Chapitre 4

4. Le modèle proposé

4.1. Introduction

Les humains virtuels sont devenus fréquemment utilisés dans les domaines éducatifs et industriels. Les applications usant de ces techniques incluent les jeux vidéo, les fabulations virtuelles, l'industrie cinématographique, le divertissement, les simulations militaires et la modélisation comportementale. [Sps+06]

Par l'humain virtuel nous voulons dire un logiciel qui imite le comportement d'un humain dans un monde virtuel et qui est équipé d'un corps virtuel visualisé par une visionneuse graphique. [Blo+06]

Selon les recherches antérieures, il a été admis que le problème ne se limite pas dans la simulation d'un humain virtuel simple ayant un comportement significatif et crédible, mais il la transcende à la simulation d'un grand monde artificiel possédant un nombre important d'humains virtuels exécutant un comportement crédible qui de plus doit se faire en temps réel. Par ailleurs, il est également admis, que la simulation du comportement de tous les acteurs virtuels par un seul ordinateur, relève plutôt de l'impossible compte tenu des ressources informatique limitées.

C'est pour cette raison que la plupart des applications exhibent des humains virtuels dans de petits mondes artificiels (une chambre ou un village,…), ou exhibent une petite partie du comportement de l'être humain (saisir un objet ou marcher,…..). Par ailleurs, une application ou une technique exhibant une grande simulation serait extrêmement utile dans les domaines des jeux vidéo et dans les fabulations virtuelles. [Sps+06]

En résumé, les problèmes de la simulation de grands mondes virtuels concernent :

- La crédibilité dans le processus de simulation du comportement des humains virtuels.
- La simulation rapide et en temps réel, mais dans les limites des ressources informatiques disponibles.

Notre travail s'est fixé pour objectif la présentation d'un modèle permettant la prise en charge des deux problèmes cités ci-dessus.

Notre modèle présenté s'est inspiré de deux projets de recherche, dont :

- Le premier étant basé sur des techniques d'animation des niveaux de détails.
- Le second s'intéresse à une technique de niveau de détail dans le domaine d'intelligence artificielle LOD IA, pour la simulation de grands environnements virtuels et crédibles.

Ces deux techniques devant concourir à la création d'environnements virtuels peuplés par des humains virtuels.

Le premier volet de recherche est entrepris par le groupe de la synthèse d'image, qui se caractérise par le niveau de détail adaptatif du système d'animation humain (ALOHA). Son but est d'animer et de rendre des humains-virtuels en temps réel [Ps 00]. Le système ALOHA s'est basé sur les limitations du système visuel humain, pour utiliser la technique de niveau de détail LOD afin de calculer :

- Les modèles moins précis quand la perte de précision est peu susceptible d'être notée ;
- Les modèles plus précis susceptibles d'être le centre de l'attention de l'observateur. [MD+02]

Le deuxième volet de recherche touche la technique LOD IA incorporée dans le projet IVE (Intelligent Virtual Environment) qui vise la simulation rapide et crédible du comportement des humains virtuels dans un grand environnement virtuel, mais dans les limites des ressources informatiques disponibles.

Dans la recherche de la solution, nous avons décidé d'utiliser les concepts décrits dans la section suivante :

4.2. Les concepts de base

Dans cette section, nous décrivons les concepts de base du modèle proposé : la hiérarchie des localisations et des processus, les objets, la planification et les règles hiérarchiques if-cond-then et la technique « LOD IA ».

4.2.1. Terminologie

Nous avons également décidé de définir une terminologie plus appropriée pour les humains virtuels. Ces derniers se composent de deux parties :

- La première partie est le corps sans aucune intelligence, c'est lui qui effectue réellement les actions : c'est un *acteur*
- La deuxième partie est le cerveau qui donne l'intelligence aux acteurs. l'humain virtuel l'utilise en tant que sa source dans la prise des décisions : on l'appelle le *centre décisionnel.*
- La troisième partie est la *mémoire* de l'humain virtuel.

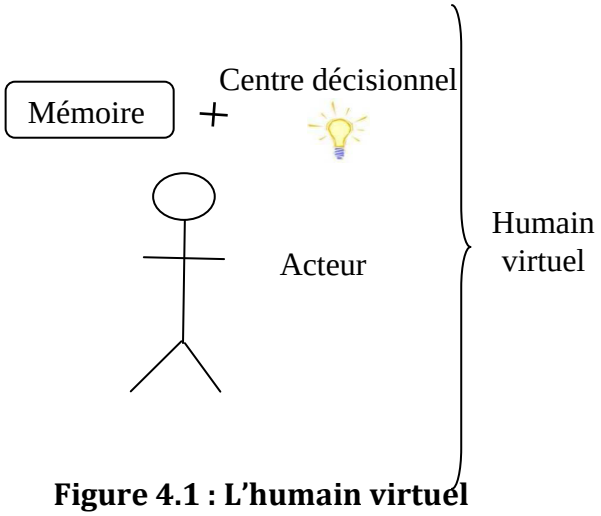


Figure 4.1 : L'humain virtuel

Et le triplet <centre décisionnel, mémoire, acteur> s'appelle un *humain virtuel*. Voir Figure 4.1

4.2.2. La hiérarchie des localisations

Le monde se décompose en des localisations organisées dans une structure hiérarchique arborescente. La racine de la hiérarchie représente le monde entier, ses enfants sont les parties de base du monde. Par exemple sur la figure 4.2 nous pouvons voir un monde qui se compose de deux villes celles-ci se composent à leur tour par d'autres endroits.

Les niveaux de la hiérarchie d'arbre correspondent aux valeurs de niveaux de détails (LOD_value), cela signifie, que nous avons seulement un monde, représente la vue la plus

approximative, ainsi nous pouvons avoir la valeur du niveau de détail du monde(LOD_value) est 1. En voyant les villes, cela nous donne la vue la plus détaillée du monde, sa valeur est donc plus grande- 2 et ainsi de suite.

La décomposition du monde en des niveaux est basée sur des critères prédéfinis par le concepteur (la distance de cette localisation par rapport à la position de l'observateur, la complexité de cette localisation, la périphérie,…….).

- *Si le critère est la distance par rapport à l'œil de l'observateur*

La localisation la plus proche à l'œil de l'observateur possède la valeur de niveau de détail plus grande (elle sera plus détaillée) que celle moins proche (qui sera moins détaillée).

- *Si le critère est la complexité de la localisation*

La localisation la moins complexe possède une valeur de niveau de détail plus grande que la plus complexe.

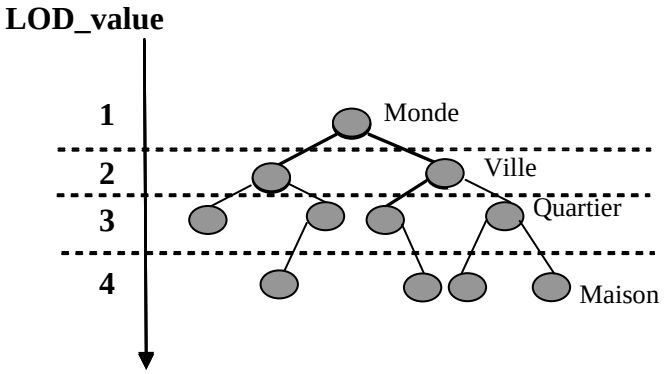


Figure 4.2 : La hiérarchie des localisations consistant en un monde et deux villes

4.2.3. Les objets

Nous pouvons distinguer deux types d'objets :

- Les acteurs : qui sont le corps des humains virtuels –sans le centre décisionnel et la mémoire-.
- Les objets statiques

Les objets sont organisés dans une structure hiérarchique arborescente ; ils peuvent contenir d'autres objets. En fait, le monde se compose d'un seul objet dans lequel d'autres objets sont situés. Cet objet parent peut également produire des objets qu'ils le composent. C'est utile quand nous ne simulons pas une partie du monde virtuel. Si la place où cet objet parent est localisé devient importante ainsi c'est ici ou nous voulons commencer à simuler, l'objet parent a toutes les informations sur des objets qu'il contient et il peut les générer.

Chaque objet est situé quelque part dans la hiérarchie des localisations. La valeur de LOD d'un objet est définie en correspondance de niveau de détails de la localisation dans laquelle l'objet est situé.

4.2.4. La planification et les règles hiérarchiques if-cond-then

La planification est un processus par lequel les humains virtuels prennent leurs décisions dans le monde virtuel afin d'acquérir leurs objectifs.

Dans le processus de planification rien n'est prévu à l'avance. L'humain virtuel décide à chaque fois que le monde change selon l'ensemble de règles qui lui ont été fournies.

Les règles ont leurs priorités et elles sont toutes traitées lorsque l'humain virtuel veut prendre une décision. La première règle qui correspond à l'état du monde (la règle est la première en fonction de sa priorité) est choisie et l'action correspondante est exécutée. Par exemple, l'humain virtuel peut avoir l'ensemble de règles suivantes : (à partir de la plus prioritaire):

1. **Si** le soleil brille, aller à la piscine.
2. **S**'il neige, faire du ski.
3. **Si** cela est vrai, rester à la maison.

Ensuite, à chaque fois que la décision doit être prise, les conditions des règles sont testées à partir du numéro 1 au numéro 3. Si l'une des conditions est vraie, l'action de la règle appropriée est exécutée et le reste des règles est ignoré. Ainsi, si le soleil brille et il neige en même temps, l'humain virtuel va toujours à la piscine et non pas au ski parce que la deuxième règle est moins prioritaire que la première. Si le soleil ne brille pas et il ne neige pas, la troisième règle est toujours exécutée parce que sa condition est toujours vraie.

Les humains virtuels prennent leur décision à partir de:

- Ses propres croyances qui représentent comment il perçoit le monde et comment il traduit ses perceptions à sa propre représentation.
- L'intention ce sont les buts de l'humain virtuel

Pour utiliser la technique de niveau de détail également à un niveau comportemental, les règles **if-cond-then** sont utilisées. Ces règles décrivent l'action dans le monde.

if conditions_sont_complies **do** sous_arbre_des_taches **then** résultat

- **Les conditions** : nous indiquent quand l'action peut être faite (par exemple, je dois avoir des ciseaux pour couper les cheveux),

- **Le sous-arbre des tâches** : sont d'autres règles au niveau inférieur ou actions atomiques qui doivent être exécutées pour atteindre l'objectif (par exemple : à couper la partie arrière, partie avant,…..). Le sous-arbre des taches peut être ignoré.
- **Le résultat** : est l'action à exécuter (couper les cheveux,….) [kkp+06]

4.2.5. La hiérarchie des processus

Un processus est la suite d'actions que l'humain virtuel exécute pour satisfaire son objectif.

Les processus sont organisés dans une structure hiérarchique étroitement liée à la hiérarchie des locations. (Voir Figure.4.3). Chaque processus peut être exécuté d'une façon atomique ou augmentée aux sous-processus. Chaque processus exécuté peut se trouver dans un de ces états :

- *Non-existant* : un super processus est exécuté. Ce processus ne sera pas visible.
- *Atomique* : le processus fonctionne d'une manière atomique. Et il effectue des changements au niveau de l'état du monde virtuel.
- *Augmenté* : le processus est augmenté aux sous-processus. Un tel processus n'effectue aucune action, il attend seulement que ses sous-processus effectuent tout le travail. Cependant, il peut devenir atomique par la suite en fonction des changements de la position de l'observateur par exemple.

Chaque processus situé, quelque part, dans la hiérarchie des processus possède une valeur de complexité (LOD_complex). Les valeurs de complexité des processus correspondent à leur profondeur dans la hiérarchie. (Voir figure.4.3)

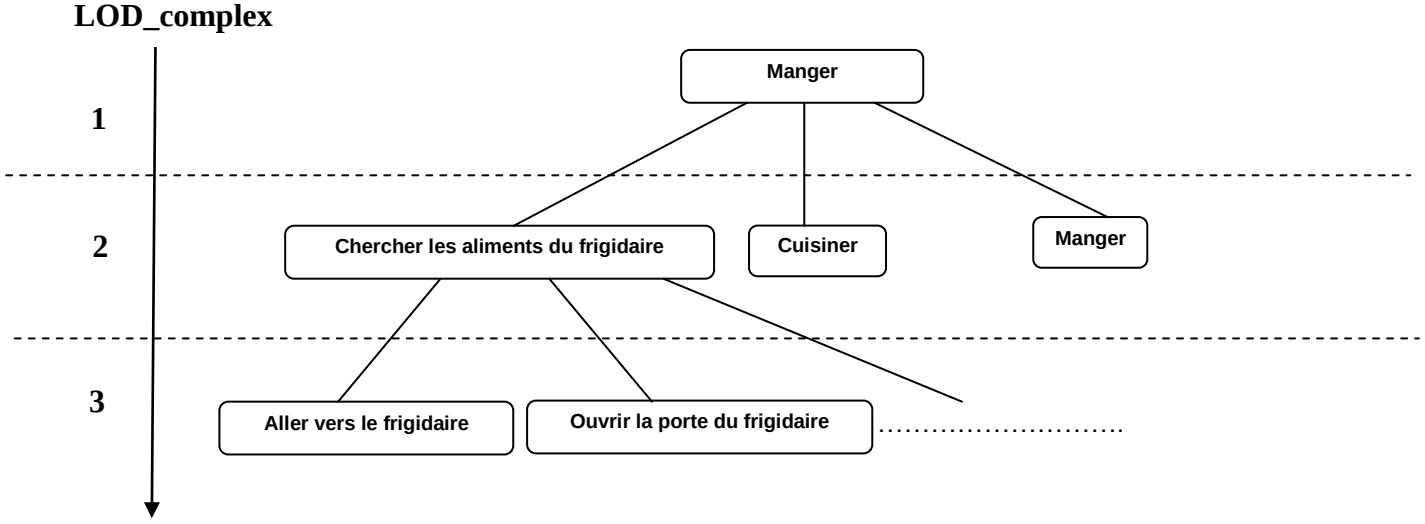


Figure4.3 :La hiérarchie des processus avec leurs valeurs de complexité

Les règles hiérarchiques if-cond-then sont utilisées pour décrire les processus. L'exécution d'une action est guidée par le concept de but. Les processus également n'augmentent pas directement aux sous-processus, mais plutôt aux sous-buts (voir la figure 4.4.a et figure 4.4.b). Dans ce concept, le centre décisionnel d'un humain virtuel obtient la liste de buts requis pour satisfaire le but du processus père et son travail est de trouver et exécuter les sous-processus appropriés.

Figure4.4.a : La hiérarchie des processus sans les buts

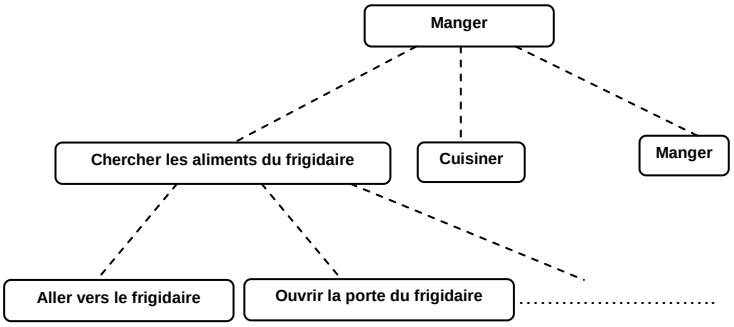
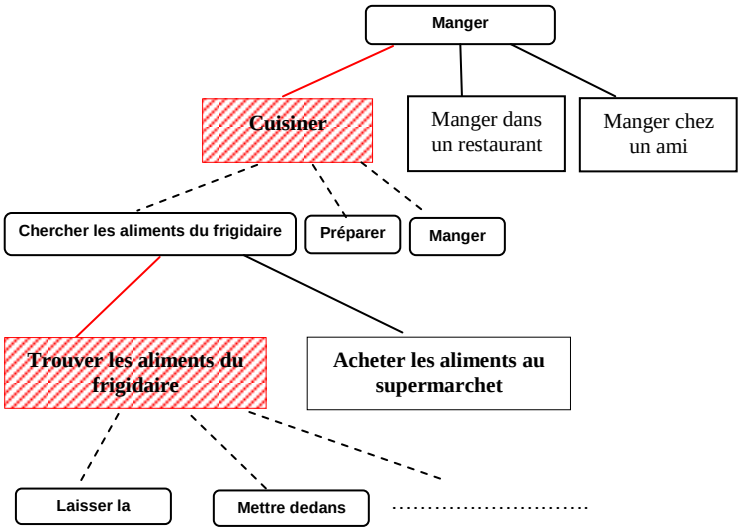


Figure 4.4.b : La hiérarchie des processus-objectif, lorsque le processus agit comme une mise en œuvre d'un but.



4.2.6. Niveau de détail comportemental (LOD IA)

L'utilisation de la technique de niveau de détail LOD au niveau comportemental consiste à effectuer un transfert du domaine des infographies des ordinateurs vers le domaine de l'intelligence artificielle. Cette dernière est basée sur une simple idée qui consiste en ce qui suit:

Il existe souvent peu de places, dans le monde artificiel à un moment donné de la simulation, ou le comportement des humains virtuels est important, celles de moindre importance n'ont pas besoin de simuler le comportement des humains virtuels avec précision, leurs simulation sera dégradée.

Si le comportement des humains virtuels est simulé partiellement, les demandes de la simulation en ressources informatiques peuvent être réduites significativement.

Par conséquent, les problèmes apparaissent au moment ou nous essayons d'identifier explicitement les places importantes(ou la simulation du comportement doit être précise), et les parties du monde moins importantes où la simulation du comportement sera approximative. Ainsi, ceci peut se résumer dans les questions suivantes :

1. Comment identifier les places importantes et celles moins importantes ?
2. Comment l'humain virtuel peut adapter son comportement selon l'importance de son endroit?

L'approche de la technique LOD au niveau comportemental proposée est étroitement liée à la hiérarchie des processus (voir section 4.2.5) et l'utilisation des règles **if-cond-then** (voir section 4.2.4).

4.3. L'architecture proposée

Une vue élevé de l'architecture proposée nous indique quatre modules importants (voir figure.4.5.) : la base de règles, le séparateur LOD, le contrôleur géométrique, le contrôleur d'animation et le contrôleur de comportement.

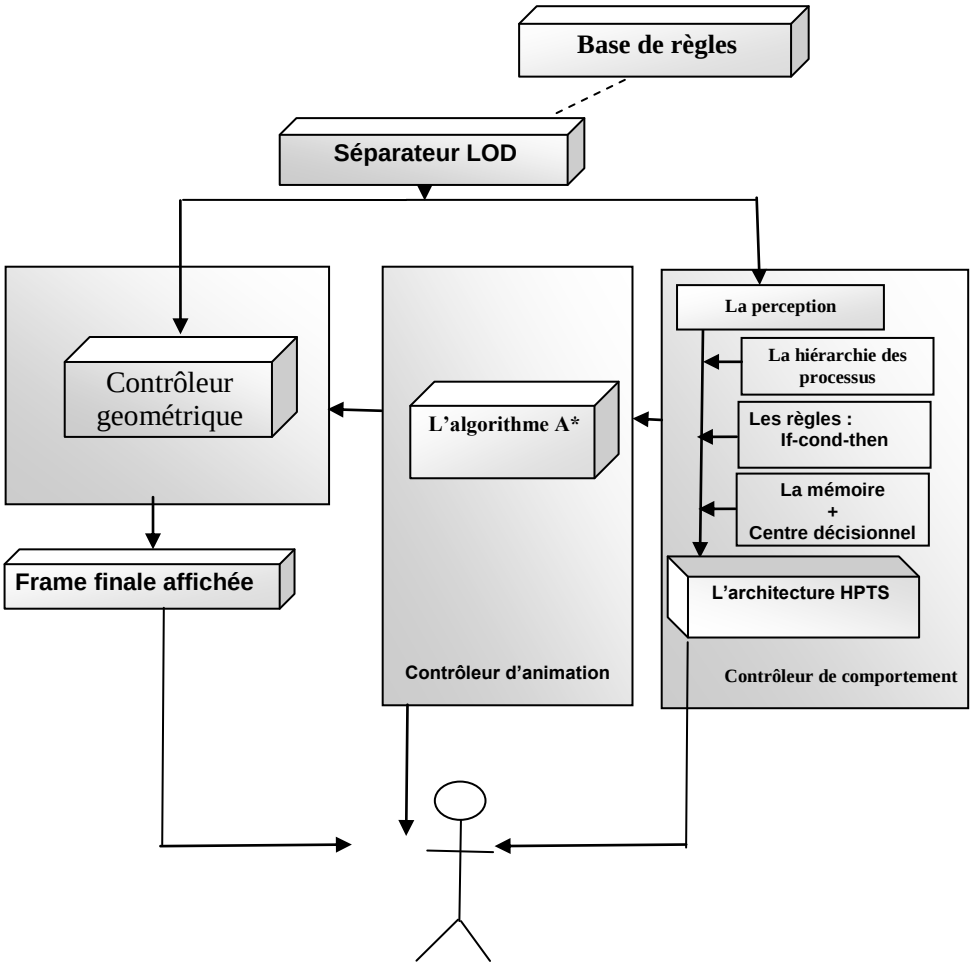


Figure.4.5 : Une vue élevée de l'architecture proposée

4.3.1. La base de règles

La base de règles est fondamentalement un ensemble de règles qui contrôle le niveau de détail pour chaque objet (statique ou un humain virtuel) qui se trouve dans le monde virtuel. L'ensemble de règles peut être réglé par le concepteur pour être en conformité avec les capacités du matériel informatique utilisé. Les règles sont spécifiées pour contrôler le niveau de détail géométrique, le mouvement et le comportement de chaque objet, basé sur des critères simples à savoir : la vitesse, la position dans le plan d'observation, la distance du spectateur,….etc. (voir section 3.2.5.1)

4.3.2. Le séparateur LOD

Ce module est le cœur de l'architecture proposée. Le séparateur LOD ne se limite pas à la définition de la structure hiérarchique et les différents niveaux de détails du monde virtuel, mais il canalise également les paramètres qui guident les contrôleurs géométriques et de

comportement. La communication entre le séparateur LOD et les deux contrôleurs se produit sur la base d'un maître / esclave.

Après avoir extraire les attributs des objets nécessaires et définir leur valeurs de niveaux de détails à l'aide des directives de la base des règles, il envoie les résultats aux contrôleurs. En outre, ce module agit comme une couche d'abstraction entre le contrôle du monde virtuel et sa composition, ce qui aboutit à un système plus extensible.

4.3.3. Le contrôleur de comportement

Notre contrôleur de comportement a la capacité de contrôler des comportements compliqués.

C'est le devoir du contrôleur de comportement pour stocker l'information observée dans la mémoire associée à chaque humain virtuel, pour mettre à jour la connaissance, il utilise ainsi l'ensemble de règle hiérarchiques et il emploie la technique de niveau de détail au niveau comportemental (LOD IA) afin d'établir le rôle approprié pour chaque humain virtuel en utilisant l'architecture HPTS (Voir section 1.2.3.2.10).

4.3.3.1. Mécanisme de sélection d'un comportement selon une intention

Une fois un humain virtuel a commis une intention (un but), il doit agir afin d'accomplir cette intention.

Ces intentions peuvent être hiérarchiquement imbriquées. Plus spécifiquement, afin d'accomplir une intention, on peut avoir des intentions secondaires.

L'algorithme de sélection, d'une action selon une intention, est exécuté par les centres décisionnels des acteurs. L'algorithme du choix d'action selon une intention alloue les intentions des humains virtuels .À un instant donné, une intention est active. Quand l'intention active est atteinte par un processus, une nouvelle intention devient active

Algorithme1 : La sélection d'une action-intention

Cet algorithme est exécuté par le centre décisionnel d'un humain virtuel

En entrée : Un acteur dans une localisation donnée et son centre décisionnel avec un ensemble des intentions

1) I ← ensemble des intentions
2) I_a ← l'intention active (à partir de I)
3) A ← action à exécuter pour satisfaire I_a
4) Passer A à l'architecture subsomption afin d'établir le rôle de l'humain virtuel
5) Lorsque l'exécution de A se termine, allez à l'étape (1)

4.3.3.2. Mécanisme hiérarchique de sélection d'un comportement selon une intention

Afin qu'un humain virtuel atteint son intention, il doit exécuter un processus qui est une suite des actions.

Chaque action est couplée avec un ensemble de paires <sub_Intention,advise> :

- Sub_intention est une intention secondaire que l'humain virtuel doit la commettre afin d'accomplir une partie de l'action.
- Un advise est une règle que l'humain virtuel devrait respecter pour activer les sub_Intention de l'action dans un certain ordre afin que l'intention originale soit satisfaire.

En conclusion, l'action sera exécutée par l'exécution de quelques autres actions qui accomplissent les sub_intention.

Par exemple, les sub_intention de l'action « Arroser un jardin », peuvent être :

(a) sub_intention1 : Trouver « ce qui a les moyens d'exécuter l'arrosage » (un bidon par exemple),

(b) sub_intention2 : Remplir,

(c) sub_intention3 : Trouver « ce qui a les moyens d'être arroser » (un jardin par exemple),

(d) sub_intention4 : Nettoyer « celui qui a les moyens d'exécuter l'arrosage ».

En respectant advise, (c) sera exécuté dans une boucle, pour chaque jardin une fois.

Voir l'algorithme 2 ci-dessous :

Algorithme2 : La sélection hiérarchique d'une action-intention

Cet algorithme est exécuté par le centre décisionnel d'un humain virtuel
En entrée : Un acteur dans une localisation donnée et son centre décisionnel avec un ensemble d'intentions

1) I ← ensemble des intentions et des sub_Intention
2) I_a ← l'intention active (à partir de I)
3) A ← action à exécuter pour satisfaire I_a (à partir de l'algorithme 1)
4) If A est une action atomique, then
   Passer A à l'architecture subsomption
   Lorsque l'exécution de A se termine, allez à l'étape (1)
5) else I← I U A [sub_Iintention] ; allez à l'étape (1)

Dans l'étape (2) de l'algorithme 2, le centre décisionnel choisit l'intention active sur la base de tous les advise des sub_Intention.

4.3.3.3. Niveau de détail d'intelligence artificielle (LOD IA)

Pendant la simulation, des valeurs de complexité de niveau de détail (LOD_complex) sont associées aux niveaux de la hiérarchie des processus par le concepteur. Par exemple, la complexité de LOD d'arroser un jardin peut être 3, alors que l'arrosage d'un lit individuel du jardin peut être 4. Puis, au cours de la simulation, à chaque instant, les valeurs de LOD sont assignées à tous les endroits. En outre, l'étape (4) de l'algorithme 2 est raffinée comme suit :

if A est une action atomique, ou A.LOD_complexe = location.LOD_value, then …..

Cela signifie que les sub_Intention de l'action A ne seront pas transmises au centre décisionnel, si la valeur de LOD est si basse. Au lieu de cela, l'action A sera exécutée comme si elle est atomique.

4.3.4. Le contrôleur d'animation

Le contrôleur d'animation est le responsable qui manipule n'importe quelle animation à n'importe quel niveau de détail spécifié par le séparateur de LOD, pour fournir un cadre extensible. C'est son devoir d'enregistrer les demandes d'animation, pour stocker les informations nécessaires, de mettre à jour les animations en cours et de gérer le stockage des vecteurs d'état pour les objets.

4.3.4.1. L'algorithme A*

Pour faire déplacer les objets d'un point à un autre, dans une scène bidimensionnelle ou tridimensionnelle, le système doit calculer le chemin optimal entre deux points. Dans notre recherche de chemin heuristique nous avons utilisés l'algorithme A* [NS76].

L'algorithme standard de recherche pour le problème de chemin le plus court dans un graphe est l'algorithme A*. C'est une recherche en largeur dirigée qui combine les avantages des autres méthodes, citées précédemment dans le chapitre 02, et qui utilise une fonction de la forme:

f (n) = g (n) + h (n) selon les étapes suivantes:

1. Ajouter le point de départ à une "liste ouverte";
2. Répéter les instructions suivantes:
    a. Rechercher la cellule ayant le plus faible coût ***f*** dans la "liste ouverte", qui devient la cellule en cours;
    b. Eliminer la cellule en cours de la "liste ouverte" et l'ajouter à la "liste fermée";
    c. Pour les 8 cellules adjacentes à la cellule en cours, faire:
        - Si on ne peut pas traverser la cellule, on l'ignore;
        - Si elle n'est pas dans la "liste ouverte", on l'y ajoute. La cellule en cours devient le parent de cette cellule. On calcule les coûts ***f, g*** et ***h*** de cette cellule.
        - Si elle est déjà dans la "liste ouverte", on teste si le chemin passant par la cellule en cours est le meilleur, en comparant les coûts ***g***;

Un coût ***g*** inférieur: signifie un meilleur chemin;

- Si c'est le cas, on change le parent de la cellule pour devenir la cellule en cours, en on recalcule les coûts ***f*** et ***g***. Si on conserve une

"liste ouverte" triée par coût ***f,*** la liste doit être retirée à ce moment là;

d. On s'arrête lorsque:

- La cellule de destination est ajoutée à la "liste ouverte";
- On ne trouve pas la cellule de destination et la "liste ouverte" est vide;

3. Enregistrez le chemin. En partant de la cellule de destination, remontez d'une cellule à son parent jusqu'à atteindre la cellule de départ.

Une vue d'ensemble du système d'animation tout en évitant les collisions est décrite dans la figure 4.6

Pour chaque agent **faire**
Appliquer l'algorithme A* pour chercher itinéraire
Fin pour
Pour chaque pas d'animation comportementale **faire**
Pour chaque agent **faire**
Prédiction de collision
Si pas de collision **alors**
L'agent suit son chemin
Sinon
Appliquer l'algorithme d'évitement de collision
Fin si
Fin pour
Fin pour

Figure 4.6 : Algorithme général du système.

4.3.5. Le contrôleur géométrique

Le contrôleur géométrique encapsule toutes les informations nécessaires pour décrire l'aspect externe de l'objet dans le cadre actuel.

A travers de la communication produite entre les deux contrôleurs (géométrique et de mouvement), un vecteur d'état qui décrit les orientations et les positions des objets est transmis à partir du contrôleur de mouvement vers le contrôleur géométrique. Ce vecteur est ensuite

enveloppé dans les surfaces extérieures, selon le niveau de détail précisé par le séparateur LOD, au niveau du contrôleur géométrique.

4.4. L'environnement choisi

L'implémentation de notre modèle proposé nécessitera la disponibilité d'un environnement physique réaliste (3D) parce qu'il nous permet de réaliser un monde virtuel proche du monde réel qui de plus devra permettre à notre humain virtuel de réaliser des mouvements réalistes proches des mouvements de l'humain réel.

Il y'a lieu, enfin de conclure que dès lors l'étape du choix de l'environnement réalisée, il sera question de trouver le moyen ou bien l'outil qui assure la simulation de toutes les caractéristiques nécessaires pour cet environnement, c'est le moteur physique.

4.4.1. Les moteurs physiques

Plusieurs travaux cherchent à faire une simulation réelles des comportements des humains virtuels dans des environnements physiques réalistes et en leur faisant reproduire des mouvements réalistes également. De ce fait, le seul moyen qui leur permet de garantir ce réalisme réside dans la simulation où l'on a à reproduire les lois de la physique. La simulation crédible du comportement des humains virtuels sur un environnement 3D incluant la physique était alors un travail relativement complexe à faire. Actuellement et depuis les années 2000, nous disposons des outils (moteurs physiques libre) et du matériel (puissance des ordinateurs) nécessaires pour une bonne simulation d'humains virtuels quelque soit leur formes ou les mouvements qu'on voudra leur faire reproduire.

Parmi les moteurs physiques qui sont à l'usage public (ou libre) nous pouvons citer, Breve[1], ODE[2], Newton Dynamics, OGRE[3]et bien d'autres, ceux là étant les plus répandus.

Notre choix c'est retourné vers le moteur physique le plus populaire qui est ODE (Open Dynamics Engine).

4.4.2. Open Dynamics Engine

Open Dynamics Engine (ODE) est une bibliothèque logicielle open source se plaçant dans la catégorie des moteurs physiques. Elle sert à simuler l'interaction physique de corps rigides. Elle est actuellement utilisée dans les jeux vidéo, les outils 3D et les outils de simulation.

[1] www.spiderland.org/
[2] http://ode.org/.
[3] http://www.ogre3d.org/.

ODE est disponible sur plusieurs plateformes (Linux, Mac OS ou Windows) et utilise une interface de programmation en C pour une plus grande compatibilité, bien qu'en interne, le code source soit écrit en C++. Cette bibliothèque possède plusieurs types de jointures et intègre un détecteur de collision avec friction. Le moteur utilise plusieurs intégrateurs en fonction de la précision et de la robustesse de la simulation désirées.

Plusieurs primitives sont disponibles et le moteur peut gérer les surfaces constituées de triangles. [ODE, 2006]

Nous utiliserons les 2 aspects d'ODE, à savoir le moteur physique et la gestion des collisions. ODE donnera un aspect « réel » à l'application.

4.4.2.1. Corps rigide

Un corps rigide simulé avec ODE possède les propriétés suivantes :

Les propriétés qui changent avec le temps sont :

- Position du vecteur (x, y, z) du point de référence du corps, généralement le point de référence doit correspondre au centre de la masse du corps.
- La vitesse linéaire correspondante au point de référence, qui est un vecteur (v_x, v_y, v_z).
- L'orientation d'un corps rigide, représentée par un Quaternion (q_s, q_x, q_y, q_z) ou une matrice 3*3 de rotation.
- Vecteur de vitesse angulaire (w_x, w_y, w_z) qui décrit comment l'orientation change
- avec le temps.

D'autres propriétés sont constantes dans le temps:

- La masse du corps.
- La position du centre de masse (centre d'inertie). Dans les simulations actuelles le centre de la masse et le point de référence doivent coïncider.
- La matrice d'inertie est une matrice 3*3 qui décrit la manière dont la masse du corps est divisée autour du centre de masse.

4.4.2.2. Intégration du temps

Le processus d'une simulation d'un corps rigide nécessite l'intégration d'une contrainte très importante qui est le temps. Chaque pas (step) d'intégration avance le temps courant d'une valeur « step size » donnée qui fait ajuster l'état de tout le corps par la nouvelle valeur du temps.

4.4.2.3. Application des forces

Pour chaque pas de simulation, l'utilisateur peut appeler une fonction qui applique une force au corps rigide. Cette force est ajoutée à une accumulation de forces qui sera remise à zéro après chaque pas d'intégration.

4.4.2.4. Application de jointures

Il existe une relation qui est appliquée entre deux corps (primitives) qui peut avoir certaines positions et orientations par rapport aux autres. Cette relation est appelée contrainte ou jointure. Il existe différents types de jointures pouvant relier les parties d'un corps rigide (primitives géométriques):

- Ball and socket (voir figure 4.7.(1))
- Hinge (voir figure 4.7(2)) ;
- Slider (voir figure 4.7(3)) ;
- Universal (voir figure 4.7 (4)) ;
- Hing-2 (voir figure 4.7(5)) ;
- Fixed ;
- Contact (voir figure 4.7(6))

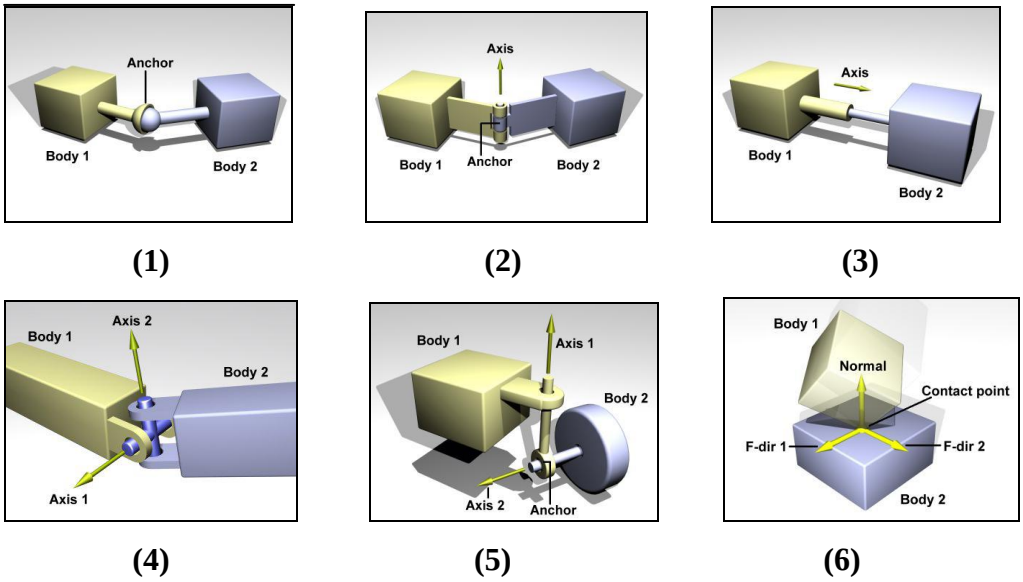


Figure 4.7: Les types de joints

Nous donnons ci-après une explication détaillée sur le fonctionnement de chacune des jointures utilisées au niveau de notre modèle proposé dans le chapitre suivant :

- **Hinge Joint**

La jointure de ce type, contraint le mouvement des deux parties reliées par cette jointure et cela pour les retourner autour d'un axe prédéterminé. Le joint de type hinge tient compte d'un degré de liberté seulement, et c'est la jointure la plus simple. Au niveau de notre simulation des humains virtuels, nous utilisons seulement 2 senseurs :

- angle hinge courant,
- le taux de l'angle hinge courant.

Notant h(t) l'angle courant entre 2 parties du corps à l'instant t, et r(t) le taux de l'angle hinge qui est défini comme la dérivée de l'angle hinge en fonction de temps :

$$r(t) = \frac{d}{dt} h(t)$$

La valeur retournée pour l'angle hinge sera entre - PI et PI, cette jointure possède deux senseurs qui peuvent fournir l'angle courant et la vitesse angulaire du corps, la jointure possède également un effecteur qui affecte la vitesse désirée en tant que son entrée.

- **Ball & socket joint**

Ce type de jointure est le plus complexe des jointures existantes dans ODE, cette jointure tient compte de 3 degrés de liberté autour des 3 axes. De plus, cette jointure fournit 2 senseurs de rétroaction pour chaque axe. Ces deux senseurs fournissent les données suivantes :

- L'angle Ball & socket pour l'axe donné,
- Le taux d'angle de Ball & socket pour l'axe donné.

Le taux de l'angle pour un axe est calculé en prenant la dérivée en ce qui concerne la période de l'angle courant sur l'axe particulier. En raison des 3 degrés de liberté, les 2 senseurs peuvent fournir jusqu'à 6 informations au RNN.

- **Contact joint**

Le joint de contact empêche le corps 1 et le corps 2 de s'interpénétrer au point de contact.

Il fait ceci en permettant seulement aux corps d'avoir une vitesse " sortante " dans la direction de la normale de contact.

Des joints de contact sont dynamiquement créés et supprimés en réponse à la détection de collision. Les joints de contact simulent le frottement au contact en appliquant des forces dans les 2 directions de frottement (friction) qui sont perpendiculaires à la normale [Smi02].

Les joints de contact ont typiquement une vie liée au pas de temps. Ils sont créés et supprimés en réponse à la détection de collisions.

- **Fixed joint**

Un joint fixe maintient une position et une orientation relatives fixes entre deux corps, ou entre un corps et l'environnement statique. L'utilisation de cette jointure n'est presque jamais une bonne idée dans la pratique, exceptée dans le cas où on procède à des corrections. Si on a besoin de deux corps pour être collés ensemble, il est préférable de les représenter comme un simple corps.

4.4.2.5. La détection de collision

ODE possède deux composants principaux: un moteur de simulation de la dynamique et un moteur de détection de collisions. Le moteur de détection de collisions fournit à l'utilisateur des informations sur la forme de chaque corps. A chaque pas de temps, le moteur de détection de collisions détermine le corps qui présente une collision potentielle avec les autres et fournit ensuite à l'utilisateur l'information sur le point de contact. L'utilisateur crée alternativement des joints de contact entre les corps.

- **Geoms**

Les objets géométriques (primitives géométriques, Sphere, Box, Capped cylinder, Ray, Triangle mesh, Simple space) abrégés par Geoms sont les éléments fondamentaux dans le système. Un Geom peut être une primitive géométrique ou un groupe de primitives géométriques qui représente un type spécifique de Geom appelé Space. Space est également similaire au concept World sauf qu'il est appliqué à la collision et non pas à la dynamique.

- **World**

L'objet du monde est un récipient pour les corps rigides et les joints. Les objets appartenant à différents mondes ne peuvent pas agir l'un sur l'autre, par exemple les corps rigides de deux mondes différents ne peuvent pas se heurter. Tous les objets dans un monde existent au même point dans le temps, ainsi une raison d'employer les mondes séparés doit simuler des systèmes à différents taux. La plupart des applications nécessiteront l'utilisation seulement d'un seul monde.

- **Space**

Un espace est un Geom ne pouvant être placé et qui peut contenir d'autres Geoms. Il est semblable au concept de corps rigide du "monde", sauf qu'il s'applique à la collision et non pas à la dynamique.

4.4.2.6. Usage de la carte graphique

ODE est un moteur logiciel, il est complètement indépendant de n'importe quel librairie graphique (i.e. une librairie associée à une carte graphique comme: PhysiX[4]). Cependant, les réalisations exploitant ODE utilisent OpenGL. Cependant, et selon les besoins des utilisateurs de ce moteur, les efforts nécessaires à l'exploitation d'une librairie avec une simulation basé ODE, sera exclusivement à la charge de l'utilisateur.[Nes09]

4.4.3. Librairies graphiques

4.4.3.1. OpenGL(Open Graphics Library)

OpenGL est une API (Application Programming Interface) multi plate-forme pour le développement d'applications gérant des images 2D et 3D. Elle a été développée en 1989 par Silicon Graphics, puis portée sur d'autres architectures en 1993. Depuis 1992, sa spécification est surveillée par l'OpenGL Architecture Review Board (ARB).

Son interface regroupe environ 150 fonctions qui peuvent être utilisées pour afficher des scènes tridimensionnelles complexes en décrivant des objets (caméras, lampes, modèles 3D…) et les opérations que l'on peut effectuer pour les manipuler.

Elle à été implémentée sur la plupart des plates-formes informatiques actuelles (Linux et Unix, Windows, MacOs, BeOS...) et est disponible pour de nombreux langages de programmation (C, C++, Java, Fortran, Ada...).

Du fait de son ouverture, de sa souplesse d'utilisation et de sa disponibilité sur toutes ces plates-formes, elle est utilisée par la majorité des applications scientifiques, industrielles ou artistiques 3D et certaines applications 2D vectorielles. [Nes09]

4.5. Conclusion

Dans ce chapitre, nous avons essayé d'expliquer notre modèle proposé qui consiste en l'utilisation de la technique de niveaux de détails à un niveau comportemental « la technique LOD IA », afin que la simulation des comportements des humains virtuels soit crédible, rapide et en temps réel, mais dans les limites des ressources informatiques disponibles. Pour ce faire,

[4] *La carte physiX d'Ageia (Nvidia), carte dédiée aux calculs physiques.*

nous avons commencé par la définition des différents concepts de base utilisées dans le modèle proposé, et ensuite nous avons défini les cinq modules importants constituant l'architecture proposée, qui sont les suivants :

- La base de règles qui est un ensemble de règles utilisées pour définir les différents niveaux de détails selon des critères déterminés par le concepteur.
- Le séparateur LOD est le responsable de la définition des différents niveaux de détails du monde virtuel.
- Le contrôleur de comportement est le responsable de la définition du comportement approprié pour chaque humain virtuel selon son niveau de détail en utilisant les règles hiérarchiques et la technique LOD IA.
- Le contrôleur d'animation est utilisé pour manipuler n'importe quelle animation à n'importe quel niveau de détail défini par le séparateur de LOD.
- Le contrôleur géométrique décrit l'aspect externe de l'objet selon son niveau de détail.

Ensuite, nous avons présenté les outils utilisés pour réaliser notre implémentation. Dans le chapitre suivant, nous allons discuter et donner les détails de l'implémentation de notre modèle ainsi que les résultats obtenus tout en les analysant.