Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mohamed Khider Biskra
Faculté des Sciences Exactes, des sciences de la Nature et de la Vie
Département d'Informatique

N° d'ordre :…...……
Série :……

Mémoire

Présenté en vue de l'obtention du diplôme de Magister en Informatique

Option**: Synthèse d'images et vie artificielle**

Titre :

Navigation Réactive d'humain Virtuel Par Planification Locale

Par :

GAGUI ALI

Soutenu le :

Devant le jury :

Pr. NourEddine DJEDI	**Prof.**	**Université de Biskra**	**Président**
Dr. Foudil CHERIF	**MCA**	**Université de Biskra**	**Rapporteur**
Dr. Med Chaouki BABAHENINI	**MCA**	**Université de Biskra**	**Examinateur**
Dr. Kamel Eddine Melkemi	**MCA**	**Université de Biskra**	**Examinateur**

Remerciements

Je remercie, avant toute personne, mon encadreur Monsieur **Dr Cherif Foudil**, pour ses précieux conseils et surtout pour sa disponibilité et sa grande aide. qui m'a a suivi et orienté, et que à leur attention, leur disponibilité et leur enthousiasme, j'ai pu arriver au bout de mes peines.

Je tiens aussi à exprimer mes sincères remerciements aux membres de mon jury pour les remarques qu'ils ont pu apporter durant la relecture et la soutenance de mon magister.
Enfin ma famille dans tout son ensemble pour l'aide et le soutien qu'on m'a donné et surtout pour l'appui de ma mère qui m'as bien motivé pour faire mon mieux.

Table de matières

Liste de figures

Résumé

Dans la littérature, le problème général de la navigation est abordé principalement suivant deux perspectives, celle des techniques délibératives et celle des techniques réactives. Les techniques réactives ont été originellement motivées par la nécessité des humains virtuel à pouvoir réagir à l'environnement afin d'éviter les obstacles. Ces techniques nécessitent d'être rapides ce qui limite fortement le temps de calcul disponible tout en évitant de rentrer en collision avec les obstacles de l'environnement. La première difficulté consiste donc à ***structurer l'environnement pour gérer efficacement les requêtes****, comme par exemple trouver un chemin ou détecter les obstacles proches.* ***L'autre difficulté réside dans son peuplement avec des entités pouvant interagir entre elles et avec l'environnement****.*

*Nous avons étudié les différentes méthodes de représentations de l'environnement adaptées au calcul de chemins. Notre choix est fait sur la décomposition cellulaire « grille », cette décomposition cellulaire nous a permis de construire un graphe sur lequel on a appliqué les dernières techniques de calcul de chemin optimum**.** Pour arriver à une navigation réaliste, on a étudié au premier lieu les caractéristiques et les comportements des personnes en déplacement dans un environnement avec des obstacles statiques, dynamiques et imprévisibles. La carte de cheminement est utilisée pour avoir des chemins libres d'obstacles statiques. On a développé notre approche pour l'évitement des obstacles dynamiques et imprévisibles.*

mots clefs *: animation comportementale, modèles réactifs, représentation de l'environnement, planification de chemin, évitement de collision.*

Résumé

Dans la littérature, le problème général de la navigation est abordé principalement suivant deux perspectives, celle des techniques délibératives et celle des techniques réactives. Les techniques réactives ont été originellement motivées par la nécessité des humains virtuel à pouvoir réagir à l'environnement afin d'éviter les obstacles. Ces techniques nécessitent d'être rapides ce qui limite fortement le temps de calcul disponible tout en évitant de rentrer en collision avec les obstacles de l'environnement. La première difficulté consiste donc à ***structurer l'environnement pour gérer efficacement les requêtes****, comme par exemple trouver un chemin ou détecter les obstacles proches.* ***L'autre difficulté réside dans son peuplement avec des entités pouvant interagir entre elles et avec l'environnement****.*

*Nous avons étudié les différentes méthodes de représentations de l'environnement adaptées au calcul de chemins. Notre choix est fait sur la décomposition cellulaire « grille », cette décomposition cellulaire nous a permis de construire un graphe sur lequel on a appliqué les dernières techniques de calcul de chemin optimum**.** Pour arriver à une navigation réaliste, on a étudié au premier lieu les caractéristiques et les comportements des personnes en déplacement dans un environnement avec des obstacles statiques, dynamiques et imprévisibles. La carte de cheminement est utilisée pour avoir des chemins libres d'obstacles statiques. On a développé notre approche pour l'évitement des obstacles dynamiques et imprévisibles.*

mots clefs *: animation comportementale, modèles réactifs, représentation de l'environnement, planification de chemin, évitement de collision.*

Abstract

In the literature, the general problem of navigation is addressed mainly in two perspectives, that of the technical and deliberative techniques reactive. Reactive techniques were originally motivated by the need for virtual humans can react to the environment to avoid obstacles. These techniques need to be fast which greatly reduces the computation time disponible.et its goal while avoiding to collide with the obstacles of the environment. That generate the more difficulty the first challenge is therefore to structure the environment to effectively manage queries, such as finding a path or detect nearby obstacles. Another difficulty lies in its settlement with entities that interact with each other and with the environment.

We have studied different methods of representations of the environment suitable for calculating paths. Our choice is made on the cellular decomposition "grid", this cellular decomposition allowed us to construct a graph on which applied the latest techniques for calculating optimum path. To arrive at a realistic navigation was studied first the characteristics and behavior of people moving in an environment with static obstacles, dynamic and unpredictable. The card is used for routing paths free of static obstacles. We have developed our approach for the avoidance of dynamic obstacles and unpredictable.

ملخص

في الطبيعة، تنقسم مشكلة الملاحة أساسا الى قسمين، التقنيات الفنية والتداولية على رد الفعل. ان الدافع لتقنيات رد الفعل يعود لحاجة الانسان عند التفاعل مع البيئة لتجنب العقبات .هذه التقنيات تحتاج إلى أن تكون سريعة مما يقلل إلى حد كبير من الوقت اللازم للحساب ، مع تفادي الاصطدام مع لمعوقات البيئة . هذا الذي يولد تحديين الاول هيكلة هذه البيئة والثاني هو العثور على المسار و الكشف عن العوائق القريبة. وهناك صعوبة أخرى تكمن في تسوية مع الكيانات التي تتفاعل مع بعضها البعض ومع البيئة.

في هذه الدراسة قد قمنا بدراسة مختلف الاساليب المناسبة لتمثيل البيئة قصد حساب المسارات. وقد قمنا باختيار طريقة التحلل الخلوي على شكل "الشبكة" ، سيسمح لنا هذا التمثيل لبناء الرسم البياني الذي يعتمد على تطبيق أحدث التقنيات للوصول إلى حساب النتيجة المتوقعة هي الطريق الأمثل. للوصول إلى واقعية الملاحة قمنا بدراسة أولا خصائص وسلوكيات الانسان الموجود في بيئة تحتوي مختلف العقبات سواء كانت هذه العقبات ثابتة او متحركة او غير متوقعة. نستخدم الخريطة لتجنيب الانسان العقبات الساكنة. وقد وضعنا نهجنا لتجنب العقبات المتحركة والتي لا يمكن التنبؤ بها.

Introduction générale

Après la deuxième moitié du vingtième siècle, la science a connu un développement dans tous les domaines, les améliorations techniques rapides des ordinateurs, ainsi que le développement des applications de simulation du monde réel en trois dimensions. Cette technologie a connue aussi une évolution appréciable touchant plusieurs domaines parmi eux: l'éducation et apprentissage, la médecine, l'architecture et les jeux vidéo etc.

Le thème de notre recherche, nous oblige à nous intéresser à l'environnement virtuel et à la simulation de l'un du comportement humain (navigation),

La navigation des piétons virtuels à l'intérieur d'un environnement a besoin d'une grande quantité d'informations. Ces données sur l'environnement représentent l'espace navigable et non- navigable, autrement dit on doit tenir compte des formes géométriques (objets statiques).

Ce déplacement nécessite la résolution des problèmes :

- Comment représenter les espaces navigables et non- navigables en 3D ?
- Comment trouver le chemin optimal de déplacement ?
- Comment résoudre le problème d'évitement de la collision avec les objets statiques, dynamiques et éventuellement tous les obstacles imprévisibles ?

Nous essayons par notre étude d'atteindre l'objectif de réaliser un système dynamique applicable sur n'importe quelle scène 3D, représentant les espaces navigables et non navigables et permettant au piéton de naviguer dans un environnement 3d tout en évitant la collision avec les objets statiques, dynamiques et éventuellement tous les obstacles imprévisibles.

Pour répondre à notre problématique et donner des illustrations aux questions posées et atteindre l'objectif prévu, on va utiliser l'un des modèles de représentation spatiale, modèle approximative à base de grille « en anglais Grid ».

Cette approche par décomposition cellulaire permet de réaliser un graphe à partir des nœuds générés par chaque élément de la division cellulaire de notre environnement. Cette approche offre la possibilité de trouver un chemin par l'application des algorithmes de la recherche sur graphe. Pour cela on a retenu l'algorithme A* « A star » qui permet de tracer un chemin optimal de sa position de départ à celle de son arrivée.

Domaines d'applications

Les domaines d'application de la navigation réactive sont variés tels que les jeux vidéo, domaines de la médecine et des soins des phobies.

Validation ergonomique de sites. La création de lieux publiques pose des problèmes d'ordre ergonomique, autrement dit, relatif à la qualité de leur utilisation, des problèmes peuvent se poser quant à la bonne navigation à l'intérieur des lieux. Dans ce cadre la navigation des humain Virtual peut être utilisée pour effectuer des validations sur sur des maquettes virtuelles, les éventuelle problèmes peuvent alors être détectée et corriger avant la construction.

Jeux Vidéo : Avec l'évolution de la puissance de calcule des GPU, cette évolution permet aux concepteurs de jeux vidéo a concentre sur le comportement des personnages par exemple **Autodesk® Gameware Navigation®** Autodesk a beaucoup investi dans la navigation dans les dernières années, la constitution d'une équipe qualifiée, basée à Paris. La société a également concentré ses ressources de développement sur la refonte Kynapse, ce qui a entraîné des annonces relativement moins sur le produit (et de nouvelles fonctionnalités) jusqu'à ce que Gameware navigation a été introduite la semaine dernière.

Robotique : Un robot est une machine équipé de capacités de perception de décision, et d'action qui lui permettent d'agir de manière autonome dans son environnement en fonction de la perception. La navigation d'un robot consiste à déterminer une suite de coordonnées que ce dernier doit suivre pour atteindre une destination donnée.les navigation réactive est ensemble de comportement fonctionnant en parallèle, contrôle le robot ,la navigation réactive de robot supprime les problème dus aux différence entre la réalité et le modèle de l'environnement du robot.

Médecine : dans le développement de simulations chirurgicales en environnement immersif nouveau besoins se font sentir en ce qui concerne les outils d'interaction et de visualisation, dans ce cadre les navigations réactive utilisée pour améliorer la navigation dans réseaux vasculaire dense.

Plan du mémoire

Ce document est organisé en deux parties. La première partie est un développement de l'état de l'art sur tous les aspects qui entrent dans le domaine de l'animation comportementale . Nous y aborderons aussi des sujets inhérents à l'étude du comportement humain, et se décompose en trois chapitres:

- Le premier chapitre présente des représentations de l'environnement adaptées au calcul de chemins et les algorithmes de planification de chemin sont présentés.
- Le second chapitre presente les différents modèles de comportement d'humain virtuel (la définition ,les proprietés …) ainsi les différentes architectures de simulation d'humains virtuels.
- Le troisième chapitre présente les modeles de comportement de navigation réactive d'humain virtuel , le processus de detection et d'évitement de collision sont traités.

La seconde partie du document est dédiée à la présentation des travaux effectués dans le cadre de ce mémoire. Elle se divise en deux chapitres:

- Le quatrième chapitre présente notre modèle de navigation reactive d'humain virtuel par planification locale. L'architecture ainsi que les détails des modules de notre système sont présentés.
- Le dernier chapitre présente l'implémentation et la validation des résultats de notre système de simulation.

Chapitre 1 :
Représentation de l'environnement et planification de chemin

Chapitre I

Représentation de l'environnement et planification de chemin

1.Introduction

La modélisation d'environnement sous forme de bases géométriques en trois dimensions devient de nos jours de plus en plus courants. Toute personne, a déjà vu, et ce depuis de nombreuses années, un bâtiment, une ville, une maison ou toutes autres formes architecturales modélisées en 3D. Et ce, dans les films d'animations (Disney), dans les jeux vidéo ou tout simplement dans les effets spéciaux de films récents.

La nécessité d'une description topologique, permettant de représenter et d'organiser l'environnement de simulation. Cette description topologique peut être plus ou moins fine, exacte ou approximative, et peut éventuellement contenir des informations sémantiques. Les algorithmes utilisés en animation comportementale pour représenter l'environnement sont étroitement liés à ceux employés en robotique [Lat91], domaine où cette représentation tient un rôle prépondérant.

Cette description est ensuite exploitée par des algorithmes de recherche de chemin. Ceux-ci peuvent se baser sur différentes informations issues de l'environnement, qu'elle soit géométriques comme les distances, ou qualitatives comme la charge cognitive associée au chemin. Cette planification de chemin constitue un premier pas vers des modèles de décision plus poussés, ouvrant la voie à une gestion comportementale individuelle.

2. Représentation des parties statiques de l'environnement :

Il existe deux grandes familles de techniques de représentation de l'environnement la décomposition en cellules et les cartes de cheminement.

2.1. Décomposition en cellules

L'espace libre est découpé en zones, correspondant à des cellules. Les relations de proximité sont représentées grâce à un graphe. Ces décompositions peuvent êtres exacts ou approximatifs.

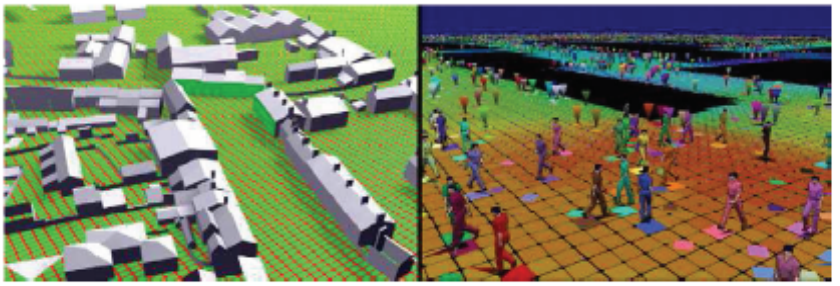


Figure.1 Utilisation de grille en animation

2.1.1. Décompositions approximatives :

L'approche la plus immédiate consiste à discrétiser l'espace de manière uniforme, suivant des figures régulières, le plus souvent des carrés [LD04] ou des rectangles.

a. *Grille uniforme*

La décomposition uniforme permet d'accéder rapidement à un point connaissant ses coordonnées. Cependant, elle souffre de certains défauts. Une cellule pouvant à la fois contenir un obstacle et une partie accessible, des informations utiles sont perdues. Ainsi, pour augmenter la précision, il faut diminuer la taille des cellules. En conséquence, la quantité de mémoire utilisée augmente de manière quadratique. Il est donc nécessaire de trouver un compromis sur la taille des cellules pour avoir une précision et une occupation mémoire satisfaisantes.

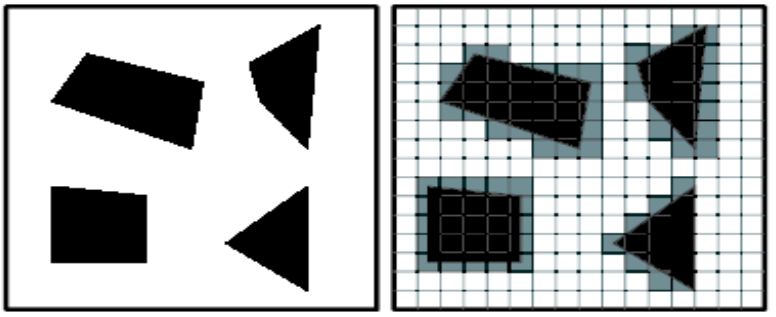


Figure.2 Exemple de grilles uniforme

Pour réduire ce problème, une évolution de ce modèle est apparue sous la forme des grilles hiérarchiques.

b. *Grilles hiérarchiques*

L'utilisation de grilles hiérarchiques consiste à découper l'espace grâce à un arbre de cellules. Les cellules est entièrement libres ou entièrement occupées sont conservée telle quelle. En revanche, les cellules mixtes sont de nouveau divisées. Le processus s'arrête quand les cellules ont atteintes une taille minimale. Cette technique permet donc une compression des données puisqu'elle ne rajoute de l'information qu'aux endroits où c'est utile.

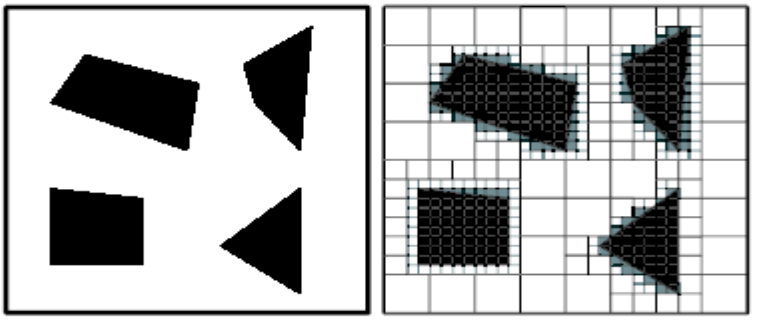


Figure.3 Exemple de grilles uniforme

c. *Décomposition par rectangle :*

Cette méthode de décomposition consiste à représente l'espace sous la forme d'une collection de rectangle, ces rectangles alignés sur les axes, et recouvrent tout l'espace et ne partagent que des frontière. Cette décomposition (les rectangles) représente une zone de navigation libre c a d une zone contenue à l'intérieur d'un obstacle ou bien une zone comprenant à la fois une zone de navigation et une partie d'obstacle.

L'avantage de cette méthode de permettre de créer des cellules de taille variable en fonction de la complexité de la géométrique de l'environnement.

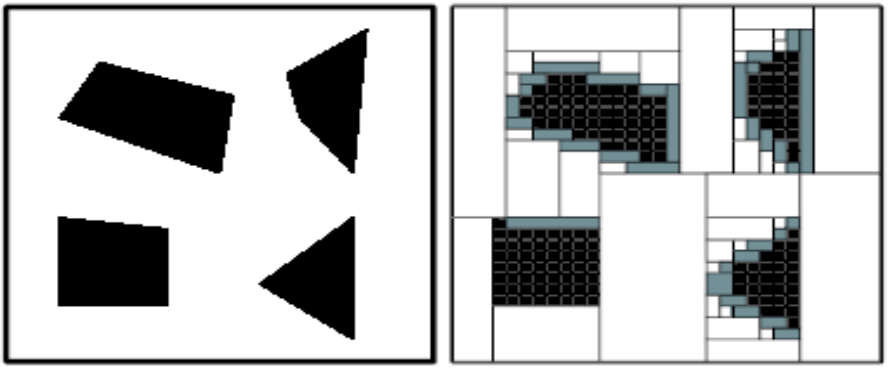


Figure.4 Décomposition par rectangle

2.1.2. Décomposition exacte :

Le but de ce méthode et de représenter exactement l'espace libre sous la forme de cellules convexes (découper l'espace navigable en cellules convexes) de différentes formes (triangle…..)

Il existe plusieurs modèles pour effectuer cette décomposition, la plus utilisée qu'est la triangulation de Delaunay, ainsi qu'à ses évolutions.

a. *Triangulation de Delaunay :*

En mathématiques et plus particulièrement en géométrie algorithmique, la **triangulation de Delaunay** d'un ensemble **P** de points du plan est une triangulation DT(**P**) telle qu'aucun point de **P** n'est à l'intérieur du cercle circonscrit d'un des triangles de DT(**P**). Les triangulations de Delaunay maximisent le plus petit angle de l'ensemble des angles des

triangles, évitant ainsi les triangles "allongés". Cette triangulation a été inventée par le mathématicien russe Boris Delaunay (1890- 1980) en 1934.

D'après la définition de Delaunay [PPD07], le cercle circonscrit d'un triangle constitué de trois points de l'ensemble de départ est vide s'il ne contient pas d'autres sommets que les siens. Ainsi, les autres points sont autorisés sur le périmètre en lui-même mais pas à l'intérieur strict du triangle.

La condition de Delaunay affirme qu'un réseau de triangles est une triangulation de Delaunay si tous les cercles circonscrits des triangles du réseau sont vides. Ceci constitue la définition originale en deux dimensions. En remplaçant les cercles par des sphères circonscrites, il est possible d'étendre la définition à la dimension trois.

Il n'existe pas de triangulation de Delaunay pour un ensemble de points alignés. De toute manière, la triangulation n'est dans ce cas pas définie.

Pour 4 points concentriques, tels que les sommets d'un rectangle, la triangulation de Delaunay n'est pas unique. Trivialement, les triangulations utilisant chacune des deux diagonales satisfont la condition de Delaunay.

Il est possible de généraliser cette notion pour des mesures non Euclidienne, sans garanti de l'existence ou de l'unicité de la triangulation.

La propriété la plus intéressante de cette triangulation [LD04] est que chaque point y est relié à son plus proche voisin par l'arête d'un triangle. Ainsi, cette triangulation peut être utilisée pour représenter l'espace navigable, les points d'entrée étant issus des obstacles. Une autre utilisation possible de cette triangulation est cette fois dynamique lors de la simulation [BY95], pour calculer un graphe de voisinage entre les entités mobiles, qui serviront cette fois-ci de point d'entrée.

b. *Triangulation de Delaunay contrainte en 2d*

Cette triangulation peut être utilisé pour effectuer une subdivision spatiale en triangle [KBT03] .les contraintes expriment alors les arêtes des polygones délimitant les obstacles peuplent les environnements, permettent d'effectuer une subdivision d'environnements sous forme de triangle.

Le nombre des arrêtes et de triangle générer la relation de Euler [BY95] le nombre de triangle utilisé pour la subdivision spatiale est donc linéaire en fonction du nombre de points, indiquant que la discrétisation est donc directement proportionnelle à la complexité géométrique de l'environnement. Cette propriété présente un avantage certain comparativement aux méthodes approximatives, où la discrétisation est fonction de la précision désirée de la représentation

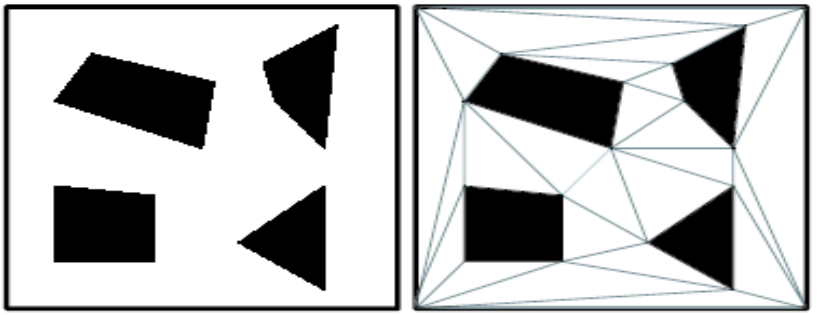
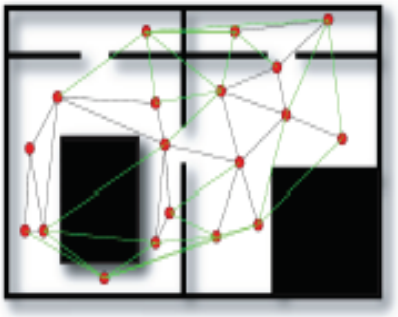
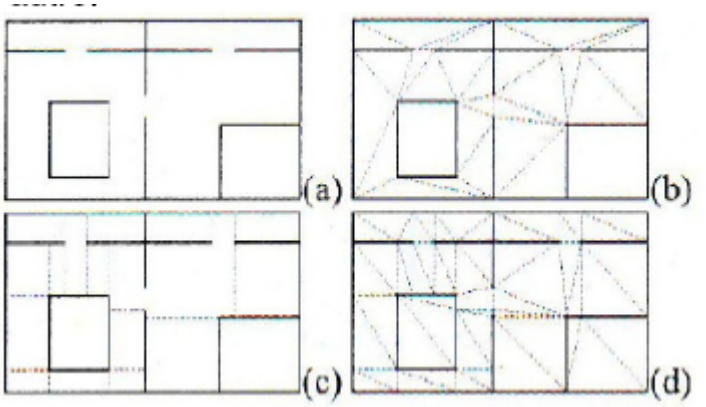


Figure.5 Exemples de triangulations de Delaunay contrainte

c. *Triangulation de Delaunay filtrée*

La triangulation de Delaunay filtrée [Lat91] est une extension de la version contrainte. Deux types de filtrages sont proposés, l'un ayant pour effet d'augmenter le nombre de triangles produits afin d'affiner la représentation de l'environnement, l'autre de le diminuer afin de tenir compte de la visibilité pour l'élaboration d'un graphe de voisinage. Premièrement, concernant son application à la subdivision spatiale, la triangulation.de Delaunay est filtrée par l'ajout progressif de contraintes représentant les goulets d'étranglement (Figure(d)). Ainsi, en considérant l'ensemble des arêtes produites, on est sûr de représenter tous les rétrécissements présents dans l'environnement. Deuxièmement, concernant son application aux graphes de voisinage, la triangulation de Delaunay est filtrée sur un principe équivalent à la triangulation contrainte, en supprimant les arêtes intersectées des obstacles de l'environnement (Figure (f)). Ainsi, cette triangulation peut servir à déterminer trivialement quels sont les voisins directs visibles d'une entité, et avec un simple parcours de graphe peut sélectionner les entités visibles à une certaine distance.





(f)Filtrage pour graphe de voisinage.
Seules les arêtes noires seront conservées, n'entrant pas en collision avec les obstacles de l'environnement.

(a):
l'environnement au départ.

(b):
après la triangulation de Delaunay

(c):
distance minimum entre les angles et les murs.

(d) :
triangulation de Delaunay en prenant en compte la distance minimum.

d. *Décomposition en trapèzes :*

Permet de décomposer l'environnement sous forme de cellules trapézoïdales [LD04], elle est basée sur un algorithme de balayage. La géométrie de l'environnement délimité par des point tries suivant l'axe des cordonnées. Les segments généré (maximum 2 segment), ayant les point sélectionné comme origine et pour extrémité la prochaine intersection avec le bord d'un polygone délimitant un obstacle.

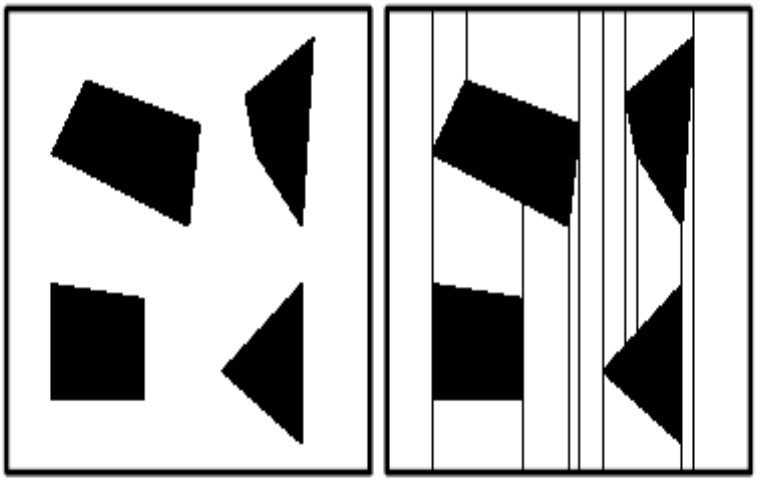
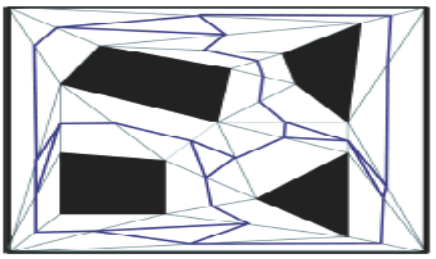


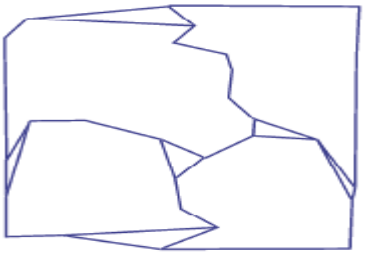
Figure.7 Décomposition en trapèzes

2.2. Cartes de cheminement :

Une carte de cheminement est une représentation implicite de l'environnement [PPD07], particulièrement bien adaptée pour la recherche de chemins. Ces techniques utilisent un graphe dont les sommets correspondent à des points clés, c'est à dire à des positions où l'humanoïde peut tenir debout sans entrer en collision avec l'environnement. Si une arrête existe entre deux sommets, il est possible de naviguer d'un point à l'autre.



Calculs sur l'environnement d'origine



Exemple de carte de cheminement

Figure.8 Cartes de cheminement

2.2.1. Cartes de cheminement déterministes.

Les sommets d'un graphe de visibilité correspondent aux coins des obstacles. Les sommets du graphe sont reliés par une arrête s'il est possible d'aller d'un point à l'autre en ligne droite, sans rencontrer d'obstacle. Dans le cas d'espaces très ouverts, cette technique produit de grands graphes.

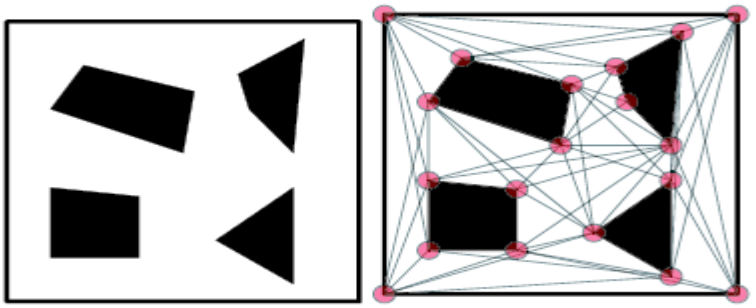


Figure.9 graphe de visibilité

2.2.2. Les diagrammes de Voronoï

Peuvent être calculés à partir d'une triangulation de Delaunay. L'objectif est de partitionner l'espace grâce à des lignes équidistantes des obstacles les plus proches. Les intersections de ces lignes constituent les points clés de la carte de cheminement. Les chemins calculés à partir de cette représentation sont assez éloignés des obstacles, contrairement à ceux obtenus à partir du graphe de visibilité. Dans le cas où l'environnement a été subdivisé en cellules, il est possible de se ramener à une carte de cheminement. Un point clé est placé sur le milieu de chaque segment libre des cellules, puis un chemin est établit entre les points clés de chaque cellule.

La génération de cartes de cheminement [LD04], chaque obstacle constitue un site l ($A_{i)}$ les frontières de chaque zone $V(A_i)$ constituer la carte de cheminement .les points clefs sont définis comme étant les point équidistants d'au moins trois obstacles, c a d à l'intersection d'au moins trois zones, Le chemin alors générer maximiser la distance aux obstacles

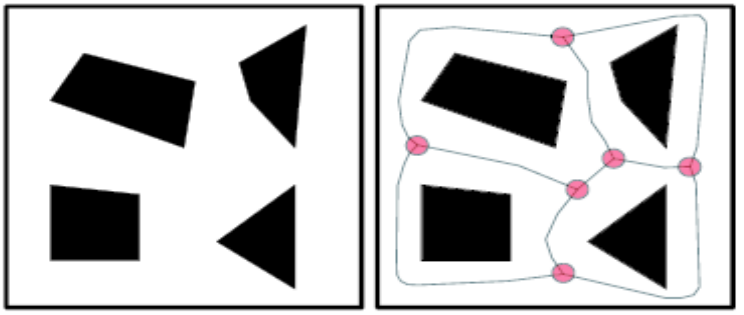


Figure.10diagrammes de Voronoï généralisé

2.2.3. Cartes de cheminement probabilistes

Plutôt que de définir les points clés de manière déterministe en s'appuyant sur les caractéristiques de l'environnement, il est possible de prendre ces points au hasard dans l'espace. Deux points sont reliés s'il n'y a pas d'obstacles entre eux. Il est nécessaire de trouver un compromis sur le nombre de points pour ne pas alourdir les calculs tout en évitant de créer des zones isolées.

3.La planification de Mouvement :

Afin d'exploiter une représentation de l'environnement, qu'elle soit approximative ou exacte, La planification de mouvement est un champ de recherche très étudié en robotique, Cette planification cherche une trajectoire de mouvement dans un environnement, généralement présenter par un graphe, donc La planification consiste à calculer un chemin permettant à l'humanoïde de relier un point à un autre en passant par des zones navigables.

Les différentes techniques utilisées sont fortement dépendantes de la façon dont l'environnement est représenté.

3.1. La planification globale

3.1.1. Recherche dans un graphe

Quelle que soit la représentation globale utilisée, les principaux algorithmes de recherche performants. L'algorithme de *Dijkstra* détermine le chemin le plus court d'un

nœud du graphe à chacun des autres.Il est notamment utilisé lorsqu'aucune heuristique viable ne peut être faite sur l'environnement. Cette technique comporte généralement un coût trop important pour être appliquée à la recherche d'un chemin entre uniquement deux nœuds du graphe. L'algorithme A* ou IDA* permet de trouver le chemin minimisant un coût de cheminement entre la configuration initiale et la configuration cible. Il nécessite d'avoir une bonne heuristique.

3.1.1.1. Algorithmes *de parcours*

Différents algorithmes de parcours de graphe peuvent être utilisés dans le cadre de la planification de chemin.

- **A) Dijkstra :**

L'algorithme de Dijkstra permet de trouver l'ensemble des meilleurs chemins entre deux nœuds du graphe. Cet algorithme utilise une file de priorité où il stocke pour chaque nœud exploré deux informations :

– la distance parcourue depuis le nœud de départ jusqu'au nœud courant .

– le nœud précédent, i.e. parcouru avant le nœud courant.

L'algorithme commence avec un nœud source correspondant généralement à la position courante de l'entité, qui n'a donc aucun prédécesseur et un compteur de distance nul. Cet algorithme fonctionne directement sur le graphe topologique, qu'il soit explicite ou implicite .Afin de trouver un chemin entre deux nœuds, en admettant que la longueur de ce chemin soit l, l'algorithme explorera par propagation circulaire l'ensemble des nœuds se trouvant à une distance inférieure à 1.

Les intérêts majeurs de cet algorithme si un chemin existe il sera forcément trouvé et son abstraction totale de la destination.

Son point faible est sa complexité calculatoire, étant de l'ordre de O(A+ N. logN) où A est le nombre d'arcs du graphe, et N son nombre de nœuds.

- **B)A* :**

Dans l'animation comportementale la performance de la recherche est primordiale, comme en animation (jeux vidéo) l'algorithme A* [Nil82] est plus utilisé. Celui-ci fonctionne d'une façon similaire à l'algorithme de *Dijkstra*, mais en ajoutant un coût prédictif aux nœuds correspondant au reste du chemin à parcourir. Le coût total est donc la longueur

réelle du chemin jusqu'au nœud, incrémentée par une valeur prédite pour le chemin menant au but, calculée par une heuristique.
L'algorithme va ainsi parcourir les nœuds dans l'ordre croissant de leurs coûts associé. La rapidité de convergence de cet algorithme est directement dépendante de la qualité de l'heuristique employée. Dans le cas de la recherche de plus court chemin, l'heuristique employée est une estimation de distance.

L'avantage de cet algorithme réside dans sa rapidité calculatoire. Son inconvénient majeur réside dans le recours à une heuristique. En effet, plus l'évaluation de la longueur de chemin sera complexe, pouvant faire intervenir d'autres paramètres que la distance (par exemple un coût associé à la sémantique), plus cette heuristique sera difficile à expliciter. Ainsi, il est difficile d'utiliser cet algorithme pour des planifications de chemin.

- **Évolutions du A* :**

Un certain nombre d'évolutions ont été proposées pour améliorer les propriétés du A* B. Logan et N. Alechina [LA98] proposent l'algorithme ABC (pour A* with Bounded Cost), permettant d'ajouter des contraintes molles à respecter lors de la planification, comme des limitations de temps ou d'énergie.

- **L'algorithme IDA***

L'algorithme A* reste très gourmand en espace. Des variantes de l'algorithme A* sont alors apparues pour réduire l'espace exploré comme c'est le cas pour IDA*.

Le principe de l'algorithme de recherche IDA* utilise une approche différente une recherche en profondeur dans le graphe supervisé par une heuristique.

Une borne F de longueur est initialisé avec la distance estimé au but(une distance maximale de parcours)en commencent l'exploration par le nœud d'origine du chemin et explorer successivement puis évalue récursivement les voisins jusqu'à ce que l'une des deux conditions suivantes soit vérifiée : soit le chemin est trouvé il s'agit du chemin optimal soit un nœud est parcouru dont le coût est supérieur à F. Dans le cas ou aucun chemin n'est trouvé la borne F est remise à jour avec la petite estimation de la distance au but évaluée précédemment. cet algorithme est plus lent que le A* traditionnel, mais nécessite moins de mémoire.

❖ **L'algorithme HPA* :**

[BMS04] (Pour Hierarchical Pathfinding A*) consiste à redécouper – abstraire – le graphe de l'environnement afin de hiérarchiser la planification. En suite, l'algorithme propose d'associer des pré-calculs de plus courts chemins entre des points clefs de chaque partie abstraite. Ainsi, cet algorithme va pouvoir évaluer un chemin de haut niveau en ne manipulant que peu de nœuds.

L'algorithme de Djiksra est peu utilisé dans le cadre de la recherche de chemin. Le choix entre A* et IDA* est préféré à l'algorithme A* dans le de domaines de taille exponentielle [Lat91] car la taille de la liste triée à l'algorithme A* devient elle –même Exponentielle.
Mais dans la cadre de la planification de chemin, cette considération n'a pas de signification, autre vue que ces deux algorithme possèdent des complexités équivalentes mais (IDA* nécessité moins de mémoires).

3.1.2. Champ de potentiel

Fajen [FWTK03] utilise une méthode de navigation inspirée de la physique, l'environnement étant représenté par un champ de potentiel. Les obstacles produisent des forces de répulsion et le but produit une force d'attraction. L'humanoïde est soumis à ces forces, qui l'attirent vers le but, tout en évitant les obstacles locaux. Cependant, cette approche peut donner une mauvaise trajectoire au niveau global. En effet, il est possible de rencontrer un minimum local, c'est à dire un point où les forces d'annulent, provoquant un blocage. Des méthodes probabilistes sont alors utilisées pour tenter de sortir de cette situation.

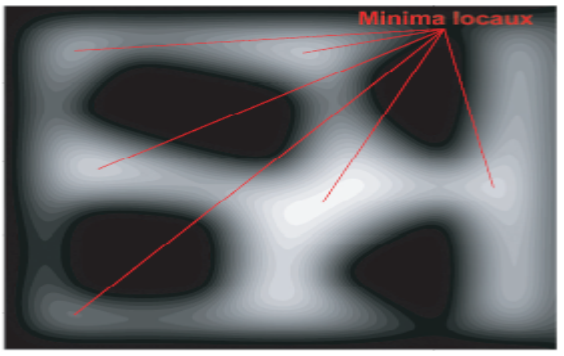


Figure.11 Une carte de champ de potentiel
en noir les obstacles (répulsion),

4. Conclusion

La représentation de l'environnement joue un rôle important, d'une part pour la planification de chemin, mais dans certains cas pour la détection de collision. Nous avons mis en évidence l'importance du choix de la représentation de l'environnement pour la navigation d'humanoïdes virtuels. Il est primordial d'une part d'éliminer les informations inutiles pour obtenir des calculs rapides et d'autre part de conserver les données importantes pour avoir un comportement réaliste. Plusieurs étapes sont nécessaires pour faire naviguer un humanoïde vers un point de l'espace. Un chemin grossier est calculé, puis affiné pour mieux correspondre à la réalité. Enfin, les modèles proposés tendent à modifier localement la trajectoire pour éviter les obstacles dynamiques.

Chapitre II : Le Comportement humain

Le Comportement Humain

1.Introduction :

Si l'on s'intéresse au comportement humain, il est nécessaire de s'intéresser à un certain nombre de sujets tels que la compréhension des mécanismes sous-tendant les fonctionnements du langage (production et perception), de la mémoire, de la perception, du contrôle musculaire ou encore des émotions. En résumé, il faut s'intéresser au fonctionnement des différentes facultés qui constituent ensemble l'esprit humain, sans oublier leur relation avec le corps.

2.Modéliser le comportement :

Pour pouvoir animer de manière réaliste un être humain virtuel, il est nécessaire disposer d'un modèle de comportement d'un être humain réel dans son environnement avant d'adapter ce modèle à l'informatique. Nous allons donc considérer en premier lieu des généralités constituant la base de tout travail sur le comportement, puis nous mettrons en valeur les points importants de la modélisation du comportement du point de vue de l'animation comportementale.

2.1. Généralités

Un être vivant évolue dans un environnement et va interagir avec celui-ci. Il peut par exemple se déplacer, saisir un objet, ou encore recevoir de l'information concernant son environnement par l'intermédiaire de ses sens. L'étude du comportement concerne classiquement l'observation de la manifestation de ces activités. Une représentation largement répandue d'un agent évoluant dans un environnement est présentée sur la figure 12. La perception permet à un agent de recevoir des informations sur son environnement. Il peut alors décider d'un but à atteindre, et va pour cela effectuer des actions qui modifieront

l'état de l'environnement. Cette première représentation constitue la brique fondamentale du travail de modélisation du comportement. Nous allons présenter quelques études réalisées sur le comportement humain ainsi que les architectures qui en ont été déduites.

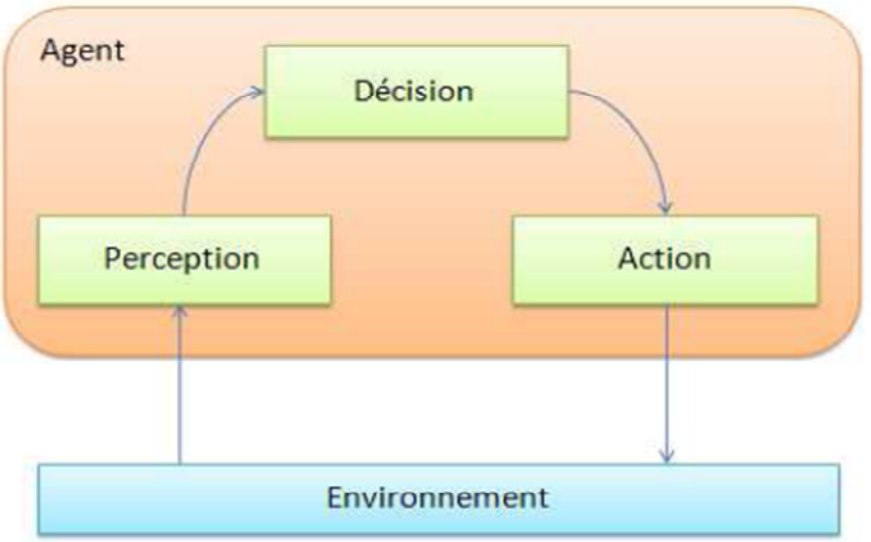


Figure 12. Schéma Perception -Décision- Action

3. Modélisation du comportement :

L'animation d'un humanoïde est généralement décomposée en cinq niveaux (Figure 13) les niveaux géométrique, cinématique, contrôle du mouvement, réactif et enfin cognitif.

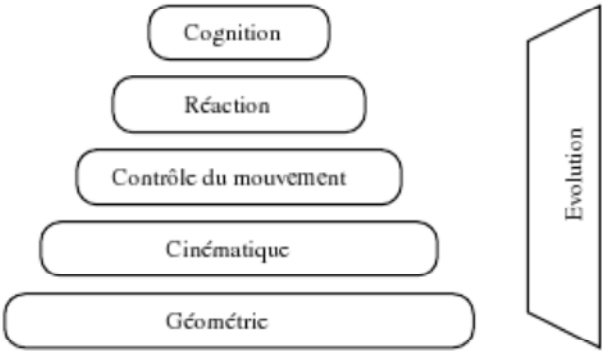


Figure.13.Pyramide de l'animation selon Terzopoulos [Lam03]

Les trois premiers niveaux peuvent être assimilés à la représentation d'un humanoïde, la dimension comportementale est caractérisée par les deux niveaux supérieurs que sont la réaction et la cognition.

On distingue généralement un comportement cognitif d'un comportement réactif selon respectivement l'usage ou le non-usage explicite d'une représentation des connaissances.

Nous nous intéresserons aux architectures réactives et cognitives puis nous terminerons ce chapitre par une conclusion.

4.Architectures réactives

Les modèles réactifs sont des modèles d´écrivant pour l'humain virtuel des comportements reliant directement la perception à l'action, sans modélisation abstraite des connaissances et sans raisonnement. Il existe trois grandes catégories de modèle réactif :

- **stimulus/réponses.**
- **règles de comportement.**
- **automates d'états finis.**

4.1. stimulus/réponses :

Les modèles stimuli-réponse se basent sur un modèle du comportement dans lequel les actions sont des réponses directes à une stimulation de l'environnement. Ils utilisent dans la plupart des cas des réseaux de neurones formels qui représentent en fait une fonction mathématique dont les paramètres sont les valeurs issues de la perception de l'environnement.

et dont le résultat indique l'action (ou les actions) à réaliser. Le principal avantage de cette approche est que les réseaux de neurones peuvent subir un processus d'apprentissage à partir d'une base d'exemples de couples (perception, action) qui permet par la suite au système de réagir de façon similaire à celle apprise lorsqu'une situation comparable se présente. L'inconvénient majeur de ce système est qu'il n'est pas interprétable. En effet, un réseau de neurones est constitué de neurones lies entre eux par des transitions ayant chacune une valeur représentant le poids de cette transition et il n'est pas possible pour un utilisateur de comprendre le lien entre ces valeurs et les sorties du réseau à cause de sa

complexité. Par conséquent, effectuer une modification du comportement nécessite de recommencer tout le processus d'apprentissage. D'autre part, cet effet boîte noire fait qu'il n'est pas possible de prévoir les réactions de l'entité et que donc seuls des tests permettent mesurer la qualité du système obtenu après apprentissage.

4.2. Les modèles à base de règles :

L'approche règles de comportement consiste à appliquer une règle de comportement qui aura été sélectionnée en fonction de la perception de l'environnement (c'est-à-dire en fonction des informations en entrée). Une des solutions pour stocker les règles est de les mettre dans un arbre de décision où chaque branche représente un comportement. Ces règles sont sélectionnées par un algorithme de parcours d'arbre. Cette approche apporte un niveau d'abstraction plus élevée que la première. Cependant, le cœur de son problème est la pondération des différents comportements. Il existe pour cela plusieurs solutions : soit un choix fixe de l'ordre des règles, soit le parcours d'un arbre des possibilités pondéré, ou soit l'utilisation d'une hiérarchie d'experts.

Dans le premier cas, on ne peut pas spécifier de comportements complexes, dans le second, on peut ne pas privilégier toujours la même règle et dans le dernier cas on peut confronter plusieurs comportements concurrents, mais en laissant le choix à un expert de plus haut niveau. Cette approche a le défaut de se limiter à un environnement limité, voire

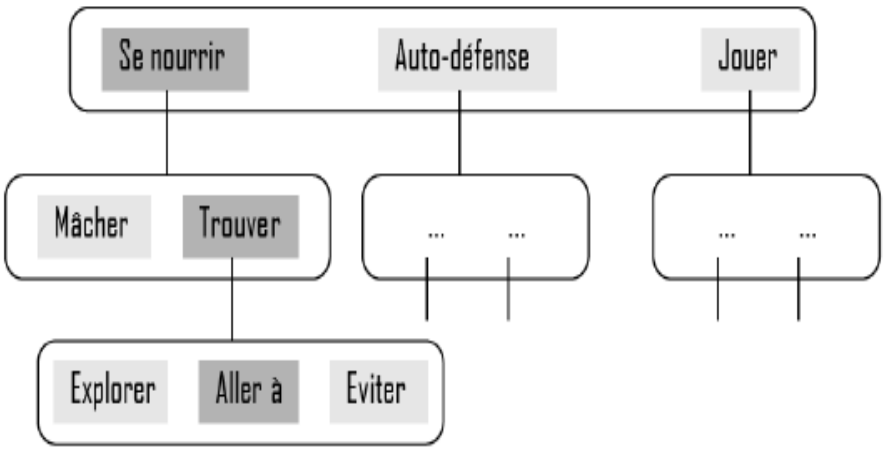


Figure.14 Arbre de décision

4.3. Les automates :

Les modèles à base d'automates permettent d'introduire des comportements reposant sur un enchainement conditionnel d'actions. En effet, chaque ´état de l'automate correspond à une tâche considérée comme unitaire et le passage d'un état à l'autre par les transitions de l'automate permet de modifier l'état et donc le comportement de l'agent. On a donc bien un système pour lequel l'enchainement des actions dépend du contexte perçu mais aussi de la dernière action effectuée. Différentes approches ont permis d'augmenter le niveau de complexité du comportement d´écrit par des systèmes à base d'automates et aussi d'introduire plusieurs niveaux d'abstraction. En premier lieu, les piles d'automates [NT97] ont apporté la notion d'appel de fonction. Lorsqu'un automate en appelle un autre, le premier est empilé alors que le second est lancé.

Une fois que le second est terminé, le premier est dépilé et reprend. D'autre part, les automates parallèles [BW95] permettent la description de comportements complexes car le comportement à une instante donne est le résultat du produit de plusieurs automates décrivant des comportements simples. Enfin, les automates parallèles hiérarchiques ajoutent une structure à un groupe d'automates parallèles. Au sein de cette structure, chaque automate père peut soit superviser l'exécution de ses fils, soit effectuer un filtrage au niveau des propositions générées par ses automates fils. Le modèle HPTS++ [LD02], basé sur des automates parallèles hiérarchiques, a pour particularité d'ajouter des notions de ressources afin de permettre la gestion des exclusions mutuelles inhérentes aux systèmes parallèles et de permettre la mise en place de priorités afin de donner plus d'importance à certains comportements. Par rapport aux approches précédentes, l'intérêt majeur des automates est la possibilité de représenter des comportements reposant sur des enchainements d'actions. Par ailleurs, deux grands avantages de ces systèmes résident dans la puissance du formalisme des automates à états finis ainsi que dans la complexité des comportements rendus possibles grâce aux différentes approches présentées.

4.4. Synthèse

Les modèles réactifs, malgré l'absence de raisonnement abstrait, permettent d'exhiber des comportements relativement complexes. Les systèmes stimuli-réponse se détachent particulièrement des autres approches par leur opacité à la compréhension et par le fait qu'ils peuvent être construits à partir d'un ensemble d'exemples de réactions, sans

connaissances a priori sur le comportement à construire. Les systèmes à base de règles sont facilement interprétables et modifiables mais manquent du pouvoir d'exprimer un comportement comportant un enchainement d'actions au cours du temps, contrairement aux approches à base d'automates.

5.Les modelés cognitifs :

Apres avoir décrit les modèles réactifs, nous présentons dans cette section différents modèles dits cognitifs. Les systèmes cognitifs, contrairement aux systèmes réactifs, utilisent une représentation abstraite du monde afin de pouvoir planifier une suite d'actions leur permettant d'atteindre un but souhaité. Ils vont se projeter dans l'avenir afin d'évaluer les conséquences de leurs actions et de comparer ce résultat avec le but qu'ils recherchent.

5.1. Calcul situationnel et dérives :

Le calcul situationnel [MH69] est un formalisme utilisant six concepts pour décrire des environnements complexes ainsi que les actions qui peuvent y être réalisées. Les concepts sont décrits dans la liste ci-dessous.

- **Les situations** : une situation correspond à l'état du monde à un instant donné.
- **Les fluents** : un fluent représente une propriété du monde.
- **Les causalités** : une causalité décrit une relation de cause à effet et permet de déduire la valeur d'une propriété à partir d'une autre.
- **Les actions** : une action est décrite par des pré-conditions et des effets et permet de modifier une situation.
- **Les stratégies** : une stratégie est une suite d'actions spécifiées par le concepteur dans le but de décrire un raisonnement scripte.
- **La connaissance** : elle sert à modéliser le fait que les entités connaissent ou non L´état de certaines propriétés du monde.

Ce formalisme permet de d´écrire un but sous la forme d'une expression booléenne portant sur des fluents et la planification consiste alors à déterminer une suite d'action permettant d'atteindre une situation dans laquelle cette expression est vérifiée. Un environnement de programmation permettant d'utiliser la puissance du formalisme du calcul situationnel est proposée par J. Funge [FT99]. Cet environnement, baptisé CML (Cognitive Modelling Language), a pour particularité de proposer un langage haut-niveau à

l'utilisateur qui n'a donc pas à formuler d'axiomes dans la logique du premier ordre utilisée par le formalisme du calcul situationnel. Par ailleurs, on note l'introduction de la notion d'Interval Valued Fluent [Fun99] qui permet de représenter le degré d'incertitude des propriétés.

Le calcul situationnel dispose d'un très grand pouvoir d'expression grâce à son formalisme qui permet donc la description de mondes complexes. Cependant c'est cela même qui lui cause du tord car la quantité de mondes possibles résultants d'une action à partir d'une situation donnée est telle que les algorithmes de planification deviennent rapidement très couteux en temps et en mémoire.

5.1.1. STRIPS

Pour pallier à ce problème, le formalisme STRIPS (STandford Research Institute Planning System) [FN71] considère un sous-ensemble du calcul situationnel. On retrouve les concepts de faits, d'actions et de situations. Malgré la perte d'expressivité, STRIPS a l'avantage de permettre l'utilisation d'algorithmes de planification applicables à des mondes complexes. On note en particulier deux algorithmes, GRAPHPLAN et HSP (Heuristic Search Planning). GRAPHPLAN [BF97] construit un graphe en couche qui alterne nœuds de type action et nœuds de type situation en partant de la situation initiale et en explorant l'ensemble des actions réalisables.

D'autre part, HSP [Bon01] utilise une heuristique pour s'orienter dans la recherche de situations. Cette approche fait qu' HSP considère un nombre moins important de situations que GRAPHPLAN et lui permet donc de considérer des problèmes plus conséquents.

5.1.2. Les réseaux de tâches hiérarchiques :

Le modèle HTN (Hierarchical Tasks Networks) introduit dans le formalisme de STRIPS une hiérarchisation conceptuelle. Trois concepts sont définis :

- les taches qui correspondent aux buts,
- les méthodes qui décrivent une suite de sous-tâches et/ou d'actions permettant de réaliser une tache et qui introduisent la hiérarchie,
- et les actions, éléments unitaires qui peuvent être réalisées sans décomposition.

Un apport intéressant du modèle HTN est le fait que pour une tâche donnée, plusieurs méthodes permettant de la résoudre peuvent être décrites. Ainsi, l'algorithme de planification cherchant à réaliser une tâche va construire un arbre ayant la tâche pour racine et la ou les méthodes associées à sa résolution pour nœuds fils (et ainsi de suite, les feuilles de l'arbre étant les actions). Les méthodes et les actions dont les pré-conditions ne sont pas vérifiées ne seront pas ajoutée et l'algorithme tient compte lors de la création des fils d'un nœud du fait que chaque fils doive être réalise (conjonction d'actions) ou que la réalisation d'un seul fils suffit (dans le cas ou plusieurs méthodes sont possibles). L'intérêt de l'apport hiérarchique des méthodes réside aussi dans le fait que cela permet d'introduire des connaissances sur la manière de résoudre (partiellement) certaines tâches tout en laissant à l'algorithme de planification la liberté de choisir parmi plusieurs chemins de résolution possibles. Cette possibilité de décrire avec précision la manière de résoudre une tâche tout en laissant des degrés de liberté explique que cette méthode soit utilisée en particulier dans la fiction interactive [CM02], domaine où il est nécessaire de maintenir une trame narrative tout en laissant à l'utilisateur la plus grande liberté d'action possible.

5.1.3. Synthèse :

Les systèmes bases sur le calcul situationnel possèdent une grande puissance d'expression grâce au formalisme utilisé qui permet la description d'environnements complexes. Mais cette puissance est source de problèmes, tel que le problème de la fenêtre qui résulte du fait que chaque action dans le formalisme du calcul situationnel est sensée décrire ce qu'elle modifie dans la situation mais aussi ce qu'elle laisse inchangé or cela devient vite impossible pour des systèmes trop complexes. Ce problème a été résolu par l'introduction de l'hypothèse du monde clos qui suppose que tout ce qui n'est pas spécifie comme étant modifié par le résultat d'une action reste inchangé. Par ailleurs, la complexité des mondes possibles fait que les algorithmes de planification deviennent rapidement trop exigeants en termes de mémoire ou de temps. C'est pourquoi des approches telles que les systèmes STRIPS ou HTN qui opèrent sur des sous-ensembles du formalisme du calcul situationnel ont été considérées, permettant ainsi de réduire la complexité de la planification de but. Enfin, un problème inhérent à ces méthodes de planification d'une suite d'actions à effectuer pour atteindre un but donné est que cela suppose l'absence de modifications de L'environnement par une entité autre que celle qui a conçu le plan. Si la situation devait être perturbée lors de l'exécution d'un plan, il faudrait alors recalculer un

nouveau plan permettant d'atteindre le but recherché. Ces systèmes ont donc une très faible réactivité face à l'évolution du monde et peuvent difficilement faire preuve d'opportunisme. Le langage ABL (A Behavior Language) et [MS04] présente un système proche de HTN introduisant une plus grande réactivité. Cependant la réactivité est difficile à mettre en place dans des systèmes bases sur le calcul situationnel, ce qui nous amène à considérer les systèmes à base de sélection d'actions.

5.2. Sélection d'actions :

La motivation à la base de la création des systèmes à base de sélection d'actions est de créer des comportements réactifs aux changements pouvant survenir dans l'environnement en résultat d'actions autres que celles réalisées par l'agent animé. Les mondes ou plusieurs agents évoluent sont typiquement des environnements ou la situation est susceptible d'être. modifiée par un agent extérieur. Parmi les premiers travaux dans ce domaine, le système ASM (Action Selection Mechanism) de P. Maes présente un algorithme qui a servi de base à la plupart des travaux qui ont suivis. Contrairement aux approches présentées précédemment ou l'on considérait la planification d'une suite d'actions permettant d'atteindre un but, il n'est désormais plus question d'un plan complet mais juste d'effectuer à chaque instant une action isolée qui doit permettre au comportement de converger de façon incrémentale vers le but souhaité. Apres que chaque action choisie ait été réalisée, le système réalise un nouveau choix qui prend en compte les changements qui sont potentiellement survenus dans l'environnement.

5.2.1. BCOOL

Le système BCOOL (Behavioral and Cognitive Object Oriented Language) [Lam03] est un système évolue décrivant un comportement reprenant les bases de l'algorithme de sélection d'actions. Dans BCOOL, le monde contient des faits et des actions. A chaque fait sont associées deux valeurs, l'activation et l'inhibition, qui prennent leurs valeurs dans l'intervalle [0...1] et qui décrivent la volonté de l'agent virtuel de satisfaire ou non ce fait. Si l'activation l'emporte sur l'inhibition, alors l'exécution des actions qui provoquent l'ajout de ce fait est recherchée. L'algorithme de sélection d'actions va alors construire un graphe de planification constitué d'une alternance de nœuds actions ou faits, structuré en couche, et qui permettra d'estimer la distance entre les buts à atteindre et les faits

considères comme vrais. Ainsi l'algorithme de sélection d'actions pourra choisir une action qui tend à rapprocher l'état du monde vers le but recherché.

5.2.2. Synthèses

Les systèmes à base de sélection d'actions ont l'avantage d'être extrêmement réactifs. En effet, âpres l'exécution de chaque action, l'agent considère quelle sera sa prochaine action en fonction de l'environnement tel qu'il est au moment du calcul. Ces modèles sont particulièrement adaptés aux mondes dynamiques et permettent de créer des comportements opportunistes. Cependant, un inconvénient majeur de ces systèmes est que lorsque plusieurs buts sont recherchés, il arrive que l'agent oscille, effectuant tour à tour une action le rapprochant d'un but et l'´éloignant d'un autre. Par ailleurs, ces systèmes ne sont pas complets et donc rien n'assure lorsqu'un plan d'actions existe pour atteindre un but qu'ils soient capables de le trouve.

5.3. Systèmes BDI

Dans l'optique de systèmes fortement réactifs aux changements de l'environnement, les systèmes de type BDI (Beliefs Desire Intentions) proposent une approche fortement inspirée du raisonnement de l'être humain. Ces systèmes reposent sur trois concepts :

- les croyances, qui représentent la connaissance que l'agent a du monde,
- les désirs, qui représentent les motivations de l'agent,
- et les intentions qui sont des plans que l'agent va utiliser pour satisfaire ses désirs.

Les systèmes BDI disposent d'un ensemble de plans fournis par le concepteur dans le but de décrire différentes manières de satisfaire certains désirs des agents. Lorsque l'agent souhaite réaliser un désir, il recherche parmi les plans dont il dispose ceux qui sont réalisables (dont les pré-conditions sont vérifiées) qui deviennent alors des intentions et qu'il va suivre en effectuant l'une après l'autre les actions qu'ils décrivent. Lorsqu'un changement survient, suite à l'´échec d'un plan en cours de réalisation ou suite à un changement dans l'environnement, la sélection de plans recommence. Le système est rapide car il ne raisonne pas sur les actions vu qu'à chaque étape la première action réalisable est effectuée. La grande réactivité des systèmes BDI a permis de les utiliser dans un contexte tel que la détermination des actions à suivre pour des avions au niveau d'un aéroport [RG95].

Malgré la rapidité et la grande réactivité de ces systèmes, un inconvénient majeur vient du manque d'inventivité de l'agent qui ne peut suivre que des plans préétablis et lorsqu'aucun plan ne permet de satisfaire un désir alors ce désir est tout simplement omis.
D'autre part, la compatibilité des désirs peut poser problème et ce en partie à cause du fait que les désirs ne sont pas exprimés dans un cadre formel très précis contrairement par exemple aux systèmes STRIPS.

6.Objets informés

Les modèles à base d'objets informés reposent sur une approche très différente de celles présentées jusqu'à présent. En effet, dans les modèles précédents, l'agent acquiert des informations sur les objets présents dans son environnement par des actions de perception qui lui apportent une connaissance sur les caractéristiques de l'objet mais pas sur les maniérés dont il peut interagir avec celui-ci. Les approches à base d'objets informés utilisent un modèle ou tout objet présent dans l'environnement contient des informations sur la manière dont les entités de l'environnement peuvent interagir avec lui. Lorsqu'un agent effectue une action de perception a l'encontre d'un objet, ces informations lui sont directement transmises. Il est donc possible d'inclure des informations déjà interprétées dans l'environnement, ce qui simplifie le processus de perception.

6.1. Les Smart Objects

Le modèle de [KBT02] présente des objets informes appelés ***smart objects***. Leur représentation est basée sur certaines caractéristiques d'interaction qui contiennent des informations sur la structure de l'objet, les endroits où il peut être saisi, sa fonctionnalité etc. L'humanoïde virtuel peut par une action de perception demander à tout objet une liste des interactions disponibles. Cette liste est générée à l'exécution et dépend de l'état dans lequel se trouve l'objet au moment de la requête. La liste des différentes formes d'interaction avec les objets constitue le langage de communication entre les acteurs et les objets, et le système permet l'utilisation d'environnements ou plusieurs entités interagissent en même temps sur les objets présents.

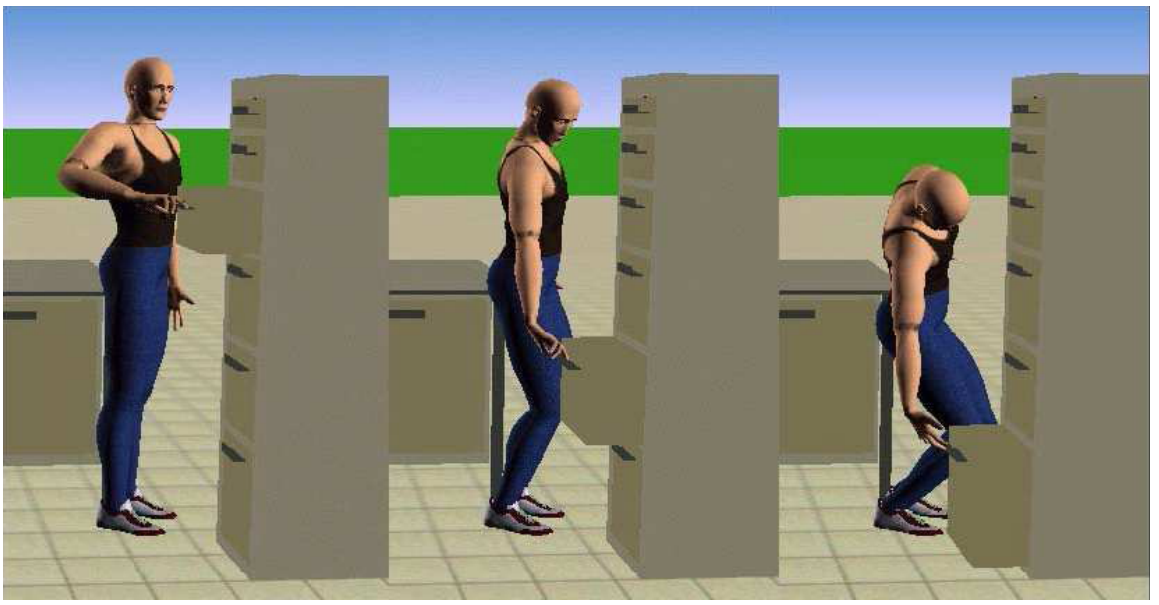


Figure.15: Exemple de Smart Object : interaction avec un casier [KBT02].

6.2. Le modèle STARFISH

Une autre approche utilisant des objets informés appelés ***synoptic objects*** est décrite par [BD04]. Cette approche repose sur un ensemble minimal d'actions primitives de bases inspirées des travaux de Schank [SA72] qui une fois combinées permettent de construire des actions complexes. Chaque action est ensuite associée à des surfaces interactives qui sont en fait les parties de la géométrie de l'objet considère concernées par l'action. L'agent souhaitant interagir avec un objet va obtenir les informations spécifiques à la manière dont il peut interagir avec l'objet. Il devra ensuite adapter son comportement en fonction des informations reçues. Ceci constitue une différence importante entre ce modèle et les smart objects qui eux vont en fait prendre le contrôle de l'humain virtuel afin d'effectuer l'action qu'il souhaite réaliser au niveau de l'objet.

7.Emotions et personnalité

Nous allons ici nous intéresser au modèle émotionnel de référence puis à une architecture de cartes émotionnelles floues s'inspirant de ce modèle.

7.1. Modèle OCC :

Le modèle OCC [OCC88,Bar02] est le modèle référence pour la synthèse d'émotions. Il spécifié 22 catégories d'émotions, basées sur des réactions à des situations. Ces situations

sont construites à partir soit d'événements concertants les buts, soit des actes d'un agent (qui peut être lui-même), soit de l'attraction ou de la répulsion d'un objet. Il dispose également de variables comme la probabilité d'un événement ou la familiarité d'un objet. Son niveau de complexité et de détails est suffisant pour couvrir la plupart des situations auxquelles un humanoïde peut être confronté.

Son processus se déroule en cinq phases :

1. **La classification**. Il s'agit de l'évaluation de l'événement, de l'action ou de l'objet, et la détermination des catégories émotionnelles affectées.
2. **La quantification**. Les intensités d'affectation des catégories sont calculées.
3. **L'interaction**. La valeur émotionnelle, définie par la classification et la quantification, interagit avec l'état émotionnel courant.
4. **L'association**. Les catégories émotionnelles doivent être associées à des expressions.
5. **L'expression**. L'état émotionnel est exprimé sur les modalités et peut influencer le comportement.

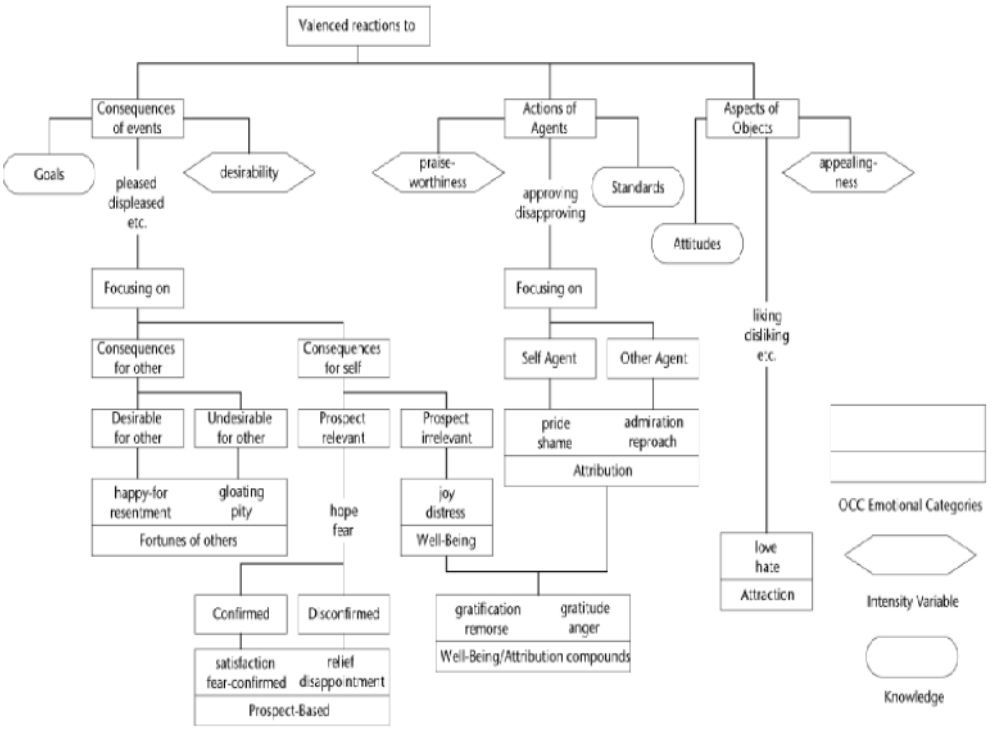


Figure. 16 Modèle OCC original [Bar02]

La phase de classification nécessite l'usage de connaissances pour déterminer les relations entre, par exemple, un objet et un but. Ces connaissances sont de trois types : les buts, les valeurs communes (standards) et les préférences (attitudes). Elles doivent contenir les informations relatives à tous les événements, actions et objets possibles, qui concernent les catégories affectées et la manière dont les intensités peuvent être calculées. La quantification de l'intensité est définie différemment selon qu'il s'agisse d'un événement, d'une action ou d'un objet : on parle respectivement de désirabilité, respectabilité (praise worthiness) et d'attirance (appealingness). De nombreuses variables peuvent intervenir dans ces calculs d'intensité, comme par exemple la probabilité d'un événement et la hiérarchie du but qui lui est associé.

La personnalité d'un personnage est décrite dans ce modèle comme sa cohérence. Si par exemple, une action le rend heureux, cela devra continuer dans le futur. Bartneck [Bar02] a apporté quelques conseils intéressants sur l'utilisation de ce modèle. Tout d'abord, le volume des connaissances peut être important. Pour réduire ce volume, on peut partager ces données avec les architectures cognitives car elles sont parfois redondantes. De plus, dans de nombreux cas, le modèle peut être simplifié en supprimant des catégories émotionnelles.

En effet, sa complexité n'est pas forcément nécessaire si l'on cherche à avoir des émotions crédibles plutôt que précises. Plusieurs modèles simplifiés existent, comme celui utilisé dans l'architecture suivante.

7.2. Cartes émotionnelles floues

Nédélec [NFSR05] a proposé une architecture qui adapte les cartes cognitives floues [PTRD01] pour représenter les émotions d'un agent. On parle alors de cartes émotionnelles floues. Les émotions peuvent être associées en paires positive/négative et sont divisées en trois groupes, selon qu'elles sont liées à l'environnement (espoir/crainte, satisfaction/déception), à la conséquence des ses propres actions (joie/tristesse, fierté/honte) ou à la conséquence des actions des autres agents (sympathie/antipathie, approbation/opposition). La structure générique de carte émotionnelle floue est présentée dans la figure 17

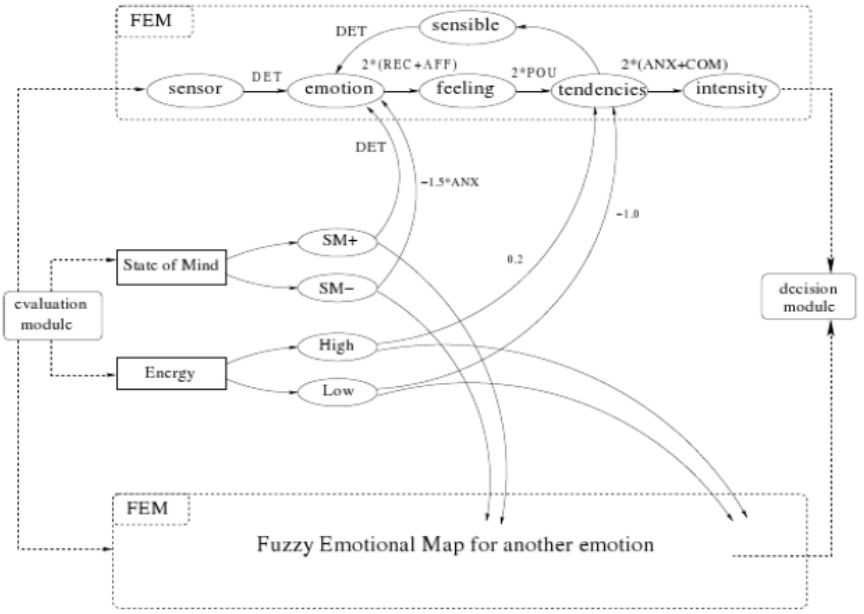


Figure. 17 Modèle FEM [NFSR05]

La personnalité est ici matérialisée par les différentes valeurs d'influence entre les concepts de la carte émotionnelle floue. Les agents sont capables d'interagir socialement, en s'échangeant certaines informations, modélisées par une matrice d'interaction sociale. Deux éléments sont pris en compte : la distance d'interaction (proxémie) et la compréhension de l'autre. La matrice est à la fois définie par des attributs statiques comme la personnalité, buts et règles de proximité, et aussi par des attributs dynamiques tels que les affinités avec les autres agents, l'état d'esprit, la fatigue et l'habitude.

Cette architecture a été utilisée dans une simulation où les agents devaient collaborer pour déblayer une zone sinistrée. Les agents avaient des personnalités différentes et au fur et mesure du déroulement de la simulation, des affinités se sont créés entre eux, selon le fait, par exemple, qu'un agent ait aidé ou non un autre agent. Leurs comportements étaient fortement influencés par ces émotions. Cependant, l'usage de douze cartes émotionnelles floues par agent s'est avéré coûteux en temps de calcul.

L'intérêt principal de cette architecture est l'élaboration des cartes émotionnelles floues ; les modules d'évaluation et de décision ont été réalisés de manière ad-hoc pour les besoins de la simulation. L'intégration de ces cartes émotionnelles floues dans une architecture cognitive apporterait plus de crédibilité à un acteur virtuel.

8. Conclusion

De l'observation de cette multitude de modelés existants et utilises pour répondre à une même problématique qui est la description d'un comportement pour humanoïdes de synthèse.

Chapitre III : La navigation réactive

Chapitre III

La navigation réactive

1.Introduction

La navigation réactive a pour but de considérer les entêtées simulées dans leur globalité mais de façon indépendante. Certains algorithmes de navigation vont plus se baser sur l'aspect réglé du mouvement, autrement dit-il utilisent un système à base de règles. D'autres utilisent des modèles plus orientés vers la notion de force comme le modèle HiDAC [PAB07], modèles tendant à représenter les humanoïdes plus sous forme d'animation de particules que de mouvement humain. Enfin, d'autres algorithmes sont basés sur la géométrie prenant en compte la direction et la vitesse des entités.

2.Modèle à base de particules

Introduit par I. Peschl [Pes71] en 1971, ce modèle fait l'analogie entre le déplacement d'individus en forte densité, n'ayant quasiment pas de liberté de déplacement, et l'écoulement de particules dans des compartiments. Ce modèle est donc un raffinement du modèle macroscopique fluidique, discrétisant le fluide par autant d'entités représentant les piétons. I. Peschl indique dans son étude que cette modélisation est pertinente pour le cas où une forte densité de personnes est confinée et en état de panique. Quand le flot se dirige vers une unique issue, on voit apparaître un phénomène d'agglutination. Au final, le débit de l'issue devient quasi nul. Le parallèle peut être fait avec des billes introduites dans un récipient percé, qui tombent grâce à la force de la gravité mais sont bloquées à cause de la poussée croissante qu'elles exercent.

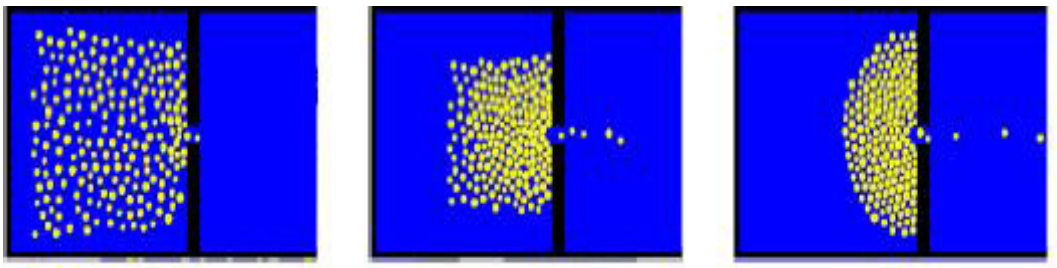


Figure.18 Phénomène d'agglutination dans le modèle de particules de D.Helbing [HFV00]

Le modèle de particules permet d'appréhender le phénomène d'arche observé par K. Togawa [Tog55] en 1955 (Figure 18). I. Peschl, en faisant des expériences à la fois sur des personnes et sur des billes de métal, a déduit les règles suivantes :

- Quand la largeur d'une issue augmente, la probabilité d'apparition d'une arche diminue tandis que la variation du débit augmente.
- La probabilité d'apparition d'une arche croît avec la densité.
- Le flux est linéairement proportionnel à la largeur de l'issue.
- Les pulsations du flux à travers une issue sont induites, le plus souvent, par la formation et la résorption des arches.

D. Helbing [HBJT00] étend ce principe avec un système de forces socio-psychologiques traduisant deux caractéristiques : premièrement la tendance à garder une certaine distance avec les autres piétons, deuxièmement la friction intervenant lors de la résolution d'un contact entre plusieurs piétons. Le cadre d'application principal de ce modèle est la gestion du comportement de panique, qui ne nécessite qu'une procédure comportementale simple (aller vers un endroit, généralement une sortie). La validité de cette approche est attestée par la reproduction de mouvements macroscopiques observables en fortes densités [HBJT05]. Des modèles ont ensuite étendu celui de D. Helbing à d'autres cadres d'application. Ainsi, A. Braun et al. [BMOB03] introduisent un lien social entre les entités au niveau de leur connaissance mutuelle, permettant une entraide en cas de panique. De la même manière, N. Pelechano et N. I. Badler [PB06] exploitent le modèle de D. Helbing pour la simulation d'évacuation d'immeubles, en prenant en compte trois catégories de population : les entraînés, les meneurs, et les suiveurs. Ces trois archétypes

vont alors disposer de différentes forces d'attraction/répulsion représentant leur attitude globale.

Enfin, T.I. Lakoba et al. [LKF05] font état du non validité du modèle de D. Helbing dans des cas de faibles densités. Ils remarquent que les distances d'anticipation sont trop faibles, provoquant des accélérations irréalistes, et que les forces de répulsions, utilisées pour éviter l'interpénétration, sont trop fortes. Ils proposent ensuite une modification des équations et paramètres permettant de conserver un réalisme visuel dans un plus large spectre de densités de personnes.

Les nombreux modèles dans la littérature exploitent le modèle de particules, et particulièrement celui de D. Helbing, pour la gestion de la navigation pédestre. Ces modèles ont démontré leur adéquation aux situations d'évacuations. Deux raisons principales tendent à restreindre les modèles de particules à cette catégorie d'utilisation. Premièrement, ils ne proposent pas de gestion comportementale évoluée, ce qui est acceptable dans les cas d'évacuation où le but des entités consiste seulement à sortir de l'environnement, mais insuffisant pour des situations moins contraintes où les individus auront des objectifs variés. Deuxièmement, le principe de prédiction lors du déplacement n'est abordé que très succinctement dans ces modèles. Cela se justifie pour l'évacuation, qui impose une distance interpersonnelle très faible, mais devient plus gênant lorsque les densités de personnes diminuent, autorisant des adaptations de mouvement plus anticipées. Ainsi, il semble difficile d'exploiter de tels modèles pour des simulations d'environnements et de situations quelconques, présentant une certaine diversité de comportements et un gradient de densités suivant les zones.

2.1. Modélisation comportementale

La modélisation comportementale diffère quelque peu de la modélisation par système de particules dans la mesure où le comportement d'une entité est défini par rapport à un certain nombre de règles, qui, dans certains cas, s'inspirent des études sur le comportement humain.

Nous évoquerons plusieurs méthodes: les approches s'appuyant sur la représentation den l'environnement sous la forme de grille régulière, les approches plus générales considérant des déplacements continus et enfin, les approches utilisant des environnements informés pour permettre l'exploitation de résultats issus de l'analyse du comportement humain.

2.1.1. Le modèle à base de règles :

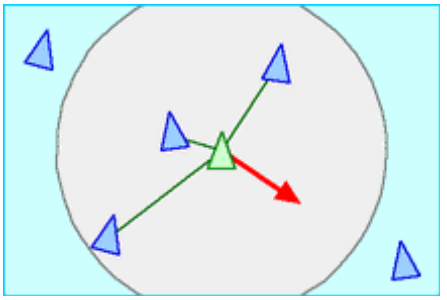
Craig Reynolds [[Rey87], avec son modèle Flocks of Boids, introduit la notion de simulation microscopique à base de règles. Le déplacement de chaque individu est régit par des règles de comportement de la forme « si condition alors action ». Trois règles sont proposées par C. Reynolds dans le cadre de l'animation de nuées (Figure 19) :

Séparation afin d'éviter d'éventuelles collision avec ses voisins.

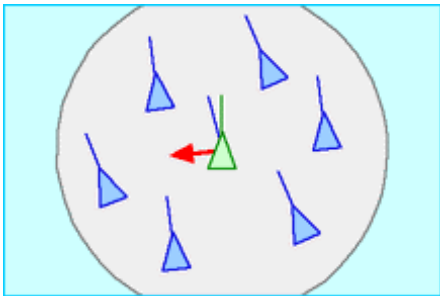
Alignement afin de réguler sa vitesse par rapport à l'ensemble du groupe.

Cohésion afin de rester proche de ses voisins.

(a) Séparation



(b) Alignement



(c) Cohésion

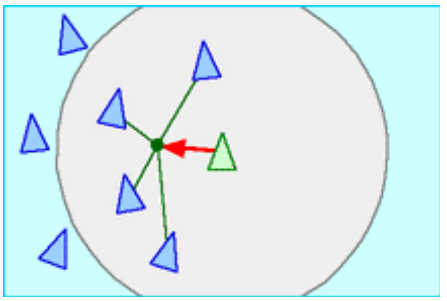


Figure.19: Les trois règles comportementales du modèles Flocks of Boids [Rey87]

De telles règles d'interrelations conduisent ainsi à un comportement de groupe émergeant, dans ce modèle flocks of boids, l'autonomie de l'entité est bridée par le fait que son comportement est uniquement défini en fonction de celui de ses voisins. On a alors deux catégories d'individus : des leaders qui choisissent réellement la trajectoire, et des suiveurs qui se contentent d'évoluer suivant le mouvement général en appliquant les règles d'évitement. C. Hartman et B. Benes [HB06] proposent de ne pas fixer le leader lors de la simulation, mais plutôt d'introduire une nouvelle règle permettant à une entité standard de prendre la place du décideur actuel. D'autres approches, comme celle de W. Shao et D. Terzopoulos [ST05] ou F. Lamarche et S. Donikian [LD04], exploitent tout de même le concept de règles sans avoir recours à un leader : les différentes possibilités d'adaptation de l'entité sont itérativement évaluées lors du déplacement, ce dernier étant cette fois-ci guidé par un processus de contrôle indépendant. Cette technique est aussi utilisée dans le monde du jeu vidéo [Gre00], où les possibilités de contrôle offertes par les règles séduisent. De plus, la faible complexité d'évaluation des modèles à base de règles est exploitée par le domaine de l'animation [LMM03] afin de peupler des environnements avec des milliers d'individus capables de se déplacer, et même d'accomplir certains comportements assez simples. Pour conclure, les modèles à base de règles proposent une approche plus souple de la navigation microscopique, permettant d'introduire des adaptations variées dans différentes situations. Néanmoins, le caractère itératif de la décision rend difficile la fusion d'informations, les règles étant indépendantes les unes des autres. Une solution, exploitée par W. Shao et D. Terzopoulos [ST05] ou encore S. Raupp Musse [Mus00], est de créer des règles gérant directement un ensemble d'informations, comme un groupe de personnes évoluant dans l'environnement. Mais cette approche a le désavantage de spécialiser le modèle, rendant encore une fois sa généralisation difficile : afin de gérer le gradient des situations pouvant apparaître lors de la navigation, un modèle à base de règles devra spécifier l'ensemble des adaptations pouvant intervenir pour l'ensemble des configurations.

2.1.2. Un modèle géométrique prédictif

F. Feurtey [Mus00] propose une approche prédictive de l'évitement de collision. Il représente l'environnement de navigation des entités en 3D, dans un repère (x, y, t).

Dans ce repère, l'ensemble des déplacements possibles pour une entité donnée se représente sous la forme d'un cône, dont la pente correspond à la vitesse maximale VM de l'entité, et dont le sommet est sa position actuelle M. Le cercle des déplacements admissibles en un laps de temps Δt est donc le cercle inclut par le cône dont le rayon est $V_M \Delta t$ (Figure 20(a)). Ensuite, une collision est représentée dans le cône sous la forme d'un segment, dans le cas où la trajectoire d'une entité voisine l'intersecté. Pour représenter l'imprécision de la prédiction de la trajectoire, en cas de changement de direction, la zone de collision potentielle est étendue par un triangle (Figure 20(c)).

F. Feurtey part du principe que pour éviter des collisions, un individu applique trois règles avec différentes priorités (Figure 20(b)) :

1. ***Préserver sa direction.***
2. ***Préserver sa vitesse.***
3. ***Préserver le temps nécessaire au déplacement.***

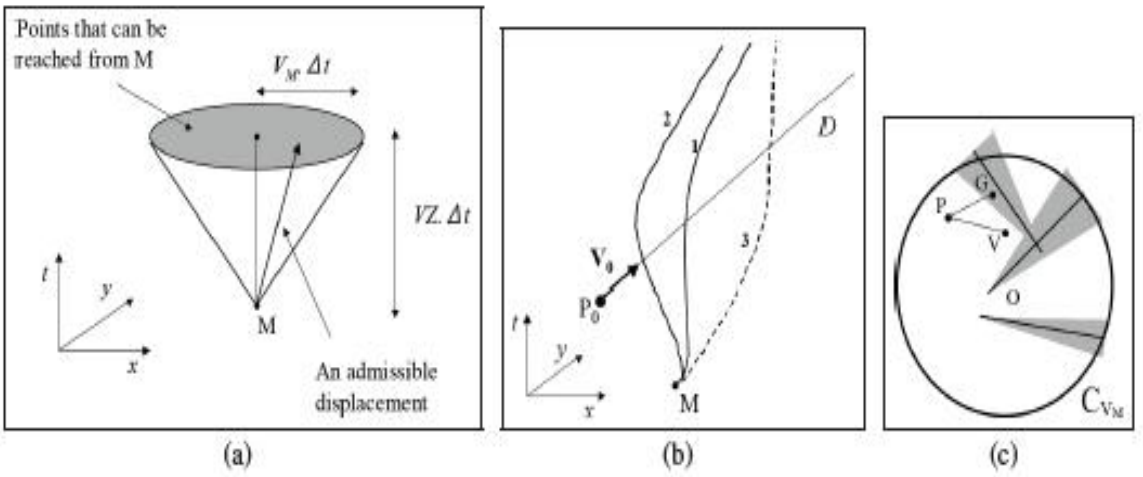


Figure 20 : Modèle prédictif de navigation de Feurtey [Mus00]

Cette méthode présente l'avantage indéniable d'être fondée sur les logiques de locomotion humaine, dont notamment les différentes capacités d'adaptation et le mécanisme de prédiction. On retrouve ainsi des références dans la littérature quant aux principes mais en avant par cette démarche, sans pour autant qu'elle soit réutilisée du fait de certains de ses inconvénients. Son premier point faible est qu'elle ne propose pas de méthode pour inclure les obstacles statiques de l'environnement, tels que les murs, qui jouent pourtant un rôle prépondérant dans la prise de décision de la navigation.

Un autre désavantage est que cette méthode considère les collisions séparément les unes des autres en appliquant sa fonction de coût, bien que celles-ci soient unifiées en une seule représentation. Ce second point bride le principe de prédiction, en négligeant des liens de dépendances entre plusieurs collisions (comme deux personnes marchant côte à côte).

Enfin, F. Furtey indique que le passage à l'échelle n'a pu être réalisé, sans doute à cause de la complexité de la méthode de recherche d'une solution. Ainsi, il ne fait état d'expériences n'impliquant qu'un nombre très limité d'entités, une douzaine, bien en deçà de l'effectif nécessaire à l'émergence d'une foule.

la représentation proposée par cette méthode dans le temps et l'espace soit un bon point de départ pour un processus de navigation réactive, tout comme le fait de considérer plusieurs possibilités de résolution de conflit. Il faut néanmoins étendre la méthode de calcul afin de fusionner les informations quant à la prise de décision, d'inclure les obstacles statiques de l'environnement, et d'améliorer la complexité du traitement pour permettre son application à des foules d'individus.

3.Classification des techniques de navigation

Nous venons de voir qu'il existe différentes tâches de navigation, donc différentes façons de se déplacer. La technique utilisée pour se déplacer doit dépendre du type de tâche à accomplir. Par exemple, une technique ne permettant que de grands déplacements risque d'être difficile à utiliser pour des tâches de manœuvre.

D'autres paramètres sont à prendre en compte dans le choix de la technique à utiliser. Par exemple, la distance à parcourir, la précision de déplacement nécessaire, le type de terrain (vallonné et plat) ou encore le nombre de degrés de liberté souhaité.

De nombreuses techniques de navigation ont été développées, principalement pour des configurations de RV et des applications 3D de bureau. Ces techniques permettent d'améliorer à la fois le déplacement et la partie cognitive de la navigation.

Devant le nombre grandissant des méthodes, de nombreuses classifications, appelées taxonomies, comparaisons et évaluations ont été effectuées [Han 97] [DB03] Chacune de ces taxonomies propose un point de vue différent sur les techniques de navigation. Au moins quatre classifications ont été présentées.

3.1. Techniques Actives ou Passives

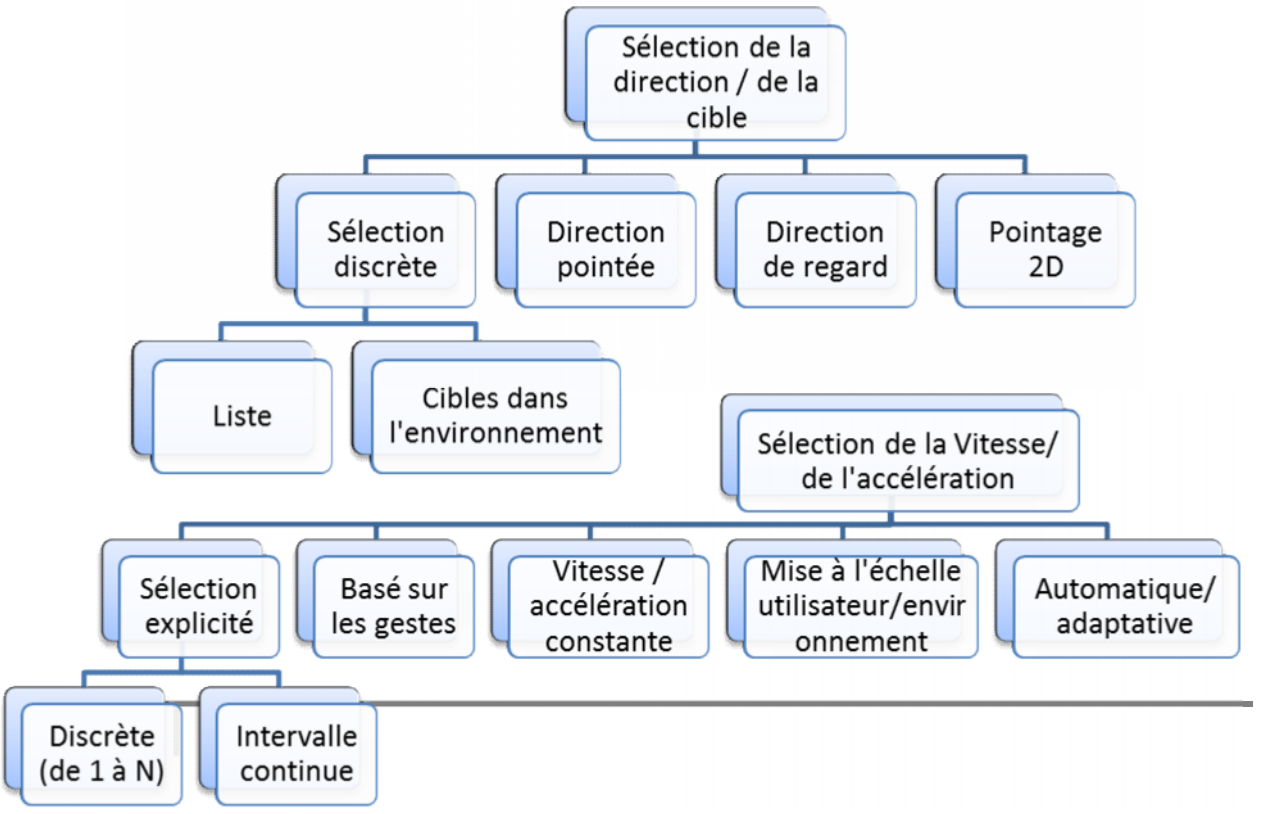
Une façon de classer les techniques est de différencier les techniques actives, où l'utilisateur contrôle directement les mouvements de la vue courante, des techniques passives, où la vue est contrôlée (totalement ou partiellement) par le système.

3.2. Techniques Physiques ou virtuelles

Il est aussi possible de différencier les techniques qui utilisent des déplacements physiques, où l'utilisateur se déplace réellement afin de modifier son point de vue, des techniques virtuelles, où le corps de l'utilisateur ne bouge pas, mais la vue 3D, si on parle aussi de métaphores du monde réel et de métaphores magiques. Beaucoup de systèmes de RV utilisent une combinaison de techniques virtuelles pour les déplacements et physiques pour les rotations (par exemple avec l'utilisation d'un casque).

3.3. En fonction de la tâche à accomplir

Dans ces classifications, le déplacement est décomposé en tâches élémentaires : sélection d'une direction (ou d'une cible), sélection de l'accélération et de la vitesse, et conditions de déplacement (voir figure 21). Une autre décomposition, plus chronologique, a également été proposée (voir figure 22).



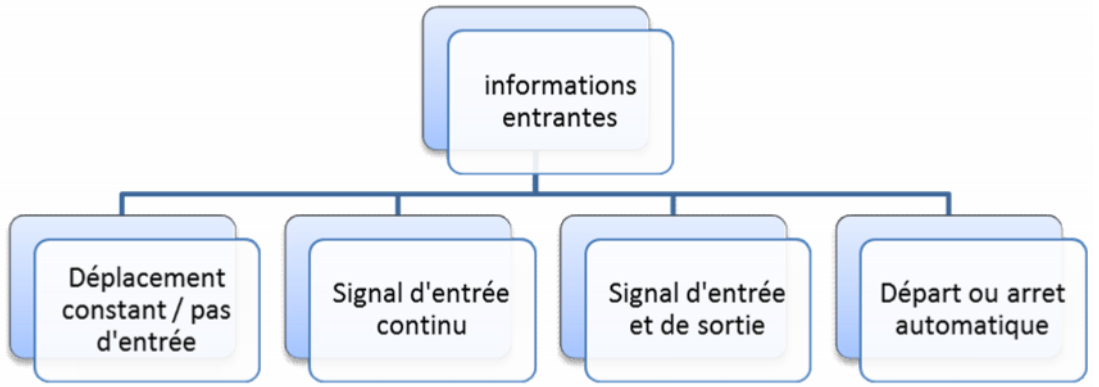


Figure21: Taxonomie des techniques de déplacement en fonction de la tâche à accomplir

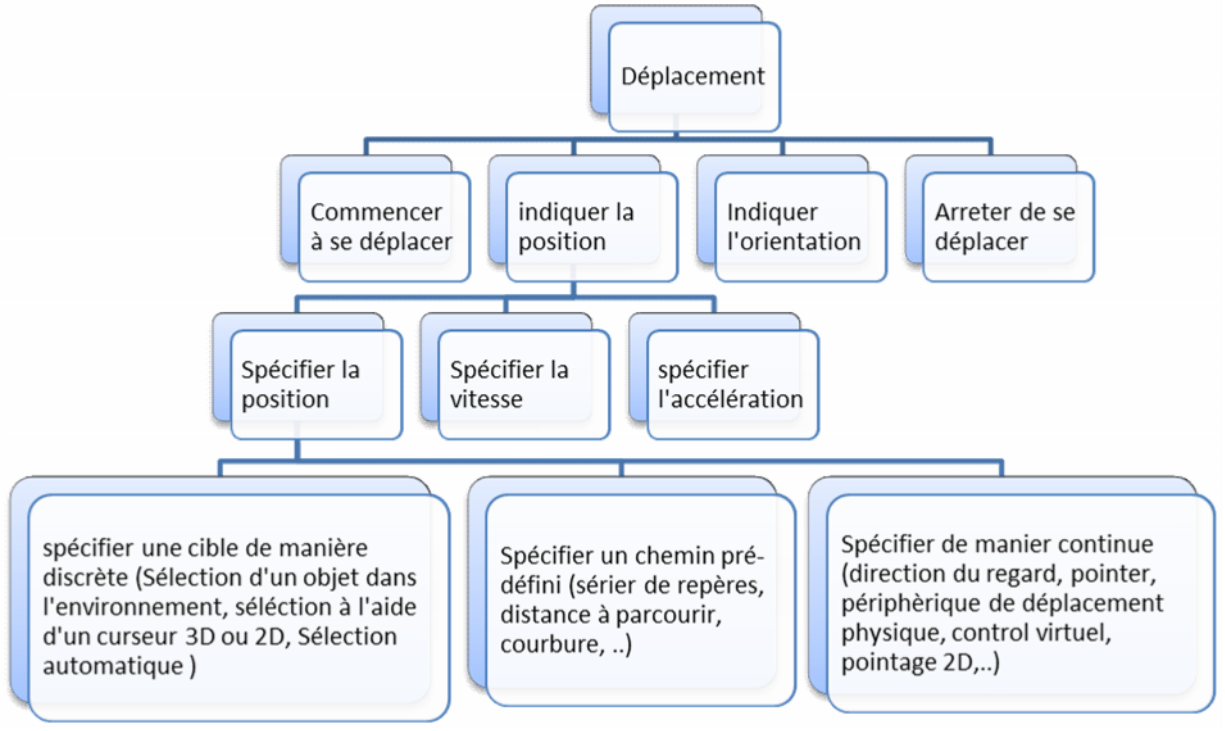


Figure 22: Taxonomie des techniques de déplacement axé sur le niveau de contrôle de l'utilisateur

3.4. Classification par Métaphore

Enfin, il est possible de classer les techniques en fonction de la métaphore qui lui correspond. Cette classification est la plus facile à comprendre et se rapproche de ce à quoi

pensent les utilisateurs. Par exemple, un utilisateur peut demander une technique de navigation qui serait « comme un tapis volant ». Dans ce cas, la classification par métaphore permet de déduire qu'il s'agit d'une technique permettant de se déplacer dans les trois dimensions.

4. Différentes modélisations : personne, groupe, foule

Différents niveaux d'échelle peuvent être utilisés pour modéliser des agents virtuels, une personne se déplaçant seule va être modélisé avec son propre comportement, tandis que plusieurs agents peuvent se grouper grâce, par exemple, à un intérêt commun. Ils forment ainsi un groupe possédant lui aussi un comportement, ce groupe peut posséder deux agents ou plus. Enfin ces groupes peuvent eux aussi faire partie d'une foule, celle-ci pouvant contenir plusieurs milliers d'humanoïdes.

4.1. Modèles d'agents virtuels

Trois chercheurs brésiliens ont proposé un algorithme [VMO04] permettant de modéliser un groupe d'agents virtuels d'où émerge un comportement plausible. Chaque agent est caractérisé à travers différents paramètres :

- **Sociabilité** : Il décrit, la motivation qu'a un agent à interagir avec d'autres agents. Il peut changer à chaque nouvelle interaction.
- **Communication** : Cela indique si l'agent est communicatif ou non. Cela peut être une valeur fixée en fonction de la personnalité de l'agent.
- **Confort** : Initialement mis à zéro, il évolue en fonction de comment se sent l'agent au niveau du confort de l'interaction.
- **Perception** : Chaque agent est capable de savoir si un agent est dans la zone de perception qui est une combinaison de la distance et de l'angle de perception.
- **Mémoire** : Cette mémoire est représentée par une collection contenant les agents avec lesquels il a interagi et la qualité de leur interaction.

Le groupe émergent est lui caractérisé par un paramètre de cohésion décrivant l'homogénéité des idées des membres du groupe. Ceci est un exemple d'implémentation possible proposée au sein de la communauté de recherche sur les entités autonomes.

Plusieurs des différents paramètres présentés ci-dessus peuvent varier, disparaitre et d'autres peuvent apparaitre en fonction du type de simulation souhaitée. D.Thalmann et S.R.Musse propose eux d'ajouter un statu émotionnel aux agents virtuels tels que tristesse, calme, joie ou encore anxiété [RMD00].

4.2. Modèles de groupes

Les groupes sont habituellement plus cohésifs s'il y a une certaine compatibilité parmi ces membres. Ce facteur de compatibilité peut varier en fonction de différents critères tels que le physique, l'idéologique, la morale ou l'intellectuel. Dans la grande majorité des groupes, les membres suivent une et même personne usuellement.
nommée "leader" [VMO04]. On peut distinguer plusieurs types de groupes : ceux ayant une forte cohésion, les membres n'abandonnent pas le groupe ; ceux ayant un plus faible taux de cohésion, ces membres restent solidaires mais peuvent s'éloigner pour aller former un autre groupe ; enfin ceux ayant la plus faible cohésion mais formant tout de même un groupe, ce dernier est temporellement très limité.
Certains chercheurs vont même simuler des groupes en utilisant un système à base de règles floues et contraindre le groupe à se mouvoir selon une forme prédéfinie [CL07] très utile dans les Battle Games.

4.3. Modèles de foules

Au sommet de la pyramide de modélisation d'entités se trouve celui du modèle de foule. La difficulté de cette simulation réside dans le fait que beaucoup de groupes exhibent un comportement d'une énorme complexité et d'une grande subtilité. Un modèle de foule ne doit pas seulement prendre en compte le mouvement et la navigation d'humanoïde ainsi que les contraintes environnementales mais aussi une collection déconcertante d'interactions dynamiques entre les agents. Plusieurs algorithmes ont été mis en œuvre répondant aux différentes règles régissant les comportements de foule.

F. Feurtey [Feu00] présente les variations de la vitesse d'un piéton et des ces libertés de mouvement par rapport à la densité de la foule dans laquelle il évolue. Car lorsque l'on se déplace au sein d'une foule, notre comportement ainsi que bon nombre de nos paramètres de navigation sont influencés par les mouvements et la densité de cette foule.
Ulicny et al. [UT02] utilisent une approche à plusieurs niveaux pour modéliser le comportement d'un individu au sein d'une foule en combinant les approches à base de

règles et d'automates d'états finis. Toutes ces approches sont confrontées au problème de la détermination des plus proches voisins.

Fabrice Lamarche [Lam03] a proposé une approche modulaire pour l'animation de foules en temps réel fondée sur la prise en compte de quelques règles de comportement et sur un modèle de locomotion issu de la biomécanique.

4.3.1. Classification des méthodes de foule

La classification suivante décrit les méthodes, qui peuvent être employées pour représenter les foules:

- Méthodes de foules basées images: définissent les méthodes de transformation de l'image et de vision à saisissent l'information de vraies foules.
- Modèles de foule basés physique: utilisés pour décrire le mouvement de foule sur la base de fondements de la physique.
- Modèles procéduraux: basés sur la description d'équations paramétriques.
- Modèles comportementaux: fournissent un outil pour simuler des foules sur des fondements basés comportement.

5. Évitement d'obstacles

Les méthodes d'évitement d'obstacles que nous présentons sont efficaces à condition d'avoir une perception correcte de l'environnement. Elles seront par exemple très efficaces avec un télémètre laser, mais donneront des résultats plus bruités avec des sonars. Pour limiter ce problème, il est possible d'appliquer ces méthodes sur une représentation locale de l'environnement qui sera construite en fonction des données de quelques instants précédents.

5.1. Méthode des champs de potentiel

Dans la méthode d'évitement d'obstacles par champs de potentiels, on assimile le mobile(robot, humanoïde) à une particule se déplaçant suivant les lignes de courant d'un potentiel créé en fonction de l'environnement perçu par le mobile. Ce potentiel traduit différents objectifs tels que l'évitement d'obstacles ou une direction de déplacement préférée. Il est calculé par sommation de différentes primitives de potentiels traduisant chacun de ces objectifs (Figure 21). Ces différents potentiels peuvent avoir une étendue

spatiale limitée ou non (par exemple, n'avoir une influence que près des obstacles) et leur intensité peut dépendre ou non de la distance.

Le gradient de ce potentiel donne, en chaque point de l'espace, la direction de déplacement du mobile (Figure 21). Comme c'est ce gradient, et non la valeur absolue du potentiel, qui nous intéresse, il est possible de calculer directement enchaque point sa valeur par une simple somme vectorielle en ajoutant les valeurs issues des différents potentiels primitifs.

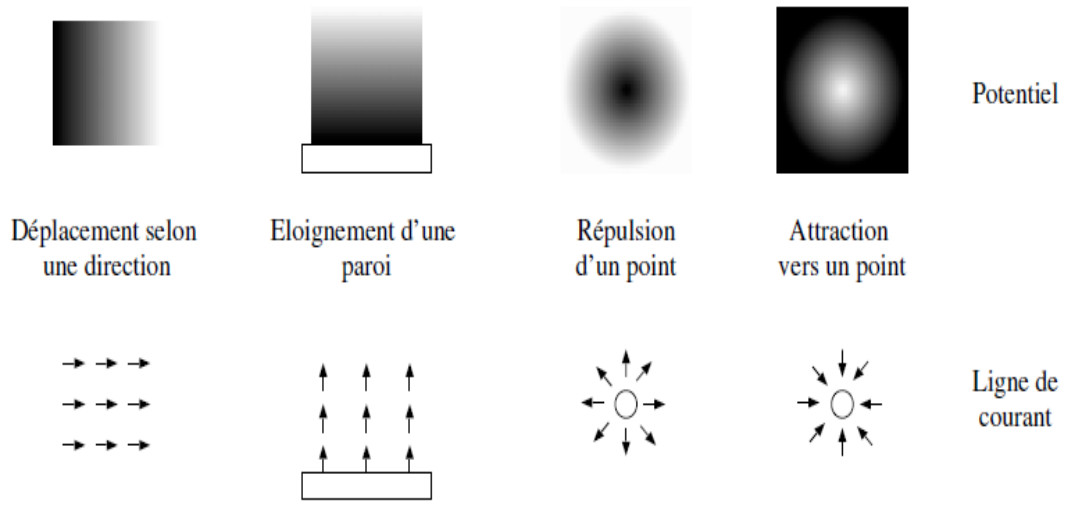


Figure.23 Illustration de potentiels primitifs dont la combinaison guide les Déplacements.

Le principal inconvénient de cette méthode d'évitement d'obstacles est l'existence, pour certaines configurations d'obstacles (relativement courantes) de minimum locaux du potentiel qui ne permettent pas de décider de la direction à prendre (Figure 23). Ce problème peut être traité de différentes façons. Il est par exemple possible de déclencher un comportement particulier lorsque l'on rencontre un tel minimum (déplacement aléatoire, suivi de murs ....). Il est aussi possible d'imposer que le potentiel calculé soit une fonction harmonique, ce qui garantit qu'il n'ait pas de minima, mais complexifie beaucoup son calcul.

5.2. Méthode Vector Field Histogram

Cette méthode a été conçue spécifiquement pour utiliser une grille d'occupation locale construite à partir de capteurs à ultrasons. Cette grille est construite de manière très rapide

par la méthode "Histogrammic in motion mapping" qui produit une grille dont chaque cellule contient un nombre d'autant plus élevé qu'elle a souvent été perçue comme contenant un obstacle (Figure 24).

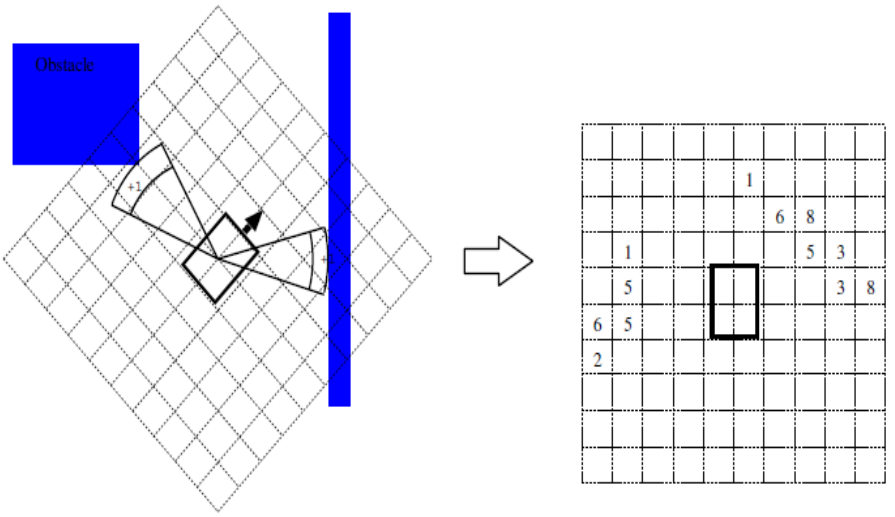


Figure.24 Grille d'occupation locale construite par la méthode "Histogrammic in motion mapping".

Un histogramme représentant l'occupation de l'environnement autour du mobile est ensuite construit à partir de cette grille d'occupation locale. Pour cela, l'environnement est discrétisé en secteurs angulaires pour lesquels la somme des valeurs des cellules est calculée (figure 25). Un seuil utilisé pour déterminer les directions possibles: toutes les directions dont la valeur est inférieure au seuil sont considérées. Le choix de la direction est finalement réalisé permis les directions possibles en fonction de contraintes externes (par exemple la direction la plus proche de la direction du but).

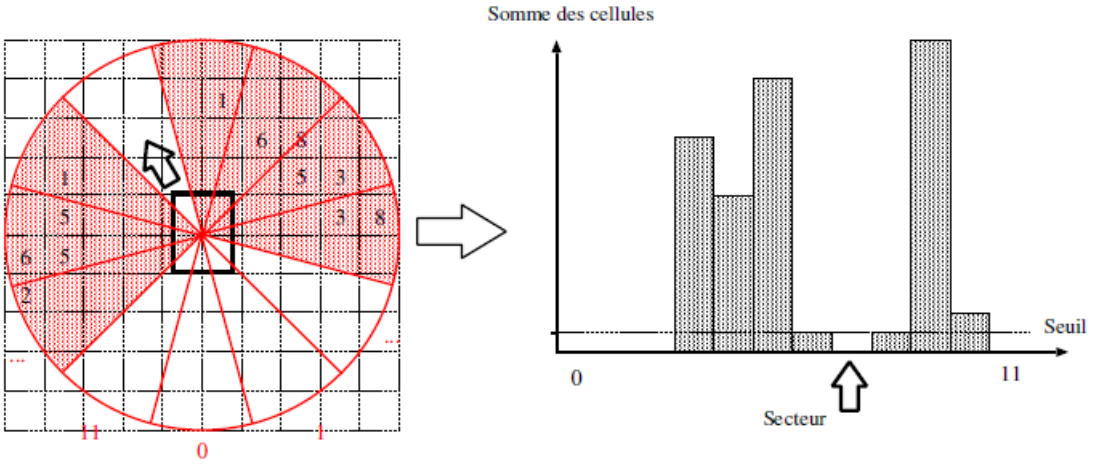


Figure.25 Utilisation de l'histogramme des obstacles pour déterminer la direction de déplacement

Cette méthode est extrêmement rapide et a permis historiquement un déplacement réactif à des vitesses assez élevées (environ 1 m/s). Diverses améliorations pour permettre le réglage de la vitesse en fonction de la densité des obstacles sont possibles.

5.3. Méthode de la fenêtre dynamique

La méthode de la fenêtre dynamique permet, à partir de la perception locale de l'environnement, de sélectionner un couple (v,ω) de vitesses de translation et de rotation du mobile qui répond à différentes contraintes, dont celle d'éviter les obstacles. Un tel couple de vitesses, lorsqu'il est appliqué au mobile, produit une trajectoire circulaire, pour laquelle la satisfaction des différentes contraintes peut être évaluée. A l'issu de l'évaluation de toutes les contraintes pour tous les couples de vitesses possibles, la méthode de la fenêtre dynamique permet de sélectionner le couple le plus pertinent (qui répond le mieux aux contraintes).

Figure.26 Contrainte d'évitement d'obstacles pour la méthode de la fenêtre Dynamique

La première contrainte est la contrainte d'évitement d'obstacles. C'est une contrainte dure au sens où elle est binaire (succès / échec) et doit obligatoirement être satisfaite. Elle est évaluée pour chacune des trajectoires possibles à partir de la perception locale de l'environnement à un instant donné et de la position estimée à un pas de temps fixé dans le futur pour la trajectoire courante. Si le mobile n'a pas rencontré d'obstacles à cet horizon, la contrainte est respectée ; dans le cas contraire, elle ne l'est pas (Figure 26).

Le respect ou le non-respect de cette contrainte est reporté dans un graphe des vitesses qui indique, pour chaque couple de vitesses possible (donc chaque trajectoire), si le mobile va ou ne va pas rencontrer un obstacle (Figure 27). Dans ce graphe, il est alors possible de tracer la fenêtre des vitesses accessibles au prochain pas de temps à partir des vitesses courantes du mobile et des valeurs d'accélération et décélération maximales. C'est cette fenêtre qui donne son nom à la méthode car elle permet de prendre en compte la dynamique du mobile (à travers la capacité de freinage et d'accélération). Il reste alors à choisir, au sein de cette fenêtre, un couple de vitesses qui ne conduise pas à percuter un obstacle pour garantir un déplacement sûr du mobile.

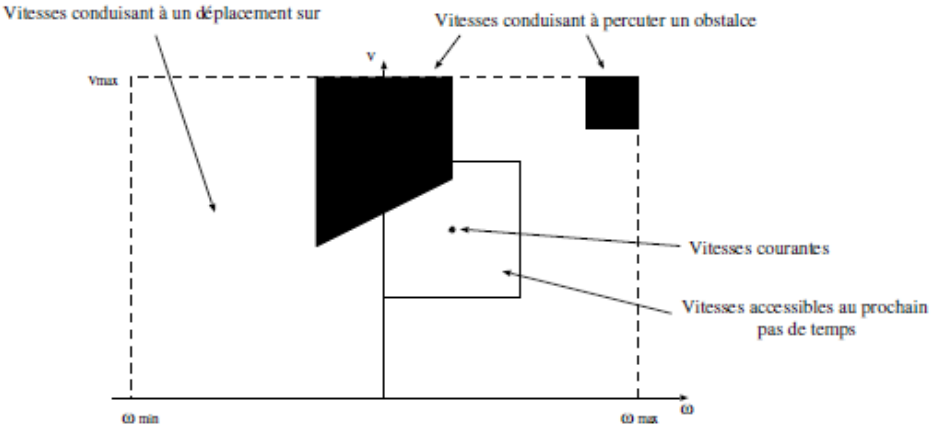


Figure. 27 Fenêtre de sélection des vitesses

Pour faire le choix parmi toutes les vitesses possibles au sein de cette fenêtre, il est possible d'utiliser des contraintes "souples" supplémentaires pour exprimer des préférences au sein de cet espace des vitesses accessibles. Ces contraintes s'expriment par une fonction de coût G(v, $\mathcal{E}$) qui est en général la somme de plusieurs termes. Ces termes peuvent exprimer une préférence a priori sur les vitesses, une préférence pour les trajectoires s'éloignant le plus des obstacles, ou une préférence de direction si l'on dispose par exemple d'une estimation de la direction d'un but à long terme (Figure 28). Le couple

de vitesses minimisant ce coût au sein de la fenêtre est alors sélectionné. Il garantit un déplacement sans rencontrer d'obstacles et le meilleur respect possible des contraintes souples dans ce cadre.

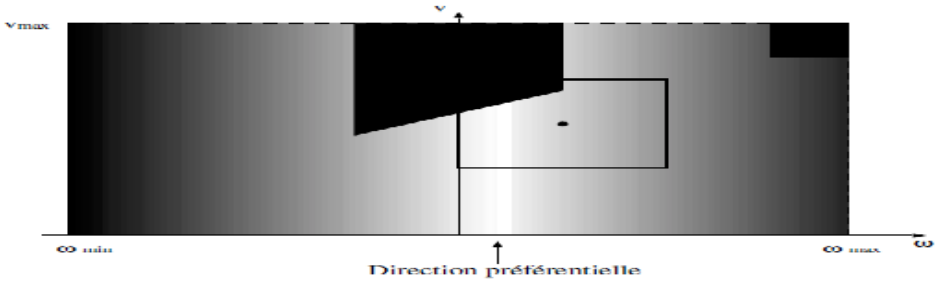


Figure. 28 Contrainte "souple" exprimant une préférence sur la direction à prendre

6.Détections de voisinage et évitement

6.1. Détection de la collision

La notion de détection de collision consiste à déterminer si, à une instante donne, plusieurs entités partagent une même partie de l'espace.

cette notion est nécessaire afin de permettre l'implantation d'application de simulation .une application de simulation doit permettre de reproduire la réalité de la manière la plus précise possible. Cela implique de respecter au mieux les réactions physiques des objets réels. Sans la notion de détection de collision, les objets déplace en traversant tous types d'obstacles.

Une collision entre plusieurs objets donne lieu à une réponse physique qui dépend des caractéristiques des objets et du domaine d'application .dans ce partie nous présentons l'ensemble des grande familles d'algorithme que l'on peut rencontrer. Le problème majeur concernant la détection de collision est le suivant :

Comment trouver les éléments en collision dans un temps minimum ?

6.2. Type Détections de la collision :

Nous présentons ici un bref état de l'art du domaine de la détection de collisions. Seules les méthodes les plus représentatives sont citées ci-dessous.

Une méthode répandue consiste à utiliser des boites englobant es pour réduire la complexité des tests d'intersections. Ces tests, effectués sur des polyèdres simples, comme

des AABB (Axis-aligned bounding box) [CLMP95], OBB (Oriented Bounding Box) [GLM96] ou k-dop (K Discrete Oriented Polytope) [KHM▫98], permettent d'éliminer des collisions éventuelles de la scène sans avoir à effectuer de tests précis. L'inconvénient de ces méthodes se situe au niveau de la mise à jour de la structure lorsque l'on considère des objets déformables.

Une approche entièrement différente consiste à rechercher les points les plus proches entre deux objets afin de tester s'il y a, ou non, collision. On citera notamment ici les algorithmes GJK et Lin-Canny [GJK88, LC91]. Ce type d'algorithme ne s'utilise que sur des objets convexes ou sur une décomposition convexe mais n'est pas particulièrement adapté aux environnements déformables complexes, car aucune modification topologique n'est supportée.

Une autre approche intéressante, utilisée en simulation, est l'utilisation de la multi résolution. Dans [OL03], un système appelé *CLODs* – Contact Level Of Details est décrit. Le principe est le suivant : si une collision a lieu le long d'une surface de contact large, alors cette surface peut être considérée à une résolution plus faible. Un autre concept lié aux CLOD qui a été introduit par Hubbard dans [Hub95] est le principe de temps critique. Le calcul est fait à la résolution la plus précise atteignable dans un temps limite fixé.

Une autre manière de diminuer le nombre de tests est l'utilisation d'une approche de type Monte-Carlo. Par le biais d'une approche stochastique [GD04] ou évolutionniste [Jou06], ces méthodes sélectionnent un échantillonnage sur les objets à tester et utilisent la cohérence temporelle pour faire tendre cet échantillonnage vers les points de proximité entre les objets.

Certains travaux, plus proches des applications que nous visons, s'attellent à traiter le cas d'objets naviguant dans de grands environnements, comme la simulation d'endoscopie ou d'angioscopie [Gei00, Aco04, LCDN06, WDS▫07]. Dans ces approches les mobiles sont souvent modélisés comme un ensemble discret de points ou de segments selon la précision requise pour le simulateur. Les approches de [Aco04, WDS▫07] utilisent une hiérarchisation spatiale ou une carte de distance pour détecter la collision. Avec ce type de structure, une requête de collision a une complexité de l'ordre de $O(logn)$ où n est la taille de la structure. De plus, ces modèles ne supportent pas facilement les modifications de topologie ou les déformations.

La méthode de [LCDN06] est basée sur un arbre de cellules, chacune des branches représentant une partie du vaisseau. Cette approche est particulièrement adaptée pour la

navigation dans des vaisseaux sanguins. Mais de même que précédemment, cette méthode ne prend pas en compte des éventuels changements de topologie. En outre, cette méthode nécessite un deuxième niveau de détails pour représenter la paroi des vaisseaux.

Les travaux de [Gei00] sont basés sur une décomposition spatiale de l'espace en tétraèdres, et sur l'utilisation de la cohérence temporelle. Cette décomposition permet l'utilisation de l'algorithme sur des structures non tubulaires. L'inconvénient de cette méthode est le manque de structure topologique. Ce manque se traduit par une restriction des modifications possibles sur l'environnement en temps interactif. De plus, tous les tétraèdres sont exprimés de manière explicite dans le modèle.

6.2.1. Intersection d'objets convexes

Parmi les optimisations proposées en détection de collision on trouve les algorithmes de recherche de proximité entre deux objets rigides convexes. Un objet est dit convexe si pour toute paire de points de cet objet, le segment qui les joint est entièrement contenu dans l'objet. Ces méthodes se basent sur cette propriété pour orienter la recherche de distance minimale entre deux objets. Cette distance, lorsqu'elle est nulle (ou négative pour une distance orientée), indique la présence d'une intersection entre ces deux objets.

Nous présentons cette famille d'algorithmes car elle est fréquemment utilisée. Son utilisation provient du fait que les intersections d'objets concaves (i.e. non convexes) peuvent être ramenées à des ensembles de tests d'intersections d'objets convexes qui sont moins coûteux à effectuer.

Parmi les méthodes d'intersections d'objets convexes, les plus répandues sont celles de :

- Gilbert, Johnson et Keerthi;
- Dobkin et Kirkpatrick ;
- Lin et Canny.

A partir de ces méthodes, de nombreuses améliorations et extensions ont été proposées. Nous présentons ici les méthodes de base et quelques extensions de la littérature.

a. *Gilbert, Johnson et Keerthi*

L'algorithme de Gilbert, Johnson et Keerthi [GJK88], plus communément connu sous le nom de GJK, vise à calculer la distance entre deux objets convexes séparés, ou à donner une approximation de la distance d'interpénétration pour deux objets en collision.

Cette estimation de la distance est basée sur la différence de Minkowski. La distance entre deux polygones convexes est exprimée directement selon cette différence. Le calcul de cette distance est ramené au calcul de la distance minimum à l'origine de la différence de Minkowski entre les objets à tester.

La différence de Minkowski n'est pas calculée intégralement mais de manière incrémentale pour les tests. La complexité de cette méthode est donc en m + *n, m* et *n* étant le nombre de sommets des objets à tester.

Une version améliorée de l'algorithme consiste à réutiliser en permanence les points les plus proches de la dernière itération [Cam97]. Lorsque les déplacements sont faibles, les nouveaux éléments les plus proches sont proches des anciens, les trouver nécessite donc moins de calculs.

Les travaux de Vlack *et al.* [VT01] proposent une extension de l'algorithme GJK à la détection de collision de manière continue. Ils consistent à considérer deux polyèdres convexes en déplacement et à identifier, en balayant l'espace parcouru par les polyèdres, si des collisions surviennent.

b.Dobkin-Kirkpatrick

L'algorithme de Dobkin-Kirkpatrick [DK90] consiste à précalculer une représentation hiérarchique des polyèdres de manière incrémentale en un temps *m* + *n, m* et *n* correspondant au nombre de sommets des polyèdres que l'on souhaite tester. Le plus bas niveau de la hiérarchie est le polyèdre original. Le niveau le plus élevé correspond à un tétraèdre. Pour construire un niveau de la hiérarchie, des sommets non-adjacents sont supprimés et associés à des faces.

A partir du niveau le plus élevé de la hiérarchie, les éléments les plus proches sont recherchés. Une fois ces éléments trouvés, seule la partie de la hiérarchie contenant ces éléments est raffinée : les polyèdres étant convexes, les éléments les plus proches au niveau inférieur sont soit ceux déjà trouvés, soit appartenant à la partie qui vient d'être raffinée.

La complexité de cet algorithme de recherche est, dans le pire des cas, en *logim)* x *log(n),* si les éléments les plus proches se trouvent au niveau d'une feuille de la hiérarchie.

c.Lin-Canny

L'algorithme de Lin-Canny [LC91] calcule la distance entre deux polyèdres convexes de manière incrémentale. A chaque élément des polyèdres (faces, arêtes et sommets) est associée la zone de l'espace contenant les primitives dont il est le plus proche. Ces zones définissent des *régions de Voronoi* comme montré par la figure 29.

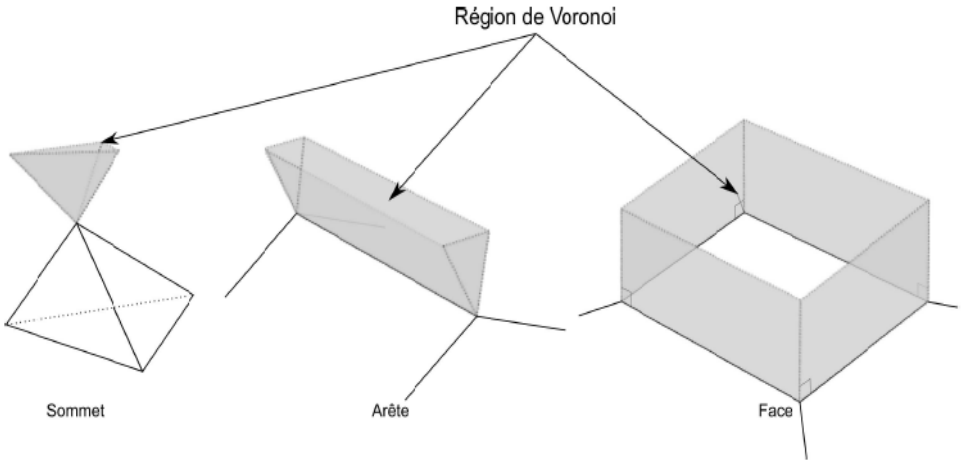


Figure.29 : Région de Voronoi d'un sommet, d'une arête et d'une face d'un polyèdre convexe

Le principe sur lequel se base cet algorithme est le suivant : pour deux objets P et Q, auxquels appartiennent respectivement deux éléments x et y, si x appartient à la zone de Voronoi de y et y appartient à la région de Voronoi de x alors x et y sont les éléments les plus proches entre P et Q. Les éléments x et y contiennent donc les deux points les plus proches entre P et Q. Ces deux éléments permettent de calculer la distance entre les deux objets et de définir s'il y a ou non une collision.

L'algorithme est initialisé avec un élément de chaque objet et progresse de manière itérative sur la surface des deux objets afin de trouver les deux éléments les plus proches en se basant sur les régions de Voronoi. La complexité d'une itération complète est linéaire en fonction du nombre de sommets des polyèdres dans le pire des cas. Le cas où les éléments sélectionnés à l'initialisation sont les éléments les plus proches, ou voisins des éléments les plus proches, donne un résultat en temps quasiment constant, car la progression itérative à effectuer sur les surfaces est nulle ou faible.

Un algorithme robuste se basant sur ce même principe a été implanté par Mirtich [Mir98] sous le nom V-Clip (pour Voronoi-Clip). V-Clip se base également sur une association de chaque élément du maillage à une région de Voronoi, il corrige certaines situations problématiques comme la génération de boucles infinies lors de la collision de deux objets.

6.2.2. Partitionnement de l'espace

Les méthodes de partitionnement de l'espace consistent à subdiviser l'espace autour des objets en sous-ensembles disjoints. Ce partitionnement permet de n'avoir à tester des intersections de primitives que lorsque ces dernières sont incluses dans le même sous-ensemble.

Le partitionnement de l'espace par des grilles de *voxels* [Lev66, Tur90] est une approche suivant ce principe. Deux difficultés sont à noter pour cette approche. La première est le fait de devoir mettre à jour constamment la notion d'appartenance d'une primitive à un ou plusieurs voxels. La deuxième concerne le choix de la taille des voxels. Les voxels doivent être suffisamment fins pour pouvoir être sélectifs et diminuer le nombre de primitives à tester. Ils doivent également être suffisamment grossiers pour qu'il n'y ait pas trop de duplications de tests d'intersections. Chaque fois qu'une primitive est incluse dans plusieurs cellules elle est associée à chacune d'elles. Si plusieurs primitives proches sont dupliquées dans de nombreux voxels, le nombre de tests d'intersections à effectuer augmente.

L'utilisation d'une table de hachage sur des grilles régulières de voxels permet d'accélérer les tests d'intersections [THM+03, EL07]. Nous détaillons son utilisation. Les objets sont représentés par des ensembles de tétraèdres. Chacun des sommets et des tétraèdres de la scène sont associé à la grille de voxels par le biais d'une table de hachage. Lorsqu'un sommet appartient à un tétraèdre de la scène, une collision a lieu. L'utilisation d'une table de hachage permet ici d'effectuer des tests d'appartenance uniquement pour les sommets et les tétraèdres partageant la même entrée dans la table de hachage. La limite de cette méthode provient du fait qu'elle se base uniquement sur des tests d'appartenance de sommets à des tétraèdres, les collisions d'arêtes ne sont pas détectées. La question du choix de la taille de la grille est fait ici selon la longueur moyenne des arêtes formant les tétraèdres.

Le problème principal du partitionnement spatial par des grilles régulières est l'incapacité à traiter de manière optimale les objets distribués de manière non uniforme [Mou04]. C'est pourquoi des grilles irrégulières ont été mises en place.

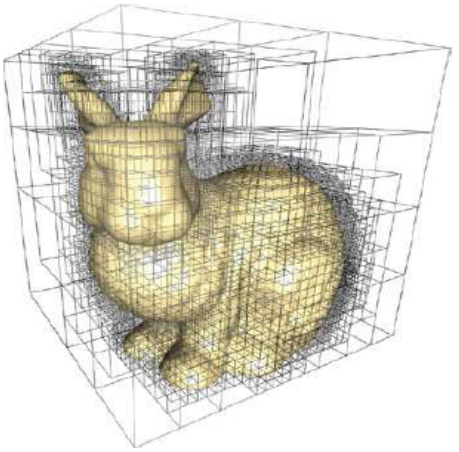


Figure. 30 exemple d'octree

6.2.3. Hiérarchie de volumes englobants

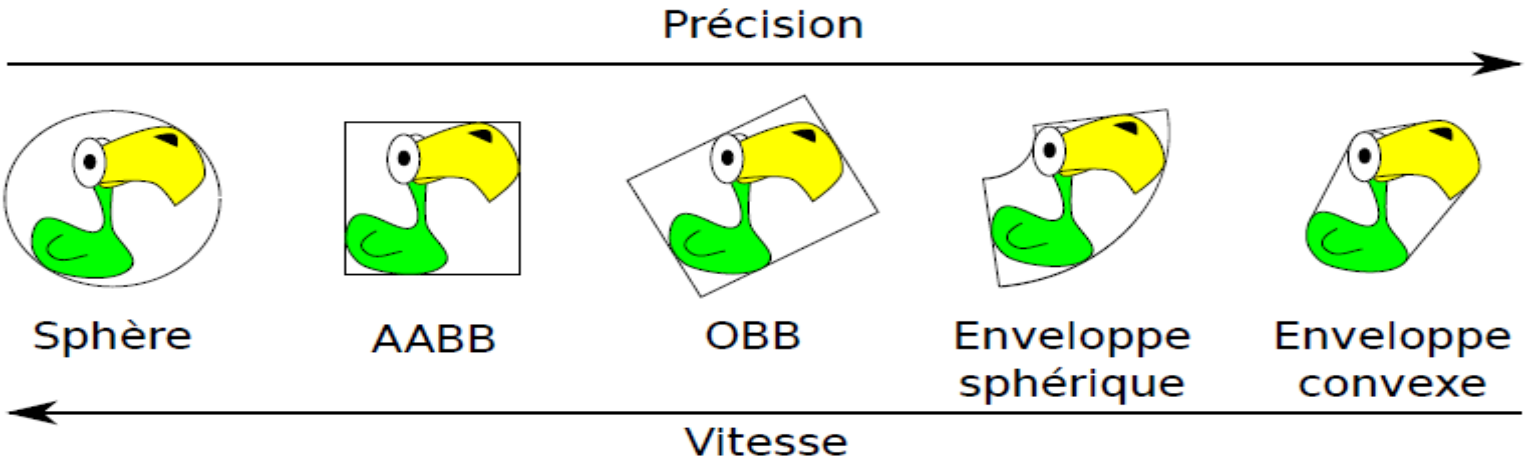
Afin de détecter s'il y a, ou non, une intersection entre deux volumes, l'utilisation de volumes englobant permet de limiter le nombre de tests géométriques lorsque les objets sont séparés. Un volume englobant est un volume fermé associé à un ou à une partie d'un objet de manière à le contenir entièrement. La limitation du nombre de tests géométriques provient du fait que les volumes englobants sont construits avec peu de primitives comparativement aux objets que l'on souhaite tester. Si l'on considère deux objets dans deux volumes englobants, s'il n'y a pas d'intersection de leurs volumes englobants alors les objets ne peuvent pas être en collision.

Une hiérarchie de volumes englobants, plus connue sous le nom de *Bounding Volume Hierarchy* (BVH) consiste à associer une arborescence de volumes englobants à un objet. Chaque nœud de l'arborescence est associé à un sous-ensemble de nœuds jusqu'à atteindre les feuilles. Les nœuds sont associés à des volumes englobants, plus ou moins complexe. Les feuilles de l'arborescence contiennent des primitives de l'objet géométrique englobé. Chaque nœud père englobe l'ensemble des volumes de ces nœuds fils de manière à ce que l'ensemble des primitives des feuilles de l'arbre soit inclus. L'idée de cette méthode est que

les tests sur les volumes des nœuds de l'arbre permettent d'élaguer rapidement des ensembles de tests exacts d'intersections de primitives, plus coûteux en temps de calcul.

Divers types de volumes englobants ont été proposés dans la littérature, chacun ayant des caractéristiques propres. Nous effectuons un classement de ceux-ci selon la simplicité de leur forme. Pour chacune de ces formes, diverses implantations ont été mises en œuvre : on trouve par exemple des hiérarchies de sphères [Hub93], de tétraèdres [JFSO06, JS08], de boîtes isothétiques (AABB : *Axis Align Bounding Box)* [CLMP95], de boîtes englobantes orientées (OBB : *Oriented Bounding Boxes)* [GLM96, RKC02a], de volumes balayés par des sphères (SSV : *Swept Sphère Volumes)* [LGLM99], de polytopes à orientation discrète formés de *k* plans : les fc-DOP *(Discrète Oriented Polytope)* [KHM+98], d'enveloppes sphériques [KPLM98] et d'enveloppes convexes. La figure 31 illustre l'application de certains de ces volumes englobants sur un même objet dans le plan.

Plus le volume englobant est simple, plus les tests d'intersections sont rapides. Ainsi pour une sphère seule la distance entre deux points est à mesurer. Cependant lorsque le volume englobant n'est pas adapté à l'objet qu'il approxime, la quantité d'espace vide



donnant une réponse positive à une potentielle intersection peut être importante.

Figure.31 : Ensemble des BVH ordonnés selon leur complexité. Plus un volume englobant est simple : plus un test d'inclusion est rapide; Plus un volume englobant est complexe : plus la précision du test d'inclusion est élevée.

Les méthodes classiques de BVH placent des volumes englobants autour des primitives composant l'objet. Des extensions volumiques ont été proposées pour pallier le problème de l'inclusion complète d'un objet dans un autre. En effet, lorsqu'un objet se retrouve totalement inclus, aucun test de feuilles de la méthode classique ne permet de déterminer la présence d'une collision. Les travaux de Liu *et al.* [LqWhX07] et de Weller *et al.* [WZ09a, WZ09b] proposent des solutions de recouvrement de l'intérieur des objets par des volumes englobants considérés comme contenant de la matière. Ils proposent respectivement des extensions nommées *Inner Space Bounding Volume Hierarchy* et *Inner Sphère Tree.* Cette dernière méthode vise à évaluer le volume d'interpénétration efficacement, en donnant une estimation correcte du volume de collision.

Le choix d'un type de volume englobant doit être fait en prenant en compte divers critères [Ebe04] que nous allons énumérer. Parmi ces critères, la concordance de la forme est assez importante. Un volume englobant doit être adapté à la forme des objets à tester, dans le cas contraire des tests d'inclusions donneront de fausses réponses positives de nombreuses fois. Par exemple, pour un objet avec une dimension grandement supérieure aux autres *(e.g.* une hache), un volume englobant de type sphères contient un grand espace vide, alors qu'une boite (de type AABB ou OBB par exemple) est plus proche de l'objet à tester.

Le coût d'un test d'intersection entre deux hiérarchies de volumes englobants entre également dans les critères à prendre en compte. Ce coût peut être évalué selon l'équation 1.1 [WHG84, GLM96].

$$T = N_v \text{x } C_v + N_p \text{x } C_p + N_u \text{x } C_u \qquad (1.1)$$

Dans cette équation nous avons :

- T : le coût total de la fonction de test
- iV„ : le nombre de boites englobantes testées
- C_v: le coût du test de collision entre deux boites englobantes
- N_p: le nombre de paires de primitives testées
- C_p: le coût du test de collision entre deux primitives
- N_u: le nombre de paires de volumes englobants à mettre à jour
- C_u: le coût d'une mise à jour d'un volume englobant

La notion de mise à jour apparaissant ici, avec le paramètre $N_u x\ C_u$, concerne les mouvements de rotation. L'ensemble des volumes englobants est invariant par translation. Par contre, les volumes englobants de type AABB ou k-dop ne sont pas invariants par rotation. Pour ces deux types de volumes englobants, la notion d'alignement sur les axes est importante, il n'est pas possible d'appliquer une rotation de l'objet sans avoir à prendre en compte une modification du volume englobant.

De manière plus générale, on ne considère que le nombre de tests à effectuer pour la détection entre une primitive et une hiérarchie de volume englobant pour un arbre de profondeur n est de l'ordre de log (n**).**

Les tests d'intersections de volumes englobants composés de plans s'effectuent facilement. Il suffit de vérifier que les projections des boites englobâtes selon les différents axes définis par les plans indiquent qu'il existe au moins un plan séparant les deux objets.

a. *Construction de hiérarchies de volumes englobants*

Il existe deux méthodes principales de construction de hiérarchies de volumes englobants : de haut en bas et de bas en haut.

La méthode de construction de haut en bas consiste à partitionner l'ensemble des primitives de l'objet en deux (ou plus) sous-ensembles. Ces sous-ensembles sont ensuite associe a des volumes englobant. Ce partitionnement est répété jusqu'à à atteindre un nombre acceptable de primitives dans les feuilles de l'arbre.

La méthode de construction de bas en haut débuter par la création de l'ensemble des feuilles et regroupe ensuite les feuilles par ensemble de deux (ou plus) pour former des nouveaux nœuds. Ces nœuds sont alors eux même regroupé de manière itérative JQ à obtenir un seul et unique nœud père.

6.2.4. Approximation

Certains systèmes de détection de collision effectuent des approximations de fournir une estimation des directions d'impact. Ces approches provient de deux notions importantes. La première est liée au fait que la perception de réalisme dans une application de simulation est souvent inexacte, utilisateurs ne sont généralement pas capables de déterminer quand une réponse à une collision correspond à une réalité physique ou à une approximation. la seconde notion, qui est la notion de temps-critique prend son sens. On souhaite généralement pouvoir définir une borne maximum de temps pour l'exécution d'un cycle de

simulation (détection collision, réponse physique et visualisation). Cette borne peut être présente de répondre à des requêtes d'information des outils d'interactions, tel que systèmes à retour d'effort, ou simplement parce que l'on souhaite une animation fluide.

6.2.5. Accélération matérielle

L'évolution actuelle du matériel, tant des cartes graphiques que des processeurs, tend vers des architectures parallèles. Les algorithmes présentés sont pour la plupart parallélisables. Certains travaux se sont penchés sur la question de l'optimisation de ces algorithmes pour prendre en compte les contraintes inhérentes aux architectures parallèles. La parallélisation, pour être performante, doit : ***minimiser les temps de transfert de données et minimiser les zones bloquantes des algorithmes afin que les ressources puissent être exploitées à leur maximum***.

Certains des travaux portant sur uneparallélisation adaptée aux architectures matérielles actuelles. Nous abordons la question de l'exécution parallèle de tests sur des hiérarchies de volumes englobants, puis nous présentons certains des travaux concernant l'utilisation cartes graphiques. Cette partie nous permet également d'aborder la notion détection de collision basée sur des techniques de rendu.

6.2.5.1. Parallélisation de la comparaison de deux BVH

Un ensemble de méthodes vise la parallélisation de l'exécution des tests d'intersections entre des hiérarchies de volumes englobant [HFSQ01, KHH+09, TMT10]. En effet, les comparaisons entre volumes englobant sont répétées un grand nombre de fois et se prêtent particulièrement à la parallélisassions.

La parallélisassions devient rentable lorsque suffisamment de tests d'intersections doivent être effectués et lorsque ces tests sont correctement répartis sur les différents systèmes de calculs, qu'ils soient effectués sur des architectures multi-cœurs ou sur des cartes graphiques.

Une contrainte supplémentaire, liée à l'utilisation de cartes graphiques, provient de la taille limitée de la mémoire disponible pour stocker l'ensemble des informations. Il est nécessaire d'opter pour des solutions qui limitent le nombre de transferts d'information des processeurs vers la carte graphique [ZK07].

Lorsque peu de calculs sont effectués en parallèle, le temps nécessaire au transfert des données peut être supérieur au gain de temps apporté par la parallélisation. Gress *et al.* [GGK06] proposent de restreindre l'exécution parallèle de l'algorithme aux situations pour lesquelles le degré de parallélisme est suffisant. Ils parallélisent ainsi les tests d'intersection de volumes englobants uniquement lorsque le nombre de tests à effectuer atteint le nombre de 256.

6.2.5.2. Utilisation des cartes graphiques

L'architecture des cartes graphiques est particulière et permet certains types d'optimisation que nous évoquons ici. Les cartes graphiques ou GPU *(Graphics Processing Unit)* sont construites selon une architecture multi-cœurs possédant leur propre mémoire. Elles ont été développées afin de permettre l'accélération de certains types de calculs nécessaire au rendu d'images. Elles sont donc efficaces pour effectuer des calculs concernant l'analyse de la géométrie de scènes en 3D.

Après la présentation des optimisations sur GPU, nous revenons dans un dernier point sur l'ensemble des limitations que présente l'utilisation de cartes graphiques pour les méthodes de détection de collision.

I. Intersection basé image

Les cartes graphiques sont spécialisées pour le rendu de scènes 3D comportant de multiples objets. Récemment, un ensemble de méthodes a été proposé pour exploiter cette propriété afin d'accélérer les tests de détection de collision.

Parmi d'autres, HofT *et al.* utilisent une première phase grossière à partir d'une hiérarchie de volumes englobant dans l'espace objet (sur CPU : *Central processing Unit),* une deuxième phase est effectuée sur carte graphique pour les calculs de proximités entre des régions proches [HZLM01, HIZ+02]. Les *calculs* d'intersections se basent sur les techniques de rendus standards : la carte graphique permet d'identifier, selon un point de vue donné, si deux éléments possèdent les mêmes coordonnées géométriques et permet donc d'indiquer la présence d'une intersection.

La méthode de Govindaraju *et al.* [GKJ+05] utilise également une accélération sur carte graphique. L'ensemble des tests d'intersections de primitives effectué sur carte graphique. Cette méthode a également été étendue dans [GKLM07] afin d'effectuer l'ensemble des

tests grossiers de volumes englobant carte graphique. L'ensemble des tests grossiers est effectué par projection sur les différents axes du repère par rendu d'image.

L'ensemble de ces méthodes est limité par la résolution choisie pour les tests d'intersection entre primitives et ne permet pas la gestion des contacts. L'augmentation de la résolution du rendu image peut diminuer les problèmes d'approximation dus à la discrétisation des primitives, cependant, quelle que soit l'augmentation de la résolution, des approximations sont toujours présentes, problème de la gestion des contacts vient de la représentation discrète des données géométriques. La frontière entre deux objets est vue, elle aussi, selon la résolution choisie pour le rendu.

Les méthodes de Govindaraju *et al.* et de Jan *et al.* [GRLM03, JH08] évitent ce problème d'approximation des approches basées images en renvoyant l'ensemble des primitives vues comme en collision au CPU. Les calculs exacts sont alors effectués dans l'espace objet et ne contiennent plus d'erreurs liées à résolution. Ce système ne prend pas en compte le fait que certaines collisions peuvent simplement ne pas avoir *été* identifiées dans l'espace image.

Pour pallier le problème de la résolution, Govindaraju *et al.* [GLM04] ont également proposé une méthode basée sur l'ajout de boites englobantes de type OBB autour de chaque primitive, de manière à ce que la discrétisation voxels lors du rendu image ne manque pas de collisions.

II. Solveur sur carte graphique

Wong et Baciu ont porté un solveur d'équations polynomiales ainsi qu'un algorithme de calcul de la plus courte distance entre un point et un triangle sur carte graphique [WB05]. Ces implantations sur cartes graphiques se sont avérées plus performantes, malgré les temps de transfert nécessaires, que des implantations sur processeurs. Elles ont ainsi été utilisées pour des algorithmes de détection de collision sur des modèles déformables.

III. Élimination de paires d'objets

Sud *et al.* ont proposé une méthode d'optimisation basée sur des calculs géométriques effectués sur carte graphique permettant de trouver l'ensemble des paires d'objets à évaluer parmi un ensemble de *n* objets.

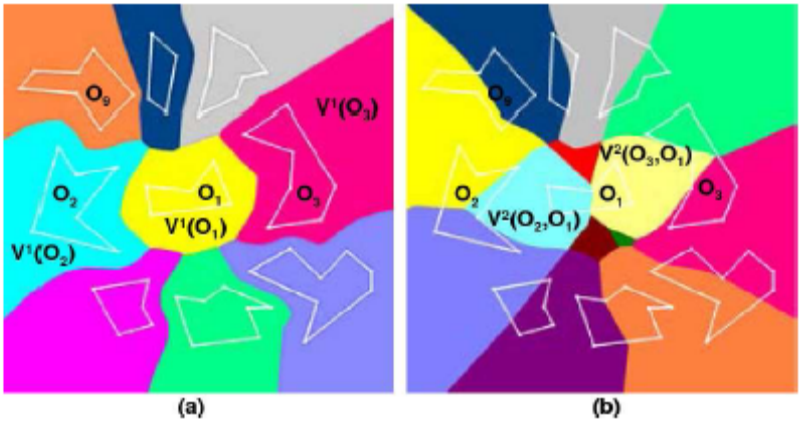


Figure 32: Diagramme de Voronoi du premier et deuxième ordre pour 9 polygones, a) Diagramme de Voronoi du premier ordre : chaque couleur représente l'ensemble des points les plus proches d'un polygone, le polygone O_1 à 8 voisins, b) Diagramme de Voronoi du deuxième ordre : chaque couleur représente les régions proches entre deux polygones, le polygone O_1 a alors uniquement 2 voisins. Image de [SGG+06].

La méthode de Sud *et al.* [SGG+06] construit un diagramme de Voronoi de deuxième ordre sur carte graphique. Ce diagramme de Voronoi permet de réduire, comme l'illustre la figure 32, le nombre de paires d'objets à tester parmi *n* objets. Sans utiliser de diagramme de Voronoi, tester les potentielles intersections entre *n* objets, nécessite *n2* tests. Son utilisation couplée à des tests sur des AABB permet de limiter ce nombre de tests. Une diminution des temps de calcul d'un facteur 30 à 50 a été observée par rapport à des tests sur des AABB seuls pour des environnements complexes déformables.

IV. Limitations

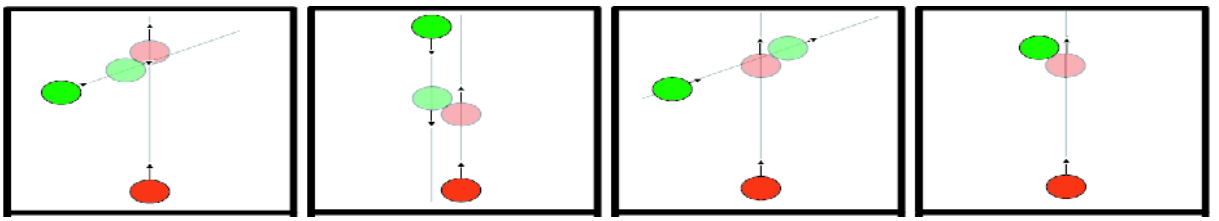
Les limitations actuelles des cartes graphiques concernent différents points, la latence induite par les temps de transfert d'information des processeurs vers les cartes graphiques est un problème important à prendre en considération. De plus, la taille mémoire des cartes graphiques est à l'heure actuelle plus restreinte que celle disponible pour les processeurs. Il n'est donc pas possible pour des scènes complexes, de transférer l'intégralité d'une simulation sur celles-ci. La précision des nombres représentés sur carte graphique est également plus faible que celle disponible sur CPU et entraîne donc des erreurs d'arrondis dans certaines.

7. Evitement de la collision

Tout au long de nos déplacements citadins nous sommes constamment amenés à nous éviter les uns les autres afin d'éviter une collision. Ceci met en relief le fait que nous

sommes en perceptuelle analyse de la direction et de la vitesse des personnes qui nous entourent. F.Lamarche et S.Donikian [LD04] nous présentent les quatre principaux types de collision *: frontale, statique, de dos et de côté* (figure 33).

Mais il existe également deux types d'entités avec lesquelles il est possible d'entrer en collision : les entités statiques et celles dynamiques.



Dan le dos **de face** **en arrière** **statique**

Figure. 33 : Quatre types de collision. L'entité dont on cherche à éviter la collision est en rouge

7.1. Adaptation pour l'évitement

Afin d'éviter une collision, différents paramètres peuvent être modifiés. En fonction du type de collision détectée, une adaptation de la vitesse peut être souhaitée telle une accélération ou une décélération, mais également un changement de direction tel l'évitement à droite ou à gauche. Le choix des paramètres à adapter dépend de la configuration spatiale et temporelle de l'éventuelle collision. Evitement d'entités statiques. Si l'objet à éviter est fixe, la technique la plus utilisée est celle de la modification de la planification de chemin. En d'autres termes cela consiste à recalculer le chemin à suivre en allant désormais de la position actuelle, en passant à droite ou à gauche de l'objet pour reprendre ensuite la direction de la destination souhaitée. Evitement d'entités dynamiques. G.Thomas et S.Donikian [TD00] propose une méthode minimisant le nombre d'interactions entre les piétons afin d'éviter la collision. Leur méthode prend en compte la distance entre les agents et choisis quelle règle d'évitement choisir pour minimiser les interactions. Avant chaque changement de direction, une prédiction de la future configuration est étudiée afin de déterminer si elle ne donnera pas lieu à une collision. Pelechano et al. [PAB07] dans leurs modèles à base de force propose, lorsqu'un piéton pénètre dans le rectangle d'influence présent autour de chaque agent, d'appliquer une force tangentielle afin de modifier légèrement la trajectoire et d'éviter ainsi la collision. S.Paris

et al. [PPD07] propose une méthode de prévision de ces collisions. Pour cela, il représente dans un espace en 3D, les directions et les vitesses de l'agent de référence sous forme conique, tandis que la position de l'agent voisin est représentée sous forme cylindrique représentant sa vitesse et sa direction. Les collisions futures sont les surfaces d'intersection entre le cylindre et le cône.

8. Conclusion

Dans ce chapitre Nous avons exposé les techniques de la navigation réactive. La première conclusion que l'on peut tirer des modèles présentés est qu'aucun de ceux-ci ne permet, seul, de gérer toutes les situations potentielles des mouvements de personnes. Certains sont plus axés sur l'obtention de performances de calcul, d'autres sur la robustesse de la résolution, certains donnent une grande part à l'organisation sociale, de ces modèles est donc de les exploiter en dehors du contexte pour lequel ils ont été créés, et validés. D'autre part, ces modèles n'abordent pas le caractère plus global de la décision relative au déplacement. Nous verrons par la suite que d'autres techniques sont nécessaires pour traiter ce sujet, et sont généralement utilisées conjointement aux modèles exposés ici.

Chapitre IV : Le modèle proposé

Chapitre IV

Le modèle proposé

1.Introduction

La recherche de modèles tentant de reproduire un comportement humain fait prendre conscience de la complexité des processus impliqués. Ainsi, la locomotion peut paraître simple puisqu'elle est en général apprise aux premiers mois de la vie mais elle se révèle difficile à mettre en œuvre dans un environnement virtuel. Il est entre autres nécessaire de percevoir l'environnement, de prévoir quels passages emprunter, de coordonner ses mouvements de manière à se déplacer le long du chemin en minimisant l'effort à fournir et d'éviter les obstacles apparaissant au dernier moment.

De nombreux modèles ont été proposés notamment pour la robotique et plus récemment pour l'animation d'humanoïdes virtuels. Certaines approches utilisent une représentation de l'environnement, planifient un chemin, puis suivent la trajectoire en évitant les obstacles dynamiques.

Nous avons proposé un modèle de navigation plus réaliste, la longueur de pas et la vitesse étant adaptées en fonction de la structure de l'environnement. La généricité du modèle le rend utilisable pour de la simulation en temps réel s'il est paramétré avec les lois de commandes extraites de l'analyse du déplacement humain.

1.1. Vue globale sur le modèle :

A partir de l'environnement 3D, une structure de données est construite pour traiter efficacement les taches de navigation. Lorsque l'utilisateur indique le point de destination pour un humanoïde, un chemin global est calculé, puis la trajectoire est optimisée.

Cette approche est simple, permets un calcul rapide mais le réalisme de l'animation produite pourrait être amélioré. Il serait souhaitable que la vitesse soit adaptée en fonction de différentes composantes de l'environnement et de la trajectoire, comme les changements de hauteur, les virages et les rétrécissements de l'espace. Ces adaptations devraient être faites progressivement afin de prendre en compte une certaine anticipation. La solution que nous allons développer par la suite prend en compte ces paramètres sans pour autant sacrifier le temps de calcul.

L'espace navigable est décomposé en zones, Pour prendre en compte les problèmes de navigation nous avons définis un modèle paramétrable de l'humanoïde et ses objectifs. Un troisième axe d'amélioration concerne la manière de traiter les virages importants.

Paramètres de l'humanoïde

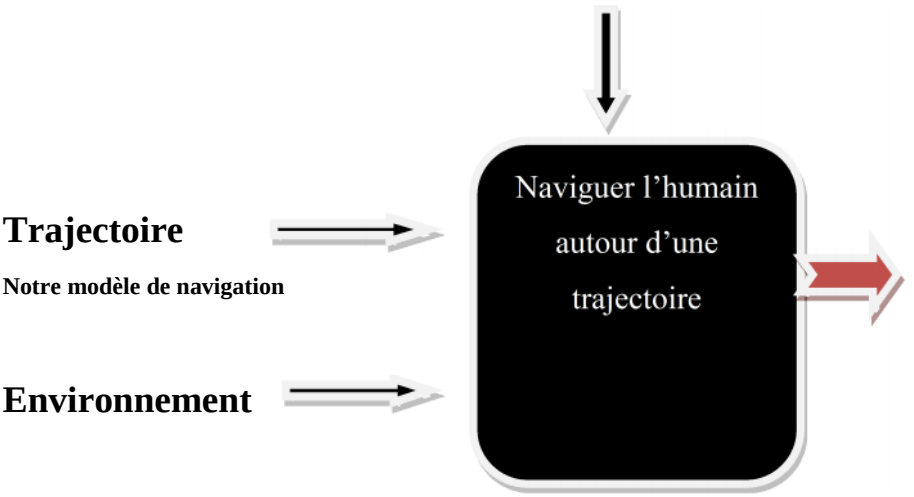


Figure. 34 Notre module de la navigation réactive prend en entrée une représentation de l'environnement ainsi que la trajectoire à suivre et calcule le chemin au cours du temps L'ajustement des paramètres permet une grande souplesse du modèle selon les besoins particuliers de l'application et une adaptation à la morphologie de chaque humanoïde.

2. La représentation de L'environnement et planification de chemin :

L'environnement virtuel est au centre de notre problématique. Il se doit d'être aussi proche de la réalité que possible, et doit permettre l'extraction de données non seulement pour le déroulement de la simulation, en vue d'une utilisation par les agents autonomes, mais aussi pour l'étude a posteriori, en vue de la caractérisation des résultats.

Ainsi, il va intégrer l'ensemble des informations nécessaires à la prise de décision de nos agents autonomes, que ce soit la géométrie et la topologie des lieux, ou encore l'emplacement des différents équipements. L'environnement va de plus fournir un accès aisé et rapide à un ensemble de données, tels que les flux de personnes ou la caractérisation des espaces visibles.

Pour permettre une navigation optimale des entités, nous allons présentés une organisation topologique des données la carte de cheminement, puis nous allons continuerons avec l'information proposé de l'environnement soit :

Statique : pré-calculs

ou

Dynamique : via des fonctionnalités nous finirons

2.1. Génération de la carte de cheminement « Grille »

La connaissance des informations sur l'environnement est une phase primordiale pour déterminer l'espace navigable et non- navigable, autrement dit les formes géométriques (objets statiques).

Ce qui nous conduit à réaliser l'un des modèles de représentation spatiale, modèle approximative à base de grille « en anglais Grid ».

Ce modèle est plus pratique en animation comportementale. Il nous conduit à le générer par des tests sur tous les espaces de l'environnement. Ce test se fait par la diffusion d'ensemble des nœuds sur toute la scène, ces nœuds capables de détecter l'espace par leurs positions navigables ou non navigables.

Le nœud aura une structure spéciale qu'on va expliquer ci-après.

2.1.1. Structure et fonctionnement du nœud

Avant d'entamer la composition de la structure d'un nœud, il y a lieu de préciser sa forme, c'est une représentation de forme sphérique au milieu d'un champ de détection de collision de forme à base carrée (Y = coté) et sa Hauteur est évolutive (H) selon les données de départ

Le champ de détection de collision est composée de cinq (5) parties, chaque partie a une fonction déterminée exécutable selon la classification (N° qui lui a été attribué), le principe est de détecter la collision avec les objets statiques.

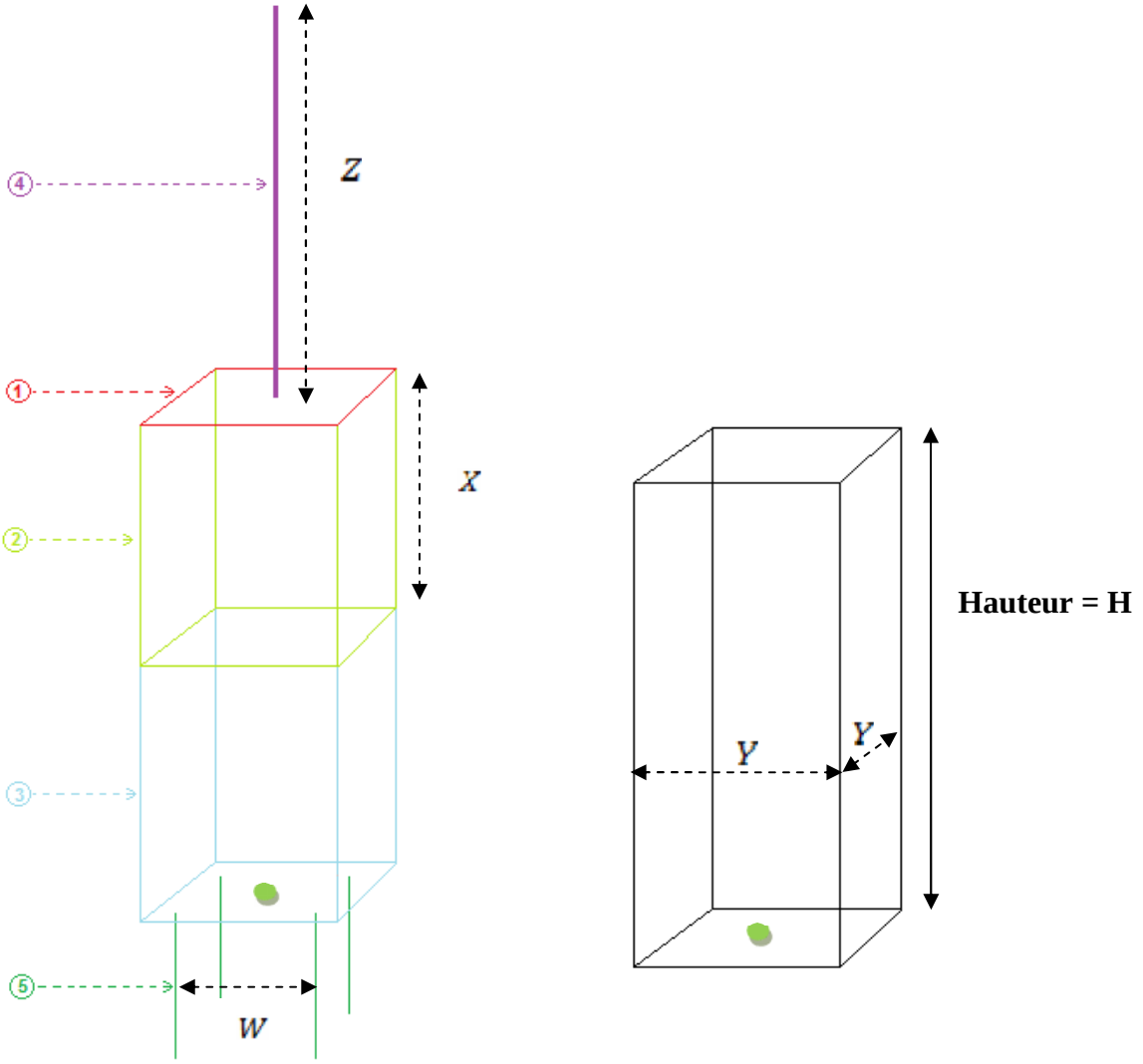


Figure 35: représente la structure générale du nœud

✓ **La partie « 1 » :**

C'est une représentation cubique dont sa hauteur est très réduite permet de détecter une collision avec les objets statiques. Et en cas où il détecte une collision le nœud se déplace en hauteur d'égale mesure du nœud.

✓ **La partie « 2 » :**

Le même principe de détection de collision que la partie 1, mais la hauteur de cette partie égale à « X »:

$$X = \frac{\text{Hauteur}}{2}$$

Si en cas de détection d'une collision, le nœud se déplace de la moitié de sa hauteur « X ».

✓ **La partie « 3 » :**

Cette partie est identique à la partie « 2 », mais le déplacement du nœud se fait d'égale distance modifiable selon les besoins, dont on a retenu dans notre programme la mesure de $10\ Cm$

✓ **La partie « 4 » :**

Un Ray-Caste[1] dont sa longueur égale à celle de l'environnement, en cas de détection d'un objet, l'algorithme crée un autre nœud dans l'emplacement où détecte la collision.

✓ **La partie « 5 » :**

Quatre Ray-Caste[1] avec une longueur et écartement modifiables selon les besoins, dans notre application on a choisi $50\ Cm$ de longueur et $30\ Cm$ d'écartement.

Pour expliquer la fonction de ces Ray-Castes on a simulé cette dernière comme des pieds d'une chaise. Si les quatre pieds sont équilibrés le nœud est réussi, sinon le nœud se détruit.

L'écartement entre les Ray-Castes représenté par « W » (modifiables pour le besoin), pour une meilleur utilisation de l'espace navigable.

L'organigramme ci-dessous illustre le mécanisme de génération de la représentation.

[1] C'est un détecteur fonctionnant comme un radar : lançant un rayon du point fixe vers un objet afin d'obtenir des informations utiles.

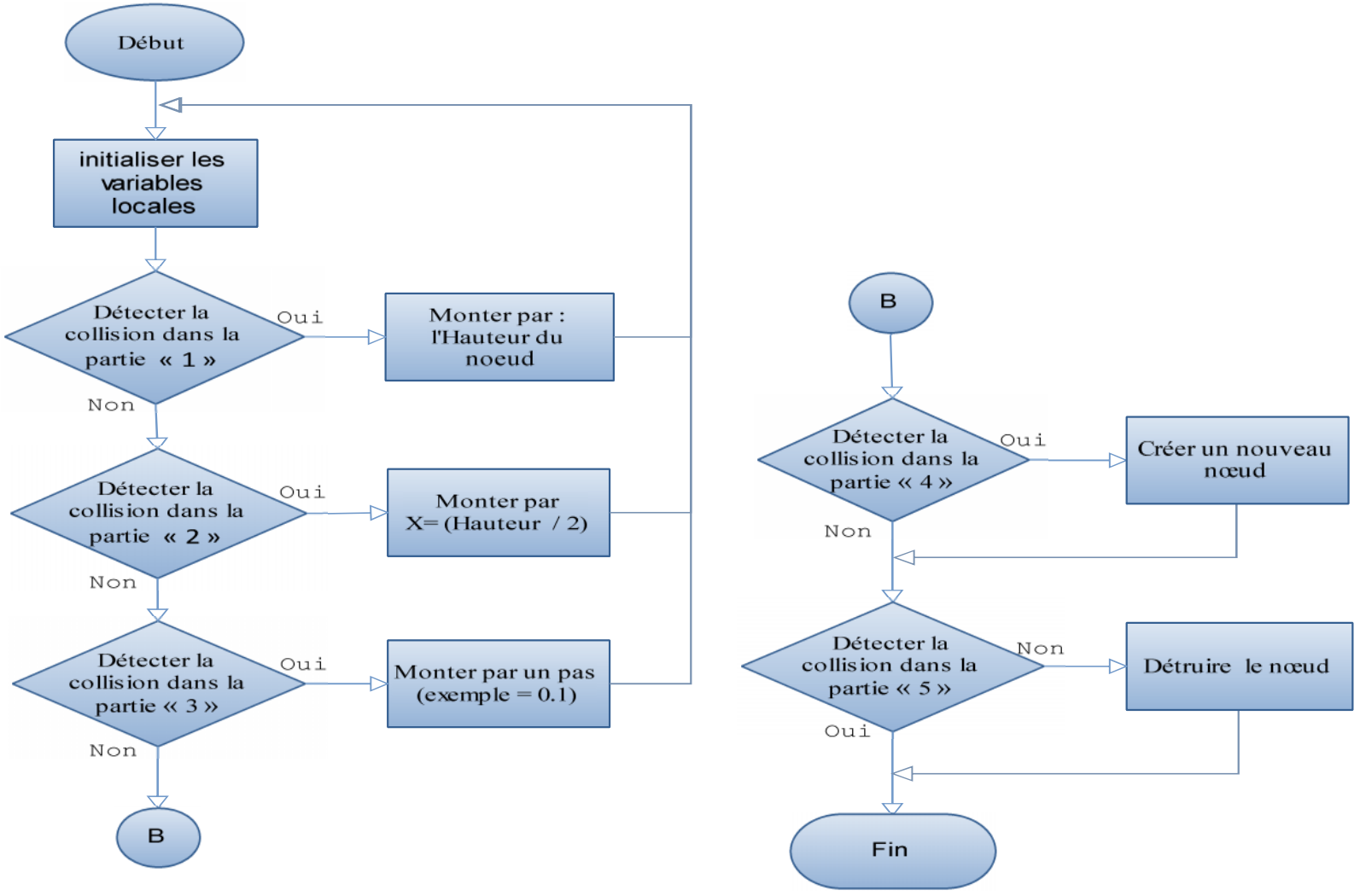


Figure 36 représente les étapes de la fonction du nœud

2.1.2. La diffusion des nœuds

Après la génération du nœud, on provoque la diffusion des nœuds sur toute la surface de la scène. La diffusion se fait pour parcourir tous les points de la scène, nous précisons que les distances entre les nœuds sont équidistantes (égale à « Y » ce qui donne une ramification totale de la scène.

Chacun nœud a son propre rôle pour tester l'espace où l'humanoïde virtuel peut se déplacer

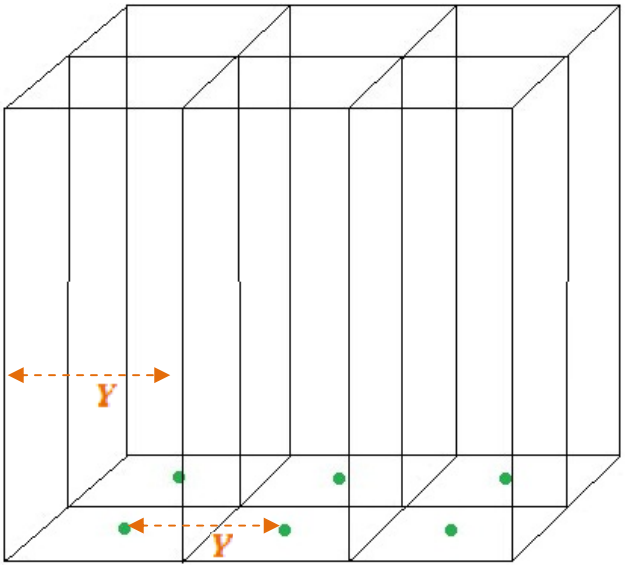


Figure 37: représente la disposision de six (6) neouds dans l'environnement

2.1.3. La connexion des nœuds

A la phase finale de la ramification des nœuds, chaque nœud procédera à tester son propre espace. A la suite de cette opération, il en résulte des espaces navigables et non-navigables dans l'environnement.

Chacun de ces nœuds « navigables » doit se relier avec son proche voisin (adjacent), qui nous permet d'obtenir un graphe des nœuds.

Figure 38 : la connection des noeuds entre eux

2.1.4. L'inondation se remplit « Flood fill »

Après avoir constitué le graphe, on utilise la technique « Flood fill » , qui nous a permis de nous donner l'image de la scène dans laquelle une constitution des groupes de nœuds connectés entre eux.

Alors on obtiendra une carte de cheminement partagée en groupe de nœuds reliés entre eux qu'on pourrait les considérer comme des lacs (lacs des nœuds).

Cette technique nous donne une facilité d'exploration d'un nœud dans le graphe lors de la recherche.

Les figures 39 et 40 illustrent cette technique, chaque groupe de nœuds aura une couleur distincte.

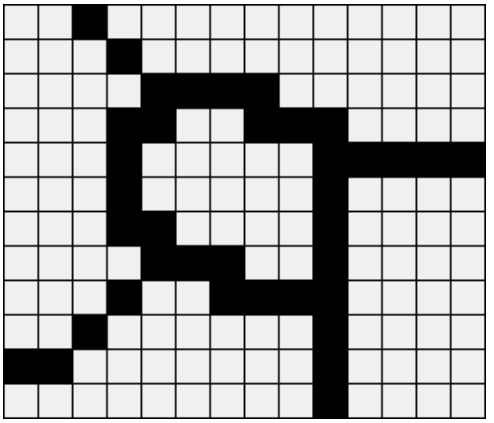


Figure 39: Catre de cheminement son "flood fill"

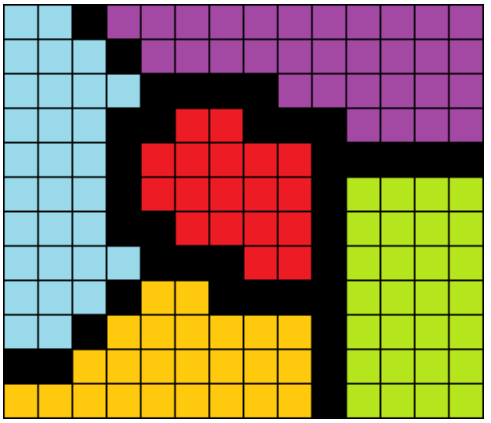


Figure 40: Catre de cheminement avec "flood fill"

2.1.5. Exporté la carte de cheminement sur un support physique

La construction de la carte de cheminement avec toutes les étapes qu'on a expliqué précédemment prennent beaucoup de temps. Pour cela on a proposé de sauvegarder ces informations sur un support physique.

Ce mécanisme se doit de sauvegarder les données en format spécifique (dans notre cas en utilise le format XML). Pour éviter le recalcule de la carte de cheminement et la persistance des données, une sauvegarde dans un autre format (Epx. Fichier TEXT) reste acceptable.

La figure ci-dessous illustre le mécanisme de génération de la carte de cheminement.

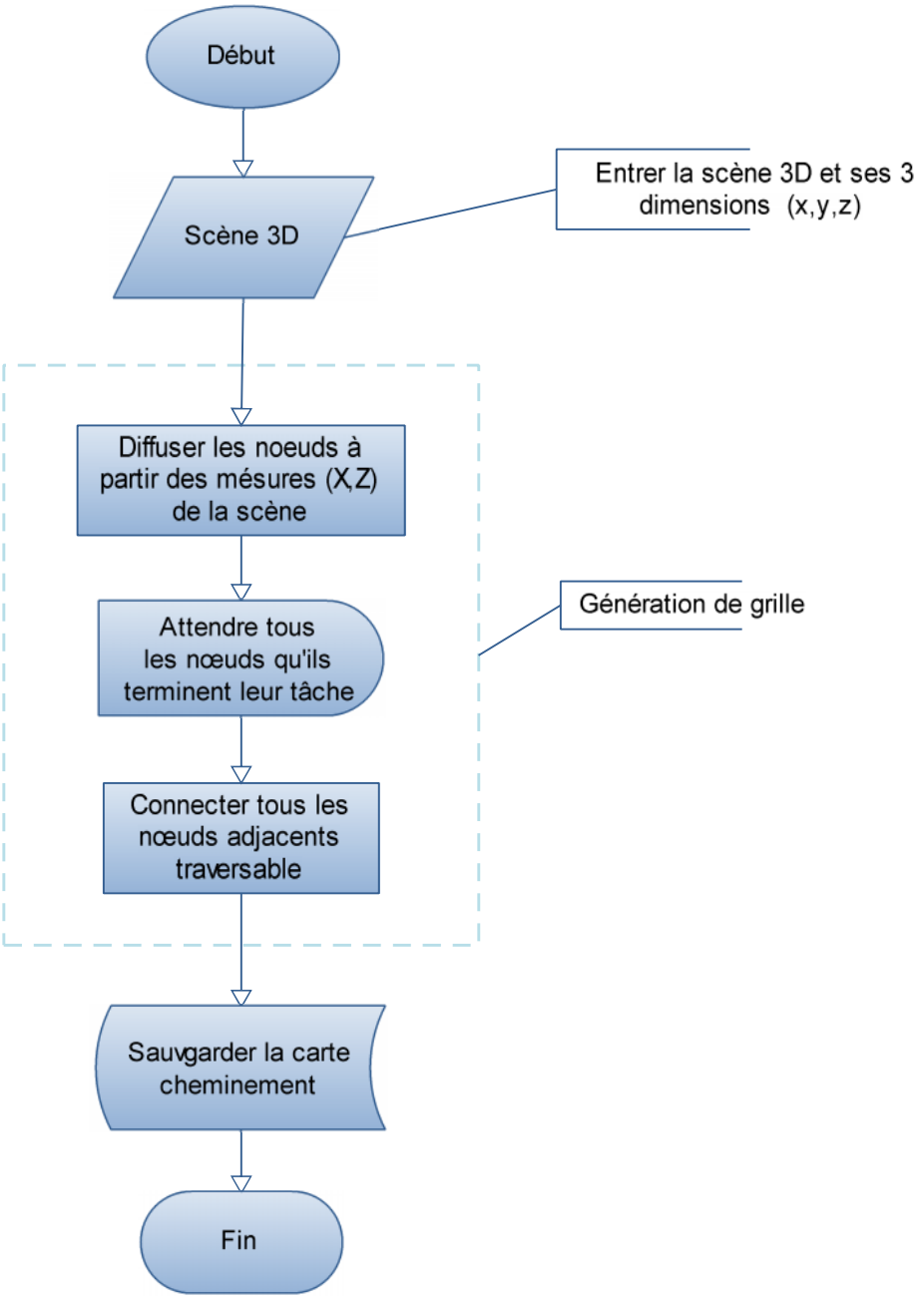


Figure. 41représente le mécanisme de génération de la carte de cheminement

3. Notre modèle d'humain virtuel

❖ Principe général :

Notre modèle d'humanoïde est une spécialisation de cet interacteur, augmentant ses capacités par l'ajout d'attributs et de fonctionnalités.

Notre humain virtuel dispose donc de toutes les propriétés conceptuelles d'un interacteur, auxquelles vont s'ajouter des facultés plus concrètes et incarnées telles que la perception visuelle ou la navigation réactive. Nous allons voir que l'ensemble de ces tâches basiques sont intégrées au sein du modèle d'humain virtuel, pour obtenir un agent totalement autonome capable de prendre ses propres décisions.

Cette intégration repose sur la fameuse pyramide comportementale de A. Newell (Figure 42)

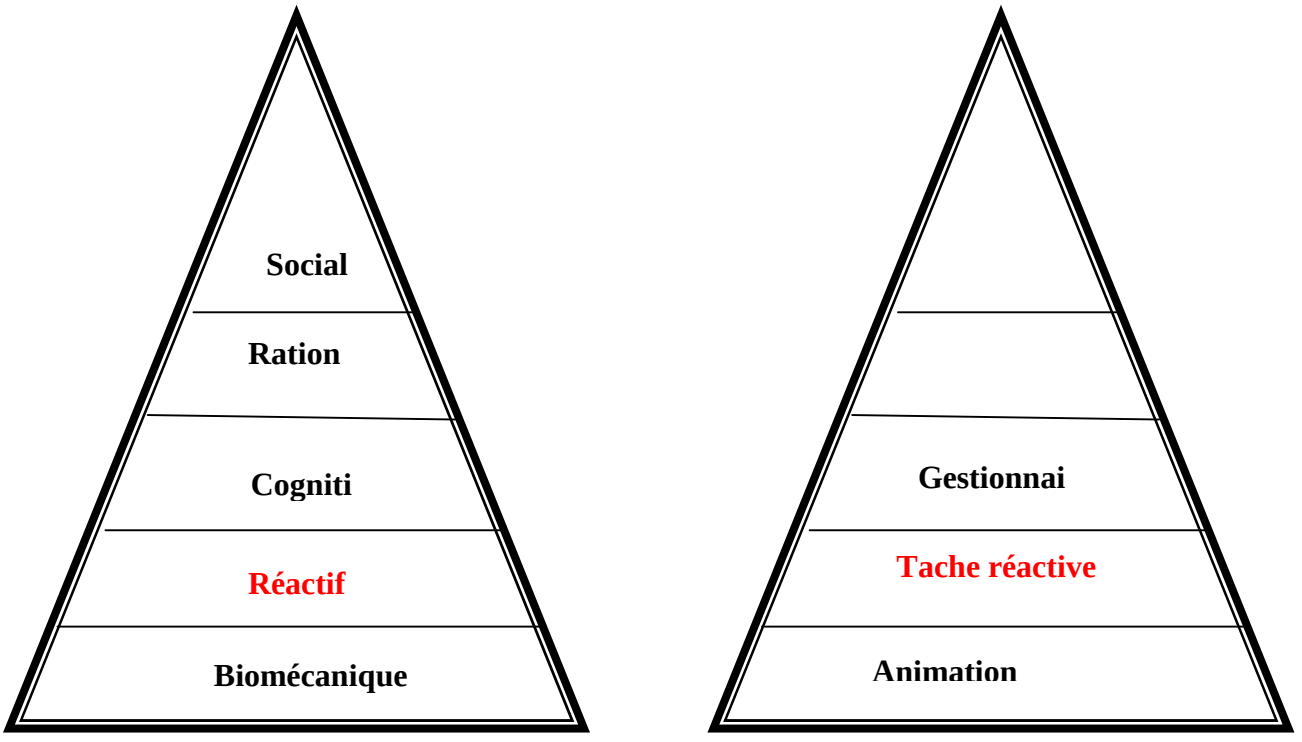


Figure. 42: A gauche, la pyramide comportementale originelle. A droite, L'équivalent de la pyramide comportementale dans notre modèle.

Détaillons chacun des étages de la pyramide telle qu'il se présente dans notre modèle.

Biomécanique La base de la pyramide est constituée des tâches les moins conscientes pour un humanoïde. Elle peut être considérée comme automatique, et à ce titre s'exécute avec la plus haute fréquence du système : 20Hz. Ce sont les modules de locomotion et d'animation qui incarnent la biomécanique de l'humain virtuel dans notre modèle.

Réactif : Le premier étage de la pyramide est constitué de tâches simples, atomiques, pour l'humanoïde. Elles peuvent être rafraîchies automatiquement avec une fréquence faible, et/ou en réaction à des évènements. Ces tâches sont disponibles quel que soit l'humanoïde simulé, et constituent leur base comportementale tandis que l'ensemble de leurs données sont directement accessibles par ce dernier. Nous pouvons ***citer dans les tâches de cet étage les modules de planification de chemin, d'optimisation visuelle, d'observation et de mise à jour de la base de connaissance, ou encore d'évitement de collision.***

3.1. Accéder aux informations de l'environnement :

La première nécessité d'un être humain est de percevoir son environnement. Cette perception est alors synthétisée et stockée dans la mémoire pour son utilisation ultérieure.

Notre modèle d'humanoïde applique ce principe à la lettre. Nous allons donc décrire ses capacités de perception de l'environnement, puis l'organisation de sa mémoire et les informations qu'il y stocke.

3.1.1. Perception :

Les humains peuvent percevoir un champ visuel de 120° à 180°, le dernier est le plus utilisé. Actuellement notre système est placé pour détecter des objets tombés dans un champ de vision de 90° à droite et à gauche la direction du mouvement. Ceci est calculé à partir du produit scalaire entre la direction du vecteur de mouvement et le vecteur joignant la position actuelle avec chaque objet dans la chambre. Puisque le produit scalaire nous donne le cosinus de l'angle entre les deux vecteurs, si cette valeur est plus grande que 0 elle signifie les chutes d'objets dans la zone de vision de l'agent.

Dans la figure 4.4, l'agent perçoit son environnement, il lance des rayons à travers les autres objets qui existent dans la même chambre, puis par le calcul du cosinus des angles entre la direction de mouvement et des rayons, l'agent i détermine quel objet existe dans sa zone de vision, mur, l'obstacle ou un autre agent j.

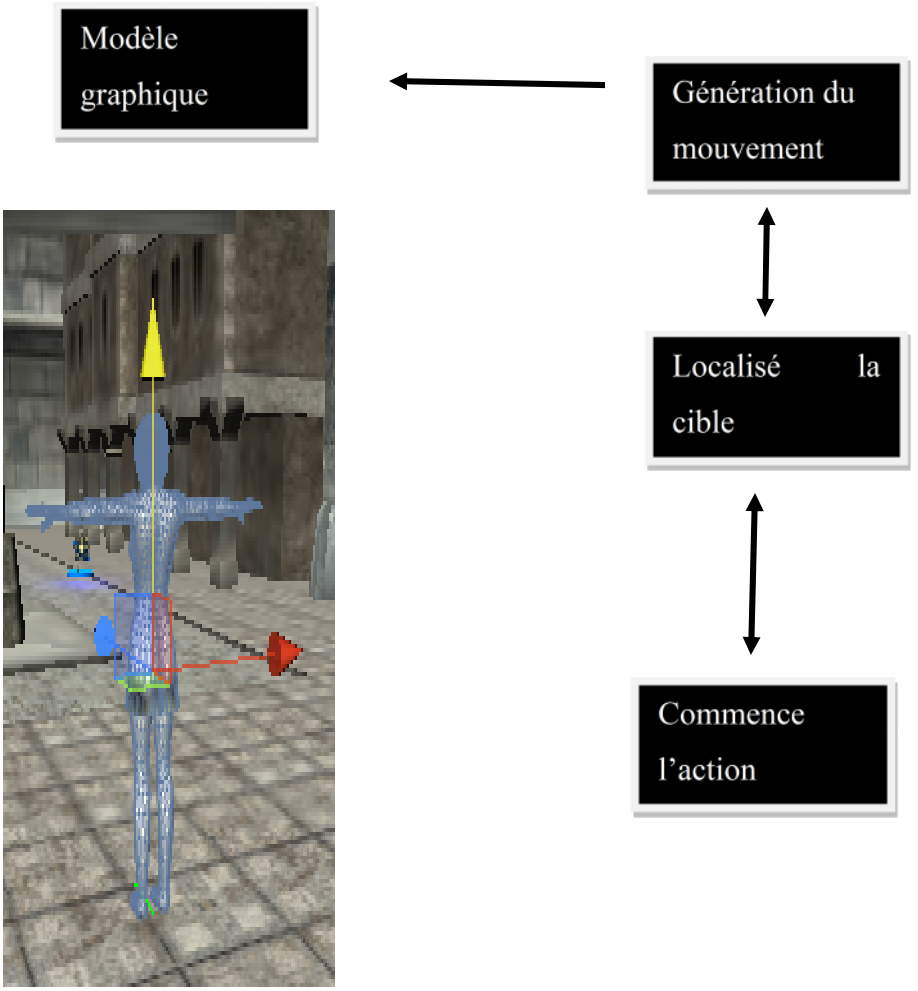


Figure 43 modelés humanoïdes

3.2. Conclusion

Nous venons de présenter les deux premiers mécanismes de l'humain virtuel : sa perception et sa mémoire. La perception constitue le premier comportement réactif de notre modèle. Elle permet d'extraire la sémantique visible de l'environnement, et donc de rendre nos entités non omniscientes. La mémoire est complémentaire à cette perception, permettant la persistance des informations récupérées. Nous avons ainsi proposé un modèle simplifié de connaissance topologique répondant aux contraintes de stockage imposées par nos objectifs. Il a tout de même été évoqué la possibilité d'étendre le mécanisme mémoriel de manière générique.

Nous verrons que ces deux mécanismes sont fondamentaux dans notre architecture d'humain virtuel, car ils participent à la plupart des autres comportements réactifs ou cognitifs.

4. Navigation Réactive

La planification de chemin, a permis de réaliser une décision de déplacement presque exclusivement basée sur les obstacles statiques de l'environnement. Afin de considérer la dynamique d'autres entités, l'humain virtuel nécessite un comportement complémentaire considérant l'environnement d'un point de vue plus local et immédiat. Cette tâche, dénommée ***navigation réactive***, va ainsi permettre à l'agent autonome d'éviter les collisions avec ses voisins, tout en essayant de respecter au mieux les directives des autres comportements.

Son objectif global est de permettre à l'entité de se rendre à sa destination, qui est représentée par la direction choisie lors de la planification de chemin.

4.1. Architecture de navigation

Les études sociologiques et cognitives de [LW92, RQ98, Gof71], montre que la navigation de l'être humain s'avère être caractérisée par un compromis entre plusieurs règles de comportement. Ces règles se composé en trois catégories :

- **Optimisation de la trajectoire.** Lorsqu'il navigue, l'être humain a tendance à optimiser sa trajectoire en minimisant l'angle entre sa direction et sa cible, tout en empruntant un chemin minimisant la dépense énergétique. D'autre part, la détermination de la cible courante (i.e. de la direction vers laquelle se diriger), se fait par l'utilisation de la perception. Autrement dit, l'homme a tendance à se diriger plus ou moins en ligne droite vers la zone visible appartenant à son chemin, si le chemin ainsi créé est libre d'obstacles. Cependant, dans la réalité, la ligne droite est une tendance mais la trajectoire s'avère exhiber une légère courbure qui pourrait être due à l'inertie du piéton [BJ03].
- **Respect de l'espace personnel.** L'être humain, lorsqu'il navigue dans un environnement peuplé cherche à maintenir une certaine distance par rapport aux obstacles statiques et aux autres humain peuplant l'environnement. Cette distance caractérise l'espace personnel, une zone libre autour du l'être humain dans laquelle seules les personnes appartenant au cercle des connaissances sont admises. Le non-respect de cette zone peut traduire deux types de situation :

    1. Soit il s'agit d'un groupe d'humain évoluant ensemble.

2. Soit la densité de population est trop élevée pour pouvoir respecter cette contrainte. Cette contrainte s'avère donc être une contrainte lâche dont le respect dépend grandement de la densité de population. En environnement très dense, elle n'est plus applicable.

- **Évitement de collision.** Chaque humain naviguant dans une zone prend en compte les autres humains de son voisinage proche et adopte une réaction en fonction d'une éventuelle collision détectée. Cette réaction dépend d'un certain nombre de règles sociales [Tho99] ainsi que de la configuration de l'interaction: collision de face, collision arrière...

Pour bien respecte Les règles l'espace personnel ainsi que d'évitement de collision ont besoin d'autre informations sur le voisinage de l'entité. Ce voisinage se traduit par un ensemble d'entités, typologiquement proches de l'entité concernée par la navigation, avec lesquelles une interaction éventuelle peut se produire. La règle de respect de l'espace personnel travaille sur une aire d'influence à rayon limité autour de l'entité alors que la règle d'évitement de collision travaille en priorité sur un temps de prédiction de collision, dépendant grandement de la densité de population. Ces propriétés soulèvent le problème crucial de la détection des voisins d'une entité et surtout de sa complexité.

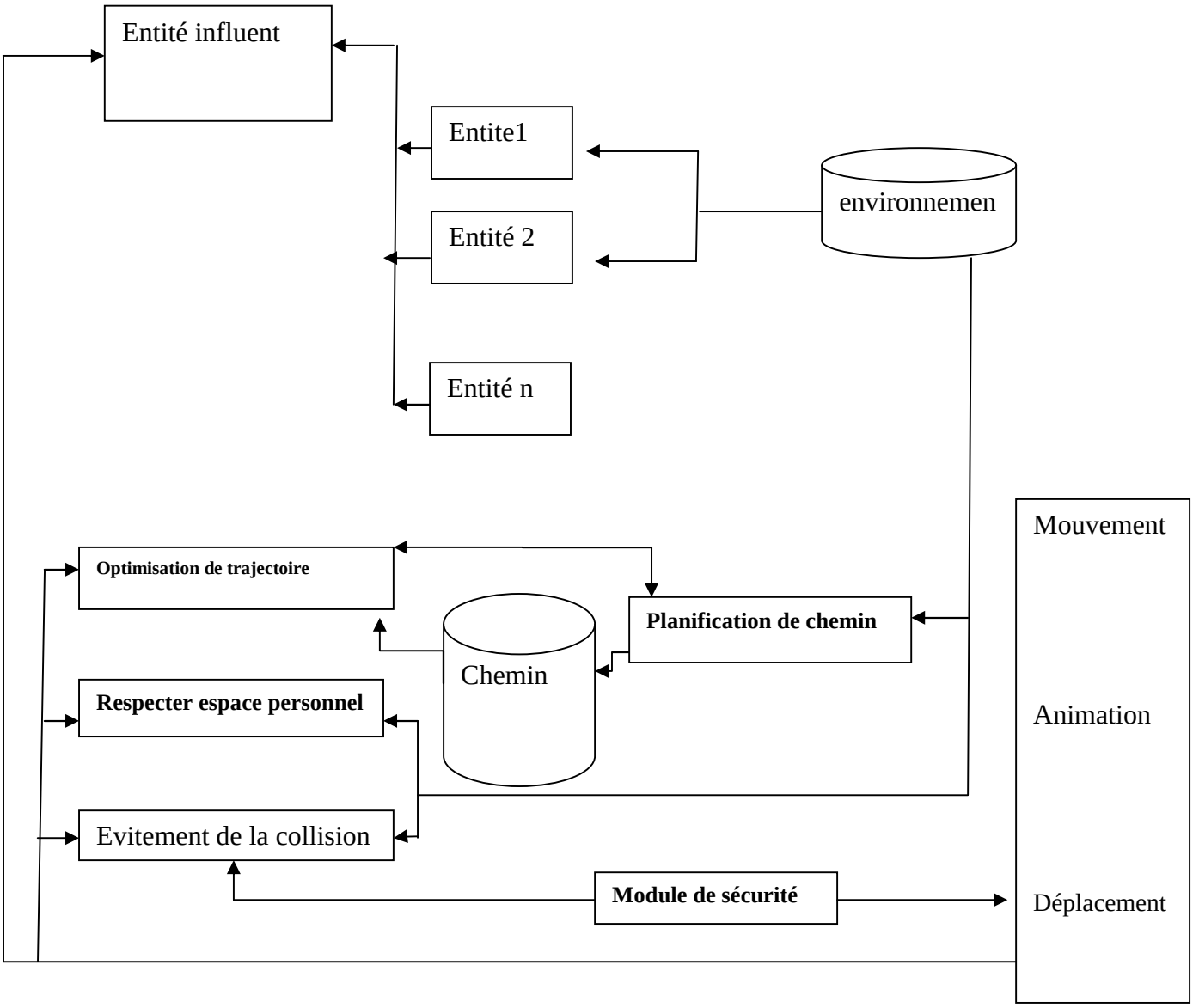


Figure. 44 Architecture de système

4.2. Modélisation de la navigation

Après l'élaboration de la carte de cheminement on procède à la modélisation de la navigation.

On fait appel à la carte de cheminement dans laquelle sont consignés les renseignements sur la géométrie de la scène. L'exploitation de cette carte sera par l'application de l'algorithme A* dans le but de l'obtention de chemin optimum.

Aussi on essaie de procéder d'une façon systématique à éviter les obstacles statiques, dynamiques et éventuellement tous les obstacles imprévisibles par exemple une obstruction d'une route (l'écroulement d'un mure).

4.2.1. L'exploitation de la carte de cheminement

Au niveau de cette phase on fait appel à la carte de cheminement et la mettre dans la RAM (Random Access Memory).

Cette procédure offre à l'agent l'accès aux données consignées dans la RAM pour les utiliser dans le calcule du chemin optimale (algorithme A*).

4.2.2. La recherche d'un chemin optimal « Algorithme de A* »

Le principe de l'algorithme réside dans le fait que pour chaque nœud de la carte, il continue sa recherche en prenant en compte le nœud voisin ayant le coût le plus faible, pour l'obtention de chemin optimal.

Le but de cette opération est de trouver le nœud cible. L'obtention de ce dernier nous assura du passage par le chemin optimal.

L'obtention des nœuds voisins dépend de la représentation de l'environnement.

L'algorithme utilise deux listes, « **liste-ouverte »** et **« liste-fermée »,** pour le stockage des nœuds pendant l'opération.

Le principe général de l'algorithme A* est l'évaluation de coût totale d'un nœud :

Coût total (F) = Coût depuis la source (G) + Coût vers la destination (H).

❖ Méthodes de calcul de l'heuristique

Les techniques utilisées pour calculer le coût vers une destination donnée (distance heuristique) qui on a étudié dans le chapitre précédent appliquée seulement dans le cas d'environnement deux dimensions.

Cette méthode on l'a utilisée par extension pour résoudre notre cas de trois dimensions (3D)

Pour calculer la distance heuristique entre deux points P1et P2. Leurs positions représentées par les trois coordonnées comme suit: $P1\begin{pmatrix} x1 \\ y1 \\ z1 \end{pmatrix}$, $P2\begin{pmatrix} x2 \\ y2 \\ z2 \end{pmatrix}$.

Le calcul de la distance entre ces deux points se fait selon la formule de calcul choisie:

✓ La distance Manhattan « Manhattan distance »

La fonction de calcul de la distance Manhattan est:

$$h(n) = abs(x2 - x1) + abs(y2 - y1) + abs(z2 - z1)$$

✓ La distance diagonale « Diagonal distance »

Le calcul de la distance diagonale nous impose à déterminer deux parties : (partie1 et partie2), comme il est illustré dans la figure 41.

La première partie représente la distance horizontale (2D) et la deuxième partie représente une distance 3D ayant une forme oblique.

On doit déterminer un point qu'on va appeler « M » se point se situe entre les deux parties qu'on va symboliser par :

$$M\begin{pmatrix} xm \\ ym \\ zm \end{pmatrix}$$

Le calcule de M se fait par l'application de la formule ci après :

$$M = ((x2 - (y2 - y1)), (y2 - (y2 - y1)), (z2 - (y2 - y1)))$$

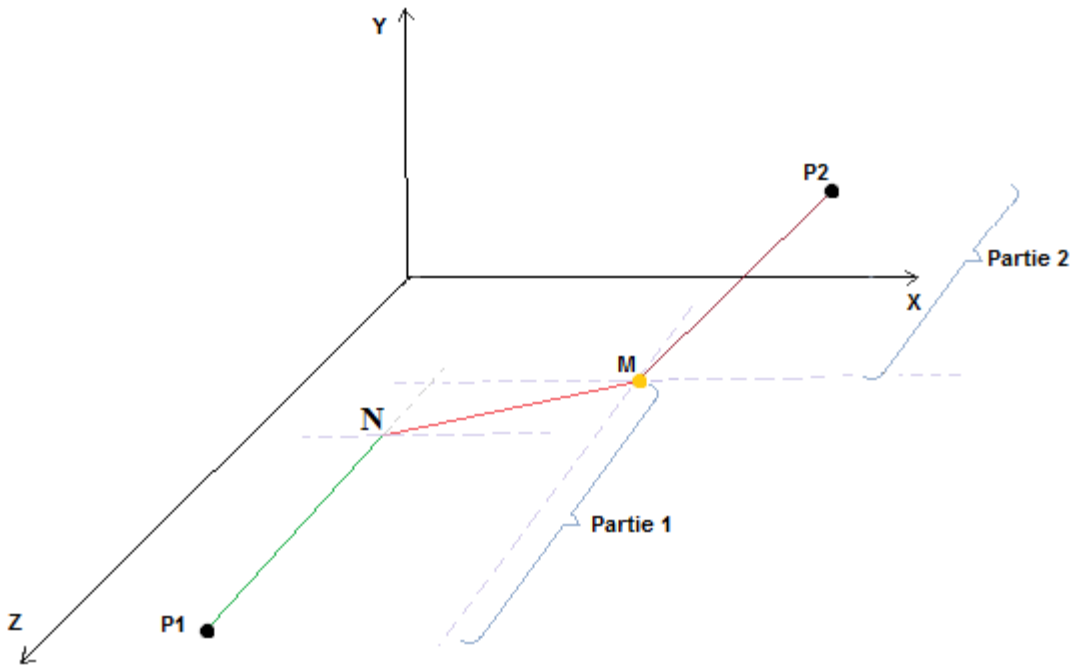


Figure. 45: signe algébrique représente la distance diagonale en 3d

Après avoir fixé le point M on procède au calcul de la distance de P1 à M (partie1)

- Partie 1 :
  Cette partie est composée de deux sous parties : la distance droite P1 à N et la partie diagonale qu'on désigne par N à M.

$$h_{diagonal} = Sqrt(2 \times \left(Min\left(abs(mx - x1), abs(z2 - z1)\right)\right)^2)$$

$$h_{droit} = Max\left(abs(xm - x1), abs(z2 - z1)\right) - Min\left(abs(xm - x1), abs(z2 - z1)\right)$$

$$distance_{partie1} = h_{diagonal} + h_{droit}$$

- Partie 2 :
  La distance de la partie 2 est égale à l'Euclidean en 3D de « M » jusqu'au point « P2 »

Après avoir calculé la distance des deux parties, on additionne leur valeur que nous obtenons la distance diagonale en 3D par l'application de la formule suivante :

$$H = distance_{partie1} + distance_{partie2}$$

✓ La distance Euclidean « Euclidean distance »

En appliquant la méthode de calcul de la distance Euclidean nous obtenons le résultat de la fonction suivante :

$$H = Sqrt((x2-x1)^2 + (y2-y1)^2 + (z2-z1)^2)$$

Pour bien illustrer cet algorithme, on a la présenté sous une forme d'un organigrammedans la figure 46.

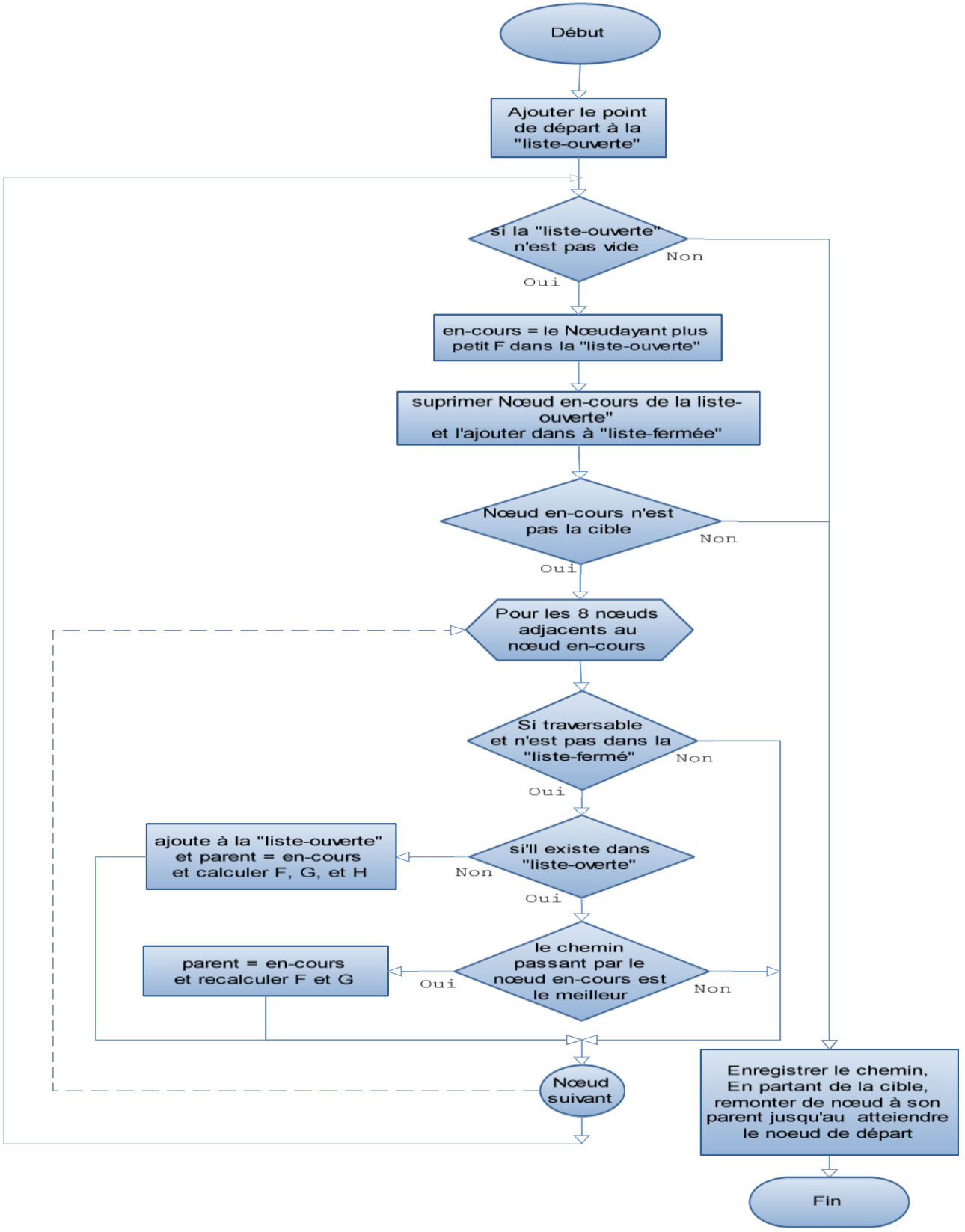


Figure. 46 représentes le déroulement de l'algorithme A*

4.2.3. Comment éviter les obstacles ?

L'évitement des obstacles dans notre environnement virtuel se présente selon leur nature: statiques, dynamiques et imprévisibles.

a- Détection de voisinage et évitement de collision :

Lors de nos déplacements, nous sommes constamment amenés à nous éviter les uns les autres afin d'éviter une collision. Ceci met en relief le fait que nous sommes en perpétuelle analyse de la direction et de la vitesse des personnes qui nous entourent Il existe également deux types d'entités avec lesquelles il est possible d'entrer en collision : les entités statiques et celles dynamiques. Afin D'éviter ces collisions, la meilleure solution est de savoir les prédire.

b- Connaissance de l'entourage :

Différentes techniques ont été proposées comme les Bin-Lattices de C.Reynolds [Rey00] correspondant à une subdivision de l'espace en boîtes appelées bins l'approche à base de règles inventé par Reynolds, cette approche permet de simuler des comportements de groupes d'entités à partir d'un petit nombre de règles appelées comportements appliqués à chacune des entités. Ce modèle a été utilisé pour simuler des bancs de poissons, des vols groupés d'oiseaux et donne un résultat plus réaliste.

La triangulation de Delaunay filtré (Figure. 06 chapitre1) est également utilisée par F.Lamarche propose de l'utiliser entre les personne [LD04] ainsi chaque personne est relié à ses voisins eux-mêmes reliés à leurs propres voisins Deux types de filtrages sont proposés, l'un ayant pour effet d'augmenter le nombre de triangles produits afin d'affiner la représentation de l'environnement, l'autre de le diminuer afin de tenir compte de la visibilité pour l'élaboration d'un graphe de voisinage.

Premièrement l'ajout progressif de contraintes représentant les goulets d'étranglement Deuxièmement, concernant son application aux graphes de voisinage, la triangulation de Delaunay est filtrée sur un principe équivalent à la triangulation contrainte, en supprimant Les arêtes intersectées des obstacles de l'environnement.

Afin d'éviter une collision, différents paramètres peuvent être modifiés. En fonction du type de collision détectée, une adaptation de la vitesse peut être souhaitée telle une accélération ou une décélération, mais également un changement de direction tel

l'évitement à droite ou à gauche. Le choix des paramètres à adapter dépend de la configuration spatiale et temporelle de l'éventuelle collision.

Evitement d'entités dynamiques : G.Thomas et S.Donikian [TD00] propose une méthode minimisant le nombre d'interactions entre les personnes afin d'éviter la collision. Leur méthode prend en compte la distance entre les personnes et choisis quelle règle d'évitement choisir pour minimiser les interactions. Avant chaque changement de direction, une prédiction de la future configuration est étudiée afin de déterminer si elle ne donnera pas lieu à une collision.

Pelechano et al. [PAB07] dans leurs modèles à base de force propose, lorsqu'une personne pénètre dans le rectangle d'influence présent autour de chaque agent, d'appliquer une force tangentielle afin de modifier légèrement la trajectoire et d'éviter ainsi la collision. S.Paris et al. [PPD07] propose une méthode de prévision de ces collisions. Pour cela, il représente dans un espace en 3D, les directions et la vitesse de personne de référence sous forme conique, tandis que la position de l'agent voisin est représentée sous forme cylindrique représentant sa vitesse et sa direction. Les collisions futures sont les surfaces d'intersection entre le cylindre et le cône.

V. *Notre algorithme*

Notre approche basée sur une combinaison de des informations géométriques et des règles psychologiques pour permettre une grande variété de comportements ressemblant à ceux de personnes réelles. Utilise attributs psychologiques et règles géométriques (distance de zones d'influence, par rapport angles de direction (90° ou 45°)) pour éliminer les artefacts irréalistes et de permettre à nouveau comportements :

- Poussant à travers un groupe.
- Agents qui tombent et deviennent des obstacles.
- Réaction en temps réel à des changements dans l'environnement.

I. Evitement de collision pour les obstacles

❖ La prévention de collision :

Pour chaque obstacle, un mur et un personne nous avons besoin pour calculer sa distance à personne i et si elle est suffisamment proche (entre dans la rectangle d'influence), puis on calculer l'angle entre l'agent i est dans la direction souhaitée et la ligne reliant le centre de

l'agent i et l'obstacle. Cette l'information est utilisée pour déterminer si elle tombe à l'intérieur du rectangle d'influence (figure a).

La distance et l'angle fournir suffisamment d'informations pour établir la pertinence de cet obstacle est de la trajectoire. Comme ils naviguer dans le l'environnement.

❖ Prévention d'autre Agent

Pour simuler le comportement d'homme nous incluons les règles qui modifient certains paramètres qui influent sur les personnes pour éviter les obstacles.

Les paramètres sont les suivants:

- Distance aux obstacles
- Direction d'autres agents par rapport à l'agent i est souhaitée vecteur vitesse (vi).
- état de groupe.

Si une personne apparaît dans le rectangle d'influence des modifications dans le sens du mouvement et dans la trajectoire pour éviter la collision.

L'angle entre les vecteurs de vitesse deux agents détermine si leurs mouvements sont confluentes ou opposition. Cet angle est également utilisé pour simuler humaine la prise de décisions sur la façon de réagir à une collision imminente. Par exemple, si nous marchons sur le côté gauche d'un couloir, et une autre personne se dirige vers nous, sur notre droite, aucun de nous ne changer de direction, mais si nous sommes à la fois marche au milieu du couloir, la majorité des gens ont tendance à se déplacer vers leur droite. Par conséquent, lorsque les vecteurs de vitesse sont presque colinéaires, les forces tangentielles pointera vers la droite.

Supposons qu'un agent i détecte j, mandataire l que possible obstacles (Figure 4). Nous calculons le vecteur de distance vers i agent pour chacun d'entre eux (dji et dli). Agent j est plus loin que l, mais comme il se déplace contre i agent, l'algorithme de perception établit cet obstacle comme ayant priorité plus élevée. Nous sélectionnons un agent doit être évitée si elle tombe dans le rectangle d'influence, à moins que l'agent est la marche dans la direction opposée et avec la distance plus petite.

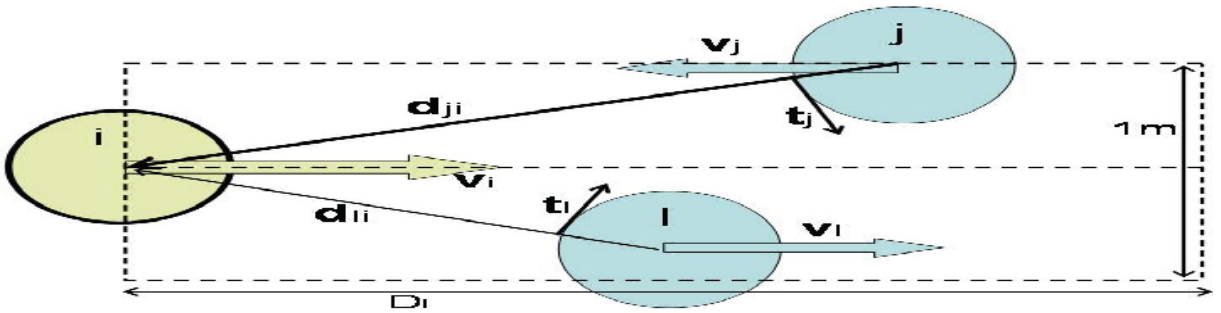


Figure .47 prévention de collision

❖ Evitement d'obstacles statiques

Le problème d'évitement des obstacles statiques est résolu par la confection d'une grille. Dans laquelle, il est déterminé l'espace navigable et l'espace non navigable par la présence d'obstacle statique (mur, etc.).

❖ Evitement d'obstacles dynamiques

Nous appelons obstacle dynamique tout objet mobile dans l'environnement survient en cours de route (piéton, voiture .etc.).

Pour éviter ce type d'obstacle nous essayons de réaliser une technique qui pourrait éviter ces obstacles. Cette technique est basée sur le principe de champ potentiel. Elle dote les piétons de l'environnement par des forces de répulsion.

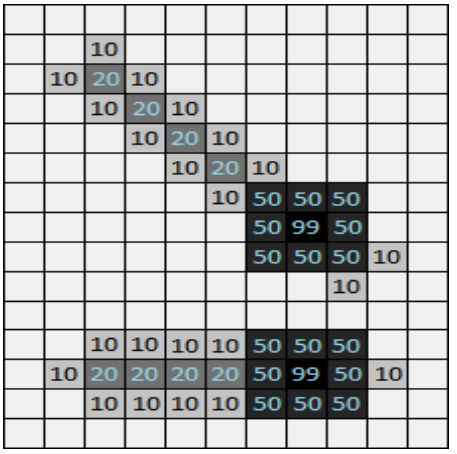


Figure 48 : le champ potentiel de deux piétons

Pour permettre à tout piéton de se déplacer plus précisément, nous augmentons le coût « F » du nœud de son appui et ceux de ses voisins ainsi que les nœuds constituant son chemin (voir figure 49).

Les coûts des nœuds tendent à diminuer selon leur position para port au piéton.

Au cour de déplacement du piéton, chaque piéton lorsqu'il détecte un objet dynamique, il teste le coût de cet objet et recalcule son chemin dans la partie contenant le champ potentiel par le changement de sa trajectoire à droite ou à gauche.

Par exemple :

						10			
						10	50	50	50
				10	10	10	50	99	50
						10	50	50	50
					50	50	50		
					50	99	50		
					50	50	50		

Figure 49 –a l'évitement de collision

					10		10		
					10	50	50	50	
				20	20	50	99	50	
				50	50	100	50	50	
				50	99	50			
				50	50	50			

Figure 49 –b le F elvé

Lors de déplacement il teste si personne le cout de zone doit eleve donc calculer nouveau chemin .

Algorithme d'évitement de collision

Début

- Changer les couts du chaque nœud appartiens de chemin
- Régénérer la liste des nœuds
- Recalculer A*
- Générer le nouveau chemin
- Appliquer les règles psychologie
- Rétablir le chemin final

Fin

L'organigramme ci-dessous illustre cette technique.

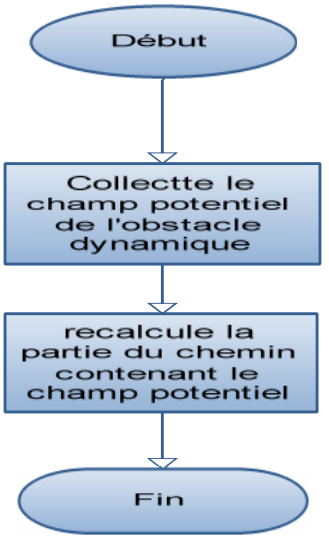


Figure 50: technique d'évitement d'un obstacle dynamique

❖ Evitement d'obstacles imprévisibles

Nous appelons obstacle imprévisible tout événement qui pourrait survenir en cours de route (écroulement d'un mur).

Le piéton se déplaçant dans l'environnement virtuel trace un chemin pour atteindre une cible si au cas où il surgit un obstacle imprévu, il l'évite et il le prendra en compte dans son prochain déplacement.

La détection de ce type des obstacles conduit à générer par des tests sur le chemin de déplacement de piéton. Ce test se fait par la diffusion des cinq « 5 » nœuds sur les cinq « 5 » points du chemin (le nombre des nœuds de test modifiable selon les besoins), ces nœuds capables d'explorer l'espace par leurs positions en détectant l'existence des obstacles éventuels imprévisibles ou non.

Si un nœud détecte un objet éventuel, il doit tester les nœuds voisins, si ces derniers détectent à leur tour un obstacle et qu'ils sont visibles par le piéton, l'ensemble des nœuds seront en face d'obstacle inaccessible.

Ce type de nœud aura une structure qu'on a présentée dans la figure 10.

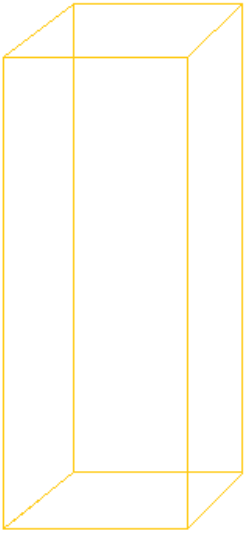


Figure 51 : représente la structure du nœud de la détection les obstacles imprévisibles

5. Comment atteindre la cible ?

On doit réaliser d'une façon dynamique que chaque piéton se déplace pour atteindre la cible. En cas de changement de la position de cette cible les piétons la suivent.

Le principe de cette technique est que la cible est dotée d'une liste des piétons qui la poursuivent. En cas de changement de la position de la cible, cette dernière envoi à chaque piéton répertorié un message lui annonçant la nouvelle position.

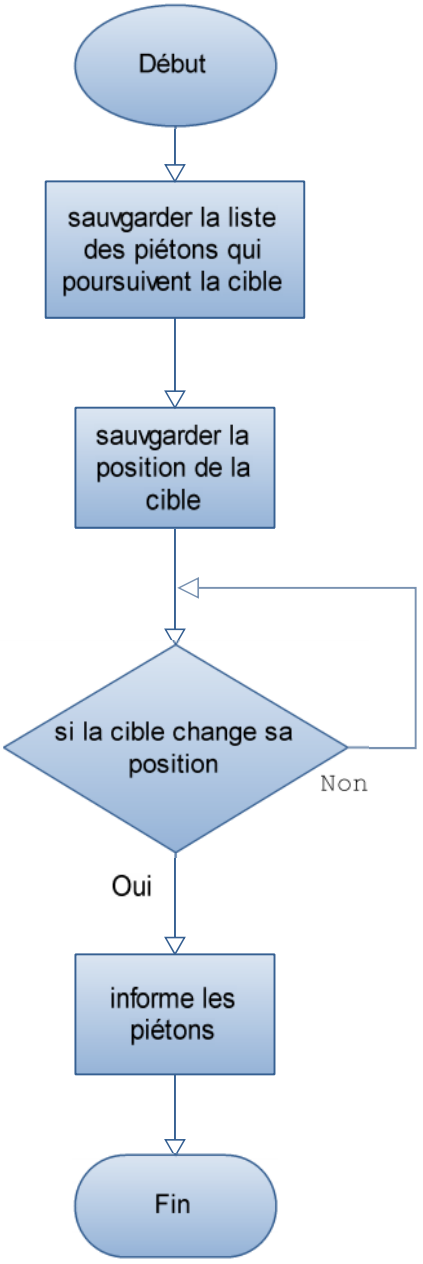


Figure. 52 organigrammes représentent les différentes étapes pour atteindre la cible

La figure ci-après résume tout ce qu'on a dit sur la navigation.

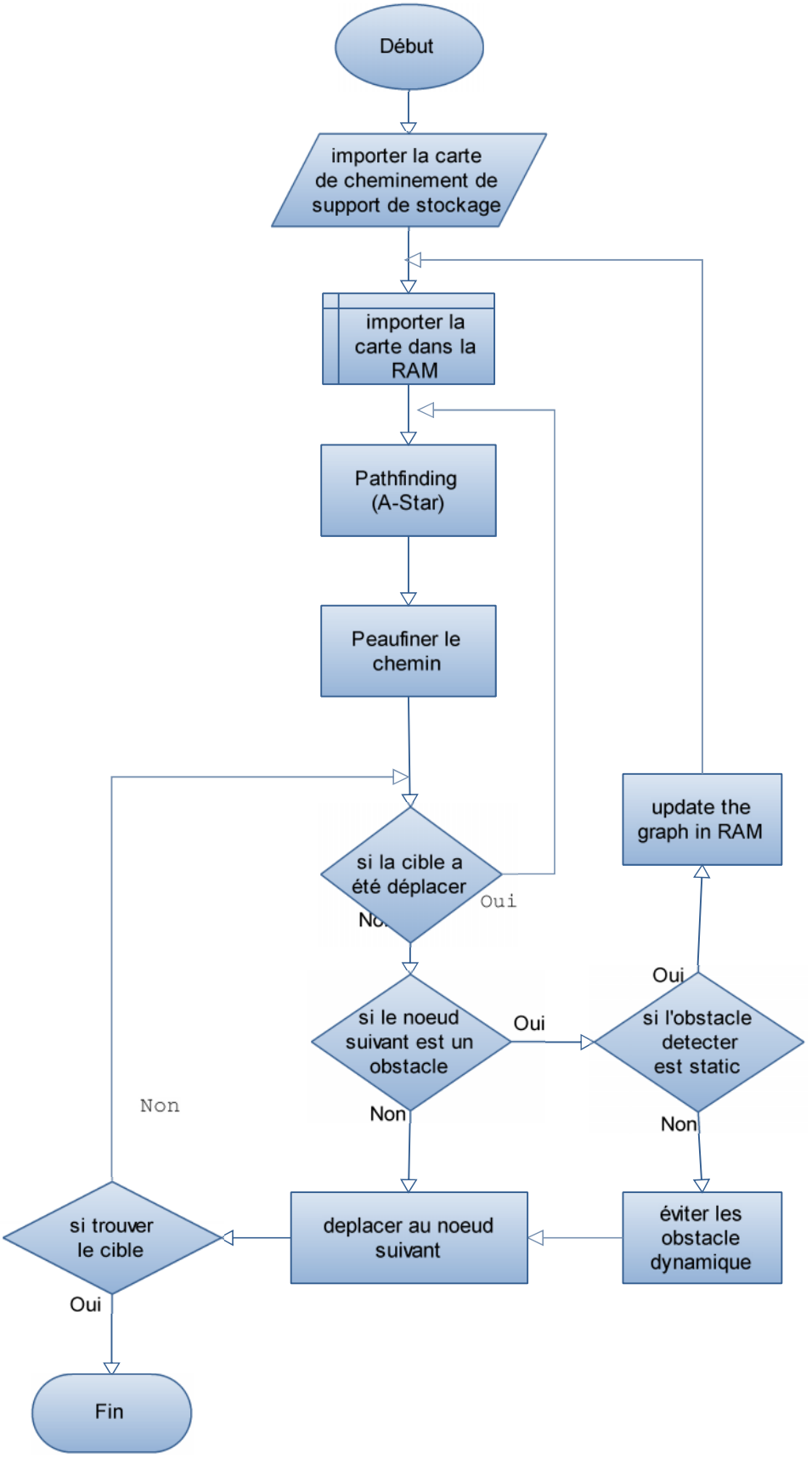


Figure. 53: organigramme représente les différentes étapes de la navigation

6.Conclusion

La navigation forme le dernier comportement réactif de l'humain virtuel. Elle permet de tenir compte des informations ponctuelles influençant son déplacement. L'approche que nous proposons permet d'appliquer les principes: prédiction, robustesse, minimisation des ajustements, et réactivité.

Son avantage majeur est de permettre la fusion efficace d'informations de nature différente, que ce soit des obstacles gênant la navigation des humains, ou des attracteurs incarnant les objectifs secondaires de l'humanoïde. Son fonctionnement est indépendant des autres comportements, si ce n'est la prise en charge de leurs directives. D'autre part, ce comportement est individualisé par l'introduction de facteurs qualitatifs pour la quantification des éléments de l'environnement.

Enfin, la méthode de calcul permet de simplifier l'espace de recherche des solutions .Cela permet d'atteindre un certain niveau de performances nécessaire à la simulation d'une multitude d'individus.

Chapitre V

Implémentation et résultats

1.Introduction

Dans ce chapitre nous présentons la dernière étape de la réalisation d'un système de navigation d'humains virtuels, nous présentons d'abord l'architecture du système implémenté, nous définissons chaque partie et nous finissons ce chapitre par la présentation des résultats obtenus selon les cas étudiés.

2.Implémentation du système

Les d'individus se déplacent dans un environnement virtuel qui se compose d'un ensemble d'obstacles, des lieux fermés et des intersections entre deux voies. Nous traitons ces individus de plusieurs façons ; le premier cas c'est d'étudier le comportement de chaque individus séparément aux autres, il se dirige vers un but prédéterminé en se déplaçant librement dans l'environnement 3D simulé, tout en évitant les obstacles rencontrés dans le chemin et ainsi d'éviter la collision avec les autres individus, la direction vers le but est réalisé par la recherche du plus court chemin qui amène vers ce but.

2.1. Langage de programmation et bibliothèque graphique

La réalisation de ces déférentes étapes a besoin d'un outil développement et des langages de programmation. Pour cela nous choisissons comme outil de développement Unity 3D et comme langages de programmations principale le C# et comme langage auxiliaire XML. Avant d'entamer cette partie une brève présentation sur :

1) l'outil « Unity 3D »

2) langages de programmation

3) une présentation de résultat de la mise en œuvre.

2.1.1 Aperçu sur l'outil de développement de l'application

2.1.1.1 Unity 3D

Unity 3D est un outil de développement 3d et il est aussi le moteur de jeux vidéo. Il permet de produire simplement (le jeu), grâce à une interface d'intégration d'objets et de scripts très intuitive, des jeux de standard professionnel pour Mac, Pc et le web. Une version du moteur existant également pour la Wii, l'iPhone et l'Android, il offre un accès aux consoles et aux mobiles une fois que vous en maîtrisez les bases. [Gw10]

L'éditeur d'Unity intègre des composants préconfigurés évitant le développement de code assez fastidieux. Les principaux composants sont : (Unity3D)

- ✓ un moteur physique basé sur PhysX de la société Nvidia.
- ✓ un système de collision.
- ✓ un système de ragdoll.
- ✓ un moteur d'ombre (version pro).
- ✓ une interface en drag and drop.

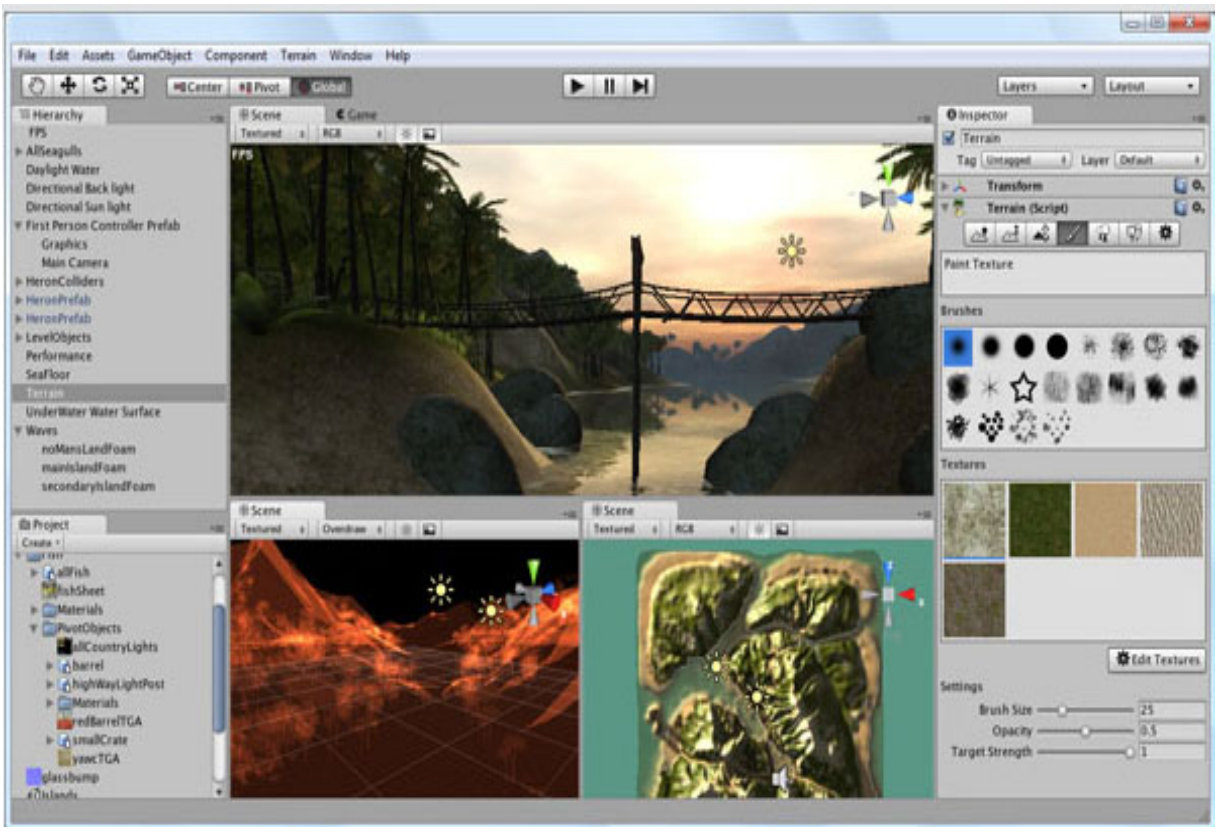


Figure 54 : capture d'image de L'éditeur d'Unity 3D

Unity3D offre un choix entre 3 langues différentes de programmation:

- javascript (ou Uniscript)
- C#
- Boo (dérivé du Python).

2.2. Description des langages de développement

Dans cette partie on consacre pour la description les deux langages de programmation :

2.2.1. Langage C#

Le langage C# (prononcer C Sharp) a été conçu par Microsoft dans l'optique d'accompagner sa plateforme .Net. Le Framework.Net est un ensemble de bibliothèques de classes associé à un environnement d'exécution à l'instar du Java Devlopment Kit (JDK) de Sun Microsystems.

D'autres outils d'exécution et de développement existent notamment dans le monde libre. Mono parrainé par Novell et DotGNU Portable.Net constituent une alternative au Framework.Net sur des systèmes d'exploitation Linux, NetBSD, FreeBSD, Solaris, ou encore MacOS X, et y compris d'ailleurs sur ceux de Microsoft.

Le langage de programmation C# s'inspire assez largement de la syntaxe générale C/C++ comme les expressions et les instructions, et de certaines caractéristiques du langage Java comme la gestion automatique de la mémoire (Garbage Collector). CSharp est donc un langage orienté objet bénéficiant d'innovations censées le rendre plus sûr et convivial.

La compilation du code source C#, génère un langage intermediaire (MSIL : MicroSoft Intermediate Language) tout comme Java produit son fameux byte-code. Pour l'exécution de ce langage intermédiaire un environnement d'exécution multi-langages est utilisé. Cet environnement se dénomme : CLR (Common Language Runtime). Il intègre un moteur d'exécution, un défragmenteur de mémoire, un système de sécurité, un compilateur Just-In-Time (JIT : en temps utile) et une abondante architecture de classes. (Lel12)

Le choix de ce langage par nous est pour deux raisons : Le C# est un langage de programmation qui est à la fois simple et robuste.

2.2.2. Langage XML

Le XML (eXtended Markup Language) est un langage de balisage extensible standardisé par le World Wide Web Consortium (W3C) qui s'occupe également de la standardisation du langage HTML et des feuilles de style entre autres.

Le HTML permet de mettre en forme un document contenant diverses données formatées en titres, en paragraphes, en listes, en tableaux, etc. Ainsi, le langage HTML possède des

balises destinées essentiellement à la présentation des données que le développeur lui fournit et en aucun cas, il ne tente de les interpréter.

Par contre, les balises du XML définissent plutôt la sémantique (le sens) des données. C'est-à-dire, que le balisage créé par le développeur donnera une signification précise des données fournies.

Par exemple, une liste de noms et de prénoms d'employés dans une entreprise quelconque sera balisé comme suit par du HTML :

```
<ul>
<li>Jean Bernard</li>
<li>Jean-Yves Dupré<li>
                                        ...
</ul>
```

Alors que dans le XML, le balisage adoptera une autre forme plus adaptée aux données :

```
<?xml version="1.0" ?>
- <ENTREPRISE>
  - <EMPLOYE SECU_SOC="1.80.12.75.120.058/51">
      <NOM>Bernard</NOM>
      <PRENOM>Jean</PRENOM>
    </EMPLOYE>
  - <EMPLOYE SECU_SOC="1.51.02.38.032.181/18">
      <NOM>Jean-Yves</NOM>
      <PRENOM>Jean-Yves</PRENOM>
    </EMPLOYE>
  </ENTREPRISE>
```

Comme nous pouvons le constater, le XML est une structure arborescente dont les nœuds de l'arbre contiennent des données. (note).

3.Présentation de système

Dans cette section on présentera les différentes scriptes utilisés dans ce système et les éléments préfabriqués (Prefabs) ainsi que les étapes de mise en œuvre de système.

3.1. Présentation de différents scriptes utilisés

On a codé les différents scriptes en précisant à chacun un rôle qui dans leur ensemble assurera le bon fonctionnement du système (voir la figure 55).

Figure 55: représente les déférents scriptes dans le projet

Pour mieux illustrer cette section, on essaie de présenter globalement tous les fonctionnements des scriptes:

- **Spreader :** diffuse les nœuds selon la taille de la scène et génére le fichier XML.

  Ce scripte contient les configurations suivantes:

  - **Node prefab** : contient le nœud de test qu'on doit le diffuser.
  - **Scence length (X, Y, Z) :** la taille de l'espace qu'on veut la représenter.
  - **Step offset :** la distance maximale entre deux nœuds superposés, il peut les relier.
  - **Draw the Drid :** permet de dessinerla grille.
  - **Draw Gizmos :** affiche le champ de détection des nœuds.
  - **Render Nodes :** affiche les nœuds.

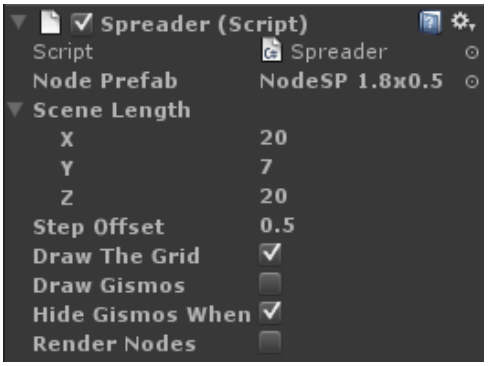


Figure 56: représente le contenu de scripte « spreadre »

- **Tester :** gère les fonctionnements du nœud (tester l'emplacement navigable et non navigable), tout en utilisant les deux scriptes « **testerChild01 »** et **« testerChild02 »**. Ce scripte contient ensemble de configuration (voire la figure 57) :
  - **Test Offset** : donne la distance de déplacement en hauteur lors de la détection d'une collision du nœud au niveau de sa partie 3.
  - **Agent space** : représente l'écartement de Ray-Caste dans la partie 4 du nœud.
  - **Down Test :** donnela longueur des Ray-Castes de la partie 4 du nœud.
  - **Draw gizmos :** affiche le champ de détection du nœud.

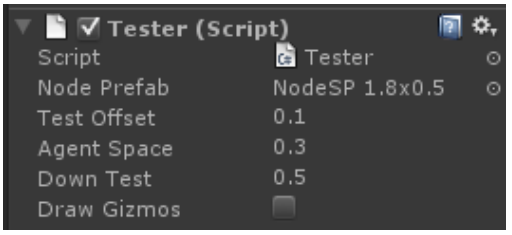


Figure 57 : représente le contenu de scripte « Tester »

- **Node** : représente la structure du nœud dans le graphe.
- **Player** : on peut le simuler au cerveau du piéton, il permet d'appeler le scripte « A_STAR » pour avoir le tracé de son chemin. Ce scripte offre des options (configurations) (voir la figure 58) :
- **Target Game testerChild01 :** aide le scripte « Tester » par la génération de la détection de collision dans la « partie 1 » du nœud (voire le chapitre 3).
- **testerChild02 :** le même fonctionnement de scripte « **testerChild01 »** mais celui si s'occupe de la détection de collision du nœud dans ses parties 2 et 3.

- **Flooder :** permet de nous donner l'image de la scène dans laquelle une constitution des groupes de nœuds connectés entre eux (Flool fill).
- **loadTheGraph** : appelle la carte de cheminement du fichier XML et l'introduit dans la RAM (Random Access Memory).

  On peut modifier par ce dernier scripte l'option d'affichage de grille.

Figure 58 : représente le contenu du scripte « LoadTheGraph »

- **Object** : spécifie la cible.
- **Team Work** : choisit l'option d'acquisition des modifications dans l'environnement et avertit le groupe.
- **Team Name** : spécifie l'appartenance du piéton au groupe.
- **Draw Path** : affiche le chemin du piéton.

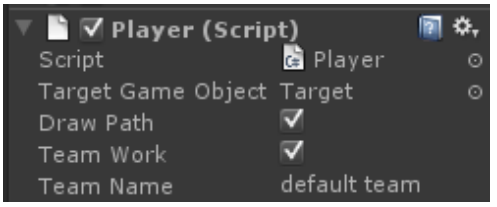


Figure 59 : représente le contenu du scripte « Player »

- **A_STAR** : contient le code de l'algorithme A* permettant de calculer le chemin optimum et elle contient aussi les options suivantes (voir la figure 60):
  - **Sorted Open List** : dresse la « liste ouverte » de l'algorithme et la trie par coût « F » décroissant (permet l'optimisation de calcul de A*).
  - **Cost Penalty** : permet d'obliger le piéton de favoriser son déplacement en avant et minimiser les zigzags en cour de son chemin.
  - **Heuristic Function :** le choix de calcul A* par les différentes heuristiques représentées dans la figure 60 .

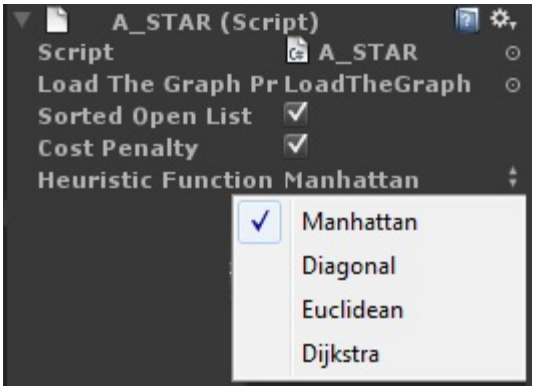


Figure 60 : représente le contenu du scripte « A_STAR »

- **AIController :** contrôle les mouvements du piéton virtuel au sein de l'environnement : le déplacement dans le tracé de son chemin et l'état de mouvement (marcher ou courir) et sa vitesse (voir la figure 61).

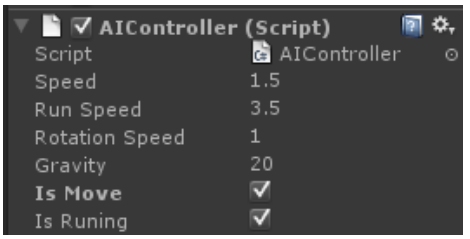


Figure 61: représente le contenu du scripte «AIController»

- **UpdateStatic :** elle gère la détection des obstacles imprévisibles pour que le piéton les évitera et les prendra en compte dans son prochain déplacement (mise à jour de la carte de cheminement), tout en utilisant le scripte « **TesterStatic »**.

  Il existe plusieurs options permettant l'initialisation le script (voir la figure 56) :

  - **UpdateForward** : le nombre des nœuds diffusés dans le parcours du piéton.
  - **Testing Distance** : la distance de visibilité du nœud testé par le piéton.
  - **Draw Gizmos** : affiche le champ de détection du nœud.

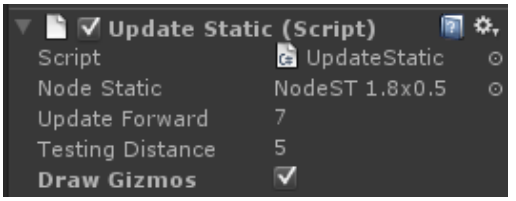


Figure 62 :représente le contenu de scripte « Update Static »

- **TesterStatic :** aidele scripte « **UpdateStatic »** pour gérer le fonctionnement du nœud (nœud qui teste les objets imprévisibles).

- **TargetObserver :** la cible est dotée d'une liste des piétons qui la poursuivent. En cas de changement de sa position, cette dernière envoie à chaque piéton répertorié un message lui annonçant la nouvelle position.

Figure 63 : représente le contenu de scripte « TargetObserver »

3.2. Présentation de différents préfabriqués « Prefabs »

Nous avons utilisé les « prefabs » ils nous permettent de stocker les objets avec des leurs composants et leur configuration (voir la figure 60).

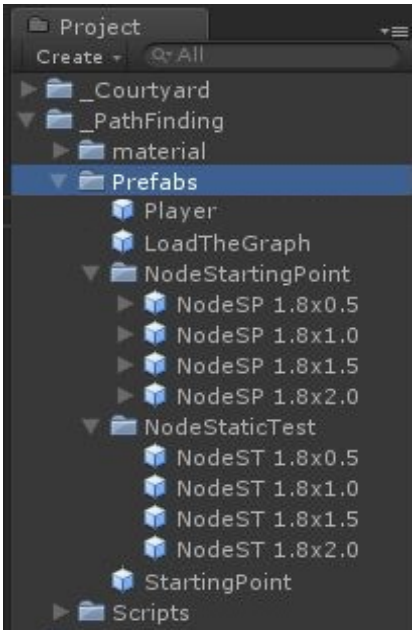


Figure 64 : représente les déférents préfabriqués utilisés dans le projet

Avant d'entamer l'explication des préfabriqués, nous devons tout d'abord définir deux composantes communes dans tous les préfabriqués : « Transform » et « Box Collider ».

Transform : contient toutes les transformations de l'objet telle que : sa position, sa rotation et son volume (voir la figure 65).

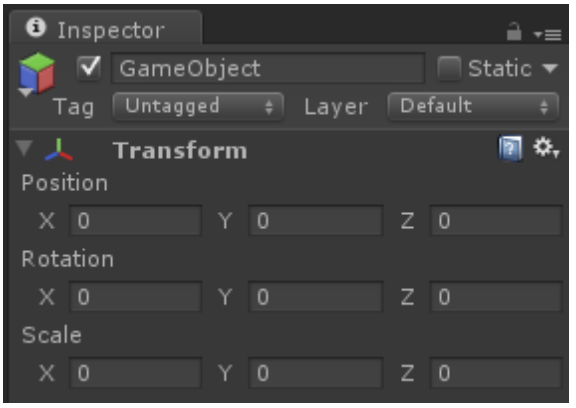


Figure 65 : représente le contenu du composant "Transform"

Box collider : représente le champ de collision de l'objet et permet aussi de modifier leur transformation par :

Center : la position de box collider par à port à leur objet

Size : la taille de champ de collision (voir la figure 66).

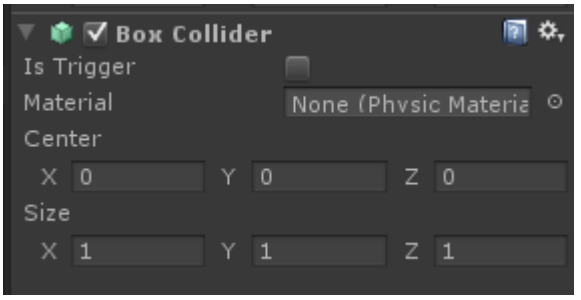


Figure 66 : représente le contenu du composant "Box Collider"

Chacune de ces préfabriqués a une caractéristique spécifique nous les expliquons ci-dessous:

- **StartingPoint** : contient le scripte « spreader » avec leurs configurations.
- **Node SP :** est un nœud de test de l'espace navigable et non navigable, il existe trois différents nœuds, de taille distincte de leurs « box collider » en fonction de l'espace de détection. Ces nœuds auront des tailles réduites (voire la structure du nœud dans le

chapitre conception). Le « Node SP » contient aussi le scripte « **Tester »** avec leurs configurations.

- **Node ST :** est un nœud de test de l'espace contenant des obstacles imprévisibles, le même principe que le nœud (**Node SP**). Il contient le scripte « **TesterStatic ».**
- **loadTheGraph** : contient le scripte «loadTheGraph » fait appel à la carte de cheminement du fichier XML.
- **Player :** représente le piéton qui navigue dans notre environnement et elle contient les déférents scriptes générant le déplacement : Player, A_STAR, AIConctroller, Uapdate Static.

3.3. La mise en œuvre de la carte de cheminement

La schématisation de la carte de cheminement a été déjà expliquée dans le chapitre conception, dans cette section nous allons expliquer sa mise en œuvre, par la présentation des résultats de ses différentes étapes.

3.3.1. Présentation et fonctionnement du nœud réalisé

Après avoir donné la structure du nœud dans le chapitre conception, nous présentons ci-dessous sa forme finale.

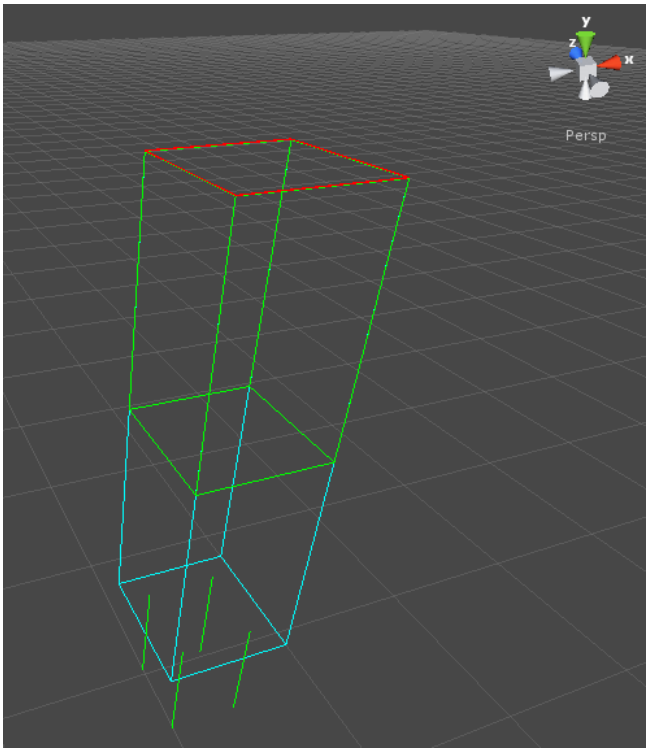


Figure 67 : représente la forme du nœud réalisé

Pour illustrer les fonctionnements du nœud nous présentons les différentes tâches d'un nœud par une des séquences photos montrant deux « 2 » cas de nœud possibles ,l'un au bord et l'autre au milieu le 1er ne pouvant évoluer pacque ses points d'appuis étaient hors de terrain ce qu'ils lui ont provoqué sa chute et la second a réussi son évolution ainsi il détecte un autre espace navigable en hauteur l'incitant à créer un autre nœud en hauteur (voire la figure 68).

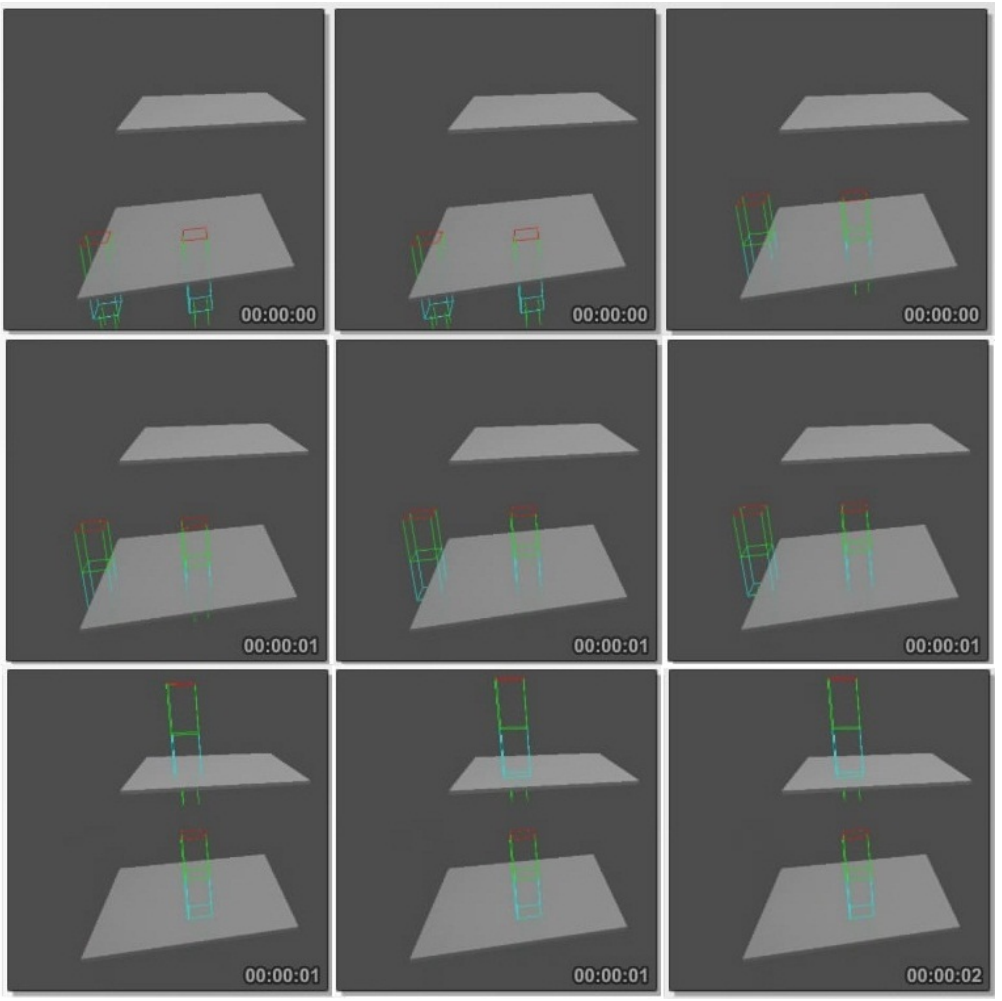


Figure 68 : présente les différentes tâches d'un nœud

3.3.2. La diffusion et la connexion des nœuds

Nous illustrons par les deux exemples ci- dessous les deux notions : la diffusion et la connections des nœuds sur une scène statique.

Exemple 1 : dans cet exemple on a incité la diffusion des nœuds dans une scène avec une petite plateforme statique. Les séquences figurant (69, 70, 71) le développement continue de la cartes de cheminement.

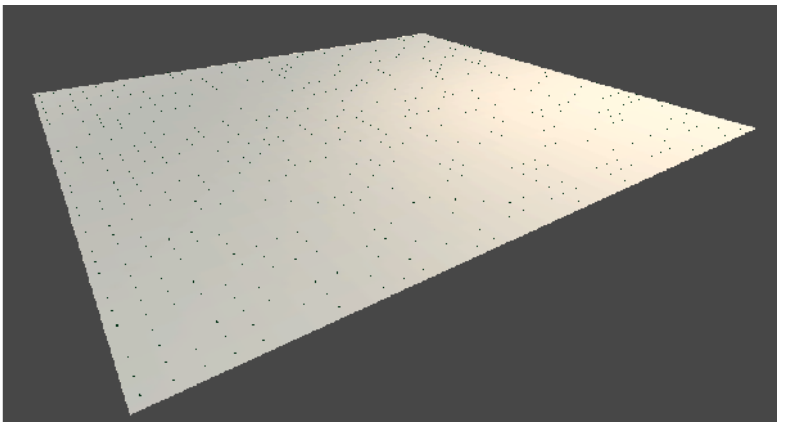


Figure 69 : la diffusion des nouds dans une plateforme statique

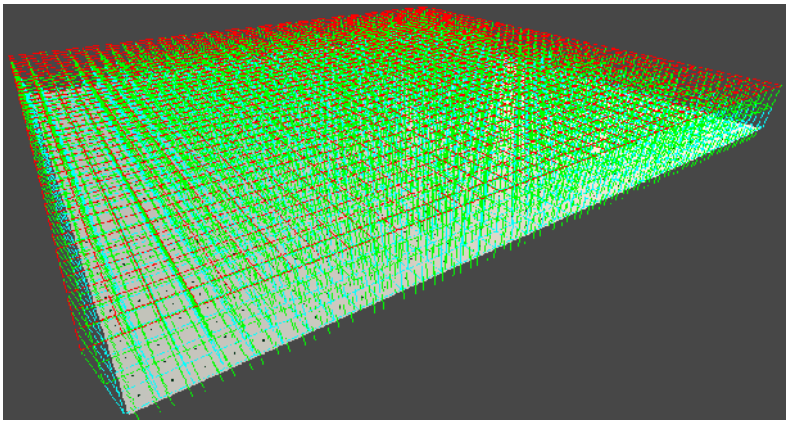


Figure 70 : la diffusion des nœuds avec l'affichage de champs de détection

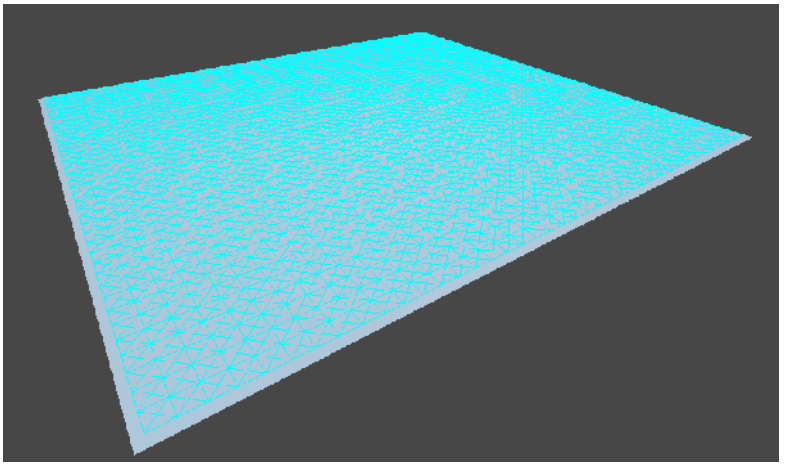


Figure 71 : représente la phase finale de la carte de cheminement.

Exemple 2 : on illustre cet exemple par une séquences photos montrons l'application de l'évolution des nœuds sur une zone déterminée de la scène, les séquences ci-dessous illustre les développements par étage des nœuds.

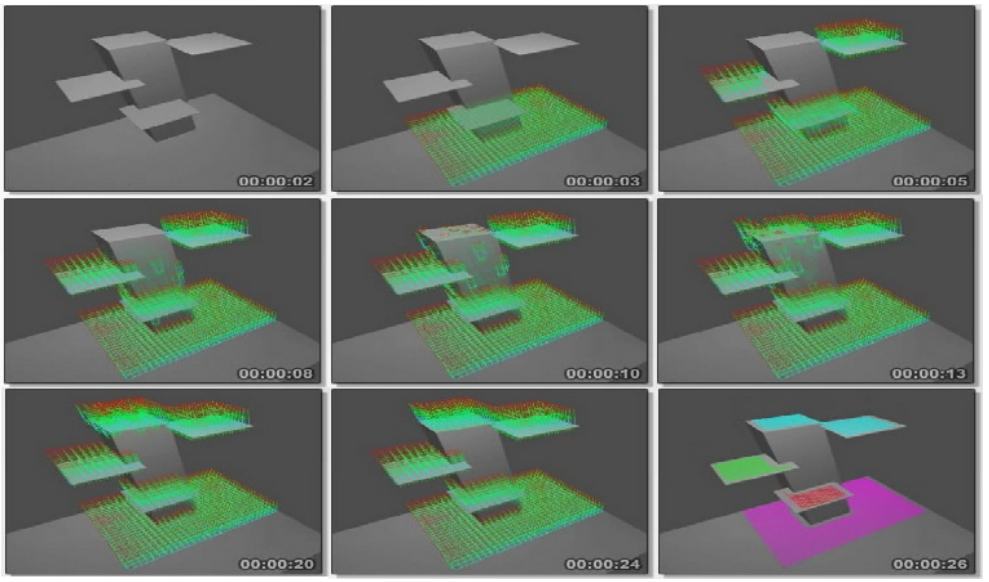


Figure 72 : la diffusion des nœuds dans une scène complexe

3.3.3. L'inondation se remplit « Flood fill »

L'utilisation de la technique « Flood fill », nous a permis de vous présenter l'image de la scène dans laquelle se constituent des groupes de nœuds connectés entre eux.

Par la même technique, on obtiendra à la fin de l'opération une carte de cheminement partagée en groupe de nœuds reliés qu'on pourrait les considérer comme des lacs (lacs des nœuds).

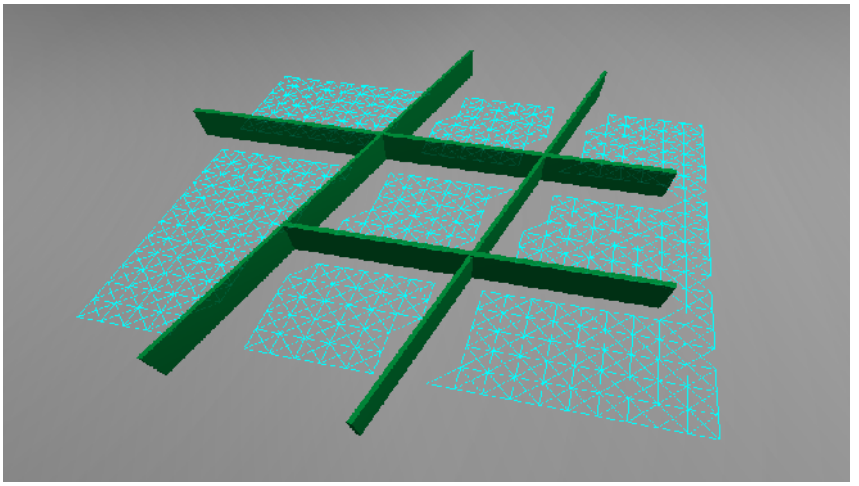


Figure 73 : Une carte de cheminement normale

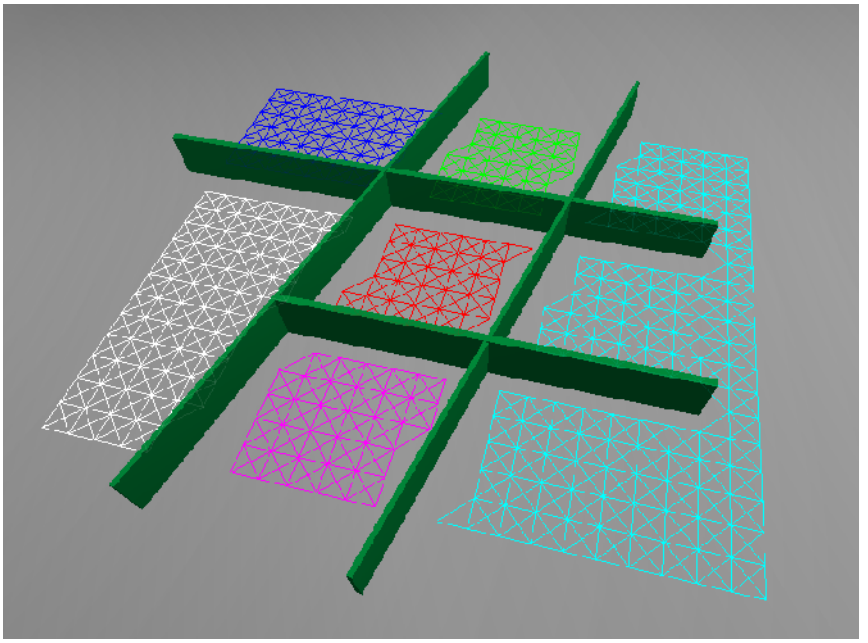


Figure 74 : Une carte de cheminement avec la technique "Flood fiil"

3.3.4. Le contenu du fichier XML

Les informations résultantes de la carte de cheminement seront sauvegardées sur un support physique sous une forme spécifique, notre choix a été fixé pour la forme d'un fichier XML

Ce choix se justifie pour le stockage d'information d'une manière structurée facilitant l'exploitation des données.

La forme particulière que nous avons donnée au fichier XML sera composée d'ensembles de balises. Les balises que nous allons symboliser par (<n>…</n>) elles

représentent les informations des nœuds. Chaque balise d'un nœud contient les informations sur son emplacement, sa position et ses voisins navigables.

Nous avons retenu une partie de notre fichier XML représentant la structure de trois nœuds que nous expliquons en détails, la figure 69 montre la structure des nœuds et leurs contenus :

- **N° 01 :** représente un nœud.
- **N° 02 :** représente de nom du nœud.
- **N° 03 :** représente l'emplacement du nœud dans la grille.
- **N° 04 :** représentel'index de groupe permet de spécifier son appartenance au groupe (Food fill).
- **N° 05 :** représente la position du nœud dans la scène.
- **N° 06 :** représenteles nœuds voisins navigables.
- **N° 07 :** représente le nom du nœud voisin.
- **N° 08 :** représente l'emplacement du nœud voisin.
- **N° 09 :** représente la distance entre le nœud et son voisin.

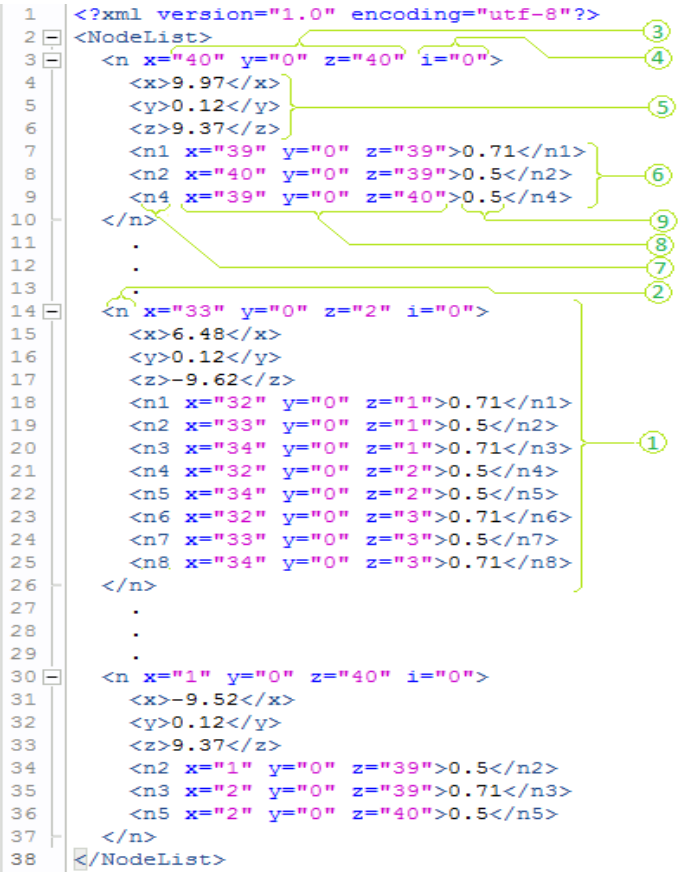


Figure 75. Une capture d'image sur une partie de fichier XML

3.4. La navigation

Après la réalisation de la carte de cheminement on procède au résultat de la **mise en œuvre de la navigation.**

3.4.1. Trouver la cible

On fait appel à la carte de cheminement à partir de script « **loadTheGraph** ». L'exploitation de cette carte sera par l'application de script **A_STAR** (algorithme A*) dans le but de l'obtention de chemin optimum en évitant les obstacles statiques.

Pour illustrer l'application de l'algorithme A* on a retenu une séquence photos montrant le déplacement d'un piéton dans une scène 3D. On a représenté le piéton par un cylindre vert et la cible par une capsule bleue (voir la figure 76).

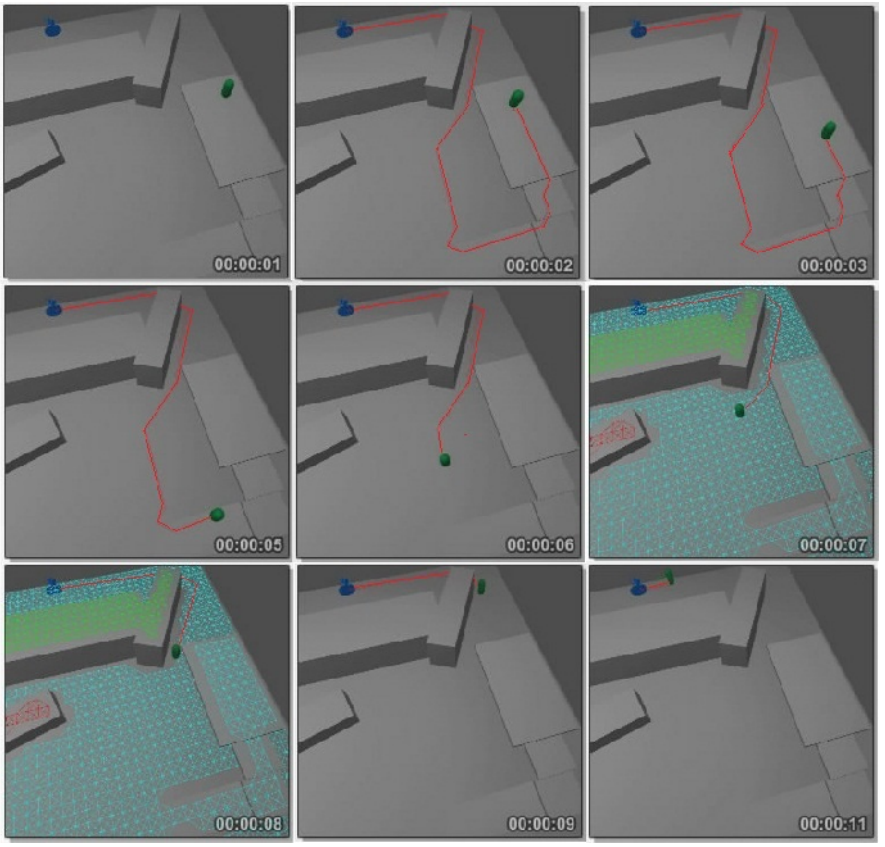


Figure 76 : l'application de A* dans une scène 3D.

3.4.2. Peaufiner le chemin proposé

Un chemin peut parfois être accidenté entre des points successifs, le piéton qui se déplace peut être amené à faire des changements de directions très brusques.

Pour éviter ce cas on essaie de réaliser une technique permettant de minimiser les brisures, de façon que le piéton prenne un chemin en forme d'une courbe au moment de l'évitement d'un objet statique.

3.4.3. Evitement des obstacles

On a essayé d'une façon systématique à éviter les obstacles dynamiques et éventuellement tous les obstacles imprévisibles (une obstruction d'une route ou écroulement d'un mur).

VI. *Evitement des obstacles dynamiques :*

Nous définissons qu'un obstacle dynamique est tout objet mobile dans l'environnement qui peut être rencontré en cours de route (piéton, voiture .etc.).

L'exemple ci-dessous illustre d'une manière schématique cette technique.

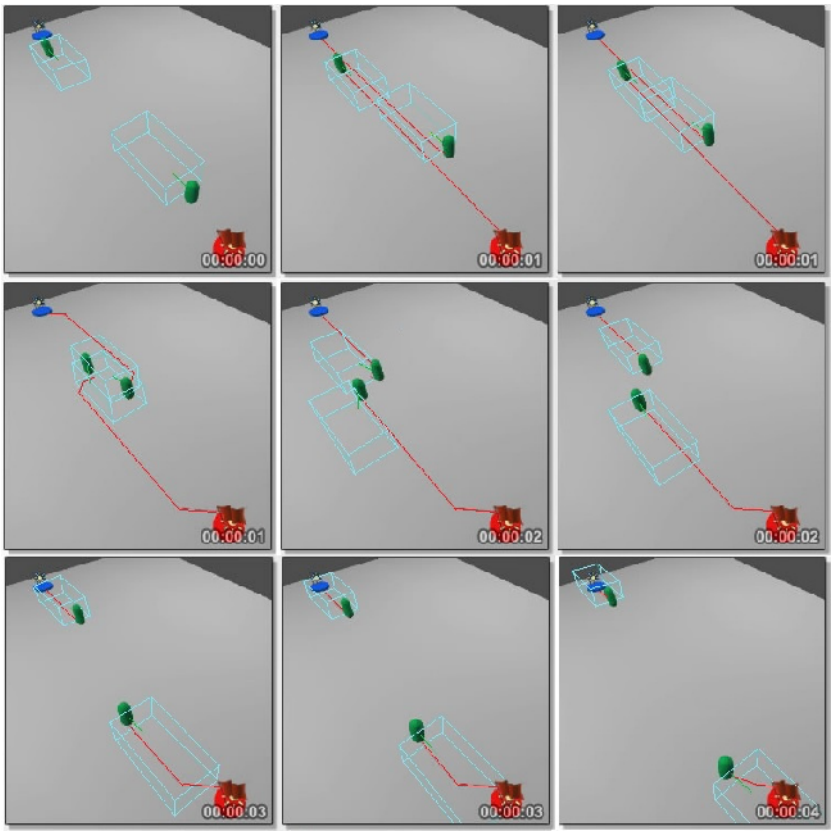


Figure 77. Évitement des obstacles dynamique

Evitement des obstacles imprévisibles et mise à jour de la carte de cheminement :

On va vous donner dans cette partie deux options d'apprentissage de l'environnement par les piétons :

- la première montre que chaque piéton en cas d'obstacle imprévisible tente de l'éviter et le prend en compte dans sa carte de cheminement,

La figure ci-dessous illustre ce cas.

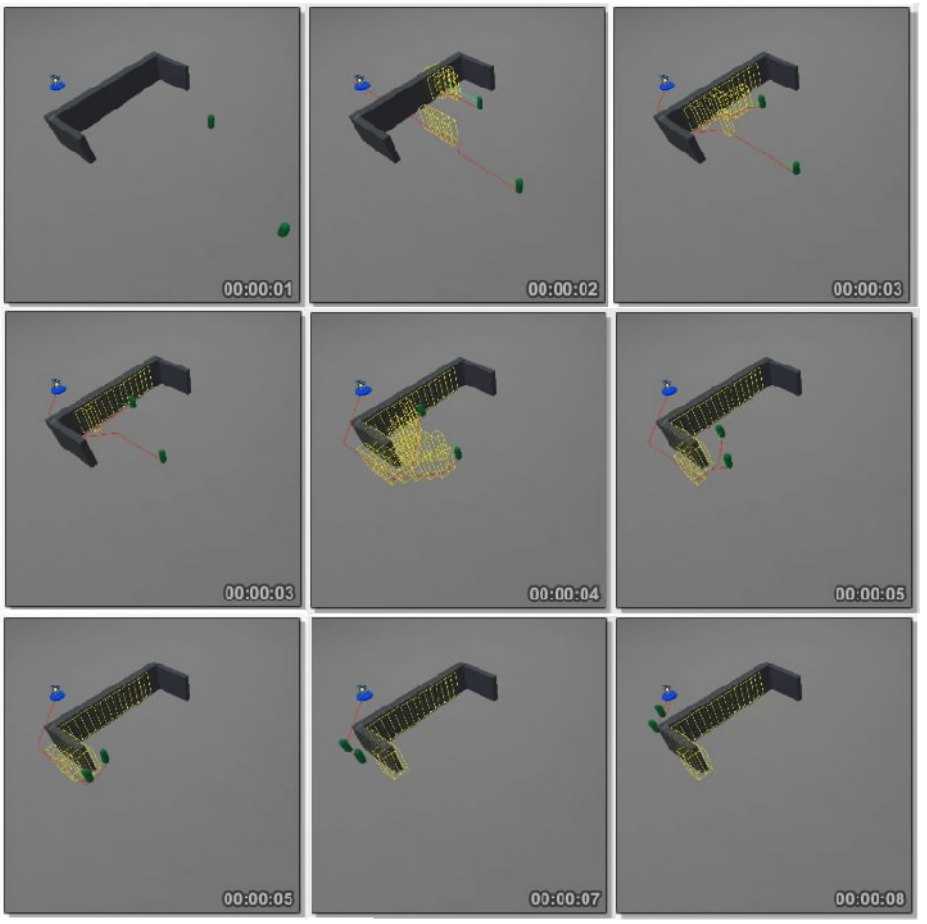


Figure 78 : capture d'image représente l'apprentissage individuel de l'environnement

- et la seconde montre que le 1er piéton ayant déjà pris connaissance du terrain et des obstacles qui y existent, informe les autres piétons de leurs existences pour les considérer dans leurs futurs déplacements.

La figure ci-dessous illustre ce cas.

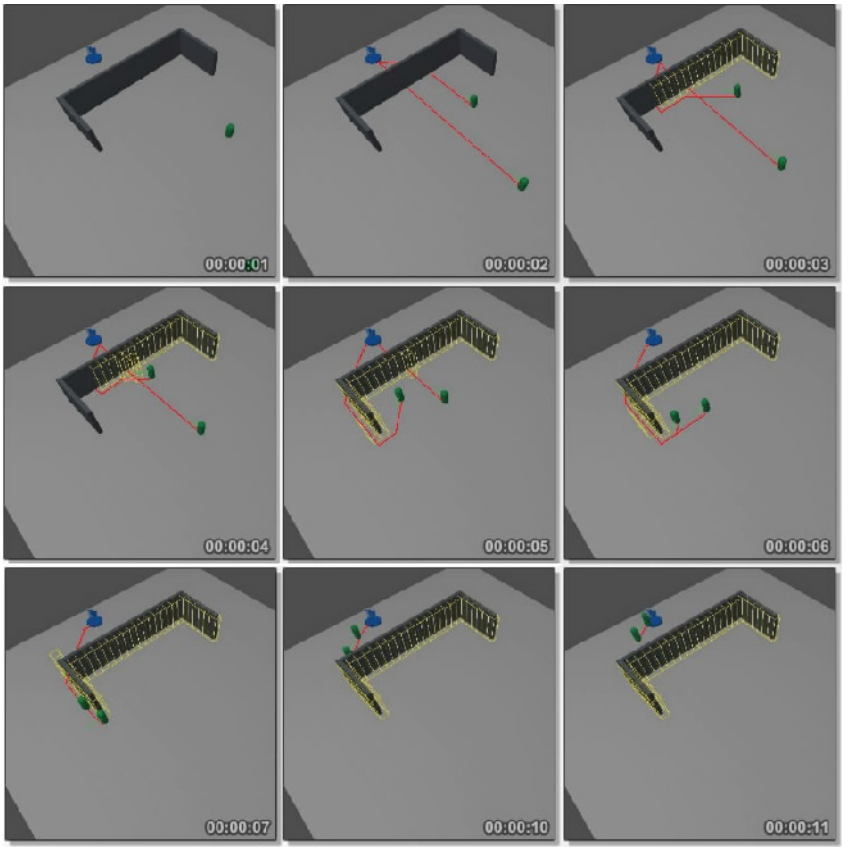


Figure 79 :capture d'image représente l'apprentissage de l'environnement en groupe

Après ces exercices d'apprentissage le piéton (les piétons) mettra à jour de la carte de cheminement (voir la figure ci-dessous).

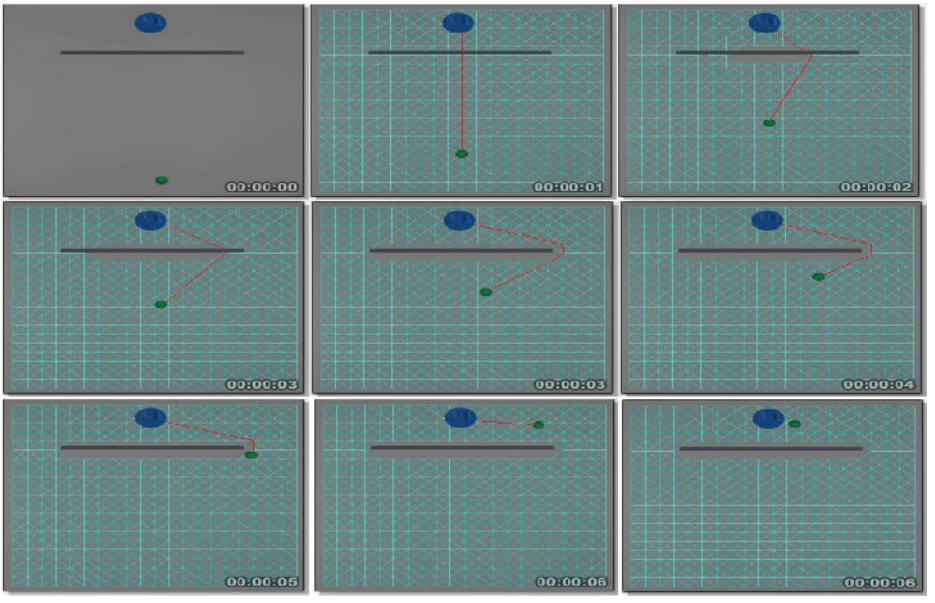


Figure 80: la mise à jour de la carte de cheminement

4. Présentation du résultat final

Dans cette section nous présentons les résultats obtenus de projet final, le principe consiste à :

Mise en place d'une grande scène, dont on l'a modifiée selon les besoins et on a essayé d'appliquer le résultat de notre recherche sur cette scène. (voir les figures ci-dessous).

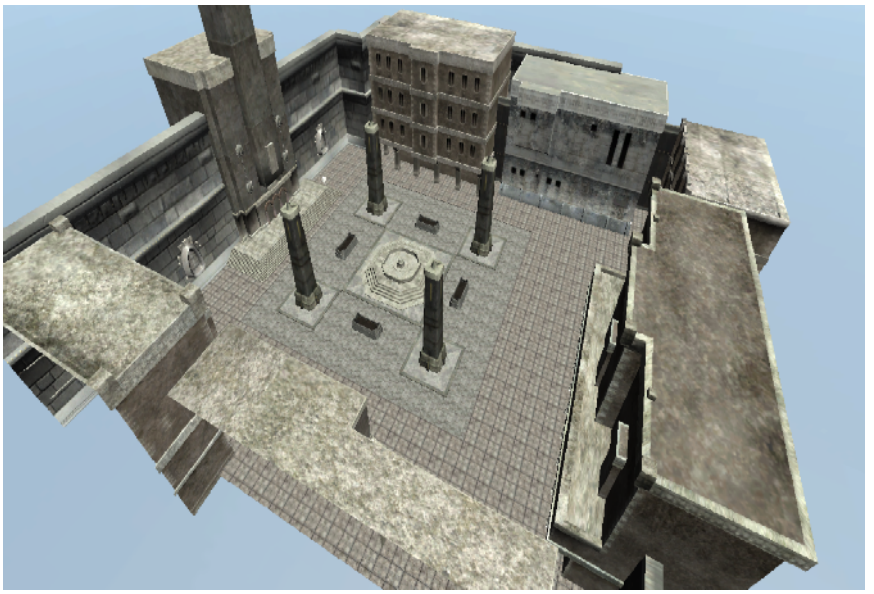


Figure 81 : capture d'image de la scène utilisée

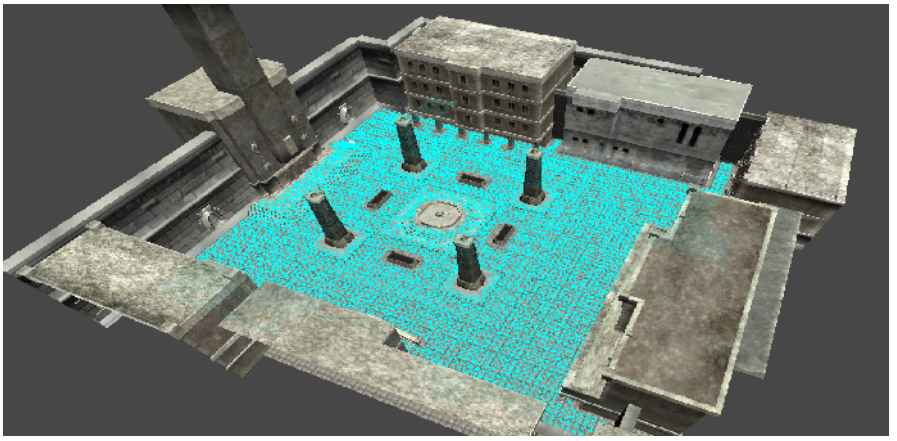


Figure 82 : application de la carte de cheminement à cette scène**.**

La figure 81 représente notre scène avant la construction la carte de cheminement et la figure 82 après la construction la carte de cheminement.

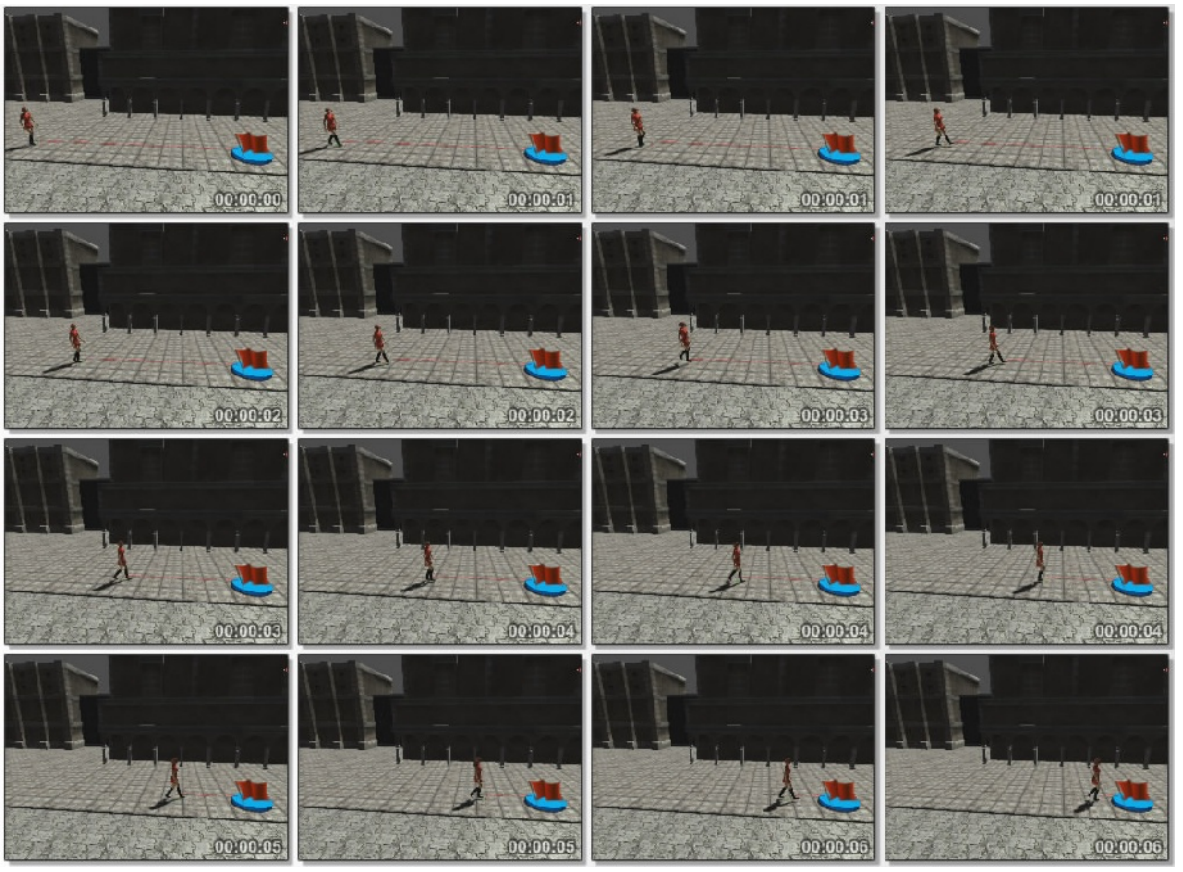


Figure .83 la navigation de l'agent ver le but.

Le premier exemple (figure 83) montre comment l'humanoïde naviguer dans la scène vere la cible par la recherche de chemin optimale.

En cas d'escalier :

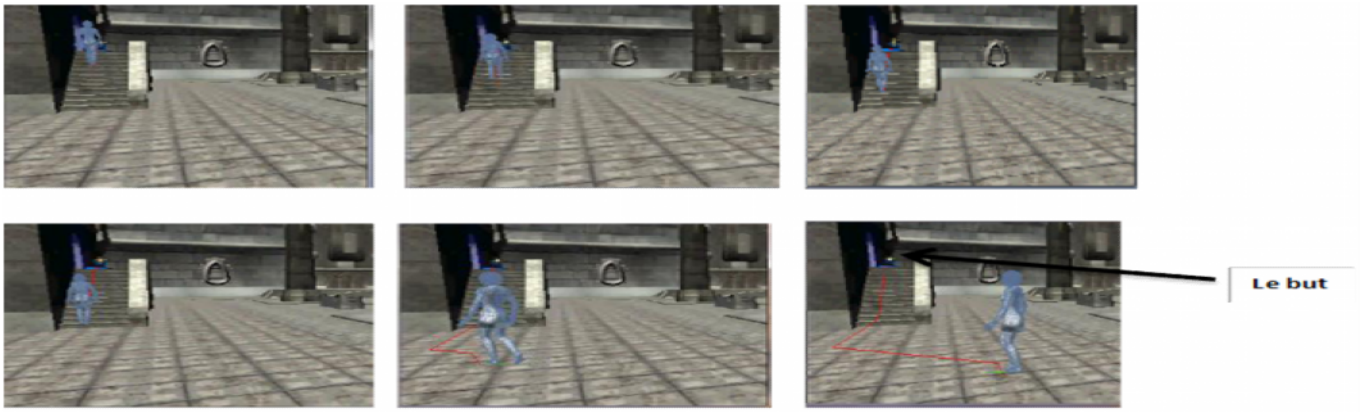


Figure 84 :l'agent monte l'escalier

Le deuxième exemple (figure 84) montre comment l'agent aller vers le but, par la recherche de chemin optimale et monter l'escalier pas par pas de façon réaliste.il traverser un plat puis un environnement encline.

En cas de groupe

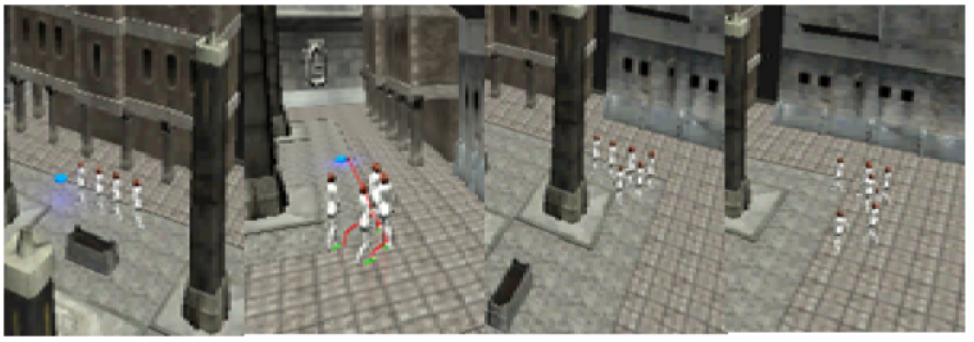


Figure 85 le groupe des agents aller vers le but

Le troisième exemple (Figure 85) montre la navigation de groupe des agents vers le même but et même environnement mais contais plusieurs obstacle le trait rouge montre le chemin optimal pour chaque agent en cas collision chaque agent mise a jour la carte de cheminement et doit faire une planification locale pour éviter cet collision, dans la figure 85 l'agent atteint le but sans collision.

5.Conclusion

Nous avons présenté un modèle de suivi de trajectoire faisant intervenir de nombreux facteurs, le choix d'une stratégie pour franchir les obstacles. Les algorithmes sont suffisamment efficaces pour permettre une exécution en temps réel sans pour autant sacrifier le réalisme de l'animation produite.

Notre modèle a été conçu de manière générique permettant ainsi de décrire de nouveaux comportements suivant les besoins. D'autre part, les comportements ne sont pas spécifiés de manière explicite, ils émergent suivant la spécificité de chaque situation.

Conclusion Générale et Perspectives

Nous avons traité au sein de ce document la problématique de Navigation réactive de l'humain virtuel via la planification locale. Cette problématique est renforcée dans notre cadre d'application par la nécessité de tenir compte des besoins d'interaction des humains.

Le choix de notre thème qui est la recherche d'un chemin optimum dans un monde virtuel et à moindre coût d'effort, nous conduit à l'utilisation de toutes les techniques mises au point par les chercheurs dans ce domaine

Notre approche consiste à l'élaboration de la scène, sa décomposition cellulaire que nous avons appelé couramment la « grille » dans notre recherche, cette décomposition cellulaire nous a permis à la construction d'un graphe sur lequel on a appliqué les dernières techniques de calcul pour arriver au résultat attendu qui est le chemin optimum mais notre humanoïde virtuel, au cour de son déplacement rencontre des obstacles qu'il faut les résoudre.

La résolution de ces problèmes se distingue par la nature de l'obstacle : statique, dynamique ou imprévisible, la technique de base est la carte de cheminement pour les trois cas. Cette carte qu'on peut la comparer à une représentation mentale sur la géométrie de la scène sur laquelle notre humanoïde peut se déplacer en évitant la collision avec les objets statiques. En cas d'un obstacle dynamique, on utilise des techniques adéquates permettant de l'éviter.

Et dans le cas des obstacles imprévisibles des techniques spéciales qu'on a mis au point pour résoudre ce cas. Ces dernières permettent à notre piéton virtuel de détecter l'obstacle et l'éviter et le prendre en compte dans son prochain périple. Le but de l'emploi de ces différentes techniques vise à atteindre l'optimisation du trajet parcouru.

L'application de ce type de cas de la recherche d'un chemin optimum dans la vie réelle peut se réaliser dans les cas des catastrophes : chavirement d'un navire, le cas d'incendie jeux vidéos etc..

Ensuite, nous illustrons l'implémentation de notre système, en étudiant la représentation de notre environnement, le programmation des humains virtuels conduisant la détection et l'évitement des obstacles, en illustrant des situations convenables.

Perspectives

Notre système donne des résultats acceptables, mais des extensions à ce travail nous paraissent intéressantes restent à compléter comme :

- L'ajout d'une couche cognitive dans les comportements humains, l'apprentissage dans son processus de navigation et le raisonnement.
- L'ajout d'une couche de raisonnement pour le comportement de l'humain virtuel permettant de gérer des activités à long terme c-a-d tenir compte d'un plan comportemental plus général a l'inverse de Ces processus ne sont capables de prendre des décisions que de façon relativement isolée.

Bibliographie

[BY95] J.-D. Boisonnat et M. Yvinec. Géométrie algorithmique. Collection Informatique. EDISCIENCE international, 1995.

[Bar02] Christoph Bartneck. Integrating the occ model of emotions in embodied characters. Workshop on Virtual Conversational Characters, 2002.

[BD04] Badawi.M. et Donikian, S. (2004). Autonomous agents interacting with their virtual environment through synoptic objects.

[BF97] Blum, A. L. et Furst, M. L. (1997). Fast planning through planning graph analysis. Artificial Intelligence,

[BMOB03] Adriana Braun, Soraia R. Musse, Luiz P. L. de Oliveira, et Bardo E. J. Bodmann. Modeling individual behaviors in crowd simulation. Dans CASA '03 : Proceedings of the 16th International Conference on Computer Animation and Social Agents (CASA 2003), page 143, Washington, DC, USA, 2003. IEEE Computer Society.

[BMS04] Adi Botea, Martin Müller, et Jonathan Schaeffer. Near optimal hierarchical pathfinding. Journal of Game Development, 1(1) :7–28, 2004.

[Bon01] Bonet, B. (2001). Planning as heuristic search. Artificial Intelligence.

[BW95] Badler, N. I. et Webber, B. L. (1995). Planning and parallel transition networks : Animation's new frontiers. In Pacific Graphics '95.

[BY98] J.-D. Boissonnat et M. Yvinec. Algorithmic Geometry. Cambridge University Press, 1998.

[Cam97] Stephen Cameron. Enhancing gjk: Computing minimum and penetration distances between convex polyhedra. In Proceedings of International Conference on Robotics and Automation, pages 3112-3117, 1997.

[CL07] Jen-Yao Chang and Tsai-Yen Li. Simulating crowd motion with shape preference and fuzzy rules. Proceedings of International Symposium on Artificial Life and Robotics, 2007.

[CLMP95] Jonathan D. Cohen, Ming C. Lin, Dinesh Manocha, and Madhav Ponamgi. l-collide: an interactive and exact collisi- III detection system for large-scale environments. In SI3D '95: Proceedings of the 1995 symposium on Interactive 3D graphics, pages 189-ff. New York, NY, USA, 1995. ACM Press.

[CLMP95] Jonathan D. Cohen, Ming C. Lin, Dinesh Manocha, and Madhav Ponamgi. l-collide: an interactive and exact collisi- III detection system for large-scale environments. In SI3D '95: Proceedings of the 1995 symposium on Interactive 3D graphics, pages 189-ff., New York, NY, USA, 1995. ACM Press.

[CM02] Cavazza, M., Charles, F. et Mead, S. J. (2002). Interacting with virtual characters in interactive storytelling. In AAMAS 02 : Proceedings of the first international joint conference on Autonomous agents and multiagent systems, pages 318–325, New York, NY, USA. ACM.

[DK90] David P. Dobkin and David G. Kirkpatrick. Determining the separation of preprocessed polyhedra - a unified approach. In ICALP '90: Proceedings of the 17th International Colloquium on Automata, Languages and Programming, pages 400-£113, London, UK, 1990. Springer-Verlag.

[Feu00] Franck Feurtey. Simulating the collision avoidance behavior of pedestrians. Department of Electronic Engineering, 2000.

[FN71] Fikes, R. E. et Nilsson, N. J. (1971). Strips : A new approach to the application of theorem proving to problem solving. Artificial Intelligence.

[FT99] Funge, J., Tu, X. et Terzopoulos, D. (1999). Cognitive modeling : knowledge, reasoning and planning for intelligent characters. In SIGGRAPH 99 : Proceedings of the 26th annual conference on Computer graphics and interactive techniques, pages 29–38, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

[Fun99] Funge, J. (1999). Representing knowledge within the situation calculus using intervalvalued epistemic fluents. Journal of Reliable Computing.

[FWTK03] B. R. Fajen, W. H. Warren, S. Temizer and L. Kaelbing, A dynamical model of visually-guided steering, obstacle avoidance, and route selection, International journal of Computer Vision 54, 13–34 (2003).

[GD04] Stephans Guy and Gilles Debunne. Monte-carlo collision detection. Technical Report RR-5136, INRIA, March 2004.

[Gei00] Bernhard Geiger. Real-time collision detection and response for complex environments. In CGI '00: Proceedituts of the International Conference on Computer Graphics, page 105, 2000.

[GGK06] Alexander Grefl" Michael Guthe, and Reinhard Klein. Gpu-based collision detection for deformable parameterized surfaces. Computer Graphics Forum, 25(3):497-506, September 2006.

[GJK88, E.G. Gilbert, D.W. Johnson, and S.S. Keen hi. A fast procedure for coinputing the distance between complex' objects in three-dimensional space. Robotics and A utomotion, IEEE Journal of {see also IEEE Transactions on Robotics and Automation], 4(2):193-203, Apr 1988.

M.C. Lin and J.F. Canny. A fast algorithm fOI" incremental distance calculation. *Robotics and Automation, 11J91. Proceedinqs.*

,LC91] *1991 IEEE International Conference on,* pages 1008-1014 vol.2, Apr 1991.

[GJK88] E.G. Gilbert, D.W. Johnson, and S.S. Keen hi. A fast procedure for

coinputing the distance between complex' objects in three-dimensional space. *Robotics and A utomotion, IEEE Journalof {see also IEEE Transactions on Robotics and Automation],4(2):193-203,* Apr 1988.

[GKJ+05] Naga K. Govindaraju, David Knott, Nitin Jain, Ilknur Kabul, Rasmus Tamstorf, Russell Gayle, Ming C. Gin, and Dinesh Manocha. Interactive collision detection between deformable models using chromatic decomposition. ACM Trans. Graph., 24(3):991-999,2005.

[GKLM07] Naga K. Govindaraju, Ilknur Kabul, Ming C. Lin,and Dinesh Manocha. Fast continuous collision detection among deformable models using graphics processors. Comput. Graph. 31(1):5-14, 2007.

[GLM04] Naga K. Govindaraju, Ming C. Lin, and Dinesh Manocha. Fast and reliable collision culling using graphics hardware, In VRST '04: Proceedings of the ACM symposium on Virtual reality software and technology, pages 2-9, New York, NY, USA, 2004. ACM.

[GLM96] S, Gottschalk, M. C. Lin, and D. Manocha. Obbtree: a hierarchical structure for rapid interference detection. In SIGURAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, pages 171-180, New York USA, 1996. ACM Press.

[Gre00] Robin Green. Steering behaviours. Dans Tutorial at SigGraph’00, 2000.

[GRLM03, Naga K. Govindaraju, Stephane Redon, Ming C. Lin, and Dinesh Manocha. Cullide: interactive collision detection between complex models in large environments using graphics hardware. In HWWS '03: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware, pages 25-32, .vire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

Hanyoung Jang and JungHyun Han. Fast collision detection using

the a-buffer. Vis. Comput., 24(7):659-667, 2008.

JH08]

[Gw10] Goldstone, Will. 2010. Unity, Développez des jeux 3D avec. Le Programmeur. s.l. : Pearson, Page 300, 2010.

[HB06] Christopher Hartman et Bedrich Benes. Autonomous boids : Research articles. Computer Animation and Virtual Worlds, 2006.

[HBJT00] Dirk Helbing, Lubos Buzna, Anders Johansson, et Torsten Werner. Selforganized pedestrian crowd dynamics : Experiments, simulations, and design solutions. Transportation Science, 2000.

[HBJT05] Dirk Helbing, Lubos Buzna, Anders Johansson, et Torsten Werner. Selforganized pedestrian crowd dynamics : Experiments, simulations, and design solutions. Transportation Science, 2005.

[HFSQ01, W. Huagen, F. Zhaowei Fan, G. Shuming, and P. Qunsheng. A parallel collision det'ection algorithm base on hybrid bounding volume hierarchy. In Proc. of CAD and Graphics, pages 521-528, 2001.

Duksu Kim, Jae-Pil Heo, .Jaehyuk Huh, .John Kim, and Sung-Eui Yoon. HPCCD: Hybrid parallel continuous collision detection.

KHH+09, *Computer Graphics' Forum (Pacific Graphics),* 28(7), 2009.

Min Tang, Dinesh Manocha, and Ruofeng Tong. Mccd: Multi-core collision detection between deformable models. *Graphical Models,* 72(2):7-23,2010.

TMT10]

[HFSQ01,]. W. Huagen, F. Zhaowei Fan, G. Shuming, and P. Qunsheng. A parallel collision det'ection algorithm base on hybrid bounding volume hierarchy. In Proc. of CAD and Graphics, pages 521-528, 2001.

[HFV00] Dirk Helbing, I. Farkas, et Tamas Vicsek. Simulating dynamical features of escape panic. Nature, 407 :487–490, 2000.

[Hub93] P.M. Hubbard. Interactive collision detection. In IEEE Symposium on Research Frontiers in Virtual Reality, pages 24-31, 1993.

[Hub95] Philip M. Hubbard. Collision detection for interactive graphics applications. IEEE Trans. on Visualization 'and Comput. Graph., 1(3):218-230, 1995.

[HZLM01, Kenneth E. Hoff, III, Andrew Zaferakis, Ming Lin, and Dinesh Manocha. Fast and simple 2d geometric proximity queries using graphics hardware, In I3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics, pages 145-148, New York, NY, USA, 2001. ACM,

HIZ+02] Kenneth E. Hoff, III, Andrew Zaferakis, Ming Lin, and Dinesh Manocha. Fast 3d geometric proximity queries between rigid and deformable models using graphics hardware ace ·leration. In *TechnicalReport TR02-004, Department of Computer Science, UNCChapel Hill,2002.*

[JFSO06] J. J. Jimenez, F. R. Feito, R. J. Segura, and C. J. Ogavar. Particle oriented collision detection using simplicial coverings and tetratrees. Computer Graphics Forum, 25(1):53-68, 2006

[JH08] Hanyoung Jang and JungHyun Han. Fast collision detection using the a-buffer. Vis. Comput., 24(7):659-667, 2008.

[KBT02] Kallmann, M. et Thalmann, D. Modeling behaviors of interactive objects for real-time virtual environments. Journal of Visual Languages and Computing,. Computer. Graphics Lab-LIG, Swiss Fed. Inst. of Technol., Lausanne, Switzerland. 13(2):177–95,2002.

[KBT03] M. Kallmann, H. Bieri, et D. Thalmann. Fully dynamic constrained Delaunay triangulations. Geometric Modelling for Scientific Visualization,

2003.

[KHH+09] Duksu Kim, Jae-Pil Heo, .Jaehyuk Huh, .John Kim, and Sung-Eui Yoon. HPCCD: Hybrid parallel continuous collision detection. Computer Graphics' Forum (Pacific Graphics), 2009.

[KHM+98] James T. Klosowski, Martin Held, Joseph S.B. Mitchell, Henry Sowizral, and Karel Zikan. Efficient collision detrction using bounding volume hierarchies of k-DOPs. IEEE Transactions on Visualization and Computer Graphics, volume 4(1):2l-36, 1998.

[KHM▫98] James T. Klosowski, Martin Held, Joseph S.B. Mitchell, Henry Sowizral, and Karel Zikan. Efficient collision detrction using bounding volume hierarchies of k-DOPs. IEEE Transactions on Visualization and Computer Graphics, volume 4(1):2l-36, 1998.

[KPLM98] Shankar Krishnan, Amol Pattekar, Ming C. Lin, and Dinesh Manocha. Spherical shell: a higher order bounding volume for. Fast proximity queries. In WAFR : Proceedings of the third workshop on the algorithmic foundations of robotics on Robotics : the algorithmic perspective, pages 177-190, Natick, MA, USA, 1998.

[KPLM98] Shankar Krishnan, Amol Pattekar, Ming C. Lin, and Dinesh Manocha. Spherical shell: a higher order bounding volume for. Fast proximity queries. In WAFR : Proceedings of the third workshop on the algorithmic foundations of robotics on Robotics : the algorithmic perspective, pages 177-190, Natick, MA, USA, 1998.

[LA98] B. Logan et N. Alechina. A* with bounded costs. Dans Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98), pages 444,449, 1998.

[Lam03] Fabrice Lamarche. Humanoïdes virtuels, réaction et cognition : une

architecture pour leur autonomie. PhD thesis, 2003.

[Lat91] J.-C. Latombe. Robot Motion Planning. Boston : Kluwer Academic Publishers, Boston, 1991.

[LC91] M.C. Lin and J.F. Canny. A fast algorithm fOI" incremental distance calculation. Robotics and Automation, 11J91. Proceedinqs., 1991 IEEE International Conference on, pages 1008-1014 vol.2, Apr 1991.

[LCDN06] Julien Lenoir, Stephane Cotin, Christian Duriez, and Paul F. Neumann. Interactive physically-based simulation of catheter and guidewire. Computers fj Graphics, 2006.

[LCDN06] Julien Lenoir, Stephane Cotin, Christian Duriez, and Paul F. Neumann. Interactive physically-based simulation of catheter and guidewire. Computers fj Graphics, 2006.

[LD04] Fabrice Lamarche and Stéphane Donikian. Crowd of virtual humans : a new approach for real time navigation in complex and structured environments. Eurographics, 2004.

[Lev66,Tur90] Greg Turk. Interactive collision detection for molecular graphics. Technical report, Chapel Hill, NC, USA, 1990.

[LGLM99] Eric Larsen, Stefan Gottschalk, Ming C. Lin, and Dinesh Manocha. Fast proximity queries with swept sphere volumes. 1999.

[LKF05] T. I. Lakoba, D. J. Kaup, et N. M. Finkelstein. Modifications of the helbingmolnár- farkas-vicsek social force model for pedestrian evolution. Simulation,2005.

[LMM03] Céline Loscos, David Marchal, et Alexandre Meyer. Intuitive crowd behaviour in dense urban environments using local laws. Dans TPCG ’03 :

Proceedings of the Theory and Practice of Computer Graphics 2003, page 122, Washington, DC, USA, 2003. IEEE Computer Society.

[LqWhX07] Li Liu, Zhao qi Wang, and Shi hong Xia. A volumetric b~unding volume hierarchy for collision detection. In Conipuier-Aided Design and Computer Graphics 10th IEEE International Conference on, pages 485-488, 2007.

[LqWhX07] Li Liu, Zhao qi Wang, and Shi hong Xia. A volumetric b~unding volume hierarchy for collision detection. In Conipuier-Aided Design and Computer Graphics, 2007 10th IEEE International Conference on, pages 485-488, 2007.

[MH69] McCarthy, J. et Hayes, P. J. (1969). Some philosophical problems from the standpoint of artificial intelligence. In Meltzer, B. et Michie, D.editeurs : Machine Intelligence 4, pages 463–502. Edinburgh University Press.

[Mou04] David M. Mount. Handbook of Discrete and Computational Geometry, chapter 38 - Geometric Intersection. eRC Press, 2004.

[MS04] Mateas, M. et Stern, A. (2004). A behavior language : Joint action and behavioral idioms. Springer.

[Mus00] Soraia Raupp Musse. Human crowd modelling with various levels of behaviour control. Thèse de Doctorat, EPFL : École Polytechnique Fédérale de Lausanne, Lausanne, Suisse, 2000.

[Mus00] Soraia Raupp Musse. Human crowd modelling with various levels of behaviour control. Thèse de Doctorat, EPFL : École Polytechnique Fédérale de Lausanne, Lausanne, Suisse, 2000.

[NFSR05] Alexis Nédélec, Dominique Follut, Cyril Septseault, and G Rozec. Emotions, personality and social interactions modelling in a multiagent environment. In CASA 2005, October 2005.

[Nil82] N.J. Nilsson. Principles of artificial intelligence. Springer-Verlag, 1982

[NT97] Noser, H. et Thalmann, D. Sensor based synthetic actors in a tennis game simulation. In Computer Graphics International ’97, pages 189–198. IEEE Computer Society Press,1997.

[OCC88] Andrew Ortony, Gerald Clore, and Allan Collins. The cognitive structure of emotions. Cambridge University Press, 1988.

[OL03] Miguel A. Otaduy and Ming C. Lin. Clods: dual hierarchies for multiresolution collision detection. In SGP '03: Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing, pages 94-101, Aire-la-Ville, Switzerland, 2003. Eurographics Association.

[PAB07] N Pelechano, J M Allbeck, and N I Badler. Controlling individual agents in highdensity crowd simulation. Association for Computing Machinery, 2007.

[PB06] Nuria Pelechano et Norman I. Badler. Modeling crowd and trained leader behavior during building evacuation. IEEE Comput. Graph. Appl., 26(6) :80–86, 2006.

[Pes71] I. Peschl. Passage capacity of door opennings in panic situations. BAUN, 1971.

[PPD07] Sébastien Paris, Julien Pettré, and Stéphane Donikian. Pedestrian reactive navigation for crow simulation : a predictive approach. Eurographics, 2007.

[PTRD01] Marc Parenthoën, Jacques Tisseau, P. Reignier, and F. Dory. Agent's perception and charactors in virtual worlds ; put fuzzy cognitive maps to work. VRIC'01, 2001.

[Rey87] C. W. Reynolds. Flocks, herds, and schools : A distributed behavioral model. Computer Graphics, 1987.

[RKC02a] Stephane Redon, Abderrahmane Kheddar, and Sabino Coquillart. Fast continuous collision detection between rigid bodies. In Proceedings

of Eurographics, 2002.

[RMD00] Soraia Raupp Musse and Thalmann Daniel. Hierarchical model for real time simulation of virtual crowds. Computer Graphics, 2000.

[SA72] Schank, R. C., Goldman, N. M., Rieger, C. J. et Riesbeck, C. K. . Primitive concepts underlying verbs of thought. Rapport technique, Stanford, CA, USA,1972.

[SGG+06] Avneesh Sud, Naga Govindaraju, Russell Gayle, Ilknur Kabul, and Dinesh Manocha. Fast proximity computation among deformable models using discrete voronoi diagrams. A CM Trams. Graph., 25(3):1144-1153,2006.

[ST05] Wei Shao et Demetri Terzopoulos. Autonomous pedestrians. : Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation, pages 19–28, New York, NY, USA, 2005. ACM Press.

[TD00] Gwenola Thomas and Stéphane Donikian. Virtual humans animation in informed urban environments. IEEE, 2000.

[TMT10] Min Tang, Dinesh Manocha, and Ruofeng Tong. Mccd: Multi-core collision detection between deformable models. Graphical Models, 2010.

[Tog55] K. Togawa. Study on fire escapes based on observations of multitude currents. Technical report, Ministry of construction, 1955.

[VMO04] M.B. Villamil, S.R. Musse, and P.L de Oliveira. A model for generating and animating groups of virtual agents. Sao-Leopoldo Brazil, 2004.

[VT01] Kevin Vlack and Susumu Tachi. Fast and accurate spacioremporal intersection detection with the gjk algorithm. ICAT: International Conference on Artificial Reality and Telexistence, pages 79-85, 2001.

[WB05] Wingo Sai-Keung Wong and George Baciu. Cpu-based intrinsic collision

detection for deformable surfaces: Collision detection and deformable objects. Comput. Animat. Virtual Worlds, page 153-161,2005.

[WZ09a, Rene Weller and Gabriel Zachmann. Inner sphere trees for proximity and penetration queries. *Robotics: Science andSystems Conference (RSS),* Seattle, WA, USA, 2009.

WZ09b] Rene Weller and Gabriel Zachmann. Inner sphere trees for proximity and penetration queries. In *2009 Robotics: Science and Systems Conference (RSS),* Seattle, WA, USA, 2009.

[WZ09a] Rene Weller and Gabriel Zachmann. Inner sphere trees for proximity and penetration queries. Robotics: Science and Systems Conference (RSS), Seattle, WA, USA, 2009.

[WZ09b] Rene Weller and Gabriel Zachmann. A unified approach for physically-based simulations and haptic rendering. In *Sandbox'09: Proceedings of the 2009 ACM SIGGRAPH Symposium onVideo Games*, pages 151-159, New York, NY, USA, 2009. ACM.

[ZK07] Xinyu Zhang and Young J. Kim. Interactive collision detection for deformable models using streaming aabbs. IEEE Transactions on Visualization and Computer Graphics, 13(2):318-329, 2007.