC ou son inhibition. Il est alors possible, en présentant divers stimuli et en observant la réponse de l'animal, de déterminer quel type de stimulus il a perçu.

Notons qu'il est possible de créer une "névrose expérimentale" lorsque SC1 et SC2 en viennent à être indiscernables (transformation d'une forme en une autre par exemple) : des troubles du comportement apparaissent alors chez l'animal.

Discriminatif avec Inversion

Une variation de l'apprentissage discriminatif simple consiste, lorsque la réponse de l'animal est stabilisée, à inverser la consigne. Des répétitions de ce protocole ne modifient pas le temps d'apprentissage de certaines espèces (poissons), alors que d'autres en tirent parti et "apprennent à apprendre" (mammifères).

Probabiliste

Une variation sur les apprentissages discriminatifs consiste en une situation dans laquelle l'expérimentateur récompense une des réponse avec une probabilité de p et l'autre avec une probabilité de $1 - p$ (typiquement, $p = 0.8$). Deux types de réactions apparaissent alors. Certaines espèces (poissons) finissent par choisir systématiquement le comportement le plus renforcé (*maximisation*), alors que d'autres (cas des mammifères) produisent chaque réponse avec sa probabilité respective (*matching*).

La plupart des apprentissages en milieu naturel sont de type probabiliste.

Par Observation

Les vertébrés sont souvent capables d'observer leurs congénères, et d'en tirer des informations. Dans le cas du comportement des mésanges décrit au E.7.2, la rapidité de propagation suggère qu'elle a été apprise par imitation d'individus ayant initialement acquis ce comportement seuls, par apprentissage associatif. De ce mécanisme procède aussi l'apprentissage du chant décrit en E.3.3.

Les mammifères apprennent de leurs congénères durant leur enfance ; par exemple, des singes élevés dans l'isolement n'ont aucune réponse sociale adaptée, en particulier d'un point de vue sexuel et parental. La chasse en meute semble être un comportement appris par observation des adultes par les jeunes.

Topographique

Il s'agit ici d'observer les capacités de repérage de l'animal dans un environnement nouveau ; on construit en général un labyrinthe à cet effet. Une série de comportements récurrents

sont constatés chez les espèces réussissant bien ce type de tests : éviter de tourner en rond, alterner les choix de directions, accélérer près du but, etc. Par ailleurs, certains animaux (rats) utilisent durant une phase initiale des indices extéroceptifs (vision, olfaction) pour ensuite, une fois l'environnement mieux connu, se guider essentiellement sur les sensations kinesthésiques (distance parcourue, etc.).

Ce type de test permet, en plus d'une mémorisation de points-clefs, l'exhibition de capacités plus abstraites de construction de cartes, voire de capacités de manipulation de celles-ci par l'utilisation de raccourcis (détours de locomotion).

E.7.4 Analyse

L'apprentissage vise à des *changements adaptatifs* du comportement de l'animal ; tout comme l'instinct, il vise donc à la mise en place d'une série de réponses adaptant au mieux l'animal à son environnement. À ce propos, il est à noter que les études des capacités d'apprentissage des animaux (les rats en particulier) dans des conditions expérimentales et les classifications qui en ont résulté ont beaucoup moins de crédit aujourd'hui. On considère plutôt que ces capacités ont été sélectionnées du fait de leur efficacité dans un environnement qui leur est spécifique (tout comme leurs comportements), et ne sauraient donc être étudiées sans y faire référence. Par ailleurs, l'apprentissage est considéré comme une capacité à extraire de cet environnement uniquement ses caractéristiques pertinentes, ce biais étant donc spécifique à cette espèce dans cet environnement. L'animal n'apprend donc pas tant une association simple stimulus-récompense qu'à identifier les situations lui permettant d'attendre des gains importants. Il n'est d'ailleurs pas toujours conditionné par un système de renforcement : les animaux explorent par exemple spontanément leur environnement, et y sont ainsi plus efficaces que ceux pour qui il est étranger.

Des compétences cognitives précises spécialisant les apprentissages existent : les animaux stockant de la nourriture mémorisent efficacement les indices topologiques, par exemple. Cependant, il existe des mécanismes communs : le conditionnement classique, par exemple, est exhibable aussi bien chez les abeilles que les pieuvres ou les mammifères.

[26] interprète les actions animales complexes comme des associations de stéréotypes déclenchés par des stimuli simples et modulés par des capacités d'apprentissage finalement superficielles. Les capacités d'apprentissage sont en tout cas déterminées génétiquement, et manifestement soumises à pression de sélection du fait de la relative précision des périodes d'apprentissages et des stimuli pouvant les déclencher.

Bibliographie

[1] P. Agati, Y. Brémont, and G. Delville. *Mécanique du solide - Applications industrielles*. Dunod, 1986.

[2] Omar Ahmad, James Cremer, Joseph Kearney, Peter Willemsen, and Stuart Hansen. Hierarchical, concurrent state machines for behavior modeling and scenario control. In *Proceedings of the 1994 Conference on AI, Planning, and Simulation in High Autonomy Systems*, pages 36–42, Gainesville, Florida, USA, December 1994.

[3] Alias|Wavefront. Maya. `http://www.alias.com/eng/index.shtml`.

[4] Jose Annunziato. A review of virtual actor and figure animation techniques.

[5] Ronald C. Arkin. Towards the unification of navigational planning and reactive control. In *Working Notes of the AAAI Spring Symposium on Robot Navigation*, pages 1–5, Stanford University, March 1989.

[6] Ronald C. Arkin. The multiple dimensions of action-oriented robotics perception : Fission, fusion and fashion. In *Working Notes of the AAAI Fall Symp. on Sensory Aspects of Robotic Intelligence*, Monterey, CA, 1991.

[7] K. J. Astrom, T. Hagglund, C. C. Hang, and W. K. Ho. Automatic tuning and adaptation for pid controllers—a survey. In L. Dugard, M. M'Saad, and I. D. Landau, editors, *Adaptive Systems in Control and Signal Processing*, pages 371–376, Oxford, U.K., 1992. Pergamon.

[8] Paolo Baerlocher and Ronan Boulic. Parametrization and range of motion of the ball-and-socket joint. In *Deformable Avatars*, pages 180–190, 2000.

[9] C.B. Barber, D.P. Dobkin, and H.T. Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Trans. on Mathematical Software*, 22(4) :469–483, Dec 1996.

[10] A.H. Barr. Global and local deformations of solid primitives. *Computer Graphics (SIGGRAPH'84 Proceedings)*, 18(3) :21–31, 1984.

[11] M. Bertin, J.P. Faroux, and J. Renault. *Mécanique 2*. Dunod université, 1985.

[12] Carole Blanc. *Techniques de modélisation et de déformation de surfaces pour la synthèse d'images*. PhD thesis, Université Bordeaux I, 1994.

[13] Blender. `http://www.blender3d.com`.

[14] Bruce M. Blumberg. Autonomous animated interactive characters : Do we need them ? In *Computer Graphics International (Proceedings)*, pages 29–37. IEEE, 1997.

[15] Bruce M. Blumberg and Tinsley A. Galyean. Multi-level direction of autonomous creatures for real-time virtual environments. In *Computer Graphics Proceedings*, pages 47–54, 1995.

[16] Christophe Bordeux, Ronan Boulic, and Daniel Thalmann. An efficient and flexible perception pipeline for autonomous agents. *Eurographics*, 18(3) :23–30, 1999.

[17] Gareth Bradshaw and Carol O'Sullivan. Sphere-tree construction using dynamic medial axis approximation. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 33–40. ACM Press, 2002.

[18] David C. Brogan and Jessica K. Hodgins. Group behaviors for systems with significant dynamics. *Autonomous Robots*, 4 :137–153, 1997.

[19] Rodney A. Brooks. A robot that walks ; emergent behaviors from a carefully evolved network. *Neural Computation*, 1(2) :253–262, Summer 1989.

[20] Rodney A. Brooks. Elephants don't play chess. *Robotics and Autonomous Systems*, 6 :3–15, 1990.

[21] Rodney A. Brooks. How to build complete creatures rather than isolated cognitive simulators. In Kurt VanLehn, editor, *Architectures for Intelligence*, pages 225–240, Hillsdale, New Jersey, 1991. Lawrence Erlbaum Associates.

[22] Rodney A. Brooks. Integrated systems based on behaviors. *SIGART Bulletin*, 2(4) :46–50, Août 1991.

[23] Rodney A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47 :139–159, 1991.

[24] Rodney A. Brooks. The role of learning in autonomous robots. In *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, pages 5–10, San Mateo, CA, 1991. Morgan Kaufmann.

[25] Armin Bruderlin and Tom Calvert. Knowledge-driven, interactive animation of human running. In *Proc. Graphics Interface 96*, pages 213–221, Toronto, Canada, May 1996.

[26] Neil Campbell. *Biologie*. De Boeck Université, 1995.

[27] Somporn Chuai-Aree. PlantVR. `http://www.iwr.uni-heidelberg.de/~Somporn.ChuaiAree/software/plantvr.htm`.

[28] Michael F. Cohen. Interactive spacetime control for animation. In *Computer Graphics Proceedings*, volume 26, pages 293–302, 1992.

[29] MPEG Committee. MPEG-4 description. `http://www.chiariglione.org/mpeg/standards/mpeg-4/mpeg-4.htm`.

[30] James Cremer and Joseph Kearney. Improvisation and opportunism in scenario control for driving simulation. In *First Workshop on Simulation and Interaction in Virtual Environments*, pages 114–119, The University of Iowa, USA, July 1995.

[31] Stéphane Donikian. HPTS : a behaviour modelling language for autonomous agents. In *Proceedings of the fifth international conference on Autonomous agents*, pages 401–408, Montreal, Canada, Mai 2001.

[32] Marco Dorigo. ALECSYS and the AutonoMouse : Learning to control a real robot by distributed classifier systems. *Machine Learning*, 19(3) :209–240, 1995.

[33] Marco Dorigo and Marco Colombetti. Robot shaping : Developing autonomous agents through learning. *Artificial Intelligence Journal*, 71 :321–370, 1994.

[34] Brett Douville, Libby Levison, and Norman I. Badler. Task-level object grasping for simulated agents. *Presence*, 5(4) :416–430, 1996.

[35] John C. Doyle, Bruce A. Francis, and Allen R. Tannenbaum. *Feedback control theory*. MacMillan, New York, 1992.

[36] Ronan Boulic et al. Human free-walking model for a real-time interactive design of gaits. In N. Magnenat-Thalmann and D. Thalmann, editors, *Computer Animation*, pages 61–79, Tokyo, 1990. Springer-Verlag.

[37] Henry Farreny and Malik Ghallab. *Élements d'intelligence artificielle*. Hermes, 1990.

[38] T. Firsova, D. Ivanov, V. Kuriakin, E. Martinova, R. Rodyushkin, and V. Zhislina. Life-like MPEG-4 3D "talking head" (beyond standard). In Dimitri Plemenos, editor, *Computer Graphics and Artificial Intelligence (3IA'2002 Proceedings))*, pages 115–126. MSI - AFIG, May 2002.

[39] Thomas A. Funkhouser. RING : A client-server system for multi-user virtual environments. In *Symposium on Interactive 3D Graphics*, pages 85–92, 209, 1995.

[40] Alain Gallo. *Les Animaux, Psychologie et Comportement*. Editions Milan, 1999.

[41] Jean-Gabriel Ganascia. *L'âme-machine*. Seuil, 1990.

[42] J.J. Gibson. *The Perception of the Visual World*. Riverside Press, Cambridge, 1950.

[43] J.J. Gibson. *The Ecological Approach to Visual Perception*. Houghton Mifflin, Boston MA, 1979.

[44] Stanford University Courses. Gino van den Bergen. Proximity query and penetration depth computation on 3D game objects. `http://graphics.stanford.edu/courses/cs468-01-fall/Papers/van-den-bergen.pdf`.

[45] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, 1989.

[46] Radek Grzeszczuk and Demetri Terzopoulos. Automated learning of muscle-actuated locomotion through control abstraction. In *Computer Graphics Proceedings*, pages 63–70, 1995.

[47] Radek Grzeszczuk, Demetri Terzopoulos, and Geoffrey Hinton. Neuroanimator : Fast neural network emulation and control of physics-based models. In *Computer Graphics Proceedings*, pages 9–20, 1998.

[48] J. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

[49] François Félix Ingrand, Raja Chatila, Rachid Alami, and Frédéric Robert. Embedded control of autonomous robots using procedural reasoning. In *Proceedings of the 7th International Conference on Advanced Robotics (ICAR'95)*, pages 855–861, September 1995.

[50] Institute of Mechanical Engineering. AnyBody Project. http://anybody.auc.dk.

[51] E. Gilbert D. Johnson and S. Keerthi. A fast procedure for computing the distance between complex objects in three dimensional space. *IEEE Journal of Robotics and Automation*, 4(2) :193–203, 1988.

[52] K. Kähler, J. Haber, and H.-P. Seidel. Geometry-based muscle modeling for facial animation. In *Proceedings Graphics Interface 2001*, pages 37–46, 2001.

[53] P Kalra, A Mangili, N Magnenat-Thalmann, and D Thalmann. Simulation of facial muscle actions based on rational free form deformations. *Computer Graphics Forum*, 11(3) :59–69, 1992.

[54] Scott A. King, Richard E. Parent, and Barbara L. Olsafsky. A muscle-based 3D parametric lip model for speech-synchronized facial animation. In Nadia Magnenat-Thalmann and Daniel Thalmann, editors, *Deformable Avatars*, pages 12–23. Kluwer Academic Publishers, 2001.

[55] Rolf M. Koch, Markus H. Gross, and Albert Bosshard. Emotion editing using finite elements. *Computer Graphics Forum*, 17(3) :295–302, 1998.

[56] Sumedha Kshirsagar, Stephane Garchery, and Nadia Magnenat-Thalmann. Feature point based mesh deformation applied to MPEG-4 facial animation. In Nadia Magnenat-Thalmann and Daniel Thalmann, editors, *Deformable Avatars*, pages 24–34. Kluwer Academic Publishers, 2001.

[57] J.Y. Lettvin, H.R. Maturana, W.S McCulloch, and W.H. Pitts. What the frog's eye tells the frog's brain. *Proceedings of the IRE*, 47(11) :1940–51, 1959.

[58] L.Herda, R. Urtasun, P. Fua, and A. Hanson. Automatic determination of shoulder joint limits using quaternion field boundaries. In *5th International Conference on Automatic Face and Gesture Recognition*, pages 95–100, Washinton DC, USA, May 2002.

[59] Ik Soo Lim and Daniel Thalmann. A vector-space representation of motion data for example-based motion synthesis. In Nadia Magnenat-Thalmann and Daniel Thalmann, editors, *Deformable Avatars*, pages 169–179. Kluwer Academic Publishers, 2001.

[60] Brian Lingard. Animating articulated structures. http://www.cs.wpi.edu/~matt/courses/cs563/talks/animhtml/anim.html.

[61] Pattie Maes and Rodney A. Brooks. Learning to coordinate behaviors. In *National Conference on Artificial Intelligence*, pages 796–802, 1990. learning.pdf.

[62] Aubrey Manning and Marian Stamp Dawkins. *An Introduction to Animal Behaviour*. Cambridge University Press, 1998.

[63] R. Mas-Sanso. and D. Thalmann. A hand control and automatic grasping system for synthetic actors. In *Proceedings of Eurographic'94*, pages 167–178, 1994.

[64] Michael Mateas. An Oz-centic review of interactive drama and believable agents. In M. Wooldridge and M. Veloso, editors, *AI Today : Recent Trends and Developments (Lecture Notes in AI 1600)*, pages 297–328. Springer, 1999.

[65] Michael McKenna and David Zeltzer. Dynamic simulation of autonomous legged locomotion. *Computer Graphics*, 24(4) :29–38, 1990.

[66] S. McMillan. *Computational Dynamics for Robotic Systems on Land and Under Water*. PhD thesis, The Ohio State University, Columbus, OH, Summer 1994.

[67] Scott McMillan. Dynamechs. `http://dynamechs.sourceforge.net`.

[68] Orazio Miglino, Henrik Hautop Lund, and Stefano Nolfi. Evolving mobile robots in simulated and real environments. *Artificial Life*, 2(4) :417–434, 1995. miglino.simreal.pdf.

[69] M. Mikita. 3D free-form deformation : Basic and extended algorithms. In W. Purgathofer, editor, *Proc. of the* 12^{th} *Spring Conference on Computer Graphics*, pages 183–191. Comenius University, Bratislava, 1996.

[70] M. Minsky. Frame-system theory. In P. Johnson-Laird and P.C. Wason, editors, *Thinking : Readings in cognitive Science*, Cambridge, 1977. Cambridge University Press.

[71] M. L. Minsky. *The society of mind*. Simon and Schuter, New york, 1986.

[72] Brian Mirtich. VClip. `http://www.merl.com/projects/vclip`.

[73] Brian Mirtich. Fast and accurate computation of polyhedral mass properties. *J. Graph. Tools*, 1(2) :31–50, 1996.

[74] Brian Mirtich. V-Clip : fast and robust polyhedral collision detection. *ACM Transactions on Graphics*, 17(3) :177–208, July 1998.

[75] Frank Multon, Laure France, Marie-Paule Cani-Gascuel, and Gilles Debunne. Computer animation of human walking : a survey. *The Journal of Visualization and Computer Animation*, 10 :39–54, 1999.

[76] Jean-Christophe Nebel. Soft tissue modelling from 3D scanned data. In Nadia Magnenat-Thalmann and Daniel Thalmann, editors, *Deformable Avatars*, pages 85–97. Kluwer Academic Publishers, 2001.

[77] J. Thomas Ngo and Jo Marks. Spacetime constraints revisited. In *Computer Graphics Proceedings*, pages 343–350, 1993.

[78] Hansrudi Noser, Olivier Renault, Daniel Thalmann, and Nadia Magnenat Thalmann. Navigation for digital actors based on synthetic vision, memory and learning. *Computer & Graphics*, 19(1) :7–19, 1995.

[79] J. O'Rourke and N. Badler. Decomposition of three-dimensional objects into spheres. *IEEE Trans. On Pattern Analysis and Machine Intelligence*, 1(3) :295–305, July 1979. PAMI-1(3) 417.

[80] Carol O'Sullivan, John Dingliana, Fabio Ganovelli, and Gareth Bradshaw. Collision handling for virtual environments. In *Eurographics Tutorial*, 2001.

[81] I. J. Palmer and R. L. Grimsdale. Collision detection for animation using sphere trees. *Computer Graphics Forum*, 14(4) :105–16, May 1995.

[82] R.E. Parent. A system for sculpting 3-D data. *ACM SIGGRAPH Computer Graphics*, 11(2) :138–147, 1977.

[83] Olivier Parisy and Christophe Schick. A physically realistic framework for the generation of high-level animation controllers. In *Smart Graphics*, pages 47–54, Hawthorne, New York, USA, 2002. ACM Press.

[84] Olivier Parisy, Christophe Schlick, and Benoît Crespin. Object-oriented reformulation and extension of implicit free-form deformations. In *Deformable Avatars*, pages 72–84. Kluwer, 2000.

[85] Ken Perlin and Athomas Goldberg. Improv : A system for scripting interactive actors in virtual worlds. In *Computer Graphics Proceedings*, pages 205–216, 1996.

[86] Stéphane Popinet. GNU Triangulated Surfaces. `http://gts.sourceforge.net`.

[87] Franco P. Prepara and Michael Ian Shamos. *Computational Geometry - An Introduction*. Springer-Verlag, 1985.

[88] John Rasmussen, Johan Dahlquist, Michael Damsgaard, Mark de Zee, and Sören T. Christensen. Musculoskeletal modeling as an ergonomic design method. In *International Ergonomics Association XVth Triennial Conference*, Seoul, Korea, 24-29 August 2003.

[89] Jacques Renault. *Précis de physique*. Dunod, 1996.

[90] Craig W. Reynolds. Computer animation with scripts and actors. In *Computer Graphics Proceedings*, volume 16(3), pages 289–296, 1982.

[91] Craig W. Reynolds. Not bumping into things. In *Notes for the SIGGRAPH '88 course Developments in Physically-Based Modeling*. ACM SIGGRAPH, 1988.

[92] Craig W. Reynolds. Competition, coevolution and the game of tag. In Rodney A. Brooks and Pattie Maes, editors, *Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, pages 59–69, MIT, Cambridge, MA, USA, 6-8 July 1994. MIT Press.

[93] Thomas Rist and Elisabeth André. Building smart embodied virtual characters. In Andreas Butz, Antonio Krüger, and Patrick Olivier, editors, *Smart Graphics*, pages 123–130. Springer, 2003.

[94] Chuai-Aree S., Jaeger W., Bock H.G., Siripant S., and C. Lursinsap. PlantVR : An algorithm for generating plant shoot and root growth using applied lindenmayer systems. In *2003' International Symposium on Plant growth Modeling, simulation, visualization and their Applications*, pages 55–68, Beijing (P.R.China), October 13-16 2003.

[95] Cédric Sanza, Christophe Destruel, and Yves Duthen. Agents autonomes pour l'interaction adaptative dans les mondes virtuels. In *AFIG 97*, Rennes, 1-3 décembre 1997. AFIG.

[96] T.W. Sederberg and S.R. Parry. Free-form deformations of polygonal data – déformations quelconques de polygones. In *Actes du deuxième colloque image (Semaine internationale de l'image électronique), Nice*, pages 633–639, 1986.

[97] T.W. Sederberg and S.R. Parry. Free-form deformations of solid geometric models. *Computer Graphics (SIGGRAPH'86 Proceedings)*, 20(4) :151–160, 1986.

[98] Sigel Project. `http://ls11-www.cs.uni-dortmund.de/people/sigel`.

[99] Karl Sims. Evolving virtual creatures. In *Computer Graphics Proceedings*, pages 15–22, 1994.

[100] Russell Smith. Open Dynamic Engine. `http://opende.sourceforge.net`.

[101] H-Anim Standard. `http://www.h-anim.org`.

[102] Bradley Stuart, Patrick Baker, and Yiannis Aloimonos. Towards the ultimate motion capture technology. In Nadia Magnenat-Thalmann and Daniel Thalmann, editors, *Deformable Avatars*, pages 143–157. Kluwer Academic Publishers, 2001.

[103] David J. Sturman. Character motion systems. In *SIGGRAPH 94 : Course 9*, 1994.

[104] Daniel Thalmann. Robotics methods for task-level and behavioral animation. In Daniel Thalmann, editor, *Scientific Visualization and Graphics Simulation*, pages 129–147. Wiley, July 1990.

[105] Gwenola Thomas. *Environnements Virtuels Urbains : Modélisation des informations nécessaires à la Simulation d'Humanoïdes*. PhD thesis, Université Rennes I, IRISA, équipe SIAMES, 1999.

[106] Xiaoyuan Tu and Demetri Terzopoulos. Artificial fishes : Physics, locomotion, perception, behavior. In *Computer Graphics Proceedings*, pages 43–50, 1994.

[107] Valve Software. `http://www.valvesoftware.com`.

[108] Michiel van de Panne and Eugene Fiume. Sensor-actuator networks. In *Computer Graphics Proceedings*, pages 335–342, 1993.

[109] Michiel van de Panne, Eugene Fiume, and Zvonko Vranesic. Reusable motion synthesis using state-space controllers. In *Computer Graphics Proceedings*, volume 24(4), pages 225–234, 1990.

[110] Gino van den Bergen. Efficient collision detection of complex deformable models using AABB trees. *Journal of Graphics Tools : JGT*, 2(4) :1–14, 1997.

[111] Bonnie Webber and Norman Badler. Animation through reactions, transition nets and plans. In *Proceedings of the International Workshop on Human Interface Technology*, Aizu, Japan, October 1995.

[112] Eric Werner. What ants cannot do. In J.W. Perram and J.P. Müller, editors, *Proceedings of the 6th European Workshop on Modelling Autonomous Agents : Distributed Software Agents and Applications*, pages 19–39. Springer Verlag, 1996.

[113] Stewart W. Wilson. The animat path to AI. In J.A. Meyer and S.W. Wilson, editors, *From Animals to Animats*, pages 15–21, Cambridge, Massachusetts, 1991. The MIT Press / Bradford Books.

[114] Andrew Witkin and Zoran Popović. Motion warping. In *Computer Graphics Proceedings*, pages 105–108, 1995.

[115] Ming C. Lin Young J. Kim and Dinesh Manocha. DEEP : dual-space expansion for estimating penetration depth between convex polytopes. In *IEEE International Conference on Robotics and Automation*, May 2002.

Résumé

L'animation de créatures virtuelles est un domaine ayant bénéficié d'un effort de recherche important en informatique graphique. Des techniques telles l'interpolation de clefs ou la capture de mouvement sont très largement utilisées, mais imposent un travail important à l'animateur du fait des possibilités limitées de réutilisation des mouvements produits.

En parallèle à ces techniques "en boucle ouverte", des approches basées sur des outils de plus haut niveau ont vu le jour : simulation physique, algorithmes moteurs, systèmes perceptifs, etc. Elles présentent l'intérêt d'abstraire le mouvement en fournissant à l'animateur des outils de contrôle aux paramètres moins nombreux et plus génériques que les techniques traditionnelles.

Cette problématique du contrôle d'un système en fonction d'objectifs, ainsi que certains des outils décrits se retrouvent dans le domaine de la vie artificielle. Celle-ci s'intéresse en effet à la synthèse de systèmes se comportant de manière similaire aux êtres vivants. Des implémentations robotiques de ces techniques ont montré les possibilités offertes par la genèse de comportements dans un environnement complexe.

Constatant un besoin de généricité des outils d'animation en informatique graphique, et l'intérêt d'un environnement simulé complexe en vie artificielle, nous étudierons le rapprochement des problématiques de ces communautés via une plate-forme commune, articulée autour de la notion de simulation physique et la génération aisée de modèles physiques de créatures articifielles.

Nous étudierons un cadre de travail commun tenant compte des besoins et contraintes de ces disciplines, les difficultés posées par l'utilisation d'une simulation physique et en quoi les outils proposés en facilitent l'accès. Nous montrerons enfin les usages envisageables pour cette plate-forme, et les apports attendus dans les domaines de l'informatique graphique et de la vie artificielle.

Mots Clefs

Génération de Modèle Physique, Simulation Physique, Contrôle d'Animation, Comportement, Vie Artificielle

Summary

The animation and computer simulation of virtual creatures are topics which have benefitted from a strong research impulse from the community of computer graphics. Some techniques, such as keyframing of motion capture, are widely used but impose a great burden on the animator because of the limited re-use possibilities of produced movements in new projects.

Parallel to those open loop techniques, other approaches based on higher level tools have been developped : physical simulation, locomotion / prehension algorithms, perceptive systems, etc. They all have the interest of abstracting the movement by bringing the animator control tools with higher-level parameters, and in a smaller number than traditionnal techniques.

This problem of system control with respect to some goals, and some described tools have counterparts in the artificial life domain. This discipline addresses the issue of the genesis of biological phenomenom by synthetising systems which behave in a way similar to life beings. Beyond computer simulations, robotic implementations of those techniques have shown their practical interest and possibilities offered by behaviours generation in a complex environment.

Basing our work on the observation of a need of genericity for animation tools in computer graphics, and on the interest of a complex environment in which artificial life simulations can take place, we will show the usefulness of bringing together the issues of those communities using a common platform, based itself on a physical simulation and the easy generation of physical models of artificial creatures.

We will hence study a framework taking into account needs and constraints of those disciplines, the issues raised by physical simulation use and how the tools we are proposing make it easier to use. We will finally show how this platform can be used, and the expected contributions either from a computer graphics or artificial life point of view.

Keywords

Physical Model Generation, Physical Simulation, Animation Control, Behaviour, Artificial Life