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 :

Modélisation topologique et sémantique de l'environnement

Par :
Mme Amina BOUGUETITICHE

Soutenu le : … /…/2011

Devant le jury :

Pr. N. DJEDI	**PR**	**Université de Biskra**	**Président**
Dr. M. BOURAHLA	**MCA**	**Université de M'sila**	**Examinateur**
Dr. M. C. BABAHENINI	**MCA**	**Université de Biskra**	**Examinateur**
Dr. F. CHERIF	**MCA**	**Université de Biskra**	**Rapporteur**

Remerciements

Avant de commencer mes études en post graduation, ma connaissance en matière de synthèse d'images et vie artificielle était au plus bas, voire quasiment nulle. Or durant ces études, ma curiosité m'a amené à découvrir différents domaines tous aussi intéressants les uns que les autres. Le travail effectué avec Dr. Foudil CHERIF m'a également beaucoup apporté, tant au point de vue recherche qu'ingénierie. Ce sujet de magistère m'a vraiment passionné, j'aurai vraiment souhaité poursuivre mes travaux encore pendant quelques temps. Il est vrai que cette période fut rapide voire très courte mais j'y ai énormément appris. Mon choix de poursuite en ce domaine était, avant de commencer, un peu hésitant ; Il est désormais, sûr et définitif.

Aujourd'hui, je suis heureuse d'avoir l'occasion de présenter ma reconnaissance et ma gratitude à tous ceux qui m'ont soutenu de près ou de loin pour mener à bien la réalisation de ce mémoire. Je tiens à remercier ces personnes pour m'avoir permis de réaliser ce travail, que ce soit pour leurs conseils, leurs critiques ou encore leurs soutiens. Je m'excuse de ne pouvoir toutes les citer, mais je sais combien je dois à chacun.

Je remercie particulièrement :

Dr. Foudil CHERIF pour la justesse de son encadrement et pour sa grande vivacité intellectuelle, sa disponibilité, son œil critique et ses nombreux conseils toujours constructifs.

L'ensemble des membres du jury d'avoir accepté d'être les examinateurs. C'est un réel honneur de présenter mon travail devant une assemblée aussi prestigieuse qu'agréable.

Les collègues de la post graduation qui ont fait que l'ambiance et l'atmosphère de travail soient des plus agréables.

Mes amies Malika, Mébarka, Nadia et Najla dont le soutien et l'aide étaient sans limites.

Mes parents, mes frères et ma sœur qui ont toujours été là pour me soutenir dans les phases les plus difficiles. Je les remercie d'avoir toujours cru en moi et de m'avoir permis de faire mes études dans les meilleures conditions.

Enfin, j'ai bien sûr gardé le meilleur pour la fin : Mon mari (la personne qui mérite d'avoir son nom au même titre que moi sur ce mémoire) ! Mais les mots ne servent à rien : merci à toi Mohammed ! Tu as su faire preuve de beaucoup de compréhension vis à vis des sacrifices que j'ai faits notamment en ce qui concerne ma vie privée. Merci d'avoir franchi avec moi cette étape importante de ma vie, Merci à toi, je n'y serais pas arrivé sans toi.

Table des matières

Liste de figures

Liste de Tableaux

Résumé

Les environnements 3D sont apparus dans un premier temps dépourvus d'entités mobiles et purement géométriques. Des personnages virtuels y ont, par la suite, été ajoutés, créant ainsi une pseudo-vie au sein même de ces environnements. Ces personnages n'étaient qu'animation, ils suivaient une trajectoire déterminée au préalable par leurs créateurs respectifs. Au fur et à mesure, ces personnages se sont dotés d'une certaine autonomie, ils étaient désormais capables d'analyser brièvement le monde extérieur afin de choisir la meilleure stratégie à adopter.

Ces humanoïdes sont à l'heure d'aujourd'hui capables de se déplacer au sein d'un environnement complexe, de s'éviter les uns les autres ou bien même de capter une certaine sémantique environnementale. Cette navigation individuelle est un élément essentiel au réalisme des simulations. Afin d'accroître ce réalisme nous nous sommes penchés sur le problème de la modélisation de l'environnement au sein duquel se trouvent ces personnages autonomes. Pour faciliter la navigation de ces derniers, il est important de pouvoir capter la sémantique de cet environnement. En effet, dans le monde qui nous entoure, notre comportement se calque sur notre analyse des lieux au sein desquels nous naviguons. Nous n'adoptons pas la même attitude lorsque nous nous déplaçons sur un trottoir que lorsque nous traversons une rue avec une circulation dense. Notre vitesse est en permanence adaptée à l'environnement et varie progressivement. Ainsi, elle est réduite à l'approche d'un obstacle, d'un escalier ou d'un virage.

Nous proposons un modèle unifié de l'environnement qui peut prendre en compte la topologie et la sémantique des lieux. La représentation topologique de l'environnement est en 2D1/2. Elle est obtenue automatiquement à l'aide d'une décomposition exacte des surfaces navigables en des cellules convexes, plutôt qu'une décomposition sous forme de grille jugée trop approximative. A cette représentation topologique on associe à certaines zones de la surface navigable le type de la zone (rue, gazon, porte, escalier,…etc). Cette information permet à l'humanoïde de connaître le lieu ou il se trouve et lui permet ainsi d'adapter son comportement.

Abstract

3D environments have appeared at first devoid of mobile entities and purely geometric. Virtual characters have been subsequently added, creating a pseudo-life within these environments. These characters were an animation; they followed a path predetermined by their respective creators. As and when these characters are endowed with certain autonomy, they were now able to analyze briefly the outside world to choose the best strategy.

These humanoids are on time today able to move in a complex environment, to save themselves from each other or even to capture some semantic environment. This personal navigation is essential to the realism of simulations. To increase the realism we focused on the problem of modeling the environment in which these characters are independent. To facilitate navigation of the latter, it is important to capture the semantics of this environment. Indeed, the world around us, our behavior is modeled on our site analysis in which we sail. We do not adopt the same attitude when we travel on a sidewalk when we cross a street with heavy traffic. Our speed is continuously adapted to the environment and varies gradually. Thus, it is reduced when approaching an obstacle, a staircase or a turn.

We propose a unified model of the environment that can take into account the topology and semantics of the premises. The topological representation of the environment is 2D1/2.It is obtained automatically by using an exact decomposition of navigable surfaces in convex cells, rather than a breakdown in a grid considered too rough. To the topological representation we associate with certain areas of the surface the type of area (street, grass, doors, stairs, etc ...). This information allows the humanoid to know the place where it is located and thus allows it to adapt its behavior.

ملخص

البيئات الافتراضية ثلاثية الأبعاد ظهرت في الأول خالية من الكيانات المتحركة و هندسية بحتة. تم في وقت لاحق اضافة شخصيات خيالية اليها ، مما خلق شبه حياة داخل هذه البيئات. وكانت هذه الشخصيات عبارة عن رسوم متحركة ، تتبع مسار محدد سلفا من قبل المصممين لها. شيئا فشيئا أصبحت هذه الكيانات ذاتية التحكم ، وأصبحت بعد ذلك قادرة على تحليل العالم الخارجي لاختيار أفضل استراتيجية تتبعها.

الشخصيات الافتراضية في وقتنا الحالي قادرة على التحرك في بيئة معقدة ، تتفادى بعضها البعض و حتى قادرة على التقاط بعض الدلالات البيئية. هذا التنقل داخل البيئة أمر ضروري لواقعية المحاكاة. لزيادة الواقعية ركزنا على مشكلة نمذجة البيئة التي تتواجد بها هذه الشخصيات. لتسهيل التنقل لهذه الأخيرة ، من المهم لها التقاط دلالات هذه البيئة. والواقع أن سلوكنا كبشر يأتي على غرار تحليلنا للعالم الذي نتحرك فيه. نحن لا نعتمد على نفس السلوك عندما نتحرك على الرصيف وعندما نعبرشارعا يتميزبحركة مرور كثيفة. سرعتنا تتكيف باستمرار مع البيئة وتختلف تدريجيا. وهكذا ، يتم تقليلها عندما نقترب من حاجز أو الدرج أو منعطف.

نقترح نموذج موحد للبيئة الذي يمكن أن يأخذ في الاعتبار طوبولوجيا ودلالات الأماكن. التمثيل الطوبولوجي للبيئة هو على شكل بعدين و نصف ويتم الحصول عليه تلقائيا باستخدام التقسيم الدقيق للأسطح الصالحة للتنقل الى خلايا محدبة الشكل ، بدلا من التقسيم على شكل شبكة من الخلايا مربعة الشكل تعتبر غير دقيقة للغاية. لهذا النموذج الطوبولوجي نقرن بعض المناطق من المساحات القابلة للتنقل عليها بنوع المنطقة (طريق, عشب, باب, سلالم....). هذه المعلومات تسمح للشخصية الافتراضية بالتعرف على الأماكن التي تتواجد بها و بالتالي تسمح لها بتأقلم سلوكها.

Introduction générale

Introduction

Les ordinateurs permettent de représenter de grands mondes virtuels tels que ville, gare, musée… cette représentation basée sue des images de synthèse 3D est un outil d'aide aux concepteurs de bâtiments ou d'aménagement urbains. Afin de rendre ces environnements vivants, des personnages virtuels sont ensuite apparus pour peupler ces environnements.

Dans un premier temps, le comportement émergent de chaque personnage était modélisé à l'aide d'outils d'animation. Ceux-ci permettaient l'animation de chaque personnage en décrivant leurs mouvements et leur déplacement dans le monde virtuel. Cette description exhaustive limitait les animations à des séquences répétitives et demandait beaucoup de temps. C'est le cas par exemple des premiers films d'animation.

Afin d'offrir des outils performants permettant d'automatiser l'animation de ces personnages, la recherche s'est orienté vers le peuplent du monde virtuel à l'aide d'entités autonomes ayant un comportement émergeant plausible : il s'agit du domaine de l'*Animation Comportementale*. Dans le cadre de ce document, nous allons principalement nous intéresser aux humanoïdes autonomes. Ce sont des entités autonomes c'est-à-dire que chaque entité prend une décision en fonction de sa perception locale et de l'existence d'un objectif à atteindre pour réagir en effectuant des actions dans le monde virtuel telles que se déplacer, prendre un objet.

Domaines d'application

Jeux vidéo. Les joueurs demandent de plus en plus de réalisme de la part des jeux. En permettant au personnage contrôlé par le joueur d'interagir avec son environnement augmenterait la sensation d'immersion.

Films et effets spéciaux. La génération automatique d'animations d'interaction entre un personnage généré par ordinateur et son environnement faciliterait le travail des animateurs qui doivent, actuellement, générer les animations à la main, image par image.

Réalité virtuelle. Des agents autonomes dans un monde virtuel, ne peuvent se contenter de juste s'y déplacer. Les doter de plusieurs moyens d'interagir avec leur environnement en temps réel augmenterait la sensation d'immersion de l'utilisateur.

Ergonomie et validation de sites. Les lieux interactifs permettraient aussi de tester les interactions d'un nouveau modèle d'endroit sur une maquette virtuelle. De plus, il serait possible de tester l'effet qu'aurait le placement de différents types de lieux ou d'objets dans un environnement et ainsi permettre la validation de sites avant leur construction.

Position du problème

Le domaine de l'animation comportementale vise donc à proposer des humanoïdes virtuels avec une apparence et un comportement se rapprochant de ceux de vrais êtres humains. Cependant, une représentation de l'environnement sous la forme d'une géométrie 3D n'est pas bien adaptée pour le peuplement. La structure de données utilisée la plupart du temps, est un ensemble de points placés en trois dimensions. Ces points sont associés pour former les facettes triangulaires qui composent les objets de la scène. Ces données brutes sont très coûteuses à parcourir pour prédire les collisions et planifier un chemin. La première difficulté consiste à structurer l'environnement pour gérer efficacement les requêtes, comme par exemple trouver un chemin ou détecter les obstacles proches. On va donc découper et simplifier cet espace pour en extraire des caractéristiques qui nous intéresserons pour l'animation et plus particulièrement la gestion de la navigation des humanoïdes.

En effet, pour pouvoir se déplacer correctement dans son environnement, une entité doit en avoir sa propre représentation. On doit dès lors définir une structure permettant de représenter la topologie du terrain, mais aussi la présence ou non d'entités et d'objets interactif présents à proximité de chaque humanoïde. Par exemple, chacun doit pouvoir décider, selon le but qu'il veut atteindre, du meilleur chemin à adopter. Les entités doivent pour ce faire avoir accès aux informations à leurs portées (connaissances personnelles de la topologie des lieux, informations disponibles dans leurs champs de vision, ...)

Nous nous plaçons ici dans le cadre de la modélisation non seulement de la géométrie de la scène, mais aussi de toutes les informations pertinentes pour les entités comportementales à simuler. En effet, il est nécessaire de représenter les éléments symboliques importants de l'environnement dans lequel les entités vont évoluer, éléments qui vont influer sur leurs comportements. Ainsi, pour reproduire des comportements réalistes d'entités autonomes

évoluant dans une scène synthétique, il est nécessaire de posséder d'autres informations que la seule connaissance de la géométrie de ladite scène.

Notre approche

La plupart des structures représentant l'environnement travaillent en 2D. Ces techniques ne prennent en compte aucune notion de hauteur, et sont donc contraintes à manipuler des zones de navigation planes. Nous proposons une solution en deux dimensions et demi (2D1/2) consistant à ajouter une notion de hauteur aux zones navigables ce qui offre la possibilité de conserver la simplicité des algorithmes en 2D tout en permettant de distinguer, par exemple, le trottoir de la route. La technique adoptée donc est le maillage de navigation pour représenter les zones navigables.

Dans notre vie de tous les jours, notre manière de nous déplacer et de focaliser notre attention varie en fonction de l'environnement ou on évolue. Par exemple un piéton fera plus attention au trottoir sur lequel il se trouve et il délaissera la surveillance de la route. D'autres types d'environnements comme les ascenseurs ou les tapis roulants nécessitent une adaptation du comportement. Ces quelques exemple illustrent le concept d'*affordance* introduit par J.J.Gibson [GIB86]. Ce concept représente le fait qu'un endroit ou un objet offre aux humains ou aux animaux des possibilités de s'y déplacer ou de l'utiliser. Pour augmenter la plausibilité du comportement des humanoïdes, nous proposons de modéliser ce concept. Ainsi le deuxième aspect nécessaire à notre modèle de l'environnement est la prise en compte de la typologie des lieux dans le raisonnement des entités simulées, et donc de proposer un environnement informé. Pour ce faire, nous avons besoin d'intégrer les zones interactives à notre description de l'environnement.

Plan du mémoire

Pour répondre à cette problématique, nous organiserons ce document en cinq chapitres :

Dans le chapitre I « Animation comportementale » : nous allons introduire le contexte de notre travail, expliquer la terminologie utilisée dans ce document et présenter l'état de l'art du domaine.

Dans le chapitre II « Environnements virtuels » : Nous nous concentrerons sur les environnements virtuels et nous présenterons quelques approches sur la manière de modéliser ces environnements.

Dans le chapitre III « Environnements informés » : nous présenterons quelques travaux réalisé sur la notion d'environnements informés.

Dans le chapitre IV « Contribution » : nous présenterons TopoSyn, l'implémentation de notre système pour des agents humanoïdes.

Dans le chapitre V « Implémentation et Résultats » : nous exposerons les résultats obtenus.

La conclusion mettra en avant les perspectives d'évolution du modèle proposé.

Première partie

État de l’art

Chapitre 1

L’animation comportementale

1 Animation comportementale

1.1 Introduction

La simulation comportementale, présentée dans la section 1.2, est une discipline récente faisant partie de l'animation et astreinte à la recherche des comportements pour les créatures et les personnages qui peuplent les environnements virtuels. Ces agents doivent être avant tout réalistes et accomplir de manière autonome des objectifs assignés par l'animateur. Pour ce faire, ils possèdent un contrôleur, en charge du mécanisme de sélection de l'action (section 1.3). Dans la section suivante, en 1.4, nous présenterons les modèles informatiques simulant le comportement humain. En s'inspirant de disciplines variées, les agents de l'animation comportementale mettent en oeuvre différentes approches. On retiendra toutefois la taxonomie suivante : le comportement réactifs (section 1.4.4.1) et le comportement cognitif (section 1.4.4.2). Dans la section 1.5, nous nous intéressons à la capacité de perception d'un agent. Différentes techniques sont utilisées pour que les personnages puissent retirer de l'environnement les informations qui leur seront utiles. Enfin, la section 1.6 conclura ce chapitre en récapitulant les différentes techniques de représentation de connaissances.

1.2 L'animation comportementale

L'animation — ou simulation — comportementale est une branche de l'animation ayant pour objectif la production de comportements pour les acteurs de l'animation. L'animation par ordinateur est elle-même issue de l'informatique graphique, une discipline qui a pour but de produire des images grâce à des moyens informatiques. De la même manière que la photographie a donnée naissance au cinéma, l'animation est apparue dès que les moyens techniques se sont révélés suffisants. Aujourd'hui, elle occupe une place prépondérante dans la création de films, de jeux vidéo, d'applications de réalité virtuelle (visites virtuelles de villes ou de lieux historiques, communautés virtuelles, simulations d'entraînement, etc.).

Deux acteurs aux rôles bien différents interviennent en animation. L'animateur est une personne physique responsable de la définition de la scène et des personnages. Son rôle exige différents degrés d'implication. L'utilisateur est aussi une personne physique, observateur objectif de l'animation — du film —, ou bien subjectivement immergé dans la scène par le truchement d'un avatar [1] — jeu vidéo, simulation, etc.

[1] Un avatar est un personnage virtuel représentant un utilisateur.

(a) Le film d'animation «Mulan» de Disney.

(b) Le jeu vidéo « Les Sims 2 » de Electronic Arts.

(c) L'environnement virtuel du jeu en ligne « Second Life ».

(d) La simulation d'entraînement Firefighter Training (IRIT).

(e) Visite virtuelle de la ville de Grenoble (Artesia).

(f) Visite virtuelle de la cité des sciences et de l'industrie (IRISA).

Figure 1-1 – Quelques exemples d'applications de l'animation : le cinéma d'animation (a), les jeux vidéos (b, c), les simulations d'entraînement (d) ou la visite virtuelle de lieux touristiques (e, f)

Le point crucial de l'animation est donc l'immersion de l'utilisateur. Ce terme peut être définit comme la faculté de se projeter virtuellement dans le film, le jeu ou la simulation, en s'imaginant à la place d'un personnage ou de son avatar. Cette immersion dépend bien évidemment de la qualité du rendu graphique des scènes, mais aussi du comportement des figurants ou des personnages non joueurs. En effet, il est difficile d'envisager l'immersion de l'utilisateur dans une scène vide ou comportant des personnages non animés.

1.3 Les agents virtuels

Pour introduire la notion d'agent, D. Panzoli [Pan08] s'est reposé sur la définition de l'animation comportementale définie dans [RHJL01] : « L'animation comportementale est une partie de l'animation qui se rapproche des systèmes réels de par son principe de fonctionnement en assignant aux acteurs ou systèmes animés des comportements indépendants. Ces derniers ne seront alors plus régis par un système global gérant le mouvement de tous les acteurs mais par un mécanisme de décision local placé dans chaque individu. L'animation comportementale est donc un moyen de faire interagir de manière naturelle des acteurs en simulant leurs capacités dans un environnement».

Ainsi, dans le domaine de l'animation comportementale le terme d'agent selon D. Panzoli désigne une créature artificielle autonome, incarnée par une représentation graphique dans un environnement virtuel. Cette propriété d'autonomie est cruciale car elle est garante de

la cohérence des agents. Elle implique l'utilisation d'un mécanisme dit «de sélection de l'action».

1.3.1 Les propriétés des agents

Parmi les nombreuses propriétés pouvant qualifier un agent, les suivantes sont récurrentes en animation comportementale selon D. Panzoli [Pan08].

L'autonomie permet à l'agent de produire des actions à partir de perceptions, et éventuellement d'états internes ou d'une mémoire. Aucune intervention extérieure, de l'animateur en particulier, n'est requise pour contrôler l'agent. Cette propriété est commune à tous les agents de l'animation comportementale.

La réactivité de l'agent est sa capacité à agir rapidement, lui permettant également de saisir des opportunités.

L'intentionnalité ou pro-activité définit l'aptitude de l'agent à manipuler des buts explicites et accomplir des plans. L'agent intentionnel possède l'initiative des ses actions, ce qui peut apparaître comme contradictoire à la notion de réactivité.

La sociabilité permet à l'agent de communiquer avec les autres agents, ou avec l'utilisateur par le biais d'un langage symbolique.

La situation d'un agent traduit le fait que son comportement est intégralement défini par son environnement, par le biais de l'évolution ou de l'apprentissage. Toutes les capacités de l'agent situé sont nécessaires à sa survie dans l'environnement.

L'adaptabilité de l'agent traduit son aptitude à maintenir sa situation, c'est-à-dire à s'adapter aux changements de l'environnement. Pour cela, l'agent adaptatif possède des mécanismes d'apprentissage et/ou d'évolution qui s'appliquent durant la vie de l'agent dans l'environnement.

1.3.2 Humanoïde

L'humanoïde selon G. Thomas [Tho99] est défini par plusieurs modèles interdépendants se situant à différents niveaux d'abstraction. La figure 1.2 illustre cette décomposition en couches. Le premier modèle fournit l'apparence de l'humanoïde ; c'est le modèle géométrique. Ce modèle géométrique articulé est animé grâce à un modèle restituant les capacités motrices d'un humain (la marche, la préhension. etc.). Les capacités motrices sont quant à elles contrôlées par un modèle décisionnel. Ce modèle décisionnel couplé à un modèle de perception permet à un humanoïde de décider et de planifier ses actions en fonction de l'environnement dans lequel il évolue.

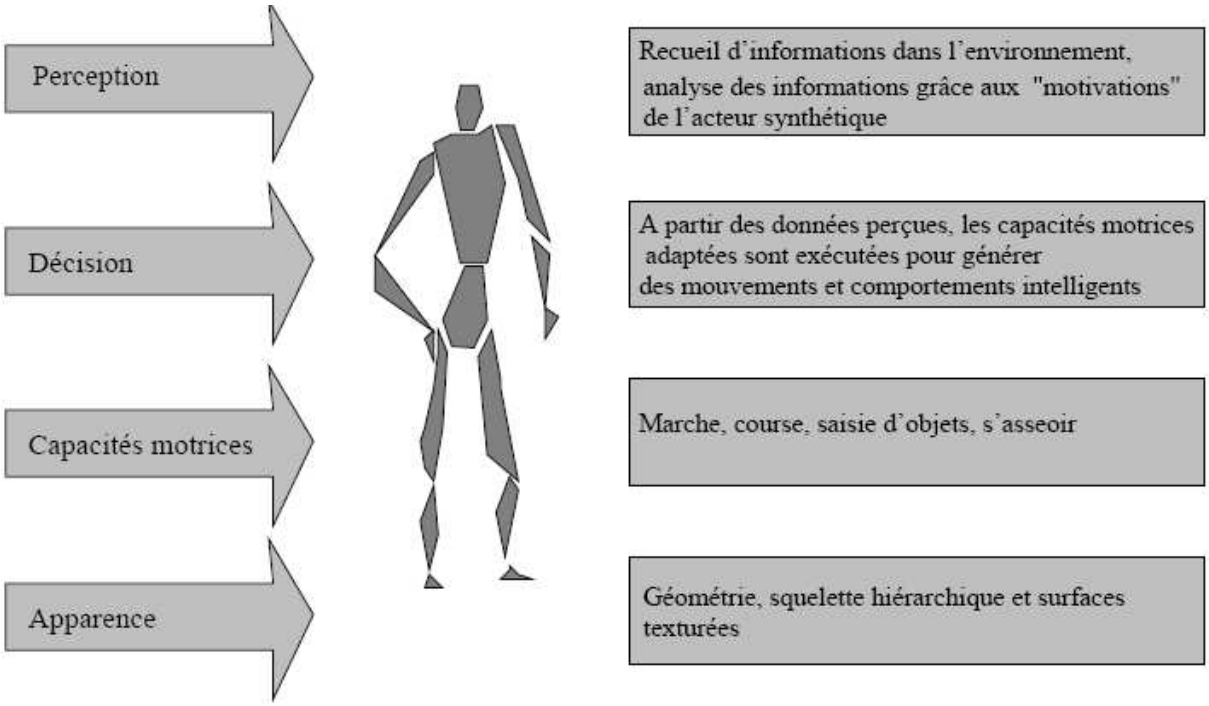


Figure 1-2 – Les modèles de l'humanoïde [Pan08].

1.3.3 Principe de sélection de l'action

Du point de vue de l'animation comportementale, produire le comportement d'un agent revient à définir et modéliser ses interactions avec l'environnement.

Selon le modèle général défini dans [TT94], et schématisé dans la figure 1.3, celles-ci sont synthétisées à travers une boucle perception!décision!action. Cette boucle se répète tant que la simulation n'est pas arrêtée, que l'agent est vivant ou qu'il n'a pas atteint son but — si tant est qu'il en possède un. Elle consiste en trois phases :

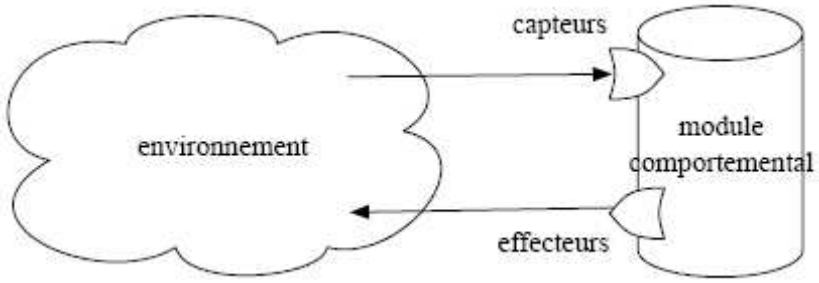


Figure 1-3 – Le modèle général comportemental tel que défini dans [TT94]. Ce schéma illustre un point essentiel : bien que l'agent soit immergé dans un environnement, il est conceptuellement distinct de celui-ci et il assume la gestion de son comportement de manière autonome, par le biais de capteurs et d'effecteurs.

La phase de perception consiste à extraire des états de l'environnement et les transformer en valeurs exploitables. Pour ce faire, l'agent est muni de capteurs plus ou moins spécialisés dans l'extraction de caractéristiques perceptives particulières. On trouve des capteurs réalistes comme des capteurs de vision qui reprennent le fonctionnement de l'oeil ou des capteurs de collision qui simulent le sens du toucher. A l'inverse, on peut utiliser des capteurs imaginaires comme un capteur de position absolue dans l'espace, un capteur de distance à un objet particulier, etc.

La phase de décision a pour but de prendre en compte la perception, et les éventuelles informations internes à l'agent, afin de proposer une action.

La phase d'action est prise en charge par des effecteurs. Ils agissent dans l'environnement en traduisant chaque action décidée par le processus de décision en un changement d'état dans l'environnement.

Ce principe de fonctionnement, librement inspiré de l'être humain, est commun à tous les agents de l'animation comportementale. Perception, décision et action sont gérées par un contrôleur d'animation — ou module comportemental — intégré à l'agent. Nous verrons dans les sections suivantes qu'il existe de nombreuses techniques dédiées à la production du contrôleur d'un agent, chacune conduisant à des propriétés différentes.

1.4 Comportement et réalisation d'actions

1.4.1 Le comportement

Le comportement est une « manière d'être et d'agir des Animaux et des Hommes, manifestation objective de leur activité globale » (d'après Piéron, 1907 ; cité dans Grand Larousse de la Psychologie). Un synonyme est la conduite. Ainsi, pour le psychologue, le comportement est un ensemble de phénomènes observables, qu'il faut distinguer des activités mentales (ex. raisonnement) ou encore des variables internes au sujet (ex. motivation). La démarche consiste ensuite à inférer le fonctionnement interne du sujet à partir de son comportement.

1.4.2 L'action

D'après J. Ferber [Fer95], l'action est en premier lieu ce qui modifie l'état global du monde, mais c'est aussi une tentative d'influer sur le monde qui peut échouer : son modèle d'action comme réponse à des influences permet de prendre en compte les lois de l'univers et

les tentatives d'action des autres agents. Cependant, nous préférerons la définition de X. Tu [Tu96] :

L'action est la plus petite unité décomposable du comportement, à partir de laquelle peuvent être décrits tous les comportements.

Au besoin, le résultat de l'action dans l'environnement pourra être connu par l'intermédiaire de la perception. Cette définition a l'avantage de ne pas distinguer les actions internes des actions externes, c'est-à-dire de ne pas réduire l'action à l'interaction avec l'environnement. Elle ne distingue pas non plus les actions sur l'environnement de la communication. Par la suite, nous nous intéresserons donc aux actions en tant que primitives permettant à un agent d'agir sur lui-même et sur son environnement au sens large, c'est-à-dire en y comprenant les autres agents. Le niveau d'abstraction des actions dépend de l'application: l'unité comportementale d'un robot lui permettant de «tourner à gauche» peut nécessiter l'activation de plusieurs moteurs, c'est-à-dire l'utilisation de plusieurs commandes, et pourtant être considérée comme une action atomique, c'est-à-dire comme l'une des plus petites unités manipulables par le mécanisme de sélection de l'action.

1.4.3 Hiérarchisation des comportements

En fonction des informations perçues dans son environnement, l'être humain connaît les actions qu'il peut y effectuer. Désormais il doit en choisir une à réaliser et pour cela le raisonnement entre en jeu. A. Newel [New90] propose dans ses travaux sur l'analyse de l'architecture cognitive, un modèle hiérarchique à cinq niveaux. Dans cette représentation chaque niveau de raisonnement est caractérisé selon deux critères couplés : le temps de réalisation des différentes actions et la complexité. Ils possèdent chacun des relations avec les niveaux supérieurs et inférieurs. La décomposition de A. Newel peut se représenter sous forme pyramidale et est définie comme suit (voir figure 1.4) :

- Le niveau biomécanique : Il correspond aux fonctions de base, comme le mouvement des muscles. Il a des actions allant de 100 ms à 10 ms.
- Le niveau réactif : Il concerne les comportements réflexes et ceux ne nécessitant pas de connaissance.
- Le niveau cognitif : Il représente les comportements utilisant la connaissance pour raisonner. Il peut être décomposé en plusieurs sous niveaux :
    - *Action délibérée* permet de trancher entre une opération plutôt qu'une autre via les informations issues de l'environnement.
    - *Opération* permet d'effectuer des opérations élémentaires déjà existantes.

- *Tâche unitaire* permet la composition d'opérations non élémentaires.

- Le niveau rationnel : Il définit les tâches à effectuer et les priorités entre celles-ci. Les actions peuvent aller d'une minute à plusieurs heures.
- Le niveau social : Il est le sommet de la pyramide, c'est donc le niveau dépendant de tous les autres niveaux, par conséquent le plus complexe. Il situe les êtres humains entre eux, d'un point de vue social et relationnel. Ses actions peuvent aller d'un jour à plusieurs semaines voire plusieurs mois.

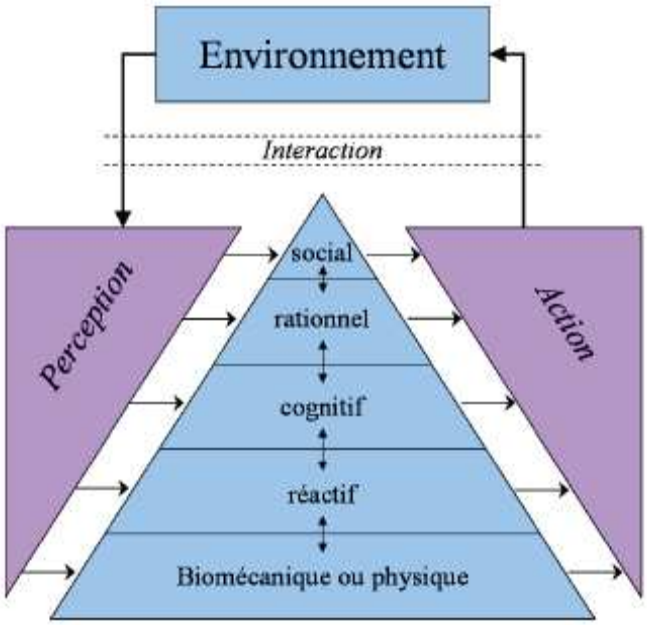


Figure 1-4 – Hiérarchisation des comportements illustrée par A. Newell dans sa pyramide comportementale.

1.4.4 Modèles de comportements

La reproduction d'une certaine forme de raisonnement humain était à l'origine du domaine de l'intelligence artificielle. Avec la démocratisation des jeux vidéos, puis des logiciels de simulations, ce sujet est peu à peu devenu une préoccupation majeure des communautés graphiques, résultant en la création de la thématique d'animation comportementale. D'abord consacrée à la reproduction du déplacement des individus, ce domaine de recherche s'est de plus en plus tourné vers la résolution de comportements plus complexes.

Au cours de cette section, nous présenterons donc deux catégories d'approches utilisées pour simuler le comportement. Premièrement, les modèles réactifs où le raisonnement à court terme est prépondérant. Deuxièmement, les modèles rationnels qui tentent de reproduire le raisonnement à long terme.

1.4.4.1 Comportement réactif

Les comportements réactifs sont ceux qui font le moins intervenir la capacité décisionnelle réfléchie d'un être humain, ne nécessitant que peu ou pas de symbolisation des connaissances. Ce sont les comportements qui correspondent à l'étage conscient le plus bas dans la pyramide de A. Newel (Figure 1.4). Ils peuvent être classifiés en deux catégories.

Tous d'abord, les comportements réflexes, issus d'une réaction automatique à un évènement. On peut ici parler de comportement pré-câblé, où l'effecteur de l'organisme va directement réagir à un stimulus perçu par un capteur, sans avoir recours au processus de sélection d'action, et donc sans que l'information transite jusqu'à la mémoire de travail. Les insectes fonctionnent pratiquement exclusivement avec ce type de comportement, et pour un être humain on peut prendre pour exemple l'équilibre, qui sera conservé par une action réflexe des muscles pilotée par l'oreille interne.

La deuxième catégorie est plus proche de l'étage décisionnel. Elle correspond à l'ensemble des comportements spécialisés auxquels peuvent avoir recours les procédures cognitives plus complexes. Ces comportements sont à usage tellement courant que leur renforcement les a ancrés au plus profond du processus décisionnel, ne nécessitant donc plus d'effort conscient pour y avoir recours. L'exemple déjà utilisé de la conduite automobile illustre cette catégorie de comportements, où un conducteur confirmé ne sera plus conscient qu'il change les vitesses au cours de son parcours, mais seulement du chemin qu'il est en train de suivre.

Il existe principalement trois modèles pour reproduire ces comportements : stimulus/ réponse, basés règles, et les automates.

1. **Stimulus/réponse**

Les modèles stimulus réponse sont directement basés sur le principe de pré-câblage. Ils vont modéliser le comportement comme la transformation de signaux tout au long du parcours entre les capteurs et les effecteurs.

Les systèmes à base de réseaux de neurones [Abd94] entrent dans cette catégorie. Ces réseaux sont capables d'approximer une fonction continue avec n'importe quel degré de précision. Il vont ainsi représenter le processus de décision en essayant de reproduire ses caractéristiques biologiques : un graphe de modules calculatoires très simples mais très nombreux. Ce principe a été utilisé par M. van de Panne et al. [PF93] pour créer un contrôle élémentaire de la locomotion.

Le défaut majeur de ces modèles réside dans leur très faible niveau d'expressivité. Il est ainsi facile d'exprimer un problème mathématique booléen avec ces réseaux, mais la création d'un comportement plus complexe conduit à une explosion de la complexité du graphe. De plus, chaque neurone du graphe ayant un fonctionnement indépendant, la création d'une configuration précise est elle aussi très compliquée, nécessitant le paramétrage complet du système. L'avantage majeur de ces procédures est qu'elles supportent l'apprentissage par renforcement, d'où l'utilisation des réseaux de neurones en intelligence artificielle, notamment pour la création de classifieurs [Mac03].

2. Basés règles

Tout comme la précédente approche, les modèles basés règles vont traiter un ensemble de stimuli produits par différents capteurs. Par contre, le processus de décision peut ici être plus abstrait, permettant à un module de traiter totalement une tâche et de produire le contrôle adéquat. Ces règles peuvent être planes, représentant des comportements relativement indépendants, ou organisées hiérarchiquement, sous forme d'arbre, afin de gérer l'enchaînement des actions.

Un des modèles basés règle spécialisé dans la gestion du comportement de navigation, avec les *Flocks* de C. Reynolds [Rey87]. Ce principe a été repris par F. Lamarche et al. [LD04] ainsi que W. Shao et al. [ST05], toujours en organisant les règles de façon plane et en les évaluant itérativement pour converger vers un résultat. Notons que ces règles planes ne sont généralement utilisées que pour simuler des comportements simples, comme des animaux, ou dans un cadre très restreint pour l'humain. En effet, avec la multiplication du nombre de règles le problème de la concurrence prend de plus en plus d'ampleur avec cette approche, qui ne dispose pas de mécanisme explicite pour la gestion des conflits. Une conséquence en est l'apparition de discontinuités dans le comportement, pouvant aboutir à des oscillations de la prise de décision.

L'organisation sous forme d'arbre se rapproche plus du processus décisionnel [BF97]. Chaque noeud de l'arbre représente ainsi une action atomique, et c'est le parcours de l'arbre, par des algorithmes classiques, qui simulera le processus de décision. Notons tout de même que cette organisation est bijective avec les règles planes, et ne constitue donc qu'un instrument de modélisation.

3. Automates

Les approches à base d'automates sont les plus récentes pour simuler les comportements. Ces approches se basent sur la forte expressivité procurée par les représentations en machines à états pour décrire finement les étapes et enchaînements

d'actions d'un comportement. Ainsi, chaque action d'un comportement sera représentée par un état de l'automate, tandis que les possibilités d'adaptation seront identifiées par les transitions.

O. Renault et al. [RTT90] ont été des pionniers dans l'utilisation de cette technique pour la simulation de navigation d'êtres humains. Du fait de sa souplesse d'utilisation, ce procédé s'est rapidement répandu dans la communauté informatique. Certaines évolutions ont ensuite permis une exécution parallèle et/ou hiérarchique des automates, autorisant respectivement des comportements simultanés et ordonnés. On a ainsi pu voir apparaître les modèles à base de piles d'automates [NT97], où chaque comportement spécialisé est représenté par un automate, qui sont empilés dans l'ordre de leur exécution. Nous pouvons aussi citer les Pat-Nets [BRW97] où des automates parallèles sont utilisés. Enfin, le modèle HPTS [DR95] et son extension HPTS++ [LD02] qui gère à la fois le parallélisme, la hiérarchisation, et la synchronisation entre automates.

Le recours à des automates permet une grande souplesse dans l'expression des comportements. Malgré tout, cette approche est généralement utilisée pour représenter des tâches simples, de comportements réactifs. Deux faits expliquent cela. Premièrement, les actions sont généralement décrites avec une granularité très fine dans les automates, contraignant d'autant plus l'autonomie du système. En effet, l'abstraction du comportement à son simple but est ici peu envisageable, un comportement n'étant alors représenté que par un automate à un seul état. Deuxièmement, tous les enchaînements entre actions doivent être prévus à l'avance pour être envisagés lors de la résolution. Cela rend d'autant plus difficile l'augmentation des comportements gérés, nécessitant une refonte des comportements dépendants déjà disponibles, et conduisant potentiellement à une explosion combinatoire des transitions possibles entre états.

Ces deux inconvénients sont en partie gérés par les automates hiérarchiques, qui imposent tout de même de pouvoir discrétiser les comportements en des modules indépendants. L'avantage majeur de cette approche comparativement aux règles est sa gestion de la continuité dans le comportement. En effet, les états des automates constituent implicitement une mémoire de contexte, les transitions pouvant alors être vues comme des règles locales.

1.4.4.2 Comportement cognitif

Au cœur du mécanisme de décision réside le procédé de sélection d'action. Son but est de trier les différentes tâches requises par l'humain, pour sélectionner celles pouvant ou

devant s'exécuter immédiatement. C'est donc ce mécanisme qui se charge de la discrimination entre plusieurs actions concurrentes vis-à-vis de ressources physiques ou intellectuelles de l'individu.

Deux méthodes permettent cette sélection d'action (Figure 1.5) : la coopération, qui va fusionner plusieurs tâches, et la compétition, qui ne va en sélectionner qu'une seule. La méthode coopérative fonctionne généralement sur des informations quantitatives, en opérant via des fonctions arithmétiques comme la somme ou la moyenne. La méthode compétitive sélectionne une tâche via une fonction de coût associée, symbolisant la priorité de l'action à entreprendre, éventuellement couplée à des informations qualitatives comme l'utilisation de ressources.

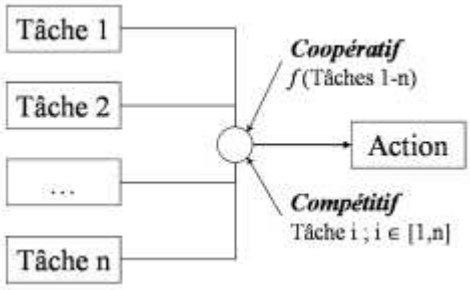


Figure 1-5 – Principe de la sélection d'action. L'action sélectionnée est le résultat d'une opération de fusion et/ou de discrimination entre les tâches requises.

Ce mécanisme est primordial pour la simulation du comportement rationnel. Nous allons voir que les différentes approches étudiées ci-après procèdent à cette sélection en utilisant l'une des deux méthodes précitées, et parfois en les associant.

1. L'Action Selection Mechanism

Le modèle ASM, proposé par P. Maes [Mae 90], fait émerger la décision par l'activation et l'inhibition dynamique des actions. Dans ce système multi-agents, chaque entité est vue comme un ensemble de modules ayant chacun ses compétences spécifiques. Un module de compétence est représentée par quatre valeurs (c,a,d,a) :

c : les pré-conditions de l'action, nécessaires pour son exécution ;

a : les effets d'ajout de l'action ;

d : les effets d'inhibition de l'action ;

a : le niveau d'activation de l'action.

Le modèle peut alors créer des liens entre les modules, de succession et inversement de précédence, ou encore de conflit. Finalement, la sélection s'effectue par l'évaluation du niveau d'activation des actions. Ce niveau est augmenté par les buts de l'agent, mais aussi par

les successeurs d'une action exécutable, ou ses prédécesseurs dans le cas contraire. Il est diminué par les actions entrant en conflit avec l'action courante. Ce système utilise donc une sélection compétitive, basée sur le niveau d'activation.

Plusieurs évolutions de cet algorithme ont été proposées. Le plus proche de la théorie de A. Newel est sans doute le modèle de V. Decugis et J. Ferber [DF98], avec un ASM hiérarchique où les modules de plus bas niveau correspondent aux actions réactives, tandis que ceux de haut niveau correspondent à des comportements plus complexes.

Ce modèle est très intéressant de par sa simplicité de mise en oeuvre, et son fort potentiel d'évolution (l'ajout d'une nouvelle action y est relativement aisé). Néanmoins, ce modèle n'assure pas la robustesse de la décision : si un plan existe, il n'est pas forcément obtenu par cette approche. De plus, le problème des oscillations caractéristique des règles reste posé.

2. Le Situation Calculus

Le situation calculus, introduit par J. McCarthy et P.J. Hayes [MH69], est un modèle basé sur la projection dans le temps de l'individu, permettant d'évaluer les conséquences de ses actions. Cette approche est basée sur six concepts définissant l'environnement de simulation, en utilisant une sémantique basée sur les mathématiques booléennes :

Les situations. Elles représentent des états complets du monde à un instant donné. Ce sont en quelque sorte des photographies instantanées, constituées de faits valides, pouvant ensuite servir comme point de départ du raisonnement.

Les fluents. Ils décrivent une propriété dynamique du monde. Ils sont représentés par une fonction d'une situation, renvoyant l'état correspondant de la propriété.

Les causalités. Ce sont littéralement des relations de cause à effet, associant l'émergence d'un fluent à un autre fluent dans une situation donnée.

Les actions. Elles servent à modifier les situations. Un ensemble de pré-conditions doivent être satisfaites dans la situation courante pour que l'action puisse s'exécuter. Les effets ainsi produits peuvent alors être conditionnels, variant suivant la situation.

Les stratégies. Ce sont des enchaînements d'actions produisant un comportement global. Elles correspondent à des étapes de raisonnement scriptées.

La connaissance et le savoir-faire. Ces deux facultés permettent d'augmenter l'abstraction des actions réalisables. Elles permettent ainsi de décrire des actions potentielles suivant le niveau de connaissance, ou encore des actions permettant d'acquérir de nouvelles connaissances.

Ce formalisme est largement utilisé dans le domaine de l'intelligence artificielle car il permet d'exprimer des mondes très complexes. La sélection d'action y est faite par l'évaluation de tous les enchaînements d'actions permettant d'aboutir à la situation désirée depuis la situation courante. Cette planification aura donc pour effet de produire une série d'actions à réaliser. Dans le domaine de l'animation comportementale, J. Funge [Fun98], avec son modèle CML (pour Cognitive Modeling Language), spécialise le situation calculus pour décrire les comportements de haut niveau d'un agent autonome. Il introduit notamment le principe d'incertitude avec les IVE fluents (pour Interval-Valued Epistemic fluents), qui ne sont plus à valeur unique, mais compris entre deux bornes.

3. Le modèle STRIPS

Le modèle STRIPS (pour STanford Research Institute Problem Solver), introduit par R.E. Fikes et N.J. Nilsson [FN71], est un formalisme simplifié du situation calculus. Il permet ainsi de proposer des algorithmes plus efficaces, la description y étant plus contrainte. Le monde y est représenté par un ensemble de faits, pouvant être présents ou absents. Des actions peuvent ensuite être décrites, possédant des pré-conditions, liste de faits devant être vérifiés, et des effets, sous la forme d'une liste d'ajouts et d'une liste de retraits de faits du monde. La sélection d'actions va alors consister à planifier une série d'actions menant de l'état courant du monde à l'état désiré.

Plusieurs algorithmes existent pour effectuer cette planification. On peut notamment citer Graphplan [BF97], qui va représenter l'évolution du monde par un graphe en couches, alternant des nœuds de type action et des nœuds de type fait. La construction du graphe part de l'état actuel du monde, puis va construire une série de couches jusqu'à ce que l'une d'entre elles ne contienne plus d'actions concurrentes, i.e. qui invalident mutuellement leurs pré-conditions ou leurs effets. Un algorithme va alors effectuer un chaînage arrière, en partant des buts souhaités et en retenant successivement la série d'actions non concurrentes qui reviennent à l'état courant du monde. Cet algorithme garantit l'obtention d'une solution si elle existe, mais le développement du graphe est polynomial en temps et en espace de recherche. Une autre approche existe, nommée HSP [BG01] (pour Heuristic Search Planning), qui va utiliser une heuristique pour limiter le développement du graphe de recherche. Cette heuristique est extraite de la définition du monde sous la forme d'opérateurs STRIPS. Son principe est de négliger les exclusions mutuelles entre actions, et donc de ne considérer qu'un sous ensemble du problème posé. Il existe plusieurs évolutions de cet algorithme, dont certaines ont une évaluation de l'état courant du monde vers les buts, ou dans le sens inverse (aussi dénommé de haut en bas, ou de bas en haut).

4. Les comportements orientés buts

Les comportements orientés buts, connus sous le nom de BDI (Beliefs Desires Intentions), sont une application directe des théories de M.E. Bratman [Bra87]. Trois concepts régissent le fonctionnement de ce modèle :

Les croyances sont les connaissances personnelles qu'un agent a du monde, pouvant être incomplètes voir fausses. En effet, celles-ci sont basées sur les perceptions de l'individu, et sa capacité à appréhender la dynamique de son environnement.

Les désirs sont les buts explicites de l'agent, ce qu'il veut réaliser à plus ou moins long terme.

Les intentions sont les plans comportementaux de l'agent lui permettant d'assouvir ses désirs.

A.S. Rao et M.P. Georgeff [RG95] furent les premiers à proposer une architecture basée sur les BDI. Le principe général de ce modèle est de fournir à l'individu un ensemble de plans lui permettant d'accomplir différents désirs, faisant de lui une sorte de système expert. L'exécution de ces plans est conditionnée par un ensemble de faits relatifs à l'environnement et à l'individu. Dans le cas où plusieurs plans sont sélectionnés, un système d'arbitrage est alors mis en œuvre pour les départager. Une fois qu'un plan est choisi, il devient une intention, et ses actions sont exécutées l'une après l'autre jusqu'à ce qu'une nouvelle sélection soit nécessaire, soit à cause d'une modification des connaissances, soit du fait de la réussite ou de l'échec du plan. Chaque action composant le plan peut alors devenir un sous-but, et provoquer ainsi l'exécution d'un nouveau plan.

1.5 Perception

« Le but de la perception n'est pas de fournir la description générale d'une scène mais d'extraire une information spécifique pour la tâche impliquée dans l'activité en cours. » W. Warren [War95].

Cette citation de W. Warren définit la perception comme un processus actif, analysant les informations perçues au fur et à mesure pour en extraire les données pertinentes compte tenu de l'activité actuelle. Cette extraction de données peut être considérée comme la première phase de la symbolisation des connaissances, où l'être humain doit associer une symbolique aux informations afin de pouvoir les trier.

1.5.1 Simulation de la perception

Il existe deux approches principales pour doter les agents virtuels de perception visuelle: la vision synthétique, et la perception par bases de données [Her06].

a. Vision Synthétique

Dans la technique de la vision synthétique, une image de ce que devrait percevoir le personnage est produit. Cette image peut ensuite être utilisée par l'agent autonome pour déterminer quels sont les objets présents, les distances entre eux, leur vitesse... Ces images doivent être générées, soit par des techniques de projection [FK03], soit par des techniques de lancer de rayon [AK89], permettant un rendu plus réaliste. Dans un rendu par projection, deux modes de coloration sont généralement couplés, afin que l'agent puisse ultérieurement exploiter l'image : la non prise en compte des ombres et des textures (flat shading) et l'utilisation d'une couleur unique par objet ou par catégorie d'objet (false coloring). Ces techniques ont comme principal intérêt d'intégrer le calcul des occlusions entre objets.

b. Bases de données

Dans tout type d'environnements informés, il est aussi possible de percevoir en utilisant des bases de données : les informations géométriques et sémantiques des objets présents dans l'environnement peuvent être obtenues par le personnage en interrogeant ces bases. Dans ce cas là, les occlusions ne peuvent être déterminées, et il faut utiliser des techniques d'algorithmiques géométriques comme le lancer de rayon.

Le tableau 1.1 synthétise les avantages et inconvénients des approches par vision synthétique et par bases de données.

	Vision Synthétique	Base de données
avantages	calculs occlusions entre objets simulation image rétinienne	rapidité facilité d'exploitation
inconvénients	vitesse de génération	calculs occlusions entre objets flexibilité
facteurs limitants	résolution des images fréquence de rendu	création des bases de données mise à jour
identifications des objets	identifiant unique de couleur par objet	directe
calcul des distances	interrogation directe de l'objet lancer de rayons lecture du Z-buffer stéréoscopie	interrogation directe de l'objet lancer de rayons

Tableau 1.1 Comparaison de la vision synthétique et de la perception par bases de données [Her06]

1.5.2 Mécanismes de traitement

Toutes ces informations doivent être transmises à un moment ou à un autre à la partie décisionnelle. C. Septseault [Sep07] distingue deux approches pour récupérer les informations:

- l'approche « bottom-up » ou perception passive. Le module perceptif choisit quelles informations seront enregistrées ou envoyées à la partie décisionnelle. Cette approche est en général utilisée avec une vision synthétique.
- l'approche « top-down » ou perception active. La partie décisionnelle choisit les informations dont elle a besoin. Pour cela, elle peut orienter l'attention du personnage sur une zone, sur des objets ou sur certaines propriétés. Les objets choisis sont en général liés à la tâche en cours, ou nécessaires pour continuer l'élaboration d'un plan pour atteindre un des buts du personnage. Dans un environnement informé, cela permet au personnage d'orienter les requêtes qui sont utilisées pour interroger les bases de données de l'environnement.

Différentes approches existent quant à la manière de traiter ces perceptions :

- Soit elles n'ont aucune sémantique pour l'agent, comme pour l'approche Animat [GM 03], où l'association de la perception à l'action est faite par apprentissage ou par évolution
- Soit l'agent possède des capacités pour reconnaître les informations comme dans C4 [BIDIB01], cela permet d'avoir des personnages ayant chacun une compréhension différente de l'environnement. Par exemple un personnage sourd ne prendra pas en compte les informations sonores, un mouton ne pourra pas faire la différence entre un loup et un chien, alors qu'un chien la fera.
- Soit c'est l'environnement qui fournit à l'agent le sens à associer à ces perceptions (comme par exemple dans les *Smart Object* [Kal01]).

1.6 Représentation des connaissances

1.6.1 Catégories de connaissances

N. Richard [Ric 01] distingue deux catégories de connaissances utilisées par un agent :

- **connaissance explicite** : c'est une connaissance symbolique et manipulable (typique de l'Intelligence Artificielle traditionnelle) ;

- **connaissance implicite** : c'est une connaissance intégrée au système, notamment au niveau des processus de décision et de perception (connaissance préprogrammée ou acquise dans un réseau de neurones par exemple).

De plus, ces connaissances peuvent être catégorisées selon la durée de leur utilité : les **connaissances permanentes** correspondent aux connaissances innées implicites ou explicites, tandis que les **connaissances temporaires** sont créées par l'agent au fur et à mesure de son expérience ; ces connaissances acquises sont généralement explicites et sont stockées dans des mémoires à court- ou moyen-terme. R. Arkin propose une liste de connaissances typiquement utiles à un agent [Ark98] :

- connaissances spatiales : comment naviguer dans le monde ?
- connaissances sur les objets : quelles sont les catégories ou les instances d'objets du monde ? Comment reconnaître ces objets et comment interagir avec eux ?
- connaissances de la perception : comment percevoir le monde ?
- connaissances comportementales : comment réagir aux différentes situations rencontrées?
- connaissances de soi : quelles sont les limites et les compétences de l'agent en termes de perception et d'action ?

La connaissance *a priori* de l'environnement permet entre autres de contraindre le système perceptif. Elle facilite en particulier l'interprétation des stimuli, en sachant ce qu'il faut s'attendre à trouver, ainsi que la focalisation de l'attention, en sachant où porter de préférence son attention. Elle pose cependant le problème de la distinction entre les connaissances acquises et les connaissances innées (préprogrammées) d'un agent. La connaissance innée impose la signification des informations perçues et des résultats probables des actions choisies sur l'environnement, et régit donc en partie le comportement d'un agent.

1.6.2 Représentation mentale de l'environnement de navigation

La représentation mentale, dite topologique, permet à un être humain de se repérer dans l'environnement. Elle passe par une phase d'abstraction, c'est-à-dire une interprétation personnelle de la géométrie des lieux. Une telle abstraction, dénommée carte cognitive spatiale, peut être vue comme un système de filtrage, où l'être humain ne va pas stocker l'ensemble de la géographie perçue, mais plutôt en extraire des lieux marquants (landmarks en anglais). Ces zones d'intérêt sont des endroits dont la localisation est mieux connue, et serviront ensuite de point de repère à d'autres lieux adjacents. Waller et al. [WLG$^+$02] indiquent que ces lieux marquants, une fois associés entre eux, forment une mémoire spatiale

impliquée dans plusieurs tâches mentales. Néanmoins, Wender et al. [WHR+03] signalent que la mémoire spatiale ne se résume pas à la simple association de lieux marquants. Par l'intermédiaire d'expériences, ils montrent que la construction d'une carte cognitive spatiale ne se fait pas en une seule étape, mais est un phénomène redondant au cours du temps. Tout d'abord l'être humain extrait des lieux marquants, puis des chemins, et enfin une organisation topographique des lieux. Ensuite, il est capable de se rappeler d'un chemin en se servant de cette carte cognitive spatiale, grâce à un mécanisme de rappel depuis la mémoire.

Afin d'agencer ces lieux marquants, R.L. Klatzky [Kla98] décrit deux sortes de repérage pour l'être humain : allocentrique et égocentrique. La méthode égocentrique organise les éléments perçus relativement à la position de l'entité qui les perçoit. A l'opposé, la méthode allocentrique organise les éléments globalement, suivant un repère absolu totalement décorrélé de tout point de vue subjective et perceptive.

Pour finir, des études spécifient que la représentation mentale de l'espace est hiérarchique. Ainsi, J. Wiener et H. Mallot [WM03] définissent la notion de niveaux de détails dans les représentations spatiales liées au processus de navigation. Ils indiquent qu'un minimum de deux niveaux hiérarchiques est utilisé, le premier étant formé d'un ensemble de places, le second de régions constituées d'agglomérats de places.

Le principe de carte cognitive spatiale est utilisé dans certains modèles informatiques, comme les IHT-graph de R. Thomas et al. [TD06]. Néanmoins, de part leur complexité calculatoire, ces modèles ne sont généralement utilisés que pour la simulation d'un nombre très limité d'humains, bien en delà de l'effectif nécessaire à une foule.

1.6.3 Planification d'un chemin

La planification de chemin (aussi dénommée navigation dans la littérature anglaise) consiste à lister les lieux par lesquels passer pour relier la position courante d'un humain à sa destination. Ce processus mental est bien entendu très fortement lié à la représentation mentale de l'espace, étant une exploitation directe des informations contenues dans la carte cognitive spatiale de l'individu.

D'autres informations peuvent tout de même intervenir dans ce processus. Par exemple, A. Tom et M. Denis [TD03] ont étudié les degrés d'utilisation des lieux marquant et des noms de rues dans la planification de chemin. Ils indiquent que, du fait de leur plus faible corrélation à la topographie, les noms de rues sont beaucoup moins utilisés.

Concernant le fonctionnement de la tâche de planification de chemin, plusieurs études montrent qu'il s'agit d'un procédé hiérarchique. Ainsi, S. Steck et H. Mallot [SM00]

indiquent que des lieux marquants locaux et globaux interviennent dans le processus, mais de façon différente suivant les personnes : certaines n'utilisent que l'un ou l'autre, ou encore les deux à la fois. De plus, J. Wiener et H. Mallot [WM03] introduisent dans leur carte mentale hiérarchique les concepts de place et région définissant deux niveaux d'abstraction topologique. Ils expliquent que la planification de chemin se fait d'abord localement au niveau des places, puis globalement au niveau des régions dès que le processus s'éloigne du lieu marquant courant de l'individu (Figure 1.6).

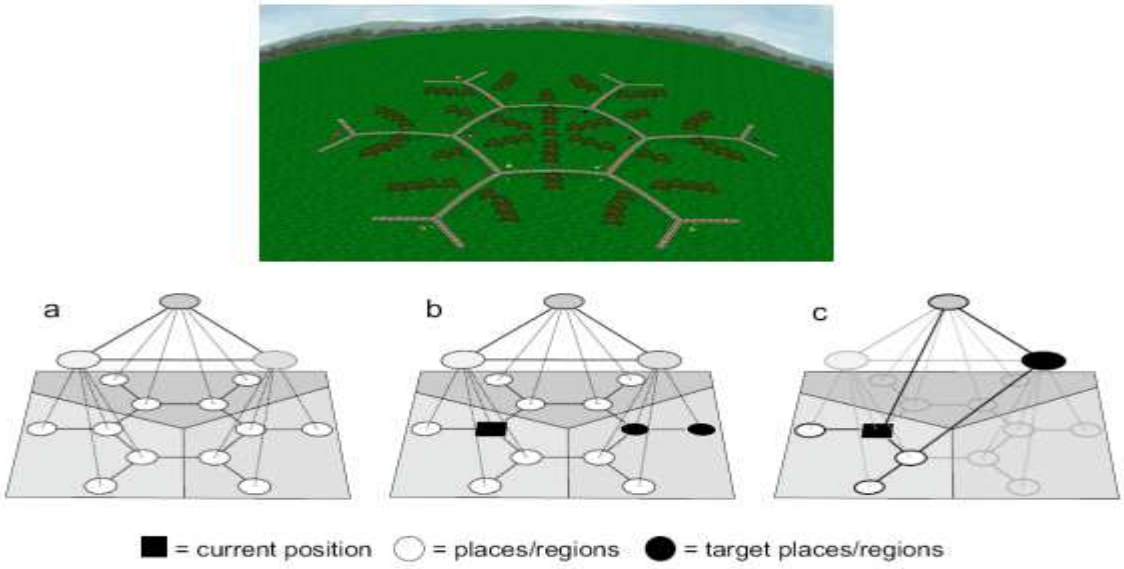


Figure 1-6 – Représentation mentale hiérarchique pour la planification de chemin selon J. Wiener et H. Mallot. En haut, l'environnement de test, et son équivalent mental en a. En b les conditions de départ (position et destinations), et en c les chemins hiérarchiques.

1.6.4 Interprétation des données et connaissance

Fondements psychologiques. J.J. Gibson est un psychologue ayant posé les fondations d'un courant écologique[2] aujourd'hui très actif en psychologie de la perception [Gib86]. Selon lui, l'environnement offre à l'agent en mouvement des significations directement utiles pour l'action, des « disponibilités physiques perceptibles », qu'il appelle « les affordances ». Le concept d'affordance qualifie une relation entre un organisme et une façon d'agir appropriée à la fois aux *opportunités offertes par l'environnement* et au *potentiel de l'acteur*. Par exemple, un objet est susceptible d'être attrapée, un endroit peut constituer une bonne cachette, un évènement peut causer un envol ou une approche, et un tiers peut être soit un ami soit un ennemi. Cette approche écologique est exploitable dans des modèles informatiques. En effet, avec une approche orientée objet, les objets perçus par l'acteur peuvent contenir des attributs

[2] Étude des êtres vivants au sein des milieux où ils vivent, ainsi que leur interaction avec ces milieux.

et des méthodes directement utilisables comme paramètres des actions : ce sont les *opportunites offertes par l'environnement*.

1.6.5 Techniques de représentation des connaissances

Cette représentation peut être créée par le module de la perception, ou directement récupérée de l'environnement s'il est informé. Pour cela, plusieurs techniques peuvent être utilisées.

1.6.5.1 Arbre de perception

Lorsque l'environnement n'est pas informé, il faut analyser les informations provenant des capteurs pour lui trouver un sens. Dans C4, l'équipe de Blumberg utilise un arbre de perception (PerceptTree) pour représenter une information [BID$^+$01] : un percept est une classification atomique qui modélise certains aspects des informations provenant du système de perception. Ces percepts sont organisés en hiérarchie : lorsqu'une information arrive du système de perception, elle passe à travers l'arbre de percepts. Les percepts qui reconnaissent l'information enregistrent leur confiance dans l'information et les données extraites dans un PerceptMemory (voir fig. 1.7). Ce PerceptMemory est ensuite comparé à ceux contenus dans la mémoire de travail pour savoir s'il s'agit bien d'une nouvelle information.

Ce système est utilisé pour trois raisons :

- Faire des erreurs honnêtes, une créature qui peut se tromper lorsqu'elle perçoit peut être plus crédible,
- Apprendre des concepts généralisables,
- Utiliser les mêmes informations pour différentes créatures, mais avec des sens éventuellement différents.

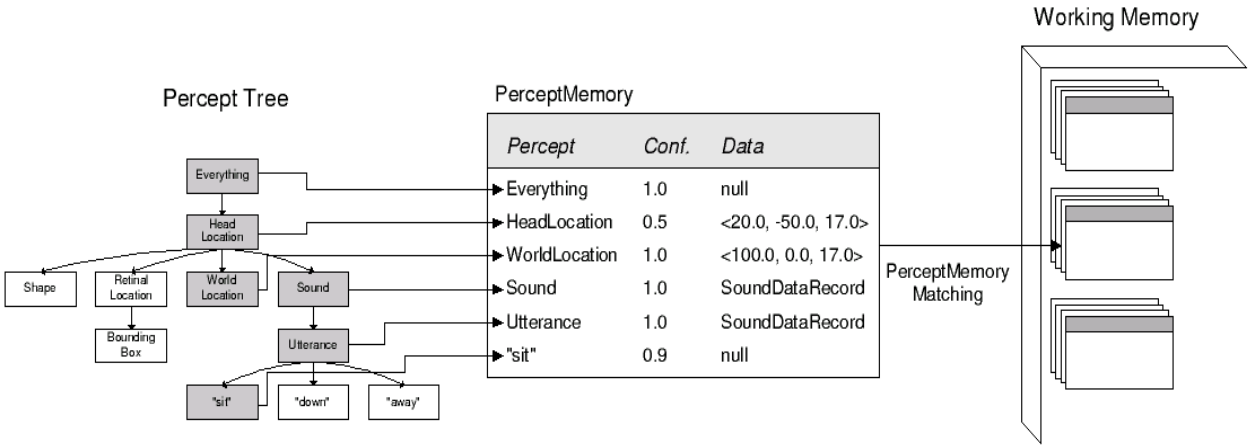


Figure 1-7 – Formation d'un *PerceptMemory* [BID$^+$01].

1.6.5.2 Carte de l'environnement

L'environnement en tant qu'espace peut aussi être représenté dans les agents, sous la forme d'une « carte ». Ainsi D. Isla et B. Blumberg [IB02] rajoutent dans C4 la carte probabiliste d'occupation. Cette carte permet d'estimer où se trouvent les objets mobiles présents dans l'environnement. Cette carte est mise à jour par les informations positives ou négatives fournies par le système perceptif. Un système de diffusion de la probabilité permet de prendre en compte le temps pour mettre à jour la confiance dans la position de l'objet mobile. R. Thomas [Tho05] utilise des cartes cognitives spatiales pour ces agents : l'agent se déplace dans un environnement modélisé sous la forme de carte topographique. La carte cognitive spatiale de l'agent est un filtre sur cette carte topographique (Figure 1.8). Elle donne accès aux objets qui ont déjà été vus et rend invisible les autres. Ces agents sont utilisés dans le cadre de simulations de navigation de piétons.

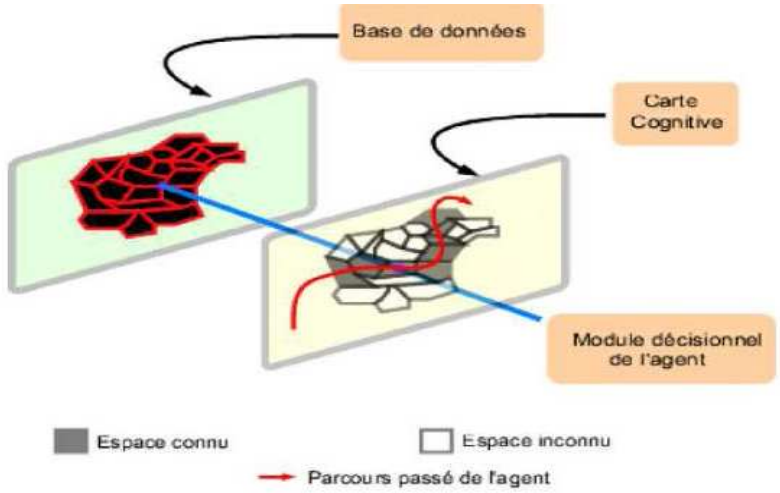


Figure 1-8 – La carte cognitive spatiale comme filtre dynamique de la carte topographique.

1.6.5.3 Représentation par fluents

Dans l'architecture utilisée par J. Funge [Fun98], CML, les informations provenant de l'environnement sont transformées en fluents[3]. A ces fluents sont associés des intervalles[4] qui permettent de prendre en compte l'aspect incertain sur la valeur des fluents (voir figure 1.9). L'intervalle augmente de plus en plus si le fluent n'est pas perçu de nouveau. Percevoir permet de rendre ces intervalles plus petits. La taille de l'intervalle permet aussi de savoir si la valeur d'un fluent est connue ou non, et s'il est réellement utilisable (un intervalle trop grand

[3] Un fluent est une condition qui change au cours du temps. Elle peut être représentée par des prédicats de la logique du premier ordre ou par des fonctions - par exemple on(box,table,t) indique que la boîte est sur la table au moment t.

[4] Par exemple l'intervalle Ispeed(Si) = (10, 50) indique que l'objet à une vitesse qui est connue comme étant comprise entre 10 et 50 au moment Si.

ne donne plus aucune indication). On peut aussi inférer sur la taille de l'intervalle de manière à proposer des actions de perception au bon moment.

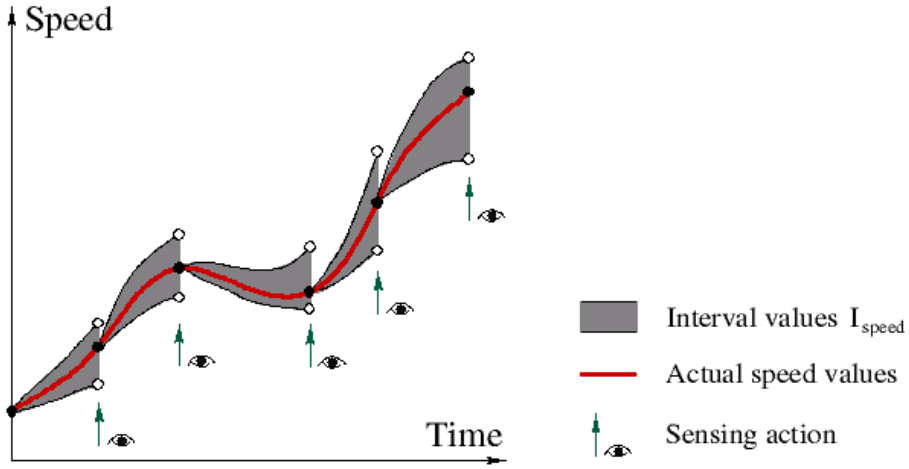


Figure 1-9 – Intervalle pressenti d'un fluent entourant la valeur réelle [Fun98].

1.7 Conclusion

Un agent est une entité autonome qui sait percevoir son environnement. Cela lui permet de prendre une décision concernant le comportement qu'il va devoir adopter. Si l'agent sait planifier, on parle d'agent cognitif. Si la décision est une simple réponse réflexe, on parle d'agent réactif. Une fois leur décision prise, les agents agissent sur leur environnement. L'action menée a bien, ils sont amenés à nouveau à réévaluer leur environnement pour savoir si les modifications qu'ils ont engendrées sur leur environnement sont conformes aux modifications attendues et connaître le nouvel état de l'environnement qui les entourent.

Nous avons pu nous familiariser dans cette section avec les deux catégories de modélisation du comportement. D'une part les comportements réactifs, relativement simples et à court terme, et d'autre part les comportements rationnels, plus complexes et régissant le plan d'action des individus sur le long terme.

Nous pouvons conclure que les modèles basés règles et automates, pour la gestion du comportement réactif, sont directement exploitables. Nous avons pourtant évoqué le bénéfice de les coupler, afin de profiter du potentiel d'évolution de l'un ainsi que de la robustesse de l'autre.

La plupart des modèles de comportement rationnel semblent quant à eux difficilement exploitables, surtout de par leur complexité de résolution. De plus, leur potentiel d'inférence du savoir-faire n'est pas réellement indispensable à la simulation des circulations de personnes. L'un d'entre eux, le modèle BDI, semble tout de même approprié car il offre une approche plus descriptive du comportement. Nous devons tout de même mettre l'accent sur le

manque de lien avec la topologie, évidemment très important pour l'étude des circulations de personnes. Un autre point désavantageux concerne la faible autonomie des entités simulées, d'autant plus bridée que les plans d'actions sont scriptés.

La définition des comportements consiste à définir le raisonnement et les actions des acteurs en fonction de leur environnement et de la perception qu'ils en ont. L'acteur voit ce qui se passe autour de lui, décide de son action en fonction de sa perception et de ses connaissances, agit sur lui-même (par exemple en apprenant une nouvelle notion) et sur son environnement (par exemple en ouvrant une porte). Il est donc en interaction constante avec son environnement par le biais de capteurs sensoriels et d'effecteurs agissant directement sur l'environnement. La simulation d'humanoïdes autonomes dotés de comportements sophistiqués passe par la prise en compte de l'environnement.

Chapitre 2

Environnement virtuel

2 Environnement virtuel

2.1 Introduction

Les deux notions, comportement et environnement, sont intimement liées. En effet, le comportement se résume à l'interaction d'un organisme avec le milieu dans lequel il évolue, son environnement, lequel pouvant être composé aussi bien d'objets que d'autres organismes. L'organisme humain, par exemple, est constamment dans une boucle d'interaction avec son environnement. Il peut soit utiliser ses capteurs (yeux, nez, oreilles, etc.) pour acquérir des données, soit ses effecteurs (mains, pieds, etc.) pour produire un effet. De ce fait, il est nécessaire de définir ce qu'est l'environnement pour pouvoir étudier le comportement section 2.2.

L'être humain doit prendre en compte beaucoup de paramètre lors de son interaction avec son environnement. Prenant comme exemple son déplacement, avant de pouvoir le faire, il a besoin de se représenter cet environnement. Cette représentation mentale va en effet lui servir à raisonner sur cet environnement, que ce soit pour évaluer un chemin afin d'atteindre une localisation précise, ou encore pour organiser les tâches qu'il doit effectuer (comme les interactions avec des objets). C'est à la représentation topologique permettant le premier type de raisonnement – la navigation – que nous allons nous attacher en premier lieu, section 2.5, pour revenir plus tard dans le chapitre suivant sur le second type de représentation qui est la représentation sémantique.

La représentation topologique est ensuite exploitée par des algorithmes de recherche de chemin. Ceux-ci peuvent se baser sur différentes informations issues de l'environnement, qu'elles soient géométriques comme les distances, ou qualitatives comme la charge cognitive associée au chemin et c'est ce qu'on va voir dans la section 2.6.

2.2 Environnement Virtuel

Comme nous l'avons vu dans le chapitre1 un agent autonome doit exister dans un environnement virtuel. Ce terme d'Environnement Virtuel, soit *virtual environment* en anglais fut introduit au début des années 90 par les chercheurs du MIT.

Un environnement virtuel peut être vu comme un environnement synthétique, dans le sens d'un univers réel ou imaginaire simulé par ordinateur. Des informations sensorielles complémentaires (sonores, tactiles, ...) peuvent venir enrichir ce modèle. Pour Slater [Sla94], cet environnement virtuel devient immersif lorsque la représentation virtuelle du corps de l'utilisateur (tout ou partie) est affichée dans l'environnement virtuel, et qu'il peut réaliser un

certain nombre d'actions sur cet environnement (à l'aide de capteurs de position et d'orientation, de gants de données, ...).

Un "monde virtuel" est un analogue ordinateur d'un environnement physique. Généralement, un utilisateur est incarné dans le monde, et peut utiliser cette entité (appelée avatar) pour se déplacer dans le monde et d'interagir avec son contenu. La plupart de ces environnements soutiennent de nombreux utilisateurs simultanément, de sorte qu'ils ne sont pas seulement des espaces physiques, mais sociaux. C'est la définition donnée par P. Doyle [Doy04].

Dans le cadre des SMA, l'environnement est l'ensemble des conditions extérieures susceptibles d'agir sur le fonctionnement d'un système. Un agent autonome évolue de façon continue dans un environnement, dans lequel peuvent exister d'autres agents. Puisqu'un agent agit sur son environnement et que l'environnement agit sur l'agent, il existe une réelle interaction entre ces deux entités.

Du point de vue d'un agent, l'environnement correspond à «tout ce qui lui est extérieur»: les autres agents font aussi *a priori* partie de l'environnement [Ric01].

En revenant au dictionnaire, nous constatons que la définition du mot virtuel, en termes de l'informatique, c'est: ce qui n'est pas physiquement existant en tant que telle, mais fait par un logiciel pour paraître à le faire. Quand à la définition informatique de l'environnement, c'est: l'ensemble de la structure dans laquelle un utilisateur, un ordinateur, ou un programme évolue.

Passant à la définition globale de mot environnement, l'environnement est l'ensemble des conditions naturelles (physiques, chimiques, biologiques) et culturelles (sociologiques) susceptibles d'agir sur les organismes vivants et les activités humaines ce qui nous amène à cette définition:

Un environnement virtuel est un environnement généré par ordinateur, composé d'objets, dans lequel des agents autonomes évoluent.

Figure 2-1 – Environnements Virtuels.

2.3 Utilisations des environnements virtuels

Parmi les principales utilisations des environnements virtuels, nous pouvons citer :

- Les outils pédagogiques utilisant la simulation interactive, intégrant éventuellement des tuteurs personnalisés artificiels comme STEVE;
- La téléopération, qui permet de contrôler un robot à distance : le monde virtuel est alors une reconstitution de l'environnement réel ayant pour but de faciliter la manipulation du robot;
- La simulation scientifique, permettant principalement l'observation de phénomènes physiques (tels que les collisions de particules) ou éthologiques (tels que les colonies de fourmis ou les sociétés de grands singes), et parfois l'interaction, par exemple dans le cas de la manipulation de composés chimiques;
- Le grand domaine du divertissement, comprenant entre autres :

- Les jeux vidéos, en particulier les jeux faisant intervenir des entités autonomes, telles que des «animaux de compagnie virtuels» comme dans CREATURES ou des humains virtuels comme dans THE SIMS;
- la création dynamique d'histoires dans des mondes virtuels (*virtual storytelling*);
- les effets spéciaux pour le cinéma (mouvements de foules, fumée, *etc.*);

- ou encore le prototypage virtuel, la création artistique et la thérapie.

2.4 Construction des environnements virtuels

Les outils traditionnels de construction d'environnements virtuels 3D sont les modeleurs géométriques. Les modeleurs 3D permettent de construire des objets géométriques et de les intégrer dans une scène. Dans ces modeleurs, la création passe par la manipulation de primitives graphiques en 3 dimensions : points, entités filaires (lignes et arcs), surfaciques (polygones, surfaces splines) et volumiques (simples et complexes). Le réalisme des objets et des scènes est finalement obtenu en leur associant des matériaux et des textures (photographies).

La tendance chez les producteurs de modeleurs, est d'intégrer des outils de modélisation, d'animation et de rendu. Des systèmes complets sont donc disponibles pour créer et évoluer dans des scènes telles les villes virtuelles.

2.5 La modélisation d'environnements virtuels

Pour permettre aux personnages artificiels d'interagir avec leur environnement virtuel, une représentation appropriée de l'environnement est indispensable dans le but de supporter efficacement la perception, la navigation ou la planification de chemin.

2.5.1 La modélisation

Un modèle est initialement une « maquette, réduction, reproduction d'un objet sous une forme simplifiée pour être soumis à des mesures, des calculs, des tests physiques qui ne pourraient être appliqués commodément à l'objet lui-même (…) » (Encyclopédia Universalis, p 121 ; cité par [Lep03]. Par une extension récente, est aussi considéré comme un modèle toute « figurations ou reproductions qui servent les buts de la connaissance » (Encyclopédia Universalis, p 121 ; cité dans [Lep03]).

Au delà du détail de cette définition, trois éléments nous apparaissent importants à souligner :

1. Un modèle est d'abord une **simplification** : modéliser consiste à retenir seulement certaines caractéristiques « pertinentes » de l'objet ou de la situation figurée, et rend explicite certaines des propriétés et relations reliant ces caractéristiques. Le fait qu'un modèle ne soit pas « complet » n'est par conséquent pas une critique recevable ; la question cruciale concerne plutôt le choix des sous-ensembles de caractéristiques représentées et non représentées.
2. Tout modèle est **exprimé au moyen d'une notation** plus ou moins formelle, en d'autres termes un code. Par exemple, il peut s'agir d'une expression mathématique, d'un graphe, de diagrammes, de boîtes et de flèches, voire d'une collection de descriptions littéraires. Plus généralement pour tout objet considéré et son modèle, n'importe quelle notation ou formalisme de représentation peut a priori être utilisé. Il en découle, entre autres conséquences, que l'objet ne partage ni les propriétés ni la nature du formalisme de notation exploité pour le figurer, quelle que soit la discipline d'emprunt et l'épistémologie dudit formalisme. De ce fait, utiliser une expression mathématique pour décrire le raisonnement ne présuppose pas que le raisonnement soit mathématique. De même, l'usage de représentation des connaissances sous forme de réseaux de neurones ou de règles de production ne présuppose pas l'existence concrète de telles structures dans la tête du sujet humain. Cependant, certains formalismes ont probablement des relations plus étroites que d'autres avec l'objet modélisé : il peut s'agir d'une analogie ou alors de propriétés véritablement communes.
3. Enfin, **un modèle est un outil**. Il est conçu pour servir à quelqu'un, de manière à faire quelque chose. Un modèle n'est donc ni absolu ni unique. Au contraire, il y a une infinie variété de modèles potentiels pour un seul objet donné. L'usage et le type d'utilisateur influence les critères et les objectifs concrets à atteindre en termes de modélisation.

2.6 Modèles existants

Lorsque les techniques les plus courantes de modélisation d'environnement existant été développé pour la communauté des robots et pour leur navigation, de nouveaux modèles ont aussi apparu dans le domaine de l'animation par ordinateur avec l'intérêt croissant aux agents autonomes et l'animation comportementale. Nous allons faire un bref survol des différentes techniques dans le reste de cette section.

2.6.1 Représentations sous forme de carte en robotique

La grille d'occupation [Elf87] représente l'environnement avec une grille de résolution fixe. Cette grille multidimensionnelle maintient l'état d'occupation de chaque cellule. Ce modèle a été largement utilisé depuis son introduction en raison de sa représentation simple mais riche, mais elle a deux inconvénients majeurs:

1. La grille peut difficilement saisir les relations topologiques entre les régions;
2. Elle souffre généralement de coût élevé dans le temps et l'espace Si les grilles fines sont utilisées.

Une autre approche est *La carte fonctionnelle* [LD91], dans laquelle l'environnement est représenté avec des fonctionnalités paramétriques tels que des points, lignes, cylindres, coins…etc, et est complété par des informations sur les caractéristiques telles que la position, la géométrie, la couleur, etc. c'est un modèle tout à fait applicable pour le traitement visuel des données sensorielles. Toutefois, une telle technique n'est pas utile pour un environnement non structuré, où il n'est pas toujours possible de trouver des géométries clairement distinctes.

Les approches topologiques [KB91] utilisent les graphes pour représenter l'environnement. Les nœuds et les arcs du graphe indiquent les relations entre les régions, tels que la contiguïté et la connectivité. D'une part, cette abstraction compacte facilite certaines tâches, telles que la planification de chemin, mais d'autre part, il ne prend pas en charge la navigation détaillée en raison de son manque d'information métriques, telles que la position absolue.

Étant donné les avantages et les inconvénients des différentes approches, les chercheurs ont commencé à combiner différentes techniques pour former des cartes hybrides [TB96]. Les modèles combinés peuvent bénéficier de toutes les techniques utilisées et sont en mesure de démontrer un rendement satisfaisant.

2.6.2 Modèles d'environnement en animation

Dans le domaine de l'animation par ordinateur, comme les personnages de synthèse sont de plus en plus sophistiqués, la modélisation du monde qui les entoure a attiré l'attention des chercheurs progressivement. Ainsi, il est devenu de plus en plus évident que, dans ce contexte, un modèle de l'environnement doit aller bien au-delà de la représentation des cartes. Il doit également faciliter tous les processus de perception, d'interprétation, et l'interaction, ce qui en fait un problème de recherche difficile. Nous allons faire, dans la section qui suit, un tour d'horizon des théories liées à la représentation d'environnement.

2.7 Environnement de déplacement

La capacité à se déplacer est l'un des comportements élémentaires les plus indispensables pour un individu. C'est pourquoi ce sujet est l'un des plus anciens concernant la simulation de personnes. Il dispose ainsi d'un grand nombre de modèles et d'acquis, mais reste tout de même très ouvert encore aujourd'hui.

La représentation de l'environnement de déplacement est constituée de deux parties. La première doit être une simplification et restructuration des données décrivant à l'origine le terrain. Cette simplification est nécessaire afin de réduire le coût de calcul du modèle de navigation, en enlevant les données que le modèle n'utilisera pas. L'autre partie est la représentation du voisinage de l'humanoïde constitué des autres humanoïdes.

2.7.1 Représentation topologique d'environnement

Une description topologique de l'environnement permet de représenter et d'organiser l'environnement de simulation. Cette description 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, domaine où cette représentation tient un rôle prépondérant.

2.7.1.1 Représentations exactes de l'environnement

Les représentations exactes de l'environnement vont chercher à organiser les données spatiales tout en conservant intégralement les informations qu'elles contiennent à l'origine. La méthode utilisée consiste généralement à découper l'espace navigable en cellules convexes de différentes formes. Il existe plusieurs modèles pour effectuer cette subdivision, mais nous nous attacherons ici à la plus utilisée qu'est la triangulation de Delaunay, ainsi qu'à ses évolutions.

Triangulation de Delaunay

La triangulation de Delaunay [Boi98] crée un ensemble de triangles en réunissant des points fournis en entrée. L'algorithme d'unification respecte pour contrainte que le cercle circonscrit à un triangle ne contienne aucun autre point que les trois sommets qui le composent. Une conséquence de cette propriété est que l'angle minimum d'un triangle produit est maximisé.

La propriété la plus intéressante de cette triangulation 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 [LAM 04], pour calculer un graphe de voisinage entre les entités mobiles, qui serviront cette fois-ci de point d'entrée.

Triangulation de Delaunay contrainte

La triangulation de Delaunay contrainte [Che87] permet de modérer les contraintes de la version standard afin de conserver certaines arêtes de la définition graphique d'origine. Pour se faire, la contrainte du cercle circonscrit à un triangle est modifiée, pour spécifier que tout point inclus dans ce cercle ne peut être relié à tous les points du triangle sans intersecter un segment contraint.

Cette triangulation étend la propriété du plus proche voisin en permettant d'y associer une notion de visibilité. Si l'on considère que les arêtes contraintes coupent la visibilité d'un point à l'autre, la propriété de plus proche voisin devient : chaque point est relié à son plus proche voisin visible.

La triangulation de Delaunay contrainte est aussi utilisée pour obtenir une subdivision spatiale en triangles [KBT03], en introduisant les obstacles à la navigation sous la forme d'autant de segments contraints (Figure 2.2). Il a été prouvé que le nombre de triangles produits lors d'une telle subdivision est 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.

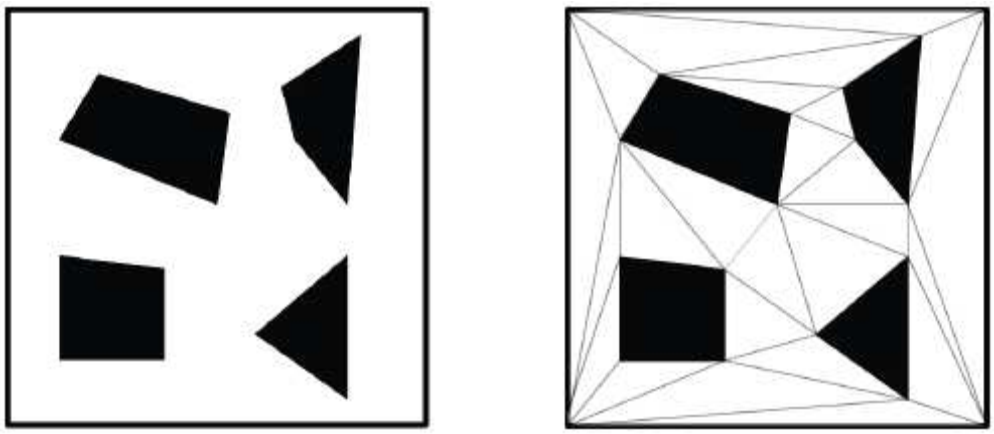


Figure 2-2 – Exemple de triangulation de Delaunay contrainte.

Triangulation de Delaunay filtrée

La triangulation de Delaunay filtrée [LD04] 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.

Dans [LD04], F. Lamarche et S. Donikian proposent ce type de décomposition, afin d'optimiser l'organisation de l'espace. Le premier point est de ramener un environnement 3D en un environnement 2D par projection sur un plan 2D. Grâce à une triangulation de Delaunay, ils subdivisent l'espace une première fois en cellules triangles. Une caractéristique importante pour la navigation d'humanoïdes est la présence de goulots d'étranglement dans l'environnement ainsi que leurs positions. Pour connaître ces emplacements, il faut calculer la distance minimale entre les coins et les murs, ils vont ainsi avoir une carte 2D de l'environnement avec une subdivision en cellules efficace comme il est montré sur la figure 2.3.

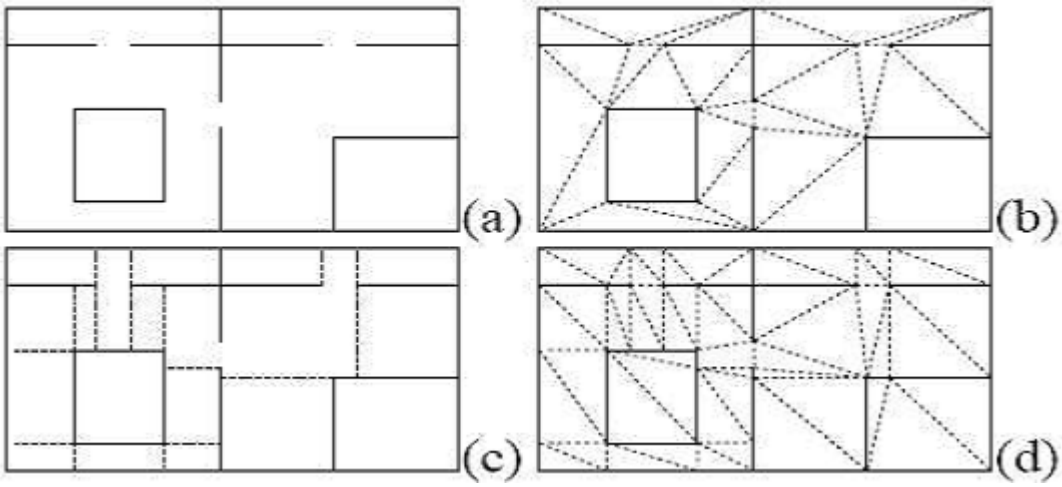


Figure 2-3 – Construction de l'environnement : (a) Carte 2D. (b) Triangulation de Delaunay. (c) Distances minimaux coins murs. (d) Triangulation et distances minimales coins murs [LD04].

Pour simplifier cette subdivision et diminuer le nombre de cellules triangles en un nombre moins élevé de cellules convexes, un algorithme de fusion de triangles est utilisé ce qui conserve les informations sur les goulots d'étranglement. Ils peuvent ainsi extraire de cette carte un graphe de connexité relativement simplifié (Figure 2.4), chaque état étant une cellule convexe, et chaque transition représentant la connexité entre cellules.

Ils peuvent alors déduire simplement de ce graphe la topologie du terrain. Ainsi, un nœud d'arité 0 correspond à une cellule isolée, une noeud d'arité 1 correspond à une voie sans issue, un noeud d'arité 2 est un couloir, et une arité supérieure indique un croisement. Cette topologie abstraite simplifie alors la planification de chemin (on peut par exemple voir plusieurs "couloirs" connectés comme un seul passage).

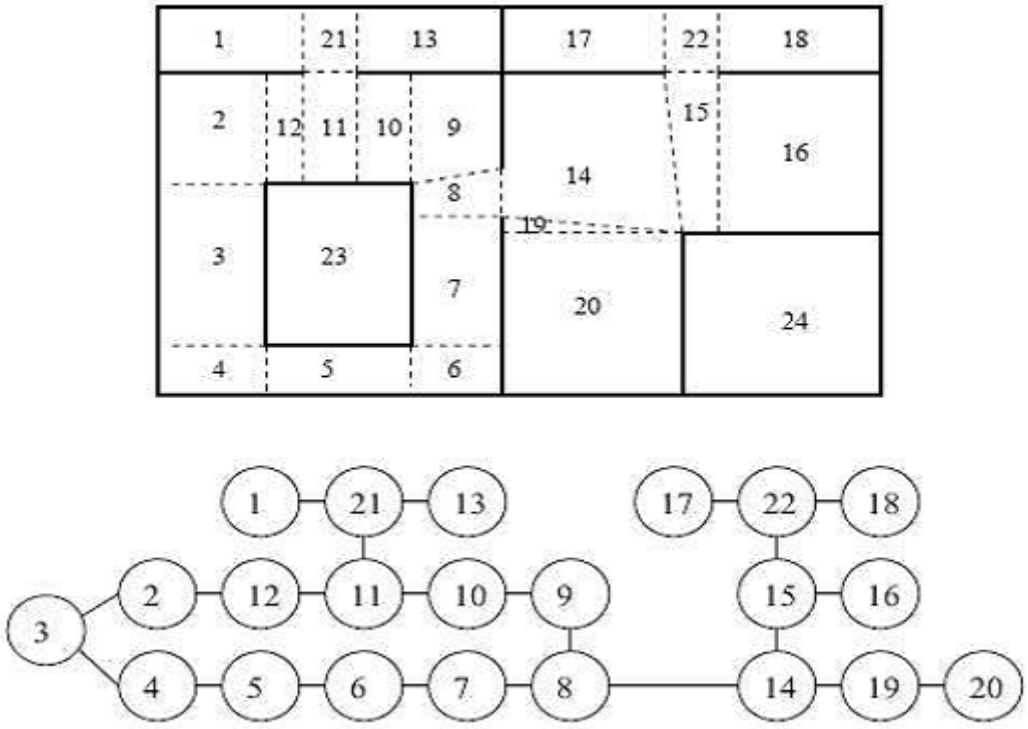


Figure 2-4 – Carte simplifiée et graphe associé [LD04].

Dans [PDB 05], S. Paris, S. Donikian et N. Bonvallet reprennent ces travaux et proposent des améliorations concernant l'abstraction topologique. Ils ont ainsi introduit un nouveau type de cellules : les cellules virtuelles représentant les limites de l'environnement. Ainsi, une cellule virtuelle connectée à une cellule "réelle" sera un point d'entrée/sortie de l'environnement.

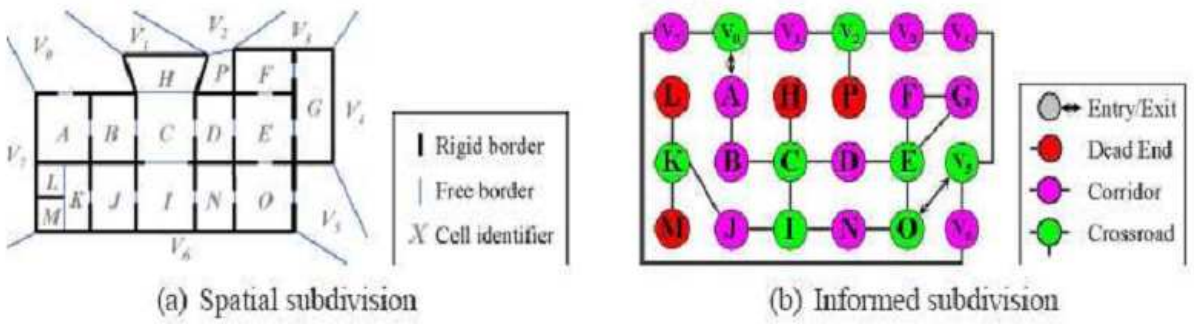


Figure 2-5 – Division spatiale et graphe associé [PDB05].

Cependant, la prolifération des cellules en environnement complexe peut mener à un coût de calcul excessif dans la planification de chemin. Ils ont alors introduit deux niveaux d'abstraction, afin de simplifier le graphe de connexité. Cette abstraction est réalisée par des fusions successives de cellules. Par exemple, tous les couloirs connexes sont groupés en un seul couloir, ou encore, un couloir et une voie sans issues connexes deviennent un cul de sac. L'utilisation de seulement deux abstractions successives peut mener à des abstractions incomplètes dans certains cas, cependant elle réduit de façon significative le coût de calcul lors de la planification de chemins.

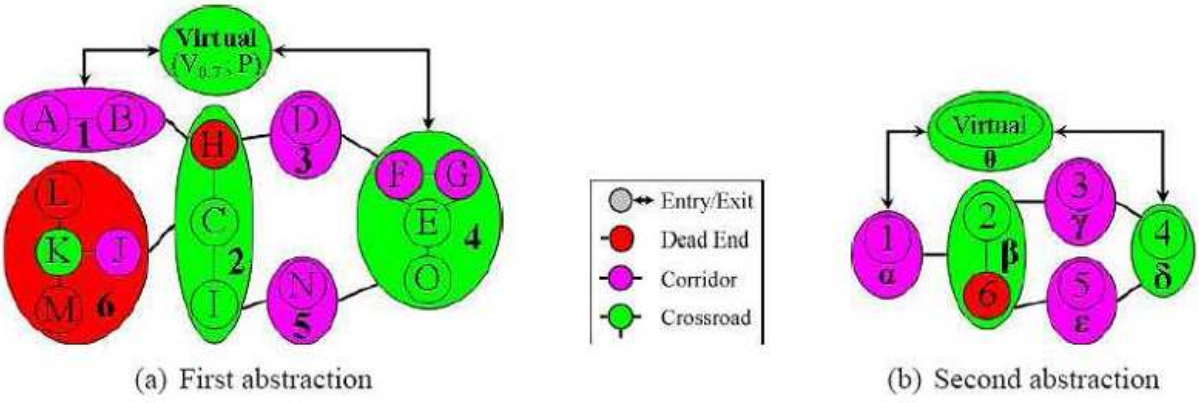


Figure 2-6 – Abstractions successives [PDB05].

2.7.1.2 Représentations approximatives de l'environnement

Les représentations approximatives de l'environnement sont les plus utilisées en animation comportementale, du fait de leur facilité de mise en œuvre. Ces représentations vont décrire l'espace libre – de navigation – avec des formes géométriques simples telles que des carrés ou des segments. Deux modèles entrent dans cette catégorie : les modèles à base de grilles, et les cartes de cheminement.

Modèles à base de grilles

Le premier modèle de représentation approximative utilise des grilles régulières, formées de cellules carrées en deux dimensions (Figure 2.7 (a)) et cubiques en trois dimensions. L'environnement est donc pavé par ces cellules, qui peuvent avoir trois états : libre, partiellement obstruée, et obstacle.

La précision de la représentation obtenue dépend directement de la taille des cellules utilisées: plus les cellules sont grandes, moins la représentation est précise. Bien entendu, l'augmentation de la précision induit une augmentation proportionnelle de l'occupation mémoire. L'occupation mémoire de cette méthode constitue ainsi son premier point faible, se répercutant directement sur la complexité de recherche de chemin à l'intérieur de l'environnement [TB96].

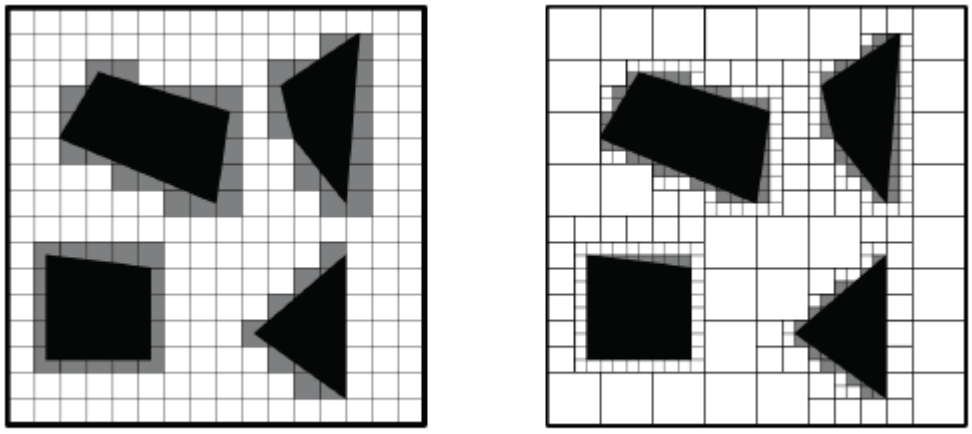


Figure 2-7 – Exemple de grilles régulières.

Pour réduire ce problème, une évolution de ce modèle est apparue sous la forme des grilles hiérarchiques [Sha06]. Cette méthode décrit l'espace navigable par une succession de grilles de plus en plus précises, organisées sous forme d'arbre. L'algorithme de construction va commencer par paver l'environnement avec les cases les moins précises, puis découper récursivement les cases partiellement obstruées, en quatre parties pour la 2D (formation d'un quadtree – Figure 2.7(b)), ou en huit pour la 3D (formation d'un octree). L'algorithme arrête les subdivisions dès qu'une précision fixée est atteinte, ou si la case produite n'est plus partiellement obstruée. Cette méthode est donc d'autant plus avantageuse que l'environnement est peu dense en obstacles.

Les méthodes à base de grilles sont fortement utilisées en animation comportementale du fait de leur simplicité de mise en œuvre et de leur rapidité d'exploitation. On peut ainsi voir des animations de milliers d'individus [TLC02] basées sur ce mode de représentation (Figure 2.8).

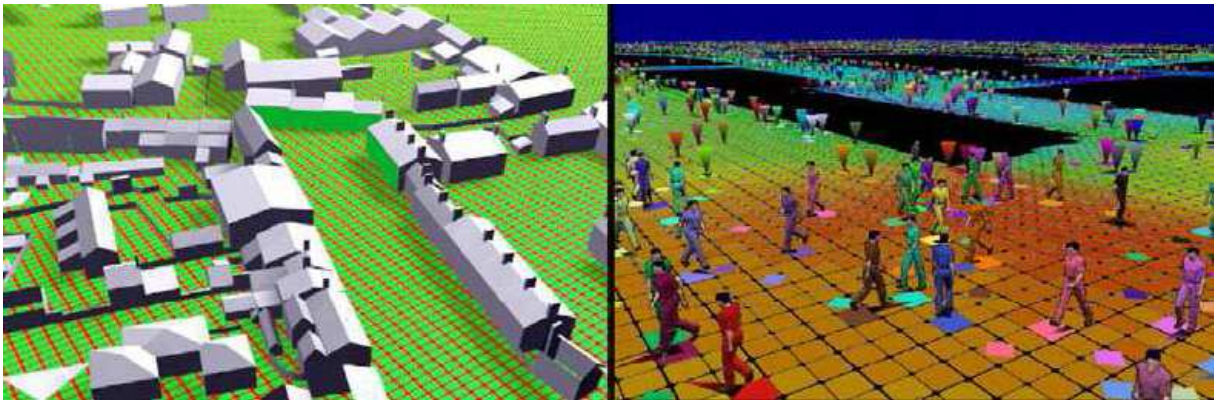


Figure 2-8 – Utilisation de grille en animation.

Les grilles sont très difficilement généralisables à des environnements quelconques, notamment s'ils comportent des obstacles non alignés sur les axes, et dans un cadre de simulation où l'individu devra se déplacer finement. Néanmoins, la rapidité d'accès aux informations qu'elle produit fait de cette méthode un complément envisageable à une approche plus fine de représentation de l'environnement. Pour finir, il faut remarquer que le niveau de discrétisation de la grille introduit implicitement une limite aux densités de populations que l'on peut simuler [ABG05]. Cela rend aussi le déplacement des entités discret, et les contraint généralement à avoir une taille standardisée.

Une présentation de la construction de ces grilles à partir de grilles uniformes est donnée dans la thèse de doctorat de W. Shao [Sha06].

Modèle de SHAO WEI

w. Shao [Sha06] représente l'environnement virtuel par une collection hiérarchique des cartes. Comme illustré dans la figure. 2.9, le modèle comprend :

(i) une carte topologique qui représente la structure topologique entre les différentes parties du monde virtuel. Liées à cette carte,

(ii) des cartes de perception, qui fournissent des informations pertinentes aux requêtes de perception, et

(iii) des cartes de cheminement, qui permettent la planification des chemins en ligne pour la navigation. Enfin, au niveau le plus bas, il y a

(iv) les objets spécialisés qui prennent en charge les requêtes de perception.

Dans la carte topologique, les nœuds correspondent à des régions de l'environnement et les arêtes représentent l'accessibilité entre les régions. Une région est un volume limité dans l'espace 3D (comme une salle, un couloir, un escalier ou même un étage entier) ainsi que tous les objets à l'intérieur de ce volume (par exemple, sol, murs, bancs). La représentation

suppose que la surface navigable dans une région peut être représentée par un plan horizontal sans perte d'information géométrique nécessaire. Par conséquent, l'espace 3D peut être adéquatement représenté au sein de la carte topologique 2D par plusieurs cartes planaires, améliorant ainsi la simplicité et l'efficacité des requêtes sur l'environnement.

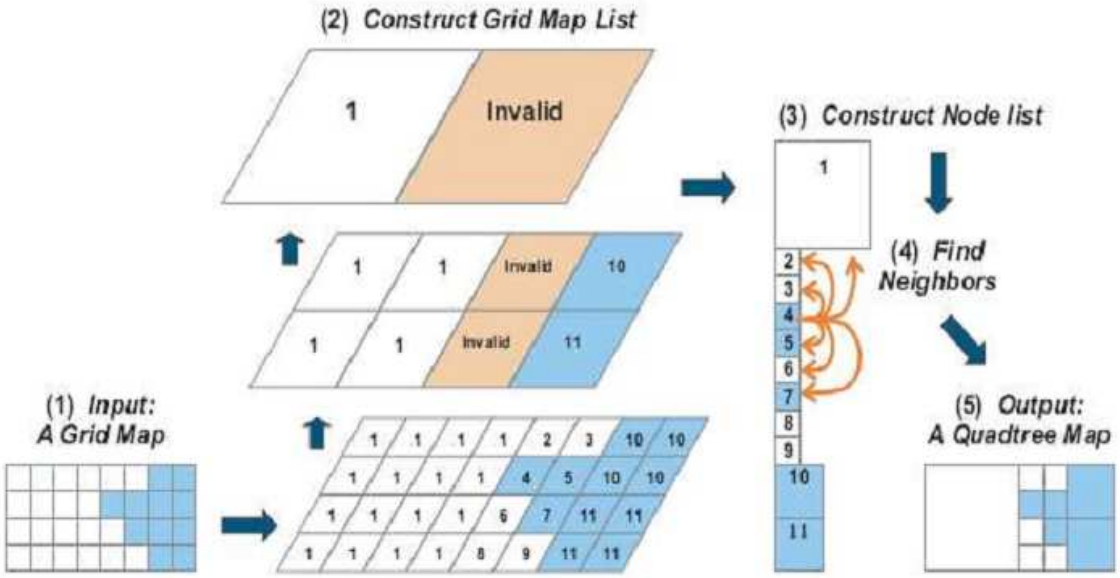


Figure 2-9 – Construction d'un quadtree représentant l'espace [Sha06].

Les cartes de perception comprennent des grilles qui représentent des objets statiques de l'environnement à l'échelle locale, par régions, ainsi qu'une carte globale qui assure le suivi d'objets mobiles, généralement les piétons virtuels. Chaque cellule de la grille des objets statiques, dont la taille des cellules est typiquement de 0,2 ~ 0,3 mètres, stocke des informations permettant d'identifier tous les objets qui occupent les cellule de la zone. Chaque cellule de la grille de la carte des objets mobiles stock et met à jour des identificateurs de tous les agents actuellement dans sa zone cellulaire. Ainsi, elle sert simplement à identifier les agents à proximité, plutôt que de déterminer leur position exacte, elle emploie des cellules dont la taille est en rapport avec la plage de détection visuels d'un piéton (actuellement fixé à 5 mètres).

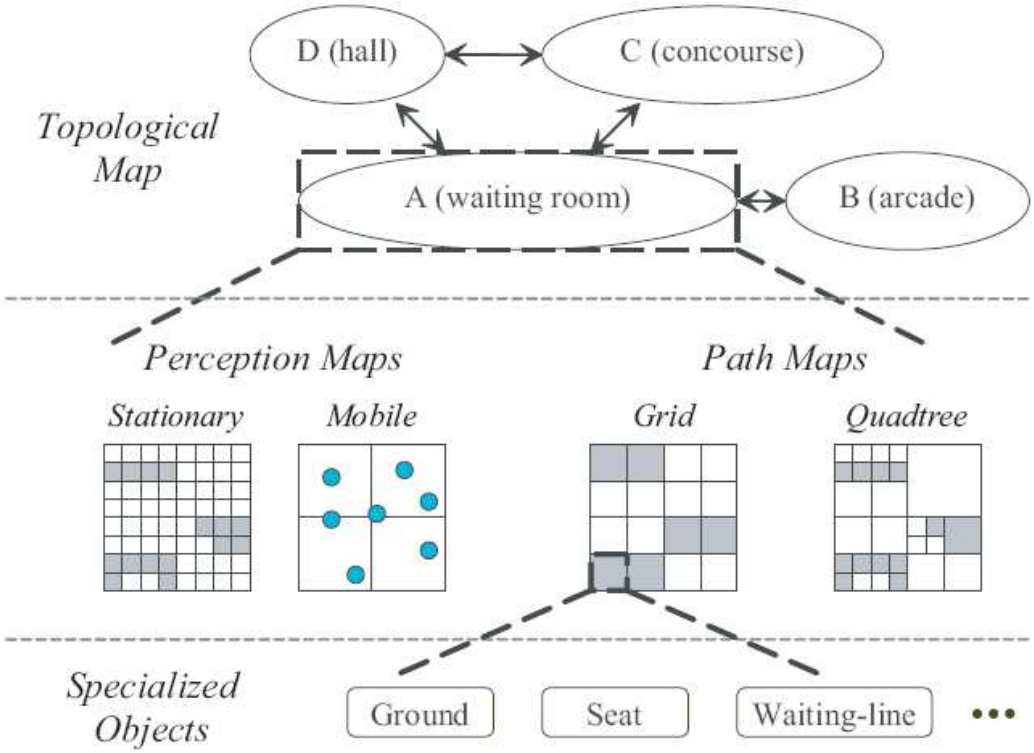


Figure 2-10 – Hierarchical environment model [Shao06].

Les cartes de cheminement comprennent une quadtree qui prend en charge d'une manière globale, la planification de chemin, et une grille qui prend en charge la planification de chemin à courte portée. Chaque noeud de la carte stocke des informations sur son niveau dans le quadtree, la position de la zone couverte par le nœud, le type d'occupation (sol, obstacle, siège, etc.), et des pointeurs vers les nœuds voisins, ainsi que des informations à utiliser dans la planification de chemin, comme une variable de distance (par exemple, combien mesure la distance entre un nœud et un point de départ donné) et un facteur de congestion (la partie de la zone du nœud qui est occupé par les piétons). La carte quadtree prend en charge l'exécution de plusieurs variantes de l'algorithme A*, qui sont utilisés pour calculer les chemins quasi-optimale vers les objectifs souhaités. Shao a montré que la carte quadtree est utilisé pour la planification d'environ 94% des chemins. Les 6% restants des chemins sont calculés, à partir de la grille, qui prend également en charge l'exécution d'un algorithme A* en fournissant des les chemins détaillés, de courte portée avec la présence d'obstacles, lorsque c'est nécessaire. Un exemple typique de son utilisation, c'est quand un piéton se trouve derrière une chaise ou un banc et doit naviguer autour de lui afin de s'asseoir.

Les objets spécialisés au niveau le plus bas de la hiérarchie de l'environnement, sont en mesure de fournir des réponses aux requêtes qui ne peuvent pas être traitées directement par

les cartes de perception. Ils le rendent facile pour les contrôleurs de comportement pour obtenir des informations de perception de haut niveau à partie du monde virtuel.

2.7.1.3 Cartes de cheminement

Les cartes de cheminement discrétisent l'espace navigable sous la forme d'un réseau de chemins. Ce réseau est obtenu en reliant des points clefs répartis à l'intérieur de l'environnement (Figure 2.11). Plusieurs méthodes existent pour créer des cartes de cheminement, différentes dans la manière de créer et de relier ces points clefs.

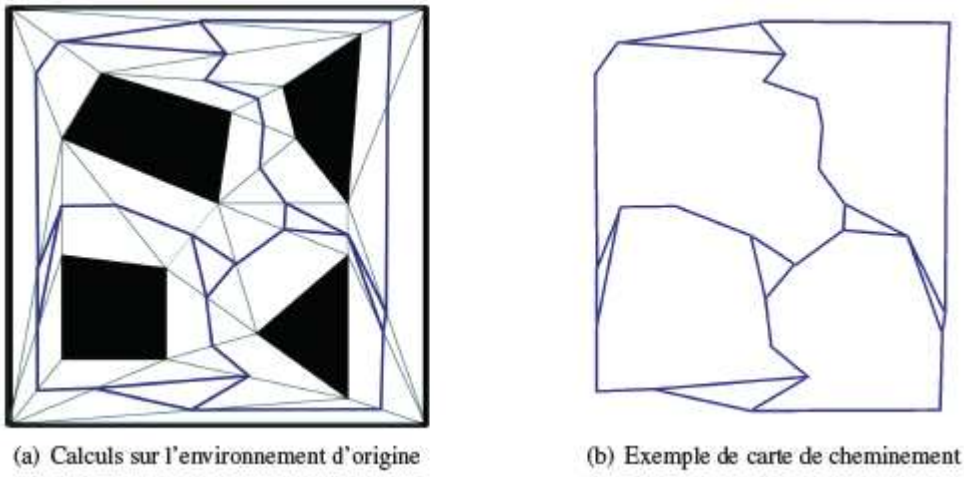


Figure 2-11 – Exemple de carte de cheminement. A gauche, triangulation de Delaunay (gris) de l'environnement d'origine, et carte de cheminement déduite (bleu). A droite, un exemple de carte de cheminement obtenue.

Graphe de visibilité

Cette méthode utilise les sommets des polygones représentant les obstacles comme points clefs. Les points clefs sont ensuite reliés deux à deux s'ils sont mutuellement visibles, c'est à dire si l'on peut tracer une ligne droite passant par les deux points sans rencontrer d'obstacle. Les graphes ainsi créés minimisent les distances parcourues, assurant d'obtenir des chemins de longueur minimale. La taille du graphe généré est ici dépendante du nombre de points mutuellement visibles, et est donc d'autant plus importante que l'environnement est ouvert avec des obstacles ponctuels et disparates.

Diagramme de Voronoï généralisé

Cette méthode est basée sur une notion d'équidistance aux obstacles de l'environnement. Pour se faire, un ensemble de sites sont évalués au sein de l'environnement, correspondant aux obstacles, dont les intersections vont former les points clefs. Les chemins ainsi générés maximisent la distance aux obstacles. Ce calcul s'avère complexe, mais son approximation peut être obtenue rapidement par des méthodes utilisant les cartes graphiques,

où le calcul est obtenu directement en effectuant le rendu des sites. Une autre méthode utilise la triangulation de Delaunay pour obtenir les points clefs du diagramme de Voronoï généralisé, en se servant du centre des triangles calculés. La carte de cheminement ainsi produite peut être considérée comme un condensé des informations de la triangulation, ne contenant plus la définition géométrique des obstacles de l'environnement.

Cartes de cheminement probabilistes

Avec cette méthode, la définition géométrique de l'environnement n'est pas utilisée comme support direct de la construction. En effet, les points clefs sont ici obtenus par une distribution aléatoire au sein de l'environnement navigable. Deux points sont ensuite reliés s'il existe un chemin libre de collision entre eux. Ce test est effectué en se basant sur la géométrie de l'environnement, mais aussi sur la taille de l'entité qui se déplace. Cette méthode est plus largement utilisée en planification de mouvement qu'en navigation, afin de générer des déplacements assez complexes (sauter, se baisser, marcher sur des piliers).

Pour conclure, les cartes de cheminement présentent le net avantage de fournir une description très condensée de l'environnement, ne nécessitant une prise de décision qu'au niveau des points clefs. Néanmoins, leur définition seule n'est pas suffisante pour gérer la complexité de la locomotion humaine. Par exemple, leur exploitation devient très difficile lorsqu'il s'agit de gérer le croisement des entités le long d'un même chemin. Certaines méthodes proposent de régler ce problème en subdivisant chaque chemin dans sa largeur, les gérant ainsi comme un ensemble de rails parallèles entre lesquels vont pouvoir transiter les entités en mouvement. Mais, même si une telle méthode produit des animations visuellement plausibles, son fonctionnement est trop éloigné du comportement humain pour être exploité dans des simulations réalistes. Les représentations sous forme de cartes de cheminement semblent donc bien adaptées à un processus de planification de chemin général, ne tenant pas compte des autres entités, mais beaucoup moins à un processus de navigation, gérant lui les mouvements locaux.

2.7.1.4 Le modèle de champs de potentiels

Le modèle à base de champs de potentiels consiste en une définition de l'environnement permettant directement de résoudre les déplacements de personnes. Les champs de potentiels caractérisent les obstacles de l'environnement par des forces de répulsion, et les buts par des forces d'attraction. Un gradient de forces, ou champ de potentiel, est alors déduit en chaque point de l'environnement comme étant une somme pondérée, le plus souvent par la distance,

des potentiels de répulsion et du potentiel lié au but (Figure 2.12). La navigation d'une entité est alors simulée par une descente de gradient depuis sa position dans l'environnement.

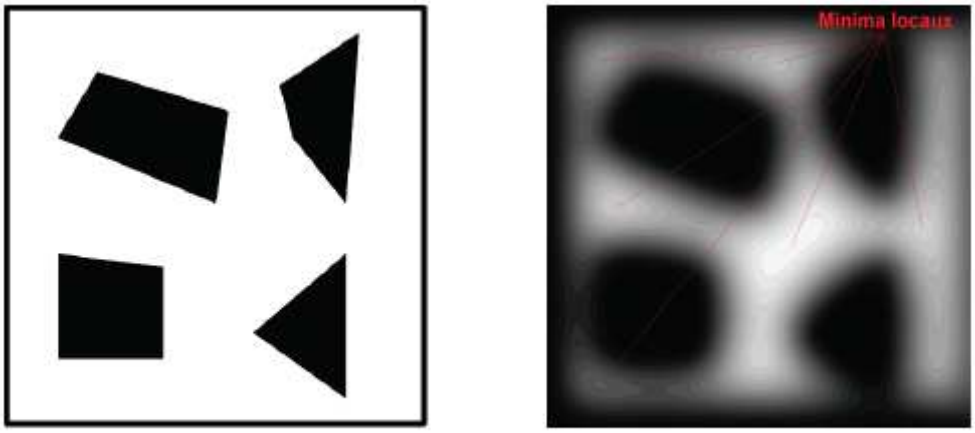


Figure 2-12 – Carte de champs de potentiels : en noir les obstacles (répulsion), en blanc les zones de navigation (attraction). Le gradient de gris exprime la valeur de potentiel dans l'environnement. Cet exemple contient 6 minima locaux : zones isolées à potentiel d'attraction maximal.

Les méthodes basées sur les champs de potentiels s'avèrent simples et efficaces, mais posent le problème des minima locaux : zones de l'environnement où un potentiel minimal isolé apparaît (Figure 2.12 (b)). Ainsi, la méthode de navigation associée va pousser l'entité à se déplacer vers le minimum local le plus proche, qui ne représente pas forcément son but. Pour pallier ce type de problème, des méthodes à base de marche aléatoire sont utilisées. Par exemple, dans RPP (Random Path Planner) [BL91], lorsqu'un minimum local est atteint, un ensemble de configurations aléatoires sont tirées puis testées avec une phase de sortie du minimum et une phase de convergence vers le prochain minimum. Les informations sont alors stockées dans un graphe dont les nœuds sont les minima locaux et les arcs traduisent des chemins entre deux minima. D'autres méthodes existent à base de tirage aléatoire de direction à suivre [CRR01], plutôt que de configuration, pour échapper au minimum local. Ces méthodes ne sont cependant pas très adaptées à l'animation comportementale. Les comportements induits par les tirages aléatoires, s'ils sont acceptables pour des robots, ne le sont pas pour des humanoïdes car ils sont en dehors de la logique de navigation humaine qui comporte une part importante de planification.

De manière générale, les méthodes à base de champs de potentiels ne sont pas applicables directement à la simulation d'humains, du fait du manque de souplesse dans la méthode de contrôle. On peut néanmoins trouver quelques cas d'application à grande échelle [GSN04], où le comportement individuel est noyé dans la caractérisation globale des mouvements de milliers d'individus. Notons tout de même que cette technique a deux grands

avantages qu'il serait bon de conserver dans une évolution permettant une décision individuelle plus importante. Premièrement, cette méthode permet la fusion de l'information au sein de l'environnement, traduisant tout ce qui intervient dans la navigation par des potentiels qui sont mélangés. Deuxièmement, et c'est une conséquence directe du premier point, ces méthodes introduisent une abstraction très forte de l'environnement, grâce à laquelle l'individu ne raisonne plus sur des entités perçues indépendantes (que ce soit la topologie des lieux ou les autres individus), mais directement sur une abstraction spécialisée pour sa tâche de déplacement.

2.7.2 Représentation du voisinage

Une première méthode pourrait être de calculer toutes les distances inter entités, cependant, pour des raisons évidentes de coût, cette solution est très peu adaptée à la simulation de foules pouvant impliquer plusieurs milliers d'individus potentiels. Nous allons présenter quelques structures de voisinage, certaines utilisant une subdivision spatiale pré calculée, d'autres utilisent des techniques spécifiques.

2.7.2.1 Exploiter la subdivision spatiale

La première typologie d'approche consiste à utiliser la subdivision spatiale. On informe les cellules que les humanoïdes traversent, ainsi lors de la recherche des voisins on aura juste à considérer quelques cellules autour de l'humanoïde. C'est la méthode choisie par C. Lascos et al [LMM03]. En utilisant une grille régulière. W. Shao [Sha06] propose une variante de cette technique. Il parcourt une grille régulière fine en lançant des rayons depuis la position de l'humanoïde. Le problème est d'ajuster la taille des cellules pour ne pas avoir trop de cellules à parcourir et pour ne pas avoir trop d'humanoïdes dans une cellule. Cet ajustement est encore moins évident dans le cas de décomposition en grille.

2.7.2.2 Structures spécialisées

La deuxième approche est de créer une structure de donnée dédiée au voisinage. Dans [LD04], il est proposé d'utiliser un graphe décrivant le voisinage d'une entité. Ce graphe est calculé à l'aide d'une triangulation de Delaunay sur les positions des humanoïdes environnants et filtré grâce à des informations de visibilité extraites de la subdivision spatiale (les arêtes vertes ôtées du graphe de voisinage dans la figure 2.13). Ce graphe de voisinage est mis à jour à chaque pas de temps.

Pour garantir un comportement réaliste, les liens entre les entités non visibles mutuellement sont tronqués. Un graphe alors est obtenu dont les sommets sont les entités, et les arcs les liens de visibilité entre entités.

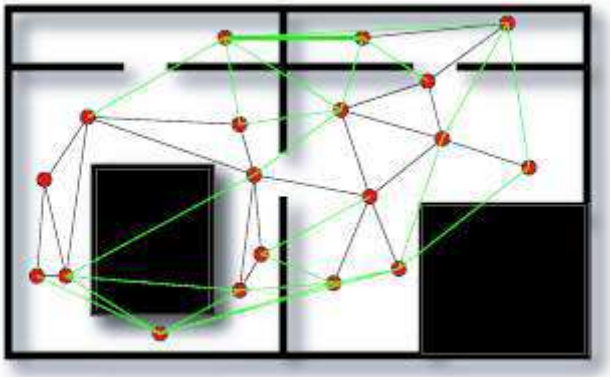


Figure 2-13 Triangulation de Delaunay filtrée par visibilité. Les points rouges sont les entités dynamiques, reliées à leurs plus proches voisins visibles par les arcs noirs. Les arcs verts sont filtrés [LD04].

N. O'Hara [Oha02] construit un quadtree sur l'ensemble des piétons. La taille des cellules de l'arbre s'adapte à la densité d'humanoïdes ; une cellule contiendra toujours quasiment le même nombre d'humanoïdes.

2.8 Planification de chemin

Afin d'exploiter une représentation de l'environnement, qu'elle soit approximative ou exacte, les simulations ont généralement recours à un procédé nommé planification de chemin. Ce procédé a pour but d'extraire un chemin reliant la position courante de l'entité à une destination donnée. Une méthode classique pour effectuer cette planification consiste à utiliser des algorithmes de parcours de graphe, bien que certaines méthodes utilisent d'autres techniques (comme la descente de gradient pour le cas des champs de potentiel, section 2.5.1.4). En effet, toutes les représentations topologiques que nous avons introduites sont exploitables sous la forme d'un graphe de points de passages, où les nœuds représentent les zones navigables et les arcs les connexions entre ces zones.

2.8.1 Algorithmes de calcul du chemin

La navigation des humanoïdes dans l'environnement correspond, le plus souvent, à se déplacer d'un point à un autre. La planification doit donc calculer un chemin entre ces deux points, en évitant les obstacles. Deux grandes catégories existent : la planification de chemin à l'aide de graphe et la planification de chemin avec les champs de potentiels.

Les algorithmes de planification de chemin sur les graphes

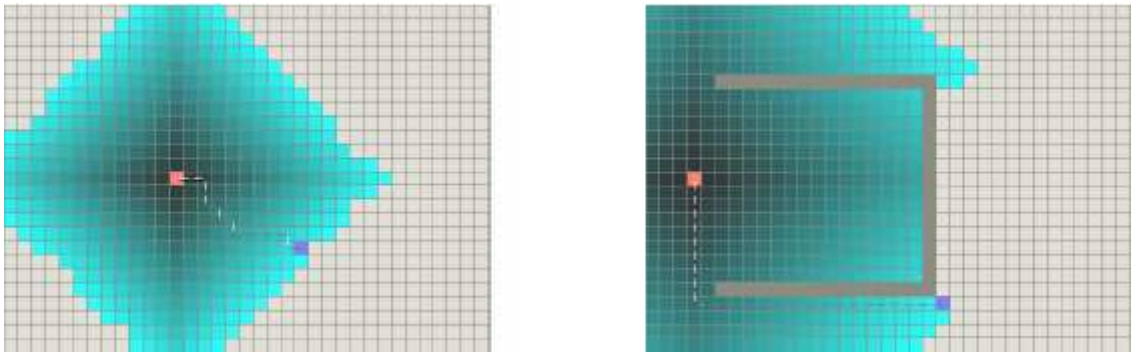
Les algorithmes de calcul de chemin sur les graphes utilisent des graphes ou les arcs correspondent à la possibilité de passer d'un nœud à l'autre. Ces arcs sont valués pour avoir une estimation du coût. Ce coût représente souvent une notion de distance spatiale, mais nous verrons dans la section suivante que des heuristiques plus complexes peuvent être utilisées. Les algorithmes que nous décrirons ici se basent sur un critère de minimisation de coût pour extraire le plus court chemin.

Dijkstra

L'algorithme de Dijkstra, aussi dénommé Flood-fill [LRDD90], 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 (par exemple avec les cartes de cheminement) ou implicite (par exemple avec les grilles). 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 l (Figure 2.14).



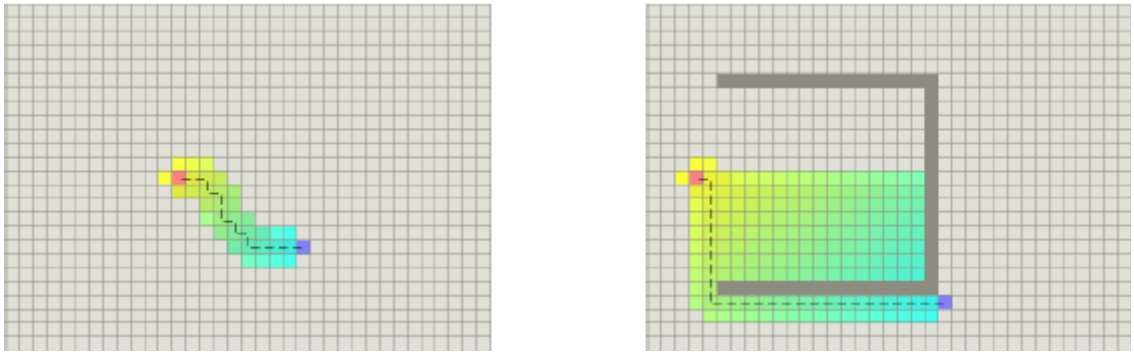
(a) Environnement non contraint (b) Environnement avec obstacle concave

Figure 2-14 Fonctionnement de l'algorithme de Dijkstra dans des cas contraints ou non. Le carré rose représente le nœud source, le mauve la destination. Le gradient de bleus correspond à la distance à l'origine, le plus clair étant le plus éloigné.

Les intérêts majeurs de cet algorithme sont sa robustesse (si un chemin existe il sera forcément trouvé) et son abstraction totale de la destination, si ce n'est comme condition de

fin. Son point faible est sa complexité calculatoire. Malgré tout, ce point faible peut aussi être un avantage, l'algorithme supportant des recherches guidées par un but plus conceptuel, associé à la sémantique des nœuds parcourus.

A* Dans les cas où la performance de la recherche est primordiale, comme en animation comportementale (dans le cas des jeux vidéos par exemple), 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 associés (Figure 2.15). 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, généralement la norme géométrique séparant le nœud courant du but.



(a) Environnement non contraint (b) Environnement avec obstacle concave

Figure 2-15 Fonctionnement de l'algorithme A* dans des cas contraints ou non. Le carré rose représente le nœud source, le mauve la destination. Le gradient de jaune à bleu correspond au coût total du chemin passant par le nœud (somme de la longueur réelle et prédite), celui-ci étant d'autant plus petit que sa couleur est bleue.

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 dont le but n'est pas clairement identifié dans le graphe topologique, comme pour des recherches exploratoires où à buts multiples.

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* [Kor85] commence par effectuer une recherche en profondeur dans le graphe. Tout d'abord, une distance maximale de parcours D est estimée comme étant la distance au but. L'exploration commence alors avec le nœud d'origine, puis évalue récursivement les voisins jusqu'à ce que l'une des deux conditions suivantes soit vérifiée : soit un chemin est trouvé, il est alors minimal, soit un nœud est parcouru dont le coût est supérieur à D. Dans ce dernier cas, la recherche continue avec une borne D remise à jour par la plus petite estimation de la distance au but évaluée précédemment. Dans les faits, 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. Ensuite, 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.

2.8.2 Critères de planification de chemin

Différents critères peuvent être évalués pour le coût des algorithmes précédents. Les plus anciennes méthodes, ainsi que la plupart de celles utilisées en animation, se basent uniquement sur la distance séparant les noeuds. Mais, comme l'ont montré les études en psychologie cette information de distance est nécessaire, mais certainement pas suffisante pour rendre compte de la complexité du raisonnement humain. Ainsi, dans des simulations se voulant plus réalistes, d'autres critères d'évaluation sont proposés.

Les calculs de ces chemins reposent sur l'algorithme de graphes, ainsi, pour une entité ayant une destination unique, on calcule les chemins possibles grâce à A* (pour des destinations multiples, on pourra utiliser un algorithme de type "flood fill"). Dans [LD04], F. Lamarche et S. Donikian proposent un algorithme hiérarchique de calcul de chemins exploitant la structure d'abstractions hiérarchique introduite. Ils commencent par calculer les chemins au niveau d'abstraction le plus haut, puis ils descendent dans la hiérarchie. Le coût du calcul est alors significativement diminué.

Dans [PDB05], Paris *et al.* proposent une planification de chemin basée sur une heuristique multicritères. Ce coût est alors ramené à une somme pondérée de critères (distance, densité de population, changements de direction, sens des flux de population) convertis en unités temporelles. Ainsi, le chemin choisi sera celui dont le coût temporel est le

plus faible. On obtient alors un bon compromis entre tous les critères à prendre en compte. La planification se fait ici aussi de manière hiérarchique de façon à tirer profit des deux niveaux d'abstraction décrits précédemment et ainsi réduire les coûts de calcul.

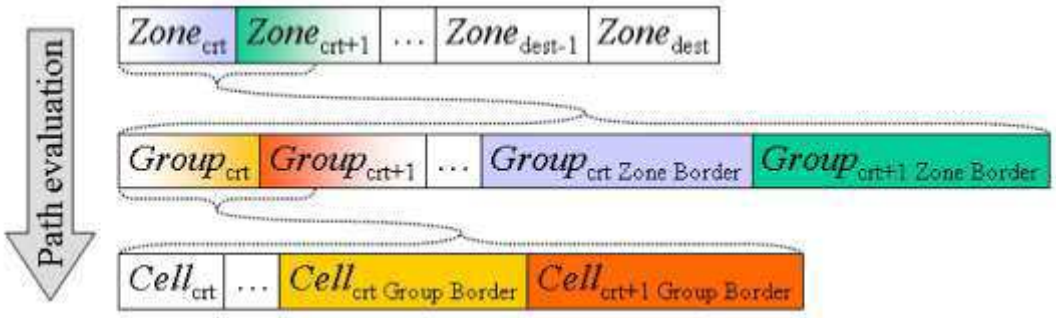


Figure 2-16 – Planification hiérarchique de chemins (un groupe est une abstraction de niveau 1, une zone une abstraction de niveau 2) [PDB05]

2.9 Conclusion

Nous nous sommes penchés tout au long de ce chapitre sur la représentation de l'environnement virtuel. Nous avons ainsi exposé les techniques permettant la représentation de l'environnement. Ainsi, on peut résumer les caractéristiques de chacune de ces techniques au tableau 2.1.

Le premier point que l'on peut soulever est la difficulté d'associer réalisme et performances. Notamment concernant les représentations de l'environnement, on a pu voir que les versions approximatives offrent un accès rapide à la description topologique, alors que les approches exactes permettent la conservation maximale de l'information géométrique. Il est indéniable que dans le cadre de simulations devant produire des résultats comparables au réel, le réalisme prend la place la plus importante.

Le deuxième point c'est que ces différentes techniques se basent, la plupart du temps, sur une projection en 2D de l'environnement pour rendre les calculs performants, mais cela introduit aussi une perte d'informations. En effet, la notion de hauteur des obstacles permet mieux de simuler les comportements des humanoïdes. Tout d'abords, elle permet de ne pas contraindre le déplacement des humanoïdes à des sols plans et peut donc, par exemple gérer des escaliers. Ensuite, elle permet de différencier plus facilement les différentes zones navigables comme par exemple le trottoir et la chaussée. Enfin, cette prise en compte de la hauteur permet de simuler correctement les obstacles de faible hauteur tels qu'un muret qui contraint le déplacement mais pas la visibilité des humanoïdes. L'utilisation d'une structure en $2D^{1/2}$ qui intègre cette notion de hauteur peut donc générer des simulations plus réalistes.

Le dernier point concerne les paramètres devant être pris en compte pour évaluer un chemin. La planification de chemin repose sur les connaissances à priori de l'entité sur l'environnement, sur les conditions dans lesquelles elle se trouve (densité, appartenance à un groupe, situation de panique ...etc) sur les informations qu'elle peut extraire de son environnement (informations données par les objets, visibilité) et donc sur un "coût" associé à chaque chemin possible.

La méthode de représentation			Points forts	Limites
Explicite	Décomposition en cellule	Représentations approximatives Grilles, Quadtree	Rapide à construire et facile à parcourir	La résolution de la grille ne s'adapte pas aux détails de l'environnement. Les obstacles doivent être aligné sur les axes sinon on obtient une décomposition en escalier.
		Représentations exactes Cellules convexes	Exprimer toute l'information relative à l'espace de navigation.	les environnements à géométrie complexe, le nombre de triangles obtenus augmente très rapidement.
Implicite	Cartes de cheminement	Graphe de visibilité	Donnent des informations directement utilisables pour planifier le chemin	bien adaptées à un processus de planification de chemin général
		Diagramme de Voronoï généralisé		
		Cartes de cheminement probabilistes		
	Champs de potentiels		simples et efficaces	le problème des minima locaux

Tableau 2.1 Récapitulatif des méthodes de représentation topologique d'environnement.

Chapitre 3

Environnement informé

3 Environnement informé

3.1 Introduction

Nous allons nous intéresser dans ce chapitre à présenter un nouveau type d'environnement: l'environnement informé. En effet, disposant de bases de données 3D de l'environnement, le besoin d'une représentation adéquate pour la détection des obstacles et la planification de chemin est primordial. Il est également important de pouvoir attacher une sémantique à cet environnement, permettant par exemple d'adapter le comportement au fait que l'humanoïde soit sur un trottoir ou sur un passage piéton. Toutes ces informations ne pouvant être déduites de la géométrie, il faut donc y ajouter des informations sémantiques.

Dans le cadre de l'animation comportementale, associer de l'information symbolique aux objets de l'environnement, plusieurs chercheurs ont expérimenté cette approche. Nous allons donc, exposer le modèle proposé par chaque chercheur tout au long de ce chapitre.

3.2 Un environnement informé

Un environnement informé présente des informations concernant un endroit, des localisations d'objets ou de lieux. L'intérêt de disposer d'informations portées par les objets de l'environnement virtuel a été montré, dans un premier temps, par M. Kallmann et *al.* [KT98] avec la notion de *Smart Objects*. L'objectif est d'enrichir l'environnement par des annotations sur sa sémantique et de doter les objets d'informations permettant aux agents d'interagir avec eux (positions d'interaction, gestes, actions...), ainsi que de propriétés sémantiques manipulées par les actions des objets tout en ayant des comportements simples. Cette idée renforce l'autonomie des agents et facilitent la planification de leurs actions.

3.3 Le modèle de Kallmann : Objets intelligents

Les objets intelligents ***Smart Objects***, introduits par M. Kallmann et D. Thalmann [KT98], sont basés sur la théorie écologique de la perception de J.J. Gibson [Gib86]. Un smart object contient l'ensemble des informations nécessaires à une interaction. Ces informations sont ensuite transmises à l'individu lorsqu'il effectue l'interaction.

Quatre types de données constituent un objet intelligent:

1. Les propriétés physiques de l'objet, comme la taille, la forme, ou le centre de masse de l'objet.

2. Les informations concernant l'interaction, servant à l'acteur pour accomplir cette interaction, comme l'endroit où se positionner, ou encore l'identification des éléments nécessaires (comme une poignée ou un bouton).
3. Les comportements internes à l'objet, pouvant être déclenchés par une interaction, et pouvant être conditionnés par des variables internes à l'objet, comme la fonction d'impression d'une imprimante.
4. Les comportements attendus de l'acteur lors de chaque interaction avec l'objet.

Les comportements de plus haut niveau sont ensuite décrits grâce à des automates à états finis. Les états sont ici constitués par les actions de base pouvant être accomplies par ou avec l'objet. Une animation est alors pourvue, en se basant sur un ensemble de gestes de base. Ces gestes sont adaptés en temps réel par un processus de cinématique inverse et de la planification de mouvement, afin de prendre en compte la position relative de l'agent vis à vis de l'objet (Figure 3.1).

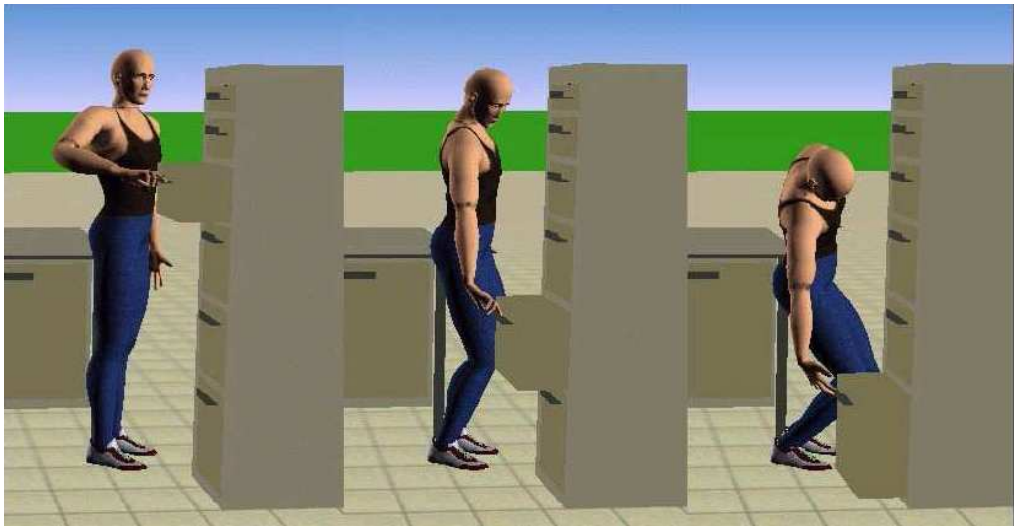


Figure 3-1 – Exemple de Smart Object : interaction avec un casier [KT98].

3.3.1 SOMOD

M. Kallmann a proposé un outil dénommé SOMOD[5] permettant de créer ce type d'objets. Cet outil donne la possibilité de réutiliser les objets et les fonctionnalités définies précédemment (Figure 3.2).

[5] Smart Object MODeler application

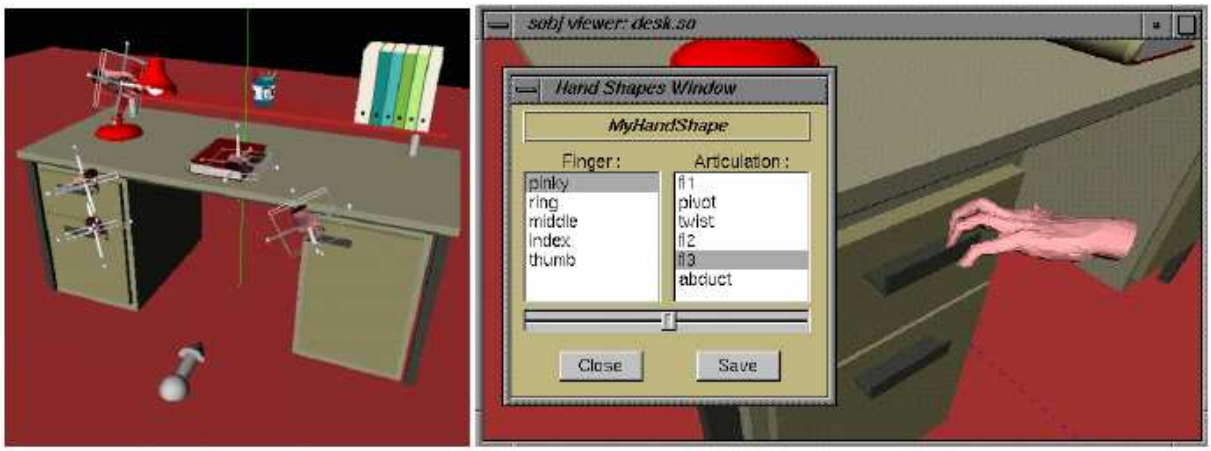


Figure 3-2 – Modélisation des objets intelligents [KT98].

3.3.2 Amélioration du modèle

Cette première approche de l'interaction était donc beaucoup plus orientée vers le réalisme visuel immédiat que vers la résolution comportementale à long terme. Des travaux ultérieurs tendent à rejoindre cette vision comportementale plus globale [Aba06]. En se basant sur le formalisme STRIPS, et plus particulièrement l'algorithme Graphplan, les smart objects sont étendus en associant des pré-conditions et des effets aux actions.

Ces informations seront ensuite utilisées dans le processus de décision de l'acteur (Figure 3.2), divisé en quatre étapes :

1. L'acteur collecte les informations du monde relatives à ses interactions.
2. L'acteur prépare sa décision en formalisant le problème.
3. La planification est effectuée, produisant un ensemble de plans.
4. Le plan relatif au but de l'acteur est déroulé, exécutant les actions de bas niveau correspondantes.

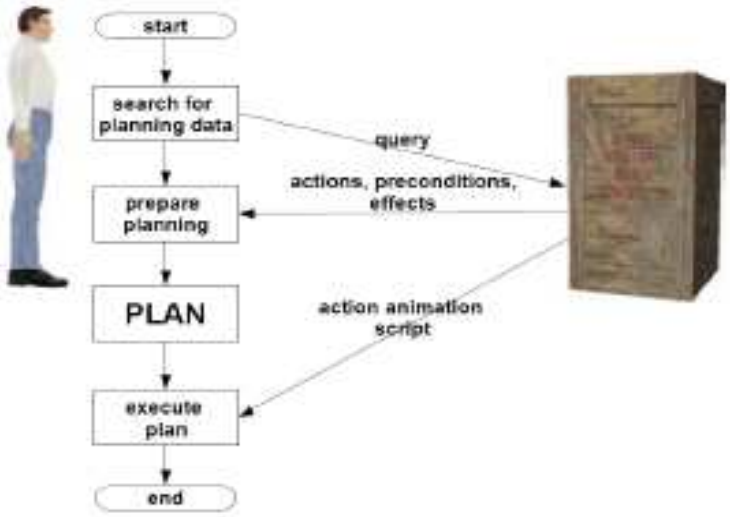


Figure 3-3 – Processus de planification avec les Smart Object [Aba06].

Une autre amélioration a été apportée afin de permettre la collaboration entre agents en introduisant le concept de délégation de tâche [Aba06]. Cette dernière permet à un agent d'aider un autre agent à la résolution d'un but (Figure 3.4). Pour se faire, le concept de facilitateur est introduit, qui est une sorte de superviseur auquel les agents vont pouvoir déléguer – transférer – des tâches. Ce superviseur va ensuite évaluer les autres agents du monde pour en trouver un qui peut accomplir la tâche, et lui transférer ce but. Le superviseur prévient ensuite l'agent d'origine de l'évolution de la tâche déléguée.

Figure 3-4 – Délégation de tâches avec les Smart Object. L'agent Martin doit porter une caisse dans une autre pièce, et se rend compte qu'il doit pour cela ouvrir une porte. Dans l'impossibilité de l'accomplir, il délègue cette tâche à l'agent Gino [Aba06].

3.3.3 Synthèse

Les objets intelligents constituent un bond en avant concernant la modélisation de l'interaction dans des environnements virtuels. Ils démontrent que la théorie écologique de J.J. Gibson est particulièrement bien adaptée à ce cadre d'application, simplifiant grandement la tâche de description et ainsi la mise en œuvre des simulations. Néanmoins, quelques observations peuvent être retenues quant à ce modèle. Tout d'abord, le fait de centraliser toute l'information dans les objets permet certes de simplifier leur description, mais contraint aussi l'évolutivité du système sur le long terme. En effet, l'ajout d'une nouvelle interaction nécessite la modification des objets qui la supportent. De même, le partage d'une interaction entre plusieurs catégories d'objets impose sa redéfinition successive. L'autre point qui n'est a

priori pas abordé concerne la définition d'agents autonomes proposant eux-mêmes des interactions. On peut intuitivement imaginer que ces agents puissent être définis sur le même principe, bien que leurs aspects dynamique et autonome risquent de poser problème.

Finalement, concernant la planification d'actions, les *smart objects* sont couplés aux STRIPS et en acquièrent donc les avantages et les inconvénients. Il n'est donc pas envisageable d'utiliser cette approche pour des foules d'individus, surtout du fait des performances. Nous pouvons conclure que ce modèle est très bien adapté à la gestion de tâches physiques, apportant un réalisme visuel certain, mais peut-être moins à la gestion de comportements à plus long terme, pour un grand nombre d'individus.

3.4 Le modèle de Farenc : Environnement informé

Dans son travail, N. Farenc [Far01] a présenté un environnement virtuel qui crée une base de données consacrée à la simulation urbaine de la vie. La base de données permet l'intégration de ce qu'elle a appelé la connaissance urbaine afin de simuler des comportements plus réalistes.

Sur la base de ce concept, à un environnement complexe, elle a ajouté une information représentative de la connaissance en milieu urbain. L'idée principale est de créer la scène et au cours de ce processus lui associer des informations via une interface entre le concepteur et un constructeur de base de données. Ce qui implique d'ajouter une couche sémantique sur une base correspondant à une scène classique (ensemble d'objets graphiques) modélisés à l'aide d'un logiciel graphique (Figure 3.5). La couche sémantique associe aux objets des propriétés utilisables lors de la simulation de la vie urbaine.

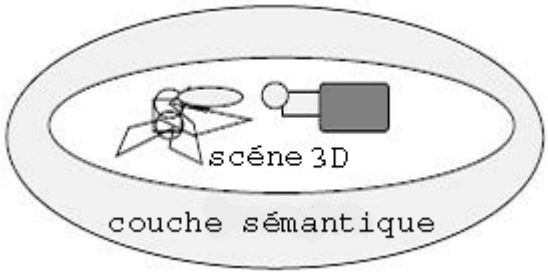


Figure 3-5 – Le schéma de modélisation [Far01].

3.4.1 Entités environnementales et décomposition hiérarchique

Le modèle de scène de N. Farenc avec Environnement Informé correspond à un ensemble d'entités environnementales (les ENVs) définissant une base de données. Un ENV représente une surface ou un volume et a une information sémantique associée. Un ENV

simple peut se composer de différents types d'objets tels que des objets perçus comme obstacles (des arbres ou des murs par exemple) et des objets utilisés pour des interactions spécifiques (objets, portes ou escalier roulants).

Un ensemble de règles élémentaires sont définis, en ce qui concerne les diverses caractéristiques d'emplacement et la définition des comportements humains pendant les interactions avec des objets ou avec d'autres humains. Puisque, cette méthode a l'inconvénient de manipuler tout et les règles doivent couvrir tous les sujets, et les règles efficaces pour un contexte urbain sont trop nombreuses et complexes. N. Farenc a choisi de distribuer l'information ou la connaissances aux applications spécifiques. De cette façon, une application peut traiter d'une comportement interne d'une foule tandis qu'une autre est consacrées aux interactions avec les objets (position de main, mouvement du corps et des objets).

Avec un environnement complexe énorme tel qu'une ville, N. Farenc devait considérer le problème de traiter une grande quantité de donnés pendant l'accès ou la manipulation. Son approche était de définir quelques zones structurées. Les zones sont subdivisées en sous domaines, ou groupées, selon le niveau de l'information. Ainsi, par analogie à une carte géographique, elle a décomposé une grande zone en sous domaines avec l'information inhérente au niveau de la description. Au niveau de ville, à la base de données, elle a associé l'information correspondant aux axes principaux de la ville pour entrer ou sortir. Ces axes principaux permettent de traverser la ville. À un niveau plus bas, ces axes sont identifiés comme des rues. Dans la base de données, le niveau de rue fournit des informations au sujet des passages à piétons et des trottoirs. Puisque elle utilisait la notion d'encapsulation, la même surface peut appartenir d'abord à un trottoir, puis à une rue, puis à un bloc et au niveau le plus élevé, à la ville. Cette classification correspond à une hiérarchie (Figure 3.6), triant et rangeant toutes les données. La ville est ainsi divisée en plusieurs zones, selon leurs propriétés géographiques et fonctionnelles.

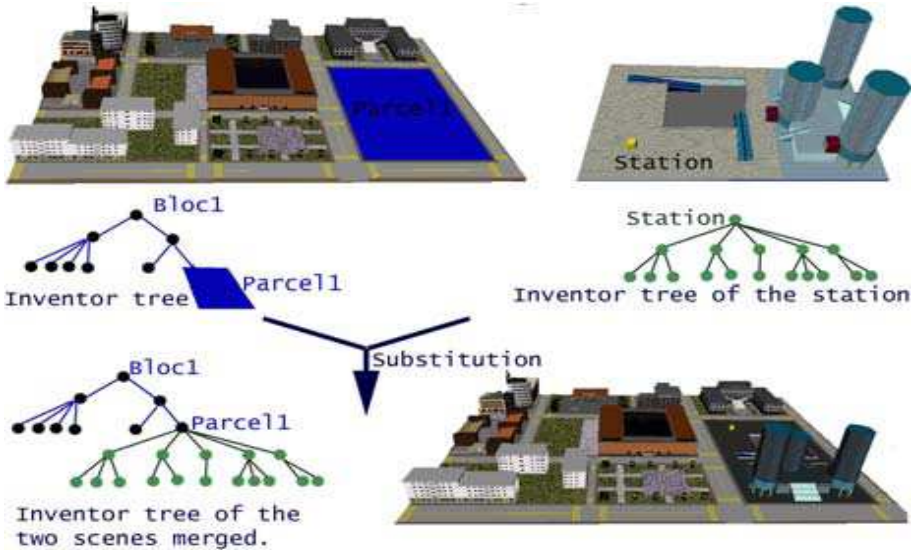


Figure 3-6 – Vue de la hiérarchie et de la méthode pour créer une nouvelle scène et la base de données associée [Far01].

La création d'une base de données repose sur une scène segmentée. Toutes les régions sur le terrain qui fournissent des informations spécifiques en termes de fonctionnalité sont modélisées comme des objets distincts. La figure 3.7 montre un trottoir sélectionné. Seule sa propre représentation est affichée en vue. La figure 3.8 illustre la hiérarchie correspondant à la segmentation de la scène, l'objet sélectionné et les différentes étiquettes utilisées pour fournir de l'information.

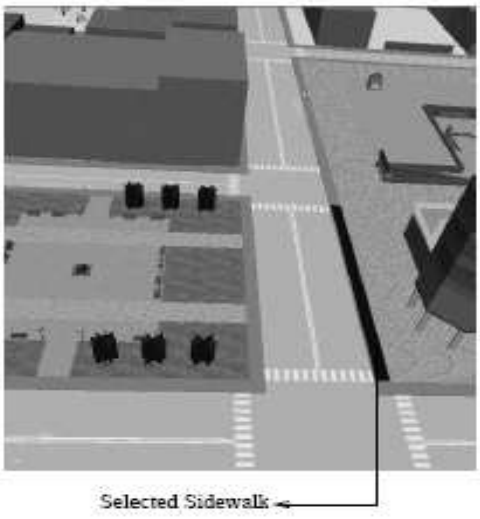


Figure 3-7 – Vue de la ville avec un trottoir sélectionné [Far01].

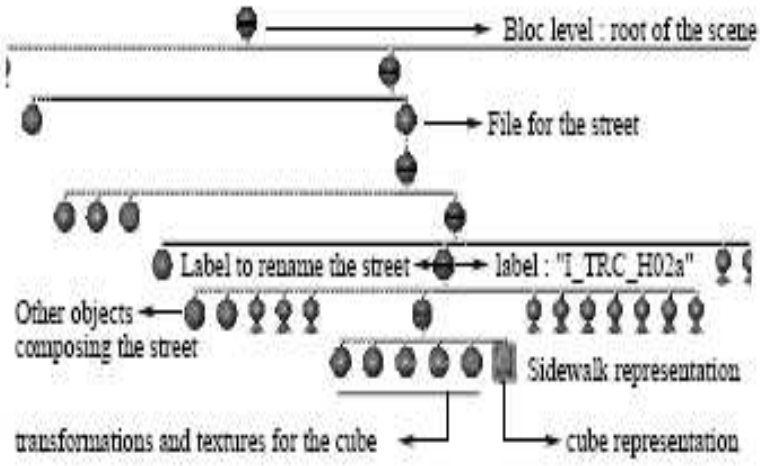


Figure 3-8 – Vue de la hiérarchie pour un trottoir sélectionné [Far01].

Différentes catégories d'objets

La base de données graphique est composée de différents types d'objets dont certains ne sont utilisés que pour effectuer des calculs spécifiques tels que la détection de collision ou de données sur les comportements.

Certains objets de la scène 3D nécessitent une fonctionnalité, ce qui pourrait être soit interne à l'objet (le mouvement de la porte) ou externe (le panneau). Dans le dernier cas, il informe les autres agents de l'IVE. Le Smart Object consiste à deux types de fichiers de données: la description géométrique et la description des fonctionnalités.

3.4.2 Entités mobiles et création de chemins

Une zone définie comme environnement informé fournit les sous-domaines (ENV), avec la liste d'objets avec lesquels il faut éviter toute collision. Afin de réduire au minimum le nombre d'objets, des objets périphériques tels que des murs ne sont pas inclus dans la surface ENV. La base de données correspond à la perception nécessaire pour se rendre compte de tous les objets à l'intérieur d'un ENV ou de toutes les surfaces touchant un ENV afin d'exécuter plus efficacement la perception virtuelle. La hiérarchie de décomposition ne fait aucune distinction entre un parc et un cimetière. Tous les deux sont des éléments dans la ville. Afin d'indiquer une telle connaissance, une étiquette est ajoutée dans la définition de l'ENV, pour permettre une meilleure spécification.

L'environnement informé fournit également la possibilité de faire circuler des mobiles dans la ville en utilisant la base de données. L'idée est de calculer le meilleur chemin d'un point à encore, selon le type mobile utilisé. Une liste de points peut être suffisante pour un

piéton mais pas pour d'autres types de mobiles. C'est pour cela que les chemins sont représentés par quelques zones, qui sont les ENVs avec des points d'entrée/sortie pour quitter et entrer dans une nouvelle surface selon le type de mobile et la sémantique attachée aux zones. Un piéton ne peut pas utiliser une voie réservée aux bus pour marcher, par exemple. Au niveau de la rue, un chemin pour piéton passe par des trottoirs, des passages à piétons et autres voies. Il y a donc des types de graphes différents suivant les entités, mais des ENV permettant une connexion entre graphes. Ainsi, les arrêts de bus sont les liens entre les graphes pour les piétons et ceux pour les bus. Pour définir un chemin spécifique, l'usager peut utiliser une interface développée.

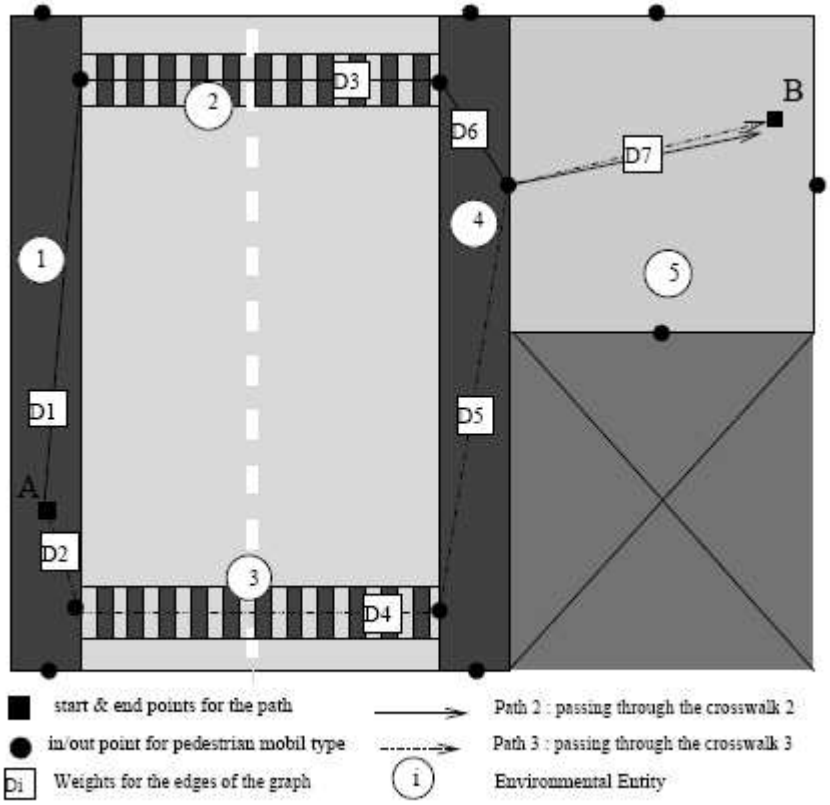


Figure 3-9 – Chemin à travers une rue montrant les dépendances des bords [Far01].

Optimisation du chemin

Les chemins sont calculés en utilisant les points ENV situé au milieu des côtés ENV. Les chemins peuvent présenté quelques formes déchiquetées, et afin d'éviter ce type d'incohérence, N. Farenc a amélioré le calcul du chemin en ajoutant une optimisation. Afin de lisser la trajectoire, cette optimisation analyse le chemin d'accès et pour chaque point dans le chemin, calcule un nouvel point sur les bords reliant les deux ENV connexes. Elle a laissé une distance minimale à partir de la frontière de l'ENV. La figure 3.10 (à gauche) montre un

exemple de calcul de trajectoire pour un piéton dans le parc, et la figure 3.10 (à droite) présente le même chemin en utilisant l'optimisation.

En utilisant la décomposition hiérarchique, le problème de la planification de chemin est simplifié. Cependant, la figure 3.10 illustre l'incohérence dans une telle simplification, à cause de l'utilisation des points d'entrée / sortie. L'optimisation trouve le meilleur chemin d'accès correspondant à la trajectoire initiale de choix, mais la distance associée est fausse (distance en utilisant les points d'entrée / sortie). Une meilleure solution a été l'application de l'algorithme de Perez Lozano, mais en raison de la nécessité de rapidité de calcul du chemin, ça ne pouvait pas être appliqué à une grande scène comme une ville. Pour utiliser la décomposition hiérarchique, qui peut être associée à une étape de pré-traitement, une solution consistait à appliquer l'algorithme de Perez Lozano localement (dans la parcelle ENV du parc par exemple).

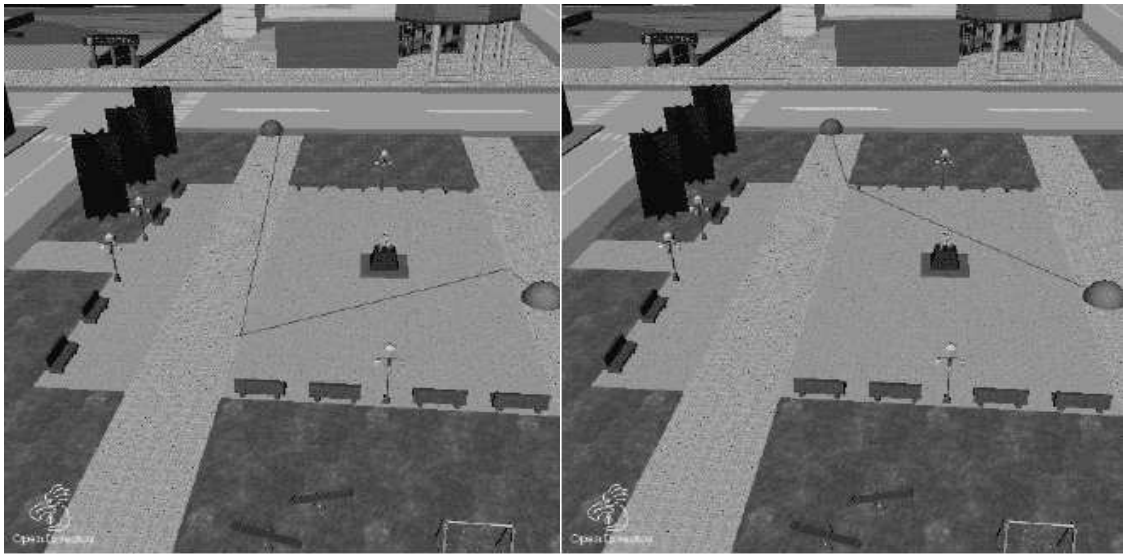


Figure 3-10 – Calcul de chemin sans optimisation (à gauche), avec optimisation (à droite) [Far01].

3.4.3 Planification d'Actions

Un exemple de planification de l'action lors de la simulation est l'exemple des escaliers. Un agent doit suivre un chemin pour aller d'un endroit à l'autre, et pendant son déplacement, il doit monter les escaliers. L'environnement permet à l'agent avec un fichier image clé pour effectuer le mouvement de monter les escaliers. Le fichier image clé a été pré-traité avant la simulation. La base de données fournit un lien entre l'ENV et le fichier image clé. Pendant la simulation, l'ENV a informé les agents de quel fichier image clé à jouer pour l'ascension.

3.4.4 Synthèse

Les équipes qui ont précédé N. Farenc se sont attaquées à la construction de bâtiments ou de villes virtuelles, ou la reconstruction d'une vraie ville en utilisant du traitement d'image, des données archéologiques et des outils sophistiqués. La différence principale entre l'approche de N. Farenc et les études précédentes est que N. Farenc s'est concentrée sur la simulation d'une ville peuplée d'humains virtuels réalistes. Elle a prouvé que, par comparaison avec des simulations utilisant des icônes qui représentent des humains, les simulations avec les humains réalistes permettent une meilleure approche pour évaluer de vraies contraintes dans une simulation d'une vraie ville.

3.5 Le modèle de Doyle: Annotations

P. Doyle [Doy02] propose d'« annoter » les objets avec les connaissances spécifiques liées à l'environnement et de ne laisser dans l'agent que les compétences abstraites. Il présente un *framework* où chaque information provenant de l'environnement est annotée. Le type d'annotation est défini en utilisant six propriétés appelées « dimensions ». Quatre de ces dimensions caractérisent les annotations :

- **{descriptive, directive}** Lorsqu'il s'agit d'un fait sans indication sur la manière de l'utiliser, ou d'une information contenant une demande de modification de son comportement.
- **{entité, relation, opération, évènement}** L'information peut décrire une entité, c'est-à-dire un élément de l'environnement, que ce soit un objet physique ou un concept abstrait. Elle peut aussi décrire une relation entre entités, une opération qui représente une action, ou encore un évènement qui correspond à un changement dans l'environnement (ou à de la communication).
- **{description, contexte, intention, structure}** L'information peut être une description, un contexte, qui indique que certaines informations ne sont pertinentes que dans certaines circonstances. L'intention indique que certains éléments permettent de réussir un but. Ce type d'annotation peut contenir les détails nécessaires à cette réussite. La structure décris des connexions entre annotations.
- **{effective, affective}** la différence entre l'aspect opérationnel d'une information et l'aspect émotionnel ou qualitatif. Deux autres dimensions définissent de quelle manière une annotation sera transmise et stockée.

- **{passive, active}** Pour définir la manière dont le personnage va percevoir l'annotation: si cette annotation est passive, il va devoir aller la chercher pour la connaître, si elle est active, elle sera envoyée au personnage même s'il ne l'a pas explicitement demandée.
- **{statique, dynamique}** Pour indiquer si les faits que décrit l'annotation vont changer ou non au cours du temps.

Le système d'annotation permet aux personnages d'évoluer dans des environnements non familiers en étant crédibles et compétents. Il permet aussi de faciliter la mise en place de plusieurs personnages avec des ressources limitées.

S. Aubry [Aub07] utilise lui aussi la notion d'annotation d'environnements virtuels, afin d'avoir à la fois un environnement réaliste et des données abstraites liées à celui-ci (théorie des IRVE[6] de D.A. Bowman et al. [BNC+03]). Contrairement aux travaux de Doyle, les annotations servent ici de moyen de communication avec l'utilisateur, dans le cadre, par exemple, de travail collaboratif.

3.6 Ontologies

D'autres travaux ont été réalisés pour exprimer le contenu d'un environnement, qu'il soit visuel ou non avec un seul langage de description. Par exemple, Otto [Ott05] parle d' « Environnements Virtuels Sémantiques » où des techniques du Web sémantique sont utilisées, le but étant de trouver une base indépendante du logiciel de description d'environnements virtuels. M.A. Gutiérrez [Gut05] propose lui aussi une représentation sémantique de la géométrie de l'objet, ainsi que de ses fonctionnalités intrinsèques et des différentes interactions qu'il offre.

3.7 Le modèle de Badawi

M. Badawi [Bad06] a proposé une méthode ou l'information qui est stockée dans les objets est une description globale du processus d'interaction, sans être trop détaillée, une sorte de synopsis d'action. Ces objets sont appelés Objets Synoptiques. Il utilise des Surfaces Interactives pour décrire les surfaces sur l'objet utilisées pendant l'interaction ainsi que l'espace autour de l'objet affecté. De plus, l'objet informe l'agent des actions à accomplir à travers des Actions de Base qui, associées aux surfaces interactives, décrivent le processus d'interaction à travers des Actions Complexes. Les actions de bases sont spécifiques à chaque

[6] Information-Rich Virtual Environment

type d'agent et sont donc interprétées différemment par l'agent lui-même, résultant en une variété d'interactions possibles pour un même objet.

3.7.1 S.T.A.R.F.I.S.H.

STARFISH est l'acronyme de Synoptic-objects Tracking Actions Received From Interactive Surfaces and Humanoids[7].

STARFISH, est un nouveau système qui permet de définir facilement les interactions entre les agents autonomes et les objets de l'environnement. Le noyau STARFISH est les objets synoptiques qui sont des objets contenant des informations décrivant la façon dont ils peuvent être en interaction avec. Cette information est rendue disponible à travers deux composantes principales:

- **Les actions**
- **Les surfaces interactives**

3.7.2 Les actions

a. Les actions primitives

Give	Transfer a relationship (give, take)
Transfer	Transfer location of an object (go, carry)
Displace	Apply force to an object (push, throw)
Move	Move own body part (kick, reach)
Grasp	Grab an object (grasp, let go)
Ingest	Take an object into own body (eat)
Speak	roduce sound (say, sing)
Attend	Focus sense organ (listen, look at)

Tableau 3.1 – Actions basiques de STARFISH [Bad06].

M. Badawi a définit un groupe de sept actions primitives, qu'il appel *Actions de Base*:

- **TRANSFER** (transférer): Cette action est utilisée pour déplacer l'agent, lui faire changer de position. Elle utilise généralement des surfaces d'influence pour obtenir les informations de placement par rapport à l'objet. Elle ne contient pas d'informations de haut niveau pour la planification de chemin ou d'évitement de collision, elle instruit simplement l'agent d'aller d'un point A à un point B. La

[7] Objets synoptiques pour suivre les actions reçues par des surfaces interactives et des humanoïdes.

manière dont le déplacement vers la cible se fait dépend entièrement du module comportemental de l'agent.

- **MOVE** (déplacer): Cette action est utilisée pour déplacer une partie du corps de l'agent. Cette action est la plus complexe des actions de base. Elle décrit des actions aussi variées que bouger un bras, baisser la tête, donner un coup de pied, s'asseoir, etc. A l'instar de transfer, cette action ne possède pas de planification de trajectoire et d'évitement de collision. Elle indique le déplacement d'une partie du corps vers une cible et c'est au moteur d'animation de veiller à ne pas entrer en collision avec les objets du monde. move permet aussi d'utiliser des outils et de déplacer des objets. Par exemple, si l'agent utilise un marteau pour frapper un clou, move instruira la main tenant le marteau de se déplacer. Mais, dans ce cas, ce sera la SI du marteau qui est utilisée pour déterminer les points de contact, et non celle de la main.
- **GRASP** (attraper): Cette action est utilisée pour asservir la position d'un objet à celle de l'agent. L'objet calculera donc sa position par rapport à celle de l'agent quand il est grasp-é par ce dernier. grasp informe également l'agent des SI de l'objet attrapé dans le cas d'éventuels move.
- **INGEST** (ingérer): utilisée par l'acteur de l'action pour prendre un objet dans son propre corps. Elle permet de définir des conteneurs avec une capacité quelconque et ainsi de simuler leur degré de remplissage. Elle permet aussi à un agent de manger ou boire, par exemple, pour assouvir des besoins physiologiques.
- **EXPEL** (expulser): Action inverse de ingest. Elle est utilisée pour enlever des objets d'un conteneur qui les a injecté, au cas où une telle opération est possible.
- **TELL** (dire): Donne des informations à l'agent au travers de ses senseurs. Elle n'utilise pas de SI, mais cible des senseurs spécifiques. Elle peut être utilisée pour fournir des informations sensorielles (visuelles, auditives, etc.) mais aussi des informations physique quantifiables comme la température, par exemple. Elle peut aussi être utilisée en mode diffusion et mettre l'information à disposition de tous les agents, sans cibler de sens spécifique. Dans le cas où tell n'est pas en mode diffusion, l'agent doit effectuer un attend correspondant au senseur utilisé par tell pour récupérer l'information.
- **ATTEND** (attention): Utilisée par l'agent pour récupérer l'information donnée par tell.

b. Les actions complexes

Les actions complexes permettent de spécifier l'enchaînement des actions de bases, associées aux SI, afin de décrire des processus d'interaction complets.

Automates

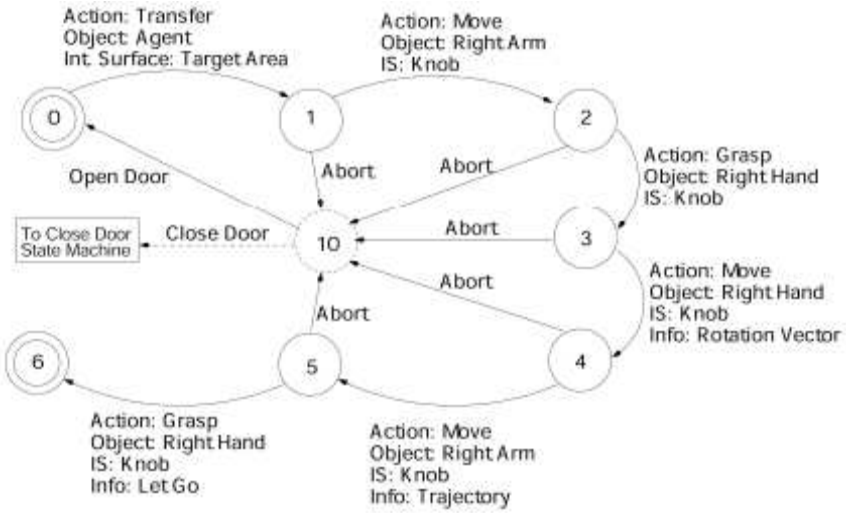


Figure 3-11 – La machine d'état fini de l'action d'ouverture d'une porte [Bad06].

Les actions complexes sont plus qu'un simple enchaînement d'actions. Le passage à l'étape suivante est dépendant du résultat de l'action de base courante. Par exemple, si un agent est à la portée de sa cible, il peut effectuer un grasp sinon, il doit d'abord effectuer un transfert. Ce type de processus peut être décrit à travers des automates. Un nœud de l'automate contient une action de base, sa SI associée, et les ressources utilisées par l'action. Une ressource peut, par exemple, être le bras droit pour une action move qui déplace le bras droit. La transition entre les nœuds de l'automate se fait lorsque l'action courante est terminée et que les ressources de l'action suivante sont disponibles.

Grâce à cette notion de ressources, il est possible de lancer plusieurs automates en parallèle et ainsi permettre à l'agent d'effectuer plusieurs actions en même temps, sous réserve qu'elles n'utilisent pas les mêmes ressources. Certains nœuds de l'automate sont considérés comme clé. Ces nœuds définissent un état stable de l'objet. Par exemple une porte peut être soit ouverte soit fermée. Cependant, si l'interaction est interrompue alors que l'objet n'est pas dans un état clé, l'automate est placé dans un état dit de transition qui informera un éventuel agent voulant interagir avec l'objet des actions à effectuer pour ramener l'objet dans un état clé.

3.7.3 Les surfaces interactives

Lorsqu'un humain veut interagir avec un objet il essaie de déterminer sa fonctionnalité au travers d'indications visuelles de l'objet qu'il perçoit. Ces propriétés sont appelées *affordances* [Gib86] ou tout simplement information dans le monde. Dans un environnement virtuel, M. Badawi propose de placer ces informations dans le monde lui-même. Pour cela il utilise des surfaces interactives. Il distingue deux types de surfaces interactives:

1. **Surface d'Interaction:** ce sont des surfaces faisant partie de la géométrie de l'objet. Elles indiquent les parties de l'objet prenant part à l'interaction. Ces surfaces indiquent spécifiquement les parties de l'objet associées à une action de base. Par exemple, la figure 3.12 représente la surface d'interaction associée à move pour placer les mains de l'agent au bon endroit avant d'effectuer un grasp.
2. **Surface d'Influence:** ces surfaces n'appartiennent pas à la géométrie de l'objet. Elles décrivent l'espace entourant l'objet affecté par le processus d'interaction. Les surfaces d'influence peuvent être de deux types différents: positif et négatif. Les surfaces d'influence positives indiquent l'espace dans lequel l'agent doit se trouver pour interagir avec l'objet. Les surfaces d'influence négatives, quant à elles, indiquent l'espace à éviter lors de l'interaction. La figure 3.13 (a) montre une surface d'influence positive associée à un siège. L'agent doit se placer dans la zone marquée en jaune pour s'asseoir dessus. La figure 3.13 (b), elle, montre une surface d'influence négative associée à une porte. La surface bleue indique la trajectoire de la porte et l'agent ne doit donc pas être placé dans cette zone lors de l'interaction.

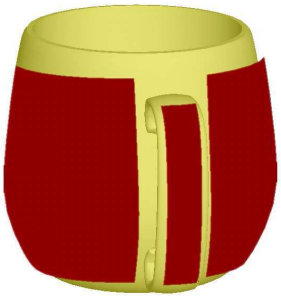


Figure 3-12 – Une tasse avec une surface d'interaction, en rouge [Bad06].

(a) Surface d'influence positive pour un siège.
(b) Surface d'influence négative pour une porte.
Figure 3-13– Les deux types de surfaces d'influence [Bad06].

Pour mieux décrire les affordances d'un objet, une SI est toujours associée à une action de base.

3.7.4 Synthèse

Il existait deux approches pour permettre à un agent autonome d'interagir avec son environnement. La première consiste à placer tout le processus d'interaction dans l'agent lui-même et lui permettre de déduire l'utilisation des objets à partir de ses propres informations internes. Cette approche présente la limite d'être très complexe et coûteuse en temps de calcul assez rapidement. La deuxième approche consiste à stocker l'information dans l'environnement lui-même, les objets prenant en charge l'animation de l'agent lors de l'interaction. Bien que plus légère en temps de calcul, cette approche produit toujours exactement les mêmes animations qui ne peuvent pas être réadaptées. De plus, les interactions ainsi décrites sont spécifiques à un certain type d'agent et sont définies une fois pour toutes.

M. Badawi a proposé une méthode hybride qui allie le meilleur des deux approches citées précédemment. Les objets synoptiques permettent de séparer la forme d'un objet de sa fonction et la description des interactions fournies par les objets synoptiques sont génériques et indépendantes du type d'agent. Mais d'un autre coté, la qualité de l'animation résultante dépend directement du choix des SI et les objets synoptiques peuvent gérer l'interaction par un seul agent. Aucune collaboration n'est possible dans S*tarfish*.

3.8 Modèles de Paris

3.8.1 La représentation topologique

Subdivision spatiale informée

La première étape pour obtenir la représentation de l'environnement consiste en la subdivision de la base graphique 2D de cet environnement par une triangulation de Delaunay contrainte [LD04] de plans architecturaux au format *AutoCAD*. Les cellules produites lors de cette subdivision forment un graphe dont les noeuds sont typés en fonction de leur niveau de connexité topologique *c* : cul de sac (*c* = 1, en rouge), couloir (*c* = 2, en violet), et carrefour (*c* = 3, en vert).

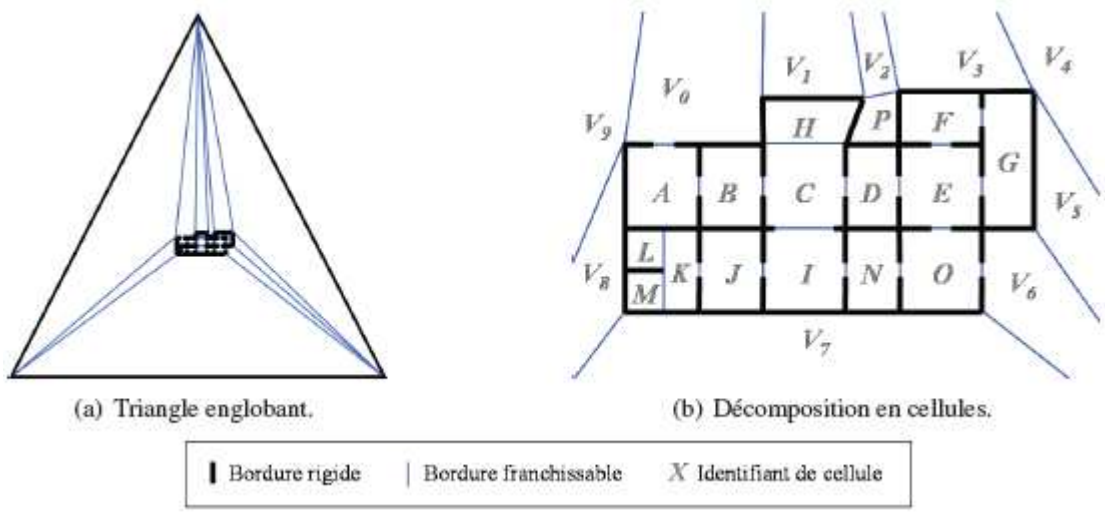


Figure 3-14 – Subdivision spatiale préalable à l'abstraction [Par07].

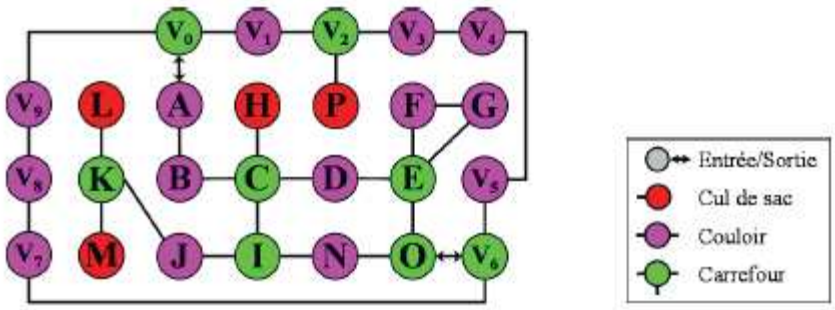
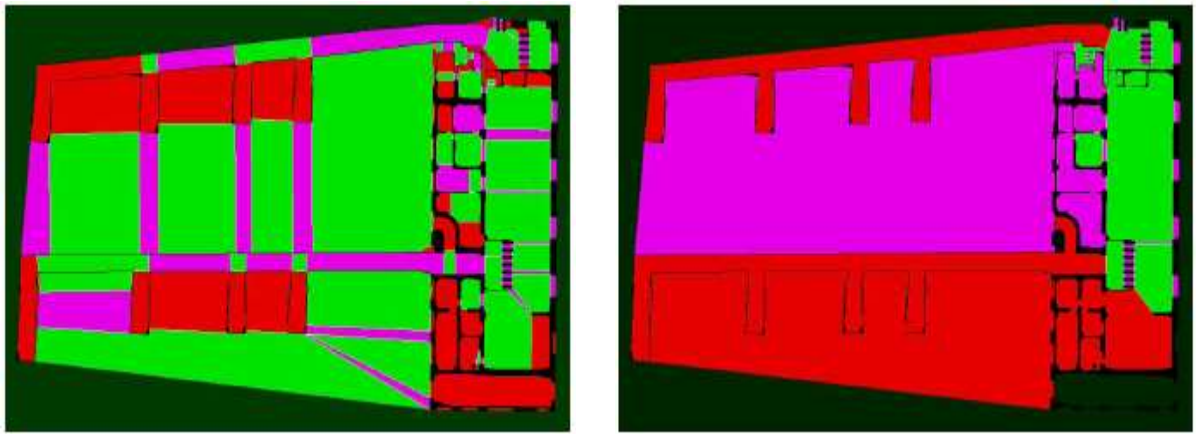


Figure 3-15 – Graphe topologique extrait de la subdivision [Par07].

Abstraction

Comme il est montré sur la figure 3.15, une subdivision produit un grand nombre de cellules, même pour un environnement relativement simple. C'est pourquoi S. Paris a mis en place un processus d'abstraction de l'environnement [Par07] visant à faciliter l'exploitation d'une telle représentation. L'abstraction se déroule en deux étapes, produisant deux graphes successifs de plus en plus condensés, organisés hiérarchiquement, i.e. dont les nœuds contiennent un sous-ensemble du graphe précédent. Les nœuds du premier niveau d'abstraction sont nommés des *groupes*, et ceux du second niveau des *zones*. L'heuristique de

regroupement des cellules se base à la fois sur leurs propriétés géométriques, maximisant l'aspect convexe des groupes pour obtenir un découpage plus naturel, et sur leurs propriétés topologiques, réunissant les nœuds de même catégorie afin de mettre en évidence des zones de navigation plus globales. Comme c'est montré sur la figure 3.16, l'abstraction produit une représentation plus intuitive de l'environnement, et elle a l'avantage de réduire le nombre de nœuds à manipuler.



1ère abstraction (*153 groupes*) 2ème abstraction (*40 zones*)

Figure 3-16 – Abstractions successives de la gare de St Denis [Par07].

Graphe informé

Une fois ce graphe hiérarchique obtenu, un ensemble de données pré calculées est associés à certains de ses nœuds. Tout d'abord les champs de visibilité potentiels sont calculés au travers de l'environnement, qui vont servir pour le comportement d'observation des entités.

D'autre part, les plus courts chemins sont aussi précalculés, ainsi que leur largeur de passage minimale, à l'intérieur de chaque groupe entre chacune de ses frontières franchissables, ce qui servira lors de la planification de chemin. Une grille orientée est aussi affectée à chaque groupe, dont les cases vont stocker dynamiquement lors de la simulation le nombre d'entités qu'elles contiennent. Ce procédé permet ensuite de faire un calcul fin de densité pour chaque paire de connexion du groupe, et ainsi d'obtenir une valeur de densité tenant compte de la topologie des lieux (et ainsi des regroupements gênant la circulation ou non). La méthode de calcul utilisée prend la forme d'une planification de chemin au sein de la grille (figure 3.17), avec pour origine la case correspondant à l'entrée du groupe envisagée, et pour destination la sortie évaluée. Le chemin est évalué en cherchant à minimiser la distance

parcourue, mais aussi la densité des cases traversées. A la fin, c'est la densité la plus importante des cases composant le chemin qui est retenue.

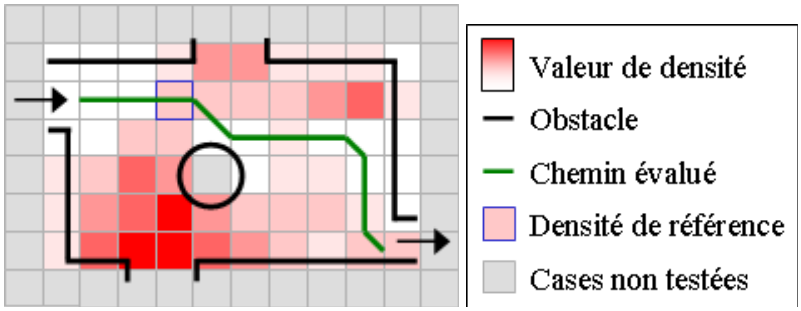


Figure 3-17 Calcul fin de densité utilisant une grille orientée [Par07].

Finalement, un compteur est associé à chaque frontière intergroupe, qui est mis à jour par chaque entité traversant le groupe en direction de cette frontière afin de permettre le calcul des flots de personnes dans le groupe.

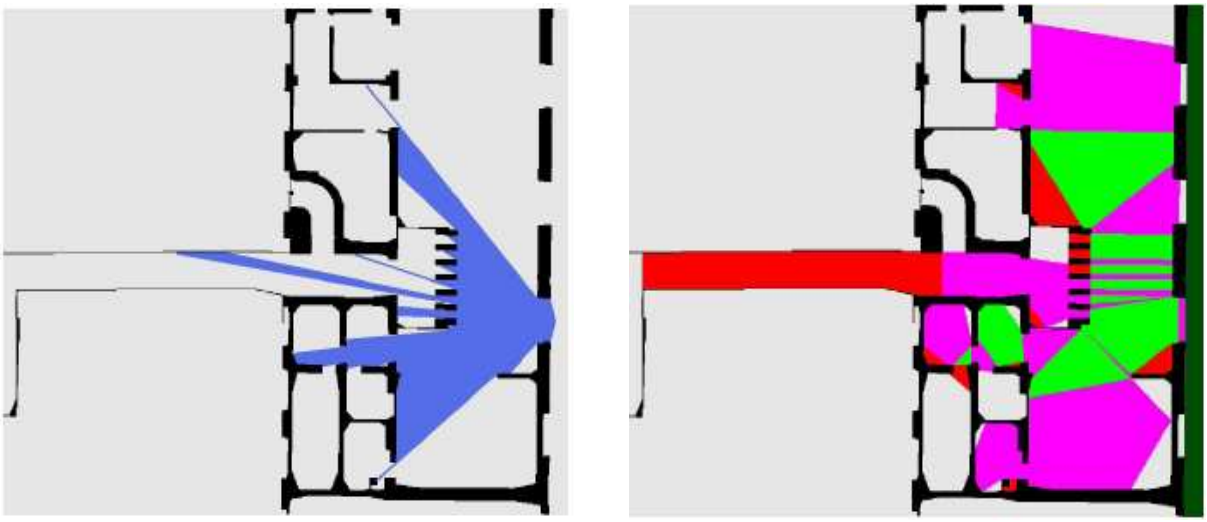
3.8.2 Champs de visibilité

Pour pouvoir optimiser l'accès aux données lors de la simulation ainsi que la navigation des entités, Paris propose d'associer au graphe représentant l'environnement certaines informations. Une information capitale pour une navigation réaliste est la visibilité que peut avoir un humanoïde en une position et une orientation donnée.

Ainsi, dans [Par07], il introduit la notion de PVS, (*Potential Visibility Sets* en anglais). Puisque il serait très coûteux de vouloir calculer l'ensemble des cellules visibles depuis chaque position dans l'environnement du fait du nombre potentiellement très important de cellules. De plus, l'information serait à fortiori redondante. Il choisit alors de calculer la visibilité pour certaines positions représentatives.

Ces PVS se présentent sous la forme d'arbres dont la racine est une frontière franchissable de l'environnement, et dont chaque nœud correspond à la prochaine frontière franchissable potentiellement visible. Le procédé de calcul est une planification de chemin au travers de l'environnement. La figure 3.18 montre un PVS représenté sous la forme de frustra composés, ainsi que la correspondance en matière de connectivité des cellules perçues. Un frustrum est ici la projection dans le plan d'un cône de vision, et est donc représenté par un triangle dont l'un des sommets est la position de l'entité, et les deux autres sommets les limites du champ de vision dans l'environnement. La propriété la plus importante d'un tel

frustrum est que l'ensemble de sa surface couvre une zone libre, i.e. sans obstacle, de l'environnement.



(a) Frustra composés (b) Connectivité perçue

Figure 3-18 – Exemple de champ de visibilité (PVS) [Par07].

3.8.3 Les objets interactifs

Le modèle d'objets de S. Paris se dénomme BIIO, pour Behavioral Interactive and Introspective Objects. Dans BIIO tout est objet, que ce soit les équipements, les humanoïdes, ou encore des concepts plus abstraits comme les comportements.

Un objet BIIO est composé de deux sortes de constituants : des attributs et des fonctionnalités.

Les attributs représentent les propriétés internes d'un objet, et donc son état. Les fonctionnalités représentent les procédures accessibles sur l'objet. Elles permettent de décrire une action de l'objet sur l'environnement de simulation. Par exemple, un ascenseur aura comme attribut son panneau de commandes, qui aura lui même comme attributs ses boutons de contrôle. Un attribut pourra aussi être la vitesse de déplacement vertical, tandis qu'un ascenseur pourra avoir des fonctionnalités pour ouvrir ou fermer ses portes, ou encore pour se rendre à l'étage passé en paramètre.

L'architecture BIIO est fortement axée sur la notion d'héritage de propriétés, permettant de définir des concepts génériques qui seront ensuite spécialisés pour décrire les éléments d'un environnement. Ce principe définit qu'un objet dispose de l'ensemble des constituants, attributs comme fonctionnalités, de son ou ses parents. L'unification de concepts se fait donc en créant un nouveau type d'objet qui hérite de plusieurs types prédéfinis, associant ainsi

l'ensemble de leurs capacités. La spécialisation se fait par la redéfinition dans un nouveau type d'objet de fonctionnalités disponibles dans ses parents, ainsi que par l'ajout de nouveaux constituants. A titre d'exemple, un ascenseur unifie les concepts d'objet électrique et de conteneur d'objets, tout en les spécialisant par l'ajout des constituants cités précédemment.

La figure 3.19 présente les relations de déclaration existant entre les concepts. On peut y voir qu'un objet physique gère une collection de ressources, représentant ses états internes variables. Cet objet physique est spécialisé en effecteur et en interacteur, qui sont les intervenant d'une interaction. Ces deux spécialisations disposent chacune d'une collection d'affordances, représentant leurs possibilités d'interaction respectives. L'interacteur, qui correspond aux entités autonomes, comporte aussi une collection de tâches cognitives incarnant ses comportements. L'effecteur, qui correspond globalement aux équipements, comporte quant à lui une collection de files d'attente et d'utilisation. Ces dernières, qui représentent la façon d'accéder à une interaction, disposent chacune d'une zone d'ancrage stipulant leur localisation respective. Enfin, une affordance indique les types d'interacteur et d'effecteur pouvant la réaliser, ainsi que les files d'attente et d'utilisation qu'elle implique.

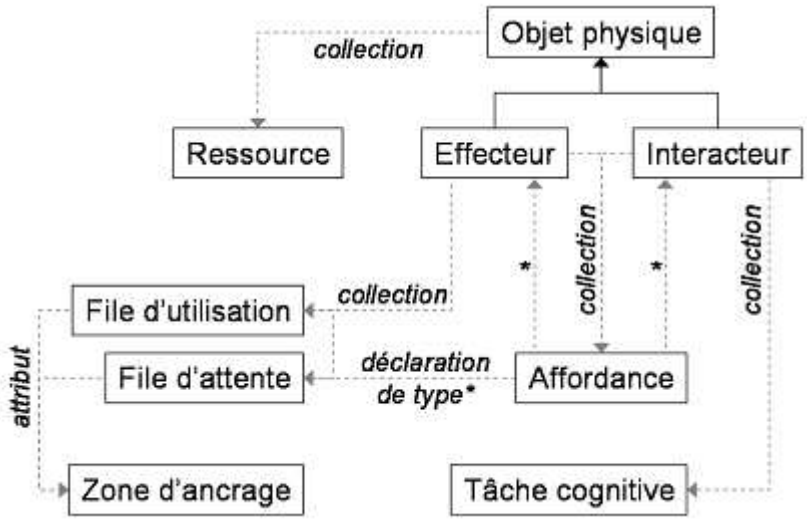


Figure 3-19 – Relations de déclaration entre les concepts de BIIO. Les flèches en pointillés représentent les relations, les flèches continues représentent un lien d'héritage [Par07].

La figure 3.20 présente les relations d'exécution existant entre les concepts. L'objet physique se charge ici de la mise à jour de ses ressources. L'effecteur se contente de la gestion des files d'attentes, les files d'utilisation ayant un fonctionnement évènementiel. L'interacteur gère quant à lui l'exécution de ses affordances et tâches cognitives, et est capable de s'abonner et de consulter les files d'attente et d'utilisation de l'environnement. Les affordances et les tâches cognitives s'influencent mutuellement, et contrôlent le

comportement de l'interacteur. De plus, elles se servent des ressources de ce dernier pour gérer leur propre fonctionnement.

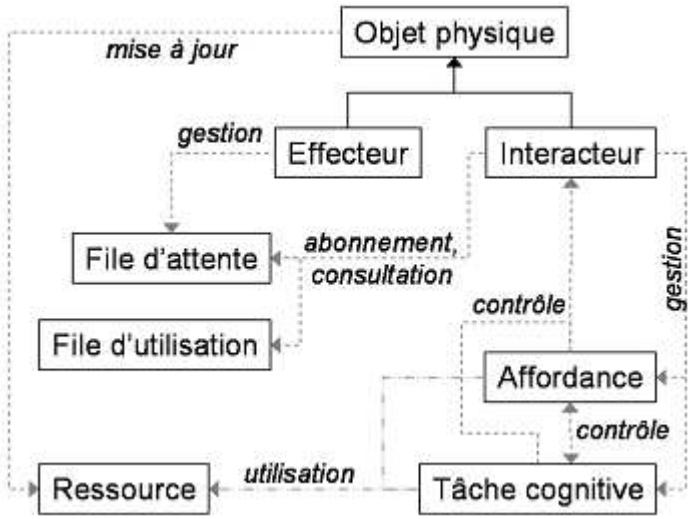


Figure 3-20 – Relations d'exécution entre les concepts de BIIO. Les flèches en pointillés représentent les relations, les flèches continues représentent un lien d'héritage [Par07].

Description des interactions

Les comportements sont décrits de la même manière que les objets interactifs, permettant ainsi une description hiérarchique. Un comportement contient quatre fonctionnalités, toutes relatives à un acteur identifié (l'entité simulée qui veut accomplir l'interaction). Premièrement, une *pré-condition dite rationnelle* valide l'utilisation du comportement vis-à-vis d'une catégorie d'objet (par exemple, puis-je utiliser un distributeur de billets). Deuxièmement, une *pré-condition dite locale* valide l'utilisation du comportement vis-à-vis d'une instance d'objet identifiée (par exemple, puis-je utiliser le distributeur de billets qui se trouve dans le hall d'entrée). Troisièmement, un *effet* qui peut potentiellement modifier l'état de l'acteur, ou de l'objet interactif. Et pour finir, une *durée d'interaction*, qui peut être variable ou fixe.

Intégration des objets BIIO

Les objets BIIO sont déclinés en deux catégories topologiques : objet statique (dont la position est figée durant la simulation), et objet dynamique (qui va pouvoir se déplacer). Les objets statiques sont directement insérés dans le graphe d'abstraction de l'environnement sous la forme de noeuds sémantiques dits *conceptuels* (figure 3.21). Ainsi, les objets statiques bénéficient automatiquement de l'ensemble des propriétés des noeuds du graphe, que ce soit au niveau des pré-calculs effectués (PVS, etc.), ou au niveau des algorithmes de parcours

applicables (comme la planification de chemin que nous verrons par la suite). De plus, une sous-catégorie des objets statiques est proposée, les objets traversables, qui pourront relier deux zones (comme pour un escalator) ou plus (un ascenseur) de l'environnement.

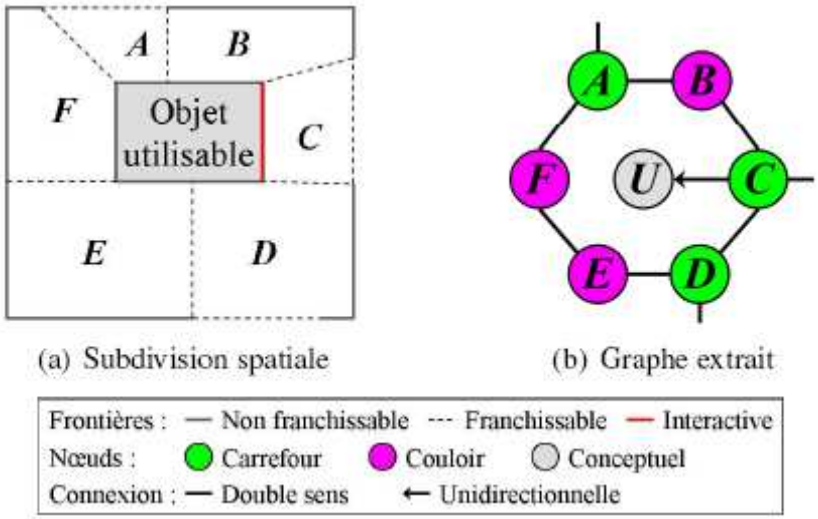


Figure 3-21 – Intégration d'un objet utilisable statique dans le graphe de l'environnement [Par07].

Concernant les objets dynamiques, Paris propose de les intégrer par référencement dans des nœuds pré-existants du graphe de l'environnement, permettant ainsi de conserver un lien avec la topologie. Il a ainsi proposé un certain nombre de procédures permettant d'obtenir un équivalent rapide et efficace aux procédures proposées par les noeuds du graphe environnemental, comme les *dPVS* (pour *dynamic PVS*) qui permettent d'obtenir très rapidement le potentiel de visibilité des objets dynamiques en se basant sur les PVS de la zone de l'environnement où ils se trouvent (figure 3.22).

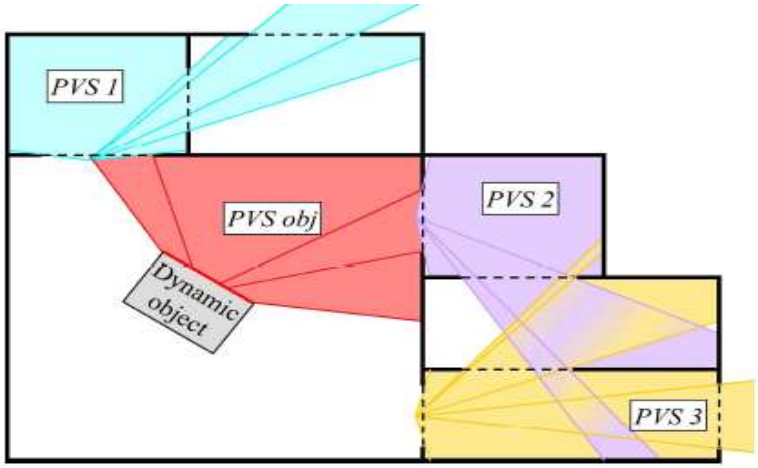


Figure 3-22 – Calcul du dPVS - le frustrum de visibilité de l'objet (en rouge) est connecté aux PVS statiques de l'environnement qui sont visibles [Par07].

3.8.4 Synthèse

S. Paris a présenté une architecture complète pour effectuer des simulations de foule d'une manière microscopique, c'est à dire en simulant chaque entité de manière autonome. Premièrement, il a proposé une représentation de l'environnement efficace en termes de calculs, obtenue de manière automatique, et comportant un ensemble d'informations utilisables à moindre coût par les entités. Comme il a proposé un moteur de description des objets interactifs de la simulation, permettant d'enrichir le potentiel comportemental des entités simulées. Il a mis l'accent concernant ces interactions sur l'intégration des objets interactifs directement dans la procédure de planification de chemin de l'humanoïde. Cette dernière, basée sur une connaissance individuelle de l'environnement, prend en compte plusieurs critères de décisions simultanément.

3.9 Conclusion

Dans ce chapitre, nous avons d'abord décrit ce qui est pour nous le meilleur type d'environnement virtuel: l'environnement informé. Ce type d'environnement permet de déplacer une partie des connaissances nécessaires au personnage dans l'environnement lui-même. Il permet aussi de donner le maximum d'informations utiles au personnage, et d'éviter ainsi un fastidieux travail d'analyse et de compréhension pour le module perceptif.

Ensuite, nous avons vu les modèles permettant d'informer des environnements de simulation. De manière générale, le lien entre le comportement et la topologie dans la prise de décision est assez peu abordé. Nous retiendrons néanmoins l'approche des objets intelligents (*smarts objects*) qui, en associant l'interaction aux objets de l'environnement, propose déjà ce lien au niveau de la description. D'autre par, le modèle STARFISH offre des perspectives séduisantes pour un couplage à des mécanismes de décision. Il dispose en effet d'un lien fort avec la topologie, grâce aux zones d'interactions, et associe une sémantique à l'interaction, grâce à ses huit primitives. Son intégration à des comportements de plus haut niveau pourrait ainsi reposer sur ces primitives, et se faire de manière quasi transparente.

Le tableau suivant récapitule les modèles présentés tout au long de ce chapitre :

Modèle	Auteur	Référence	Nature des informations	Application
Les objets intelligents	Marcelo Kallmann	[Kal98]	Interaction avec les objets	Simulation d'humanoïdes virtuels
Environ-nement Informé	Nathalie Farenc	[Far01]	Informations urbaines	Simulation d'humanoïdes virtuels dans un contexte urbain
Les annotations	Patrick Owen Doyle	[Doy04]	Connaissances spécifiques liées à l'environnement	Simulation d'humanoïdes virtuels
Ontologie	Mario Gutierrez	[Gut05]	Sémantique et fonctionnalité des objets	Simulation d'humanoïdes virtuels
	Karsten A. Otto	[Ott05]	Contenu d'un monde virtuel	Simulation d'humanoïdes virtuels
Les objets intelligents	Tolga Abaci	[Aba06]	Interaction avec les objets	Simulation d'humanoïdes virtuels
Les objets synoptiques	Marwan Badawi	[Bad06]	Interaction avec les objets	Simulation d'humanoïdes virtuels
Objets interactifs (BIIO) Champs de visibilité (PVS)	Sébastien Paris	[Par07]	Interaction avec les objets Topologie (décomposition en cellules) Planification hiérarchique prend en compte plusieurs critères de décisions simultanément comporte différentes conditions de fin, aussi bien géographiques que conceptuelles	Simulation microscopique de foule

Tableau 3.2 Récapitulatif des méthodes de représentation sémantique d'environnement.

Deuxième partie

Contribution

Chapitre 4

Modèle proposé

4 Modèle proposé

4.1 Introduction

La réalisation de l'état de l'art avait pour but de définir l'environnement virtuel, l'humanoïde virtuel et la relation entre les deux. Ces études nous ont permis de nous positionner par rapport aux systèmes existants et expliquent les choix et le modèle décrit dans cette partie. Ces choix ont porté sur la modélisation de l'environnement et sur la définition de l'humanoïde qui évoluera dans cet environnement. Notre travail se situe dans la continuité de l'évolution de la recherche en animation comportementale visant à doter les humanoïdes de capacités de décision de plus en plus évoluées. Cette autonomie des humanoïdes permet d'alléger la tâche des animateurs.

Nous allons donc présenter dans ce chapitre notre contribution. La problématique est d'abord exposée. Ensuite, nous allons donner l'explication de nos choix en ce qui concerne le modèle d'environnement ainsi que le modèle d'humanoïde. Nous continuerons par décrire notre modèle ; la façon dont nous organisons les données topologiques pour permettre une navigation optimale des entités. Nous finirons avec l'ensemble des informations associées à l'environnement.

4.2 Problématique

Nous avons vu dans l'état de l'art que la représentation de l'environnement se plaçait au centre du comportement autonome, en ayant un impact fort sur les interactions de l'agent avec l'environnement. Nous avons ainsi identifié dans la partie précédente plusieurs problématiques qu'il est indispensable de traiter pour aboutir à un modèle permettant des simulations réalistes.

Premièrement, notre modèle doit être capable d'associer une sémantique à l'environnement de simulation. Cette étape est nécessaire à tout modèle comportemental, que l'information soit spécifiée de façon globale ou directement au sein de l'environnement. Cette deuxième approche est la plus exploitable pour cette étude, grâce à sa mise en oeuvre implicite de l'aspect situé du comportement.

Le deuxième point à traiter concerne directement la représentation de ces environnements. Celle-ci doit être aussi proche du réel que possible, et permettre une extraction rapide des informations nécessaires à la circulation d'entités autonomes. Parmi ces informations on peut envisager l'ensemble des caractéristiques géométriques et topologiques

de l'environnement, mais aussi l'emplacement des lieux de l'interaction. Ainsi, le modèle d'environnement doit intégrer des zones interactives, et donc la sémantique qui y est associée, pour permettre la mise en oeuvre des aspects incarné et situé du comportement.

Enfin, le troisième et dernier point de notre modèle concerne les connaissances à fournir à l'humanoïde. Dans la première partie, nous avons présenté les différents composants d'un humanoïde (perception et décision) ; l'élément essentiel ressortant de cette étude est que la connaissance de l'environnement est nécessaire au fonctionnement de ces différents composants. Selon ces composants, la connaissance sur l'environnement va de la géométrie à des informations et des structures symboliques. Par exemple, pour l'activité de navigation dans un environnement virtuel, les informations nécessaires à la simulation de l'humanoïde sont :

- La hauteur du sol pour positionner correctement les pieds de l'humanoïde. Ces informations sont donc nécessaires à l'activité de marche.
- Le type d'un lieu influence le choix de l'itinéraire et le comportement de l'humanoïde lorsqu'il le parcourt. Si on prend l'exemple d'un environnement urbain ou le comportement du piéton est complètement différent selon qu'il soit sur un trottoir ou au milieu de la route. La théorie des affordances de Gibson [Gib86] corrobore la manipulation d'informations symboliques dans l'environnement qui seront exploitées par le modèle décisionnel. Les lieux seront caractérisés par leur configuration, leur type, mais aussi par les objets qu'ils contiennent. Dans un esprit proche de celui de Badawi et de ses objets synoptiques [Bad06], il est possible d'associer des informations symboliques et procédurales à des zones d'interaction.
- La topologie de l'environnement permet à l'humanoïde de planifier ses parcours. Ces informations sont exploitées par le modèle décisionnel de l'humanoïde qui possède donc une carte mentale de l'environnement grâce à la structure topologique.

Ces niveaux d'abstraction pour la connaissance de l'environnement seront présents dans notre modèle.

4.3 Choix du modèle d'environnement

Modèle géométrique

La scène virtuelle en 3 dimensions est un ensemble de polygones. Ce format est compatible avec la technique utilisée pour la construction du modèle topologique.

Construction du modèle

Nous avons mentionné que les travaux de Paris [Par07] utilisent la représentation exacte de l'environnement. Ces travaux sont très proches de notre travail au niveau de la structure topologique manipulée ; la grande différence est dans le mode de construction de cette structure. Motivés par les avantages du maillage de navigation, nous avons choisie d'utiliser un maillage de polygones pour représenter l'espace navigable. Ce choix de représenter la topologie de l'environnement par un maillage de navigation nous permet de manipuler l'information de l'hauteur. Ce qui implique une représentation d'environnement exacte et en 2D1/2.

Structures et informations symboliques

Farenc [Far01] propose un modèle d'environnement urbain composé d'ENVs (ENtités enVironnementales) organisées dans des structures hiérarchiques et topologiques. Notre approche est similaire ; nous avons proposé plusieurs catégories d'espaces de navigation fédérés dans une structure topologique. De la même façon que Badawi [Bad06] des espaces d'interactions avec les objets sont présents dans nos structures. Ces informations répondent aux besoins de localiser les objets interactifs dans l'environnement.

4.4 Choix du modèle d'humanoïde

Dans le cadre de ce mémoire, les deux couches de l'humanoïde qui nous concernent sont la décision et la perception. Du point de vue du modèle décisionnel, les capacités motrices sont considérées comme une boite noire à laquelle sont transmises des consignes niveau tâche (seulement la locomotion pour l'instant). L'ensemble des capacités motrices est pour l'instant réduit ; notre humanoïde n'est pas capable d'ouvrir une porte, de monter les escaliers. Cependant, avec la seule activité de locomotion contrôlable en vitesse et en orientation, il est possible, grâce au modèle décisionnel, de générer des comportements variés et réalistes. Le modèle décisionnel est fondé sur les machines d'états finis. Grâce à la hiérarchie de ce modèle, nous pouvons par exemple représenter le comportement de navigation et sa décomposition en sous tâches.

4.5 Description du modèle proposé

Le modèle de l'environnement doit permettre à l'acteur d'aller partout. Pour cela, le piéton doit connaître l'espace dans lequel il se trouve (son type, sa fonction). Nous avons jugé nécessaire de fournir à l'acteur une modélisation de l'environnement afin qu'il puisse évoluer sans avoir à reconstruire la structure de l'environnement à partir des seules informations géométriques. Disposer d'une représentation symbolique de l'environnement permet d'obtenir des temps de simulation raisonnables ; de plus, la nature de l'environnement est prise en compte pour la création de la connaissance spatiale.

Une représentation symbolique de l'environnement peut-être un ensemble de points de passages et de chemins que l'acteur doit suivre. Cependant, cette représentation n'est pas suffisante pour que l'acteur puisse naviguer partout. Nous avons donc choisi de définir l'environnement comme un ensemble de cellules convexes ; cela permet aux acteurs de passer d'une cellule à l'autre et de naviguer efficacement dans chaque cellule. Le modèle d'environnement proposé ici consiste en une représentation sous forme de maillage de navigation.

Un environnement renferme des informations topologiques et géométriques. Il présente également des associations entre les lieux (surface géométrique) et de l'information sémantique. L'information géométrique provient directement à partir du modèle en trois dimensions: la scène. Une fois la scène créée on en extrait les informations topologiques, au cours de ce processus on lui associe des informations sémantiques. Ainsi, l'idée principale est d'ajouter une couche sémantique sur une base correspondant à un modèle topologique (Figure 4.1). La couche sémantique associe aux surfaces navigables des propriétés utilisables lors d'une simulation.

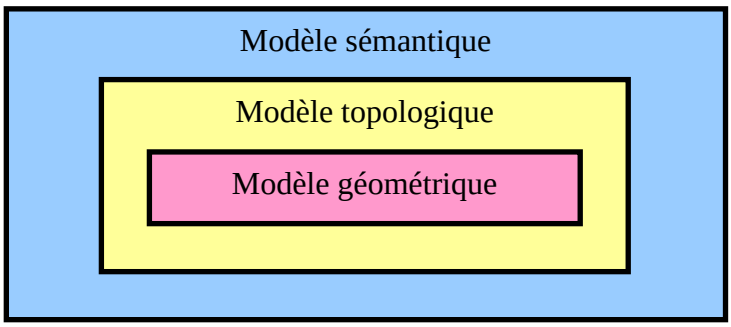


Figure 4-1 – Schéma de modélisation.

4.5.1 Structure topologique

La structure pour laquelle nous avons opté est le maillage de navigation (Figure 4.2). Elle est considérée par Tozour [Toz04] comme l'une des meilleures méthodes et la plus communément utilisée pour fournir de l'information sur l'espace navigable aux agents est de créer un maillage de formes convexes qui représente toutes les zones de l'environnement virtuel que l'agent est capable de traverser.

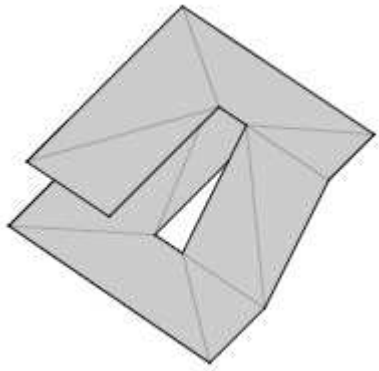
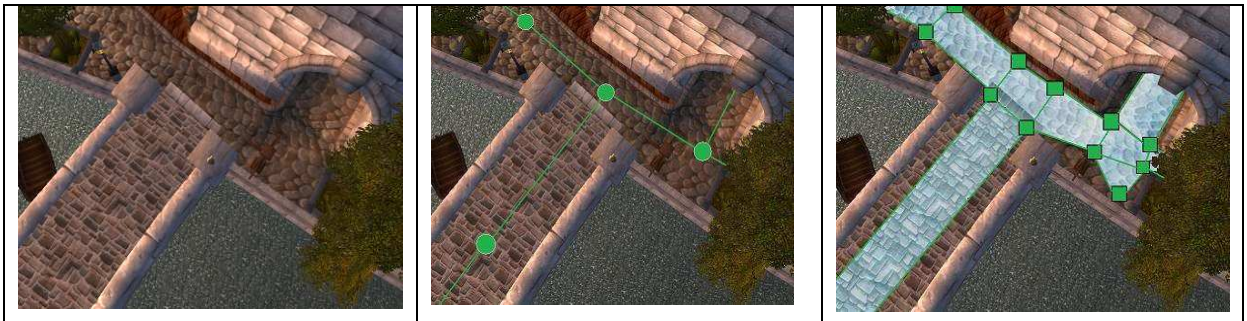


Figure 4-2 – Un maillage de navigation.

Le maillage de navigation (*Navigation Mesh* en anglais) appelé dans le domaine de la robotique *Meadow maps,* est un ensemble de polygones convexes qui décrit l'espace de navigation d'un environnement virtuel [HYD09].

4.5.1.1 Motivations

Il y a plusieurs points positifs qui nous ont motivé pour choisir le maillage de navigation qui peut se révéler avantageux pour un moteur exécutant une simulation ou un jeu:



a) Partie de Stormwind City dans World of WarCraft	b) La même zone avec une carte de cheminement	c) La même zone avec un maillage de navigation

Figure 4-3 Exemple d'un maillage de navigation.

- Un maillage de navigation est un maillage de polygones décrivant la surface où les entités peuvent naviguer. Cela signifie que les entités sont libres de se déplacer sur une surface plutôt qu'être coincé sur les arcs unidimensionnels. Il modélise le monde comme un espace continu. Un exemple d'un maillage de navigation est illustré dans la figure 4.3.c. Cette représentation donne à l'acteur synthétique beaucoup plus d'informations sur le monde qui l'entoure et soutient beaucoup plus la prise de décision. C'est une représentation exacte de l'environnement.

a) Partie de Halaa dans World of WarCraft

b) La même zone avec une carte de cheminement

c) La même zone avec un maillage de navigation

Figure 4-4 – Amélioration des algorithmes de recherche de chemin.

- L'avantage le plus fondamental du maillage de navigation est l'amélioration de la performance de recherche de chemin fournit à l'agent en réduisant l'espace libre où l'agent peut naviguer depuis des milliers ou des millions de points vers des dizaines ou des centaines de régions présentes dans un maillage de navigation. Il en résulte une amélioration du temps d'exécution pour la plupart des algorithmes de recherche de chemin.

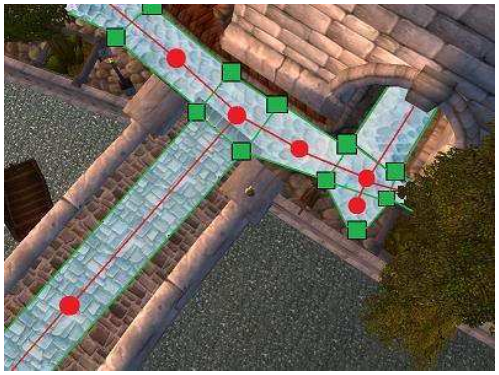
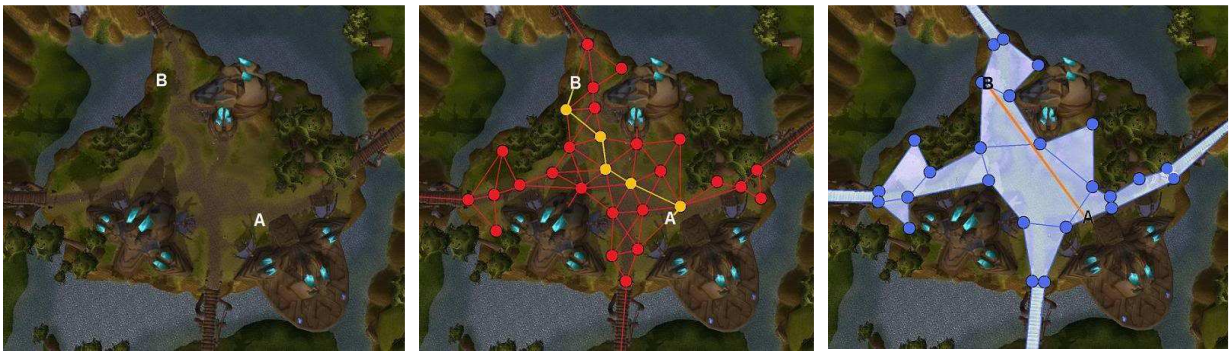


Figure 4-5 – Illustration du graphe représentant le maillage de navigation (en rouge).

- Un maillage de navigation est un graphe, tout comme un graphe de carte de cheminement, et les routines de base de recherche de chemin sont très similaires. La différence est qu'un maillage de navigation présente un polygone associé à chaque noeud du graphe figure 4.5. L'algorithme A * utilisé pour la recherche de chemin, s'applique sur un graphe, qu'il s'agisse d'une grille de cellules carrées, d'une carte de cheminement ou d'un maillage de navigation. Dans la plupart des cas, le maillage de navigation a des performances similaires à un graphe de carte de cheminement. Dans les cas où ils couvrent la même zone avec moins de noeuds (comme la figure 4.6), un maillage de navigation peut effectivement être nettement plus rapide.



a) Deux points dans Halaa dans World of WarCraft.

b) Naviguer depuis A vers B en utilisant une carte de cheminement.

c) Naviguer depuis A vers B en utilisant un maillage de navigation.

Figure 4-6 – Navigation depuis A vers B.

- Il est plus difficile de trouver des chemins réalistes avec les cartes de cheminement. En effet, le chemin entre deux points peut passer par plusieurs sommets, au lieu de passer par une ligne droite, le chemin peut être tordu, comme le montre la figure 4.6.b. Cela pourrait être partiellement résolu en carte de cheminement très dense, mais qui utilise beaucoup de mémoire. Idéalement, après avoir trouvé le chemin, il est possible de l'ajuster pour le rendre plus lisse. Le problème est, les cartes de cheminement ne dispose pas d'information en plus du graphe, ce qui rend impossible de faire l'ajustement du chemin de manière sûre et fiable.
- Le fait que le maillage de navigation représente exactement où les personnages sont autorisés à naviguer ce qui signifie qu'il est facilement possible de lisser les chemins. Tout ce qu'il faut faire, c'est de s'assurer que les splines restent à l'intérieur du maillage de navigation. La figure 4.7, montre un chemin (en rouge) et le même chemin lissée (en bleu) sur un maillage de navigation.

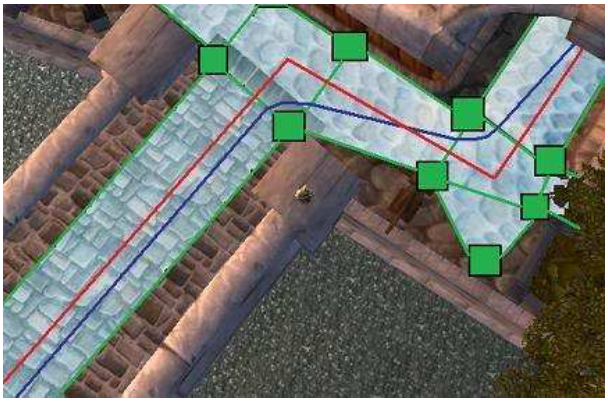


Figure 4-7 – Un chemin (en rouge) et le même chemin lissé (bleue)

- Dans un maillage de navigation, chaque polygone enregistre le paramètre "hauteur" indiquant combien d'espace libre dans le sens vertical est disponible pour ce polygone. Par exemple, le polygone passant sous le pont dans la figure 4.8 enregistre 7 unités hauteur de l'espace libre, et les entités de plus de 7 unités de haut ne peuvent pas utiliser ce polygone pour la navigation.

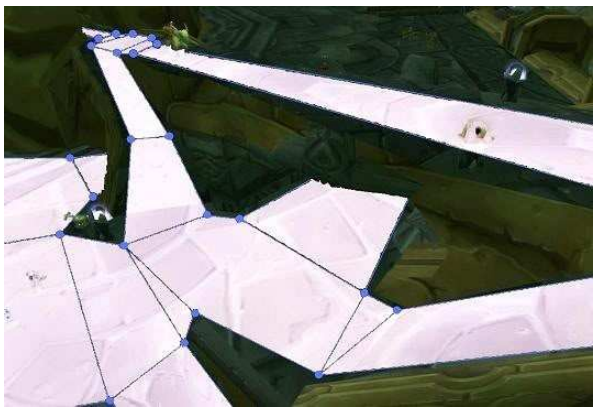
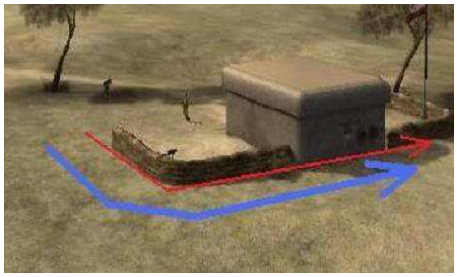


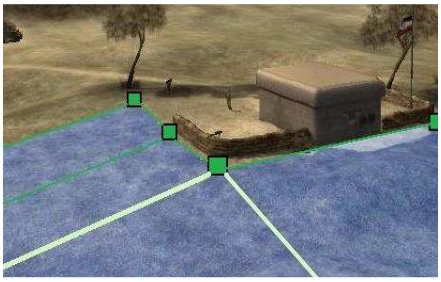
Figure 4-8 – Une partie du maillage de navigation en dessus et en dessous d'un pont dans Shattrath City.

- Une carte de cheminement ne contient généralement pas d'information sur la taille de l'entité qui peut naviguer sur le graphe. Et puisque des entités de taille différentes ne peuvent pas prendre le même chemin. Ceci peut être résolu en ayant un graphe par taille de l'entité. Ce qui implique l'utilisation d'espaces inutiles de mémoire. Pour résoudre ce problème la rechercher de chemin revient à décaler les points de passage de leurs coins figure 4.9.d.

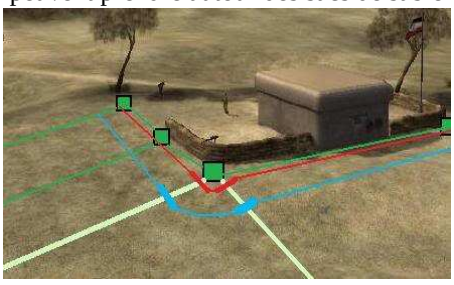
a) Un abri dans le désert



b) Les plus proches chemins possibles, q'un soldat humanoïde (en rouge) et un réservoir (en bleu) peuvent prendre autour des sacs de sable



c) Une partie d'un maillage de navigation autour de l'abri



d) Une partie d'un maillage de navigation autour de l'abri

Figure 4-9 – Un maillage de navigation peut être crée selon la taille de l'entité.

4.5.1.2 Construction du maillage de navigation

Le maillage de navigation peut être construit par décomposition à main de l'environnement virtuel. La décomposition est généralement faite en suivant une heuristique pour déterminer précisément comment la décomposition doit se produire. Elle offre d'excellentes représentations de l'environnement et une couverture étendue. Cependant, le plus grand inconvénient de cette méthode est l'exigence extrême de temps (plusieurs jours par environnement) pour bien construire une décomposition. Ainsi de meilleures méthodes sont nécessaires.

Ainsi, le maillage de navigation est construit à partir de la géométrie de l'environnement selon nombreuses méthodes. Cependant, la plupart de ces techniques de construction appartiennent à l'une des deux catégories suivantes :

Le premier ensemble de techniques utilisé pour créer un maillage de navigation est la décomposition basée sommet. En utilisant un ensemble de règles, tous les sommets fournis par la géométrie de l'environnement sont reliés les uns aux autres pour produire une série de triangles. Ces triangles peuvent ensuite être combinés selon les résultats pour formé un polygone convexe d'ordre supérieur, ceci pour réduire le nombre total de formes présentes dans le maillage de navigation. L'approche basée sommet donnent généralement une décomposition de couverture très élevée, mais peut donner lieu à des régions très petites ou

étrangement formées. L'algorithme Hertel-Mehlhorn de Hertel et Mehlhorn [HM83], est un exemple sur ce type de techniques.

La deuxième catégorie utilisée pour générer un maillage de navigation est basée sur la méthode de croissance. Une certaine forme géométrique est semée dans l'environnement, puis chaque partie de cette forme est autorisée à se croître jusqu'à ce qu'elle rencontre un obstacle. Ces parties sont ensuite connectées où elles coïncident et forment le maillage de navigation. Un exemple sur cette catégorie de techniques est la technique de Tozour ; Space Filling Volumes [Toz04].

Les méthodes basées sur la croissance fournissent des régions de forme très régulière. Par exemple la technique de Tozour ; vu qu'il utilise un carré pour créer le maillage, la technique fonctionne très bien pour des environnements où tous les obstacles sont alignés, mais pas sur des environnements avec une géométrie arbitraire ou complexe.

4.5.1.3 Méthode adoptée

La technique adoptée est celle de D. Miles et M. Mononen : Automatic Navmesh Generation via Watershed Partitioning [MM08]. Le processus général de construction du maillage est le suivant:

1. **Voxelisation** - Créer un *champ de hauteurs solide* à partir de la géométrie source.
2. **Génération de régions** - Détecter la surface supérieure du *champ de hauteurs solide* et la diviser en régions de travées contiguës.
3. **Génération de contours** - Détecter les contours des régions et former des polygones.
4. **Génération du maillage de polygones** - Subdiviser les contours en polygones convexes.
5. **Génération du maillage détaillé** - Trianguler les polygones et leurs ajouter l'information de hauteur.

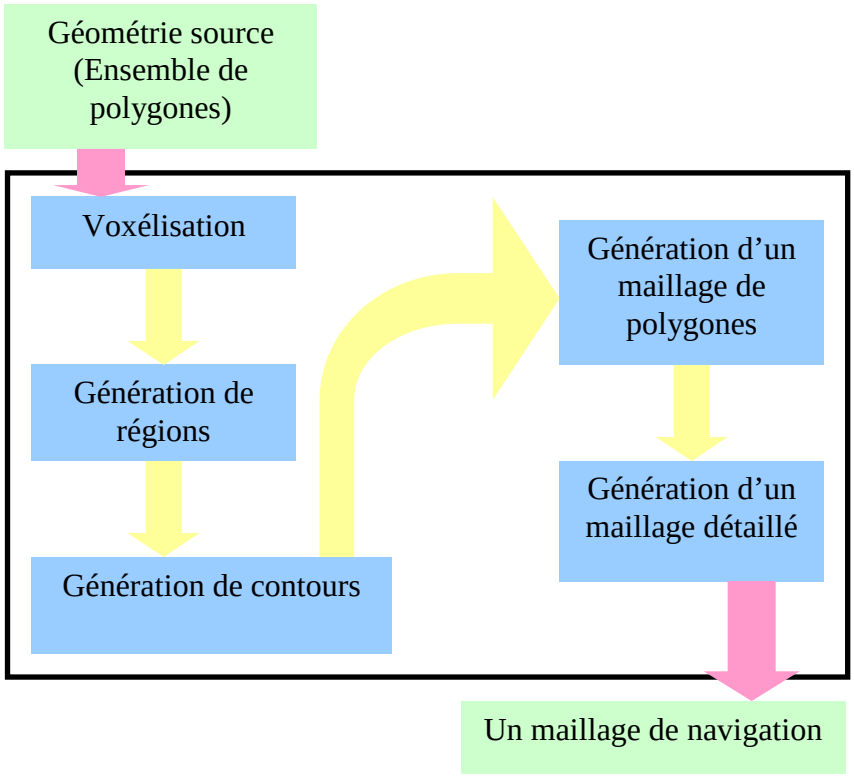


Figure 4-10 – Le processus de construction du maillage de navigation.

1) La voxelisation

La première étape dans la génération du maillage de navigation est la voxélisation. Elle consiste à détecter une boite englobant la géométrie source et créer un *champ de hauteurs solide (Solid Height Field)* pour stocker des informations sur les voxels. Ensuite, un filtrage initial de surfaces non franchissables est effectué.

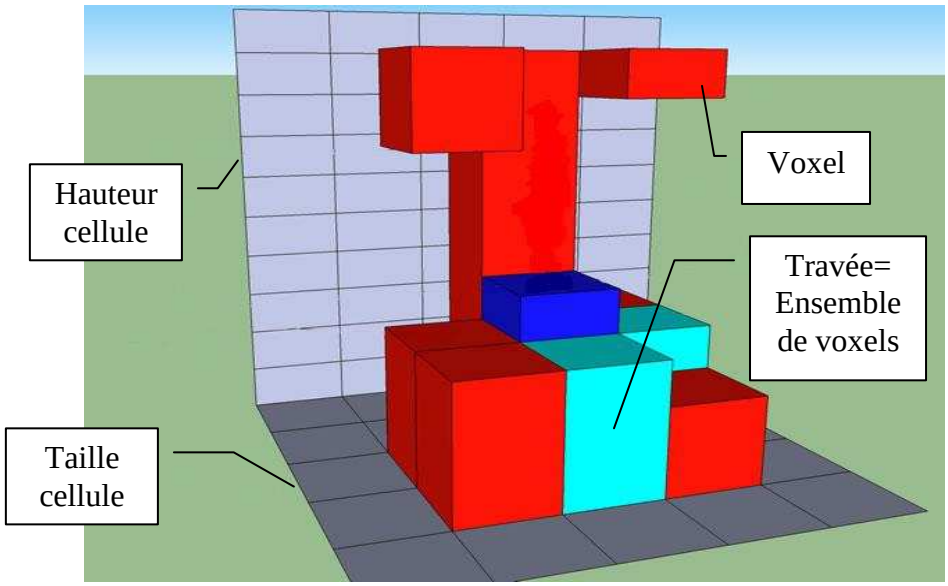


Figure 4-11 – Le champ de hauteurs solide.

Création du champ de hauteur solide

Chaque triangle dans la géométrie source est voxélisé en utilisant la voxelisation conservatrice et ajouté au champ de hauteur. La voxelisation conservatrice est un algorithme qui assure que les surfaces du polygone sont complètement couvertes par les voxels. La figure 4.12 montre un exemple d'un triangle qui a été englobé par une voxelisation conservatrice:

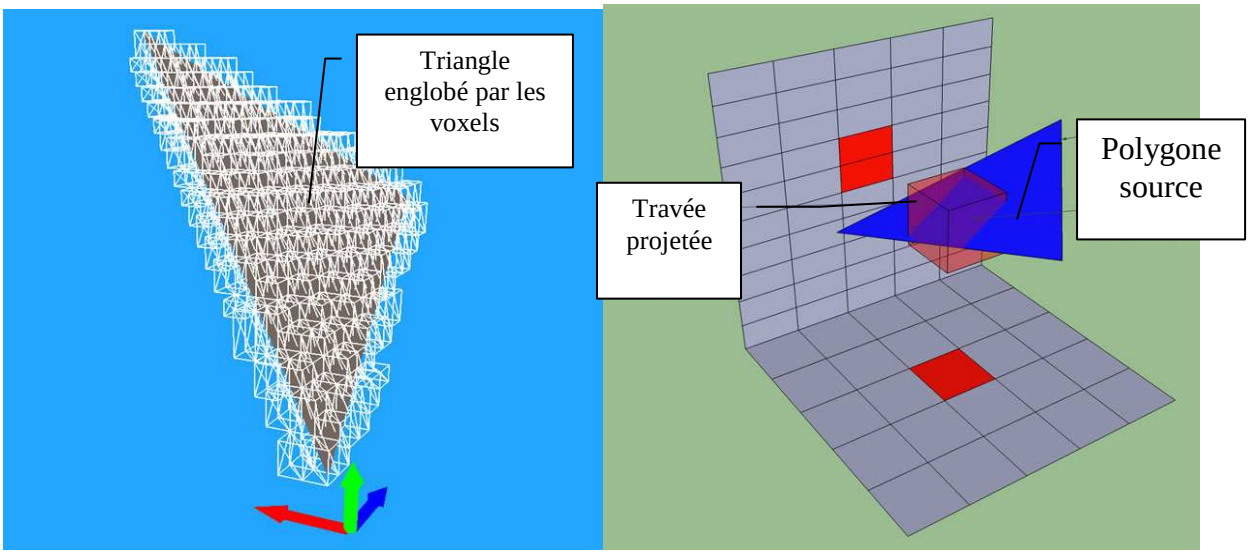


Figure 4-12 – La voxélisation conservatrice.

Filtrage initial

Après voxélisation, le champ de hauteur solide contient les travées qui englobent complètement la surface de tous les polygones de la géométrie source. Une fois la voxélisation est terminée, ces travées sont marquées comme navigables s'ils réussissent les tests suivants:

- Le sommet d'une travée est au moins à une distance minimale de la base de la travée au-dessus. (Le plus grand agent peut se tenir sur la travée sans entrer en collision avec un obstacle ci-dessus.)
- Le voxel haut de la travée représentant la géométrie avec une pente en dessous d'une valeur maximale autorisée. (La pente est suffisamment faible pour être franchissable par les agents.)
- Si le passage du haut de la travée vers le bas ; vers une travée voisine dépasse une valeur configurée, alors la travée est considérée comme une corniche et non pas navigable.

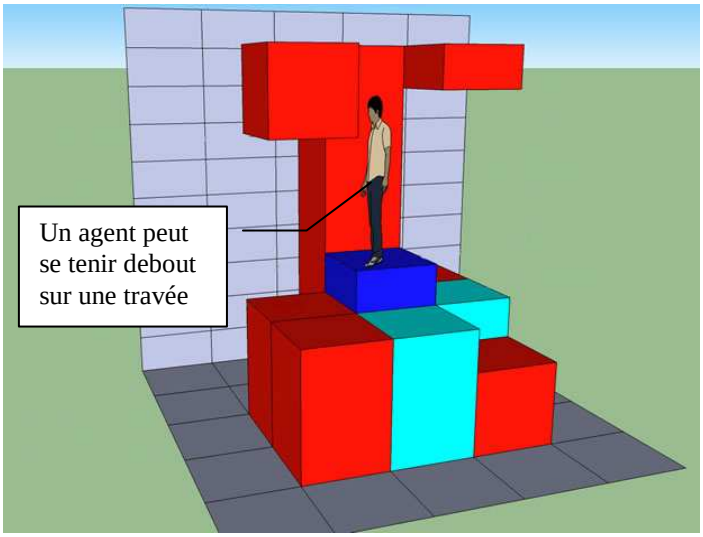


Figure 4-13 – Filtrage initiale des travées.

À la fin de cette étape, nous avons un champ de hauteurs solide qui représente la zone obstruée de la géométrie source. Un filtrage initial a été réalisé pour marquer la surface supérieure de la zone d'obstruction comme franchissable ou non.

2) Génération de régions

L'objectif de cette étape est de définir plus précisément quelle partie de la surface solide est navigable, et de séparer les zones navigables dans des régions contiguës de travées (surfaces) qui peuvent éventuellement former des polygones simples.

Créer le volume ouvert

La première étape est de transformer le champ de hauteurs solide en un champ de hauteurs ouvert (Figure 4.14), ce qui représente les surfaces pénétrables au dessus de l'espace solide.

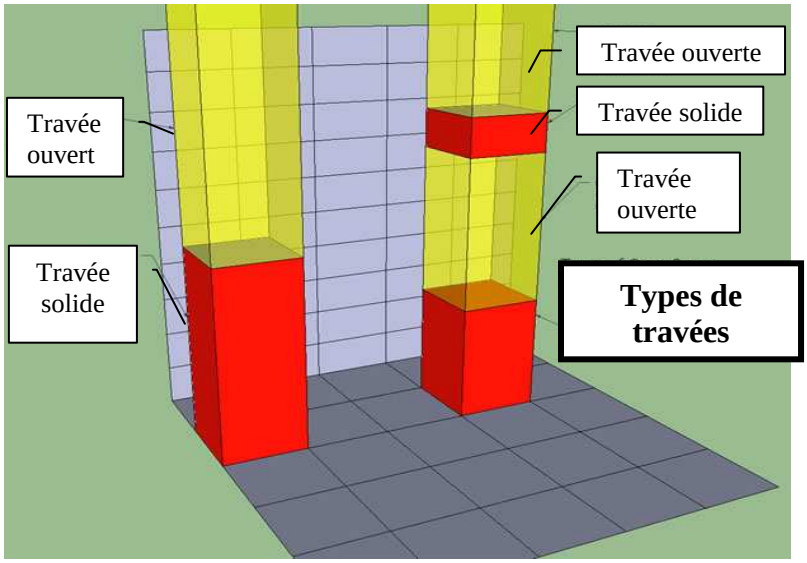


Figure 4-14 – Le champ de hauteurs ouvert.

Créer des liens de voisinage

L'étape suivante consiste à déterminer quelles sont les travées qui forment une surface de travées contiguës. Ceci est accompli grâce à la création de liens de voisinage selon les axes. Les voisins selon les axes sont les quatre voisins décalés le long des axes de la largeur/profondeur.

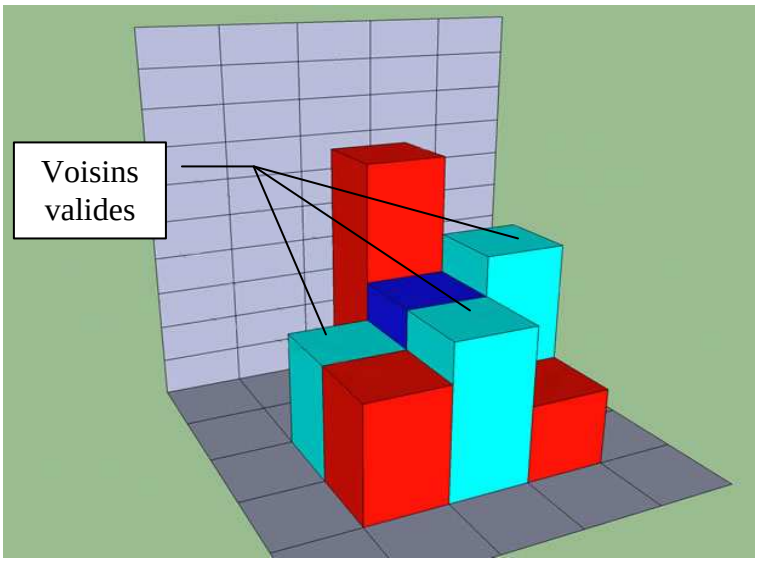


Figure 4-15 – Les voisins selon les axes.

Une travée dans la colonne voisine est considérée comme une travée voisine, si les deux conditions suivantes sont remplies:

- La montée ou descente entre les étages des deux travées est inférieure à une valeur définie. Cela permet aux surfaces telles que les marches d'escalier et bordures d'être détecté en tant que voisins valides (Figure 4.15).
- L'espace ouvert entre la travée actuelle et la voisine est assez grand pour être parcouru (Figure 4.16).

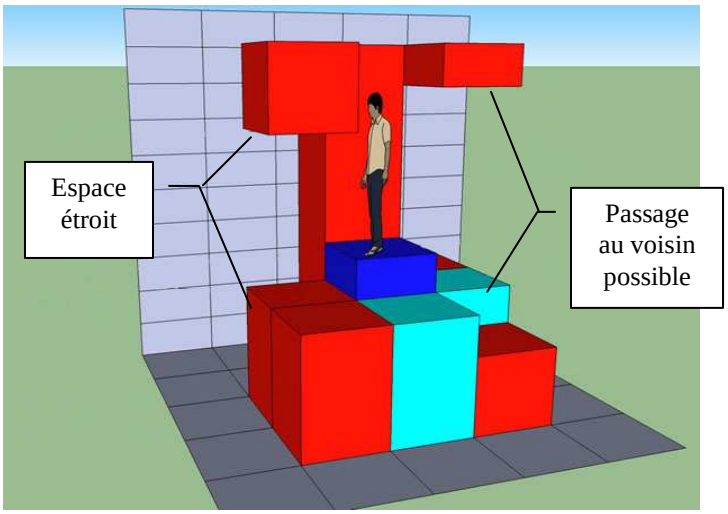


Figure 4-16 – Vérification de l'accessibilité des voisins.

Création des champs de distance

Un champ de distance consiste à estimer la distance entre chaque travée et la travée frontalière la plus proche. Cette information est nécessaire pour l'algorithme qui génère les régions. Une travée frontalière est une travée qui a moins de huit voisins. Les travées frontalières représentent la limite entre la surface navigable de la géométrie source et les obstacles (murs) ou un espace vide.

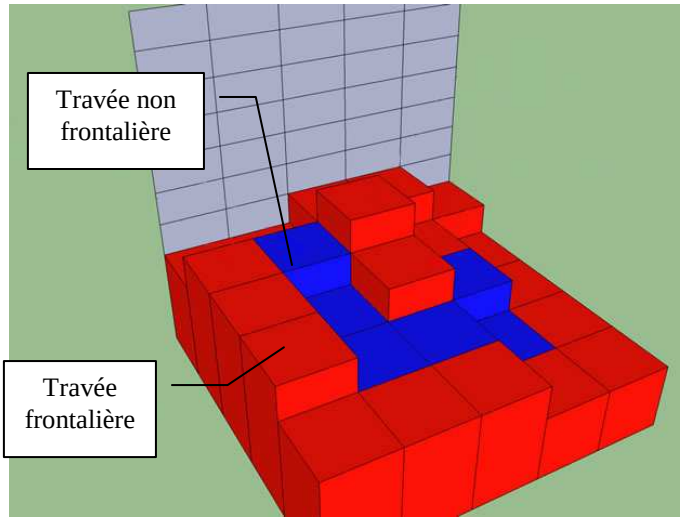


Figure 4-17 – Les travées frontalières.

Créer les régions

L'algorithme des bassins versants *Watershed Partionning* est utilisé pour la création initiale de régions. Par analogie aux bassins versants, les travées qui sont le plus loin d'une frontière représentent les points les plus bas du bassin. Une travée frontalière représente le plus haut niveau possible d'eau.

La boucle principale commence par le point le plus bas dans le bassin, puis s'incrémente avec chaque itération jusqu'à ce que le niveau d'eau maximal autorisé soit atteint. Au cours de chaque itération de la boucle, les travées qui s'étendent au dessous du niveau d'eau actuel sont localisées et une tentative est faite soit à les ajouter aux régions existantes ou de créer de nouvelles régions. Au cours de la phase d'expansion des régions, si une travée nouvellement inondée dépasse les frontières d'une région existante, elle est généralement ajoutée à la région. Toute travée nouvellement inondée qui survit à la phase d'expansion de régions est utilisée comme semence pour la création de la nouvelle région.

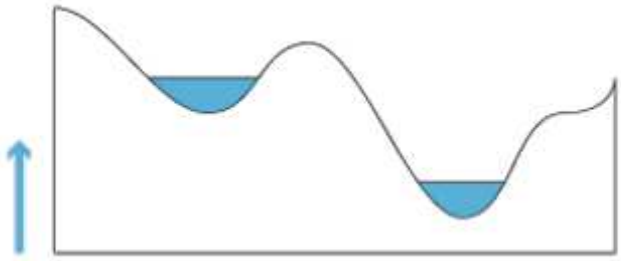


Figure 4-18 – Les bassins versants.

À la fin de cette étape nous avons un champ de hauteurs ouvert composé de régions qui représente la surface navigable de la géométrie source.

3) Génération de contours

Il s'agit de la première étape dans le processus de changement de l'espace de travail depuis l'espace voxel vers l'espace vectoriel.

Recherche des bords des régions

Au cours de cette étape, chaque bord est classé selon qu'il soit un bord région ou un bord intérieur. Les bords régions sont des bords qui s'étendent sur le voisin qui est dans une autre région. Les bords internes sont des bords qui s'étendent sur le voisin qui est dans la même région. Pour chaque travée, tous les voisins selon les axes sont vérifiés. Si un voisin n'est pas dans la même région que la travée en cours, le bord est marqué comme un bord région.

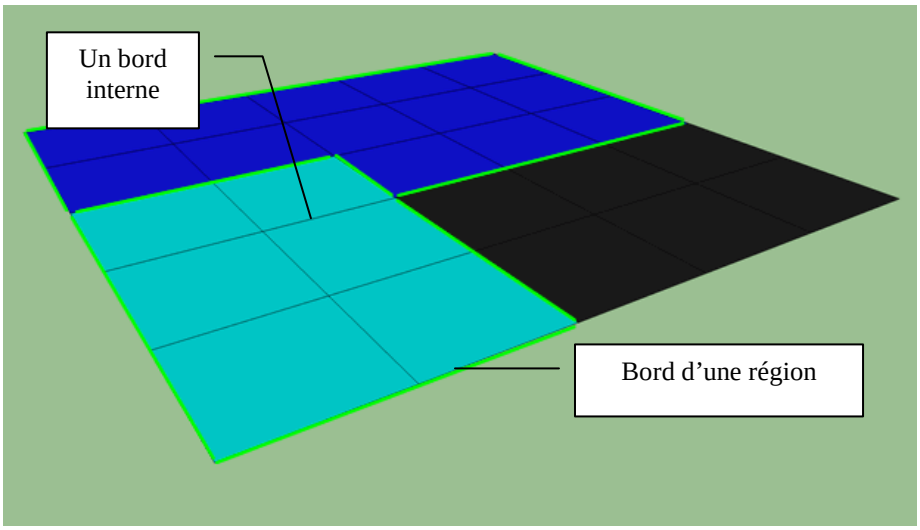


Figure 4-19 – Catégorisation des bords.

Trouver le contour des régions

Les travées sont parcourues pour trouver les bords de régions pour créer les contours.

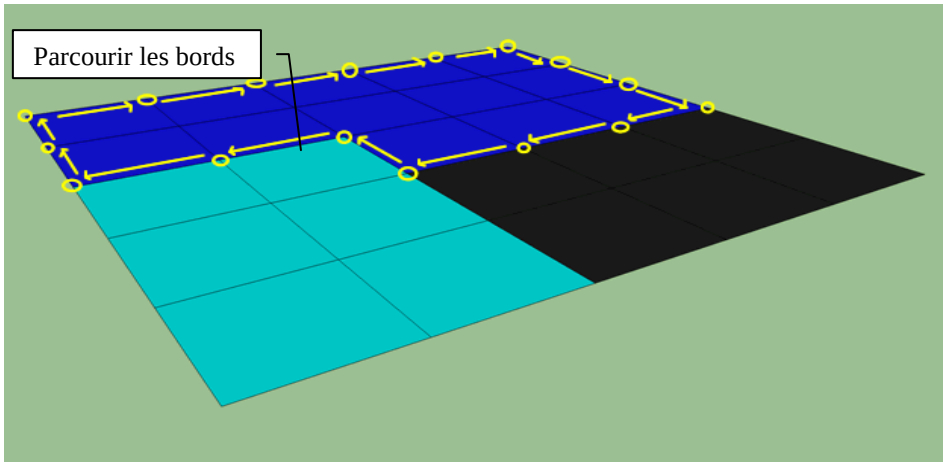


Figure 4-20 – Création des contours.

Simplifier les contours

Il existe deux types de segments de contours. Les segments qui représentent le portail entre deux régions voisines, et les segments qui représentent la bordure d'un espace «vide» appelée «région nulle». La simplification des segments consiste à se débarrasser de tous les sommets, sauf les sommets obligatoires. Les sommets obligatoires sont les sommets au bord des portails des régions.

À la fin de cette étape, la surface pénétrable est représentée par des polygones.

4) Génération du maillage de polygones

Cette étape consiste à subdiviser les polygones formés par les contours en un maillage de polygones convexes. Ceci est accompli en utilisant une triangulation adaptée pour les polygones, puis la fusion des triangles dans le plus grand polygone convexe possible.

Triangulation

La triangulation est effectuée par le parcours des segments des contours et, pour chaque groupe de trois sommets, on détermine si un triangle interne valide peut être formé. Deux algorithmes sont utilisés pour déterminer si un groupe de trois sommets forment un triangle interne valide. Le premier algorithme supprime les partitions qui se trouvent entièrement à l'extérieur du polygone. Si la partition est à l'intérieur du polygone un algorithme est utilisé afin de s'assurer qu'ils ne coupent pas les arêtes des polygones existants.

Fusionner dans un polygone convexe

La fusion ne peut se produire qu'entre les polygones créés à partir d'un seul contour. Le processus fonctionne comme suit:

1. Trouver tous les polygones qui peuvent être fusionnées.
2. De cette liste, sélectionner les deux polygones ayant la plus longue arête partagée et les fusionner.

3. Répéter jusqu'à ce qu'il ne reste plus de fusions autorisées.

Deux polygones peuvent être fusionnées si toutes les conditions suivantes sont remplies:

- Les deux polygones se partagent un bord.
- Le polygone résultant sera toujours convexe.
- Le polygone résultant n'aura pas plus d'arêtes que permis par un nombre donné.

Informations de voisinage

La dernière étape est de parcourir tous les polygones du maillage et générer les informations de voisinage. L'algorithme cherche d'autres polygones qui ont un segment avec les mêmes sommets.

À la fin de cette étape, la surface navigable est représentée par un maillage de polygones convexes.

5) Génération d'un maillage détaillé

Si les surfaces navigables du maillage source sont projetées sur le plan xz et recouvertes avec le maillage généré à l'étape précédente, les deux correspondent assez bien. Mais dans l'espace en 3 dimensions le maillage peut ne pas suivre la hauteur du contour du maillage source de façon adéquate, voir Figure 4.21.Cette étape ajoute les détails hauteur telle que le maillage détaillé correspondra à la surface du maillage source selon tous les axes.

Le patch de hauteurs

Afin d'ajouter les détails de hauteur, nous devons être en mesure de déterminer si la surface d'un polygone est trop loin de la hauteur de la travée à partir de laquelle il a été dérivé. Le patch hauteur est utilisé pour cette raison. Il contient la hauteur prévue pour chaque cellule de la grille du champ de hauteurs ouvert que le polygone coïncide avec.

Ajouter les détails de hauteurs

Les principales étapes de cette étape sont les suivants. Pour chaque polygone:

1. Echantillonner les bords du polygone. Ajouter des sommets à chaque bord qui s'écarte le plus d'une valeur donnée à partir des données du patch de hauteurs.
2. Effectuer une triangulation de Delaunay du polygone.
3. Echantillonner la surface interne du polygone. Ajouter des sommets si la surface s'écarte le plus d'une valeur donnée à partir des données du patch de hauteurs. Mise à jour de la triangulation pour chaque nouveau sommet.

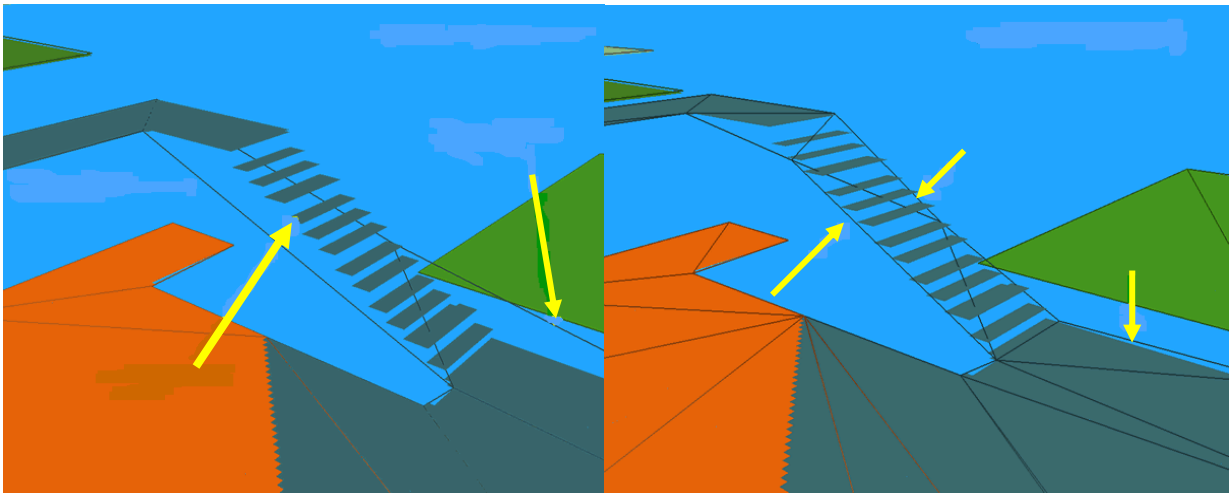


Figure 4-21 – Ajout des détails de hauteur.

À la fin de cette étape, la surface navigable est représentée par un maillage de triangles correspondant aux contours hauteur de la géométrie source.

4.5.2 Planification de chemin

Ce procédé a pour but d'extraire un chemin reliant la position actuelle de l'agent vers une destination donnée. Nous avons présenté dans la section précédente la manière de représenter les parties statiques de notre environnement, les calculs de chemin se font sur cette représentation. Il existe pour cela différentes méthodes telles que le parcours de graphe ou la descente de gradient pour les champs de potentiels. Puisque notre méthode de représentation de l'environnement est le maillage de navigation donc la structure topologique résultante est un graphe de triangles. Pour calculer le chemin entre deux points nous aurons besoins d'un algorithme de parcours de graphes. On propose d'utiliser l'algorithme A* vu sa rapidité calculatoire.

4.5.2.1 Méthode proposée

La méthode peut être divisée en deux étapes principales :

Étape 1 Étant donné deux points P1 et P2, un parcours de graphe est réalisée sur le graphe du maillage de navigation, en définissant le plus court canal (selon le graphe) reliant P1 et P2. Ce processus repère d'abord le triangle contenant P1, et applique ensuite l'algorithme A * sur le graphe du maillage jusqu'à ce que le triangle contenant P2 est trouvé. Si l'ensemble du graphe est parcouru et P2 n'est pas atteint, le triangle le plus proche de P2 est choisi.

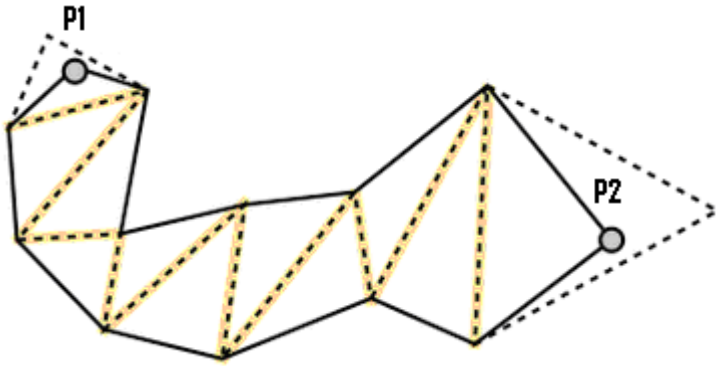


Figure 4-22 – Canal obtenu à l'aide de l'algorithme A*.

Ainsi, l'algorithme utilisé est A*: à partir d'un noeud initial et final, une évaluation heuristique est utilisée. Elle correspond à l'estimation du meilleur chemin passant par un noeud. Cette estimation permet de classer les noeuds et leur donner un rang pour les visiter.

Pour choisir un noeud, l'algorithme A* a besoin donc d'une heuristique pour estimer la distance entre ce noeud et le noeud final. Il a aussi besoin du coût de passage d'un nœud à un autre. Ce coût est en général dépendant de la distance entre le noeud de départ et celui choisi. Et c'est le cas de la méthode que nous proposons de suivre. La distance est calculée entre les milieux des segments, voir figure 4.23.

Étape 2 Le canal obtenu est équivalent à un ensemble de polygones triangulés, et donc la façon la plus simple pour traverser un ensemble de triangle consiste à utiliser les milieux des bords des triangles comme des nœuds constituant le chemin final, voir figure 4.23. Mais, il est clair que le chemin obtenu n'est pas optimal. L'algorithme de l'entonnoir Funnel Algorithm [Cha82] peut être appliqué afin de déterminer le plus court chemin joignant P1et P2 dans le canal. Cet algorithme est brièvement décrit dans la section qui suit.

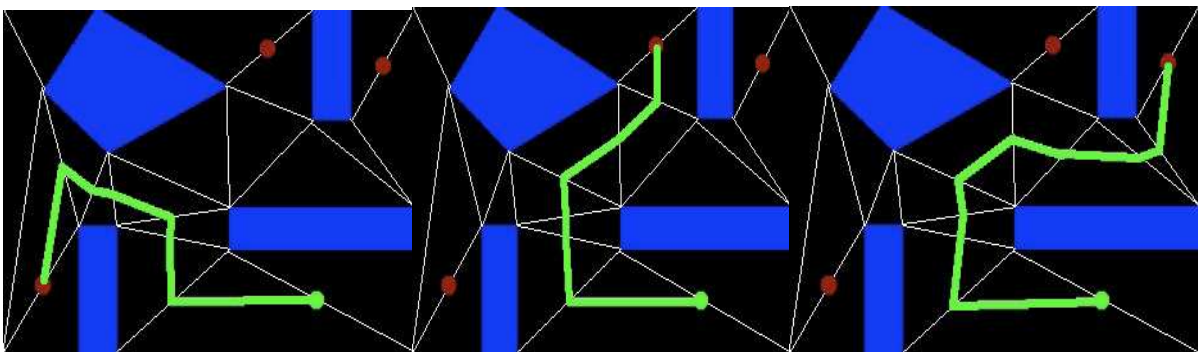


Figure 4-23 – Utilisation des milieux des bords des triangles.

4.5.2.2 Funnel algorithm

Funnel algorithm [Cha82], la traduction littéraire l'"algorithme de l'entonnoir", est un algorithme qui sert à déterminer le plus court chemin à travers un canal.

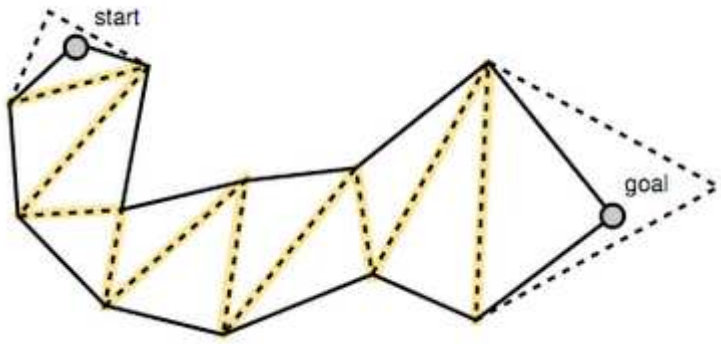


Figure 4-24 – Un canal de triangles.

La figure 4.25 montre les six étapes de l'algorithme. Chaque fois que nous traitons un nouveau segment portail (les lignes en pointillés surlignés en jaune), nous avons d'abord à:

- Vérifier si les sommets de gauche et de droite sont à l'intérieur de l'entonnoir courant (délimité par les lignes bleues et rouges), si ils le sont, nous déplaçons simplement l'entonnoir (A→D).
- Si le nouveau sommet de gauche est en dehors de l'entonnoir, ce dernier n'est pas mis à jour (E→F)
- Si le nouveau sommet de gauche est sur le segment droit de l'entonnoir (F), on ajoute le sommet droit de l'entonnoir comme un nœud du chemin et il va remplacer le sommet principal de l'entonnoir, puis redémarrez l'algorithme à partir de (G).

La même logique est valable pour le segment droit. Cette opération est répétée jusqu'à ce que tous les segments portails soient traités.

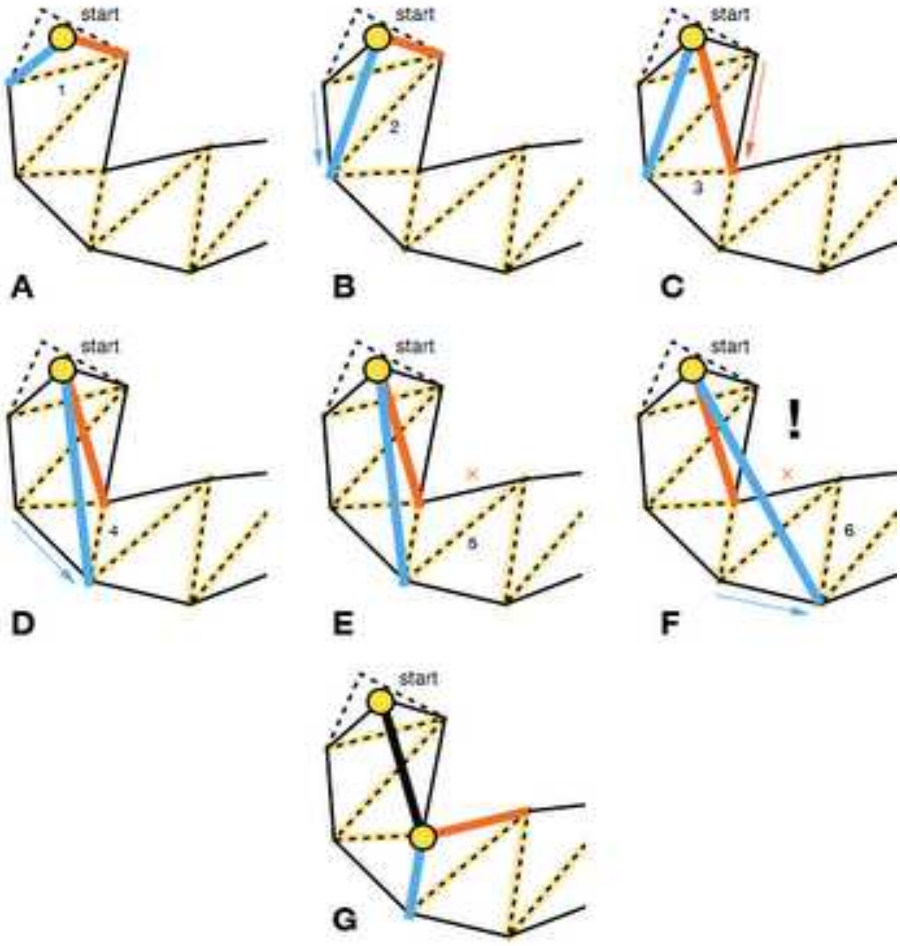


Figure 4-25 – L'algorithme de l'entonnoir.

4.5.3 Représentation sémantique

Avec un environnement informé, la navigation des humanoïdes peut être améliorée, notamment au niveau de la planification de chemin et de la simulation du comportement. En effet, lorsqu'un agent navigue dans le monde virtuel, son comportement varie en fonction de l'environnement. De plus, il interagit avec des objets tels que les ascenseurs, les portes… ainsi il devient important pour simuler un comportement émergent plus réaliste, de prendre en compte des informations sur l'environnement.

Prenons l'exemple des passages piétons. Lorsque l'humanoïde arrive au niveau d'un passage piéton, il va changer de comportement. Ce nouveau comportement va respecter les étapes logiques pour croiser la chaussée ; il va attendre le feu vert, puis il va vérifier qu'il n'y a pas de voiture qui gène pour traverser, enfin, il va traverser. Suivant ce principe, on propose d'identifier les lieux pour permettre des comportements différents pour les humanoïdes en fonction de l'endroit où ils se trouvent.

La décomposition de l'action telle que « la marche sur un trottoir » implique l'identification des endroits appelés « trottoirs » (emplacement dans l'espace et la vérification de la concordance de l'action de marche qui est associée à l'entité mobile) et l'animation du corps dans la scène virtuelle en utilisant un modèle de marche. Ainsi on peut demander à un humanoïde d'aller au cinéma ou d'aller faire du shopping et l'humanoïde pourra tout seul trouver un cinéma ou des magasins et se diriger vers eux. C'est un exemple ou l'identification des lieux influence la planification de chemin.

En caractérisant les lieux, on peut ainsi adapter le comportement de l'humanoïde en fonction de son emplacement. Ces caractéristiques supplémentaires vont permettre de traiter des déclarations comme « Allez au parc » ou des définitions spécifiques de comportement ou d'action comme « dans un parc, on lit, on joue ».

Notre modèle sémantique de l'environnement correspond à un ensemble de zones environnementales définissant une base de données. Afin d'exécuter la simulation, les surfaces que nous utilisons doivent être caractérisées, ceci implique une décomposition fine de la scène.

Nous distinguons deux types de zones environnementales en fonction de l'activité de l'humanoïde: les zones de circulation et les zones d'interaction. En effet, un humanoïde évoluant dans un monde virtuel soit il communique avec d'autres humanoïdes, soit il interagit avec des objets, soit il circule tout simplement dans cet environnement. Les deux activités interaction avec les objets et circulation nécessitent l'existence de l'humanoïde dans le lieu de l'environnement ou il va exécuter l'une de ces deux activités.

4.5.3.1 Les zones de circulation

Ce sont les lieux par lesquels un humanoïde doit passer pour arriver à une destination donnée. Ces espaces sont caractérisés par leur type; Le mode de circulation des humanoïdes dépend des espaces traversés ; suivre un trottoir, traverser une route, déambuler dans un parc, sont les principaux exemples.

Les zones de circulation sont modélisées par un ou plusieurs polygones au sol (un maillage de polygones). Les zones de circulation se construisent par le dessin d'un volume correspondant à la forme de la zone. Ensuite, étant donné que les relations hiérarchiques n'étant pas pour l'instant gérées, on caractérise chaque zone par son type : rue, passage piéton, gazon …etc. Le maillage de navigation correspondant à cette zone va être généré et intégré au maillage de navigation de l'environnement.

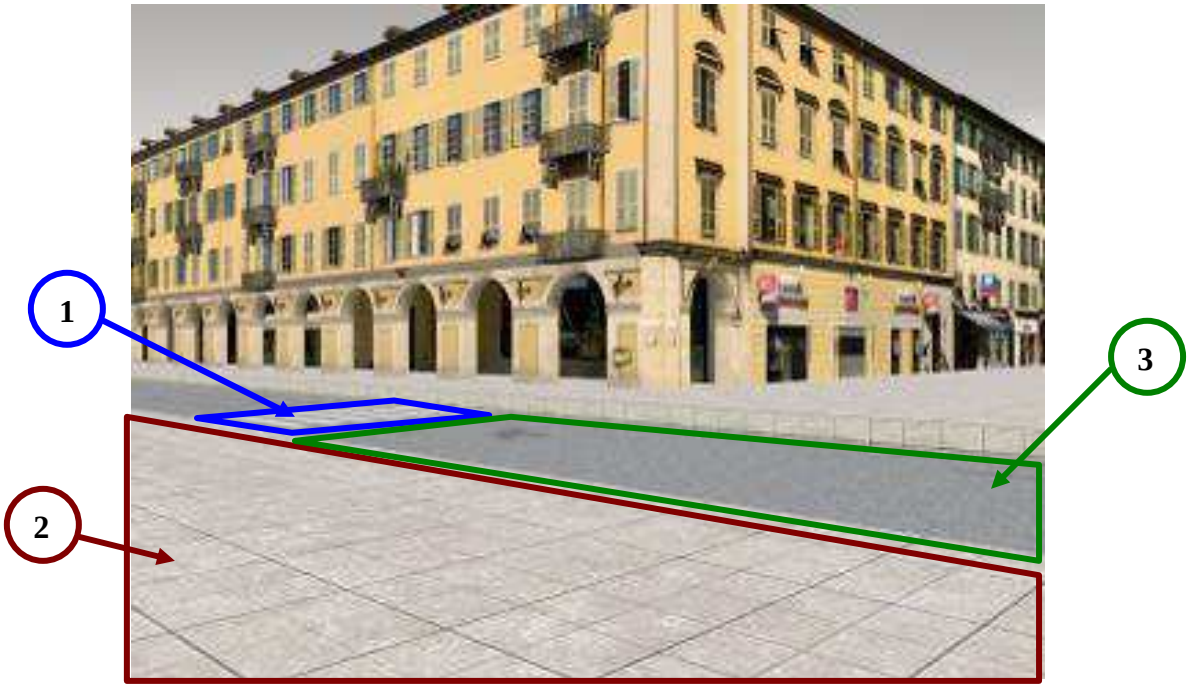


Figure 4-26 – Exemples de zones de navigation (1) passage piéton (2) trottoir (3) rue.

Donc, le mode de création des zones de navigation est manuel. Elles sont dessinées par le concepteur.

4.5.3.2 Zones d'interaction

De nombreux objets interviennent lors de la navigation des humanoïdes. Prenons l'exemple d'une porte. Lorsqu'un humanoïde arrive prés d'une porte, il doit être capable de l'ouvrir pour continuer son chemin.

Pour mémoire, lorsqu'un humain veut interagir avec un objet il essaie de déterminer sa fonctionnalité au travers d'indications visuelles de l'objet qu'il perçoit. Ces propriétés sont appelées affordances [Gib86]. M. Badawi a prouvé qu'il est possible d'effectivement placer ces informations dans le monde lui-même. Pour cela il a utilisé des surfaces interactives. Il a classé ces surfaces en deux types:

1. Surface d'Interaction: ce sont des surfaces faisant partie de la géométrie de l'objet. Elles indiquent les parties de l'objet prenant part à l'interaction.
2. Surface d'Influence: ces surfaces n'appartiennent pas à la géométrie de l'objet. Elle décrit l'espace entourant l'objet affecté par le processus d'interaction.

Ce qui nous intéresse c'est les surfaces d'influence qui indiquent l'espace dans lequel l'humanoïde doit se trouver pour interagir avec l'objet. La figure 4.27 montre une surface

d'influence positive associée à un siège. L'humanoïde doit se placer dans la zone marquée en jaune pour s'asseoir dessus.

Figure 4-27 – Surface d'influence.

Donc, nous finirons cette partie dédiée à la description de l'environnement avec la façon dont nous y intégrons les objets interactifs physiques. Notre objectif est que chaque objet physique définit soit géolocalisé, i.e. il connaît sa position dans l'espace, mais aussi le noeud du graphe topologique qui lui correspond. Le but premier de cette géolocalisation est de rendre les objets accessibles aux algorithmes basés sur notre définition topologique, dont notamment la planification de chemin.

Une catégorie d'objets interactifs est ici prise en compte : les objets statiques dont la localisation spatiale est figée au cours d'une simulation. Il y a une deuxième catégorie d'objets, les objets dynamiques pouvant se déplacer au cours d'une simulation, voire même apparaître ou disparaître ; c'est parmi nos perspectives le traitement de ce type d'objets.

Les objets statiques sont intégrés au graphe topologique sous la forme de surfaces d'influences ; i.e. pour chaque objet on ajoute au graphe topologique les nœuds représentant les cellules convexes qui représentent la surface d'influence de l'objet. Un objet pourra donc être pris pour cible d'une planification de chemin simplement en prenant pour destination le noeud qui correspond à sa surface d'influence, et sera implicitement évité lors de la navigation de par l'intégration de sa surface aux obstacles de l'environnement.

4.5.4 Modèle de l'humanoïde

Notre modèle d'humanoïde repose sur la fameuse pyramide comportementale de A. Newell (Figure 5.2), en proposant un modèle pour les deux premiers étages. Détaillons chacun des étages de la pyramide tel qu'il se présente dans notre modèle.

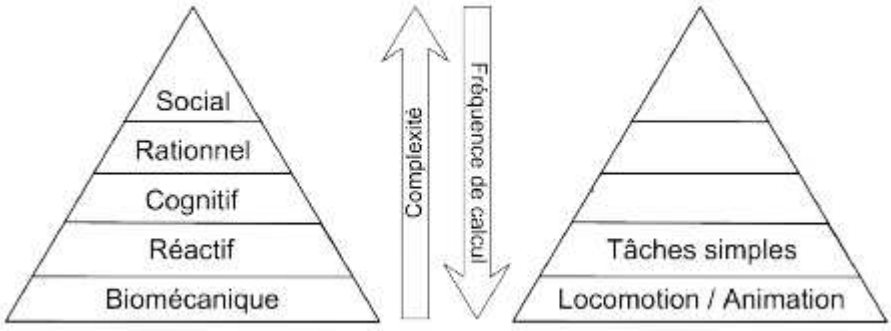


Figure 4-28 – A gauche, la pyramide comportementale originelle. A droite, l'équivalent de la pyramide comportementale dans notre modèle. Au centre, une indication des variations de complexité et fréquence de calcul suivant l'étage décisionnel.

Biomécanique : La base de la pyramide est constituée des tâches les moins conscientes pour un humanoïde. Ce sont les modules de locomotion et d'animation qui incarnent la biomécanique de l'humain virtuel dans notre modèle. Ceux-ci ne prennent aucune décision complexe, ils se contentent de suivre les directives données par l'étage supérieur de la pyramide, telles que la direction et la vitesse du mouvement.

Réactif : Le premier étage de la pyramide est constitué de tâches simples, atomiques, pour l'humanoïde. Ces tâches sont disponibles quel que soit l'humanoïde simulé, et constituent leur base comportementale. Nous pouvons citer dans les tâches de cet étage les modules de planification de chemin, ou d'évitement de collision.

4.5.4.1 Planification de chemin

Le comportement réactif proposé est la planification de chemin. Celle-ci permet d'évaluer un déplacement global, tout en répondant à deux objectifs différents : rejoindre une position identifiée, rejoindre une zone interactive identifiée. Ce dernier objectif permet ainsi de situer la prise de décision relative à l'interaction.

Atteindre un lieu

Ce premier mode correspond à une planification de chemin classique, visant à atteindre un lieu identifié dans l'environnement. Ce lieu est donc connu a priori, et communiqué à la procédure.

Choisir une interaction

L'agent autonome désire effectuer une interaction. Cette procédure va ainsi planifier un chemin dans l'environnement, à la recherche de la zone d'interaction autorisant l'interaction désirée.

4.6 Conclusion

Nous avons présenté dans ce chapitre notre représentation topologique de l'environnement et les informations s'y rapportant.

Nous avons proposé ainsi une représentation exacte en 2D1/2 permettant de rester fidèle à la description d'origine. Nous avons aussi introduit des zones spécifiques permettant d'identifier les zones de circulation comme par exemple une route, un trottoir, et les zones d'interaction comme par exemple une porte, un ascenseur…etc. Ce processus est semi automatisé, le modèle topologique étant capable d'effectuer l'extraction du maillage de navigation directement depuis la définition géométrique. Par contre, la désignation des zones environnementales (circulation, interaction) nécessite l'intervention du concepteur.

L'intérêt majeur de ce type de représentation vient des informations que l'on peut y associer. Nous proposons ainsi la typologie des lieux, qui pourrait être utilisés par nos agents autonomes avec un très faible coût. Ces informations nous permettent donc de synthétiser et d'organiser les données environnementales, pour une utilisation à la fois par le modèle d'agent autonome, afin de prendre ses décisions, mais aussi pour produire des résultats de simulation pouvant être réalistes.

Pour finir, nous avons présenté la méthode mise en oeuvre afin de situer l'interaction, par l'introduction des zones interactives directement dans notre graphe topologique. Comme nous le verrons dans le chapitre suivant, cet aspect est primordial, permettant aux algorithmes manipulant l'environnement (dont surtout la planification de chemin) de prendre directement en considération l'interaction.

Pour conclure, nous avons fortement axé cette représentation sur le réalisme comportemental, essayant de conserver au maximum l'information.

Chapitre 5

Implémentation et résultats

5 Implémentation et résultats

5.1 Introduction

Nous avons proposé dans la partie précédente un modèle de l'environnement qui permet d'atteindre un certain niveau de crédibilité du comportement humain. Néanmoins, une dernière étape est nécessaire afin de concrétiser ce travail. En effet, ce modèle ne peut être utilisé sans être intégré dans une application. Du nom de TopoSyn, cette application concrétise le travail présenté et démontre sa faisabilité.

Nous commencerons, donc ce chapitre 5, par présenter de façon générale cette application qui intègre le modèle proposé. Nous détaillerons ensuite, chaque partie en suivant un exemple qui illustrera leur principe de fonctionnement.

5.2 TopoSyn

TopoSyn est l'acronyme de **Topo**logie **Syn**optique. C'est un nouvel outil qui permet la définition de la représentation topologique et sémantique d'un environnement virtuel. Au cœur de TopoSyn des zones synoptiques qui contiennent des informations qui décrivent la manière de se comporter dans ces zones. Ces informations sont disponibles à travers deux composants :

- Un maillage de navigation qui représente la topologie de l'environnement.
- Des zones environnementales interactives qui font partie de la topologie de l'environnement et représente le couche sémantique du modèle proposé.

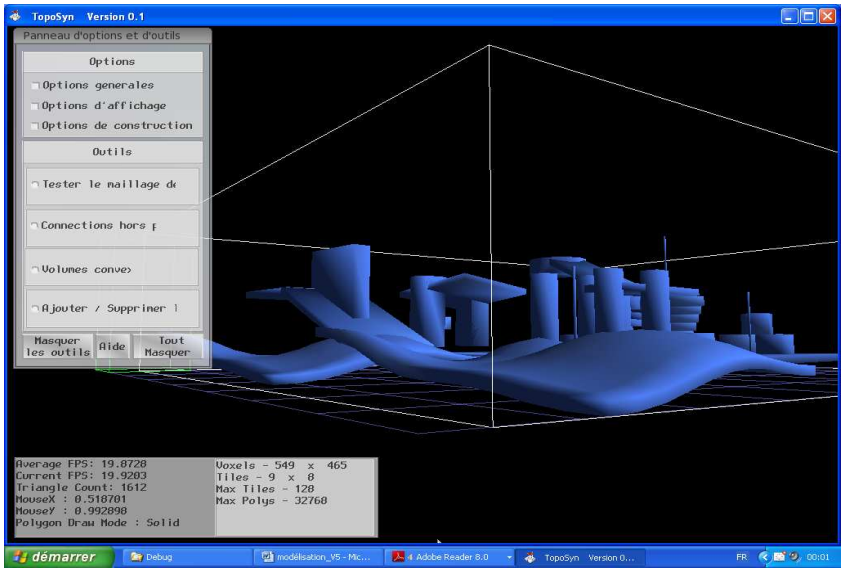


Figure 5-1 – Modélisation d'un environnement avec TopoSyn.

5.3 Vue d'ensemble

5.3.1 Les modèles utilisés

Rappelons tout d'abord les modèles utilisés dans cette architecture logicielle. Celle-ci intègre bien entendu l'ensemble des modèles proposés dans la contribution, mais aussi d'autres modèles non abordés dans ce travail.

Premièrement, la représentation de l'environnement sous forme de graphe topologique est couplée à un algorithme de détection de voisinage. Ce dernier nous permet de déterminer rapidement l'ensemble des entités voisines et visibles d'une entité de référence. Nous utilisons une grille régulière, formée de cellules carrées en deux dimensions, nous avons choisi les grilles régulières du fait de leur simplicité de mise en oeuvre et de leur rapidité d'exploitation.

Deuxièmement, l'humain virtuel nécessite un mécanisme gérant sa locomotion. Ce processus correspond à la couche la plus basse de la pyramide de A. Newell (confère Section 1.4.3, page 12), représentant la biomécanique de l'humanoïde. Nous utilisons pour cela des mouvements capturés, que nous avons choisis à partir d'une base de données de mouvements capturés fournie par Carnegie-Mellon Graphics Lab[8]. Pour adapter ces mouvements capturés à la morphologie de l'humain virtuel ainsi que de les mélanger et les ajuster, nous avons utilisé 3DS MAX (Figure 5.2). Notre choix s'est tourné vers ce modèle car il produit des mouvements crédibles.

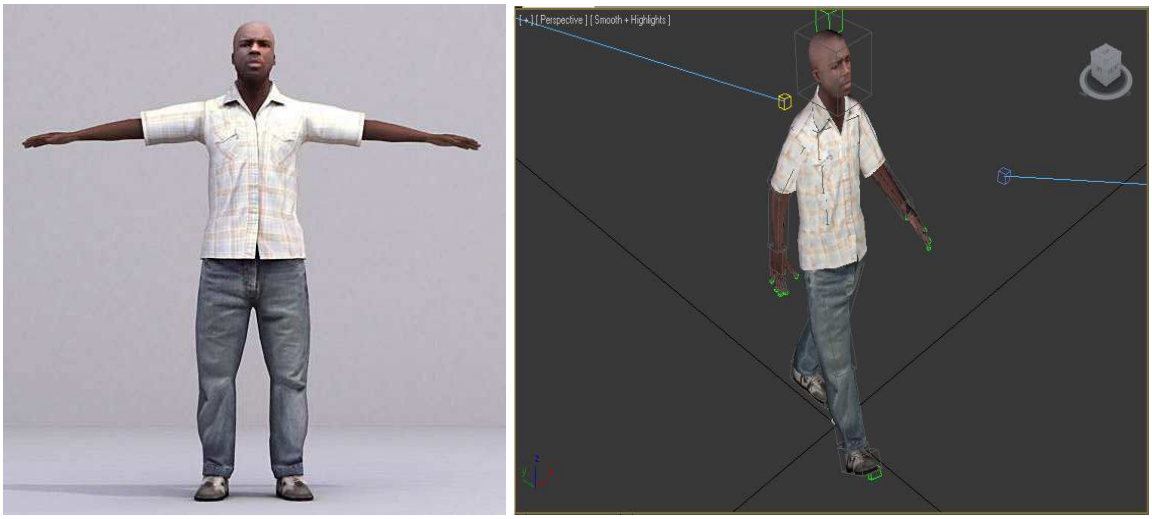


Figure 5-2 – Modèle de l'humanoïde

[8] http://mocap.cs.cmu.edu

5.3.2 Le langage de programmation

TopoSyn a été réalisé dans l'environnement de programmation Visual C++. Nous avons choisi cet environnement pour d'abord les qualités offertes par le C++, telle que la programmation orientée objet, la polyvalence, la performance et, surtout pour sa compatibilité avec le moteur de rendu Ogre3D [Ogra]. Le C++ est le langage choisi dans la plupart des applications de simulation et jeux par ordinateur.

5.3.3 Librairies logicielles utilisées

5.3.3.1 Ogre 3d

OGRE [Ogra] signifie Object-Oriented Graphics Rendering Engine, soit Moteur de Rendu Graphique Orienté-objet en français.

OGRE est entièrement écrit en C++ et de part sa conception, il permet aussi bien d'utiliser DirectX qu'OpenGL. Il intègre les dernières possibilités de ces deux librairies et est en constante évolution. Par contre, OGRE est SEULEMENT un moteur 3D, il faut rajouter des modules pour gérer le son, la physique, le réseau…

Historique

Le projet Ogre est né en 1999 de l'esprit d'un certain Steve Streeting, programmeur connu sous le pseudonyme de Sinbad et qui a eu l'idée de construire un moteur 3D indépendant de la plateforme et de l'API utilisée (Direct3D ou OpenGL). Le projet est rapidement créé sur SourceForge au début de l'année 2000 et a grossi progressivement.

La version 1.0.0 du moteur est sortie en février 2005 sous le nom de code AzaThot. Les versions 1.x se succèdent au fil des ans et de façon régulière, toujours avec un nom de code désignant une créature mystique.

En avril 2010 est sortie la version 1.7.1 dénommée Cthugha, qui en profite pour changer de licence et passer en MIT en lieu et place de la LGPL utilisée jusqu'alors.

Avantages

- Multi plateforme : OGRE fonctionne aussi bien sous Windows, Linux ou MAC. Compatible OpenGl et DirectX.
- Très performant : il gère les effets graphiques les plus récents et affiche un Frame Rate très satisfaisant.
- Offre une architecture 'professionnelle' : son design est tel qu'il peut parfaitement s'intégrer a des applications graphiques professionnelles. Grâce à la POO et sa modularité, il offre de grandes possibilités d'évolution.

- Très bien documenté : OGRE offre une API de très bonne qualité, un manuel complet et des tutoriaux. Le forum est aussi très actif.

Formats utilisés par Ogre3d pour les objets 3D

Mesh

Le fichier mesh est un fichier binaire spécifique à Ogre3D contenant l'ensemble des informations pour créer une entité graphique. Il contient la position de tous les sommets et triangles à générer par le moteur. Ce fichier est obtenu à partir d'un fichier qui provient d'une exportation d'un modeleur 3D tel que 3DSMax, Cinema 4D, Blender…etc.

Skeleton

Le skeleton est un fichier binaire spécifique à Ogre contenant l'ensemble des informations pour gérer les animations d'un mesh. Toutes les animations sont contenues dans ce fichier et ce dernier est lié au mesh du même nom. Ce fichier est obtenu à partir d'un fichier qui provient d'une exportation d'un modeleur 3D tel que 3DSMax, Cinema 4D, Blender…etc.

5.3.3.2 OgreMax Scene Exporter

Les exportateurs sont des plugins de modeleur 3D qui écrivent le maillage et le squelette d'animation dans un format de fichier compréhensible par OGRE. Ces fichiers sont respectivement au format .mesh et .skeleton.

OgreMax Scene Exporter [Ogrb] est le plugin du modeleur 3DS Max utilisé pour l'exportation des scènes de format compréhensible par 3DS Max vers des scènes de format compréhensible par Ogre3d. Nous avons utilisé la version 2.1.2 du 24 Septembre 2009.

5.3.3.3 Cegui

Crazy Eddie's GUI [Ceg] est une bibliothèque permettant de gérer des interfaces graphiques dans des contextes OpenGL, DirectX, Irrlicht et Ogre3D. Chaque interface est décrite sous forme de fichier XML et il est possible de dessiner très simplement des écrans en utilisant des outils comme le CEGUI Layout Editor. La bibliothèque est orientée objet, écrite en C++.

5.3.3.4 CELayoutEditor

CELayoutEditor [Cel] est développé par Crazy Eddie's GUI System. C'est un éditeur de fenêtre pour CEGUI. La version que nous avons travaillée avec est la version 0.7.1.

5.3.4 Logiciels utilisés

5.3.4.1 3DS MAX

Anciennement connu sous le nom "3D Studio," 3ds Max est un programme de modélisation 3D, d'animation et de rendu de la division Media and Entertainment d'Autodesk, Inc, San Rafael, CA[9] . Largement utilisé dans les domaines des jeux interactifs, des effets visuels pour les films et des modèles de conception industrielle. L'application comprend un module d'animation qui utilise la cinématique inverse, qui relie les composants afin qu'ils se déplacent ensemble, en ajoutant à l'effet de mettre un personnage à la vie.

5.3.5 L'architecture

L'architecture logicielle proposée s'organise sous la forme de trois modules intégrés au sein d'une même application (Figure 5.3) :

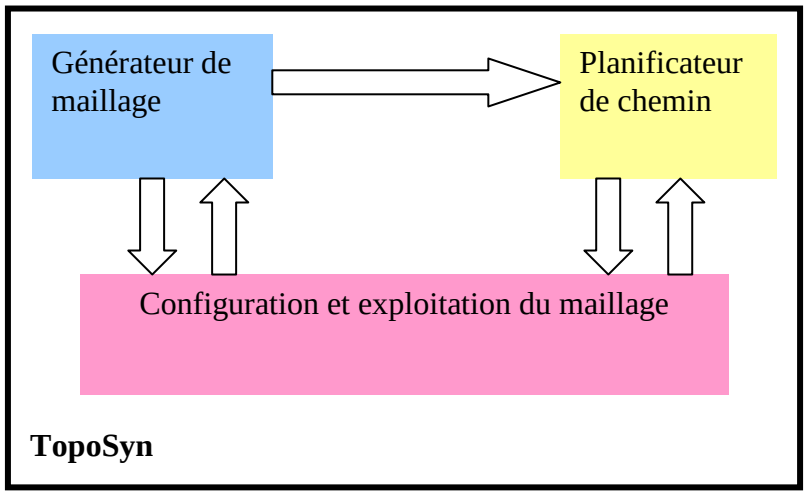


Figure 5-3 – Architecture logicielle de TopSyn.

- **Générateur de maillage :** Ce premier module sert à créer le maillage de navigation (Figure 5.4).
- **Planificateur de chemin :** Ce second module permet de calculer les chemins.
- **Configuration et exploitation du maillage :** Ce dernier module permet de configurer le maillage et d'exploiter la représentation créée pour dérouler une démonstration (Figure 5.5).

[9] www.discreet.com

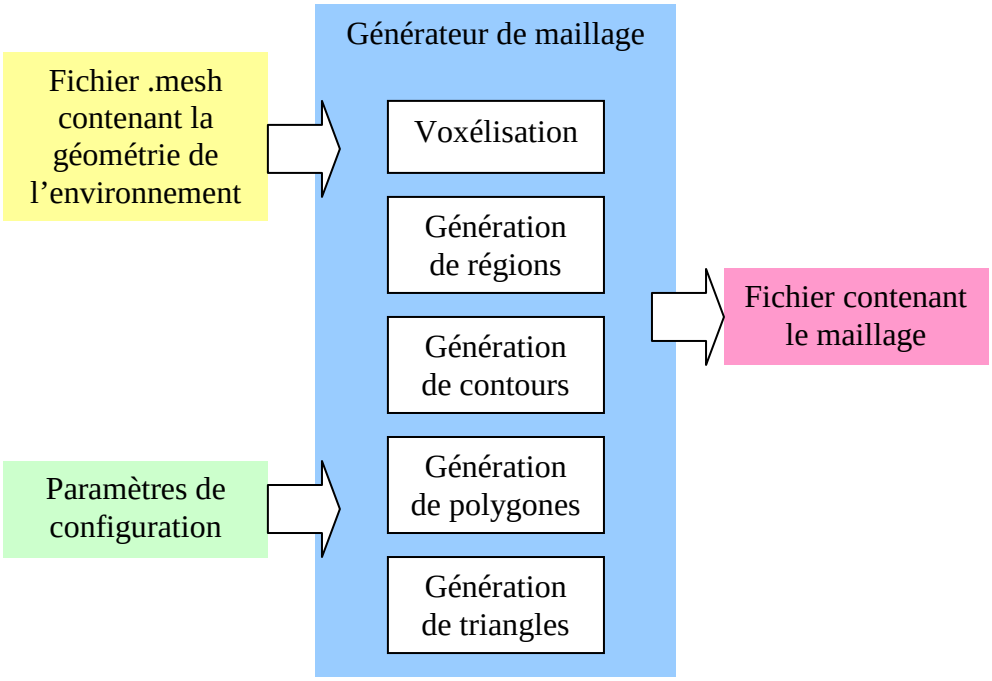


Figure 5-4 – Générateur de maillage.

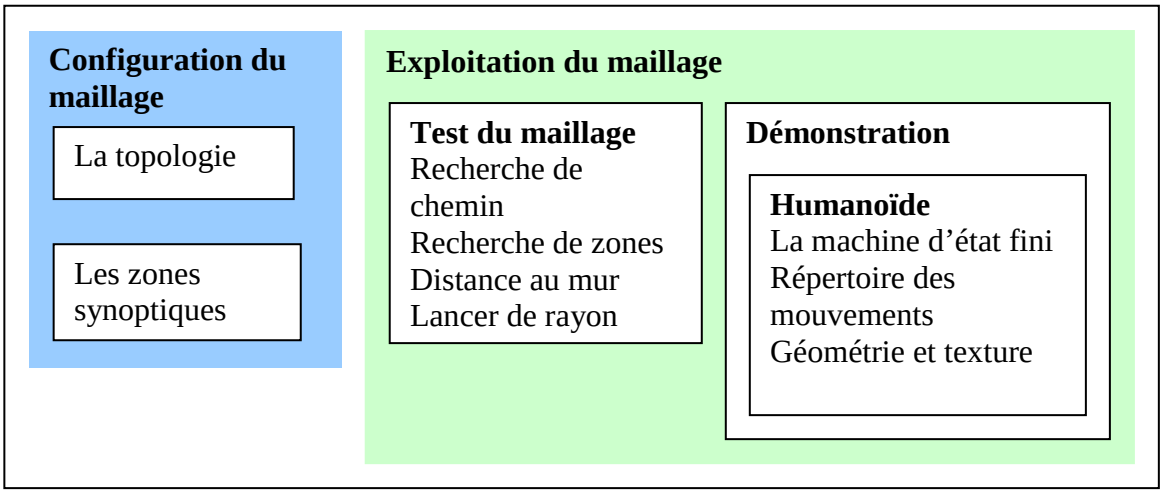


Figure 5-5 – Configuration et exploitation du maillage.

5.4 Modélisation de l'environnement

En effet, 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 démonstration, 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 objets. L'environnement va de plus fournir un accès aisé et rapide à un ensemble de zones spécifiques, tels que une rue, un passage piéton, un trottoir…etc.

Nous allons illustrer le fonctionnement de notre application TopoSyn en prenant l'exemple de l'environnement représenté dans la figure 5.6.

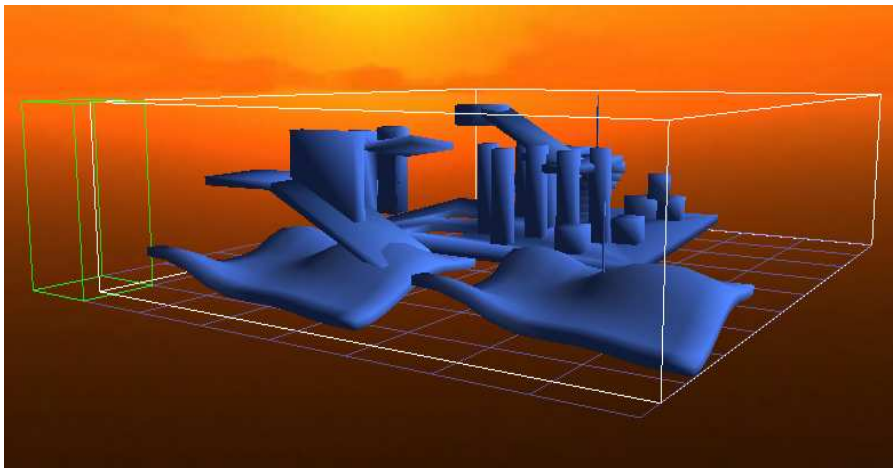


Figure 5-6 – La géométrie source

5.4.1 La représentation topologique

La première étape de l'architecture logicielle concerne la représentation topologique. Le fonctionnement de cette étape est fortement basé sur le maillage de navigation. Permettant à ce module de proposer une présentation topologique de l'environnement. Ce module se présente sous la forme de boîtes de dialogue automatisant la création du maillage de navigation. On peut voir sur les figures suivantes certaines de ces interfaces, dont :

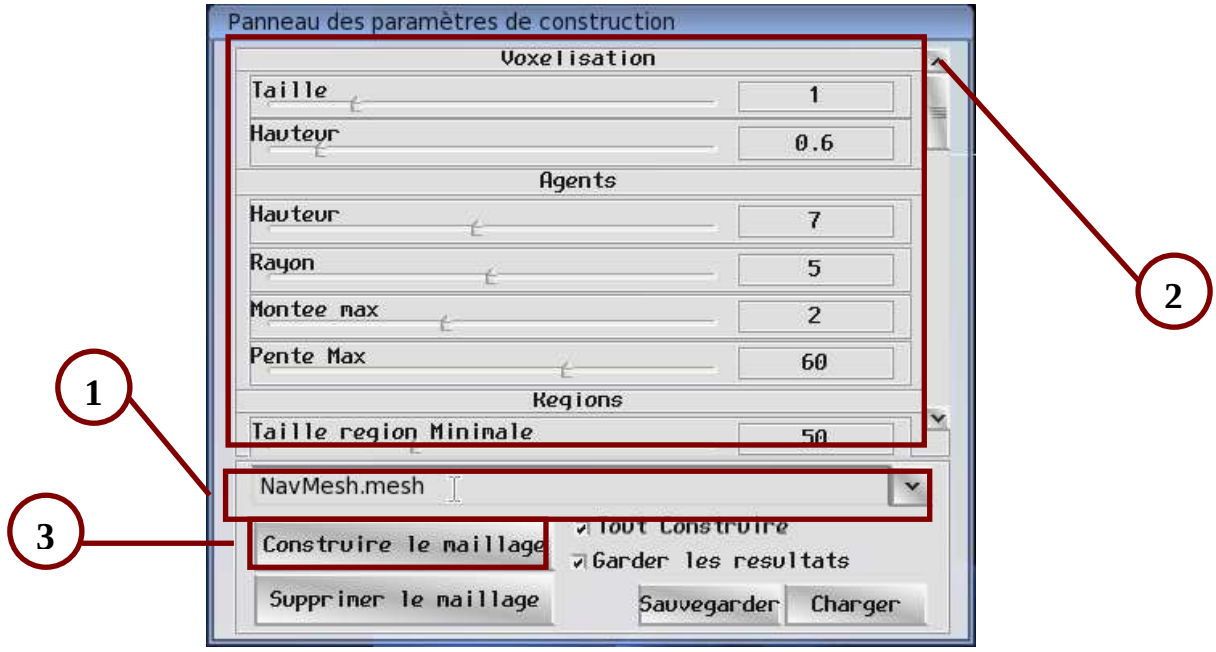


Figure 5-7 – Le panneau des paramètres de construction du maillage de navigation.

1. Sélection de la géométrie source qui est sous forme d'un fichier .mesh fichier compatible avec Ogre.
2. Spécification des paramètres de construction du maillage de navigation.
3. Construction du maillage.

Paramètres de construction

1. **Les paramètres de voxélisation :**

Les paramètres de cette section sont généralement basés sur la structure de la source et à quel point le maillage de navigation doit correspondre à la géométrie.

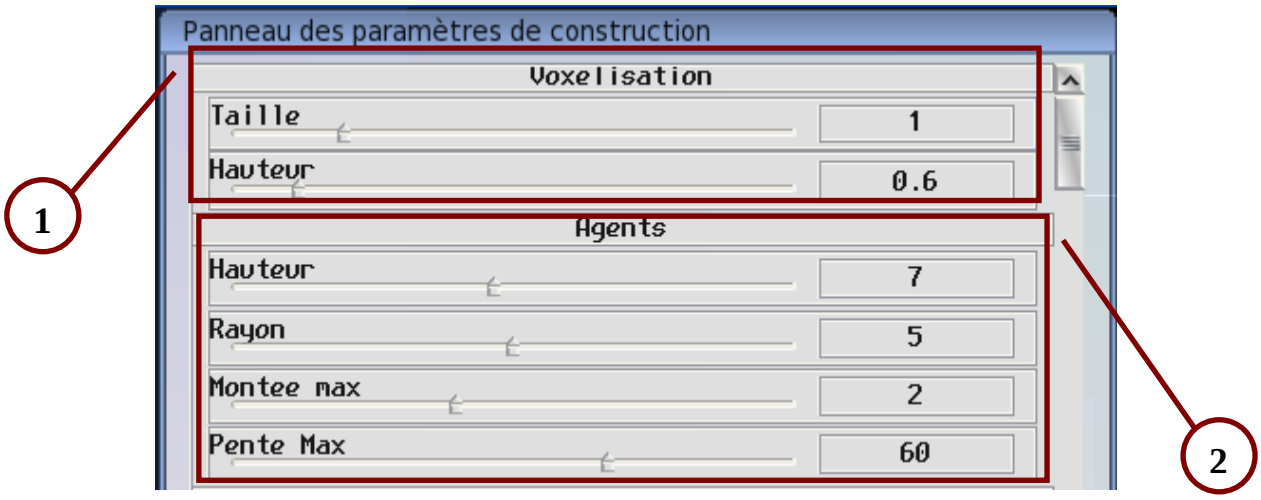


Figure 5-8 – Les paramètres de construction du maillage.

- ***la taille :***

La taille du voxel sur le plan xz à utiliser lors de l'échantillonnage de la géométrie source. Cette valeur a un effet sur la précision de la conformité du maillage de navigation final à la géométrie source sur le plan xz.

- ***la hauteur :***

La taille du voxel selon l'axe des y à utiliser lors de l'échantillonnage de la géométrie source. Cette valeur a un effet sur la précision de la conformité du maillage de navigation final à la hauteur du contour de la géométrie source. Elle permet également la détection des rebords.

Les deux principaux facteurs régissant le choix de la taille et la hauteur de cellules sont la mémoire et la performance. Plus la valeur est élevée le mieux sera le résultat. Mais c'est rare que la mémoire et la vitesse de génération du maillage soutiennent le meilleur cas.

2. Les paramètres de l'agent :

Les paramètres de cette section sont généralement basés sur le client du maillage de navigation plutôt que de la géométrie source. Le client peut être un agent de type humain, un robot, etc. Les parties de la géométrie source qui sont considéré navigables sont dictées par le type de client qui va utiliser le maillage de navigation.

- ***Hauteur :***

La hauteur minimale du « plancher » au « plafond », qui va permettre à la surface du plancher d'être considérée comme navigable. Par exemple, considérons le sol en dessous d'une table. Ce paramètre permet de détecter que la surface du sol en dessous de la table ne peut pas être traversé.

- ***Rayon :***

Représente combien est proche n'importe quelle partie d'un maillage par rapport à un obstacle de la géométrie source. Ainsi, le maillage de navigation doit généralement être réduit par le rayon du client de sorte que le planificateur de chemin ne choisira pas un chemin qui s'approche trop près des obstacles, ou passe à travers des zones qui sont trop minces pour le client

- ***Montée max :***

La hauteur de rebord maximale qui est considérée comme étant toujours navigable. Empêche les écarts mineurs de hauteur d'être détectés comme des obstacles. Cela permet au maillage de s'étaler sur les bordures, les petits objets.

- ***Pente max :***

La pente maximale qui est considéré comme franchissable (En degrés). Ce réglage détermine si le maillage de navigation va s'étaler sur des rampes lisses et un terrain ondulé. Mais il n'aura pas d'effet sur des structures telles que les escaliers.

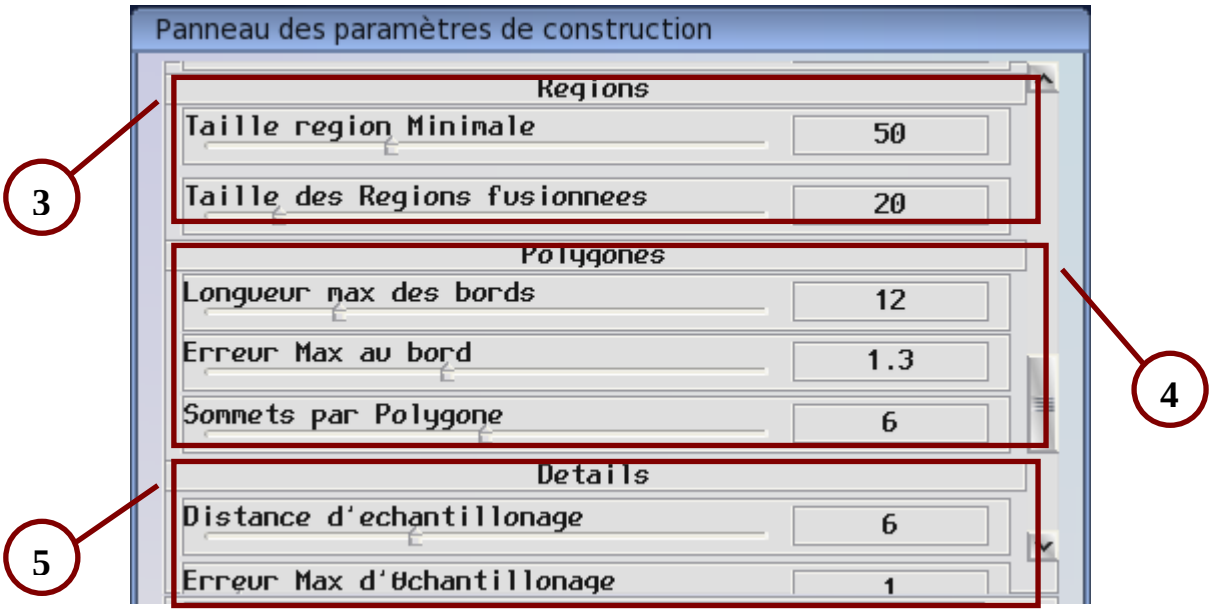


Figure 5-9 – Les paramètres de construction du maillage

3. **Les paramètres de régions :**

- ***Taille minimale des régions :***

La taille minimale pour les régions sans liens (île). Les régions qui ne sont pas connectés à n'importe quelle autre région et sont plus petits que cette taille seront abattues avant la génération du maillage. Ils ne seront plus considérés comme navigables. Avec ce paramètre correctement réglé, les surfaces des tables sont non propices à la marche

- ***Taille de régions fusionnées :***

Toutes les régions plus petites que cette taille sera, si possible, fusionnée avec des régions plus grandes.

4. **Les paramètres des polygones :**

- ***Longueur max des bords :***

Détermine la longueur maximale des bords de triangles le long de la frontière du maillage de navigation. Réduire cette valeur peut entraîner un nombre moins de triangles minces le long des bords du maillage.

Une valeur de zéro aura pour effet de désactiver cette fonctionnalité. Cette valeur n'affecte pas la précision de la détection de bord réalisé par l'écart max de bord.

- ***Ecart max au bord :***

La distance maximale entre les bords du maillage et les obstructions de la géométrie source. (S'applique uniquement sur le plan xz). La précision de ce paramètre est déterminée par la valeur de la taille des cellules xz. Une valeur plus faible résulte en des bords du maillage suivant les bords de la géométrie selon le plan xz avec plus de précision au détriment d'un nombre de triangles augmenté. Une valeur de zéro peut entraîner une forte augmentation du nombre de triangles dans le maillage final.

- ***Sommets par polygone :***

Le nombre maximal de sommets par polygone pour les polygones générés pendant le processus de conversion des contours vers des polygones.

5. **Les paramètres de détails :**

- **Distance d'échantillonnage :**

Définit la distance d'échantillonnage à utiliser pour la correspondance du maillage détaillé à la surface de la géométrie originale. Elle détermine à quel point le maillage détaillé est conforme au contour de la surface de la géométrie originale. Des valeurs élevées implique un maillage détaillé qui est plus conforme à la surface de la géométrie d'origine au prix d'un nombre de triangles final plus élevé et un coût de traitement plus élevée.

- **L'écart max d'échantillonnage :**

La distance maximale que la surface du maillage détaillé peut s'écarter de la surface de la géométrie originale. Définition de la valeur à zéro peut entraîner une forte augmentation du nombre de triangles dans le maillage détaillé à un coût de traitement élevée.

Une fois que les configurations nécessaires sont effectuées, le maillage peut être généré et enregistré pour une utilisation ultérieure.

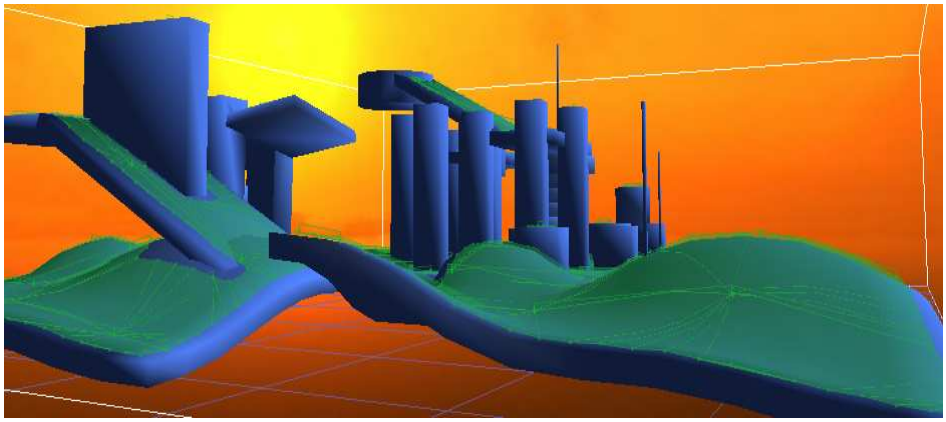


Figure 5-10 – Maillage de navigation généré.

5.4.2 Un environnement informé

L'approche proposée dans ce document consiste à créer un environnement informé composé de zones synoptiques, informant les agents du type de ces zones. Dans le chapitre précédent nous avons précisé que les zones interactives sont de deux types : zones de circulation et zones d'interaction. Cette section présentera la manière de créer de telles zones.

Créer des zones synoptiques

Les zones d'interaction sont simples à créer. Vu qu'elles n'appartiennent pas à la géométrie de l'objet. Ainsi les zones d'interaction ne sont rien d'autre qu'un volume, pouvant être paramétré à volonté par le concepteur. La forme du volume dépend de l'objet et elle est donc asservie à sa position dans l'environnement. La figure 5.11 montre la création d'une zone d'interaction.

La création des zones de circulation se fait de la même manière que les zones d'interaction (Figure 5.11). Le concepteur insère le volume représentant la zone à l'emplacement adéquat. A chaque zone (zone de circulation, zone d'interaction), est associé un attribut *type de la zone* (rue, passage piéton, gazon, porte….etc).

Intégration des zones au maillage de navigation

Au moment de la génération du maillage de navigation les volumes représentant les zones synoptiques vont être remplacé par le maillage de polygones correspondants. Les polygones représentant cette zone sont intégrés aux reste des polygones constituants le maillage complet de l'environnement ainsi cette zone est ajoutée à la base d'informations des zones constituants l'environnement.

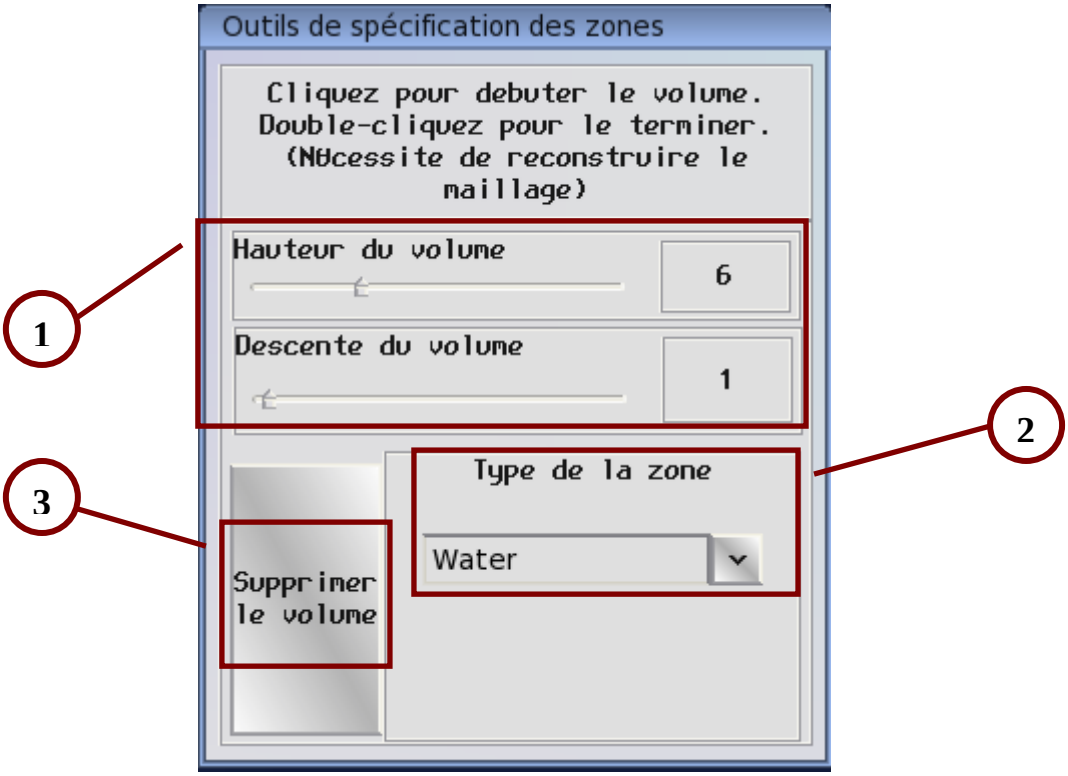


Figure 5-11 – Création des zones interactives avec TopoSyn.

5.5 Modélisation du comportement de l'humanoïde

Maintenant que notre humain virtuel est capable d'appréhender son environnement, nous allons voir son mécanisme décisionnel visant à planifier un chemin. Cette tâche peut être considérée comme réactive dans notre classification, car suffisamment habituelle pour un être humain pour ne pas nécessiter une attention trop soutenue.

Pour la tâche de planification de chemin, elle est exprimée sous la forme d'automate (Figure 5.12). Comme nous l'avons dit précédemment, le recours à des automates permet une grande souplesse dans l'expression des comportements. Cette approche est généralement utilisée pour représenter des tâches simples, de comportements réactifs.

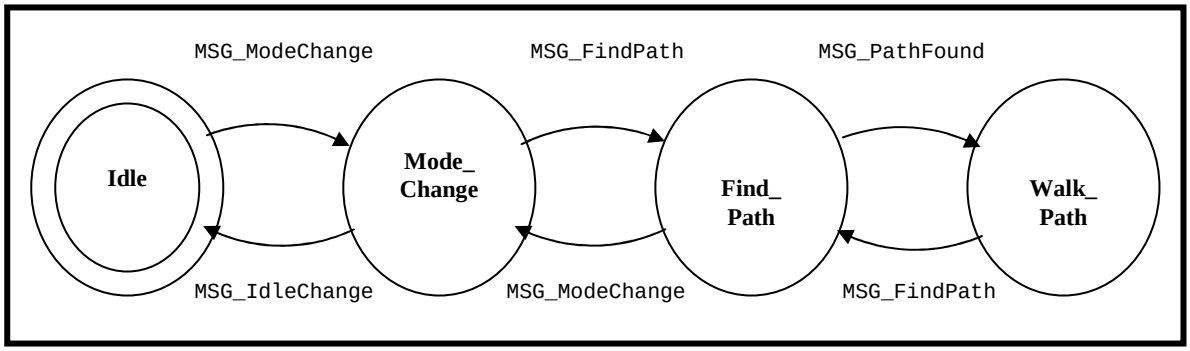


Figure 5-12 – L'automate du comportement de l'humanoïde.

5.6 Résultats

Nous présenterons dans cette section quelques résultats obtenus avec TopoSyn :

5.6.1.1 Génération du maillage de navigation

Les tests ont été menés sur un Pentium4, 3.4 GHz et 512Mo de RAM. Nous avons essayé la génération du maillage de navigation avec TopoSyn sur trois environnements différents. Les environnements choisis sont d'une taille et complexité croissantes. Le type de l'environnement est aussi différent ; les deux premiers environnement sont de type intérieur par contre le troisième est de type extérieur. Les résultats obtenus sont décrits dans les tableaux et figures qui suivent :

Environnement N°1	
Géométrie source	**5101 sommets, 10133 triangles**
Nombre de voxels	**248 x 330**
Maillage de navigation	**171 verts, 165 tris**
Espace mémoire occupé par le maillage	**3.6 kB**
Construction du maillage	**164.6 ms**

Tableau 5.1 Description du maillage de navigation de l'environnement N°1.

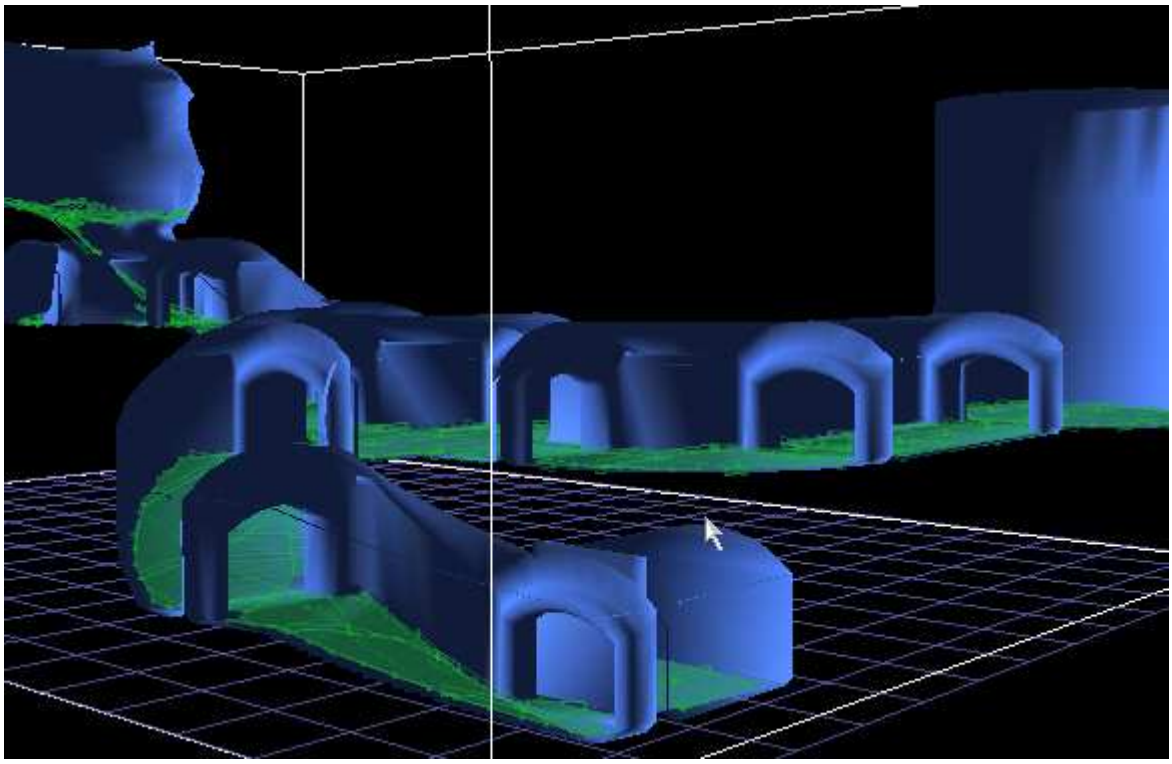


Figure 5-13 Le maillage de navigation de l'environnement N°1.

Environnement N°2	
Géométrie source	**34719 sommets, 66737 triangles**
Nombre de voxels	**1287 x 1533**
Maillage de navigation	**5193 verts, 5142 tris**
Espace mémoire occupé par le maillage	**13.8 kB**
Construction du maillage	**194.5 ms**

Tableau 5.2 Description du maillage de navigation de l'environnement N°2.

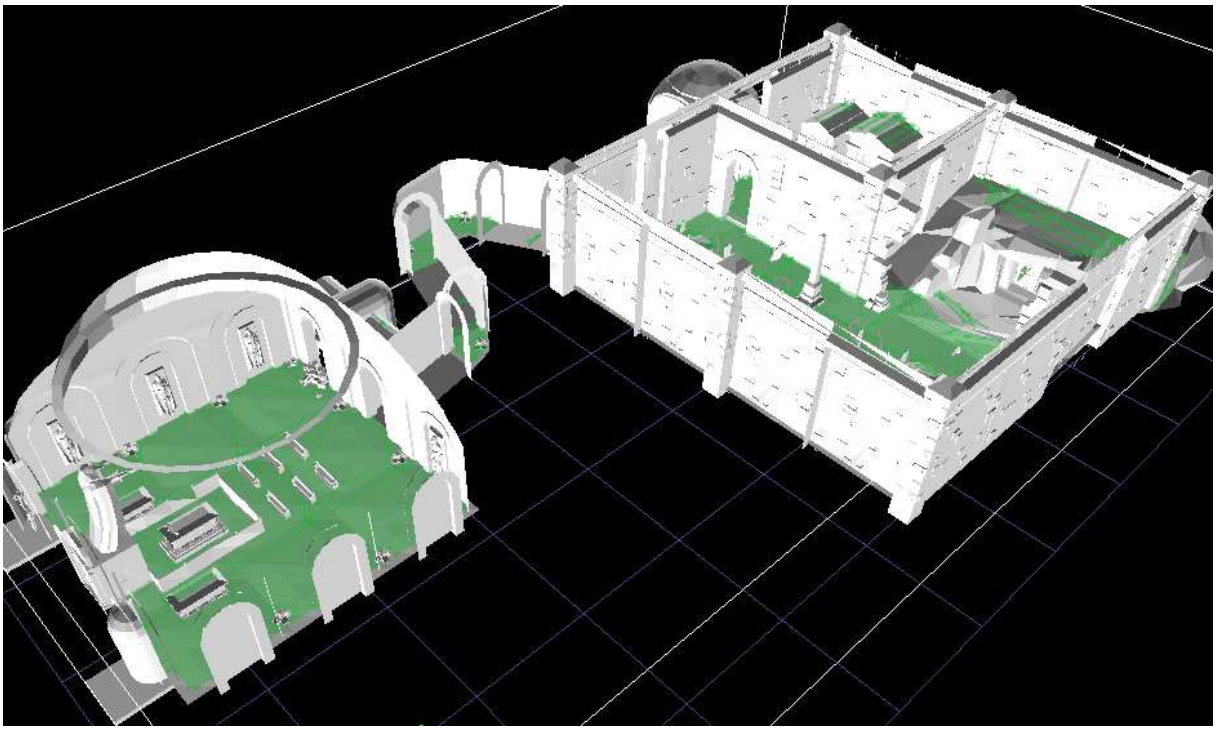


Figure 5-14 Le maillage de navigation de l'environnement N°2.

Environnement N°3	
Géométrie source	**185886 sommets, 340249 triangles**
Nombre de voxels	**1287 x 1533**
Maillage de navigation	**5193 verts, 5142 tris**
Espace mémoire occupé par le maillage	**110.8 kB**
Construction du maillage	**9414.4 ms**

Tableau 5.3 Description du maillage de navigation de l'environnement N°2.

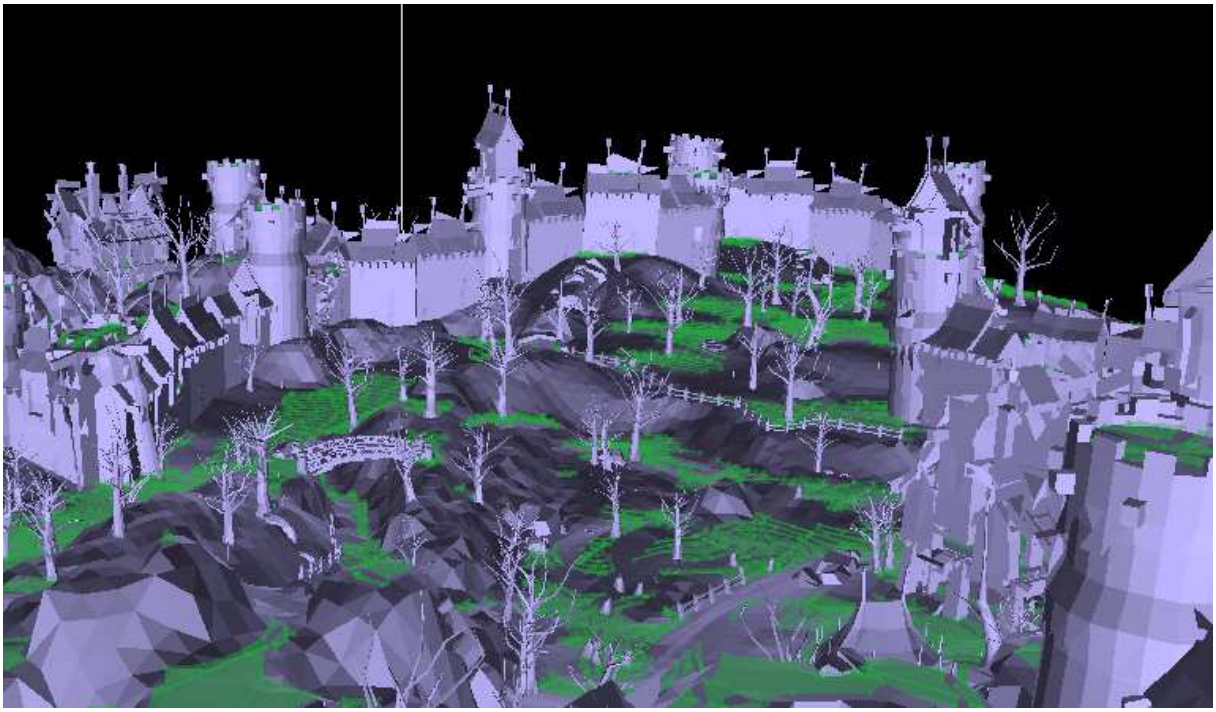


Figure 5-15 Le maillage de navigation de l'environnement N°3.

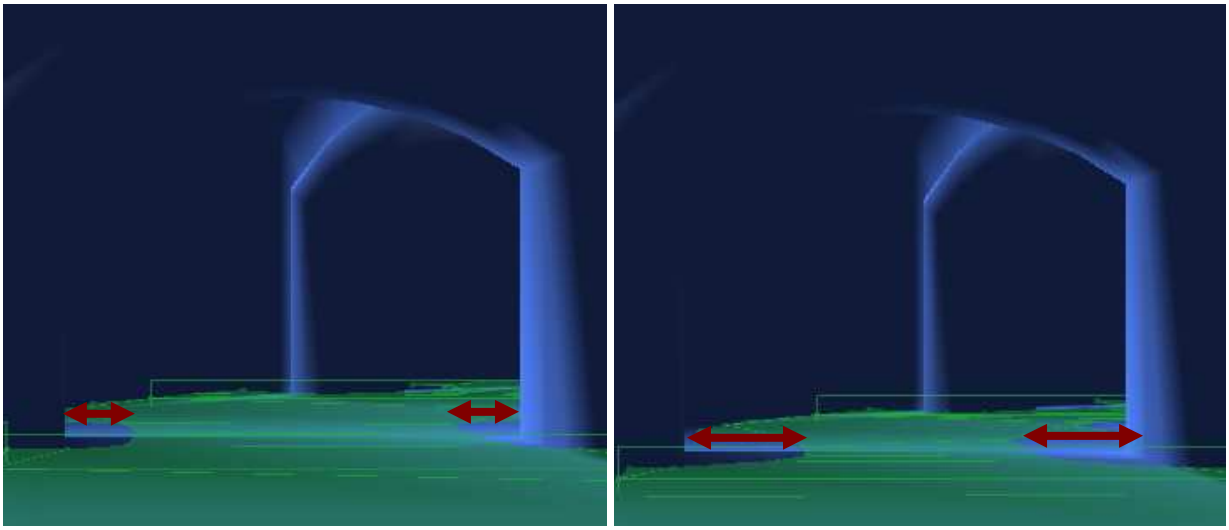
5.6.1.2 Fonctionnalités de TopoSyn

Nous allons maintenant décrire plusieurs exemples qui démontrent les fonctionnalités spécifiques de TopoSyn.

Paramétrage du maillage de navigation

La notion d'espace navigable est reliée à la morphologie de l'humanoïde : sa taille, sa largeur. La détection d'obstacles est facilitée par l'utilisation d'un volume englobant de l'humanoïde (le plus souvent un cylindre [Li04]) correspondant à l'espace nécessaire pour tenir debout. La planification a pour objectif de déplacer le volume englobant dans l'espace libre.

Ce concept est réalisable grâce au maillage de navigation et la technique adoptée pour sa génération. La figure 5.16 montre deux maillages du même environnement mais destiné à être utiliser par deux humanoïdes de taille différente. Le maillage de la figure 5.16 (a) est généré avec un rayon égal à 2 unités. Le maillage de la deuxième figure 5.16 (b) est généré avec un rayon égal à 7 unités.



(a) **(b)**

Figure 5-16 – Création du maillage avec : a) un rayon=2 b) rayon=7

La notion d'espace navigable est reliée aux capacités de l'humanoïde : sa capacité à marcher sur une colline, monter les escaliers par exemple. Le maillage de navigation peut être généré selon la pente maximale des surfaces pour qu'un humanoïde soit capable de circuler dessus. Au-delà de cette pente, le maillage ne peut être créé.

La figure 5.17 montre deux maillages du même environnement, mais destiné à être utiliser par deux humanoïdes ayant des capacités différentes. Le maillage de la figure 5.17 (a) est généré avec un pente max de 20 degrés. Le maillage de la deuxième figure 5.17 (b) est généré avec une pente de 60 degrés.

La figure 5.18 montre deux maillage du même environnement, mais destiné à être utiliser par deux humanoïdes ayant des capacités différentes (monter des escaliers). Le maillage de la figure 5.18 (a) est généré avec une monté max de 2 unités. Le maillage de la deuxième figure 5.18 (b) est généré avec une montée max de 0.5 unité.

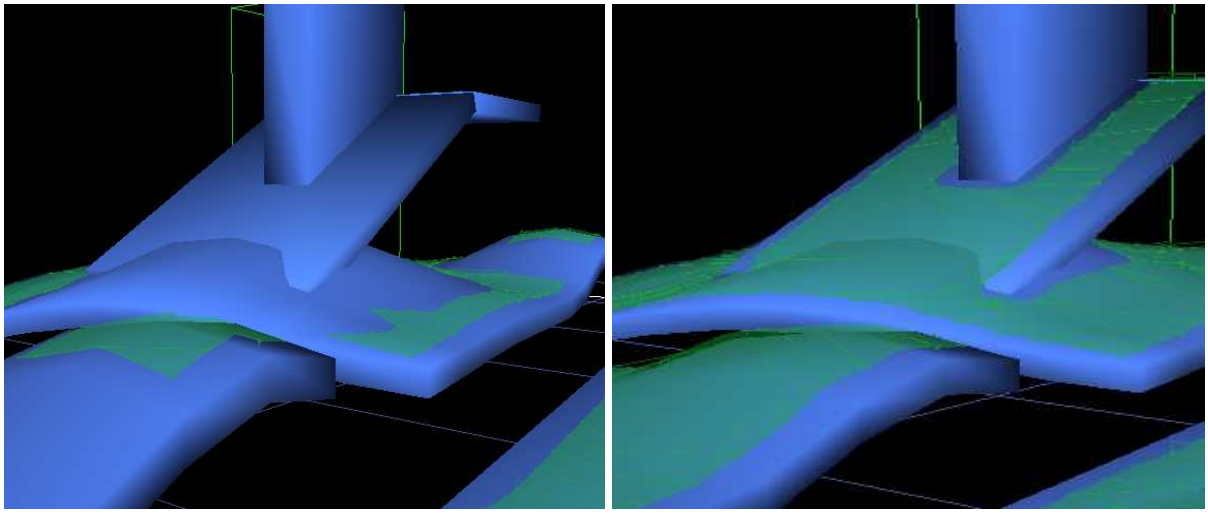


Figure 5-17 Création du maillage avec : a) pente max=20 b) pente max =60.

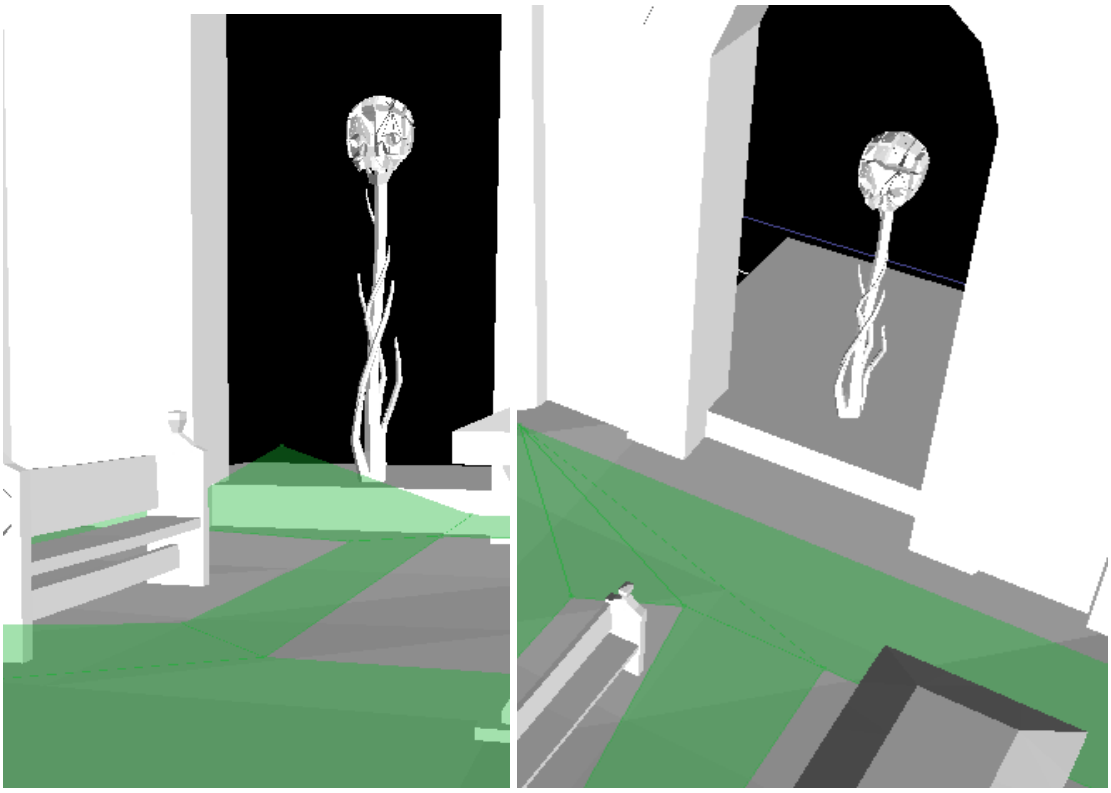


Figure 5-18 Création du maillage avec : a) montée max=2 b) montée max =0.5.

Exploitation de la base d'informations

Maintenant que notre humanoïde est capable de percevoir son environnement grâce à la représentation qu'on lui a fournie, nous allons voir son premier mécanisme de planification de chemin.

Le chemin calculé par le planificateur doit dépendre de l'environnement et des capacités de l'humanoïde. Ainsi, il est nécessaire de l'adapter aux éléments de l'environnement, voir

figure 5.19 et figure 5.20, et aux contraintes imposées par l'humanoïde, voir figure 5.21. Cette figure montre un chemin calculé selon que l'humanoïde doit éviter la zone d'eau puisqu'il n'est pas *capable* de marcher sur l'eau.

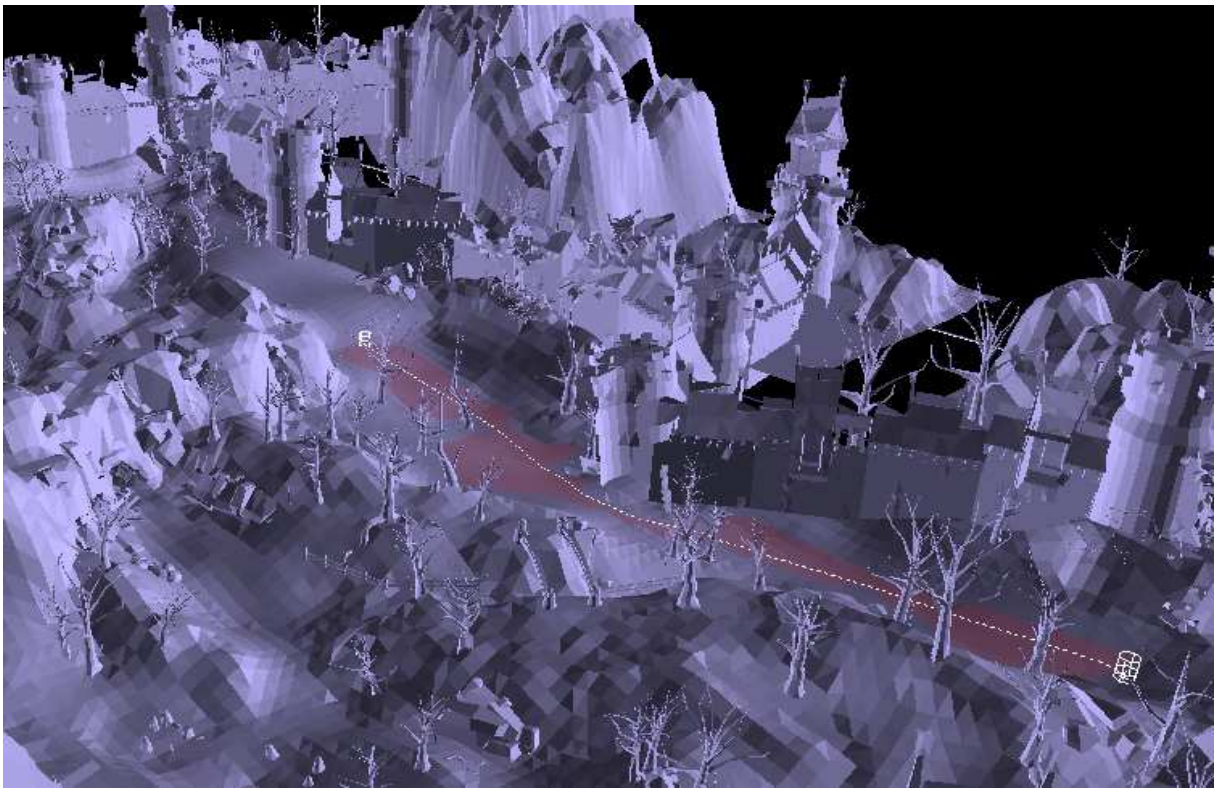


Figure 5-19 – Recherche de chemin.

Figure 5-20 – Evitement d'obstacles.

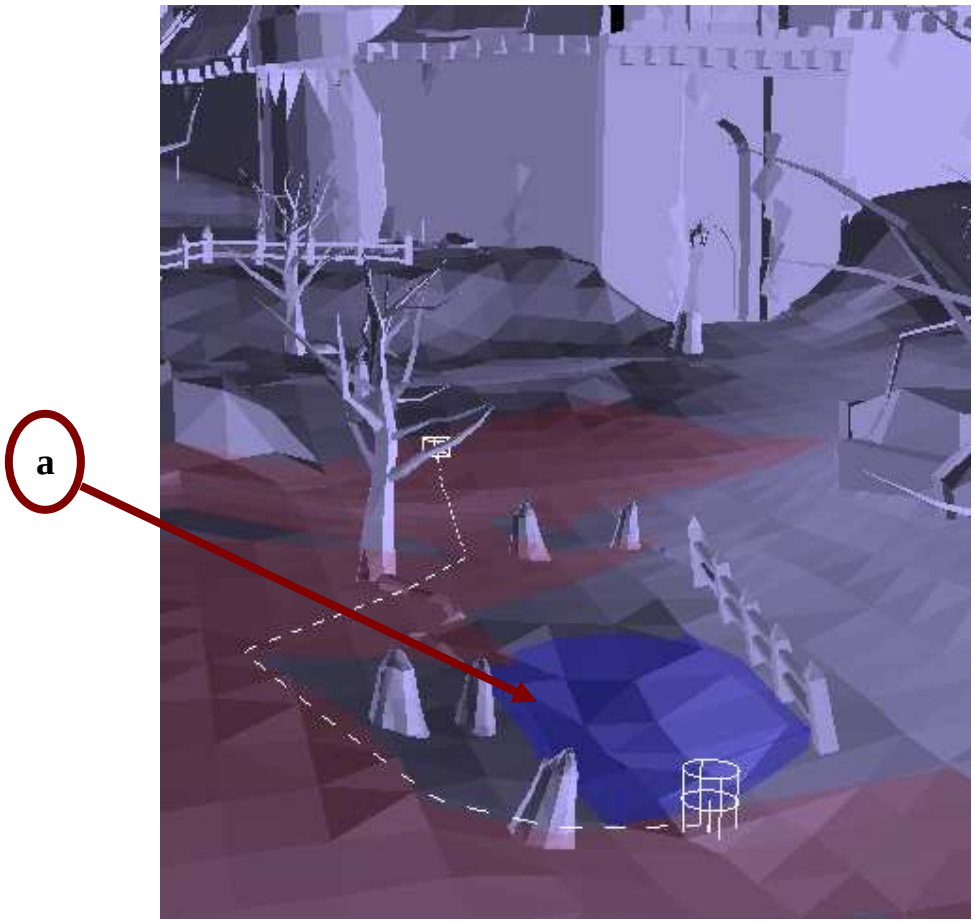


Figure 5-21 – Recherche de chemin avec contrainte (éviter la zone d'eau (a)).

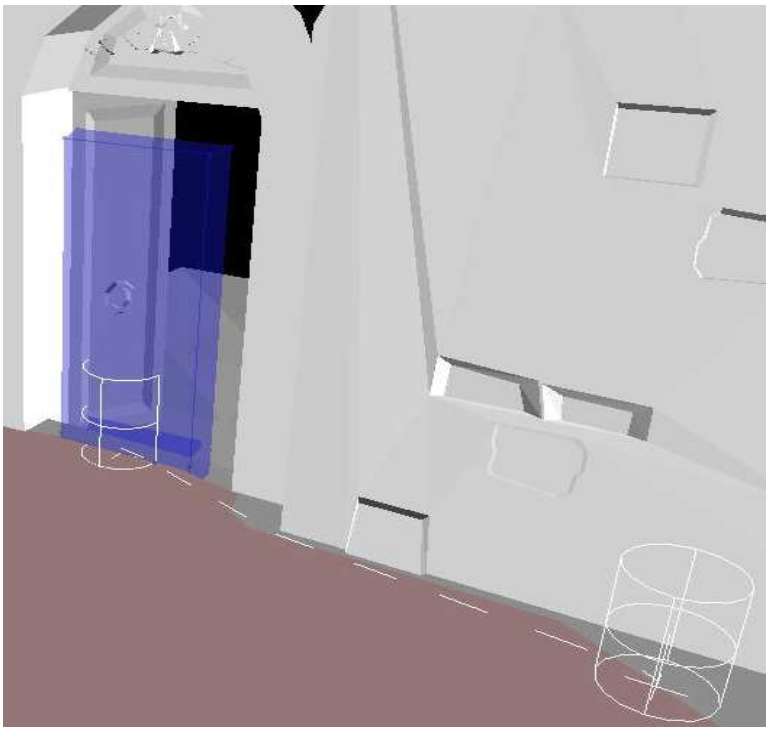


Figure 5-22 – Recherche d'une zone d'interaction de type « porte ».

La figure 5.22 nous montre un exemple sur l'intégration des objets interactifs directement dans la procédure de planification de chemin de l'humanoïde.

5.6.1.3 Démonstration

a. Configuration de la démonstration

La configuration de la démonstration se découpe en quatre étapes successives :

1. Import de l'environnement de simulation.
2. Configuration des paramètres du maillage de navigation.
3. Définition des zones synoptiques.
4. Définition des capacités de l'humanoïde.

Import de l'environnement de simulation

L'environnement de simulation définit aussi bien la topographie des lieux que le positionnement des différents éléments. Celui-ci est tout d'abord spécifié par les concepteurs via un modeleur 3D par exemple le logiciel 3DS MAX.

Ce dernier dispose d'un plugin permettant d'exporter la scène 3DS MAX vers une scène Ogre3D. Les données géométriques représentant l'environnement peuvent alors être sauvegardées au format .mesh, afin d'en permettre l'utilisation avec Ogre. Depuis ce fichier .mesh, l'outil TopoSyn peut ensuite créer le maillage représentant la topologie des lieux, et générer les zones synoptiques qui ont été spécifiées.

Par exemple, on peut voir sur la figure 5.23 (a) le fichier d'entrée .3ds de notre environnement de test (le format des fichiers 3DS max) et sur la figure 5.23 (b) le fichier d'entrée *.mesh* le format des fichiers compatible avec Ogre3D.

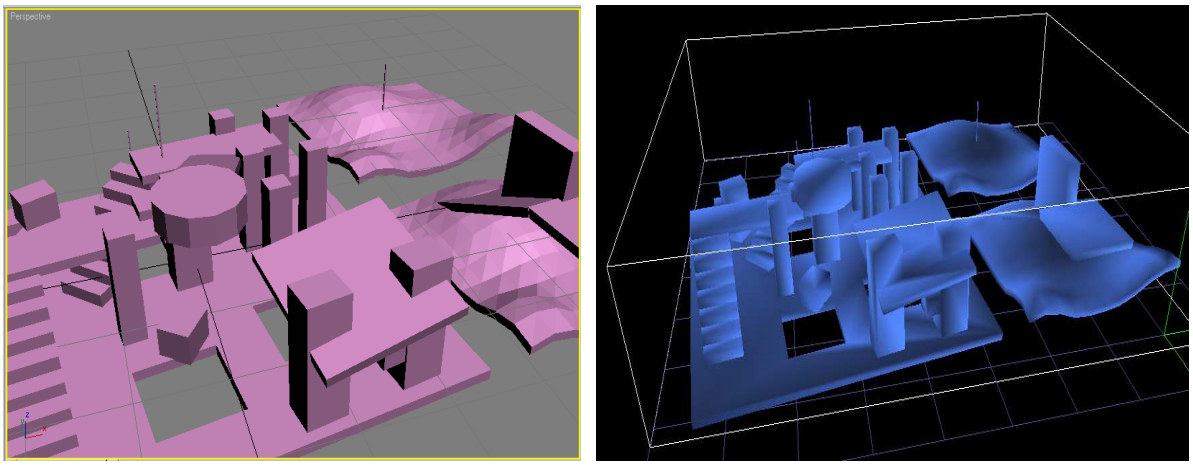


Figure 5-23 – La géométrie source (a)Le fichier d'entrée au format .3ds (b) le fichier d'entrée au format .mesh.

Configuration des paramètres du maillage de navigation

Les paramètres de configuration du maillage représentent l'ensemble des informations nécessaires à la génération du maillage de navigation, section 5.4.1. Afin de répondre aux caractéristiques de l'environnement et de l'humanoïde choisis les paramètres suivants ont été saisis :

Définition des zones synoptiques

La deuxième étape de la démonstration concerne la définition des zones synoptiques spécifiques à l'environnement. Afin de s'intégrer au modèle topologique, ces zones sont dessinées à leur emplacement adéquat.

Définition des capacités de l'humanoïde

La définition des capacités de l'humanoïde permet d'affecter des capacités aux individus créés lors de la démonstration. Les capacités d'un humanoïde sont un facteur influençant la navigation.

b. Déroulement d'une simulation

Ce module permet deux modes de démonstration. Premièrement, une démonstration avec plusieurs humanoïdes (max=15) où l'utilisateur va laisser le système générer les positions à atteindre pour les humanoïdes de façons aléatoire. Deuxièmement, une démonstration avec un seul humanoïde où l'utilisateur pourra voir l'utilisation du modèle d'environnement par un humanoïde on lui spécifiant sa destination.

Un exemple du déroulement d'une démonstration est fourni aux figures 5.24 et 5.25, illustrant le processus de recherche de chemin.

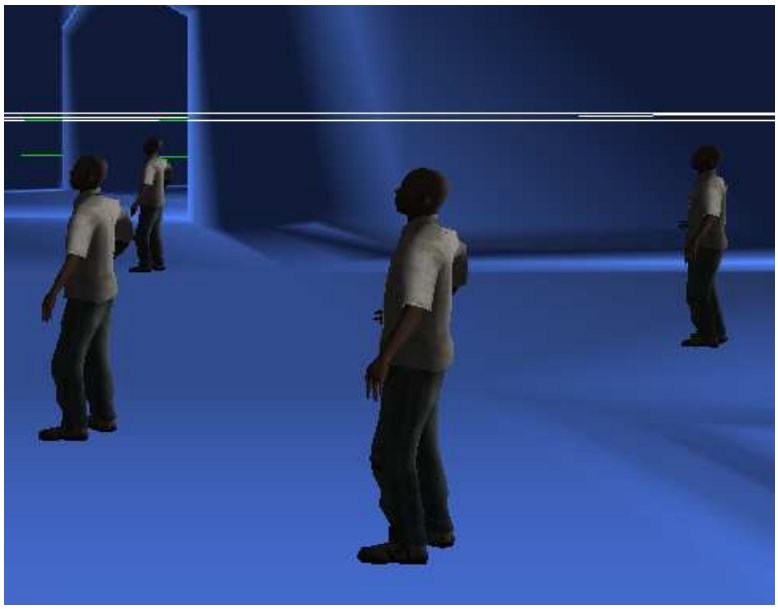


Figure 5-24 – Démonstration avec plusieurs humanoïdes.

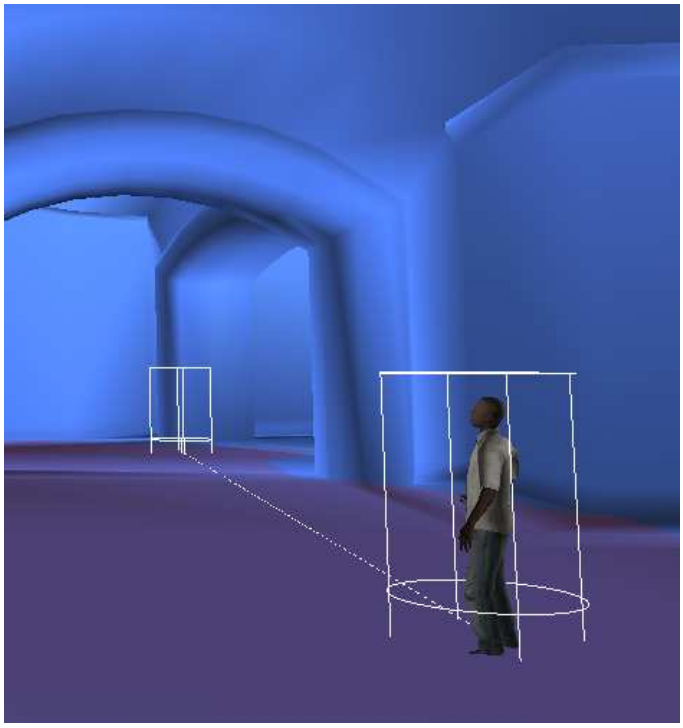


Figure 5-25 – Démonstration avec un seul humanoïde.

5.7 Evaluation du modèle

Nous avons proposé un modèle de représentation de l'environnement virtuel. Celui-ci se base sur une représentation exacte de la géométrie des lieux. Une typologie est associée à certaines zones du modèle topologique, permettant de caractériser globalement les zones de circulation et d'interaction tout en gardant un lien avec leur définition géométrique et topologique fine. De plus, les objets interactifs sont associés à la topologie, permettant de situer l'interaction dans l'espace. Pour finir, cette représentation peut être étendue par son interconnexion avec d'autres environnements utilisant la même représentation.

Comme nous l'avons vu au travers de plusieurs exemples, cette approche permet de représenter des environnements de taille diverse, allant de la ville à la maison. Une perspective d'utilisation attrayante serait de représenter l'intégralité d'une ville grâce à ce modèle, en interconnectant plusieurs environnements de catégorie sémantique différente. Il serait ainsi possible de représenter les rues de la ville par un graphe, puis chaque bâtiment par un autre graphe qui sera connecté à celui de la ville.

Nous pouvons assurer qu'une telle construction est possible, tout en envisageant que son exploitation se ferait avec une complexité de calcul raisonnable grâce à l'aspect hiérarchique de la représentation.

Rappelons que ce modèle ne s'intéresse pas à l'interaction physique entre un humanoïde et un objet, tout au plus en spécifiant des zones de l'environnement à atteindre. Par conséquence, il se concentre plus sur la localisation des lieux d'interaction que sur son réalisme visuel, comme pourrait le faire un modèle de type *Starfish* ou *Smart Object*. Une évolution souhaitable de ce modèle serait donc vers la gestion fine de l'interaction physique.

Un autre aspect non abordé par se modèle concerne les concepts nécessaires à la description et à la gestion de l'interaction d'un point de vue comportemental, comme proposé par Paris avec *BIIO*. Ainsi, une autre évolution avantageuse pour ce modèle serait l'introduction d'une architecture d'objets et de lieux interactifs. Elle définit ainsi un certain nombre d'objets et de lieux interactifs plus ou moins abstraits, dotés de fonctionnalités génériques pouvant s'adapter à des spécialisations ultérieures.

Pour conclure, nous avons proposé un outil de modélisation d'environnement virtuel TopoSyn. Celui-ci gère la chaîne de modélisation complète, depuis la modélisation topologique, avec l'ajout de zones synoptiques, jusqu'à l'exploitation finale du modèle crée. Son aspect modulaire permet une évolution progressive de cet outil, et autorise encore l'enrichissement de ses constituants.

L'application proposée se veut aussi générique que possible, afin de pouvoir être utilisée dans un large éventail d'environnements, tels que des environnements intérieurs et extérieurs. Elle permet ainsi d'exploiter le fort potentiel d'évolution offert par les zones synoptiques, pour ajouter par la suite de nouvelles informations à ces zones, et donc de nouvelles procédures comportementales. Nous proposons néanmoins une évolution de l'application concernent essentiellement la modélisation dynamique de l'environnement.

5.8 Conclusion

Nous avons abordé tout au long de ce chapitre les résultats obtenus lors de ce mémoire. Ces résultats prennent tout d'abord la forme d'un outil dédié à la modélisation d'environnements virtuels du nom de TopoSyn. Ce logiciel concrétise le modèle présenté comme contribution, tout en apportant un certain nombre de fonctionnalités permettant de les exploiter. Il permet ainsi de gérer la chaîne de modélisation complète, depuis la création du maillage de navigation, jusqu'à l'ajout de nouvelles zones interactives et l'exploitation finale des données produites. Cet outil est fortement basé sur la création de maillage de navigation selon la méthode de voxelisation et des basins versants, qui est exploité pour fournir une subdivision exacte de l'espace navigable. Un module d'ajout de zones spécifiques est

néanmoins fourni, afin de répondre au mieux aux spécificités de l'autonomie des agents virtuels. Cet outil reste ouvert quant à ses possibilités d'évolutions.

Conclusion générale

Conclusion

Nous avons traité au sein de ce document la problématique de modélisation d'environnements virtuels. Cette problématique est renforcée dans notre cadre d'application par la nécessité de tenir compte des besoins d'interaction des humains avec l'environnement ou ils évoluent.

Nous avons commencé par analyser l'état de l'existant concernant la modélisation des environnements virtuels, et l'animation comportementale. Cela nous a permis de nous familiariser avec certaines des caractéristiques du comportement humain, comme la prise de décision. Nous avons fait de ces caractéristiques les contraintes de notre approche, en observant qu'elles avaient un impact fort sur les modèles d'environnements. Nous avons aussi couvert les différentes méthodes permettant de représenter les environnements. Ces approches ont permis de définir le cadre du modèle vers lequel nous voulions nous diriger.

Dans un deuxième temps, nous avons proposé un modèle d'environnement permettent une description informée de l'environnement. Ce modèle s'articule autour de deux problématiques : représenter la topologie d'un environnement virtuel, et associer à cette représentation topologique des informations sémantiques.

Notre modèle d'environnement est fondé sur une représentation topologique de type exacte, garantissant la conservation de l'information géométrique. Cette représentation est générée en utilisant la méthode de voxélisation et des bassins versants, afin de représenter les surfaces navigables sous forme d'un maillage de polygones tout en conservant l'information de hauteur. Elle permet ainsi de générer un modèle en 2D1/2.

Cette représentation est ensuite informée grâce à la caractérisation de certaines zones de l'environnement : les zones interactives. Ces zones peuvent être soit des zones de circulation soit des zones d'interaction avec les objets. L'humanoïde en connaissant le type de la zone où il se trouve il peut changer de comportement.

Nous pouvons conclure de façon générale que la contribution présentée dans ce document répond à la problématique posée. De plus, la mise en oeuvre du modèle a été réalisée, prouvant sa faisabilité. Enfin, le travail proposé est tourné vers l'avenir, en autorisant de nombreuse évolution.

Limitations

La principale limitation de notre approche est la définition des zones interactives. La sélection d'une surface sur une partie de l'environnement très complexe peut être très fastidieuse si l'utilisateur veut être sûr qu'aucune des surfaces sélectionnées ne mèneront à un mauvais placement de l'agent. La qualité de l'animation résultante dépend directement du choix de ces zones.

Perspectives

Afin d'accroître les performances de ce modèle, il peut être important de se pencher sur différents autres points. Nous en avons relevés trois permettant d'obtenir un modèle encore plus réaliste :

Le premier point concerne les zones synoptiques d'interaction avec les objets. Les perspectives d'évolutions de ce modèle sont nombreuses, sa conception même étant orientée vers cet objectif. Nous avons tout de même identifié deux points qui apporteraient un bénéfice indéniable à cette approche. Premièrement, notre modèle ne s'intéresse pour le moment qu'à situer ces zones dans l'environnement, et laisse donc de côté la description des interactions. Son couplage à un modèle spécialisé dans cet objectif permettrait donc d'étendre son champ d'application. Nous avons identifié un modèle pouvant répondre à cette perspective, STARFISH qui utilise des éléments de description de l'interaction physique. Deuxièmement, ces zones pourraient être branchés sur un moteur de raisonnement permettant à un agent de raisonner sur les actions fournies par les zones.

Le deuxième point concerne les zones de circulation. Puisque, un humain lorsqu'il pense à une région, il pense d'abord à la région entière, puis s'il cherche un endroit particulier, il va affiner progressivement la recherche par composant de la région. Il est donc envisageable d'exprimer un environnements sous forme d'une structure hiérarchique, chaque nœud de l'hiérarchie représentant une typologie de zone de circulation différente : par exemple la ville dans son ensemble (ses rues), les habitations, et des bâtiment spécifiques (gare, stade, métro,

etc.). Il serait alors possible de simuler différentes échelles environnementales en se fondant sur une unique représentation.

Le troisième point concerne l'aspect du comportement. Afin d'obtenir des comportements plus riches, il est possible d'associer de nouveaux attributs symboliques aux zones. Par exemple, en plus de nommer une rue, il serait intéressant de la caractériser à un niveau de détail inférieur ; en associant par exemple une liste de vitrines à un coté. Le piéton pourrait ainsi flâner en regardant les vitrines. L'ajout de tels attributs ne pose pas de problèmes techniques dans l'état actuel de l'outil TopoSyn ; les briques de construction de l'environnement existant déjà, il suffit de les étendre en rajoutant de nouveaux attributs.

Références bibliographiques

[Aba06] T. Abaci. *Object manipulation and grasping for virtual humans.* Thèse de Doctorat, Ecole Polytechnique Fédérale de Lausanne, mars 2006.

[Abd94] H. Abdi. *A neural network primer*. Journal of Biological Systems, 2(3):247– 283.

[ABG05] R. Andersen, J.L. Berrou, et Alex. Gerodimos. *On some limitations of grid-based (ca) edestrian simulation models*. Dans First International Workshop on Crowd Simulation (V-Crowds'05). VRlab, EPFL, 2005.

[AK89] J. Arvo, et D. Kirk. *A survey of ray tracing acceleration techniques.* Dans An introduction to ray tracing, pages 201–262. Academic Press Ltd., London, UK.

[Ark98] R.C. Arkin. *Behavior-based robotics*. MIT Press, 1998.

[Aub07] S. Aubry. *Annotations et gestion des connaissances en environnement virtuel collaboratif*. Thèse de Doctorat, Université de Technologie de Compiègne, 2007.

[Bad06] M. Badawi. *Synoptic objects describing generic interaction processes to autonomous agents in an informed virtual environment*. Thèse de Doctorat, INSA de Rennes, décembre 2006.

[BF97] A. L. Blum et M. L. Furst. *Fast planning through planning graph analysis*. Artificial Intelligence, pages 281–300, 1997.

[BG01] B. Bonet et H. Geffner. *Planning as heuristic search*. Artificial Intelligence, pages 5–33, 2001.

[BID+01] R. Burke, D. Isla, M. Downie, Y. Ivanov, et B. Blumberg. Creature smarts: The art and architecture of a virtual brain. Dans *The Game Developers Conference*, pages 147–166, 2001.

[BL91] J. Barraquand et J.-C. Latombe. *Robot motion planning : a distributed representation approach*. Dans Internation Journal of Robotics Research, pages 628–649, 1991.

[Blu97] B. M. Blumberg. *Olds Tricks, New Dogs : Ethology and Interactive Creatures*. Thèse de Doctorat, Massachussetts Institute of Technology.

[BMS04] A. Botea, M. Müller, et J. Schaeffer. *Near optimal hierarchical pathfinding*. Journal of Game Development, pages 7–28, 2004.

[BNC+03] D. A. Bowman, C. North, J. Chen, N. F. Polys, P. S. Pyla, et U. Yilmaz. *Information-rich virtual environments: theory, tools, and research agenda*. Dans *VRST*, pages 81–90, 2003.

[BRW97] N. I. Badler, B. D. Reich, et B. L. Webber. Towards personalities for animated agents with reactive and planning behaviors. Dans Creating Personalities for Synthetic Actors, Towards Autonomous Personality Agents, pages 43–57, London, UK. Springer-Verlag, 1997.

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

[Ceg] Bibliothèque de gestion des interfaces graphiques. www.cegui.org.uk.

[Cel] Editeur de fenêtre pour Cegui. www.cegui.org.uk.

[Cha82] B. Chazelle. A theorem on polygon cutting with applications. Dans *Proceedings of the 23rd IEEE Symposium on Foundations of Computer Science*, pages 339–349, 1982.

[Che87] L. P. Chew. Constrained delaunay triangulations. Dans *Proceedings of the third annual symposium on Computational geometry*, pages 215–222. ACM Press, 1987.

[CRR01] S. Caselli, M. Reggiani, et R. Rocchi. Heuristic methods for randomized path planning in potential fields, 2001.

[DF98] V. Decugis et J. Ferber. Action selection in an autonomous agent with a hierarchical distributed reactive planning architecture. Dans *Proceedings of the 2nd International Conference on Autonomous Agents (Agents'98)*, édité par Katia P. Sycara et Michael Wooldridge, pages 354–361, New York. ACM Press, 1998.

[Don04] S. Donikian. *Modélisation, contrôle et animation d'agents virtuels autonomes évoluant dans des environnements informés et structurés*. Habilitation à Diriger des Recherches, Université de Rennes, Institut de Formation Supérieure en Informatique et en Communication, 2004.

[Doy02] P. Doyle. Believability through context using "knowledge in the world" to create intelligent characters. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 342–349. ACM Press, 2002.

[DR95] S. Donikian et E. Rutten. Reactivity, concurrency, data-flow and hierarchical preemption for behavioural animation. Dans *Programming Paradigms in Graphics*, édité par R.C. Veltkamp et E.H. Blake. Springer-Verlag, 1995.

[Elf87] A. Elfes. Sonar-based real-world mapping and navigation. *IEEE Journal of Robotics and Automation 3(3)* (June), pages 249–265,1987.

[Far01] N. Farenc. An informed environment for inhabited city simulation. Thèse de Doctorat, Ecole Polytechnique Fédérale de Lausanne, 2001.

[Fer95] J. Ferber. *Vers une intelligence collective*. InterEditions, 1995.

[FK03] R. Fernando, et M. J. Kilgard. *The Cg Tutorial: The Definitive Guide to Programmable Real-Time Graphics*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

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

[Fun98] J. Funge. *Making them behave: Cognitive Models for Computer Animation*. Thèse de Doctorat, Université de Toronto, 1998.

[Gib86] J.J. Gibson. *The Ecological Approach to Visual Perception*, chapter 8: The Theory of Affordances, pages 127–143. Lawrence Erlbaum Associates, 1986.

[GM03] A. Guillot, et J.-A. Meyer. La contribution de l'approche animat aux sciences cognitives. Dans *Cognito*, pages 1–26, 2003.

[GSN04] C. Gloor, P. Stucki, et K. Nagel. Hybrid techniques for pedestrian simulations. Dans *4th Swiss Transport Research Conference*, Monte Verità, Ascona, 2004.

[Gut05] M. A. Gutiérrez. *Semantic Virtual Environments*. . Thèse de Doctorat, Ecole Polytechnique Fédérale de Lausanne. 2005.

[Her06] D. Herviou. *La perception visuelle des entités autonomes en réalité virtuelle : Application à la simulation de trafic routier*. Thèse de Doctorat, Université de Bretagne Occidentale, 2006.

[HM83] S. Hertel, et K. Mehlhorn. Fast Triangulation of the Plane with Respect to Simple Polygons. Dans *International Conference on Foundations of Computation Theory*, 1983.

[HOF99] K. E. Hoff III, J. Keyser, M. Lin, D. Manocha, et T. Culver. Fast computation of generalized Voronoi diagrams using graphics hardware. Computer Graphics, pages 277–286, 1999.

[HYD09] D. H. Hale, G. M. Youngblood, et P. N. Dixit. Automatically-generated Convex Region Decomposition for Real-time Spatial Agent Navigation in Virtual Worlds. L'Université de North Carolina, Charlotte College of Computing and Informatics, Department of Computer Science, 2009.

[IB02] D. Isla, et B. Blumberg. Object persistence for synthetic creatures. Dans *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1356 – 1363, 2002.

[KAL01] M. Kallmann. *Object Interaction in Real-Time Virtual Environments*. Thèse de Doctorat, Ecole Polytechnique Fédérale de Lausanne, 2001.

[KB91] B. Kuipers, et Y. T. Byun. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Journal of Robotics and Autonomous Systems,* pages 47–63, 1991.

[KBT03] M. Kallmann, H. Bieri, et D. Thalmann. Fully dynamic constrained delaunay triangulations. Geometric Modelling for Scientific Visualization, 2003.

[KLA98] R. L. Klatzky. Allocentric and egocentric spatial representations: definitions, distinctions, and interconnections. Dans *Spatial Cognition : An Interdisciplinary Approach to*

Representing and Processing Spatial Knowledge, volume 1404/1998 de Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 1998.

[Kor85] R. E. Korpf. *Depth-first iterative-deepening : An optimal admissible tree search.*

[KT98] M. Kallmann et D. Thalmann. Modeling objects for interaction tasks. Dans *EGCAS*, 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.

[LD02] F. Lamarche et S. Donikian. Automatic orchestration of behaviours through the management of resources and priority levels. Dans *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'02)*, édité par M. Gini, T. Ishida, C. Castelfranchi, et W. L. Johnson, pages 1309–1316. ACM Press, 2002.

[LD04] F. Lamarche et S. Donikian. Crowds of virtual humans : a new approach for real time navigation in complex and structured environments. Computer Graphics Forum, Eurographics'04, 2004.

[LD91] J. Leonard, and H. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *IEEE Transaction on Robotics and Automation*, pages 376–382, 1991.

[Lep03] J. Leplat. Formalismes de modélisation pour l'analyse du travail et l'ergonomie, chapitre La modélisation en ergonomie à travers son histoire, pages 1–26. Presses Universitaires de France, 2003.

[LMM03] C. Lascos, D. Marchal, A. Meyer. Intuitive crowd behaviour in dense urban environment using local laws. Dans *Proceedings of theory and practice of computer graphics*. IEEE Computer Society, 2003.

[LRD+90] Jed Lengyel, Mark Reichert, Bruce R. Donald, et Donald P. Greenberg. Realtime robot motion planning using rasterizing computer graphics hardware. Dans SIGGRAPH '90 : Proceedings of the 17th annual conference on Computer graphics and interactive techniques, pages 327–335. ACM Press, 1990.

[Mac03] D.J.C. MacKay. Information Theory, Inference and Learning Algorithms. Cambridge University Press, 2003.

[Mae90] P. Maes. Situated agents can have goals. Dans Designing Autonomous Agents, édité par Patti Maes, pages 49–70. MIT Press, 1990.

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

[MM06] D. Miles et M. Mononen. Crowds In A Polygon Soup Next-Gen Path Planning. Dans *Game Developers Conference*, 2006.

[New90] Allen Newell. Unified theories of cognition. Harvard University Press, Cambridge, Massachusetts.

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

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

[Ogra] Ogre3D. Moteur de Rendu Graphique. www.ogre3d.org.

[Ogrb] OgreMax Scene Exporter.Plugin du modeleur 3DS Max. www.ogremax.com.

[Oha02] N. O'Hara. Hierarchical impostors for flocking algorithm in there dimentional space. Dans *Irish Workshop on Computer Graphics*, numéro 3, 2002.

[Ott05] K. A. Otto. The semantics of multi-user virtual environments. Dans *SVE*, 2005.

[Pan08] D. Panzoli. *Proposition de l'architecture « Cortexionist » pour l'intelligence comportementale de créatures artificielles*. Thèse de Doctorat, Université Toulouse III - Paul Sabatier, 2008.

[PDB05] S. Paris, S. Donikian, N. Bonvalet. Towards more Realistic and Efficient Virtual Environment Description and Usage. Dans *First International Workshop on Crowd Simulation (V-Crowds'05*). VRlab, EPFL, 2005.

[PF93] M. Van de Panne et E. Fiume. Sensor-actuator networks. Dans *Computer Graphics (SIGGRAPH '93 Proceedings)*, édité par J. T. Kajiya, volume 27, pages 335–342.

[Rey87] C. W. Reynolds. Flocks, herds, and schools : A distributed behavioral model. Computer Graphics, pages 25–34, 1987.

[RHJ+01] N. Rodriguez, O. Heguy, J.-P. Jessel, et H. Luga. Assistance coopérative pour la téléopération. Rapport technique, IRIT, 2001.

[Ric01] N. Richard. *Description de comportements d'agents autonomes évoluant dans des mondes virtuels.* Thèse de Doctorat, École Nationale Supérieure des Télécommunications, 2001.

[RG95] A. S. Rao et M. P. Georgeff. BDI-agents: from theory to practice. Dans *Proceedings of the First Intl*. Conference on Multiagent Systems, San Francisco, 1995.

[RTT90] O. Renault, N. Thalmann, et D. Thalmann. A vision-based approach to behavioral animation. Visualization and Computer Animation, pages 18–21, 1990.

[Sha06] W. Shao. *Animating Autonomous Pedestrians*, Thèse de Doctorat, Courant Institute of Mathematical Sciences, Université de New York, 2006.

[SM00] S. D. Steck et H. A. Mallot. The role of global and local landmarks in virtual environment navigation. Presence : Teleoperators and Virtual Environments, pages 69–83, 2000.

[TB96] S. Thrun et A. Bücken. Integrating grid-based and topological maps for mobile robot navigation. Dans *Proceeding of the AAAI Thirteenth National Conference on Artificial Intelligence*, pages 944–951. AAAI Press / MIT Press, 1996.

[TD03] A. Tom et M. Denis. Referring to landmark or street information in route directions: What difference does it make? Dans Spatial Information Theory, volume 2825/2003, pages 362–374. Springer Berlin / Heidelberg, 2003.

[TD06] R. Thomas et S. Donikian. A navigation architecture for agents based on a human-like memory and cognitive map model. Dans *Computer Animation and Social Agents*, édité par N. Magnenat-Thalmann, D. Pai, A. Paiva, et E. Wu, pages 83–93. Computer Graphics Society, 2006.

[Tho05] R. Thomas. *Modèle de mémoire de carte cognitive spatiales : application à la navigation du piéton en environnement urbain*. Thèse de Doctorat, Université de Rennes I, 2005.

[Tho99] G. Thomas. *Environnements virtuels urbains : modélisation des informations nécessaires à la simulation de piétons*. Thèse de Doctorat, Université de Rennes I, 1999.

[TLC02] F. Tecchia, C. Loscos, et Y. Chrysanthou. Visualizing crowds in real-time.Dans Computer Graphics Forum, pages 753–765, 2002.

[TT94] D. Terzopoulos, et X. Tu. Artificial fishes : Physics, locomotion, perception, behavior. Dans *SIGGRAPH'94 Computer Graphics*, pages 42–48. ACM, 1994.

[TU96] X. TU. *Artificial animals for computer animation : biomechanics, locomotion,*

[War95] W.H. Warren. Handbook of perception and cognition: Perception of Space and Motion, chapitre Self-Motion : Visual Perception and Visual Control, pages 263–325. Academic Press, 1995.

[WHR+03] K.F.Wender, D. Haun, B. Rasch, et M. Blümke. Context effects in memory for routes. Dans *Spatial Cognition III*, volume 2685/2003 de Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2003.

[WLG+02] D. Waller, J. M. Loomis, R. G. Golledge, et A. C. Beall. Place learning in humans: The role of distance and direction information. Spatial Cognition and Computation, pages 333–354, 2002.

[WM03] J. M. Wiener et H. A. Mallot. 'fine-to-coarse' route planning and navigation in regionalized environments. Spatial Cognition & Computation, pages 331–358, 2003.