Collision et adhésion d'objets élastiques

Rapport de stage de Magistère 1 (2001/2002)

Auteur : Éva Simonin

Stage effectué au sein du laboratoire TIMC
sous la tutelle de Jean-Louis Martiel et Emmanuel Promayon

Table des matières

Introduction

J'ai effectué ce stage au sein du laboratoire TIMC, sous la tutelle de Jean-Louis Martiel et Emmanuel Promayon, appartenant respectivement aux équipes TIMB (Traitement de l'Information et Modélisation en Biologie) et GMCAO (Gestes Médico-Chirurgicaux Assistés par Ordinateur). Je vais donc présenter le cadre dans lequel j'ai travaillé avant de développer le sujet de mon stage.

1- Contexte

1.1- L'équipe TIMB

Le travail du TIMB se situe au niveau de la biologie cellulaire. Plus précisément, y sont étudiés la structure du génome et la régulation de l'expression des gènes, ainsi que les interactions entre les différentes voies de signalisation (électrique, biochimique) et leur couplage avec le mouvement et les modifications de l'architecture cellulaire. Cela passe, par exemple, par le développement d'outils mathématiques qui vont permettre l'amélioration d'algorithmes utilisés dans ce contexte (algorithmes stochastiques).

Au sein de cette équipe, Jean-Louis Martiel travaille, entre autre, sur la potentiation à long terme des neurones. Celle-ci intervient lorsque le dendrite d'un neurone se trouve stimulé répétitivement. Se déclenche alors une cascade de réactions chimiques qui induisent l'apparition d'une protubérance sur le dendrite. Une des protéines impliquées dans ce phénomène est la CaMKII, dont la présence est corrélée au changement de morphologie du dendrite. Pour vérifier l'hypothèse selon laquelle cette protéine modifie localement la rigidité membranaire, des simulations sont nécessaires. La modélisation du corps élastique et isotrope que constitue le neurone nécessite donc des outils mathématiques et informatiques.

1.2- L'équipe GMCAO

L'équipe des GMCAO a pour but général d'offrir des outils informatiques aux médecins ou chirurgiens, leur permettant une pratique plus précise et moins invasive diminuant ainsi le risque pour le patient. La recherche se situe sur trois plans : perception, raisonnement et action.

Au niveau de la perception, l'imagerie médicale et le suivi d'objets à partir de capteurs est la voie la plus fouillée. L'utilisation de capteurs per-opératoires permet des

localisations tridimensionnelles (par exemple, d'un instrument chirurgical dans l'environnement de l'intervention) ou des modélisations à partir des données de ces capteurs (par exemple, reconstruction en 3D).

Dans la partie raisonnement, on essaie de simuler des objets physiques avec des données provenant de modèles réels afin de mieux comprendre leur fonctionnement exact. En effet, comprendre le comportement d'un organe au sein de son environnement est essentiel à la formation des chirurgiens susceptibles d'agir sur cet organe. Celui-ci étant un objet déformable et élastique, il faut utiliser des lois bio-mécaniques pour le modéliser correctement. C'est dans cette partie que travaille Emmanuel Promayon.

La partie action est celle qui s'occupe de l'interface homme-machine. On y met donc au point des systèmes robotisé, notamment dans le cadre de la téléopération, pour permettre des applications dans ce domaine.

Le travail d'Emmanuel Promayon s'applique à la représentation et simulation d'objets complexes composés de régions rigides, déformables (incompressibles ou non) ou encore musculaires. L'interaction de ces différentes régions en fonction de leurs propriétés physiques, afin de modéliser le comportement de l'objet face à des contraintes externes doit être définie précisément par des lois mécaniques. La bibliothèque de calculs développée pour cela est appelé Phymul. L'objectif visé est de pouvoir simuler, par exemple, le comportement du tronc durant la respiration, ce pourquoi il faut connaître les actions du diaphragme afin d'obtenir des données utilisables en entrée du modèle. En outre, Emmanuel Promayon cherche à recaler le modèle à la réalité, c'est-à-dire intégrer les données réelles de la physionomie du patient et étudier les paramètres à utiliser pour avoir un résultat le plus proche de la réalité possible.

2- Sujet du stage

2.1- Problématique

Objectif fixé : pouvoir modéliser des cellules biologiques (telles des bactéries, par exemple), de façon physiquement réaliste, au sein de leur environnement.

Pour atteindre un tel but, il faut respecter des critères de modélisation déterminés. D'une part, l'outil de modélisation se doit de suivre les lois physiques correspondant aux mouvements des cellules pour simuler correctement leur progression. En effet, les cellules sont des corps élastiquement déformables. L'outil doit donc pouvoir tenir compte de telles propriétés.

D'autre part, le fait représenter une cellule au sein de son environnement ajoute une contrainte, puisque celui-ci est constitué d'autres cellules, ainsi que de la matrice extra-cellulaire. Au cours d'un déplacement, la cellule peut donc rencontrer l'un ou l'autre de ces corps. On a alors affaire à une collision, que l'outil de modélisation doit savoir gérer au cours d'une simulation. De plus, la rencontre de deux cellules présente une certaine particularité. Lorsque les deux membranes cellulaires entrent en contact, une liaison chimique peut se créer entre celles-ci, selon leur affinité. On se retrouve alors avec deux cellules liées l'une à l'autre qui évoluent comme si elles ne formaient

qu'un seul corps. Cette liaison peut cependant être rompue si des forces suffisamment grandes sont appliquées, de manière à séparer les deux cellules. Comme celles-ci sont déformables, elles peuvent se détacher progressivement, la surface de membrane en liaison diminuant petit à petit. Cette surface peut également augmenter si elles subissent des forces les aplatissant l'une contre l'autre. L'outil de modélisation doit donc également permettre la simulation d'un tel comportement.

2.2- Solution apportée

Pour répondre à la problématique, une étude de l'art a démontré qu'utiliser PhymulOb, l'outil de modélisation d'Emmanuel Promayon, était une bonne solution, à condition de le modifier sur un certain point.

En effet, Phymul offrait déjà la possibilité de respecter les propriétés élastiques des cellules de façon physiquement réaliste. De plus, il assurait également la détection des collisions entre les différentes régions modélisées au cours de la simulation. Cependant il ne les gérait pas correctement. Une méthode de gestion avait bien été testée mais elle n'était pas réaliste pour certains points, au niveau physique et biologique.

Au cours de ce stage, j'ai donc effectué une étude de l'art, développée en première partie de ce manuscrit, avant de travailler sur Phymul, présenté en seconde partie. La recherche d'une solution adéquate pour le problème de la gestion des collisions, son implémentation dans Phymul et sa validation par test est le coté le plus technique du travail mené au cours de ce stage que j'exposerai en troisième et dernier partie.

I- Etude de l'art sur les collisions en simulation

On cherche ici à montrer que les autres travaux exécutés en modélisation n'apportent pas d'outil approprié à la problématique.

Cette étude de l'art porte sur deux aspects des collisions dont il fallait tenir compte pour le travail à accomplir. Dans un premier temps, il faut pouvoir détecter la collision, puisque rien sur un objet modélisé n'indique à priori qu'il entre en contact avec un autre objet modélisé. Dans un second temps, il faut gérer la collision. Une fois le contact entre les deux objets trouvé, il faut empêcher ceux-ci de s'interpénétrer (se rentrer dedans). L'interpénétration est en effet contraire à une simulation physiquement réaliste.

1- Détection des collisions

Puisque l'on manipule des objets géométriques 3D, on ne s'intéresse ici qu'à la détection de collection d'objets géométrique dont le frontière externe est connue explicitement. La façon dont sont détectées les collisions est dépendante du type de modélisation utilisé. C'est pourquoi on ne peut pas observer les différentes méthodes de détection sans faire le tour des différents modèles (voir fig.1), comme l'ont montré [LG98].

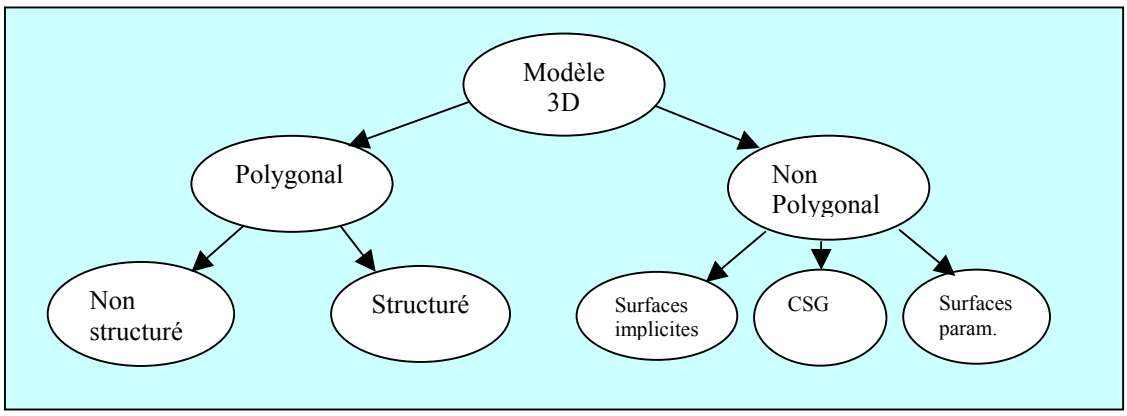

Figure 1 : Organigramme des différents modèles de représentation d'objet en 3D

1.1- Modèles polygonaux structurés

Un modèle polygonal est, comme son nom l'indique, composé de polygones, ou facettes, qui délimitent la surface de l'objet représenté. Pour obtenir un modèle, on doit donc discrétiser cet objet : il s'agit de choisir des points, à sa surface, qui définiront les facettes du modèle. On se retrouve donc avec un maillage de la forme de l'objet. On le connaît, pour les calculs de la simulation, par les points du maillage choisis.

Le problème de détection de collisions est différent selon qu'on a affaire à un modèle déformable ou non.

Les modèles non-déformables sont utilisés pour la représentation de corps rigides. Ils se gèrent par l'exploitation de la cohérence spatiale et temporelle. En effet, connaissant la vitesse, la direction d'un objet et les forces qui lui sont appliquées, on peut calculer sa trajectoire et le moment où il atteindra un certain point de l'espace. Si en ce point se trouve un obstacle, on prévoit une collision.

Des calculs de complexité linéaire permettent d'obtenir des informations supplémentaires sur les points voisins de deux objets polygonaux et donc de prévoir un contact entre ces deux corps en mouvements. Des travaux pour des simulations comportant certaines exigences de modélisation sont menés pour accroître les performances de ces calculs. Ainsi [Bar89] a développé une méthode analytique de calcul des forces exercées entre systèmes de corps rigides restant en contact.

Le problème est différent lorsqu'on a affaire à un modèle déformable. On peut toujours utiliser, pour de petites déformations, la cohérence temporelle. Cependant lorsqu'on a à modéliser un objet tel qu'un ruban, donc très déformable, ou comme des corps soumis à des forces variant aléatoirement dans le temps, cela ne suffit plus. La technique la plus utilisée pour effectuer des tests de recherche de collision est celle des volumes englobant. Il s'agit d'assimiler, au moment du test, le modèle à un volume géométrique simple, tel une sphère ou un pavé. Ce volume est dit englobant car il contient tout l'objet, ajusté de façon à épouser le plus possible son contour. Connaissant la position et la taille de deux pavés ou deux sphères, il est facile de savoir s'ils se chevauchent ou sont en contact. On sait alors que les objet géométriques qu'ils contiennent sont très proches, voire en collision.

Il y a différentes approches d'utilisation des volumes englobant. Ils peuvent être également des cônes ou d'autres polyèdres, alignés selon un axe ou selon la conformation de leur modèle associé. Ils sont souvent employés par hiérarchie, comme dans les travaux de [Got96]. C'est-à-dire qu'il y a différents jeux de volumes, pour plusieurs niveaux de test, associés à chaque modèle. Le premier niveau ne comporte qu'un volume, comme décrit ci-dessus, qui englobe tout l'objet. Aux niveaux suivants, on a plusieurs de ces boîtes pour les différentes parties du modèle, de façon à ce que leur volume total soit de plus en plus proche de celui du modèle. S'il y a une collision détectée à un niveau, on regarde si c'est le cas au niveau suivant et ainsi de suite, sinon il n'est pas nécessaire de faire plus de tests. D'autres techniques sont utilisées lorsqu'on arrive au niveau des polygones composant le modèle. [Mol01] présente ainsi une méthode de test rapide pour savoir si un triangle et une boîte se chevauchent.

1.2- Modèles non-polygonaux

On présente ici trois types de modèles non polygonaux. D'abord il y a les constructions de solides géométriques ou CSG. Il s'agit de modèles construits à partir de formes primitives comme des blocs, des cylindres, des sphères… Elles sont combinées entre elles par le biais d'opérations d'union, d'intersection et de différence. Dans ce cas, le problème de détection de collision est simplifié par le fait que ces modèles comportent des informations sur les jeux d'opérations les définissant. Parmi

ces informations se celles concernant les intersections. S'il n'y en a pas pour deux objets donnés, c'est qu'il n'y a pas de collision entre eux. Cependant, l'accès à cette information demande tout de même des méthodes de calculs efficaces pour lesquelles ont été effectués différents travaux.

Les deux autres types de modèles sont les surfaces implicites et les surfaces paramétriques. Les premières utilisent des fonctions implicites f : ⌋3}⌋ qui définissent des formes dans l'espace. Les surfaces implicites sont l'ensemble des points pour lesquels on a f(x,y,z) = 0, l'intérieur du modèle étant les points pour lesquels f(x,y,z)<0. Dans la même idée, les surfaces paramétriques sont décrites par des fonctions f : ⌋2}⌋3. Pour ces deux méthodes de modélisation, résoudre des problèmes de collision revient donc à manipuler ces fonctions.

Des différents modèles présentés, c'est le type du modèle polygonal qui convient le mieux à la problématique. Moins lourd à utiliser que les différents modèles non-polygonaux, il offre une représentation correcte des corps élastiques que sont les cellules et diverses méthodes de détection de collisions.

2- Gestion des collisions

De la même façon qu'il existe plusieurs types de modèles, il y a différents critères de simulation pou les collisions. En effet, l'état du système après une collision est dépendant des propriétés de l'objet modélisé, du fait des lois de la dynamique. On distingue donc ici les collisions dites instantanées des collisions où le contact entre les objets qui se rencontrent perdure.

2.1- Collisions instantanées

Lorsque deux objets se heurtent puis s'éloignent à nouveau l'un de l'autre sous l'effet de l'énergie libérée lors du choc, on a affaire à une collision instantanée. C'est le cas, par exemple, d'une balle qui tombe et rebondit sur le sol. [Bar89] des méthodes analytiques pour des simulations comportant des corps rigides complexes, telles des chaînes segmentées et articulées ou un alignement de dominos susceptibles de s'effondrer les uns sur les autres. Cependant ce type de collision ne correspond pas à la problématique.

2.2- Collisions avec contact

La situation est différente lorsque les deux objets entrés en collision restent en contact et continuent à évoluer l'un contre l'autre. Un exemple simple est celui d'une bille tombant puis roulant sur un plan incliné. Le problème devient plus délicat lorsqu'on a affaire à des corps déformables : lors du contact leur géométrie se modifie et peut continuer à changer selon l'évolution du système. [Vol98] présente un algorithme

de résolution de collision pour la modélisation de surfaces très déformables, notamment un long ruban tombant sur une surface inégale en s'affaissant sur lui-même.

Cependant aucun travail concernant une simulation durant laquelle non seulement deux modèles entrés en collision restent en contact, mais créent également des liaisons entre eux, n'a été apparemment mené. C'est pourtant le type de collision survenant entre cellules. Pour répondre à la problématique, il est donc apparu qu'il fallait trouver et développer une nouvelle méthode de gestion de collision. En fait, c'est pour Phymul, un outil permettant de décrire des modèles polygonaux élastiques, que cette méthode a été cherchée.

II- Présentation de Phymul

Phymul (Physically-based model for simulation) est une bibliothèque C++. Développé par Emmanuel Promayon ([Pro01]), il permet de modéliser des objets composés de plusieurs régions, reliées ou non entre elles, et de spécifier les forces qu'elles subissent. Il offre ainsi la possibilité de simuler leur comportement pour une durée de temps voulue.

1- Différents types de régions modélisables

Phymul a pour but de modéliser des objets biologiques, qui sont susceptibles de comporter des régions solides (os), élastiques (tissus, cellules), musculaires ou des régions passives (peau). Cette bibliothèque permet donc à son utilisateur de choisir, pour les différentes parties de son modèle, leur appartenance à l'une de ces trois classes. Ces parties auront les propriétés correspondantes.

Ainsi on peut représenter le tronc humain, composé des cotes (os), du diaphragme (muscle), de l'abdomen (aux propriétés élastiques) et des poumons (région passive), pour simuler son comportement durant un cycle de respiration. Pour cela, l'utilisateur de Phymul doit décrire chaque région, en précisant leur type (solide, élastique ou musculaire) et la façon dont elles sont agencées. La contraction des muscles est également fixée par l'utilisateur, qui spécifie aussi quelles sont les régions auxquelles ils sont liés. Celles-ci auront donc, lors de la simulation, un mouvement correspondant au travail musculaire.

Les différentes régions sont composées d'éléments. Décrire une région revient donc à donner la liste de ses éléments. Ceux-ci contiennent les informations suivantes : position dans l'espace, poids, éléments voisins dans la région, existence d'un lien entre cet élément et un élément d'une autre région…

Exemple d'un modèle à deux régions élastiques : début de la description

```
# Phymulob Interface Format
#
# © E Promayon, TIMC (c) 1997-2000
# saving time: 04 Sep 2002, 13:54
nbRegion 2
nbElem 160
viscosityW 600.0000000000
selfCollision yes

beginRegion 0
        type elastic
        nbElem 80
        name en_bas
        mass 75.0000000000
        shapeW 2000.0000000000
```

```
masterRegion yes
cteVol yes

beginElem 0
        coord -13.3460314706 218.3493157961 -1.8795466360
        neigh 3
        neigh 2
        neigh 1
        neigh 5
        neigh 4
endElem

beginElem 1
        coord 104.6391656554 189.1259818065 -0.8545009209
        neigh 0
  neigh 2
        neigh 7
        neigh 6
        neigh 16
        neigh 5
endElem
```

Pour modéliser une cellule, il suffit de l'assimiler à une sphère aux propriétés élastiques (voir fig.2). Les éléments choisis pour la décrire sont ceux qui déterminent la surface de cette sphère. On se retrouve ainsi avec un globe dont les facettes sont triangulaires. Il constitue une des régions du PhymulOb, la seule si on ne veut modéliser qu'une cellule. On peut également modéliser d'autres cellules dans l'entourage de la première, comme étant d'autres régions élastiques de l'objet sans liens entre elles.

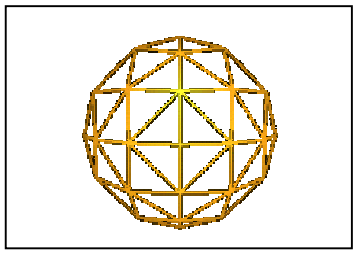

Figure 2 : Représentation d'une cellule sous Phymul

Les forces extérieures auxquelles sont soumises les régions sont elles aussi décrites par l'utilisateur. Il peut s'agir de forces entraînant toutes les parties dans la même direction (la pesanteur, par exemple), d'un point de convergence vers lequel elles vont s'agglutiner ou « mouvement brownien »… Lors de la simulation du comportement de cellules, ces forces sont donc susceptibles de les mettre contact.

2- Fonctionnement

2.1- Héritage

Un objet décrit sous Phymul, appelé aussi un PhymulOb, est donc un modèle discret, au réseau explicite (chaque élément connaît ses voisins) et dont les caractéristiques sont basées sur les propriétés physiques du corps représenté. Dans le cadre de la programmation orientée objet que permet le C++, les différentes sortes de régions du modèle ont des caractéristiques communes dont elles héritent les unes des autres (voir fig.3). Il y a des données et des méthodes communes à tous les types de région, comme la liste des éléments dont elles sont composées et d'autre qui leur sont propre. Chaque type hérite donc d'une classe région regroupant ces éléments communs. Plus précisément, les régions élastiques, solides et passives héritent de cette classe région. En revanche, la région musculaire est une région élastique avec des propriétés supplémentaire et hérite donc directement de celle-ci.

Les types des éléments constituants les régions constituent également des classes à part entière. Leur structure d'héritage est semblable à celle des régions : les éléments élastiques, solides et passifs héritent d'une classe élément et l'élément musculaire de l'élément élastique. Chaque élément connaissant sa région, les calculs peuvent se faire au niveau de l'un ou de l'autre, selon les besoins dans le corps du programme, en tenant compte correctement de leurs propriétés grâce à correspondance entre les deux structures d'héritage.

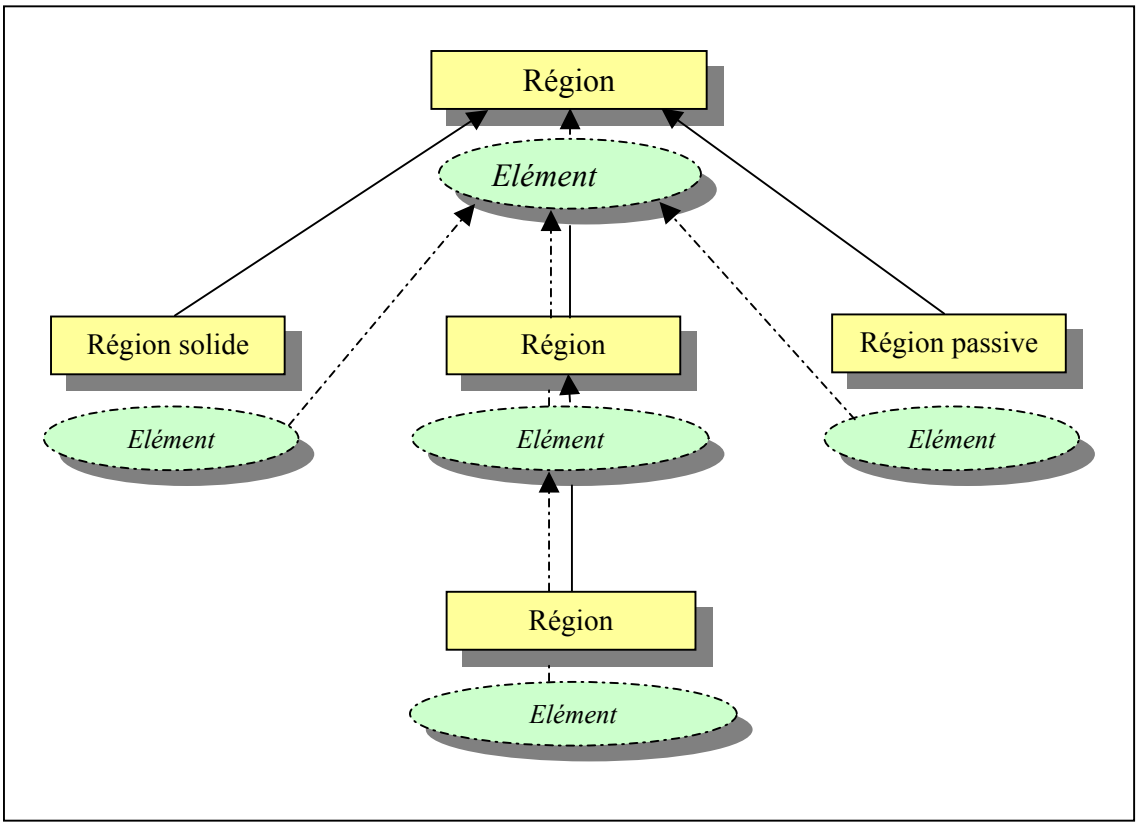

Figure 3 : Structure d'héritage de Phymul

2.2- Elasticité et contraintes

La plupart des réseaux explicites utilisent un système masse-ressort pour modéliser un objet élastique. C'est-à-dire que les éléments décrivant l'objet, qui sont aussi les nœuds de ses facettes, sont des masses reliées entre elles part des ressorts (arêtes des facettes). Lorsqu'un élément s'éloigne de sa position de repos, il y est rappelé par la force générée par ces ressorts.

Dans Phymul, on procède différemment, en définissant une mémoire de forme locale[Pro96]. Les éléments sont toujours des masses, mais la force de rappel est générée autrement. La position de repos de l'élément relativement à ses voisins est toujours connue (donnée par l'utilisateur). Lorsqu'il s'en éloigne, un ressort le relie à cette position et c'est l'élongation de celui-ci qui tend à ramener l'élément. Cette méthode présente l'avantage d'une plus grande stabilité (moins de ressorts).

On a la possibilité d'imposer différents types de contraintes au modèle, localement ou globalement. D'une part, on peut vouloir obliger des éléments d'une région à rester liés aux éléments d'une autre région. En fait, il s'agit de dire « cet élément est en liaison avec cet autre élément ». La nature du lien dépend de la nature des éléments concernés. Pour un couple d'éléments solides, le lien sera équivalent à un ressort. Ainsi un modèle représentant des os et les ligaments les assemblant peut être décrit comme composé de régions solides localement liées. Pour un couple élément élastique/élément solide, il s'agit plutôt, en fait, d'obliger l'élément élastique à rester au niveau de l'élément solide. Les forces appliquées sur l'élément élastique sont transmises à l'élément solide. Le déplacement de celui-ci en est donc affecté et sa nouvelle position est également celle de l'élément élastique. Comme la région musculaire hérite de la région élastique, on voit l'intérêt de cette contrainte pour modéliser une articulation actionnée par un muscle. La contraction du muscle exerce une force sur la partie solide, celle-ci va donc être mise en mouvement et le muscle reste lié de la même façon après ce déplacement.

Une région élastique peut se déformer et, si elle est soumise à des forces la compressant, voir son volume se réduire. C'est pourquoi on a la possibilité de lui imposer contrainte d'incompressibilité pour éviter une simulation non réaliste où un corps verrait son volume devenir quasiment nul sous l'effet de forces trop faible pour induire vraiment un tel résultat.

2.3- Simulation des forces

Pour appliquer les forces et respecter les lois de la dynamique, une masse est attribuée à chaque élément. En fait, l'ensemble des éléments constitue l'ensemble de la masse de la région. On va donc effectuer la somme des forces exercée au niveau de chacun de ces nœuds pour obtenir les déplacements et déformations attendus.

Simuler le comportement du modèle soumis à ces forces demande le calcul, à chaque instant, de leur effet sur son état selon les lois de la dynamique. À « chaque instant » ne veut pas dire en continu. Au contraire, le temps est discrétisé. Connaissant l'état de l'objet à la date t, on va calculer celui dans lequel il se trouvera à la date t+dt. Le pas de temps (dt) doit être suffisamment petit pour approximer un déroulement en

continuité. Cependant on ne peut pas non plus le faire trop petit car cela ralentit d'autant le temps de la simulation.

Le corps principal de Phymul effectuant pas après pas ces calculs est une méthode appelée doMove() comportant une grande boucle d'itération. À chaque itération le nouvel état du système dynamique est calculé. Pour cela, on exécute l'algorithme simplifié suivant :

1. Pour chaque région, pour chaque élément : calculer la somme des forces qui y sont appliquées.
2. Utiliser un schéma de discrétisation sur les équations régissant le mouvement pour calculer un nouvel état intermédiaire qui ne tient pas compte des contraintes.
3. Calculer à partir de cet état les déplacements nécessaires à la vérification des contraintes et appliquer ces déplacements aux éléments concernés.
4. Ajuster les vitesses réelles après ces déplacements imposés.

A la fin de chaque itération, le temps passé depuis le début de la simulation est augmenté de dt. Or ce pas de temps peut ne pas convenir à certain moment où des vitesses de déplacement trop grandes demandent une discrétisation plus fine. Si un problème arrive pour cette raison, on recommence l'itération en prenant un pas de temps plus petit (celui-ci est donc ajustable).

3- Problème des collisions

Dans le but de simuler le comportement de cellules au sein de leur environnement, une méthode de gestion de collisions avait déjà été testée. Si la détection des collisions était correcte, en revanche la façon dont évoluaient les cellules modélisées après être entrées en contact ne l'était pas toujours. Voici ci-dessous la description de cette méthode que le but de mon stage a été de corriger.

3.1- Détection

La détection d'une collision s'effectue à deux niveaux. D'abord, on exécute une détection au sens large pour savoir quelles sont les régions susceptibles d'être en contact, mais qui, en fait, ne sont peut-être que proches. Il s'agit d'un test simple avant une vérification plus précise sur les régions pointées par ce test.

Concrètement, on assimile temporairement chaque région à une sphère. Celle-ci a le même centre géométrique et un rayon plus grand que la distance entre ce centre et l'élément de la région qui en est le plus éloigné. On vérifie alors que ces sphères ne s'interpénètrent pas. Pour cela il suffit de savoir si la distance séparant leur centre est bien plus grande que la somme de leur rayon. Si ce n'est pas le cas, les régions concernées sont soit très proches, soit en contact.

On effectue ensuite sur la liste des régions construites après le premier test, une recherche pour savoir s'il y a vraiment eu collision. Pour cela on utilise un algorithme qui regarde si deux facettes, en l'occurrence des triangles, s'intersectent. Cela arrive si une partie du modèle entre en contact avec une autre partie, puisque celui-ci est composé de facettes triangulaires. Cette vérification est appliquée à deux régions susceptibles d'être en collision. Si c'est le cas, le test donnera en plus les paires de facettes s'intersectant. En effet, cette information est essentielle à l'étape suivante qui est la gestion des collisions ainsi détectées.

3.2- Gestion

Sachant quelles sont les facettes s'intersectant des deux régions en collision, de nouveaux liens vont être créer entre elles. En effet, le but rechercher est de maintenir accolées les parties entrées en contact, puisqu'elles représentent des cellules. Les paires de triangles listées sont donc autant d'éléments qui se retrouvent fixer à d'autres éléments par l'intermédiaire de ressorts.

Pour deux facettes à réunir, on commence par chercher quelle est la meilleure façon de le faire pour que l'élément de l'une se retrouve lié à celui de l'autre qui lui est le plus proche. Ainsi pour une facette A composée des éléments A1, A2 et A3 et la facette B lui correspondant composée des éléments B1, B2 et B3, on va par exemple instaurer un lien élastique (ressort) entre A1 et B3, A2 et B1, A3 et B2. Les triangles sont alors assemblés nœud à nœud de façon à ce qu'il ne puisse plus y avoir continuation de l'interpénétration des régions, tout en les empêchant de s'éloigner l'une de l'autre.

La liaison instaurée doit cependant pouvoir être brisée par une force suffisamment grande qui tend à séparer les cellules. Pour respecter ce critère, à chaque itération on vérifie pour chaque élément comportant un lien élastique si l'une des forces appliquées répond à ce critère. Si c'est le cas, le lien élastique est annulé et les deux éléments qu'il reliait sont à nouveau indépendant.

3.3- Inconvénients de cette méthode

Si la détection des collisions est tout à fait satisfaisante, on ne peut pas en dire autant de la gestion. La façon dont celle-ci est effectuée ne respecte pas certains points des réalismes physiques et biologiques.

Au niveau physique, rajouter des liens élastiques, donc des ressorts dans un système dynamique n'est pas sans conséquences. Cela revient à lui ajouter l'énergie potentielle inhérente à un ressort au repos. Or en réalité, il n'y a pas apport d'énergie extérieure lors de la création de liaison entre deux cellules. À cause de ces ressorts supplémentaires, les cellules modélisées présentent de petites agitations une fois qu'elles se retrouvent accolées lors de la simulation.

Au niveau biologique, deux autres points ne sont pas vérifiés. D'abord si pour une paire de triangles qui vient d'être assemblée, les éléments n'étaient pas face à face, ce qui est généralement le cas, les liens élastiques instaurés constituent une force contraignant les deux facettes à se superposer. Ainsi on observe, lors de la simulation,

une rotation des cellules modélisées au moment où elles viennent de se rencontrer, alors qu'en réalité elles n'ont pas un tel comportement.

Ensuite des liens élastiques ne sont pas suffisants pour empêcher complètement l'interpénétration. Si la force d'attraction est grande, les cellules modélisées s'aplatissent l'une contre l'autre, les liens sont étirés et les surfaces s'intersectent. Ceci est particulièrement visible pour un amas de cellules toutes attirées par la même cible : elles s'y agglutinent en s'aplatissant et s'interpénétrant. Ce n'est pas non plus un comportement observé pour des cellules réelles.

Pour pallier les inconvénients de cette méthode, une idée nouvelle a été développée, implémentée et testée dans le cadre de mon stage.

III- Travail réalisé

1- Principes de la solution apportée

L'idée développée pour remplacer la première méthode et répondre à la problématique est basée sur la création d'un nouveau type de région. Appelée région attachment (en effet, nous utilisons un terme anglais pour nommer cette région), elle hérite de la région solide.

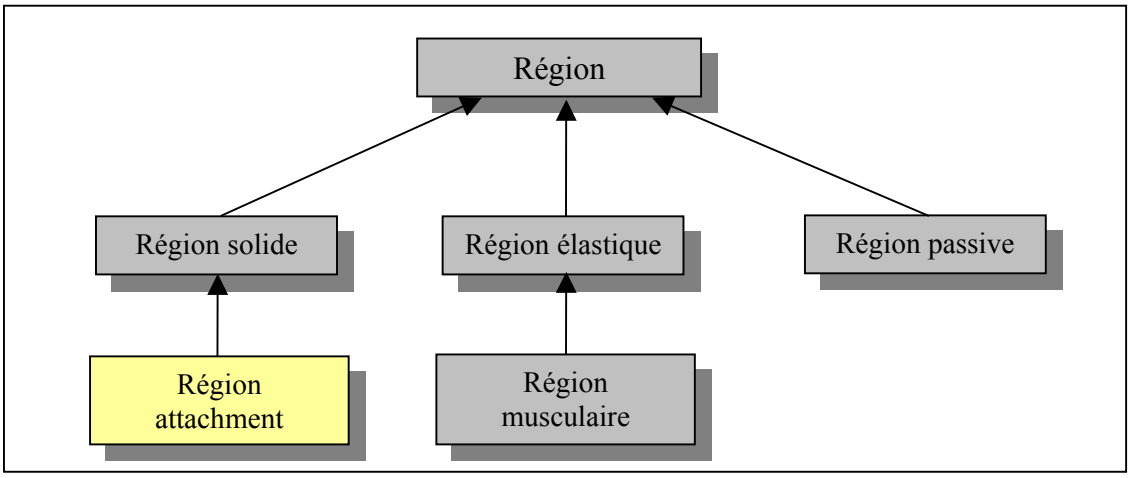

Figure 4 : Nouvelle structure d'héritage

1.1- Intervention de la région attachment

L'objectif est de faire intervenir une région de ce type lors d'une collision. Pour cela, il y a création, entre les deux parties entrées en contact, d'une région attachment à laquelle elles se retrouvent liées. La rigidité de cette région, qui hérite des caractéristiques d'une région solide (voir fig.4), fige l'ensemble des éléments qui lui sont liés. Ainsi il n'y a plus d'interpénétration possible et les cellules restent accolées.

Bien sûr, il faut également pouvoir faire disparaître la région attachment si une force suffisamment grande tendant à séparer les cellules est exercée. Cette disparition assure un retour à l'indépendance des éléments qui étaient figés.

La création de cette nouvelle région ne présente pas les inconvénients de l'ancienne méthode : il n'y a pas d'ajout d'énergie et la présence de la région attachment ne provoque pas de comportement irréaliste des cellules modélisées.

1.2- Caractéristiques de la région attachment

Pour intervenir de la façon décrite ci-dessus la région attachment requiert d'autres caractéristiques que celles acquises par héritage. D'ailleurs, la création d'un nouveau type de région signifie la création du nouveau type d'élément correspondant.

Un élément attachment présente également des particularités par rapport à un élément solide.

D'abord, aucune région attachment n'est présente au début de la simulation, cependant elles peuvent apparaître au cours de celle-ci, contrairement aux autres parties du modèle qui ont été décrites par l'utilisateur. On doit donc la créer lors d'une collision, en lui définissant les éléments qui la composent. Pour cela il faut commencer à chercher quels sont les éléments des régions en collision sont impliqués. Ils le sont s'ils sont suffisamment proches de la région collée à la leur. Pour chacun, un élément attachment va être défini aux même coordonnées et avec lequel il sera lié. Ainsi on crée une région attachment avec sa population (ensemble d'éléments) première.

Au cours des itérations suivantes, si des forces amènent les cellules accolées à s'aplatir l'une contre l'autre, d'autres éléments, proches de la région attachment déjà établie, peuvent se retrouver à proximité. Dans ce cas, on définit autant de nouveaux éléments attachments à ajouter à la population de la région en place. Si au contraire, des forces cassent en partie le lien entre les cellules, cela doit se traduire par la disparition des éléments attachments correspondants. La région attachment voit donc sa population diminuer. Lorsqu'elle devient nulle, la région est détruite.

Non seulement ce type de région peut apparaître et disparaître au cours de la simulation, mais elle est en fait entièrement définie par ses éléments dont le nombre varie (voir fig.5). À cause de ces différences, son implémentation diffère de celles qui ont été faites pour les autres régions.

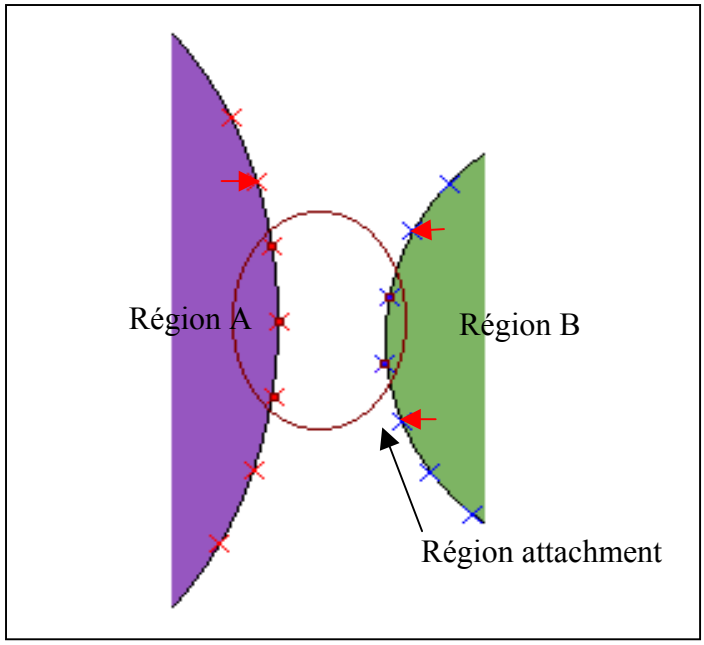

Figure 5 : Construction d'une région attachment entre deux régions élastiques A et B.
Remarque : les flèches rouges indiquent les éléments susceptibles de donner naissance à des éléments attachments si par déformation de leur région ils se rapprochent de l'autre région.

2- Implémentation

2.1- Détection des collisions et recherche d'éléments attachments

La façon de détecter les collisions a elle-même dû être changée pour livrer les informations nécessaires à la création des nouveaux éléments. La première étape de cette détection est toujours la même : utilisation de sphères englobant les cellules pour savoir quelles sont celles susceptibles d'être en collision. On construit alors une liste dite regionCollisionList qui contient des pointeurs vers des paires de régions. Ainsi, en parcourant cette liste on est sûr de regarder tous les couples de régions éventuellement en contact. Cependant, au lieu de chercher ensuite quelles sont les facettes de ces couples qui s'interpénètrent, on dresse une nouvelle liste, cette fois-ci d'éléments. Pour cela on utilise la méthode closeElemList, qui applique l'algorithme suivant pour chaque paire de régions r1 et r2 :

```
Pour chaque élément e1 de r1 :
    Pour chaque élément e2 de r2 :
        si la distance entre e1 et e2 est inférieure ou égale à ε
        alors ajouter e1 et e2 à la liste
```

La liste étant construite, elle contient autant d'éléments pour chacun desquels on va créer un élément attachment. Pour cela, on la parcourt et on crée l'élément attachment en utilisant les données contenues par l'élément courant de la liste, notamment ses coordonnées. C'est aussi pendant cette création que l'on établit un lien entre les deux éléments, lien du type élastique/solide (puisque l'élément attachment hérite de l'élément solide) présenté dans la partie II, paragraphe 2.2.

2.2- Création et fusion de région attachment

Une fois créé un nouvel élément attachment, encore faut-il l'attribuer à une région. S'il y a déjà une région attachment à proximité, on le rajoute à sa liste d'éléments. Sinon il faut en créer une nouvelle dont il sera le premier élément. La création elle-même d'une région attachment diffère peu de celle d'une région solide. Par contre, le phymulob doit connaître la liste des régions dont il est composé. On ne peut pas ajouter la nouvelle région à cette liste qui a été conçue et est utilisée dans l'idée qu'elle resterait invariante une fois établie. De plus, une région attachment n'étant qu'un remède au problème des collisions, ce n'est pas tout à fait comme si elle constituait une partie réelle du modèle. Elle indique plutôt un changement local des propriétés d'une région. C'est pourquoi la liste des régions attachments pour le phymulob est indépendante de celle des autres.

Si deux régions attachments grossissent jusqu'à se jouxter, il est intéressant d'en faire la fusion. On peut le vérifier à l'attribution d'un nouvel élément. En effet, pour savoir comment l'attribuer, on établit la liste des régions attachments voisines. Pour cela on procède de la manière suivante : l'élément élastique auquel il est lié connaît ses voisins, eux-mêmes sachant s'ils sont liés à un élément attachment ; si c'est le cas, on ajoute sa région attachment à la liste, à moins qu'elle n'y soit déjà. On peut ensuite appliquer une méthode doFusion() à cette liste pour n'avoir plus qu'une seule région

attachment localement. Dans ce cas aussi il faut mettre à jour la liste des régions attachments du phymulob.

2.3- Gestion dynamique des régions attachments

Pour prendre en compte le calcul de l'évolution dynamique des régions attachments, il a fallu adapter la méthode doMove() de Phymul. Il n'y avait rien à faire au niveau du calcul des forces appliquées. En effet, les seules forces que ces régions subissent sont celles exercées sur les éléments élastiques auxquels leurs éléments sont liés. Lorsque la somme des forces est effectuée pour les éléments élastiques, l'information est également passée aux éléments solides, et donc attachments, avec lesquels ils ont un lien.

Connaissant les forces extérieures la concernant, on applique à chaque région solide la méthode solidMove() calculant sa nouvelle position, mais sans tenir compte d'éventuelles contraintes. Il suffisait donc d'utiliser la même méthode pour les régions attachments. Il faut noter que les calculs effectués par solidMove() restent valables pour une région attachment parce que leurs éléments ont la même masse que les éléments élastiques auxquels ils sont liés. Cela ne pose pas de problème d'ajout de masse au phymulob car les régions attachments sont listées à part et ne seront donc pas prises en compte dans des calculs concernant les régions du modèle en général.

Plus loin dans la méthode doMove(), on calcule la nouvelle position des régions solides tenant compte des contraintes, par le biais d'une méthode appelée applyConstraint(). Pour les régions attachments, on a surchargé cette méthode pour qu'elle effectue également les tests de ruptures de liens et de suppressions d'éléments. En effet, lorsqu'on doit enlever un élément, il est nécessaire de remettre à jour les données de la région attachment et de recalculer une position par solidMove avant de pouvoir appliquer la méthode applyConstraint implémentée pour les régions solides.

La gestion dynamique des régions attachments n'a donc demandé que peu de modifications du fait de son héritage de la région solide.

2.4- Suppression d'éléments et de régions attachments

Les suppressions éventuelles d'éléments se font donc au cours de la méthode applyConstraint(). Pour savoir si un élément attachment doit effectivement disparaître, on lui applique un test sur les forces subies par l'élément élastique auquel il est lié. Ces forces peuvent induire un détachement des régions accolées au niveau de cet élément. On va donc chercher à savoir quelle devrait être la position du dit élément s'il n'était pas lié à une région attachment. Connaissant le pas de temps et les forces cette position X' est donnée par l'équation :

$X' = X + F*dt^2/M$

où X est la position calculée précédemment, F la somme des forces, M la masse et dt le pas de temps. Si à cette position, l'élément est trop éloigné des éléments de l'autre région, c'est-à-dire que la distance avec le plus proche est supérieure à ε, alors l'élément attachment lui correspondant doit être supprimé et la liste des éléments de sa région attachment remise à jour.

Il se peut qu'on en vienne à supprimer le dernier élément d'une région attachment. Dans ce cas, celle-ci n'a plus lieu d'être et on va pouvoir la faire disparaître de la liste. Cette vérification et cette mise à jour doivent être faites à chaque itération mais leur place dans la boucle importe peu.

3- Tests et résultats

3.1- Rencontre de deux régions élastiques sphériques

Pour tester la solution implémentée nous avons modélisé deux sphères aux propriétés élastiques, pouvant donc représenter deux cellules. Ces régions, de même taille et masse, composées de 160 éléments, sont soumises à une force de pesanteur. La première cellule est déjà tombée sur le sol et s'y trouve donc posée, légèrement aplatie sous l'effet de son poids. La seconde est positionnée au-dessus et est ainsi prête à tomber sur la première (voir fig.6-a).

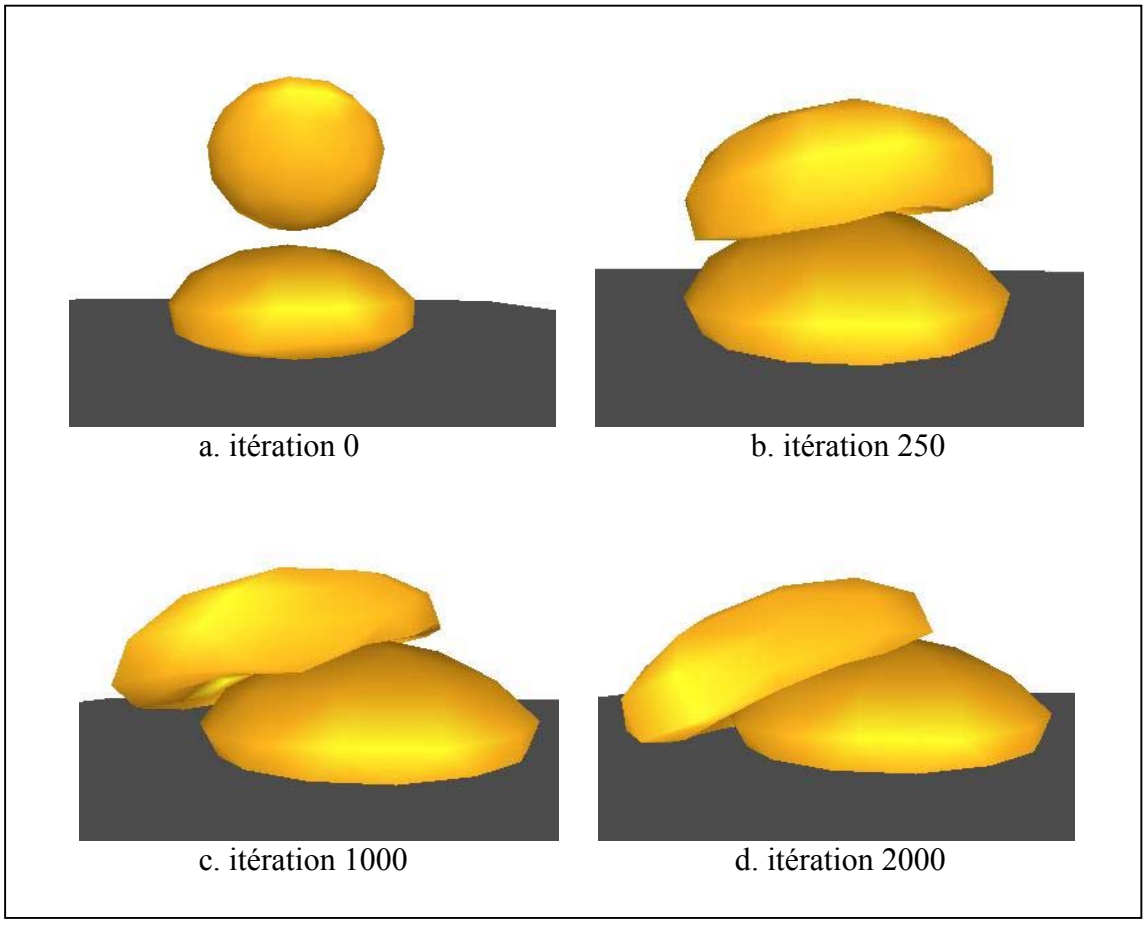

Figure 6 : Simulation de deux sphères élastiques sous l'effet de la pesanteur : images prises à différents stades.

Après lancement de la simulation, le résultat observé est celui attendu. En effet, dans un premier temps, on voit la cellule du dessus commencer à s'affaisser sur celle du dessous, sans interpénétration et sans tomber immédiatement sur le coté car elle y est collée (voir fig.6-b). Cela montre que des éléments attachments ont bien été créés entre ces deux cellules lors de leur rencontre. La cellule du dessus penche cependant d'un coté et la force de pesanteur la fait se déformer. Au bout d'un moment, du fait de cette déformation et cette force, la cellule se détache un peu de celle du dessous pour s'affaisser un peu plus de coté (voir fig.6-c). Précisément, ce sont quelques éléments liés à l'opposé de l'endroit vers lequel elle tombe qui se détachent. Puis cette cellule continue à rouler sur l'autre jusqu'à toucher le sol (voir fig.6-d), se détachant et s'attachant localement dans ce mouvement. La cellule du dessous, quant à elle, s'est trouvée déformée par son lien avec l'autre cellule l'entraînant de coté. Cette simulation montre donc bien que l'utilisation de la région attachment établit, lors de la rencontre de deux cellules, un lien qui n'est brisé que par une force suffisamment grande.

3.3- Limites

D'autres simulations, avec cette fois-ci plusieurs sphères élastiques, ont été appliqués. S'ils ont été satisfaisants, ils ont également permis de souligner la nécessité d'avoir une distance ε ad hoc pour le test de proximité. En effet, cette valeur est pour l'instant une donnée programmée. Elle a été choisie d'après la taille des facettes des sphères. Cependant, au cours de la simulation, du fait de la déformation des surfaces, ces facettes peuvent être localement agrandies ou rapetissées. On peut ainsi constater une légère interpénétration au niveau d'une cellule très déformée, là où ses facettes étaient devenues bien plus grandes qu'au début de la simulation. Le mieux serait d'avoir un ε calculé au moment de chaque test de proximité, en prenant en compte la taille des facettes à l'endroit pour lequel s'effectue la détection.

D'autre part, considérant le comportement des cellules modélisées, mes tuteurs et moi nous sommes demandés si elles ne se détachaient pas trop difficilement les unes des autres dans certains cas. En fait la question est de savoir si le test de rupture est vraiment valide. Il l'est sans doute si on considère des forces opposées appliquées l'une et l'autre à deux cellules liées. Mais dans le cas où les forces ont une direction proche et qui plus est vers l'intérieur d'une région attachment, il n'y aura pas de rupture alors que ce devrait peut-être être le cas. En effet, la position que devrait avoir l'élément lié aura beau être éloignée de celle à laquelle il est contraint, si elle est située à l'intérieur de la même région attachment, même nettement plus loin, cela ne l'autorise pas à se détacher. Il s'agit d'un cas un peu particulier, néanmoins il semble nécessaire d'en tenir compte.

Conclusion

L'implémentation d'une nouvelle région pour l'outil de modélisation que constitue Phymul a été une bonne solution au problème des collisions. Celles-ci sont maintenant gérées de façon correcte pour deux régions élastiques pouvant établir des liaisons à leur rencontre. La région attachment créée permet de figer les zones en contact pour les maintenir accolées. Comme son nombre d'éléments peut varier, on a aussi une simulation correcte de corps se détachant petit à petit, ou au contraire se déformant pour avoir une plus grande zone de contact. Phymul traitait déjà de façon physiquement réaliste l'évolution de corps élastiques, qui peuvent avoir la forme d'une cellule. Avec la modification qui a été apportée au cours de ce stage, cet outil de modélisation peut désormais respecter toutes les propriétés des cellules au cours de leur évolution dynamique. Le travail que nous avons effectué pour l'adapter a donc bien été une réponse à la problématique : modéliser des cellules au sein de leur environnement.

Il reste cependant des problèmes à régler. Une amélioration de la distance utilisée pour le test de proximité des éléments permettrait d'éviter les cas de légère interpénétration subsistant. Prochainement cette distance pour l'instant fixe sera calculée à chaque test à partir de la taille des facettes concernées. Mes tuteurs ont également avancé des idées pour implémenter un test de rupture plus physiquement réaliste. En effet, il y a certains cas pour lesquels, sur les simulations actuelles, les cellules modélisées semblent avoir trop de mal à se détacher. Cela prouve que le simple calcul de la position qu'un élément devrait avoir s'il n'était pas lié n'est pas un critère toujours pertinent. Enfin il faudrait ajuster les transmissions de forces entre les régions attachments et celles qui leur sont liés. Il semble que si la transmission est correcte dans l'autre sens, elle n'est pas vraiment satisfaisante dans celui-ci.

A plus long terme, une fois cette nouvelle méthode bien mise au point et améliorée, on peut envisager d'autres utilisations de Phymul, toujours dans le cadre de modélisations de cellules. L'environnement de celles-ci comporte également la matrice extra-cellulaire, que l'on peut représenter comme une couche élastique. Certains travaux portent sur l'action d'une cellule cette matrice lorsqu'elle y est accrochée et y exerce des forces. Phymul pourrait donc être l'outil approprié pour observer la déformation de la matrice dans un tel cas, au cours d'une simulation.

Bibliographie :

[Bar89] Barraf D. “Analytical methods for dynamic simulation of non-penetrating rigid bodies”, Computer Graphics, Siggraph’89, Boston, Volume 23 Number 3, 1989.

[Got96] Gottschalk S, Lin M.C, Manocha D. “OBBTree : a hierarchical structure for rapid interference detection”, In Proc. of ACMSiggraph’96, Technical report TR96-013, 1996.

[LG98] Ming C. Lin & Stefan Gottschalk. “Collision detection between geometric models : a survey”, In Proceeding of IMA Conference on Mathematics of Surfaces, 1998.

[Mol01] Akenine-Möller T. “Fast 3D triangle-box overlap testing”, vol6 n°1 pp29-33, Journal of Graphics Tools, 2001.

[Pro96] Promayon E, Bacconier P, Puech C. “Physically-based deformations constrained in displacements and volume”, Proc. computer Graphic Forum, Eurographics’96, 15(3):155-164, 1996.

[Pro01] Promayon E, Craighero S. “Object-oriented discrete modeling : a modular approach for human body simulation”, Workshop on deformable modeling and soft tissue simulation, 2001.

[Vol98] Volino P. “Collision and self-collision detection : efficient and robust solutions for highly deformable surfaces”, In MIRALab Copyright © Information 1998.