UNIVERSITÉ DE PAU ET DES PAYS DE L'ADOUR

ÉCOLE DOCTORALE

DES SCIENCES EXACTES ET LEURS APPLICATIONS - ED 211

T H È S E

pour obtenir le titre de

Docteur en Sciences

de l'Université de Pau et des Pays de l'Adour

Mention : Informatique

présentée par

Julien VALENTIN

Simulation du comportement humain en situation d'évacuation de bâtiment en feu

soutenue le 3 avril 2013

Jury :

Rapporteurs :	Emmanuelle Grislin-Le Strugeon, MC, HDR, Université de Valenciennes
	Marc Neveu, Professeur, Université de Bourgogne
Examinateurs :	Marc Jarry, Professeur, Université de Pau et des Pays de l'Adour
	Joël Savelli, MC, Université de Bourgogne
Co-encadrant :	Eric Gouardères, MC, Université de Pau et des Pays de l'Adour
Directeur :	Wilfrid Lefer, Professeur, Université de Pau et des Pays de l'Adour

Remerciements

Je tiens à remercier le Centre Scientifique et Technique du Bâtiment et le LIUPPA - EA3000 de l'Université de Pau et des Pays de l'Adour pour avoir pu utiliser leurs infrastructures et m'avoir fourni le matériel nécessaire à mes recherches.

Plus précisément, je remercie Wilfrid Lefer, Eric Gouardères et Florent Coudret pour leur disponibilité et leur écoute.

Concernant, les points critiques de ma recherche, je tiens également à citer Katri Matikainen pour m'avoir fourni les traductions anglaises d'articles décrivant le système d'évacuation finlandais Evac/FDS, ainsi que Nicolas Courty et Won-Ki Jeong pour leur réactivité à mes courriels.

Enfin, je remercie mes parents pour leur soutien durant ces quatre années.

Résumé : L'objectif de cette thèse est de proposer un modèle comportemental de l'être humain pour la simulation d'évacuation de bâtiment en cas d'incendie et de l'intégrer dans un outil de simulation d'évacuation de bâtiment. Le modèle proposé représente une approche individu-centré du comportement et répond aux axiomes de la Rationalité Limitée énoncés par Herbert Simon [Parthenay 2005] grâce à une conception hiérarchique des moyens cognitifs des agents simulés.
L'implémentation du modèle du comportement présente la particularité d'être intégralement réalisée en GPU (via OpenGL 2.0). Ainsi la fréquence du moteur de comportement est très proche de celle du simulateur et permet une adaptation quasi temps réel des comportements des agents à un changement de perception de leur environnement. Le logiciel d'évacuation développé permet :

- l'importation de scénario d'incendie de bâtiment simulé grâce au logiciel FDS,
- la configuration des archétypes de comportement des agents évacuant, notamment :
  - la description des hypothèses sur le monde (connaissance individuelle),
  - la configuration des comportements des agents (moyens cognitifs individuels).
- le positionnement des agents dans le bâtiment,
- la simulation de l'évacuation,
- l'enregistrement et le play-back d'un scénario d'évacuation.

Les contraintes (feu, fumée, obstacles, autres agents) captées par un agent sont interprétées par ce dernier en fonction de son archétype de comportement afin de déterminer si sa stratégie d'évacuation doit être remise en cause.

Mots clés : Simulation d'évacuation de bâtiment, diagnostic sécurité incendie, comportement humain en situation de crise, rationalité limitée.

Modelling, simulation and visualisation of human behavior during fire emergency building egress

Abstract : The main objective of this thesis is to propose a behavioral model of the human being in presence of several constraints and to integrate it into a simulation tool for building egress. The proposed model represents an individual-based approach of behavior modelisation and implement axioms of the bounded rationality set by Herbert Simon [Parthenay 2005] providing two key features :

– individual prioritization and parametisation of cognitive means,
– individual perception and knowledge management.

The proposed software has the particularity to run entirely on GPU via OpenGL 2.0. Thus the frequency of the behavior engine is very near to that of the simulator and allow adaptation of near real-time behavior of agents in a changing perception of their environment.

Keywords : Egress simulation, Human behavior in fire emergency case, Bounded rationality, Pathfinding

Table des matières

Chapitre 1

Introduction

Nous commençons par décrire le problème qui nous a été posé par notre partenaire industriel et le contexte du travail avant de conclure cette introduction par le traditionnel plan du document.

1.1 Simulation d'évacuation de bâtiment en feu

Les outils de simulation des diverses forces naturelles (incendie, vent, tremblement de terre) sur des modèles numériques des constructions de l'homme sont aujourd'hui massivement utilisés afin de mettre en exergue leurs faiblesses de conception avant même leur édification. De nombreuses normes ont été mises en place par des organismes, tels que le SFPE (Society of Fire Protection Engineers `http://www.sfpe.org`) pour le cas des incendies de bâtiment. Des organismes de certification octroient des labels, après expertise à l'aide des modèles de simulation de ces phénomènes naturels, développés par des ingénieurs. La plupart des simulateurs de propagation d'incendie actuels sont utilisés comme outil de diagnostic, pour le choix des matériaux en fonction de leur localisation dans le bâtiment et de leur potentiel combustible et toxique, ou encore pour l'implantation des dispositifs anti-incendie (extracteurs, extincteurs, alarme).

Cependant, tous ces diagnostics et précautions n'empêchent pas de dramatiques événements de se répéter :

- incendie de la fabrique de matelas de Casablanca (Maroc, 2008, 55 morts et 12 blessés) [1],
- incendie de discothèque à Shenzhen (Chine, 2008, 43 morts et 88 blessés) [2],
- incendie d'une tente lors d'un mariage (Koweït, 2009, 45 morts) [3],
- incendie de discothèque à Bangkok (Thaïlande, 2009, 60 morts et 243 blessés) [4],
- incendie dans une maison de retraite de Melle (Belgique, 2009 9 morts, 2 blessés) [5]...

Ces tristes événements soulignent la nécessité d'intégrer les occupants et leurs comportements dans les simulations d'incendie. En effet, simuler le comportement des individus présents lors d'un incendie peut permettre :

- d'optimiser la conception des bâtiments afin de favoriser leur évacuation en cas d'incendie, par exemple en élargissant les portes afin d'éviter des congestions trop importantes,
- d'implanter des aides à l'évacuation, signalisation des issues de secours, aux endroits les plus pertinents,
- de développer des procédures d'évacuation spécifiques à chaque bâtiment.

1.2 Contexte de la thèse

Nous décrivons maintenant le contexte dans lequel s'est déroulé ce travail, financé par une bourse CIFRE, et les contraintes matérielles et logicielles qui étaient les notres.

1. lien vers page LePost
2. lien vers page Nouvel Observateur
3. lien vers page France24
4. lien vers page Libération
5. lien vers page France24

1.2.1 Le Centre Scientifique et Technique du Bâtiment

Avec pour mission la sécurité et le bien être de l'homme dans ses constructions, le CSTB (Centre Scientifique et Technique du Bâtiment [6]), créé en 1947 est un établissement public à caractère industriel et commercial, placé sous la tutelle du ministre de l'Ecologie, de l'Energie, du Développement Durable et de la Mer, Direction de l'Habitat, de l'Urbanisme et des Paysages.

Organisme indépendant, le CSTB répond à des missions de service public tout en menant des activités industrielles et commerciales garantissant son équilibre financier.

Les missions du CSTB s'exercent dans les trois domaines suivants :

- la recherche scientifique et les techniques et expertises pour le secteur de la construction et du logement,
- l'amélioration de la qualité des constructions et de son environnement,
- l'amélioration de l'information des professionnels.

1.2.2 La division MOD-EVE

La mission de la division MOD-EVE repose principalement sur le couplage d'outils de simulation scientifiquement valides avec un environnement de réalité virtuelle immersif et réaliste (Salle Immersive "Le Corbusier" [7]).
Ce couplage permet une communication optimale entre Maîtrise d'Œuvre et Maîtrise d'Ouvrage en permettant la navigation interactive dans des maquettes virtuelles des sites urbains, objets de scénarios de simulation particuliers. Mais cette mission ne s'arrête pas là :

- la promotion du modèle d'échange IFC, standard du domaine de la construction (logiciel IFC-Viewer),
- la collaboration internationale avec la NASA pour les tests de structures (suite logicielle Baghera).

On peut citer comme principales simulations réalisées :

- l'étude du l'impact de l'implantation du tramway sur la pollution accoustique et atmosphérique du centre ville de Nantes,
- la simulation du trafic de véhicules au centre ville de Nice,
- simulation de l'éclairage nocturne de la ville de Rennes,
- simulation de tremblements de terre sur la Côte d'Azur (GIS Curare),
- l'étude aérodynamique du viaduc de Millau pour l'implantation des protections aux vents.

En plus du développement de ces simulations, la division MOD-EVE travaille sur l'immersion des environnements de simulation en y intégrant :

- un moteur temps réel de rendu accoustique scientifiquement valide grâce au logiciel Mithra (collaboration MODEVE/INRIA),
- divers dispositifs de navigation dans un environnement virtuel (joystick, Wiimote, dispositif à camera infrarouge...).

1.2.3 Architecture matérielle

Depuis la désuétude des solutions matérielles et logicielles graphiques de Silicon Graphics Incorporate (SGI [8]), un cluster de cartes graphiques (technologie SLI) de type NVIDIA [9] sous Linux a

6. http://www.cstb.fr
7. http://www.cstb.fr/le-cstb/equipements/realite-virtuelle.html
8. http://www.sgi.fr
9. http://www.nvidia.com

supplanté l'utilisation de l'ancien Onyx 300 (3 cartes graphiques /14 processeurs). L'architecture logicielle utilisée s'adapte aux nouvelles architectures matérielles de manière incrémentale afin de garder une compatibilité avec les anciens projets.

1.2.4 Outils de propagation de l'incendie

La division DSSF du CSTB utilise couramment le simulateur de propagation de feu Fire Dynamic Simulator (FDS [10]) développé par le National Institute of Standards and Technology (NIST [11]). Ce logiciel permet de simuler la combustion des matériaux constituant le bâtiment ainsi que la propagation du feu et de la fumée en son sein. Il est implémenté en Fortran, est multiplateforme et ses sources sont du domaine public (licence GPL).

1.2.5 Suite logicielle utilisée par la division MODEVE

Le moteur de simulation de la division MODEVE est basé sur Delta3D [12], il est développé en C++ sous licence LGPL et intègre diverses bibliothèques multimédia et de communication réseau afin de former un framework cohérent, fondé sur la communication par messages. Delta3D utilise la bibliothèque graphique OpenSceneGraph [13] pour le rendu 3D. Cette bibliothèque est une surcouche de l'API OpenGL basée sur une description hiérarchique des objets composants la scène sous forme d'un graphe de scène.

1.3 Problématique

La simulation d'évacuation de bâtiment en feu présente la particularité de se fonder sur un mécanisme très mal maitrisé : le comportement de ses occupants. En effet, comme Proulx l'a fait remarquer, aucune théorie ne fait l'unanimité dans le domaine de l'éthologie humaine [Proulx 2002] car :

- on manque de données fiables sur les interactions sociales entre individus lors de situations d'urgence,
- les chercheurs en modélisation du comportement humain sont peu enclins à fournir un modèle général car ce modèle aurait tendance à simplifier le phénomène et serait donc peu fiable.

Cependant, Sime souligne l'importance de l'intégration des facteurs psychologiques, au même titre que les facteurs d'ingénierie, dans les outils informatiques d'évaluation de la sécurité des infrastructures [Sime 1995].

Les divers facteurs influençant le comportement des évacuants doivent donc être sélectionnés avec méthode, et les processus cognitifs mis en jeu dans ce type de scénario modélisés avec pragmatisme afin de refléter une certaine rigueur scientifique de modélisation.
C'est dans cet objectif que cette thèse propose un modèle de simulation du comportement humain adapté aux situations d'urgence telles que les incendies de bâtiment.
Dans un premier temps, notre étude a permis de recueillir de la littérature les informations pertinentes concernant l'évacuation de bâtiment par ses occupants. Par la suite, un échantillon des outils existants a été analysé et critiqué selon les axes d'analyse dégagés. Puis, un nouveau modèle a été proposé afin d'y intégrer les techniques et critères de simulation les plus pertinents pour le cas d'un incendie dans un bâtiment. Ce modèle vise à combiner l'expressivité de la simulation avec des performances d'exécution se rapprochant du temps réel, en utilisant la puissance du matériel d'affichage actuel.

10. http://fire.nist.gov/fds
11. http://www.nist.gov
12. http://www.delta3d.org
13. http://www.openscengraph.org

1.4 Organisation du document

La première partie de ce document s'attache à présenter un état de l'art des domaines abordés. Une étude succincte de la littérature de la psychologie est réalisée afin de déterminer les critères psychologiques individuels et sociaux pertinents à modéliser pour la simulation de mouvement de foule en cas d'urgence. Puis plusieurs chapitres s'attachent à décrire l'état de l'art des diverses techniques informatiques permettant la simulation du comportement humain. Le domaine de la simulation de foule y est analysé au travers des différents mécanismes impliqués dans la simulation du comportement (perception, mémorisation, décision, action). Notre approche s'appuyant sur un modèle individu-centré, les principales architectures du domaine des Systèmes Multi-Agents sont ensuite présentées. Les divers systèmes de simulation d'évacuation de bâtiment existants sont ensuite analysés et évalués selon les critères identifiés. Cet état de l'art permet alors de dégager les principaux axes de développement du simulateur proposé.

La seconde partie décrit les différents travaux et contributions effectués aux regard des axes de recherches établis. Une architecture de simulation générique est proposée. Puis elle est instanciée pour le cas de l'évacuation de bâtiment en feu. Son implémentation est ensuite détaillée et notamment le recours au calcul sur GPU afin d'accélérer les traitements. Enfin nous présentons les résultats obtenus à travers les possibilités offertes à l'utilisateur du logiciel, un scénario d'évacuation de bâtiment en feu et une évaluation de la performance de notre implémentation sur GPU.

Nous concluons en rappelant les diverses propriétés du simulateur développé et les extensions possibles de cette thèse.

Première partie

Etat de l'art

Chapitre 2

Psychologie des mouvements de foule

2.1 Généralités

Les diverses études du domaine de la psychologie ont permis d'émettre certaines hypothèses sur l'apparition de comportements spécifiques lors de mouvement de foule en cas d'urgence. Les différentes approches de la modélisation du domaine de la psychologie et de l'éthologie se distinguent par le référentiel de modélisation qu'elles exploitent. Certains psychologues considèrent que l'étude du comportement doit être abordée uniquement sous l'angle des comportements mesurables produits en réponse à des stimuli de l'environnement. L'objet de l'étude est le comportement observé en faisant abstraction des processus mentaux internes de l'individu, considéré comme une "boite noire". Au contraire d'autres jugent que la compréhension des comportements ne peut s'abstraire de ces mécanismes internes. L'objet de l'étude concerne donc les processus mentaux menant à une décision, le comportement étant la conséquence de cette décision. Ce deux visions de l'analyse psychologique ont donné naissance à deux grandes tendances : la psychologie comportementale et la psychologie cognitive.

2.1.1 Psychologie comportementale

La psychologie comportementale, appelée aussi comportementalisme ou behaviorisme est basée sur l'axiome selon lequel l'analyse psychologique doit se contenter de l'observation externe, comme les autres sciences naturelles. Abandonnant explicitement l'étude des faits subjectifs de conscience, le comportementalisme choisit donc pour objet le comportement en tant que phénomène observable. Les comportementalistes rejettent toute hypothèse structurale relative aux processus mentaux : « L'étude du comportement consiste à établir les relations qui existent entre les stimulations et les réponses de l'organisme. » (Fraisse 1973). On a souvent affirmé que les comportementalistes réfutaient l'existence de processus mentaux s'intercalant entre stimulus et réponse. Il est certain que les postulats épistémologiques du comportementalisme ont amené les chercheurs de ce courant à minimiser le rôle des représentations et du traitement interne des informations. Si la « boîte noire » qu'est l'être humain est considérée comme scientifiquement impénétrable, à aucun moment les comportementalistes ne postulent l'inexistence de processus internes. Cependant, ils s'en affranchissent dans leurs théories et leurs analyses.

2.1.2 Psychologie cognitive

L'idée fondatrice de la psychologie cognitive, ou cognitivisme, est « d'identifier les processus mentaux, même inconscients, qui s'intercalent entre stimulus et réponse. » (George et Richard 1982). Les comportementalistes doutaient de la possibilité d'une investigation scientifique de ce qu'ils appelaient la « boîte noire ». Les cognitivistes, à partir des années 50, vont tenter de dépasser ce postulat en cherchant à identifier les structures et les processus hypothétiques responsables du comportement. Les deux notions de base sont l'information et le traitement de l'information. L'information est représentée par des symboles manipulés par des règles rationnelles (les traitements) permettant de déduire de

nouvelles informations qui peuvent à leur tour être traitées jusqu'à déduire la solution au problème posé.

La première application informatique exploitant cette approche est due à un partenariat entre Allen Newell (informatique et psychologie cognitive) et Herbert Simon (économie et sociologie), avec le GPS - General Problem Solver [Newell 1972] - dont le but ambitieux était de créer un « résolveur de problème humain » générique permettant de résoudre tout problème humain dans la mesure où il a été correctement formalisé. Ce fut le premier modèle complet du traitement humain de l'information. Démodé à ce jour, il contient malgré tout les concepts fondateurs des travaux actuels de traitements symboliques de l'information.

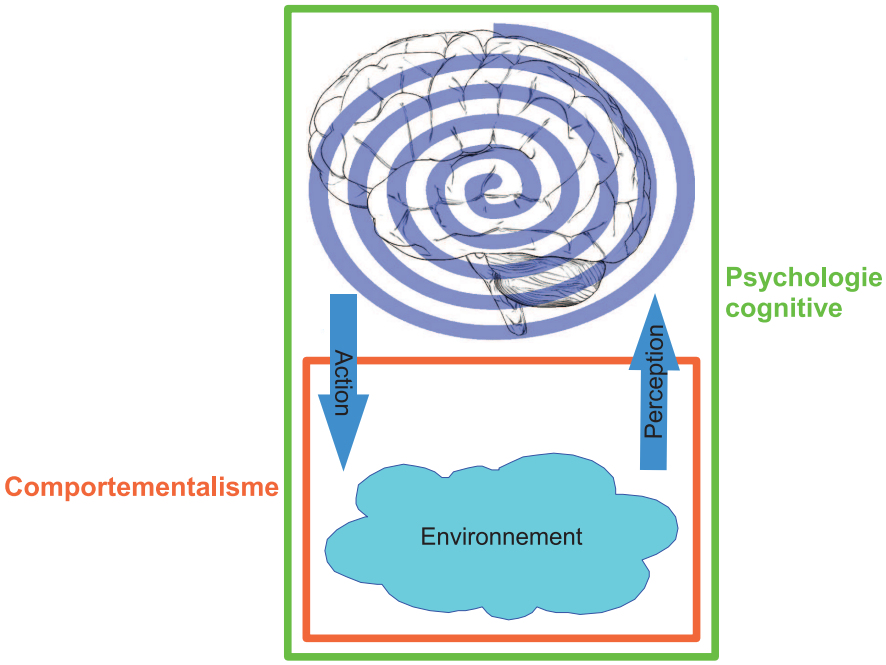


FIGURE 2.1 – Illustration des approches de la psychologie.

2.1.3 Synthèse

Ces deux orientations de l'analyse comportementale ont toutes deux donné naissance à des théories tendant à formaliser les mécanismes menant à l'observation de certains comportements. Cependant, ces différentes théories reposant sur des hypothèses aussi incertaines que diverses, l'existence d'un modèle formel du comportement de l'être humain reste donc du domaine de l'expectatif.

Dans la partie qui suit, nous allons recenser et classer diverses hypothèses relevées dans la littérature concernant le comportement humain en situation d'incendie et tenter une mise en relation de ces hypothèses, incompatibles à première vue, au sein d'une approche commune.

2.2 Hypothèses psychologiques individuelles

2.2.1 Importance de l'espace personnel

D'un point de vue comportementaliste, le respect de l'espace individuel peut être vu comme une règle qui, lorsqu'elle est violée dans un mouvement de foule, entraîne beaucoup plus de stress et d'agitation pour les individus que lorsqu'ils sont hors de la foule (Sommer 1969) en créant des zones

de forte densité d'occupation.
Ces fortes densités de foule augmentent l'inconfort et le risque pour l'individu (Fruin 2002, Bryan 2003). Dans ces conditions, les actions de l'individu, notamment le mouvement, peuvent alors être perçues comme irrationnelles car principalement causées par les forces physiques qu'il subit.

« L'Homme est un animal social, il définit et marque son territoire pour sa propre utilisation, il définit des frontières visibles ou invisibles qu'il attend que l'on respecte, il le défendra alors contre toute intrusion. » (Ashcraft et Scheflen 1976)

2.2.2 Influence du stress sur la décision

L'importance du stress dans le processus décisionnel a été soulignée par Billing en 1980. Selon lui le stress serait une fonction de la perte perçue, du temps disponible pour la décision et de l'incertitude perçue de la situation. D'autres travaux comme l'hypothèse de l'U-inversé (Yerkes-Dodson [Staal 2004]), cherchent à établir une corrélation entre le stress et les performances à la fois physiques et rationnelles d'un individu.

2.3 Les hypothèses psychologiques sociales

2.3.1 Principe de l'identité sociale

Ce principe représente la place que l'individu croit avoir dans la société. Braun a émis en 2003 l'hypothèse selon laquelle un individu dans une foule agit différemment que s'il était seul. Le modèle de March (1994) tend à renforcer cette hypothèse. Il formalise cette relation sociale par la règle suivante :
Si l'individu qui prend les décisions peut être identifié
Alors agir en fonction de son rôle
Sinon agir de son propre chef (et ainsi pouvoir être identifié comme décideur par d'autres).

2.3.2 Principe de la preuve sociale

Le principe de preuve sociale concerne le mimétisme qui peut apparaître dans des situations d'indécidabilité : *« Si les individus font tous la même chose c'est qu'ils savent quelque chose que j'ignore donc je fais comme eux. »* (Cialdini 1993).
Ce principe peut également être renforcé par d'autres facteurs comme celui de l'inhibition sociale : *«Personne ne veut paraître dépassé lors de situations d'urgence, tout le monde agit donc initialement avec calme en observant les réactions des autres »* (Batson 1998).

De ces divers principes, on peut dégager de grandes variations du comportement pour un unique individu. Ce dynamisme du comportement est fortement lié à la perception qu'ont les individus de leur situation ainsi qu'au stress qu'ils éprouvent. On peut donc remarquer que ces facteurs individuels influencent l'adaptation des individus à leur environnement. Nous allons à présent nous focaliser sur quelques théories cherchant à expliquer l'apparition de comportements non adaptatifs lors de situations d'urgence.

2.3.3 Théories d'apparition de comportements non adaptatifs

Selon l'analyse de X.Pan [Pan 2006], les comportements non adaptatifs sont causés par les hautes densités d'individus, les contraintes environnementales sévères et le haut niveau d'émotion (stress).
La théorie de rationalité limitée d'Herbert Simon [Parthenay 2005] pose comme postulat que les

capacités de raisonnement de l'être humain sont limitées en termes de capacités cognitives et d'informations disponibles, ce qui expliquerait également cette inadaptation.

De nombreuses études ont également posé des hypothèses sur les causes des comportements non adaptatifs des êtres humains en situation de crise. Ces approches ont des bases de réflexion différentes et même parfois antithétiques, ce qui explique leur diversité [Le Bon 1982] [McDougall 1973] [Quarantelli 2001].
Par exemple, concernant le non respect de l'espace individuel de ses congénères, plusieurs hypothèses ont été formulées :

- comportementalisme : les individus tendent à préserver leur espace individuel [Bryan 2003] mais quand la densité de la foule excède un certain seuil, le maintien de l'espace individuel devient impossible ce qui mène à des comportements non adaptatifs (Still 2000).
- psychologie cognitive :
  - théorie de la panique (Le Piere 1938, Le Bon 1960, McDougall 1920, Smelser 1963) : les personnes poussent dans la foule car c'est un comportement instinctif de préservation de l'individu.
  - théorie de la prise de décision (Brown 1965, Mintz 1951) : les personnes poussent dans la foule car c'est leur raison qui leur fait croire que ce comportement est préférable pour maximiser leur efficacité.
  - théorie du niveau d'urgence (Kelley et al 1965) : plus la situation apparait comme critique aux individus plus ils poussent.

L'approche comportementaliste a une vision statique du problème, le comportement est toujours le même et c'est la pression des autres qui engendre de potentielles collisions.
A l'inverse, les diverses hypothèses cognitives, qui à première vue, peuvent apparaître comme totalement différentes peuvent cependant être unifiées sur une échelle d'adaptation du comportement.
Un exemple d'unification par une échelle d'adaptation comportementale a été introduite par [Maslow 1943] lors de ces recherches sur la motivation. La pyramide des besoins (figure 2.2) schématise cette échelle à partir des observations réalisées dans les années 1940 sur la motivation.

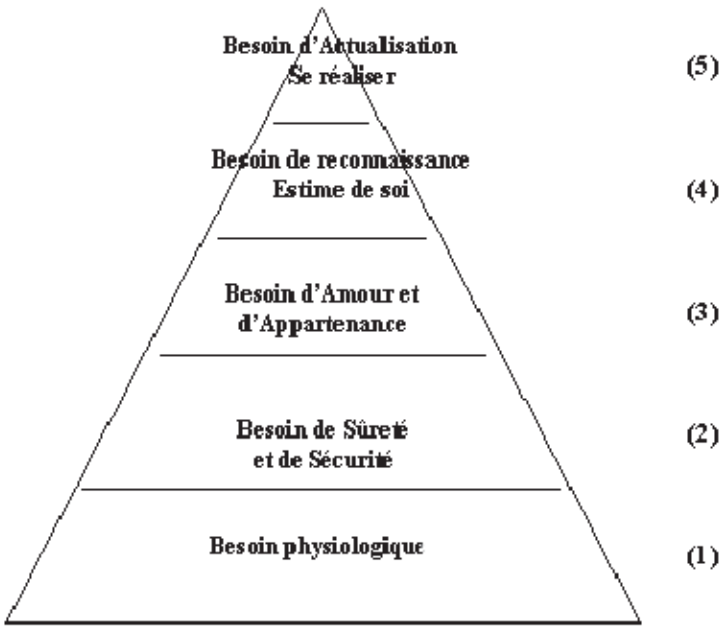


FIGURE 2.2 – Pyramide illustrative des travaux de Maslow sur la motivation.

La pyramide est constituée de cinq niveaux. Un individu recherche d'abord à satisfaire chaque besoin d'un niveau donné avant de penser aux besoins situés au niveau immédiatement supérieur de la pyramide. L'application de ce modèle pour la classification des hypothèses cognitives précédentes offre un moyen de les regrouper dans un même modèle :

1. Au niveau du besoin physiologique, on peut appliquer la théorie de la panique.
2. Au niveau du besoin de sécurité, la théorie du niveau d'urgence prévaut.
3. Aux niveaux supérieurs, où s'expriment des besoins d'accomplissement personnel, la théorie de la prise de décision semble plus adaptée à la maximisation de sa propre efficacité.

On voit alors apparaître une hiérarchie de modes de prise de décision.
Il est commode de représenter cette hiérarchie sous la forme d'une pyramide. Cependant, pour Maslow, cette hiérarchie est dynamique car l'ordre des niveaux n'est pas forcement figé.
Nous avons cité cette théorie pour illustrer un exemple d'unification. Mais, cette théorie, ainsi que la théorie du U inversé, sont souvent sujet à controverse [Staal 2004]. C'est pourquoi, dans nos travaux, nous nous contenterons d'expliquer l'inadaptation des comportements par les axiomes de base de la rationalité limitée d'Herbert Simon.

Dans la partie suivante, les différentes techniques de simulation comportementale par des moyens informatiques sont présentées.

Chapitre 3

Simulation de foule

3.1 Mécanismes cognitifs individuels

3.1.1 Généralités

La psychologie cognitive pose comme postulat que le comportement n'est pas seulement dû à une réaction à l'environnement mais est résultante des traitements internes que l'individu applique aux informations collectées dans l'environnement. Les mécanismes impliqués sont principalement :

1. La perception : l'individu ne perçoit pas forcément toutes les informations de l'environnement mais subit sa subjectivité d'observation,
2. La symbolisation des perceptions : l'individu donne du sens aux perceptions afin de pouvoir raisonner sur des symboles qu'il sait manipuler,
3. La mémorisation : l'individu stocke et extrapole les informations qu'il collecte afin de prendre des décisions les meilleures possibles,
4. La prise de décision : l'individu prend la décision la plus adaptée à son problème en fonction de ses connaissances et de ses capacités.

La figure 3.1 illustre le fonctionnement d'une simulation basée sur ces mécanismes. Les traitements sont effectués pour chaque individu à chaque instant de la simulation. Ce type de simulation est appelé **simulation microscopique** dans la mesure où chaque individu se forge sa propre solution au problème en fonction de ses connaissances et habilités propres.

La conception de ce type de simulation s'appuie donc sur une modélisation dite **individu-centrée** telle que définie par Craig Reynolds, c'est-à-dire une simulation basée sur les conséquences globales d'interactions locales des membres d'une population. Ces individus peuvent représenter des plantes et des animaux dans les écosystèmes, les véhicules en circulation, les gens dans la foule, ou des personnages autonomes dans l'animation et les jeux. Ces modèles sont généralement constitués d'un environnement dans lequel les interactions se produisent et d'un certain nombre de personnes définies en fonction de leurs comportements (règles procédurales) et de paramètres caractéristiques. Dans un modèle individu-centré, les caractéristiques de chaque individu sont suivies dans le temps. Ceci est en contraste avec des techniques qui adoptent une approche globale, dite macroscopique, qui travaille à partir de valeurs moyennes des caractéristiques de la population. Le modèle est alors utilisé pour simuler l'évolution de ces caractéristiques moyennes pour l'ensemble de la population.
La modélisation individu-centrée peut être considérée comme un sous-domaine des systèmes multi-agents, paradigme pour l'étude et la conception de systèmes informatiques qui suppose que les éléments constitutifs du système, les *agents*, sont capables d'interagir entre eux et d'effectuer des actions autonomes dans un environnement donné. Selon Ferber [Ferber 1997] : « un agent est une entité physique ou virtuelle (a.) qui est capable d'agir dans un environnement, (b.) qui peut communiquer directement

avec d'autres agents, (c.) qui est mue par un ensemble de tendances (sous la forme d'objectifs individuels ou d'une fonction de satisfaction, voire de survie, qu'elle cherche à optimiser), (d.) qui possède des ressources propres, (e.) qui est capable de percevoir (mais de manière limitée) son environnement, (f.) qui ne dispose que d'une représentation partielle de cet environnement (et éventuellement aucune), (g.) qui possède des compétences et offre des services, (h.) qui peut éventuellement se reproduire et (i.) dont le comportement tend à satisfaire ses objectifs, en tenant compte des ressources et des compétences dont elle dispose, et en fonction de sa perception, des ses représentations et des communications qu'elle reçoit ».
L'environnement dans lequel les agents évoluent peut être spatialement défini en s'appuyant sur une représentation géométrique. Ceci permet d'associer les agents à une localisation dans l'espace et permet également la mobilité des agents dans l'environnement. On parle alors de systèmes multi-agents situés. Ces modèles situés peuvent utiliser une représentation de l'espace discrète (valeurs entières, grille) ou continue (valeurs réelles).

La simulation d'évacuation de bâtiment en cas d'incendie étant typiquement un problème spatialement situé, on parlera alors d'**agent situé** lorsque l'on fera référence aux évacuants virtuels.

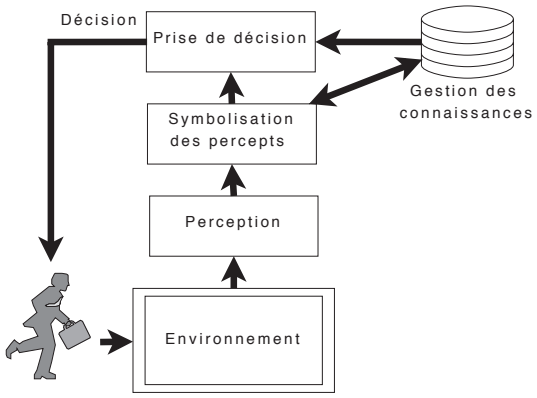


FIGURE 3.1 – Modèle cognitif simulé

Nous allons à présent présenter les divers travaux de la littérature concernant la simulation microscopique de ces processus pour le cas de l'évacuation de bâtiment en cas d'incendie.

3.1.2 La perception visuelle

La perception du monde lors de l'évacuation d'un bâtiment est principalement liée au sens de la vision. Mais dans ce type de situation, il est perturbé par des facteurs comme :

- la fumée,
- la forte densité de la foule,
- les murs.

Les autres sens n'étant pas altérés par les contraintes de l'environnement, peu de modèles ont été recensés concernant leur simulation.

3.1.2.1 Voir à travers la fumée

[Rubini 2007] décrit les transferts radiatifs dans des environnements enfumés en se basant sur la loi de Beer-Lambert-Bouguer (voir équation (3.1)) :

$$\frac{I}{I_0} = e^{-\int_0^l \alpha_{ext} dz} \tag{3.1}$$

I : intensité transmise de la lumière
I_0 : intensité incidente de la lumière
α_{ext} : coefficient d'absorption de la lumière
l : distance à l'observateur
dz : différentiel de distance à l'observateur

C'est le modèle préconisé par Jin dans le SFPE HandBook [Association 2002]. Il est basé sur les conditions d'éclairage de l'environnement. En l'absence de telles informations, la visibilité peut être approximée en utilisant le modèle suivant :

$$\alpha_{obs} = \int_0^L \alpha(s) ds \tag{3.2}$$

α_{obs} : opacité cumulée le long du rayon depuis sa source (l'agent).
L : longueur du rayon de l'agent jusqu'au dernier point visible (fumée ou obstacle opaque).
$\alpha(s)$: opacité locale de fumée à l'abscisse s du rayon.

3.1.2.2 Perception des panneaux d'aides à l'évacuation

Les travaux de [Xie 2007] offrent un modèle de lisibilité des panneaux d'aides à l'évacuation où la perception serait liée à la fois à la distance (AB) et à l'orientation d'observation (θ) mais sans prendre en compte l'enfumage des lieux (voir figure 3.2). Un cercle ($M2$) circonscrit aux bornes du panneau (C et X) définit l'espace au sein duquel le panneau est lisible. Ces travaux ont été validés par l'expérimentation. Le SFPE HandBook [Association 2002] a mis en avant un modèle simpliste

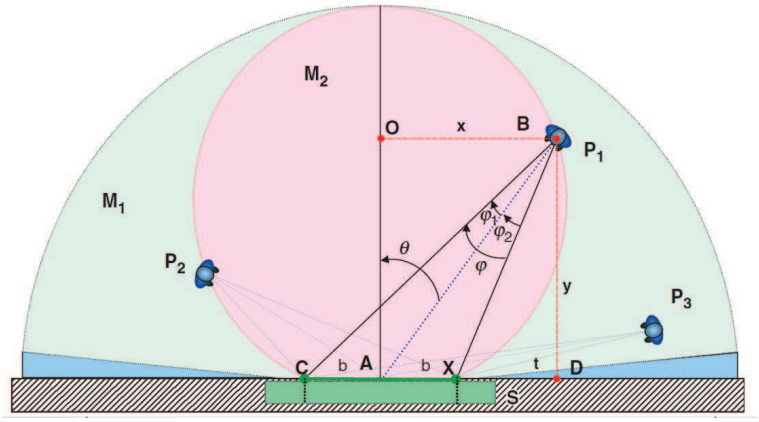


FIGURE 3.2 – Illustration de la lisibilité des panneaux d'aide à l'évacuation

afin d'intégrer la lumière émise par certains panneaux dans le modèle classique de visualisation , qui consiste à modifier la fonction de transfert pour les rayons de perception intersectant les panneaux.

3.1.3 Symbolisation des percepts

Ce processus consiste à transformer les percepts collectés dans l'environnement en symboles rationnels manipulables par un processus comportemental. Pour le cas de l'évacuation de bâtiment en feu, ce processus se charge donc d'enrichir les représentations des percepts afin que ces derniers soient manipulables par les comportements du système. En effet, à l'instar du domaine de la psychologie, les études du domaine des Systèmes Multi-Agents ont dégagé deux types de comportements :

- Les comportements navigationnels ou réactifs (psychologie comportementale) transforment les percepts bruts collectés dans l'environnement en réponse à ces stimuli,
- les comportements cognitifs (psychologie cognitive) transforment l'ensemble des symboles manipulables à la dispostion d'un agent par les règles rationnelles du système en une décision satisfaisant un objectif.

La deuxième classe nécessite une transformation des signaux perçus en représentations ayant une sémantique compatible avec les comportements du système qui les manipule.

Chaque symbole peut revêtir deux représentations duales :

- une représentation vectorielle, décrivant la localisation de l'information de façon géométrique (exemple : cercle=(position,rayon)) ;
- une représentation volumique, décrivant la localisation de l'information sous la forme d'un ensemble de cellules, chacune portant la description d'un état discret (exemple : densité d'occupation de l'espace).

Les principaux systèmes à base de règles étant de nature discréte, ils se fondent sur une partition de l'espace. La principale transformation de percept est alors la rastérisation ($\approx$ discrétisation spatiale) d'une information vectorielle.

3.1.3.1 Carte cognitive

Les cartes cognitives sont des représentations des croyances d'une personne, d'un agent ou d'un domaine particulier. Le but est d'exprimer les relations causales existant entre les concepts d'un système d'une manière simple. On définit les cartes cognitives comme des graphes orientés.
Plus formellement, une carte cognitive X est une paire X = (C, A) où C est l'ensemble de nœuds ou concepts et A est l'ensemble des arcs orientés.
La direction de chaque arc indique la direction d'influence ou de causalité. Les relations causales peuvent prendre trois différentes valeurs : + (influence positive), - (influence négative) et 0 (influence nulle).
Pour notre cas d'étude les influences positives sont des comportements d'attraction et les influences négatives des comportements de répulsion.

3.1.3.2 Carte mentale

Pour le cas de problèmes spatiaux comme l'évacuation de bâtiment, on s'intéresse principalement au déplacement des agents. Les concepts manipulés sont alors les influences des différentes contraintes de mouvement. Ces influences sont des valeurs algébriques positives (attraction) ou négatives (répulsion). Ce cas spécial de carte cognitive est appelé carte mentale et représente donc les influences localisées des diverses contraintes de mouvement dans le problème d'évacuation.

Sur la figure 3.3, une carte mentale des obstacles est représentée pour un agent omniscient ainsi que pour un agent non omniscient construisant sa représentation du monde par un mécanisme de perception par lancer de rayons.

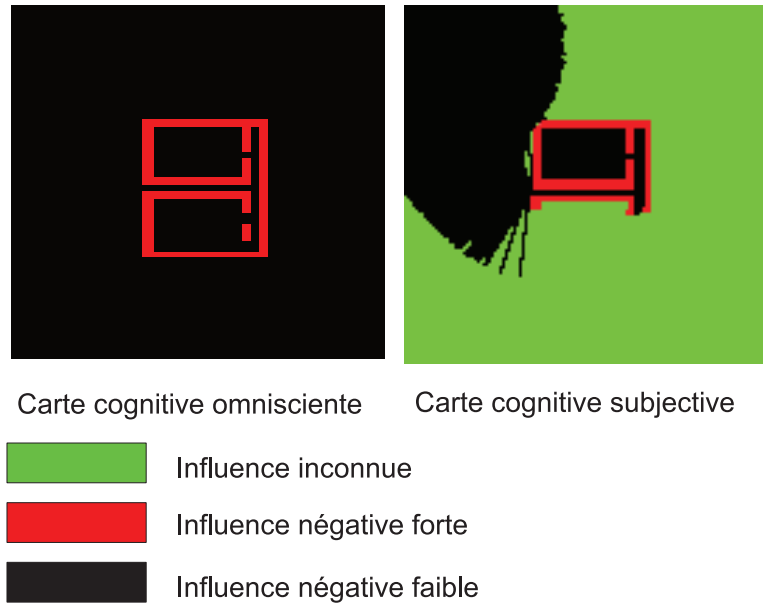


FIGURE 3.3 – Exemples de cartes mentales

Pour notre problème, le processus de symbolisation des percepts par cartes mentales consiste à calculer une représentation spatiale du coût pour chaque contrainte de mouvement. Ces cartes seront par la suite manipulées par un processus rationnel.

3.1.4 Gestion de la mémoire

Les architectures cognitives reposent en général sur une mémoire à long terme (procédurale) qui permet de stocker les règles d'actions, et une mémoire à court terme (de travail) qui contient l'état de l'agent ainsi que son dernier plan d'actions. [Newell 1972] suggère que, pour le cas de l'humain, l'information soit stockée dans plusieurs mémoires ayant des capacités et caractéristiques d'accès différentes : ainsi la mémoire à court terme (S.T.M.) a une capacité limitée et/ou une volatilité importante, alors que la mémoire à long terme (L.T.M.) a une large capacité et une durée de vie beaucoup plus importante, mais est caractérisé par une fixation plus lente et des temps d'accès plus longs.

3.1.5 Prise de décision

Elle correspond au traitement de l'information collectée (percepts) et sémantisée (cartes mentales) contenue dans les différentes mémoires dans le but de produire une décision. L'application de cette décision consiste à effectuer des actions dans l'environnement et constitue le comportement observable de l'individu. C'est donc le mécanisme le plus important à modéliser.

3.1.5.1 Le paradigme de l'action dans les Systèmes Multi-Agents

Dans le domaine de l'Intelligence Artificielle et des Systèmes Multi-Agents, le paradigme de l'action discrète unidimensionnelle s'est généralisé. Cela signifie qu'une seule action peut être entreprise par un agent à un instant de la simulation. Cela se répercute dans toutes les architectures hybrides cognitive/réactive recensées : un choix du centre décisionnel duquel sera issue l'action à entreprendre

est nécessaire. L'action est alors soit générée par une planification cognitive soit par une résolution réactive du comportement mais jamais les deux à la fois.

3.1.5.2 Le paradigme de l'action en simulation en environnement continu

Dans le domaine de la simulation, le paradigme de l'action continue est préféré car il permet un plus grand degré de liberté aux trajectoires. En effet, une action continue peut être déterminée lorsque plusieurs actions sont entreprenables en même temps sur des dimensions spatiales différentes. Par exemple pour le cas du déplacement en environnement continu, deux actions de dimensions orthogonales seraient : "Aller au Nord" (coût 3) et "Aller à l'Est" (coût 2). Au lieu de choisir la moins coûteuse des actions, on peut alors calculer un compromis entre les 2 actions possible et ainsi obtenir un vecteur de déplacement spatial $\overrightarrow{V}(\frac{1}{3}, \frac{1}{2})$. Cela induit donc un plus grand degré de liberté dans les actions possibles.

Maintenant qu'ont été décrites les informations que les comportements transforment en décision effective, les deux chapitres suivants vont décrire les différentes types de comportements engendrant une décision spatialement située. Contrairement au paradigme multi agent, nous nous focaliserons sur les comportements situés en environnement continu. De plus, l'axe de l'étude étant l'adaptation des comportements, nous classifierons les typologies de comportement selon l'optimalité de la décision vis à vis d'un objectif. Nous distinguerons alors les comportements navigationnels, qui satisfont une/des contrainte(s) sans tenir compte d'un objectif et les comportements planifiés qui tendent à optimiser l'atteinte d'un objectif en intégrant l'ensemble des contraintes.

3.2 Comportements navigationnels

3.2.1 Généralités

Les comportements navigationnels représentent les comportements issus de la branche comportementaliste de l'éthologie. Ils cherchent à reproduire les motifs physiques résultant d'une réaction à un stimulus extérieur.
Ils sont généralement utilisés en simulation en complément d'une approche planifiée du mouvement, afin d'intégrer les changements locaux de l'environnement sans faire appel à un processus coûteux de replanification. Ils se distinguent des comportements réactifs de la terminologie agent dans la mesure où ils n'intègre pas de notion d'objectif global, et de plus peuvent dans certains cas être issus d'un processus de planification locale [Ben Amor 2003]. Leur spécificité est qu'ils se concentrent sur des problèmes locaux.
Un comportement navigationnel s'exprime dans le domaine continu par une fonction de l'état du mobile qui engendre une force qui influence généralement une composante planifiée du comportement.
Une problèmatique de cette thèse étant d'étudier d'adaptation du comportement aux contraintes, une classification arbitraire des divers comportements navigationnels a été adoptée sur ce critère :

1. les comportements sans projection d'état,
2. les comportements avec projection d'état,
3. les comportements à projections d'états multiples.

3.2.2 Les comportements sans projection d'état

Ce type de comportement se base principalement sur la description du comportement tel qu'il a été observé à un instant t. C'est la manière classique de décrire le comportement d'un mobile. Il n'implique

aucune anticipation de l'état du mobile car s'exprime dans des domaines non dynamiques (position ou distance).
Ce type de comportement s'intègre aisément dans un modèle d'intégration physique du mouvement mais est bien connu pour les phénomènes d'oscillation qu'il implique. Un exemple typique de comportement navigationnel est contenu dans le comportement de panique de Helbing [Helbing 2000] qui s'exprime ainsi :

$$m_i \dot{v} = m_i \frac{Vmax_i \overrightarrow{e_i(t)} - \overrightarrow{v_i(t)}}{\tau} + \sum_{j \in I} \overrightarrow{f_{ij}} + \sum_{w \in W} \overrightarrow{f_{iw}} \tag{3.3}$$

m_i : masse de l'agent i
I : ensemble des agents
$Vmax_i$: vitesse maximum de l'agent i
$\overrightarrow{f_{ij}}$: répulsion de l'agent i pour l'agent j
$\overrightarrow{f_{iw}}$: répulsion de l'agent i pour le mur w

Cette expression fait apparaître deux composantes navigationnelles distinctes :
- la répulsion des congénères (ordre 0) : $\sum_{j \in I} \overrightarrow{f_{ij}}$
- la répulsion des murs (ordre 0) :$\sum_{w \in W} \overrightarrow{f_{iw}}$

3.2.3 Comportements à projection d'état simple

La spécificité de ces comportements est d'intégrer une projection simple de l'état du mobile dans le temps à partir de l'état du mobile (vitesse, masse, ...) dans l'expression du mouvement. Cette projection est réalisée en appliquant au mobile un modèle d'évolution de son état dans le temps.
Cela permet en particulier une meilleure anticipation de l'inertie du mobile et des contraintes et donc moins d'oscillations du comportement. Reynolds [Reynolds 2006] a augmenté la proactivité des comportements de piétons en utilisant une intégration lagrangienne simple (Steering).

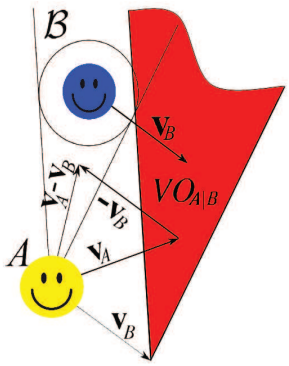


FIGURE 3.4 – Exemple de Velocity Obstacle simple

Récemment un autre type de comportement est apparu, il intégre une meilleure appréhension de la dynamique de l'environnement dans lequel évolue l'agent, en se basant sur le concept de "Velocity Obstacle" [Wilkie 2009]. Il consiste à déterminer l'espace des vitesses dans lequel les collisions sont possibles (voir figure 3.4) puis à éviter cet espace par l'application de forces.
Ce type de comportement est intéressant car il permet ainsi de satisfaire toutes les contraintes physiques liées à des obstacles géométriques mouvants ou immobiles sans violer ces dernières.
Afin d'éviter certains phénomènes d'oscillations, ces comportements ont été revisités pour donner

naissance au "Reciprocal Velocity Obstacle" [van den Berg 2008] et à d'autres dérivés (HRVO).
Cette classe de comportement inclut alors une conscience de la dynamique des agents et des contraintes dans l'expression des comportements.

3.2.4 Comportements à projections d'états multiples

Les travaux de Reynolds [Ben Amor 2003] ont été utilisés pour donner naissance au "Steering inverse" (voir figure 3.5).

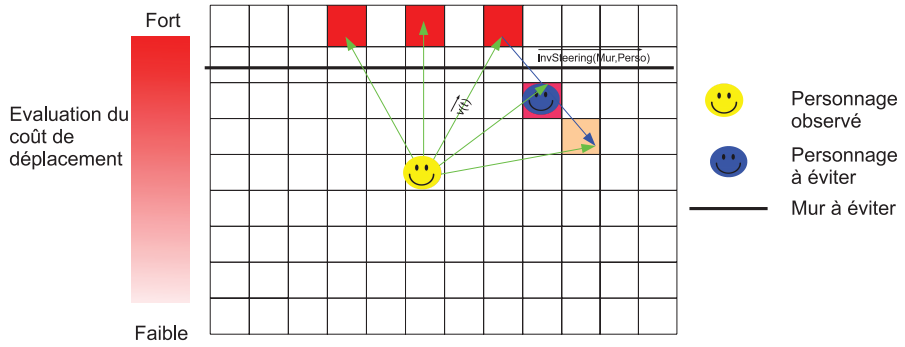


FIGURE 3.5 – Exemple de comportement de Inverse Steering sur un seul pas de temps

Le principe de ces comportements repose sur l'association d'une fonction d'évaluation de performance à chaque comportement. Ainsi, les différentes évolutions possibles de l'état de l'agent - enchaînements de comportements atomiques - peuvent être évaluées afin de sélectionner la meilleure au sens d'un certain critère de coût.

Il est à noter que cette modélisation du comportement peut également être considérée comme un comportement planifié car la fonction d'évaluation cherche à satisfaire l'ensemble des contraintes et intègre généralement une notion de but global (heuristique ou autre) permettant de satisfaire la convergence du comportement vers l'objectif.

Les divers comportements présentés permettent de résoudre les contraintes locales de mouvement. Cependant, la notion explicite de but global au problème y est généralement ignorée.
Dans le chapitre suivant, nous aborderons les techniques informatiques permettant de décider du comportement à entreprendre en prenant en compte cette notion d'objectif.

3.3 Planification de chemin

3.3.1 Généralités sur la planification d'actions

La planification est un mécanisme classique de l'Intelligence Artificielle qui consiste à établir un plan d'actions visant à satisfaire un objectif à partir d'un état initial.
Elle peut donc être formalisée ainsi :

$$Plan = Planification(E_{Obj}, E_0, A) \tag{3.4}$$

$Plan$: plan constitué d'une suite d'actions $a \in A$
$Planification$: processus de planification
E_{Obj} : caractérisation partielle ou totale d'un état but du problème
E_0 : état initial de la planification
A : ensemble des actions du système permettant de le faire évoluer vers un autre état

3.3.1.1 Définition d'un état

Un état de planification représente un état possible dans le contexte de la résolution du problème de planification qu'il définit. Il est à différencier du concept d'état du mobile représentant les propriétés de l'agent planifiant. Un état de planification est constitué de propriétés représentant les faits de la planification (état du mobile + état de l'environnement).
Exemple :
Contexte de planification de l'intention "Je veux manger" :
Etat possible -> ("Je n'ai rien à manger", "Paul a une pomme")

3.3.1.2 Définition d'une action

Une action représente une règle d'évolution qui, à partir d'un état, permet de générer un nouvel état de résolution. Elle est caractérisée par :

- une précondition d'application indiquant si oui ou non l'action peut être entreprise,
- une règle d'évolution de l'état permettant la génération d'un nouvel état,
- une postcondition indiquant l'état attendu du système après l'action.

Par exemple, l'action "Manger" peut avoir la définition suivante :

- précondition : "J'ai une pomme"
- règle : "J'ai une pomme" $\implies$ ¬ "J'ai Faim" ET ¬ "J'ai une pomme"
- post-condition : ¬ "J'ai une pomme" ET ¬ "J'ai faim"

3.3.1.3 Définition d'un but

Le but représente un état terminal du processus de planification. Il correspond à un état satisfaisant le but du problème, sa définition est donc une caractérisation généralement partielle d'un état.
Exemple : caractérisation du but dans le problème "Je veux manger" : ¬ "J'ai faim". L'état (¬ "J'ai faim", "Paul a une pomme"') satisfaisant cette caractérisation est donc un état but.
La figure 3.6 illustre un processus de planification simple.

Les algorithmes de planification se caractérisent par :

- l'ordre de la logique utilisée : pour définir les différents états du monde , on peut utiliser plusieurs formalismes logiques comme la logique des propositions, des prédicats ou encore un formalisme purement mathématique,
- le sens d'application des actions : l'application des règles peut être réalisée en partant de l'état initial jusqu'à atteindre un but par déduction (chaînage avant ou déductif), ou au contraire être effectuée à partir des buts jusqu'à remonter à l'état initial, les actions inversées sont alors vues comme des hypothèses qui sont validées si elles permettent l'atteinte de l'état initial (chaînage arrière ou inductif). On peut également utiliser un mélange des deux afin de réviser des états déjà établis par déduction (chaînage avant avec retour sur trace),

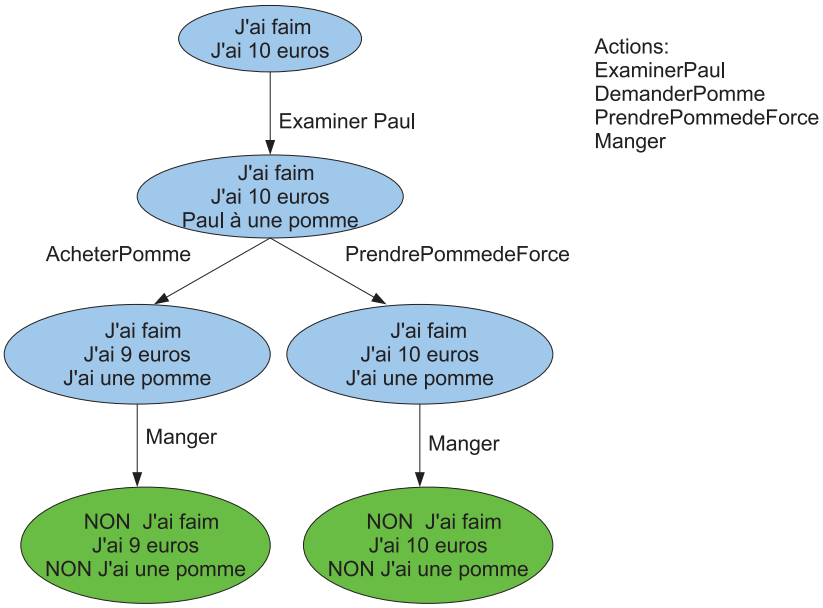


FIGURE 3.6 – Processus général d'exploration d'états

- la stratégie d'exploration des états possibles : l'exploration de l'arbre des possibilités peut être réalisée en largeur, en profondeur ou via une autre stratégie d'exploration mixte,
- la mesure de qualité du plan généré : une mesure de performance des solutions peut être déterminée ou non,
- le déterminisme du plan généré : un algorithme peut se fonder sur des mesures déterministes mais également probabilistes, ce qui implique des décisions stochastiques,
- la réutilisation des plans engendrés : un algorithme de planification peut utiliser les résultats des planifications précédentes lors de changement des actions réalisables (notamment les algorithmes à chaînage arrière).

Le principal problème de la planification par exploration d'états est qu'elle peut rapidement générer un grand nombre d'états, ce qui entraîne un coût mémoire et une complexité calculatoire très importants. C'est pourquoi la plupart des systèmes de planification sont spécialisés pour un problème, cela afin de restreindre le nombre d'états possibles et y appliquer des algorithmes spécifiques permettant d'éviter l'exploration de certaines branches. La planification de chemin est un exemple de problème très spécialisé.

3.3.2 La planification de chemin

Ce type de problème présente la particularité d'être défini dans le domaine continu qu'est l'espace. Cela se répercute sur la définition des états qui se doivent donc d'être caractérisés spatialement, mais également sur la spécialisation des algorithmes exploitant ces états.

3.3.2.1 Paramétrisation de l'environnement

Même si l'espace est continu, il va être nécessaire d'en donner une paramétrisation discrète afin d'y appliquer un algorithme de planification qui lui est discret. Deux approches sont possibles :

- partitionnement de l'espace en cellules, par exemple sous forme d'une grille régulière.
- partitionnement sous forme de graphe où les nœuds représentent des positions stratégiques, et les arcs les lignes droites praticables entre ces positions.

Partitionnement en cellules

Le partitionnement en cellules consiste à générer un recouvrement de l'espace par un ensemble de cellules dont les représentations géométriques sont en général convexes.

Grille régulière

Elle est obtenue par rastérisation des informations de l'environnement selon une grille régulière (voir figure 3.7). Le caractère régulier de la représentation a plusieurs avantages :

- adressage implicite des cellules,
- simplicité de manipulation, notamment lorsque l'on considère une implémentation sur GPU.

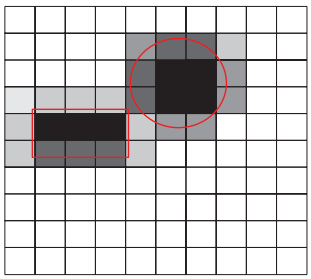


FIGURE 3.7 – Exemple de partitionnement de l'espace par grille régulière

Triangulation de Delaunay sous contraintes

Elle consiste à calculer une triangulation de Delaunay qui préserve les obstacles, c'est-à-dire l'espace non praticable. Les points et contraintes sont généralement issus d'une représentation géométrique de l'environnement. [Kallmann 2003] a étendu la triangulation de Delaunay à des environnements dynamiques en maintenant la cohérence de la triangulation lors de modifications des points. Cette technique n'est cependant pas optimale dans la mesure où elle n'est pas adaptée à la description d'environnements incluant des données de nature raster et donc non géométriques (fumée, feu, ...).

Subdivision récursive

Son principe consiste à subdiviser récursivement l'espace sur un critère donné. Les principales variantes sont :

- le Kd-Tree [Procopiuc 2003] , qui consiste à subdiviser par un seul plan judicieusement choisi à chaque itération de récursion (voir figure 3.8),
- le QuadTree, qui subdivise l'espace en deux sur chaque axe afin de créer quatre zones de même dimension à chaque niveau de subdivision.

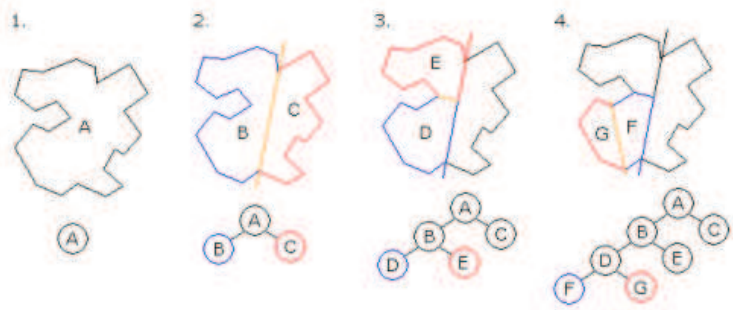


FIGURE 3.8 – Exemple de partitionnement de l'espace par Kd-tree

Le Quadtree est souvent préféré au Kd-Tree pour les applications de recherche de chemin car les dimensions spatiales sont traitées conjointement et la partition hiérarchique générée permet plusieurs niveaux d'abstraction des états spatiaux dans le processus de planification. De plus la maintenance d'un arbre Kd, dans le cas d'un environnement dynamique, est un traitement assez compliqué alors que l'insertion et la suppression dans un quadtree sont très simples.
Pour le cas du partitionnement d'un environnement 2D géométriquement défini contenant espace déambulable et obstacles, le critère peut être la présence d'un obstacle dans une cellule. L'avantage du Quadtree est double :

- organisation hiérarchique de l'environnement, ce qui permet une planification à plusieurs niveaux d'abstraction,
- mise à jour peu coûteuse lors des changements de l'environnement, ce qui permet son utilisation dans des environnements fortement dynamiques.

Partitionnement en graphe
Ce type de partition de l'espace vise à représenter tous les chemins possibles dans l'environnement. On appelle un tel graphe "carte des routes" ou Road Map.

Probabilistic Road Map (PRM)
Cette méthode consiste à échantillonner l'environnement de manière statistique et/ou stochastique afin de créer un ensemble de points pouvant être reliés par des segments qui n'intersectent aucun obstacle. De nombreux travaux ont été réalisés dans ce domaine afin d'optimiser les routes générées par cette technique [Geraerts 2002] car elle ne garantit pas :

- la couverture de l'ensemble des routes de l'environnement : certains passages peuvent ne pas être échantillonnés,
- la couverture minimale des routes possibles : certains échantillons peuvent être redondants.

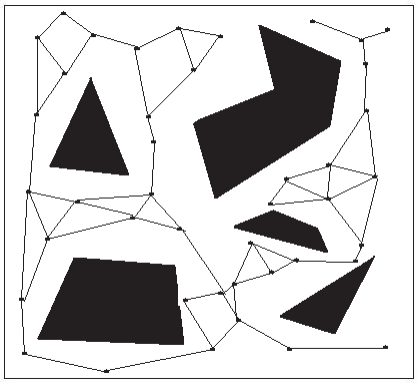


FIGURE 3.9 – Exemple de Probabilistic Road Map

Extraction de Road Map via un diagramme de Voronoï généralisé
On remplace les sommets d'un diagramme de Voronoï par les obstacles de l'environnement (non ponctuels) et on calcule le diagramme de Voronoï généralisé. La partition duale (équivalent du Delaunay d'un diagramme de Voronoï classique) est alors la roadmap. On peut notamment citer les travaux de [III 2000], qui permettent l'extraction temps-réel du diagramme de Voronoï en utilisant la puissance du GPU et par extension rend son utilisation possible pour des environnements géométriques fortement dynamiques au cours du temps.

3.3.2.2 Algorithmes de planification de chemin

L'environnement discrétisé, on peut alors définir un état de planification comme étant caractérisé par un nœud du graphe ou une cellule de l'environnement. L'utilisation d'algorithmes de planification devient alors possible. Ces algorithmes font partie des algorithmes d'optimisation. Afin d'éviter toutes explications liées au problème de cycle dans un graphe, et par suite garantir la convergence des algorithmes suivants, nous ne nous intéresserons ici qu'aux algorithmes à stratégie du "meilleur en premier". Dans ces derniers, on ne garde que la meilleure évaluation de coût pour chacun des états de l'environnement. Ils présentent alors la particularité d'avoir une occupation mémoire maximale limitée et de toujours converger vers une solution.
Ils seront illustrés à travers un partitionnement de l'environnement par grille régulière pour en simplifier l'explication.

Algorithmes de résolution du chemin dans un graphe
La particularité des algorithmes suivants est qu'ils ne prennent pas en compte la nature continue de l'espace. Ils cherchent donc à déterminer la suite d'états de planification S minimisant :
$\sum_{k=1}^{n} Cost(S_{k-1}, S_k)$ avec $S_1 = E_{init}$ et $S_n = E_{but}$
S_i : iéme cellule de la suite de cellules S
$Cost(x, y)$: coût de passage de x à y
n : nombre d'éléments de la suite S
E_{but} : état but
E_{init} : état initial

Algorithme de Dijkstra
C'est l'algorithme le plus connu de recherche du plus court chemin dans un graphe valué. Sa complexité est en $O(n)$. Il consiste à explorer l'ensemble des états en partant d'un état initial en marquant les états explorés par la somme g des valeurs de coût des arcs empruntés. Les états ainsi valués permettent d'élaborer une stratégie d'exploration en ne développant à chaque itération que l'état candidat de l'exploration le moins fortement valué : c'est la stratégie du meilleur en premier.
La convergence vers la solution optimale est garantie lorsque l'état but est atteint et que tous les autres états candidats portent un marquage g supérieur à celui de l'état but.
Il est à noter que pour retrouver le chemin optimum de l'exploration, il est nécessaire à chaque marquage effectif de mémoriser l'état père dont il est issu. Il est ainsi possible, à la fin de l'algorithme, de remonter du but jusqu'à l'état initial en suivant les états pères successifs afin de déterminer le chemin le plus court (retour sur trace). Ce processus est illustré par la figure 3.10. Dans cet exemple, les coûts de passage sont uniformes et la présence d'un obstacle inhibe toute règle de passage. Cependant, cet algorithme n'est pas très performant car il nécessite le parcours de l'ensemble des nœuds à chaque itération afin d'élire le meilleur.

Algorithme A^*
C'est l'algorithme d'optimisation du domaine de l'Intelligence Artificielle le plus utilisé. Il utilise une évaluation heuristique afin de guider l'exploration des états de résolution et ainsi converger plus rapidement vers une solution que Dijkstra mais sans garantir son optimalité. L'heuristique représente une estimation du coût minimum jusqu'à l'état but, elle est donc très dépendante du problème traité. Pour le problème du plus court chemin, les états étant localisés spatialement, une évaluation heuristique valide est la distance en ligne droite entre l'état courant et l'état but : $h(e) = distance(e, but)$.
La stratégie d'exploration est alors la même que celle utilisée par Dijkstra, à la différence que

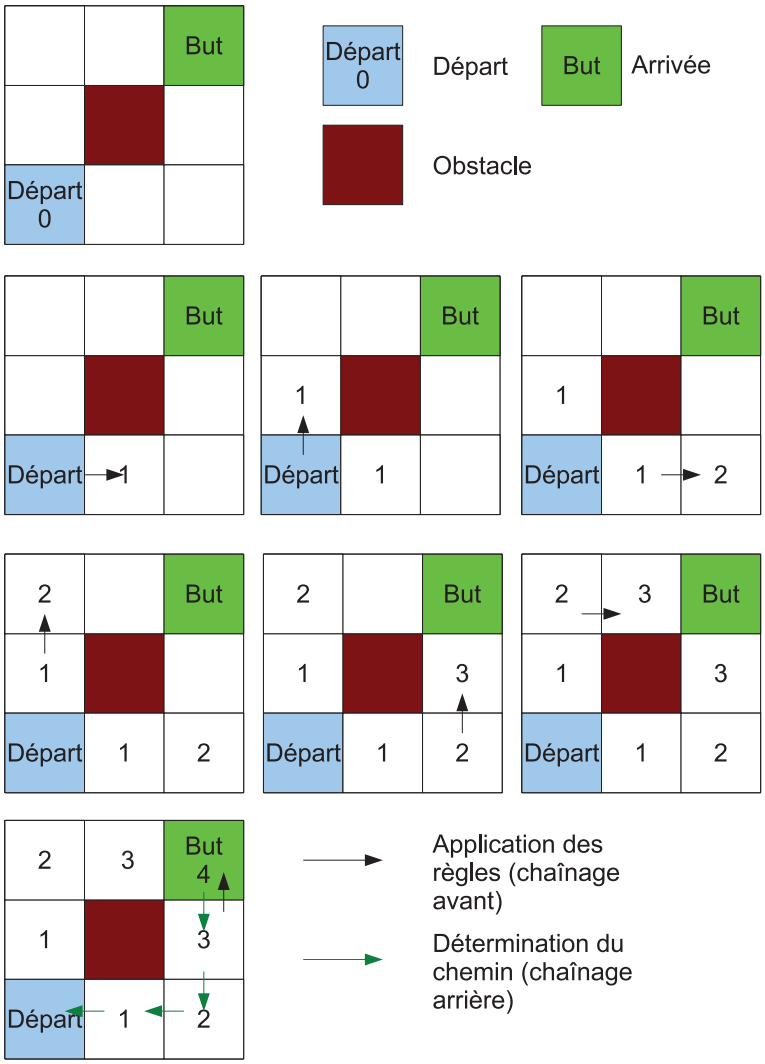


FIGURE 3.10 – Algorithme de Dijkstra.

l'évaluation du meilleur candidat à développer ne se réalise plus sur la valeur d'accumulation des coûts g mais sur la somme $f = g + h$. La convergence vers une solution optimale n'est alors plus garantie mais la plupart des applications du A^* se satisfont d'une solution suboptimale.
Cependant, le principal problème lié à l'évaluation heuristique est que sa détermination peut se révèler très complexe voire impossible.
Exemple : dans le problème du plus court chemin, si l'état but est caractérisé partiellement, il est préalablement nécessaire de déterminer sa localisation spatiale afin de rendre calculable l'heuristique. Et pire, si plusieurs états correspondent à cette caractérisation, l'heuristique devient alors : $h(e) = min(distance(e, but))\forall but \in Buts$.
Ceci accroît considérablement la complexité calculatoire de la planification.
Cet algorithme constitue la base des algorithmes modernes d'exploration de graphe et est de complexité $O(n \log n)$.
L'utilisation d'une heuristique concerne simplement l'utilisation d'un champ supplémentaire f qui sera utilisé comme clé d'ordonnancement de la liste ordonnée.

Algorithme LPA^*
Le LifeLong Planning A^* [Koenig 2002] est une variante de l'algorithme A^*, adapté à la planification de chemin en environnement dynamique. Il consiste à intégrer le concept de réparation locale de la planification effectuée à un instant t pour établir le chemin à l'instant $t+1$, en prenant en compte les changements de l'environnement entre ces deux instants.
Comme les algorithmes précédents, la planification s'effectue en chaînage avant (déductif). Mais, le principal avantage du LPA^* est qu'il permet de prendre en compte le changement dynamique des coûts des règles d'actions et de la localisation du but. Le principe de réparation s'explique aisément dans un arbre d'exploration des états simple (voir section 3.6). Lorsque les coûts de certains arcs changent, il suffit de remettre en cause les états ayant été générés à partir de ces arcs ainsi que tous les autres états déduits à partir de ce dernier. Cela permet alors de ne pas recommencer l'exploration depuis l'état initial.
Cependant, le LPA^* ne peut considérer le changement de position du mobile entre deux instants de la simulation, la planification doit être recommencée depuis le début si le mobile s'écarte du chemin déterminé.

Algorithme D^*Lite
Le D^*Lite [Koenig 2002] reprend le concept de réparation locale utilisé dans le LPA^* mais la planification s'effectue du but jusqu'au mobile, en chaînage arrière, on parle alors d'algorithme inductif. Comme conséquence, il ne peut réparer le changement dynamique du but mais intègre les changements de l'environnement ainsi que les déplacements du mobile. Enfin, il ne nécessite pas de remonter la chaîne des inductions successives (retour sur trace) afin de déterminer quelle direction doit prendre le mobile dans la mesure où la dernière induction effectuée constitue la première décision à suivre.
Le D^*Lite a également été prouvé aussi performant que sa formulation originale [Stentz 1994] D^*. Nous n'avons donc pas jugé utile de détailler cette dernière.

Algorithmes de résolution continue
Ces algorithmes prennent en compte dans la résolution la nature continue de l'environnement. Il en résulte un plus grand degré de liberté dans les décisions de trajectoires. Bien entendu, ils ne sont pas adaptés à une description de l'espace de type roadmap et requièrent un partitionnement de l'espace en cellules. Dans le domaine continu, le problème ne consiste plus en la recherche d'une suite de cellules mais d'une trajectoire optimale. Dans la mesure où la fonction de coût peut être considérée comme continue, on peut exprimer ce problème comme étant la recherche du chemin p minimisant :

$$\int_0^L Cost(p(s, x_{but}))ds \ avec \ p(0, x_{but}) = x_{init} \ et \ p(L, x_{but}) = x_{but}$$

L : longueur du chemin
$Cost(x)$: fonction de coût
x_{but} : position du but
x_{init} : position initiale
$p(s, x_{but})$: position sur le chemin à une longueur s de x_{init}.
Le problème de trajectoire optimale jusqu'à un but dans le domaine continu peut se formuler ainsi : connaissant la fonction $Cost$ symbolisant les coûts de déplacement locaux (règles valuées), construire le champ Φ correspondant au coût total C (cumul de coûts portés par les règles) en tout point de l'environnement pour se rendre à un but fixé tel que $\Phi_{but} = 0$ et $\nabla\Phi = C$.

Il suffit alors pour un mobile de suivre la direction inverse du gradient du champ Φ ($-\nabla\Phi$) pour se rendre de sa position initiale jusqu'à sa position finale.

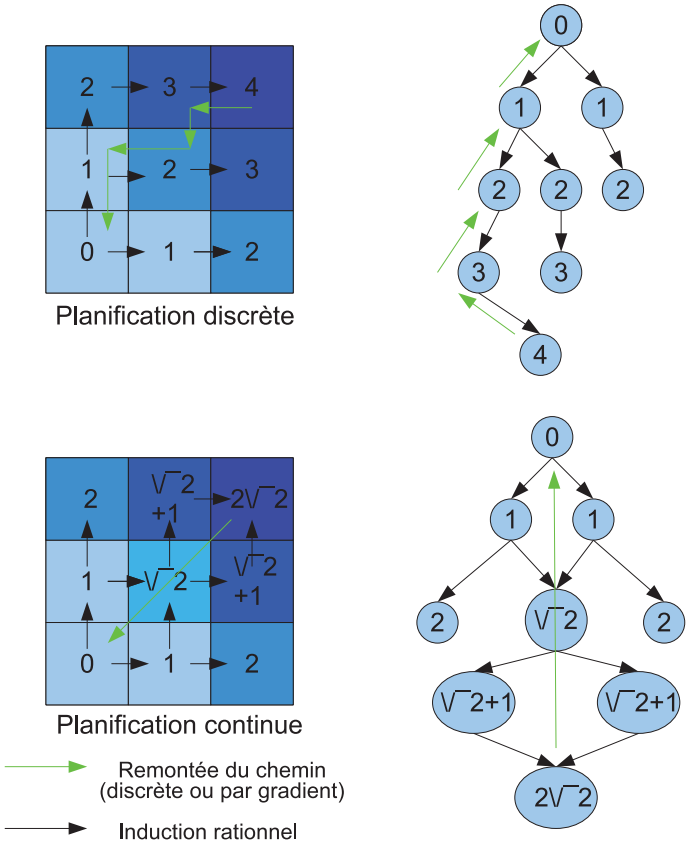


FIGURE 3.11 – Comparaison de la planification discrète et continue dans une grille régulière

L'équation eikonale $\nabla\Phi = C$ est une équation de type Hamilton-Jacobi dont la résolution s'effectue communément à l'aide des méthodes de résolution issues de la théorie des level sets [Sethian 2003].

Elles sont toutes fondées sur le concept de noyau d'interpolation permettant de déterminer une approximation bidirectionnelle du champ Φ à partir des potentiels voisins les plus faibles et de la fonction C.
Par exemple, pour une structuration en grille régulière de l'environnement le noyau d'interpolation est donné par l'équation (3.5).

$$\begin{cases} min_x = min(\Phi_{(i-1,j)} + Cost(cell_{(i,j)}, cell_{(i-1,j)}), \Phi_{(i+1,j)} + Cost(cell_{(i,j)}, cell_{(i+1,j)})) \\ min_y = min(\Phi_{(i,j-1)} + Cost(cell_{(i,j)}, cell_{(i,j-1)}), \Phi_{(i,j+1)} + Cost(cell_{(i,j)}, cell_{(i,j+1)})) \\ \left(\dfrac{(\Phi_{(i,j)} - min_x)}{Cost(cell_{(i,j)}, min_x)}\right)^2 + \left(\dfrac{(\Phi_{(i,j)} - min_y)}{Cost(cell_{(i,j)}, min_y)}\right)^2 = 1 \end{cases} \tag{3.5}$$

i, j : coordonnées de la cellule dans la grille régulière
$Cost(cell_1, cell_2)$: coût de passage de la cellule 1 à la cellule 2

Afin de s'abstraire de la relative complexité de l'équation (3.5) nous écrirons formellement sa résolution par une fonction
$\Phi_{cur} = NoyauInterpolation(\Phi_{minx}, \Phi_{miny}, Cost(cur, miny), Cost(cur, miny))$.

La principale différence conceptuelle avec un algorithme de planification discret vient alors de la possibilité pour un état d'être issu de deux états au lieu d'un seul (voir section 3.11) car les règles de coûts peuvent être spatialement interpolées.

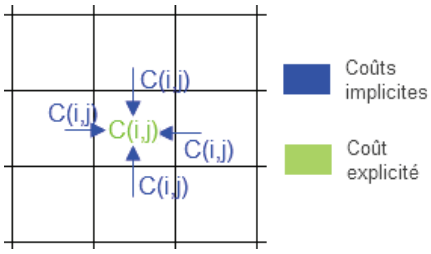


FIGURE 3.12 – Illustration du coût de passage isotropique

Nous utiliserons par la suite la grille régulière ainsi qu'une modélisation isotropique du champ de coût $Cost$, qui imlique que le coût pour passer de i à j est identique à celui pour passer de j à i. Les potentiels de coût sont alors stockés au centre des cellules et non pas entre les cellules (voir figure 3.12). Le noyau d'interpolation (3.5) peut alors se réécrire par l'équation (3.6) :

$$\begin{cases} min_x = min(\Phi_{(i-1,j)}, \Phi_{(i+1,j)}) \\ min_y = min(\Phi_{(i,j-1)}, \Phi_{(i,j+1)}) \\ \dfrac{(\Phi_{(i,j)} - min_x)^2 + (\Phi_{(i,j)} - min_y)^2}{{Cost_{(i,j)}}^2} = 1 \end{cases} \tag{3.6}$$

Les différents travaux utilisant ces méthodes de résolution vont maintenant être présentés.

Fast Marching Method (FMM)

Plutôt que de décrire la théorie des levels sets afin d'expliciter la FMM, cette dernière est décrite comme une planification discrète où la fonction de coût est interpolée de façon continue.
En effet, en reprenant l'algorithme A*, avec une heuristique nulle, appliqué en chaînage arrière (à partir du but), dans un graphe de l'environnement dont les nœuds sont les cellules d'une grille régulière et les transitions les relations de voisinages de ces cellules, on peut générer le processus illustré par la deuxième illustration de la figure 3.11. Une particularité de la planification continue est que dans certains cas une cellule de la grille peut avoir deux cellules parentes, au lieu d'une seule dans l'algorithme de Dijkstra. Un noyau d'interpolation calcule alors une combinaison linéaire des valeurs portées par les deux cellules. Cette méthode est notamment utilisée dans [Treuille 2006] afin de planifier d'une seule traite toutes les trajectoires d'individus d'une foule partageant la même fonction de coût $Cost$ ainsi que les mêmes buts.

Le méthodes de planification continues ont l'avantage, par rapport aux versions discrètes, de générer des trajectoires plus lisses, mais ceci s'obtient au prix d'une complexité calculatoire supérieure, du fait des nombreuses interpolations nécessaires.

Algorithme E^*

Le E^* [Philippsen 2007] généralise l'utilisation de noyau d'interpolation au D^*. Il intègre donc la notion de réparation locale dans la FMM et permet ainsi la planification en environnement continu et dynamique.

Fast Iterative Method (FIM)
Il s'agit d'une adaptation de la FMM pour les architectures vectorielles [Jeong 2007]. Le concept de base de cette méthode est de développer toutes les branches de l'exploration en parallèle. Cette manière de procéder entraîne bien entendu un rapide dépassement du nombre de ressources limitées de traitement parallèle mais reste cependant bien plus efficace en temps de calcul qu'une FMM. Son implémentation dépend bien entendu des moyens de communication entre ressources de traitement. L'auteur décrit une implémentation CUDA pour de gros volumes de données (grille 1024^3) ne pouvant être hébergés entièrement en mémoire graphique. Cependant le problème de planification de chemin en 2D ne nécessite pas autant de mémoire et ne justifie pas la politique de chargement mise en place par l'auteur. De plus, nous verrons par la suite que l'utilisation d'une abstraction de programmation GPU telle que CUDA n'est pas nécessaire vu le caractère fortement spatial de l'algorithme. L'algorithme de la FIM est donné au chapitre 7.

3.3.2.3 Conclusion

La planification de chemin en environnement continu nécessite donc un partitionnement de l'espace déambulable de l'environnement. Ce partitionnement se doit d'être un compromis entre la complexité de la planification et sa précision : plus les cellules sont grandes plus l'exploration est rapide mais plus la trajectoire engendrée sera grossière. Les actions possibles prennent alors la forme de règles discrètes de passage d'une cellule vers chacune de ces voisines, et peuvent aisément être valuées par une fonction de coût de passage d'une cellule à l'autre. Ce coût de passage étant unidimensionnel, il est généralement implémenté sous la forme d'une fonction de la distance entre les cellules et des diverses variables d'états (exemple : mur, autres individus, vitesse, changement de trajectoire, ...) portées par chaque cellule. Un noyau d'interpolation se charge d'interpoler un coût continu à partir des règles discrètes dans les différentes dimensions spatiales du problème.

Il est à noter dans les algorithmes de résolution continus, la fonction de coût doit également être continue dans les différentes dimensions spatiales. La planification discrète n'étant pas sujet à cette contrainte, elle permet une abstraction du problème bien plus aisément manipulable malgré le peu de degré de liberté des règles qu'elle autorise. Par exemple, les algorithmes décrits ici étant à chaînage strictement unidirectionnel, une contrainte de vitesse minimale pour sauter par dessus un obstacle ne peut être prise en compte : la planification échouera si le meilleur état devant l'obstacle ne porte pas une variable de vitesse suffisante à l'application de l'action "Sauter" car dans ce cas le potentiel de la cellule ne sera jamais révisé à une valeur plus élevée. Dans la planification discrète, cette contrainte peut être propagée aux états pères afin que ces derniers révisent la règle appliquée afin de maximiser cette vitesse. Cependant, aucun des travaux recensés n'a envisagé l'intégration du chaînage arrière dans les algorithmes de planification continus.

3.4 Quelques architectures d'agents hybrides cognitive/réactive recensées

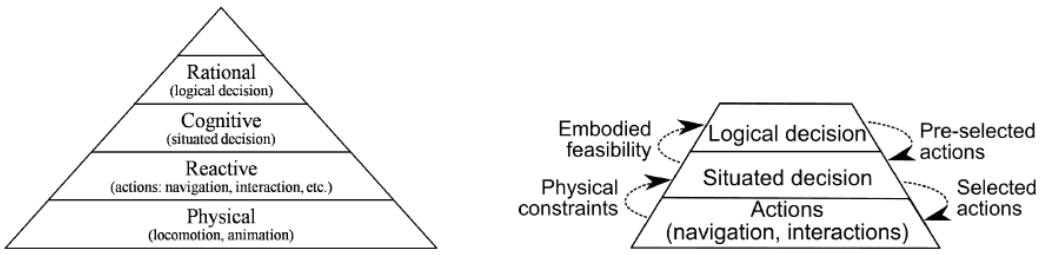
Les architectures hybrides sont massivement utilisées dans les systèmes, autant pour des questions de réactivité du système que d'adaptation des comportements. L'architecture de simulation en environnement continu utilisée dans Simulem [Donikian 2008] ne sera décrite que brièvement. En effet, nos investigations se sont plutôt orientées vers les architectures hybrides réactive/cognitive du domaine des Systèmes Multi-Agents. Ce sont des architectures en couches afin d'intégrer les différents types de

comportement d'agents. Les deux architectures suivantes représentent les deux typologies principales recensées dans la littérature en fonction des flux de contrôle entre les différentes couches :

1. une architecture verticale hiérarchisée : InteRRaP. Ce type d'architecture est contrôlé de manière séquentielle. Seule la couche inférieure est connectée aux systèmes de perception (capteurs) et d'action (effecteurs). Les informations issues du système de perception transitent à travers toutes les couches logicielles jusqu'à la couche la plus haute qui contrôle le processus délibératif de sélection d'action. Les traitements adéquats sont ensuite déclenchés en séquence au niveau de chaque couche lors d'une phase descendante jusqu'à la couche inférieure ;
2. une architecture horizontale : Touring Machine. Dans ce type d'architecture, toutes les couches logicielles sont connectées au système de perception (capteurs) et d'actions (effecteurs). Les traitements sont effectués de manière concurrente. Cela nécessite souvent un module de contrôle centralisé qui assure la consistance de la solution et le processus délibératif de sélection d'action.

3.4.1 Simulem

L'architecture Simulem est une architecture de mouvement microscopique d'individu mis en avant dans des scénarios d'utilisation de la gare de Lyon.
La figure 3.4.1 présente cette architecture à couches à passe verticale ainsi que les différents flux d'informations entre les couches.



(a) Overview of the decision mechanisms of an agent.

(b) Communication between the three upper layers of the pyramid.

FIGURE 3.13 – Vue générale de Simulem

Elle est dérivée de la pyramide de modélisation comportementale de A. Newell, en introduisant le découplage entre décision située et décision logique. En voici une explication succincte :

1. les percepts de l'environnement embarquent une sémantique d'utilisation des objets de l'environnement. Ce modèle nommée BIIO basé sur le concept d'affordance permet à chaque individu de trouver un scénario d'utilisation de ces derniers,
2. une fois le scénario d'utilisation déterminé par l'agent, la couche logique crée une séquence de but à satisfaire pour réaliser ce scénario. Ces sous buts sont transmis un-à-un à la couche sous jacente.
3. la couche située se chargera alors de déterminer le mouvement à entreprendre par l'agent pour tendre vers son but géographique (exemple : distributeur) et ensuite effectuer l'action qui lui est associée (exemple : interaction avec le distributeur).

3.4.2 L'architecture InteRRaP

InteRRaP [P. Müller 1993] est une architecture d'agents hybride utilisant le concept de couches verticales (voir figure 3.14). On peut dénoter deux types de couches différentes [Ferber 1997] :

- les couches de gauche sont de type double passe (ascendante puis descendante), et servent de structure de contrôle dans cette architecture (Control Unit),
- les couches de droite sont de type simple passe, et représente différentes bases de connaissances (Agent Knowledge Base).

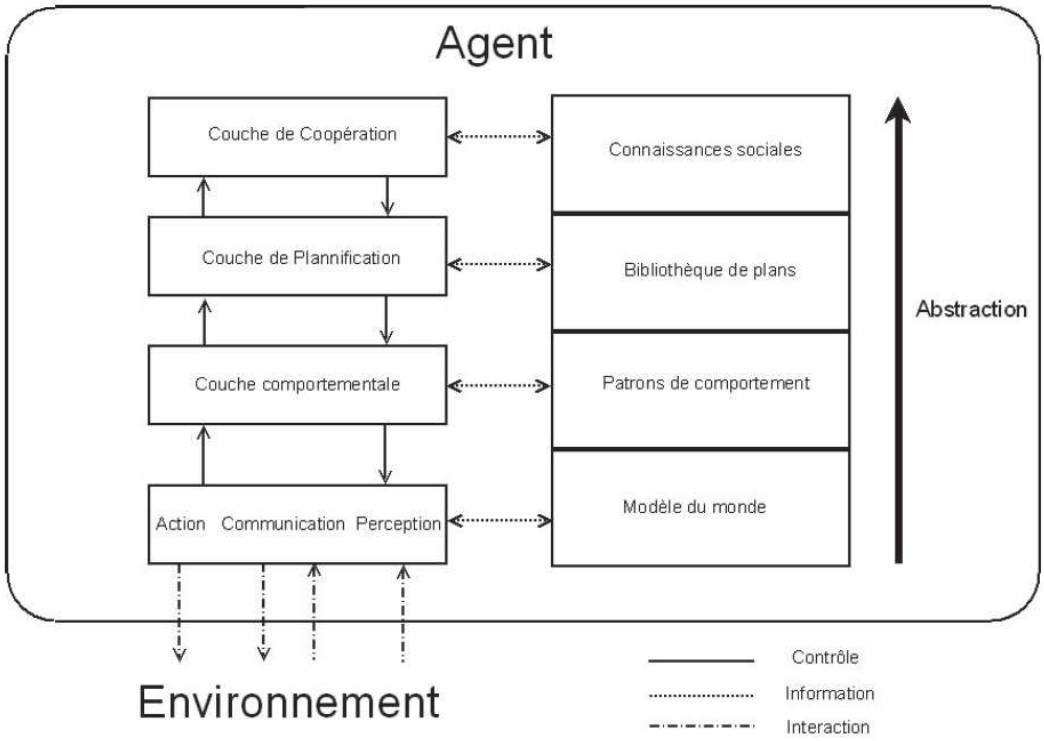


FIGURE 3.14 – Architecture InteRRap

3.4.2.1 L'unité de contrôle

C'est ce module qui contrôle ce qui sera effectué par l'agent. La première couche est l'interface avec le monde, avec sa partie capteurs, effecteurs, et outils de communications. Cette couche est en quelque sorte la couche physique d'un agent. Elle est associée à la partie modèle du monde de la base de connaissance, qui stocke les connaissances de l'agent sur le monde (ou les croyances sur le monde pour des agents BDI).
L'interface avec le monde passe les percepts reçus à la fois au modèle du monde et à la couche comportementale juste en dessus. La couche comportementale est en fait une couche réactive traitant les évènements perçus par l'interface avec le monde, que ce soit par la perception directe du monde, ou bien par réception de messages provenant d'autres agents.
Cette couche a accès à des patrons de comportements stockés dans la base de connaissance. Le comportement sélectionné répond aux stimuli reçus de l'environnement, mais ce comportement peut être de plusieurs natures. En effet, en cas de réponse à un stimulus simple, comme la perception d'un obstacle, le comportement plus ou moins primaire sélectionné sera directement transmis à l'interface avec le monde pour exécution. En cas de problème plus complexe, l'action effectuée par le comportement

peut-être un appel à une couche de plus haut niveau dans l'unité de contrôle, afin de créer un plan.
La couche de planification et celle de coopération ont à peu près la même fonction, c'est à dire à répondre au problème par la création d'un plan. Un plan est un redécoupage d'un problème pour lequel l'agent n'a aucun moyen de réponse, un but qui ne peut pas être atteint par l'utilisation simple d'un comportement primitif. Ce redécoupage de but à atteindre se fait bien évidemment en plusieurs autres buts, jusqu'à obtenir une succession de buts réalisables par l'agent (c'est à dire par une succession de buts pouvant être atteints par des primitives).
La seule différence (toutefois majeure) entre la couche de planification et celle de coopération est que les plans effectués par la couche de planification sont réalisables en local (l'agent pour faire toutes les sous-tâches tout seul), alors que la couche de coopération intervient lorsqu'une tâche ne peut être réalisée par un seul agent, et que des communications inter-agents sont nécessaires afin de coordonner des actions. Pour cela, les connaissances de l'agent sur les manières de communiquer avec d'autres agents sont stockées dans la couche de connaissances sociales de la base de connaissances.

3.4.2.2 La base de connaissance

Cette base comme vu précédemment est organisée en 4 couches, et contient toutes les données nécessaires au bon fonctionnement de l'agent. La première couche contient tout ce que sait l'agent sur le monde dans lequel il évolue, cela va des agents qu'il a rencontré, aux objets de l'environnement, la topographie...
La seconde couche, la couche de patrons de comportement contient une liste de comportements primitifs. Ensuite vient la bibliothèque de plans contenant des squelettes de plans pour faciliter l'établissement de ceux-ci par les couches faisant de la planification, qu'elle soit locale ou distribuée. Et enfin la couche de connaissances sociales contenant des protocoles de communication, de négociation, et des plans pour accomplir des tâches avec plusieurs agents à la fois.

3.4.3 TouringMachines

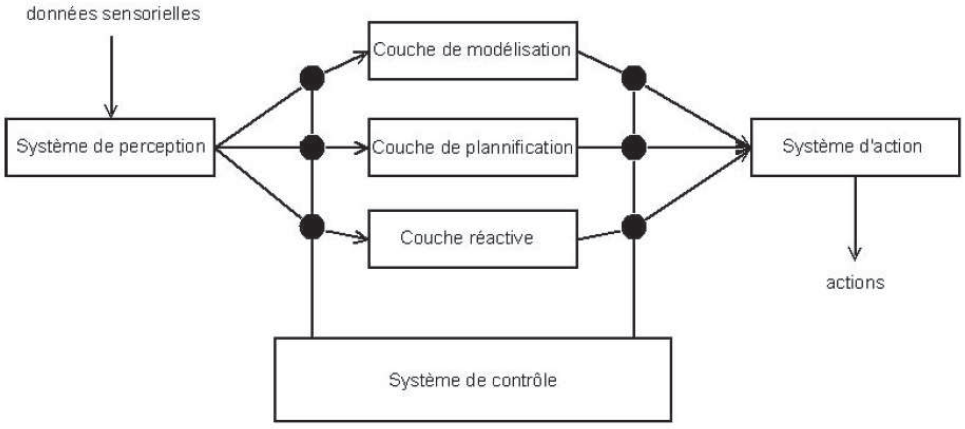


FIGURE 3.15 – Architecture de la Touring Machine

C'est un modèle hybride en couches également, ces dernières n'ont pas de rapport hiérarchique entre elles. Il y a trois couches permettant de suggérer une action ou une série d'actions, et une quatrième couche, appelée ici le système de contrôle, qui permet de sélectionner parmi ces propositions la plus judicieuse.

La première de ces trois couches propositionnelles est une couche réactive, qui contient une politique de sélection réactive (exemple : subsomption), et qui à partir des stimuli en entrée va sélectionner la réponse préprogrammée la plus adéquate.
La seconde couche est une couche de planification. Cette couche est composée de deux organes, l'un étant vraiment le planificateur, l'autre effectuant un filtre des données en entrées, ce qui permet d'éliminer des informations non nécessaires à la planification, ou non désirées.
La troisième couche est la couche de modélisation, et contient l'état de tous les objets de l'environnement, agents y compris. Celle-ci essaye d'éviter les conflits qui pourraient survenir en demandant à la couche inférieure de planification d'établir de nouveaux plans à partir de buts édités pour l'occasion, pour l'évitement de ces conflits par la couche de modélisation.
Une fois que chacune de ces trois couches ont proposé une réaction possible aux données reçues par le système de perception, le système de contrôle choisit l'action la plus adaptée. Ce système fonctionne à partir de règles de contrôle, et peut donc choisir les informations pour le système d'action (donc des actions), mais peut également filtrer les données entre le système de perception et la structure de contrôle. Le système de contrôle vérifie donc les données en entrée et en sortie de la structure de contrôle.

3.4.4 Remarques

Dans les travaux étudiés, le paradigme de l'action dans les systèmes multi agent est totalement discret, ce qui rend frustrant l'utilisation des architectures citées pour l'action continue qu'est le déplacement. En effet, ce paradigme discret impose des décisions dont le degré de liberté est très faible et de plus il n'autorise pas le mélange d'actions non orthogonales.
Face à cet état de fait, le développement d'une architecture de simulation fondée sur un paradigme d'actions continues nous est apparu nécessaire.

3.5 Prise en compte des interactions physiques

3.5.1 Introduction

La plupart des simulations ne considère pas les individus virtuels comme étant des corps dynamiques mais comme des corps cinématiques (pas de modèle physique). Cependant cette partie de modélisation dans le contexte de l'évacuation de bâtiment en feu est très importante car les forces physiques sont la principale cause de blessures lors des incendies [Fruin 1993]. Modéliser les interactions physiques est donc nécessaire afin d'évaluer les forces subies par les individus virtuels au court du temps.

3.5.2 Généralités

Les interactions physiques dans le cadre de la simulation d'évacuation en feu reposent principalement sur la détection et la réponse aux collisions des évacuants avec leurs congénères ainsi qu'avec les murs. En informatique, cette problématique est connue sous l'appellation "Dynamique de corps rigides". La vitesse et l'accélération s'expriment grâce aux formules suivantes :

$$\begin{cases} \dot{v}(t) = m\sum\overrightarrow{F(t)} = m(\overrightarrow{F_{ext}(t)} + \overrightarrow{F_{int}(t)}) \\ \dot{x}(t) = v(t) = \frac{dx}{dt} \approx \frac{x2-x1}{dt} \end{cases}$$

$\dot{v}(t)$: accélération (dérivée de la vitesse)
$F_{ext}(t)$: forces physiques extérieures à l'individu
$F_{int}(t)$: forces physiques que l'individu génère pour se déplacer
$\dot{x(t)}$: vitesse (dérivée du vecteur position)
x_i : position à l'instant i

La dynamique des corps non ponctuels requiert la prise en compte du mouvement rotationnel, exprimé par les formules suivantes : $\begin{cases} \dot{R}(t) = \omega^*(t)R(t) \approx \frac{R2-R1}{dt} \\ L(t) = I(t)\omega(t) \\ \dot{L}(t) = \tau(t) \end{cases}$

$\dot{R}(t)$: dérivée de la rotation à l'instant t
ω : vitesse angulaire (*≡ forme matricielle)
$L(t)$: moment rotationnel
$\dot{L}(t)$: dérivée du moment rotationnel
$I(t)$: tenseur d'inertie
$\tau(t)$: torque global

Dans un environnement où n corps dynamiques évoluent, le problème consiste donc à déterminer, à chaque pas de temps de la simulation, les nouvelles positions et orientations des corps en mouvement. Puis il faut détecter les collisions, ce qui aboutit à une complexité en $O(n^2)$, n étant le nombre de corps. Le traitement d'un seul pas de temps de simulation peut ainsi être très coûteux et de plus le traitement d'une collision (calcul des nouvelles trajectoires après collision) peut engendrer d'autres collisions. Pour cette raison la plupart des moteurs physiques utilisent une intégration adaptative du temps entre deux instants de simulation.

3.5.3 Détection des collisions

Afin de déterminer les corps en interaction, une stratégie de partitionnement de l'espace favorisant les requêtes de voisinage peut être utilisée. Elle doit être peu coûteuse à maintenir au cours de la simulation. Elle s'effectue donc en deux phases :

- BroadPhase : détection grossière des collisions (filtrage),
- NarrowPhase : détection précise des collisions.

A cet effet, plusieurs techniques peuvent être utilisées.

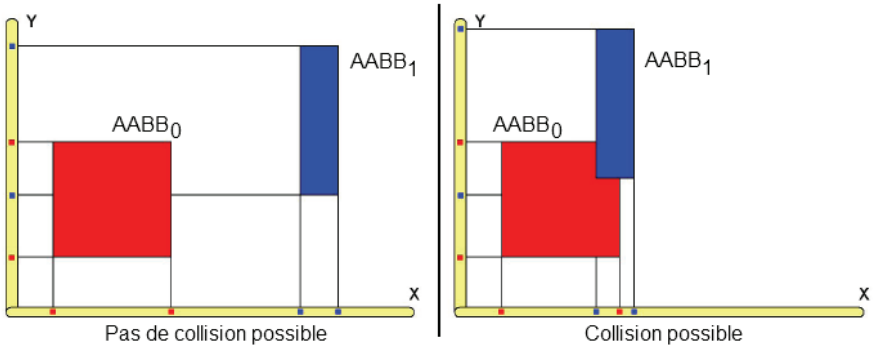


FIGURE 3.16 – Principe du Sweep and Prune

3.5.3.1 Sweep and Prune (SAP) et Axis Aligned Bounding box (AABB)

La combinaison de ces deux techniques consiste à maintenir une boîte englobante alignée sur chaque axe du repère (AABB) pour chacun des objets (voir figure 3.16). La détection des collisions est alors effectuée de façon indépendante sur chacun des axes. Ainsi, $\frac{(n^2-n)}{2}$ paires doivent être testées [Cohen 1995]. Au vu du test, les paires restantes représentent les objets considérés comme potentiellement en collision. L'algorithme 1 représente l'implémentation non optimisée.

Algorithme 1 Implémentation non optimisée des techniques SAP et AABB.

```
Pour chaque Objet o1 Faire
  Pour chaque Objet o2 Faire
    Pour chaque Axe a Faire
      Si o1.AABB.LimitesSurAxe(a).intersect(o2.AABB.LimitesSurAxe(a)) Alors
        collision entre les volumes englobants de o1 et de o2
        (faire un détection plus précise par GJK)
      Sinon
        goto next object o2
      Fin Si
    Fin Pour
  Fin Pour
Fin Pour
```

L'optimisation du Sweep and Prune consiste à maintenir les boites englobantes des objets triés sur chacun des axes. Ainsi, le test de collision de l'ensemble des objets (complexité en $n(n-1)$) peut être réalisé localement sur chacun des axes et devient donc beaucoup moins coûteux. Des travaux ont également montré la possibilité de parallèliser ces traitements sur GPU [Nguyen 2007b] grâce notamment à un hashage réparti des paires de collisions possibles.

3.5.3.2 L'algorithme de Gilbert Johnson Keerthi (GJK)

Une fois les collisions potentielles détectées à l'aide des volumes englobants, il est alors nécessaire de déterminer si oui ou non elles sont effectives sur les objets complexes. L'algorithme GJK, illustré par la figure 3.17, peut être utilisé à cet effet. Il permet de déterminer rapidement si deux objets convexes complexes A et B se recouvrent en se basant sur la différence de Minkowski (A-B). Se référer à [Van den Bergen 1999] pour plus d'informations sur l'algorithme GJK.

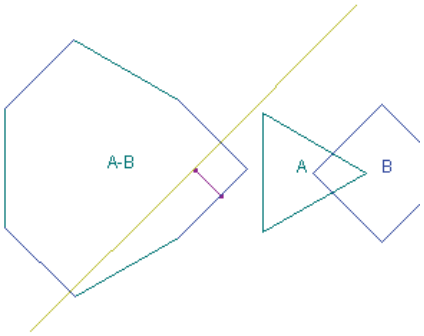


FIGURE 3.17 – Illustration de l'algorithme GJK et de la différence de Minkowski

3.5.3.3 Grille régulière sur GPU

La simulation des corps rigides sur GPU a notamment été étudié par T. Harada dans [Nguyen 2007a]. Dans ses travaux, il utilise une grille régulière afin de stocker les objets. Il bénéficie alors d'une structure indexée pouvant facilement être mise à jour et où la notion de voisinage spatial est implicite. Il n'utilise pas l'algorithme GJK dans la mesure où tous les objets sont rastérisés et ainsi approximés par un ensemble de points. Avec cette méthode simple, il simule des milliers, voire des millions (sur 4 GPUs) d'objets. Cependant cette méthode n'est pas adaptée pour des objets de tailles différentes.

3.5.4 Le traitement des collisions

Il consiste en une correction du positionnement des objets en leur appliquant la force que l'autre partie lui transmet. Pour des objets relativement mous, on peut également lui appliquer un amortissement.

3.5.5 Conclusion

Grâce à une minimisation des interactions possibles et une détection fine des collisions, les individus virtuels peuvent alors bénéficier d'une modélisation physique permettant de prendre en compte les diverses interactions physiques qu'ils subissent au cours de l'évacuation.

Le chapitre suivant s'attache à décrire les techniques informatiques permettant de doter les individus d'un moyen d'interaction avec le monde physique dans lequel ils évoluent.

3.6 Visualisation de foule

Dans un simulateur d'évacuation de bâtiment en feu, le système de visualisation a deux objectifs :

1. Visualiser un grand nombre d'individus virtuels animés,
2. Visualiser des données volumiques dynamiques telles que le feu et la fumée conjointement à des éléments géométriques tels que les murs et personnages évacuants au sein d'une scène.

La visualisation des résultats de simulation dans le contexte de l'évacuation de bâtiment en feu doit permettre la navigation temps réel de l'utilisateur dans l'environnement 3D simulé. Les techniques de rendu utilisées doivent donc toutes deux être compatibles avec une utilisation temps réel.

3.6.1 Rendu temps réel de foule

Le rendu de foule a été largement étudié par Thallman, notamment dans son très riche tutoriel sur le sujet ([Thalmann 2008]). Ce qui suit en est principalement extrait et nous encourageons le lecteur à s'y référer pour plus de détails sur le sujet.

3.6.1.1 Modèle d'individu virtuel

Un individu virtuel est généralement modélisé informatiquement par un squelette dont les "os" permettent l'animation. Un apparence géométrique est ensuite mappée sur ces derniers (voir figure 3.18).

Ce modèle couramment utilisé permet l'animation temps réel des personnages virtuels en utilisant la technique d'animation par key-frame. Cette dernière consiste à décrire une séquence temporelle

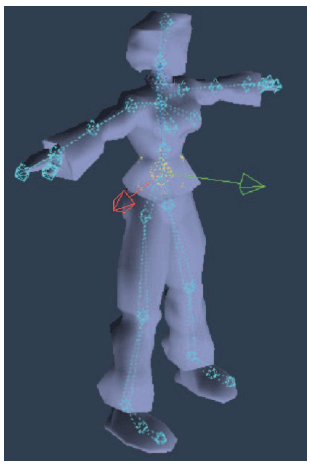


FIGURE 3.18 – Squelette d'un individu et son apparence géométrique

minimale de configurations des articulations entre les os du squelette de l'individu. Ces configurations caractéristiques d'une animation sont un ensemble d'angles formés par chaque articulation et sont implémentées par des quats, ce qui permet ainsi d'interpoler les configurations non définies (voir figure 3.19).

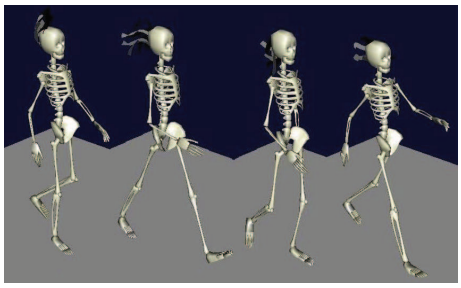


FIGURE 3.19 – Animation générée par interpolation des key frames

La visualisation en temps réel d'un grand nombre d'individus requiert d'importantes ressources de calcul d'affichage afin de maintenir un niveau de réalisme satisfaisant. Plusieurs stratégies peuvent être combinées afin de limiter l'usage de ces ressources.

3.6.1.2 Stratégies de frustum culling

Ce type de stratégie consiste à ne pas afficher les individus ne se trouvant pas dans le champ de vision de l'observateur. Le frustum culling repose sur une structuration de l'espace généralement organisée hiérarchiquement par regroupement des modèles géométriques se trouvant dans un même volume d'espace. Ainsi, afin de déterminer si un modèle doit être affiché et animé, il suffit de tester si les volumes de regroupement de tous ses pères ont une intersection avec le cône de vision. Le problème avec les personnages virtuels est que les regroupements d'individus évoluent au cours du temps. Ils nécessitent alors une stratégie de hiérarchisation spatiale adaptative et donc une structure de données associée facilement maintenable au cours du temps. Pour ce faire, une grille régulière ou encore un octree peuvent être utilisés.

3.6.1.3 Stratégies de niveau de détails

Les stratégies de niveaux de détails consistent à dégrader l'apparence et/ou l'animation des personnages se situant à des distances critiques de l'observateur. La dégradation des modèles géométriques peut-être réalisée par suppression des sommets ayant peu d'impact sur le modèle global ou encore par fusion de deux sommets très proches [Cohen 1998]. Mais pour des distances significatives, la plupart des travaux privilégient une approche par génération de billboards et/ou d'imposteurs [Dobbyn 2005].

3.6.1.4 Choix technique d'instanciation de géométries

L'instanciation de rendu des modèles géométriques est une fonctionnalité matérielle permettant l'affichage efficace de nombreuses instances du même modèle géométrique par une seule instruction. Cependant contrairement à l'utilisation des listes d'affichage compilées, l'instancing permet de spécifier des propriétés intrinsèques à chaque instance. Par exemple, des personnages virtuels différents peuvent partager la même apparence et les mêmes animations mais n'ont pas la même posture au même instant. La technique d'instanciation à choisir dépend fortement de l'API et du matériel graphique utilisés et par extension de la portabilité souhaitée de l'application développée. Par exemple, OpenGL propose deux implémentations :

- Le pseudo instancing, technique supportée par les cartes graphiques OpenGL 2.0,
- L'instancing réel, depuis l'avènement d'OpenGL 3.0 et des architectures de shaders unifiés.

3.6.2 Rendu volumique de feu et de fumée

Le rendu de fumée et de feu fait partie du domaine de la visualisation volumique de milieux participatifs. Cette discipline consiste à simuler les transferts radiatifs lumineux dans un milieu interagissant avec la lumière. Elle nécessite de simuler plusieurs phénomènes physiques d'interaction entre la lumière et le milieu dans lequel elle se propage. Pour rester simple, quatre phénomènes se doivent d'être simulés : l'émission (pour le feu), l'absorption et la diffusion entrante et sortante (voir figure 3.20).

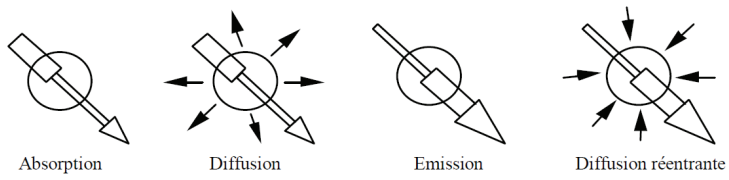


FIGURE 3.20 – Les 4 interactions lumineuses en milieu participatif

Dans notre cas, les conditions d'éclairage des maquettes de bâtiment utilisées n'ont pas été spécifiées, ce qui coupe court à la possibilité d'utiliser ces techniques évoluées pour notre simulation.

Nous nous contenterons donc de décrire les techniques de rendu volumique en l'absence des conditions d'éclairage pour un volume décrit par une grille 3D régulière portant des valeurs de concentration locale.

3.6.2.1 Rendu par lancer de rayons

Cette technique est une approche basée image du rendu volumique, elle consiste à déterminer la couleur de chacun des pixels composant l'image finale en simulant le parcours inverse de la lumière de la scène vers l'œil (voir figure 3.21 à droite). Elle est donc particulièrement adaptée lorsqu'aucune condition d'éclairage n'est connue et suit donc le modèle d'intégration optique classique (équation (3.7)) :

$$\alpha_{obs} = \int_0^L \alpha(s)ds \tag{3.7}$$

α_{obs} : opacité cumulée le long du rayon.
L : longueur du rayon.
$\alpha(s)$: opacité locale au point s du rayon.

En pratique les opacités sont données par une fonction de transfert transformant la concentration locale C du volume en une opacité et une couleur : $TF : \Re+ \implies \Re^4+$
Les opacités locales $\alpha_{loc} = TF(C)_a\Delta(s)$ sont alors sommées à intervalles réguliers le long de chaque rayon issu de chaque pixel de l'image finale. Les autre canaux de couleur de la fonction de transfert quand à eux sont sommés au prorata de l'opacité locale ($Col_{loc} = \alpha_{loc}TF(C)_{rgb}$) De nombreuses techniques ont été développées afin de diminuer la complexité de cette intégration. [Engel 2001], notamment, utilise un prétraitement afin de rendre linéaire la résolution de cette intégrale pour un volume statique, permettant ainsi un rendu instantané. Cependant, cette formulation du rendu volumique n'est pas compatible avec un rendu hybride géométrique/volumique lorsque des objets géométriques sont également rendus dans une seule et même passe, ce qui entraîne un coût supplémentaire.

3.6.2.2 Rendu par échantillonnage de plans de coupe

C'est un algorithme de rendu orienté objet. Il consiste à rendre un volume de données par une succession de plans orthogonaux à la direction d'observation, chacun portant des textures 2D générées par le résultat de la transformation en couleurs RGBA des concentrations locales C par la fonction de transfert TF. L'intégration des différents plans est effectuée par la fonctionnalité matérielle d'alpha blending. Cette technique supporte naturellement le rendu hybride géométrique/volumique, mais engendre de sévères approximations lors de l'utilisation de projection d'observation perspective (voir figure 3.21 à gauche). Cependant pour des applications où le réalisme et la précision ne sont pas de mise, cette technique représente une implémentation légère et portable d'un rendu volumique.

Après ce rapide état de l'art du système de rendu qu'un simulateur d'évacuation doit comporter, le chapitre suivant met en évidence l'importance d'une modélisation physique adaptée à des individus virtuels en situation d'évacuation.

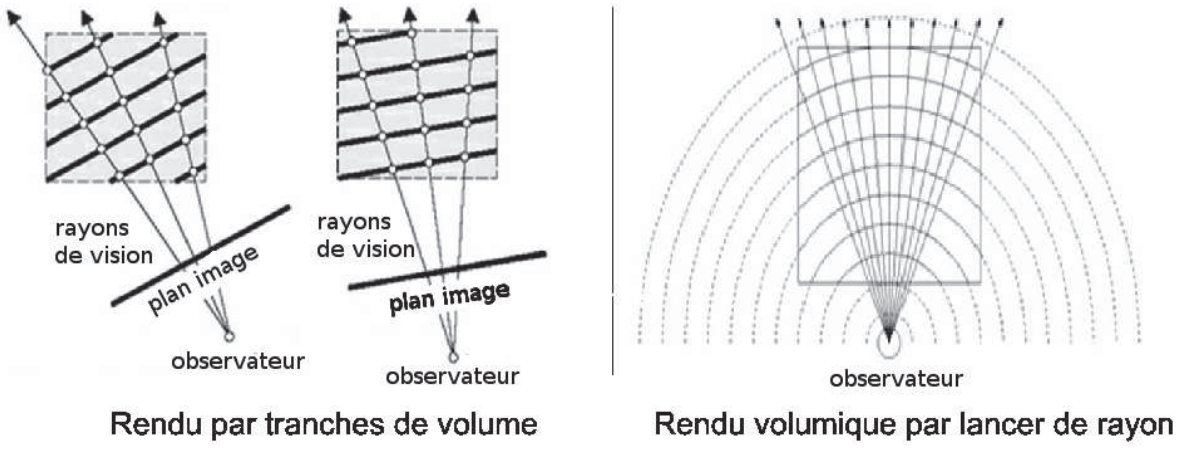


FIGURE 3.21 – Les deux principales approches de rendu volumique.

Chapitre 4

Analyse de divers simulateurs existants

De nombreux simulateurs d'évacuation de bâtiment sont actuellement disponibles. Ils se différencient sur de nombreux points, notamment sur le modèle de décision qu'ils intègrent. Les quatre systèmes exposés ici représentent un échantillon représentatif des différents modèles de prise de décision. Ils ont comme points communs d'intégrer des modèles individuels des évacuants (microscopique) et d'utiliser à la fois des comportements planifiés et navigationnels.
Un comparatif plus complet et détaillé est mis à disposition par le NIST [D. Kuligowski 2005].

4.1 Exodus

Peu d'informations sur ce simulateur ont pu être collectées dans la mesure où c'est un système commercial. Ce qui suit correspond aux informations divulguées dans des articles récemment mis à disposition [Veeraswamy 2009]. Ce système présente comme principale caractéristique d'être fondé sur des modèles statistiques issus de nombreuses expérimentations et représente donc une approche très pragmatique du comportement humain.

4.1.1 Modèle de l'environnement

Dans Exodus, le partitionnement de l'environnement est hiérarchisé suivant deux niveaux d'abstraction. Chaque étage est représenté par :

- un partitionnement grossier par des cellules convexes représentant les pièces.
- une grille régulière fine permettant une décision plus précise à l'intérieur des pièces.

Cette représentation hiérarchique permet l'utilisation d'un algorithme de planification hiérarchique de type HPA^* [Botea 2004].

4.1.2 Modèle décisionnel

La détermination du meilleur chemin entre une pièce et la sortie du bâtiment dans le graphe de cellules précédemment défini est réalisée en trois phases :

1. l'ensemble de tous les chemins possibles est déterminé par un algorithme d'exploration de graphe,
2. chacun de ces chemins se voit attribuer différents scores selon les différents critères de préférences de chemin que sont :
   - la distance parcourue,
   - le temps de parcours,
   - la moyenne des angles de trajectoires,
   - le nombre de virages,
   - le nombre de points de décision.

3. les scores sont ensuite normalisés puis pondérés par le poids accordé par chaque individu à chaque critère. Ces différents scores sont alors sommés pour donner le coût effectif individualisé.

$$Cost = \sum_{i=0}^{i<nbRPC} ScoreRPC_i * PoidsRPC_i \tag{4.1}$$

$Cost$: coût du chemin
$nbRPC$: nombre de critères de préférences
$ScoreRPC_i$: score du chemin pour le critère de préférence i
$PoidsRPC_i$: poids accordé par l'individu au critère i
L'individu sélectionne alors le chemin ayant le plus petit coût à ses yeux. Des replanifications régulières sont effectuées et un chemin alternatif est choisi si le coût de ce dernier diffère d'au moins 10% du coût du chemin suivi jusqu'alors.

4.1.3 Critiques

Le modèle de décision est bien original par rapport aux techniques de l'état de l'art mais aussi étrange que cela puisse paraître, il ne semble pas intégrer les critères primordiaux de ce genre de simulation que sont la présence de feu et de fumée dans la décision.

De plus, les décisions situées sont discrètes, ce qui implique des trajectoires comportant des angles droits et donc bénéficiant de peu de degré de liberté. Les collisions physiques entre individus ne sont pas modélisées dans la mesure où les individus ne convoitent jamais une cellule occupée.

La méthodologie de modélisation à base d'expérimentations employée est à la fois sa plus grande force et sa plus grande faiblesse. En effet, les expérimentations ne faisant varier qu'un petit nombre de paramètres, les modèles déduits ne tiennent compte que partiellement des interactions entre les nombreux facteurs influant la décision.

Enfin, afin de réduire le nombre de chemin à déterminer, les individus sont regroupés selon leur proximité et un chemin est déterminé pour chacun de ces groupes. Cette stratégie de regroupement d'individus par pièce entraîne une hypothèse forte sur le mimétisme (preuve sociale) dont le modèle d'Exodus fait preuve.

4.2 Evac/FDS

Evac est un système d'évacuation libre (licence GPL) développé par le Technical Research Centre of Finland (VTT [1]), il est fortement couplé à FDS et permet ainsi la simulation conjointe de l'évacuation et de la propagation d'incendie.
L'algorithme de décision des individus peut ainsi se résumer par [Heliövaara 2007] :

1. un choix sur critères de préférence de la catégorie de sortie
2. une optimisation du temps estimé jusqu'à la sortie pour les sorties de la catégorie élue
3. une simulation de propagation de fluide incompressible pour déterminer la trajectoire à emprunter
4. suivi du champ de vecteurs ainsi généré auquel s'ajoutent des comportements navigationnels sociaux (Helbing [Helbing 2000]) et instinctifs (peur du feu et de la fumée).

4.2.1 Modèle de l'environnement

Le partitionnement utilisé ici est emprunté à FDS. Il consiste en une description volumique minimal des matériaux du bâtiment.

1. http://www.vtt.fi

4.2.2 Modèle décisionnel

4.2.2.1 Sélection de la catégorie de sortie préférée

Les catégories se voient attribuer un ordre de préférence propre à chaque individu. Cet ordre permet alors de trier les catégories des sorties connues en fonction de :

1. la familiarité ;
2. la visibilité ;
3. la toxicité à la sortie.

La plus prioritaire est élue.

4.2.2.2 L'optimisation du temps à la sortie

L'utilisation de l'équilibrage de Nash avec un apprentissage naïf est intéressante dans le sens où l'individu prend une décision en fonction des actions des autres individus, c'est donc une forme de comportement rationnel social. La foule aura donc tendance à se répartir sur les issues convoitées.

L'expression du temps estimé à la sortie est de la forme :
$(tempsdequeueestimé + tempsestiméjusqu'àlasortie) \times pondération$

Le temps de queue estimé à la sortie i pour l'individu k s'exprime ainsi :
$(Sql(i) \times Beta(i) \times Xk)V(k,i)$
$Sql(i)$: nombre d'individus entre k et la sortie i
$Beta(i)$: capacité de la sortie i
$V(k,i)$: visibilité de la sortie par l'individu k (0 ou 1)
Xk : variable de décision de l'agent k
Le temps estimé jusqu'à la sortie i pour l'individu k s'exprime sous la forme :
$Xk \times \frac{d(k,i)}{v(k)}$
$\frac{d(k,i)}{v(k)}$: distance de k à la sortie i / vitesse actuelle de l'individu k
La pondération est décrite par l'expression :
$1 - \alpha(k) \times Xk \times Q(k,i)$
$\alpha(k)$: seuil de changement de comportement. Plus celui-ci est petit, plus l'individu sera tenté de changer de stratégie.
$Q(k,i)$: représente l'attirance de k pour l'issue i (0 ou 1)

4.2.2.3 Résolution de la trajectoire à la sortie

Le modèle de simulation de fluide de FDS est utilisé afin de créer un champ de vecteur V correspondant au mouvement d'air si la sortie sélectionnée était un extracteur. Chaque personnage suit ensuite l'équation de Helbing [Helbing 2000] augmentée de forces répulsives au feu et à la fumée en prenant comme direction désirée ($e(t)$) la direction du champ V à l'emplacement de l'individu.

4.2.3 Critiques

Evac/FDS utilise un mécanisme rationnel social en sélectionnant l'issue en fonction des actions des autres personnages, ce qui représente un exemple de comportement rationnel social sans communication.
Les communications entre individus sont approximées par un modèle de propagation spatial stochastique de l'information qui paraît une bonne approximation du mécanisme cognitif de diffusion de l'information sur un critère de proximité.

Cependant, l'algorithme utilisé pour la sélection de l'issue est basé sur la théorie des jeux à n joueurs (équilibre de Nash) et pose de nombreuses hypothèses très discutables sur la connaissance d'un individu de ces congénères et des sorties qu'ils convoitent :

- chaque individu connaît les buts que chacun de ses congénères considère comme plausibles
- l'utilisation d'une estimation de distance au but de nature géométrique dans la fonction d'utilité individuelle des individus pose une hypothèse simpliste sur le processus d'appréciation individuelle de la distance au but.

De plus, Evac ne crée qu'une base symbolique de raisonnement très succincte pour le choix de l'issue et utilise une connaissance non subjective (omniscience du bâtiment) et incomplète (pas de feu, ni de fumée, ni les autres personnes) pour le calcul du chemin.

On peut également remarquer que les comportements d'aversion au feu et à la fumée ne possèdent qu'une composante normale au phénomène : il peut donc en résulter une force dont l'orientation est l'inverse de la force résultante de la décision rationnelle avec laquelle elle est sommée. Ce qui peut entraîner une diminution de la vitesse lorsque la force rationnelle n'est pas complètement orthogonale au gradient de feu ou de fumée. Pire encore, la décision rationnelle ne prenant pas en considération le feu, on peut voir apparaître des comportements "indécis" si la force résultante de la décision navigationnelle est l'exacte opposée de celle produite par la décision rationnelle.

Enfin, la non prise en compte du feu dans la résolution planifiée de la trajectoire peut entrainer des comportements aberrants. En effet, si une sortie a été sélectionnée pour sa familiarité, les personnages vont se ruer vers la sortie comme s'il n'y avait ni feu ni fumée et se trouveront coincés si cette sortie est bouchée par du feu ou de la fumée à cause de leur répulsion à ces phénomènes. La remise en cause de la stratégie ne prenant pas en compte ces facteurs, les individus ne la modifieront donc pas.

4.3 HiDAC

Ce prototype de logiciel a été conçu par [Pelechano 2007] lors de son travail d'extension des travaux de [Reynolds 2006] pour le cas du comportement humain.

Ce modèle présente clairement les deux niveaux navigationnels et planifiés, aussi appelés respectivement ici module bas-niveau et module haut-niveau. La partie navigationnelle a des relations avec la partie planifiée pour signifier un changement d'état du monde déjà connu comme une modification des chemins possibles (une porte s'est fermée, un chemin est obstrué...), ou pour demander un nouveau point de passage. Les deux niveaux d'abstraction ont accès à des connaissances à la fois physiologiques comme l'âge ou le poids et à des connaissances psychologiques comme le stress.

4.3.1 Modèle de l'environnement

Le bâtiment est décrit par un graphe minimal de cellules convexes. Une structuration en grille régulière de la partition des individus semble être utilisée afin de favoriser les requêtes de voisinage mais n'est pas décrite explicitement dans les articles.

4.3.2 Modèle décisionnel

4.3.2.1 Module rationnel

La décision est gérée à haut-niveau par une représentation de l'environnement sous forme de grille régulière et d'un graphe de connectivité des pièces tenant le rôle de carte mentale. Chaque agent possède également la connaissances sur ses objectifs : les sorties connues, c'est à dire au minimum l'entrée qu'il a utilisé. Chaque agent possède également un état mental, qui représente le niveau actuel

de ses émotions, comme par exemple le stress. Pour gérer les états mentaux, N. Pelechano utilise un moteur émotionnel (PMFserv [Gaskins 2006]) qui sert à maintenir chaque état mental. Elle présente deux types d'agents vis-à-vis du critère de connaissance du bâtiment. Le premier type d'agent connaît le minimum sur le monde environnant, et l'autre qui est dit agent "entraîné", a la connaissance de l'intégralité de la topologie du bâtiment (la configuration des pièces) et des ses sorties (entrées et issues de secours). Le principal comportement du modèle de Pelechano décrit plus bas, est un comportement de leadership par un agent entraîné ou non, qui mène d'autres agents (non-entraînés pour une énorme majorité) vers la sortie, selon le schéma 4.1.

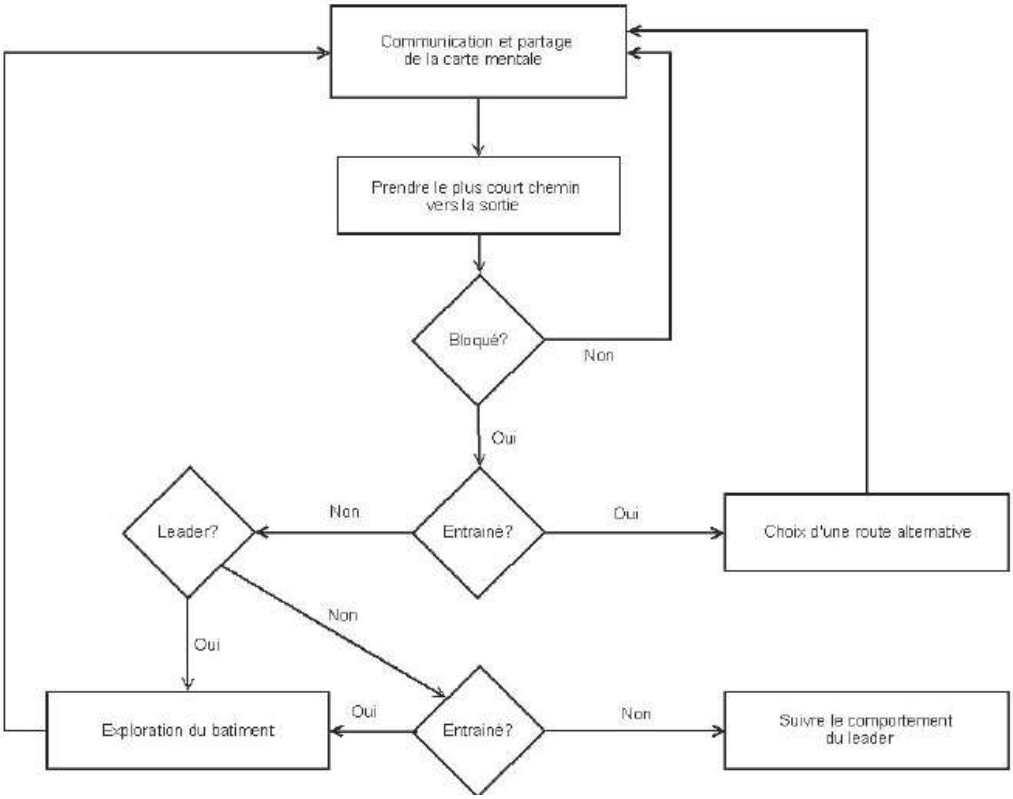


FIGURE 4.1 – Modèle décisionnel de Hidac

4.3.2.2 Module navigationnel

Le déplacement bas niveau se fait par une série d'attracteurs (checkpoints), qui attirent les agents, tout en prenant en compte pour le déplacement de chaque agent les autres forces qui repoussent les agents, comme les obstacles ou les murs. Les autres agents font également office de répulseurs par une zone personnelle prédéfinie, l'espace personnel, qui doit rester inviolé pour chaque individu. Le mouvement de l'agent est donc exprimé par l'équation (4.2) :

$$F_i^{To}[n] == F_i^{To}[n-1] + F_i^{At}[n]w_i^{At} + \sum wF_w^{Wa}i[n]w_i^{Wa} + \sum kF_k^{Ob}i[n]w_i^{Ob} + \sum jF_j^{Ot}i[n]w_i^{Ot} \quad (4.2)$$

Dans cette formule, on somme les forces qui vont entrer dans la composition du mouvement. Tout d'abord le mouvement de l'agent i au pas de temps précédent ($F_i^{To}[n-1]$) afin de garder une logique dans le déplacement sur plusieurs pas de temps (permet d'éviter certains phénomènes d'oscillation), ensuite de l'attracteur en cours qui tire l'agent vers lui ($F_i^{At}[n]$), et enfin les forces de répulsions des

murs ($F_w^{Wa}i[n]$), des objets et obstacles($F_k^{Ob}i$) et des autres agents ($F_j^{Ot}i$). Le facteur w est le poids que l'on donne à chaque force (fonction des états mentaux de l'agent). Les divers comportements exprimés par les forces prendront donc plus ou moins d'importance dans cette composition de force en fonction des émotions de l'agent.

4.3.2.3 Les comportements émergeants

Cette modélisation sous forme d'équation contrôlée par les émotions permet l'expression de divers comportements émergeant de la variation de ces dernières.

1. Le queuing (les agents se mettent en file indienne), comportement qui se retrouve lors d'un niveau de stress bas.
2. Le pushing, où des agents dépassant un niveau de stress ou d'excitation voient leur espace personnel diminuer de par la nécessité grandissante de survie. En se rapprochant d'un autre agent ayant un espace personnel plus grand, ils vont entrer dans son espace personnel sans pour autant faire entrer cet autre agent dans le leur. Cela aura pour effet de pousser un agent calme, et ainsi de faire un passage à l'agent plus stressé.
3. La chute d'un agent et la possibilité de se faire piétiner. Chaque agent a un certain niveau d'équilibre, et si il subit des forces de pushing qui dépassent un certain seuil dans la même direction, l'agent tombe et devient un obstacle piétinable si les agents environnants ont un niveau de stress élevé.
4. Propagation de la panique. Soit par communication directe entre agents, soit par perception de l'état de panique des agents environnants. La panique doit se propager selon un taux de diffusion, avec un seuil à atteindre. Si le seuil est atteint alors l'agent panique (augmentation de la vitesse, de la force .... et comportement de panique).

4.3.3 Critiques

Ce modèle s'appuie sur l'hypothèse que l'adjonction de règles locales d'inhibition/activation de comportement soit le moyen d'augmenter le réalisme des mouvements des individus dans une foule. Elle introduit ainsi une sorte de logique floue sur les comportements de Reynolds en les pondérant dynamiquement. Ce mode de contrôle par des poids est également utilisé dans le modèle d'intégration physique, permettant un lissage de la trajectoire au cours du temps et par extension d'éviter les oscillations dûes à l'utilisation de comportements navigationnels à règles discrètes. Les collisions physiques sont évitées à la manière de [Helbing 2000] en appliquant une force répulsive lors d'une interpénétration des personnages. Enfin, son principal attrait vient du fait qu'il est le seul simulateur à intégrer un moteur d'évaluation des émotions (PMFServ [2]) permettant ainsi d'influencer la décision ainsi que le mouvement effectif en résultant. Ce simulateur bénéficie d'un réalisme acceptable grâce à toutes les attentions accordées pour éviter les observations non désirées. Il n'en reste pas moins que le modèle est très spécialisé et donc difficilement généralisable. Enfin, ce simulateur d'évacuation n'ayant pas pour objet les cas d'incendie, ce dernier fait abstraction des contraintes engendrées par ces derniers (feu, fumée).

4.4 Mass Egress

Ce simulateur illustre les travaux de thèse effectués par [Pan 2006] en tentant d'intégrer les résultats de ces recherches sur l'apparition de divers comportements inadaptatifs chez l'être humain.

2. Exemple d'applications de PMFServ

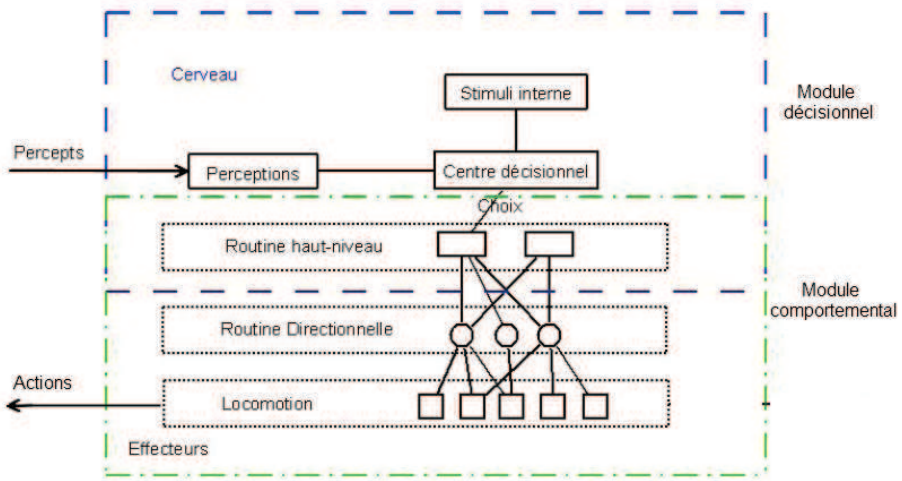


FIGURE 4.2 – Architecture de Mass Egress

L'architecture générale de Mass Egress est présentée sur la figure 4.2.

4.4.1 Modèle de l'environnement

La partition du bâtiment n'est pas décrite dans la mesure où aucun algorithme de planification n'est utilisé. Cependant, une grille régulière est utilisée pour stocker les individus favorisant les requêtes de voisinage et par suite la gestion des collisions entre individus.

4.4.2 Modèle décisionnel

A la façon de N. Pelechano, il distingue deux niveaux dans son architecture : rationnel et navigationnel (appelé ici niveau comportemental).

4.4.2.1 Le niveau rationnel

Le niveau rationnel est constitué, à l'instar de Hidac d'une multitude d'arbres de décision permettant de déduire quel comportement de bas niveau s'exprime en fonction des perceptions et de l'état mental de l'agent. Cependant aucune planification n'est effectuée à ce niveau. Le modèle proposé n'a été que partiellement implémenté compte tenu de la difficulté à estimer le stress ainsi que les divers conditions psycho-sociales d'activation des comportements (cf plus bas).

Xiaoshan Pan [Pan 2006] envisage trois manières distinctes (modes) pour résoudre un problème :

- L'instinct (réaction tropique)
- L'expérience (réaction hystérétique)
- La rationalité (réaction cognitive)

L'expérience se déclenche si la situation en cours a déjà été vécue par l'agent, auquel cas l'agent sait comment y réagir. L'instinct prend également le pas sur la rationalité si le niveau de stress du dit agent dépasse un certain seuil. Ces divers modes de décision sont modélisés dans plusieurs arbres de décision distincts.

Le module décisionnel se charge alors de déclencher l'un des comportements décrits plus bas en fonction de l'état interne de l'agent.

4.4.2.2 Le niveau comportemental

Il est constitué des divers comportements atomiques caractéristiques isolés par l'auteur :

Faire la queue
Il correspond au comportement le plus adaptatif et rationnel décrit par X.Pan (mode "rationnel"). Il apparaît lorsque le stress des individus est bas.

Compétitif
Ce comportement apparaît principalement en mode "instinctif", il correspond à faire abstraction des contraintes sociales et à maximiser son accomplissement individuel. Dans le cas de l'évacuation, il correspond au comportement de "pousseur" dans la terminologie de HiDac et s'exprime par l'inhibition de ces congénères dans le comportement.

Leader/suivant
On retrouve dans le modèle de Mass Egress, le comportement de leader et de suivant de Pelechano. Mais c'est surtout par l'existence de suivants que des leaders vont exister. En effet ce comportement de suivant se déclenchera lorsque l'agent est calme, et qu'il ne sait pas quoi faire, il décidera alors de suivre des comportements dits sociaux. Un leader peut se caractériser de différentes manières, soit par son insatisfaction (exemple : modèle SATALT [Simonin 2000]), soit par ses connaissances des lieux (leader de Pelechano), soit par son rang social supérieur (présence d'une autorité, d'un patron...), ou encore par identité sociale (fonction du rôle dans l'organisation).

Comportement de troupeau
C'est un comportement basé sur le concept de preuve sociale. Le comportement de troupeau (Herding) peut mener à la formation de congestions. Comme dans l'exemple d'une salle à plusieurs sorties. Un agent aura tendance à se diriger vers une sortie où il y a beaucoup de monde, alors que d'autres sorties de la salle pouvaient être meilleures, et surtout non encombrées. Le comportement en troupeau se fait en majorité lorsque le stress est élevé. Dans ce cas, c'est toujours de la preuve sociale, mais plus dans un cas de compétitivité négative (par opposition à la compétitivité positive qui est la coopération). Dans le cas d'un stress bas, un comportement de sélection de choix d'actions par preuve sociale et suivi serait plutôt du comportement de type file d'attente.

Altruisme
Un dernier comportement de preuve sociale et de mimétisme est l'altruisme. Selon le niveau de stress, si l'on voit quelqu'un en difficulté et que quelqu'un va l'aider, l'agent va également à son secours. Ceci est dû non seulement au mimétisme mais également au principe d'obligation sociale. En effet, voir un agent en difficulté ne va pas forcément pousser l'agent spectateur à l'aider, alors que voir un autre agent venir en aide à celui en déroute forcera inconsciemment le premier agent à apporter également son aide.

Ces divers comportements sont implémentés par des comportements réactifs embarqués dans une machine à état agissant sur les effecteurs de chaque agent.

4.4.3 Critiques

Afin d'émettre une critique précise sur les comportements de Mass Egress, ces derniers ont été analysés en termes de composantes de force afin d'en extraire deux dimensions de contraintes (au sens

	Individuelle	Sociale
Compétitif	aller(sortie)	pas d'adaptation sociale
Faire la queue	suivre file d'attente puis aller(sortie) OU arriver(file attente)	adaptation sociale implicite
Suivre leader	arriver(leader)	ne suppose pas d'adaptation sociale
Altruisme	arriver(blessé) puis porter(blessé) puis aller(sortie)	ne suppose pas d'adaptation sociale
Troupeau	aller(sortie)	adaptation sociale

TABLE 4.1 – Tableau d'analyse dimensionnelle des comportements atomiques de Mass Egress

de [Ferber 1997]) : la dimension individuelle et la dimension sociale. Cette analyse est résumée par le tableau 4.1. Le manque d'information quant à l'implémentation des divers comportements nous a poussé à émettre certaines hypothèses. Ces hypothèses (entre parenthèses dans le tableau) mettent en évidence le problème lié à définir des comportements atomiques de trop haut niveau : la perte du contrôle fin du comportement. En effet, on peut soulever cette question : les comportements " altruisme " et " suivre leader " ont-il une composante sociale ? Autrement dit, ceux qui suivent un leader seront-ils affectés par des forces de cohérence sociale comme les personnages en situation de herding ou au contraire seront-ils plutôt compétitifs ? Cette question remet en cause la classification des comportements proposée car elle n'est pas constituée de comportements disjoints du point de vue des dimensions individuelle et sociale et fait donc perdre un niveau de contrôle important sur les comportements.

Malgré l'infime partie de ces travaux ayant une implémentation concrète, ce modèle est intéressant de part l'étude sur l'adaptation comportementale en cas d'urgence sur critères psychologiques et sociaux qui y est menée. En effet, X. Pan dans ces travaux a donné des bases cognitives sur la simulation de flux humain en situation d'urgence en se fondant sur la littérature psychologique et sociale connexe. Cependant, la faiblesse cognitive de l'implémentation est frustrante car aucune décision rationnelle autre que celles modélisées statiquement dans les arbres de décision n'est possible. Il reste donc malgré la richesse du modèle, une implémentation strictement réactive.

Enfin, la prise en compte de l'incendie n'est jamais mentionnée dans ces travaux.

4.5 PathFinder

Il est le dernier simulateur en date des simulateurs d'évacuation. Il est cité car il présente la particularité d'être fondé sur la technique de Steering Inverse qui rappelons-le, permet une planification locale de la trajectoire.

4.5.1 Modèle de l'envionnement

L'environnement est partitionné à travers un éditeur puis une triangulation de Delaunay lui est appliquée afin de créer un réseau de cellules adjacentes.

4.5.2 Modèle décisionnel

Les deux niveaux décisionnels déjà présent dans les autres simulateurs y sont représentés.

4.5.2.1 Modèle de planification

Il n'est pas associé à l'individu en tant que tel. En effet, une seule planification par A^* est effectuée lors du positionnement de l'individu.

4.5.2.2 Modèle navigationnel

Le seul comportement du système est un comportement de steering inverse permettant de réparer localement le plan défini statiquement. Les différentes évolutions de l'agent se voient attribuer un coût vis-à-vis d'une évaluation intégrant les différentes contraintes de mouvement (incendie, autres individus et murs) ainsi que le plan préalablement défini (en tant qu'heuristique de coût), puis la meilleure alternative est choisie.

4.5.3 Critiques

Malgré l'originalité et la rapidité des techniques utilisées dans ce système, les principes sous-jacents basés sur un plan global et une connaissance que l'on pourrait qualifier d'omnisciente de la topologie du bâtiment sont des hypothèses simplistes sur les mécanismes cognitifs mis en œuvre lors de l'évacuation d'urgence.

4.6 Résumé comparatif et critiques

La table 4.2 résume l'analyse de ces différents systèmes :

- Exodus, malgré la richesse de son modèle cognitif, met en exergue l'importance d'un modèle de déplacement continu pour le réalisme de la simulation.
- Evac/FDS présente la particularité d'être un modèle réel de simulation sans considération temps-réel mais pose des hypothèses cognitives discutables sur la connaissance individuelle des évacuants. De plus, il met en avant la nécessité d'intégrer un modèle décisionnel évitant l'émergence de comportement bloquant l'individu dans sa progression.
- Hidac est attrayant par la richesse des comportements qu'il arbore malgré son orientation fortement temps-réel et visuellement réaliste. Il pose également des hypothèses cognitives originales tel que la connaissance limitée du bâtiment ou encore un modèle physiologique de résistance à la pression physique.
- Les travaux de X.Pan sont intéressants par le modèle logique d'apparition des comportements non adaptatifs qu'ils proposent malgré le fait que l'implémentation réalisée ne l'intègre pas.
- PathFinder quand à lui propose une alternative intéressante à la replanification intégrale du chemin lors de changement de l'environnement, mais cette alternative entraîne de nouvelles problématiques comme la validité du plan réparé ou encore l'horizon temporel de planification locale à utiliser.

On peut trouver deux critiques communes à ces systèmes :

- l'intégration des contraintes de feu et de fumée dans le raisonnement des individus est très succincte ou nulle,
- certaines hypothèses sur les connaissances individuelles sont discutables et sont en désaccord avec la théorie de la rationalité limitée.

On peut déduire des travaux précédents et de leurs critiques qu'un modèle de simulation du comportement des individus doit intégrer à la fois des comportements planifiés et navigationnels ainsi que différents niveaux de perception et de connaissance de l'environnement. Sur la base de cette analyse

	Modèle de comportement	Intégration incendie	Hypothèses fondatrices invalides
Evac/FDS	Hybride : fluide + navigationnel	Oui	Omniscience Connaissance des issues connues des autres
Mass Egress	Purement réactif	Non	Classification des comportements types
Hidac	Hybride : A^* + Steering	Non	Omniscience
Exodus	Cognitif	Oui	Omniscience
Pathfinder	Hybride : A^* + Steering inverse	Non	Omniscience + Intention fanatique
EveLife	Hybride : FIM modifiée	Oui	Néant

TABLE 4.2 – Tableau récapitulatif des principaux simulateurs d'évacuation existant ainsi que notre simulateur EveLife.

nous avons conçu le modèle de simulation microscopique qui a été implémenté dans notre simulateur EveLife. La présentation de ce modèle et de son implémentation font l'objet de la partie suivante.

Deuxième partie

Contribution

Chapitre 5

Modèle de simulation microscopique proposé

5.1 Introduction

La simulation d'évacuation de bâtiment en cas d'incendie a été analysée en se basant sur la pyramide de modélisation comportementale microscopique de Funge et al. [Funge 1999] , illustrée par la figure 5.1, qui permet une séparation conceptuelle hiérarchique des différents traitements à effectuer pour simuler le comportement d'un individu virtuel autonome.

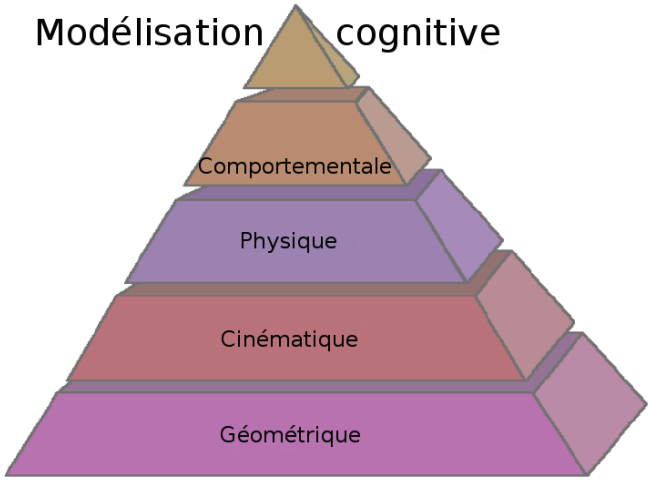


Figure 5.1 – Pyramide de simulation comportementale de Funge et al.

Il a été constaté que les deux grands mouvements de la psychologie (comportementalisme et cognitivisme) sont difficilement dissociables. En effet, ils partent tous deux des mêmes principes d'observation mais le premier cherche à trouver une explication externe à l'individu alors que le second prospecte des mécanismes internes.
Dans notre modèle, les deux couches, comportementale et cognitive, ont donc été fusionnées. Enfin, l'apex cognitif a été séparé en décision située et décision logique, à la manière des travaux de Donikian et al. [Donikian 2008].
Les processus cognitifs de gestion de la mémoire s'inspirent des travaux de Newell et Simon [Newell 1972].

Le suite de ce chapitre est organisée comme suit. Nous introduisons d'abord les termes utilisés dans ce chapitre. Puis nous décrivons la couche comportementale et cognitive de notre architecture de simulation microscopique et notre modèle d'agent situé, caractérisé par la prise en compte des axiomes de la rationalité limitée.

5.2 Terminologie utilisée

Nos travaux sont focalisés sur la décision située. Les individus simulés sont donc des agents situés. Nous allons définir les principaux termes qui seront utilisés par la suite. La figure 5.2 montre comment ces termes sont liés entre eux et permettent de définir notre modèle d'agent situé.

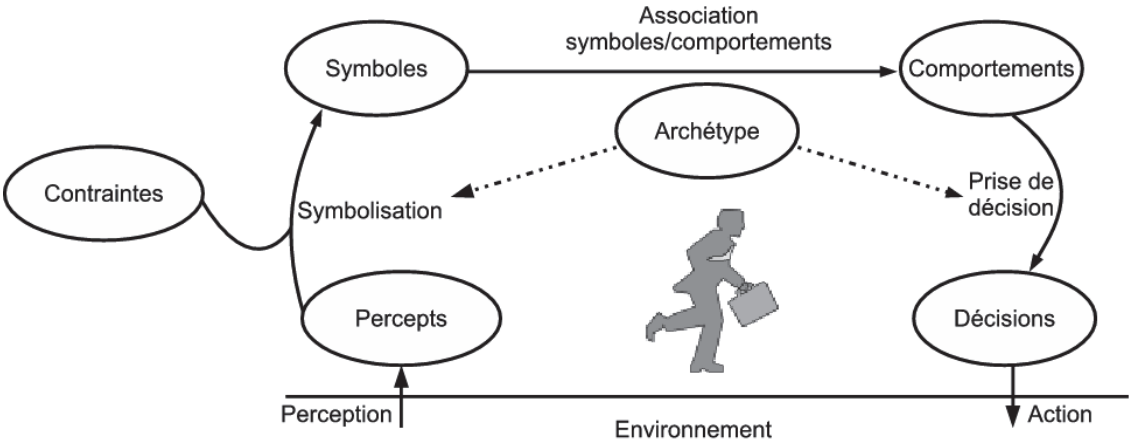


FIGURE 5.2 – Vue générale du modèle de décision sous contraintes.

Agent situé Un agent situé est une entité logicielle évoluant dans un environnement possédant une métrique. Les agents situés sont utilisés pour résoudre des problèmes ayant une forte composante spatiale. La poursuite d'un objectif géographique en est le principal exemple : il consiste alors à atteindre un objectif en prenant en compte les diverses contraintes de mouvement imposées à l'agent.

Archétype comportemental Un archétype comportemental est l'ensemble de propriétés intrinsèques de l'agent, différenciant son comportement de celui de ses congénères.

Percept Les percepts sont les éléments bruts acquis par le système de perception d'un agent situé. Aucune sémantique ne leur est alors associée, celle-ci sera liée aux processus d'interprétation qui leur seront appliqués. Exemple : un mur, un agent, ...

Contrainte Une contrainte représente une limitation des possibilités de déplacement d'un agent, provenant par exemple de la présence d'obstacles ou d'autres agents dans l'environnement. Exemple : un mur sera interprété comme une contrainte spatiale obligeant un agent à le contourner.

Symbole Les symboles sont les éléments produits par le processus de symbolisation et manipulés par les processus de prise de décision que sont les comportements. Ils possèdent donc une sémantique en accord avec ces processus. Exemple : une force associée au percept mur et représentant la contrainte que subira chaque agent s'approchant du mur.

Symbolisation La symbolisation est un processus d'enrichissement sémantique, pouvant être appliqué tant aux percepts qu'à d'autres symboles. Il produit les symboles manipulés par les comportements. Exemple : production d'une force radiale associée au percept agent afin de représenter la contrainte d'espace vital, repoussant les autres agents afin qu'ils ne s'approchent pas trop près.

Comportement Un comportement est une procédure de traitement utilisant des symboles et engendrant une décision. On distinguera les comportements navigationnels, qui ne manipulent généralement qu'un nombre réduit de symboles et produisent une réaction, et les comportements planifiés, qui utilisent l'ensemble des symboles applicables et ont pour objectif l'atteinte d'un but.

A ne pas confondre avec le comportement observé. Exemple : le comportement répulsif faisant prendre à l'agent la décision de s'écarter d'un congénère afin de satisfaire la contrainte d'espace vital.

Prise de décision La prise de décision consiste à déclencher les différents comportements, ce qui donnera donc lieu à plusieurs décisions possibles. Exemple : un comportement planifié produit la décision de se diriger vers le sud et un comportement navigationnel produit la décision de se diriger vers le nord du fait de la présence d'un autre agent très proche côté sud.

Décision Une décision est le résultat d'un comportement. Si elle est applicable, elle peut donner lieu à une action effective de l'agent dans son environnement. Exemple : vecteur déplacement dans l'espace.

Action Le processus d'action consiste à sélectionner la décision la plus adaptée en fonction de son contexte d'exécution (symboles en mémoires). Exemple : si un comportement viole une contrainte, la décision qu'il a engendrée peut être jugée inapplicable : ce sera le cas pour le déplacement au sud produit par notre exemple de comportement planifié.

5.3 Couche comportementale et cognitive

Elle est dérivée de l'architecture d'agents logique InteRRaP (voir section 3.4.2) et est spécialisée pour le problème situé qu'est le déplacement d'agents. Les principes communs aux deux architectures sont :

- la séparation des traitements et des données,
- les couches supérieures ont accès aux données des couches inférieures.

Il s'agit d'une architecture en couches verticales dans laquelle chaque couche est constituée d'une base de connaissances, qui contient des symboles sur l'état du monde, et d'un ou plusieurs processus de traitement de ces derniers : les comportements. A l'instar d'InteRRaP, le modèle d'exécution est articulé en deux phases :

1. La phase ascendante est le processus qui produit des symboles à partir des percepts captés dans l'environnement. Chaque couche a pour objectif de produire une sémantique des percepts plus complexe que sa couche inférieure.
2. La phase descendante est le processus de prise de décision. Chaque couche communique sa décision à la couche inférieure, qui sera alors enrichie selon l'aspect dont elle est chargée, jusqu'à produire une action que l'agent pourra effectuer dans son environnement.

A la manière de Simulem [Donikian 2008], notre architecture découple le problème à résoudre en plusieurs couches prenant en charge différents aspects du problème (voir figure 5.3) :

1. Une couche de coopération, chargée de négocier des plans collaboratifs et de communiquer de nouveaux buts individuels à la couche inférieure.
2. Une couche logique non située, chargée de résoudre des problèmes sans composante spatiale. Ses interactions avec la couche située sont :
   - la requête de nouvelles informations, par exemple pour déterminer si un objectif peut être atteint,
   - la transmission des nouveaux buts à atteindre.
3. Une couche logique située, chargée de générer les actions à entreprendre afin d'atteindre un objectif situé.

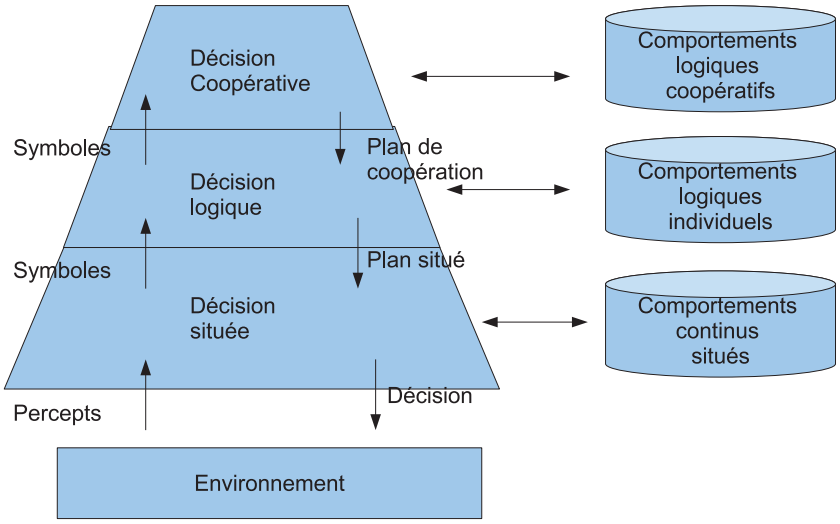


FIGURE 5.3 – Modèle général de notre architecture.

Le problème d'évacuation ayant une forte composante spatiale, nous avons concentré nos efforts sur la couche logique située et notre proposition concerne presque exclusivement cette couche. La couche décision logique ne traite pour l'instant qu'un seul comportement : « suivre une personne qui déclare connaître une issue ». Le but transmis à la couche située est de rallier l'endroit où se situe cette personne. La couche coopération n'a pas été développée. La figure 5.3 a pour but de démontrer que cette contribution s'insère parfaitement dans les travaux effectués jusque là et pourra enrichir un simulateur plus complet.

5.4 Le modèle d'agent situé

Nous détaillons maintenant le fonctionnement de la couche située et en particulier le modèle d'adaptation du comportement qui en découle, principale contribution de cette thèse. Nous expliquons d'abord l'organisation hiérarchique des processus de raisonnement de l'individu. Puis nous décrivons l'organisation et le fonctionnement de la mémoire individuelle au sein de laquelle sont stockées les informations manipulées par ces processus, ainsi que les flux d'information, ascendant et descendant, qui la parcourent. Enfin nous explicitons comment cette organisation et ces processus permettent de proposer une implémentation du principe de la rationalité limitée.

5.4.1 Hiérarchisation des comportements

L'état de l'art a mis en évidence deux grandes classes de comportements :
- les comportements navigationnels, cherchant à satisfaire une ou plusieurs contraintes directement dérivée(s) des percepts,
- les comportements planifiés, cherchant à atteindre un but en prenant en compte l'ensemble des contraintes applicables.

Les premiers sont des comportements réflexes, par exemple s'écarter promptement au contact d'un autre individu, alors que les seconds expriment le raisonnement tel que nous l'entendons habituellement, c'est-à-dire la réflexion afin de prendre la bonne décision. Ces derniers mobilisent par conséquent davantage les moyens cognitifs de l'individu et on peut raisonnablement penser qu'ils seront davantage affectés par la limitation de ces moyens cognitifs que les premiers. C'est sur la base de cette considération

que nous proposons de hiérarchiser les comportements selon le degré d'implication des moyens cognitifs nécessaires à leur exécution : comportements navigationnels en bas de la hiérarchie et comportements planifiés en haut.
Cependant la hiérarchisation peut répondre à un autre besoin, celui de changer le niveau d'abstraction en changeant la représentation d'une contrainte ou d'un ensemble de contraintes afin de créer un contexte de satisfaction du problème plus complet. Le haut de la hiérarchie correspond alors au niveaux d'abstraction les plus élevés. Il sera ainsi possible de définir une hiérarchie au sein de chaque classe de comportements.
L'implémentation du deuxième axiome de la rationalité limitée que nous proposons consistera alors à inhiber les niveaux les plus hauts de la hiérarchie, autrement dit ceux correspondant aux niveaux d'abstraction du raisonnement les plus élevés, limitant ainsi les capacités de raisonnement de l'individu. Nous détaillerons ceci à la section 5.4.3.

5.4.2 Gestion de l'information

Nous décrivons à présent la manière dont l'information dont dispose l'agent va être gérée au cours du temps, depuis sa perception par le système de perception de l'agent jusqu'à sa disparition en passant par son utilisation. Pour cela nous rappelons le principe de fonctionnement de notre architecture, inspiré de InteRRaP (voir section 3.4.2), consistant en un flux ascendant et un flux descendant (voir figure 5.3).

5.4.2.1 Phase ascendante - Symbolisation des percepts et gestion de la connaissance

L'information est d'abord détectée par les moyens de perception dont est doté l'agent, on appelle alors ces informations les percepts.
Les flux ascendants de notre modèle représentent les processus cognitifs qui transforment les percepts en symboles aux différents niveaux d'abstraction.
Le module de symbolisation des percepts est chargé, pour chaque percept, de construire une représentation symbolique de la contrainte engendrée par ce percept. Les symboles seront ensuite manipulés par les algorithmes implémentant les comportements.
Nous avons fait la distinction entre comportements navigationnels et comportements planifiés. Nous organisons la mémoire individuelle de chaque agent en deux niveaux, correspondant aux deux types de comportements :

- La mémoire à court terme stocke les symboles associés aux comportements navigationnels, autrement dit qui ne sont pas associés à la résolution d'un but. Exemple : force répulsive due à la présence d'un mur.
- La mémoire à long terme est utilisée pour les symboles associés aux contraintes liées à la résolution d'un but. Exemple : densité de la foule.

Chaque type de comportement sera simulé par un algorithme spécifique qui travaillera avec un des deux niveaux de la mémoire uniquement : la mémoire à court terme pour les comportements navigationnels et la mémoire à long terme pour les comportements planifiés.
Les deux niveaux ne sont toutefois pas étanches et les symboles de la mémoire à court terme sont transférés dans la mémoire à long terme. Par exemple la présence d'un mur génère instantanément une force répulsive mais cette information sera également utilisée pour la planification, car il serait stupide de ne pas tenir compte de la présence d'un mur quand on planifie sa route.

L'information utilisée par un individu pour prendre une décision évolue au cours du temps. Elle est enrichie par les nouvelles connaissances acquises via la perception qu'il a de son environnement mais elle s'appauvrit également au cours du temps et ce pour deux raisons : l'oubli - involontaire

- et la crédibilité accordée à l'information - volontaire. Nous ne simulons pas l'oubli mais gérons la fiabilité temporelle de l'information, dont le principe consiste à supprimer les informations lorsque ces dernières n'ont pas été validées par une observation effective plus ou moins récente, ce lap de temps étant fonction de la nature de l'information : on peut considérer qu'un mur aperçu une heure plus tôt a toutes les chances d'être encore à la même place, ce qui n'est pas le cas pour un groupe d'individus par exemple. Nous implémentons ce concept en attachant à chaque symbole un estampillage temporel correspondant à l'instant où cette information a été acquise. La fiabilité est alors définie sur l'intervalle [0,1] par la formule suivante :

$$fiab = 1 - \frac{(T_{cur} - T_{obs})}{T_{max}} \tag{5.1}$$

avec $fiab$ l'indice de fiabilité d'un symbole de contrainte dynamique, T_{obs} l'estampillage temporel de la dernière observation, T_{cur} l'instant courant et T_{max} le temps maximum de croyance à une observation dynamique, qui est propre à chaque agent.
Lorsque $fiab$ atteint 0, le symbole est considéré comme non fiable et est retiré de la mémoire.

La représentation informatique des symboles n'est pas la même selon le type de mémoire : représentation vectorielle - point dans l'espace et rayon - pour la mémoire à court terme et représentation discrète - grilles régulières - pour la mémoire à long terme. La raison de cette distinction tient à l'algorithmique utilisée : calcul d'une force répulsive dans le premier cas et algorithme du plus court chemin dans le second.
Au niveau navigationnel chaque symbole vectoriel permettra de générer une force, déduite de la valeur du symbole et appliquée à l'individu. Le comportement navigationnel qui en découlera sera fonction de la somme des forces issues des différents symboles qui s'appliquent.
Au niveau planifié, les grilles ont toutes même résolution et une combinaison linéaire de celles-ci produira une grille unique qui sera utilisée par l'algorithme de recherche de plus court chemin.

5.4.2.2 Phase descendante - Prise de décision

Une fois l'ensemble des symboles générés, la phase descendante a pour objectif de déterminer la décision de l'agent, typiquement une volonté de déplacement au sein de son environnement. Notre modèle d'agent situé étant hybride planifié/navigationnel, les deux types de comportements y sont représentés.
Les comportements navigationnels sont implémentés sous forme de règles de répulsion/attraction associées à la contrainte de mouvement qu'ils représentent. Les symboles étant représentés sous forme vectorielle dans la mémoire à court terme, la force est évaluée en calculant la normale à l'obstacle, pondérée par une fonction de la distance de l'agent à celui-ci.
Les comportements planifiés impliquent l'existence d'un but à atteindre et la décision qui en découle va avoir pour objectif de se rapprocher de ce but. Face à plusieurs buts possibles, l'agent va devoir effectuer un choix. Ce choix sera essentiellement fonction de deux paramètres : la qualité du but et l'effort nécessaire pour l'atteindre.
Par exemple imaginons qu'un agent en situation d'évacuation d'un bâtiment connaisse deux issues possibles : une issue de secours lointaine et une porte plus proche mais incertaine car elle pourrait être fermée à clé. On voit bien que le choix n'est pas évident, sauf dans le cas où la seconde porte serait sur le chemin de la première évidemment, et que deux individus confrontés à un tel dilemme pourraient faire des choix différents. Il est donc nécessaire de prendre en compte ces deux paramètres afin de bien appréhender le problème.

Une fois les buts identifiés et caractérisés et la fonction de coût pour un agent calculée en toute cellule de la grille, la prise de décision se résume à trouver le plus court chemin menant l'agent au but

le plus proche au sens de la fonction de coût et de la caractérisation de ce but. Nous fournirons des détails sur notre implémentation d'une telle méthode de résolution au chapitre 7.
On remarque au passage que si la planification requiert l'existence d'un but, elle nécessite également de définir une fonction de coût. On en reparlera à la section 6.4.

5.4.3 Implémentation du principe de la rationalité limitée

Nous décrivons maintenant comment chacun des deux axiomes de la rationalité limitée a été implémenté dans notre modèle.

5.4.3.1 Axiome 1 : simulation du manque d'informations

Cet axiome illustre le fait que si l'individu disposait de toutes les informations utiles à l'évaluation du problème, il prendrait alors la bonne décision. De fait l'individu dispose rarement de toutes les données du problème : par exemple, alors qu'un automobiliste raisonne afin de trouver le chemin le plus rapide menant de sa position à un objectif, il peut très bien ignorer qu'un accident est survenu, rendant impossible le passage par une rue qu'il s'apprête à emprunter. Nous avons par ailleurs écrit que chaque individu dispose d'informations partielles sur l'état du monde dans lequel il évolue et que cette information est propre à chaque individu (voir section 5.4.2.1).

Notre implémentation est la suivante : chaque agent dispose d'une mémoire propre, qui n'est alimentée que par sa connaissance du monde à l'instant 0 de la simulation (connaissance de la cartographie d'une ville par exemple) et par les informations résultant des processus de symbolisation, autrement dit déduites des percepts acquis par les sens de l'agent. L'histoire de chaque agent lui étant propre, deux agents ayant la même connaissance de l'environnement au départ verront par conséquent leur connaissance évoluer différemment au cours du temps, chacun ayant une connaissance différente - et donc partielle - de l'état du monde à un instant donné.

5.4.3.2 Axiome 2 : simulation des lacunes en moyens cognitifs

Le principe de notre implémentation est de contraindre les processus cognitifs (flux ascendants de l'architecture). On peut ainsi stopper la production de symboles pour chaque contrainte à un certain niveau du processus de symbolisation. Au cours de la phase descendante (prise de décision), les symboles non produits ne pourront par conséquent pas être pris en compte par les comportements qui en font usage, ce qui se traduira par le fait que l'évolution des percepts correspondants n'aura aucun impact sur le raisonnement de l'agent, ceux-ci ayant été perçus par l'agent mais non intégrés dans sa base de connaissances.
Il est à noter que ce contrôle contraint de fait à la fois le processus de symbolisation (flux ascendant de l'architecture) et celui d'exécution des comportements exprimés (flux descendant).
Trois remarques importantes s'imposent :

- l'inhibition du processus de symbolisation peut ne s'appliquer qu'à certains symboles, pour un agent donné,
- la liste des symboles pour lesquels la symbolisation est inhibée peut être différente d'un individu à l'autre,
- cette liste peut évoluer au cours du temps, reflétant les changements dans l'attitude de l'agent.

Ces propriétés sont stockées dans l'archétype de comportement de chaque agent sous forme de niveau maximal de symbolisation de chacune des contraintes. Une contrainte pourra alors être prise en compte par des comportements de niveaux d'adaptation différents suivant les individus. Cela pourra aller de sa totale inhibition (symbolisation stoppée au niveau le plus bas) jusqu'à sa prise en compte

dans la satisfaction de l'objectif. Ce contrôle permet alors de simuler les lacunes en moyens cognitifs de l'agent dans la résolution du problème, deuxième axiome de la théorie de la rationalité limitée.

5.4.4 Principes d'instanciation du modèle

Le modèle ayant été décrit, nous allons à présent détailler les étapes nécessaires afin de définir un simulateur basé sur ce modèle.

L'instanciation consiste à définir les composants suivants du modèle :

- l'environnement déambulable : position des obstacles, coût des déplacements en tout point,
- les buts ou objectifs à atteindre,
- les contraintes et leurs percepts associés, autrement dit par quels sens ils sont détectables,
- les symboles et les transformations symboliques que subissent les percepts afin de génèrer ces symboles,
- les comportements et leurs algorithmes associés,
- les individus : position initiale, connaissance initiale, archétype de comportement,
- les algorithmes simulant les différents processus de perception (un pour chaque sens).

Le chapitre suivant décrira précisément l'instanciation réalisée pour la simulation d'évacuation de bâtiment en feu.

5.5 Architecture fonctionnelle

La figure 5.4 montre l'ensemble des composants de notre simulateur ainsi que les flux d'informations qui le parcourent. Le rôle de certains éléments mérite explication :

- La couche logique est dédiée aux règles du système, elle est chargée de dégager une sémantique de but situé à partir des symboles de la couche sous-jacente et de lui transmettre ces nouveaux buts situés.
- Le module de contrôle physique vérifie que l'application de la décision ne viole pas les règles physiques du monde dans lequel évolue l'agent. Il n'est en effet pas simple de s'assurer que les comportements qui ont été définis ne vont pas conduire à une aberration physique, typiquement deux agents voulant occuper une même place au même moment. Ce module est donc chargé de transformer une décision (position et orientation) en déplacement effectif de l'agent garantissant le respect des contraintes physiques. Concrètement notre implémentation produit une force physique qui est appliquée à l'agent et va entraîner son déplacement. En l'absence de contrainte physique, cette force est calculée de sorte que le déplacement de l'agent correspondra exactement à la décision. En présence de contraintes physique, l'ensemble des forces appliquées aux agents et aux éléments de l'environnement sera prise en compte par un moteur physique de résolution, qui calculera une situation d'équilibre du système, sous forme d'un ensemble de déplacements respectant les contraintes physiques.
- Le module de contrôle des performances permet un contrôle du processus de symbolisation. Par exemple, les problèmes de planification multi-critères nécessitent d'affecter des poids arbitraires aux contraintes. Ce module peut alors permettre un adaptation dynamique de ces poids.
- Les capteurs sont l'interface de perception de l'agent, lui permettant de produire les percepts de son environnement.
- Les effecteurs sont l'interface physique entre l'agent et son environnement, lui permettant d'agir sur celui-ci à travers la couche physique.

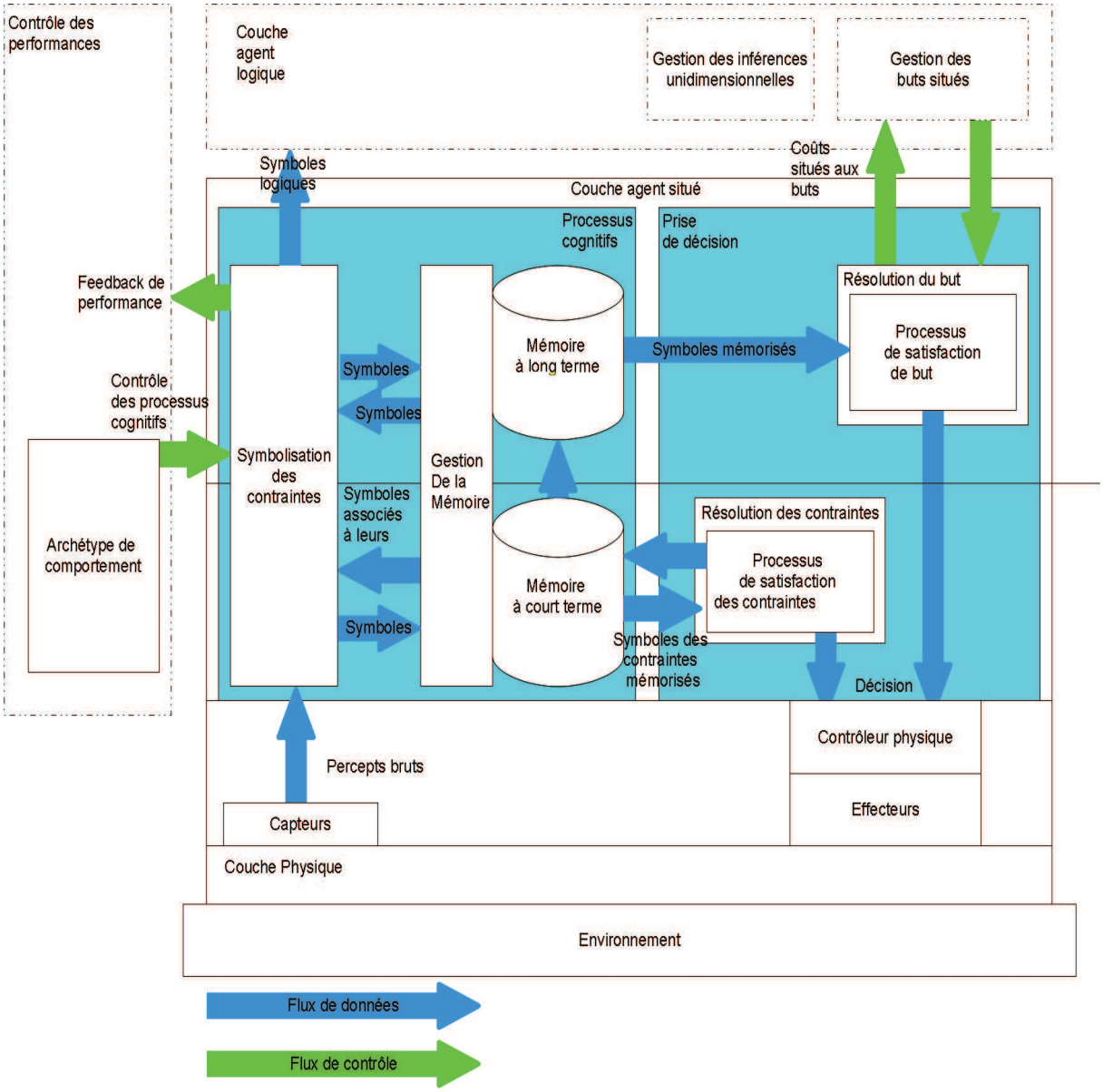


FIGURE 5.4 – Modèle fonctionnel situé hybride navigationnel/planifié proposé.

Chapitre 6

Instanciation de l'architecture microscopique proposée dans le cas de l'évacuation de bâtiment en feu

Nous allons maintenant définir les différents éléments mentionnés à la section 5.4.4 pour le cas d'un simulateur d'évacuation de bâtiment en feu basé sur le modèle proposé. Pour des raisons de clarté, nous ne suivrons pas exactement l'ordre de la section 5.4.4. Le chapitre suivant permettra de donner des détails techniques sur l'implémentation de ces éléments mais nous essaierons, tant que faire se peut, d'éviter les considérations trop techniques ici afin de nous concentrer sur l'instanciation des différents concepts.

6.1 Perception

Les différents percepts captables par les agents sont le feu, les murs du bâtiment, les autres agents, les signaux logiques de communication, la température, la fumée, quantifiée par son indice de toxicité, ainsi que les panneaux d'aide à l'évacuation. Parmi les 5 sens dont est doté l'homme, 3 sont utilisés et donc simulés :

- le toucher est simulé en sondant la cellule où se trouve l'agent,
- l'ouïe est simulée en détectant la présence de l'agent dans un cercle centré autour d'un autre agent, matérialisant la portée de la voix de cet agent,
- la vision est simulée par lancer de rayon, avec prise en compte de l'opacité due à la fumée selon la formule classique :

$$\alpha_{obs} = \int_0^L \alpha(s)ds \tag{6.1}$$

avec α_{obs} l'opacité cumulée le long du rayon, L la longueur du rayon et $\alpha(s)$ l'opacité locale à la distance s de l'origine du rayon. Les rayons sont arrêtés lorsque l'opacité cumulée atteint un certain seuil.

6.2 Symbolisation

Nous détaillons ici le processus de symbolisation, en distinguant la symbolisation des percepts captés par le système sensoriel de l'agent, les symbolisations appliqués aux symboles produits et enfin le mécanisme d'inhibition de ces processus de symbolisation.

Sens	Percept	Contrainte	Comportement
Vue	*Murs*	*Spatiale*	−
Vue	*Feu*	*Spatiale*	−
Vue	*Fumée*	*Spatiale*	−
Toucher	*Température*	*Chaleur*	−
Odorat	*Fumée*	*Toxicité*	−
Vue	*Autres agents*	*Densité*	−
Vue	*Panneaux d'aide*	*Aide*	+

TABLE 6.1 – Les percepts, le sens permettant leur détection, les contraintes navigationnelles associées et le signe du comportement résultant : + pour une force attractive, - pour une force répulsive.

6.2.1 Symbolisation des percepts

Pour le problème de l'évacuation, les principaux symboles associés aux contraintes sont :
- C_{Mur} : les obstacles perçus, issus du percept de murs,
- C_{Feu} : les obstacles perçus, issus du percept de feu,
- C_{Fume} : les obstacles perçus, issus du percept de fumée,
- C_{Temp} : la température perçue, issue du percept de température,
- C_{Tox} : la toxicité perçue, issue du percept de fumée,
- C_{Vital} : le respect des espaces vitaux des congénères, issu des percepts d'agents,
- $C_{Panneau}$: l'aide perçue, issue des percepts de panneaux d'aide à l'évacuation.

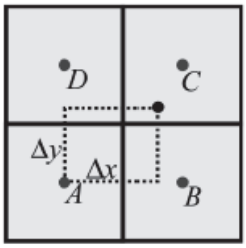
On remarque que certains éléments donnent lieu à plusieurs percepts, notre concept de percept étant l'association entre un sens de l'individu et un élément. Par exemple, la fumée est à la fois perçue visuellement et olfactivement. La table 6.1 donne la liste complète des informations perçues, leur mode de perception ainsi que le signe du symbole, qui pourra être attractif ou répulsif. L'influence du symbole sera proportionnelle à la distance de l'agent à l'élément associé, qu'il s'agisse d'une contrainte spatiale - force répulsive dont l'intensité est inversement proportionnelle à la distance - ou d'une contrainte de type toxicité où la toxicité accumulée dans l'organisme par unité de temps sera inversement proportionnelle à la distance et proportionnelle à la densité de fumée. A noter que dans ce dernier cas, nous n'avons pas intégré l'indice de toxicité de la fumée, selon le matériau composant les éléments en train de se consumer.

6.2.2 Autres symbolisations

Le processus de symbolisation ne concerne pas que les percepts et nous avons écrit qu'il pouvait consister à créer un contexte de satisfaction du problème plus complet (voir section 5.4.1). C'est notamment le cas lors du passage de la mémoire à court terme vers la mémoire à long terme, c'est-à-dire quand les symboles, qui seront utilisés par les comportement navigationnels, doivent être exportés vers un format unifié afin de pouvoir être utilisés par un comportement planifié. Nous avons écrit qu'au sein de la mémoire à court terme, les symboles étaient représentés sous forme vectorielle alors que dans la mémoire à long terme ils étaient représentés sous forme de grilles régulières. Il est donc nécessaire de leur appliquer une symbolisation qui implémentera ce changement de représentation, concrètement un échantillonnage d'une fonction continue selon une grille régulière. A noter qu'il eût été possible d'opter pour une autre structure, comme par exemple une surface triangulée, mais d'une part certaines contraintes sont dès le départ sous la forme d'une grille régulière, telle la densité de la fumée par exemple, et d'autre part l'espace déambulable au sein d'un bâtiment est le plus souvent plat, ce qui ne justifie pas le recours à une surface triangulée, sachant par ailleurs que les algorithmes travaillant

sur des grilles sont en général plus efficaces et plus facilement parallélisables. Nous avons choisi une algorithmique compatible avec ce type de structure discrète que sont les grilles régulières, de la classe des Fast Marching Methods [Sethian 1999].

Un exemple est la densité de population, associée à la contrainte d'espace vital, qui obligera tout agent à éviter le contact physique avec les autres agents. De la position des différents agents à un instant donné, représentée dans la mémoire à court terme par une position et un rayon correspondant à l'encombrement de l'agent, nous calculons une carte de densité de la population donnant en tout point de l'espace déambulable la densité en individus, la valeur 1 pour une case de la grille signifiant que toute la surface de la case est occupée par un ou plusieurs agents. Pour la prise en compte de l'encombrement des agents eux-mêmes, nous utilisons la méthode de conversion de densité décrite dans [Treuille 2006], qui assure continuité de la fonction de densité et évite que les agents soient impactés par leur propre encombrement, condition indispensable afin d'obtenir des comportements réalistes. Elle s'exprime ainsi :



$$\begin{aligned} \rho_A &= min(1-\delta x, 1-\delta y)^\lambda \\ \rho_B &= min(\delta x, 1-\delta y)^\lambda \\ \rho_C &= min(1-\delta x, \delta y)^\lambda \\ \rho_D &= min(\delta x, \delta y)^\lambda \end{aligned}$$

FIGURE 6.1 – Calcul de l'encombrement physique d'un agent.

avec ρ_A, ρ_B, ρ_C et ρ_D la contribution de l'individu à chacune des cellules 4-voisines, lui se trouvant dans celle en haut à droite (voir figure 6.1) et *lambda* un facteur permettant de définir le rapport entre l'encombrement des individus et la taille d'une cellule.

Le comportement de planification travaillant sur l'ensemble des contraintes, nous devons combiner toutes les contraintes afin de générer une représentation unique sur laquelle l'algorithme de comportement va travailler. C'est pour faciliter cette ultime symbolisation que tous les symboles au niveau planifié sont représentés de façon identique, à savoir une grille régulière de résolution fixée au départ, typiquement la résolution dans laquelle a été échantillonné l'environnement. L'ultime symbolisation va consister à calculer une grille unique comme combinaison linéaire de toutes les autres grilles. Les coefficients appliqués aux différents symboles (les grilles) sont propres à chaque agent et stockés dans son archétype comportemental.

6.2.3 Contrôle de la symbolisation

Comme nous l'avons proposé au chapitre précédent, le processus de symbolisation de chacune des contraintes peut être inhibé. Pour le cas d'un individu en situation d'évacuation d'un bâtiment, il sera possible d'inhiber le niveau planifié afin de limiter la production de comportements au niveau navigationnel. L'individu continuera alors de naviguer correctement, en évitant les obstacles qui surviennent devant lui, mais sera incapable de raisonner et en particulier de replanifier son chemin. On crée ainsi un contexte de décision en accord avec les hypothèses de la rationalité limitée.
C'est ce mécanisme que nous avons retenu afin d'implémenter le comportement d'un individu en situation de panique, si important pour la compréhension des accidents en situation d'urgence. Nous proposons que la conséquence de l'état de panique soit une limitation des capacités cognitives de l'in-

dividu. Concrètement chaque individu est doté d'une jauge mesurant son degré de panique et lorsque le niveau de cette jauge atteint un certain seuil, l'individu est considéré comme sujet à la panique et le niveau cognitif/planifié est inhibé. Cette jauge est alimentée par différents évènements survenant dans l'environnement de l'individu : par exemple si un individu découvre que l'issue qu'il espérait atteindre ne peut plus être atteinte, alors le niveau de sa jauge va augmenter.

Nous n'avons pas considéré la possibilité d'inhiber les comportements navigationnels car il nous semble peu réaliste qu'un individu fonce dans un mur, même sous l'emprise d'un mouvement de panique. Par contre, on peut imaginer qu'il bouscule une autre personne afin de forcer son passage plutôt que de rechercher à la contourner en empruntant un autre couloir par exemple, autrement dit de replanifier. Si seul le niveau planifié est inhibé, l'agent va attendre derrière l'autre agent que celui-ci le laisse passer car il ne va pas replanifier son chemin. Si au contraire la contrainte d'espace vital est inhibée, il pourra forcer son passage. Pour rendre ce comportement possible, il est nécessaire d'affiner le mécanisme d'inhibition en prenant en compte l'impact de la panique au niveau de chaque contrainte et non au niveau de l'ensemble des contraintes. C'est possible à condition de mémoriser, au sein de l'archétype comportemental de chaque agent, le niveau de la jauge au delà duquel une contrainte est inhibée et ce pour chaque contrainte indépendamment.

6.3 Caractérisation des buts

Trois catégories de buts complémentaires peuvent être distingués :

1. Se rendre à une sortie connue et pouvant être atteinte. Le but est alors la sortie et on lui attribue la valeur 0 car son atteinte correspond à la satisfaction complète du problème puisque l'agent a évacué le bâtiment.
2. Suivre une personne qui déclare connaître une issue. Le but est alors de rallier l'endroit où se situe cette personne.
3. Rechercher les bords du bâtiment ou un escalier permettant de descendre, si on sait que les sorties se trouvent au rez-de-chaussée.

La numérotation implique une hiérarchisation, qui doit amener l'agent à privilégier les buts de la catégorie 1, puis ceux de la catégorie 2 et enfin ceux de la catégorie 3. A chaque catégorie nous assignons une valeur numérique fonction de l'importance relative des buts de cette catégorie par rapport à ceux des autres catégories : 0 pour la catégorie 1, des valeurs strictement positives pour les autres, avec une valeur faible pour la catégorie 2 comparée à la catégorie 3. Ces valeurs seront injectées dans l'équation mathématique qui intégrera également le coût du déplacement pour atteindre le but. Ainsi l'agent pourra choisir un but de la catégorie 2 même s'il existe un but dans la catégorie 1. Un individu peut par exemple, bien qu'ayant connaissance d'une issue de secours, préférer suivre un autre individu qui lui indiquerait connaître un raccourci.

En l'absence de sortie accessible connue, nous considérons qu'un individu va malgré tout raisonner afin de se diriger vers ce qui peut représenter une issue potentielle. Autrement dit, un individu non soumis à la panique ne fait jamais n'importe quoi : il raisonne afin de tenter d'évacuer le bâtiment. Par conséquent il y a toujours existence d'un but et donc planification, notre système étant par conséquent inductif.

Une issue d'un bâtiment est nécessairement une ouverture vers l'extérieur et donc, en l'absence de la connaissance précise d'une telle issue, toute cellule au bord du bâtiment ou dans un escalier descendant est considérée comme une issue potentielle. En l'absence de but, nous aurions uniquement des comportements navigationnels et nous pourrions observer des phénomènes d'oscillation, tels qu'un individu oscillant entre deux murs. Or nous considérons que, dans une situation d'évacuation de bâtiment en

feu par un individu non pris de panique, ce type de comportement n'est pas réaliste. Nous pensons qu'il ne peut survenir que comme la résultante d'un état de panique et nous l'avons donc uniquement autorisé dans cette situation.

Notons cependant qu'à tout moment au cours de ses déplacements, l'individu acquiert des informations via sa perception de l'environnement, ce qui l'amène à remettre en cause le but poursuivi, et à en changer lorsqu'un but de valeur supérieure lui apparaît. Un tel but peut être de même niveau - découverte d'une issue plus proche que celle qui était visée - ou d'un niveau supérieur - rencontre d'une personne ayant connaissance d'une issue par exemple. L'agent est ainsi en permanence engagé dans un processus exploratoire, au sens où il est à l'affût de tout but meilleur que celui qui est le sien actuellement. Cependant, dans la situation particulière où il n'a pas de but de niveau 1 ou 2, ce processus exploratoire prend alors tout son sens ou plus exactement le sens qu'on lui prête habituellement, à savoir la recherche d'une issue en l'absence de toute issue connue. A noter enfin qu'il n'y a aucun risque de comportement oscillatoire durant cette phase exploratoire puisque la connaissance liée aux obstacles n'est jamais remise en cause et donc lorsque l'agent se retrouve nez à nez avec un mur, il apprend et intègre désormais dans son comportement la présence d'un mur à cet endroit.

6.4 Coût des déplacements

Il ne peut y avoir de stratégie d'évacuation et donc de choix en l'absence d'un coût associé à l'atteinte d'un but. C'est ce qui fait choisir à un individu une sortie proche plutôt qu'une plus lointaine. Chaque contrainte va ainsi, suite au processus de symbolisation, être représentée par une grille dans la mémoire à long terme. Les différentes grilles vont ensuite être combinées, dans la mémoire à long terme de chaque agent, afin d'obtenir une grille unique qui représente l'ensemble des contraintes de déplacement appliquées à l'agent. Les coefficients de cette formule peuvent être différents d'un agent à l'autre, introduisant une possibilité, parmi d'autres, d'individualisation des comportements.
L'expression générale de notre fonction de coût est :

$$C = \frac{Inconfort}{1 + Confort} \tag{6.2}$$

avec C la fonction de coût, $Inconfort$ une influence négative, représentant une répulsion locale, et $Confort$ une influence positive, représentant une attraction locale.
Cette formule implique que tout ce qui gène le déplacement fait augmenter le coût de celui-ci alors que tout ce qui le facilite en diminue le coût.
Pour le problème de l'évacuation, nous définissons l'inconfort ainsi :

$$Inconfort = CDist + \sum_i fiab_i \times \alpha_{C_i} \times C_i \tag{6.3}$$

avec $CDist$ une constante symbolisant le coût de déplacement dans un espace non contraint, $fiab_i$ la fiabilité attribuée à l'observation du percept associé à la contrainte i par cet agent (voir équation (5.1)), C_i la valeur associée à la contrainte i, normalisée sur l'intervalle [0,1], et α_{C_i} le coefficient d'importance associé à la contrainte C_i pour cet agent (stocké dans l'archétype comportemental de l'agent).
En l'absence de contrainte, ne subsiste que $CDist$, la distance devenant alors le seul critère pertinent. Le confort est essentiellement constitué des aides à l'évacuation (panneaux de signalisation) et est défini ainsi :

$$Confort = \sum_i \beta_{A_i} \times A_i \tag{6.4}$$

avec A_i la valeur associée à l'aide A_i, normalisée sur l'intervalle [0,1], et β_{A_i} le coefficient d'importance associé à l'aide A_i pour cet agent.
Les jeux de coefficients α_{C_i} et β_{A_i} font partie de l'archétype de comportement et caractérisent chaque individu. C'est en cela que notre modèle est microscopique et qu'il permet de générer une large gamme de comportements.

6.5 Modèle physique de l'agent

La modélisation physique des agents repose sur un gabarit très simple, composé de 3 cercles attachés par des joints fixes, tel qu'illustré sur la figure 6.2. A noter que seule l'occupation au sol est prise en compte, ce qui conduit à ne modéliser que l'encombrement.

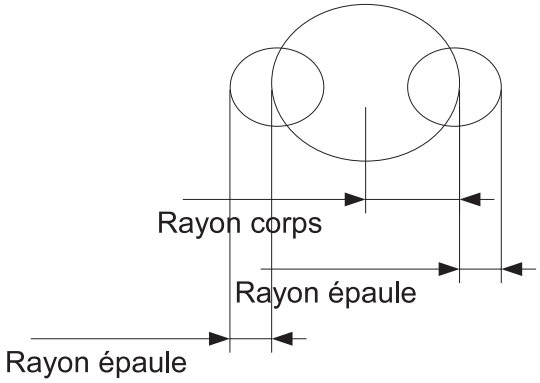


FIGURE 6.2 – Gabarit physique d'un agent

Deux modèles physiologiques sont intégrés dans notre simulateur :
- un modèle de résistance à la pression physique, basé sur le modèle de Fruin [Fruin 1993], consistant à définir la limite des forces pouvant être subies par un agent à un instant donné,
- un modèle toxicologique simple : à partir de la toxicité instantanée calculée par le simulateur FED (voir chapitre 7), une jauge cumule la toxicité reçue par chaque agent depuis le départ et lorsque celle-ci atteint une certaine limite, l'agent est considéré comme intoxiqué,
- un modèle de résistance à la chaleur : au delà d'un certain seuil, l'individu ne supporte plus la chaleur.

Lorsque l'une au moins de ces limites physiologiques est atteinte, l'agent tombe à terre et devient un obstacle.
A noter que si ces modèles sont plutôt simplistes, il est aisé de les affiner, par exemple en intégrant le niveau de toxicité dans le processus de symbolisation afin de conditionner la prise en compte de certaines informations par les comportements. Par exemple l'effet de l'intoxication pourrait consister à abaisser la limite du processus de symbolisation, ce qui aurait pour effet de limiter les moyens cognitifs de l'agent.

6.6 Archétype comportemental

L'archétype comportemental de chaque agent comprend les propriétés suivantes :

1. Propriétés physiques et physiologiques :
   - poids,
   - vitesse maximale,
   - accélération maximale,
   - seuil de résistance à la température,
   - seuil de résistance à la toxicité,
   - seuil de résistance à la pression physique.
2. Propriétés pseudo-cognitives :
   - Propriété de la perception :
     - distance maximale de vision.
   - Propriétés comportementales :
     - pondération cognitive de la contrainte de température,
     - pondération cognitive de la contrainte de toxicité reçue,
     - pondération cognitive de la contrainte de pression physique,
     - pondération cognitive de la contrainte d'espace vital,
     - pondération cognitive de la confiance aux signalisations,
     - pondération cognitive des différentes classes de buts.
   - Propriétés de rationalité limitée :
     - niveau d'abstraction maximal de chacune des contraintes,
     - carte de l'environnement connu,
     - temps maximaux de croyance des contraintes dynamiques.

Chapitre 7

Implémentation

7.1 Architecture de simulation

Nous travaillons sur la simulation de l'évacuation de bâtiment en feu dans le cadre d'une collaboration avec le CSTB [1] de Sophia Antipolis. Les modèles numériques développés par le CSTB sont des modèles au format IFC - *Industry Foundation Classes* - conçus à l'aide du logiciel Archicad [2] ou encore des modèles uniquement géométriques réalisés par des modeleurs tels que Blender. Les différents bâtiments sont visualisés à l'aide de la bibliothèque OpenSceneGraph [3]. Les scénarios d'incendie sont simulés grâce au logiciel de propagation de feu et de fumée FDS - *Fire Dynamics Simulator* [4] - mis à disposition par le NIST - *National Institute of Standards and Technology*. Ces outils représentent les contraintes technologiques de l'implémentation réalisée.

La figure 7.1 montre l'organisation globale des traitements et les flux de données associés, en distinguant les modules fournis par le partenaire industriel, celui qui est réalise par un logiciel tiers et ceux qui sont une contribution objet de cette thèse.

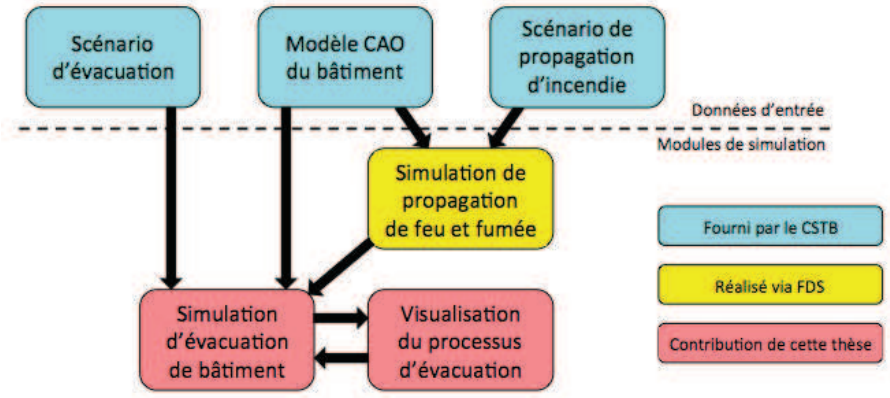


Figure 7.1 – Architecture générale de notre simulateur d'évacuation de bâtiment en feu.

7.2 Simulation de propagation d'incendie

La simulation d'incendie est un processus très coûteux, qui peut parfois nécessiter plusieurs jours de calcul. Elle s'effectue donc en amont de la simulation de l'évacuation.
Exécuter des simulations de scénarios d'incendie avec le logiciel de simulation FDS nécessite de convertir

1. Centre Scientifique et Technique du Bâtiment
2. http ://www.abvent.com/softwares/archicad
3. http ://www.openscenegraph.org
4. http ://www.fire.nist.gov/fds

la géométrie des bâtiments en voxels car FDS utilise une description volumique. Les maquettes du bâtiment sont d'abord épurées afin de n'en garder que la description géométrique. L'étape suivante consiste à introduire manuellement les méta-informations correspondant aux différents matériaux constituant le bâtiment. Techniquement, ces informations sont encodées dans l'attribut couleur des différentes primitives géométriques de la maquette épurée, à l'aide d'un modeleur 3D. La maquette subit alors une discrétisation en voxels à la résolution requise pour la simulation d'incendie. Un traitement consistant à fusionner les cellules adjacentes de même matériau est ensuite appliqué, afin de créer des pavés homogènes et ainsi réduire le nombre de cellules et donc la complexité de la simulation de propagation du feu. Enfin le fichier d'entrée de FDS est généré en associant à chaque cellule son matériau.

Le résultat de la simulation FDS est constitué des densités en feu et en fumée, de la température et de la toxicité, donnée en FED - *Fractional Effective Dose* [Speitel 1996] - pour chaque cellule à chaque pas de temps de la simulation.

7.3 Simulation d'évacuation

Chaque étage du bâtiment est représenté par plusieurs grilles régulières 2D :
- une grille statique des obstacles,
- une grille statique contenant les panneaux d'aide à l'évacuation,
- quatre grilles dynamiques extraites à partir des informations volumiques produites par FDS : toxicité, température, feu et fumée,
- une grille dynamique, réactualisée à chaque pas de temps de la simulation : la carte d'occupation de l'espace par les individus.

Ces grilles sont combinées avec des coefficients fournis dans l'archétype comportemental de chaque agent, afin de créer une grille unique qui constitue le symbole manipulé par le comportement planifié, qui est chargé de produire la décision de l'agent. Cette décision se matérialise par un déplacement souhaité, autrement dit un vecteur déplacement. Notre modèle étant microscopique, chaque individu raisonne au même instant de façon indépendante et rien ne garantit que les souhaits de déplacement des individus seront compatibles entre eux et en particulier qu'ils ne conduiront pas à des collisions entre individus. Il est donc nécessaire de valider ces déplacements afin de produire un ensemble de déplacements individuels compatibles entre eux et l'environnement, puisqu'il faut également éviter les collisions avec les obstacles tels que les murs. Pour ce faire, chaque déplacement souhaité est converti en une force à appliquer à l'agent qui, en l'absence de contrainte physique, fait se déplacer l'agent du déplacement souhaité durant le prochain pas de temps. L'ensemble des forces ainsi calculées est ensuite injecté dans le moteur physique Bullet [5], qui détermine un équilibre du système et produit un ensemble de déplacements valides. Le choix de ce moteur est justifié par son efficacité - complexité linéaire avec le nombre de vecteurs forces à traiter - et par le fait que la gestion des collisions n'était pas l'objet de nos recherches.
Puis le rendu réaliste de la simulation est effectué , en utilisant la bibliothèque d'animation d'avatars Cal3D [6] pour le calcul des postures des agents.

7.4 Prise de décision planifiée

Une fois les buts identifiés et caractérisés et la fonction de coût pour un agent calculée en toute cellule de la grille, la prise de décision se résume à trouver le plus court chemin menant l'agent au but

5. http ://www.bulletphysics.com
6. https ://gna.org/projects/cal3d

le plus proche au sens de la fonction de coût et de la caractérisation de ce but. La théorie des Level Sets [Sethian 1999] fournit une définition consistante du problème, qui a l'avantage de s'affranchir des contraintes de connexité, au sens topologique. Concrètement elle permet de calculer la distance de tout point à l'objectif le plus proche, qu'il y en ait un seul ou plusieurs, ce qui est adapté à notre problème puisque nous cherchons le chemin le plus proche à une issue du bâtiment, quelle qu'elle soit.

Partant de notre fonction de coût, stockée au sein de la grille unique issue du dernier processus de symbolisation (voir section 6.2.3), nous utilisons une variante de la Fast Marching Method [Sethian 1999] - FMM - afin de calculer la solution au problème. La FMM simule la propagation d'un front équidistant, qui recouvre progressivement tout le domaine en partant des objectifs, estampillant chaque cellule avec sa distance à l'objectif le plus proche. Le résultat est une carte des distances. L'intérêt est qu'une fois que ce calcul a été réalisé, la direction du plus court chemin à partir d'un point quelconque est fournie directement par le gradient de la carte de distances (voir figure 7.2). En utilisant un structure de données appropriée, on parvient à obtenir une complexité de la FMM en $O(N \log(N))$, N étant le nombre de cellules de la grille. Toutefois, afin d'obtenir des performances plus élevées, nous avons implémenté une version GPU de cette méthode, qui est décrite à la section 7.5.2.

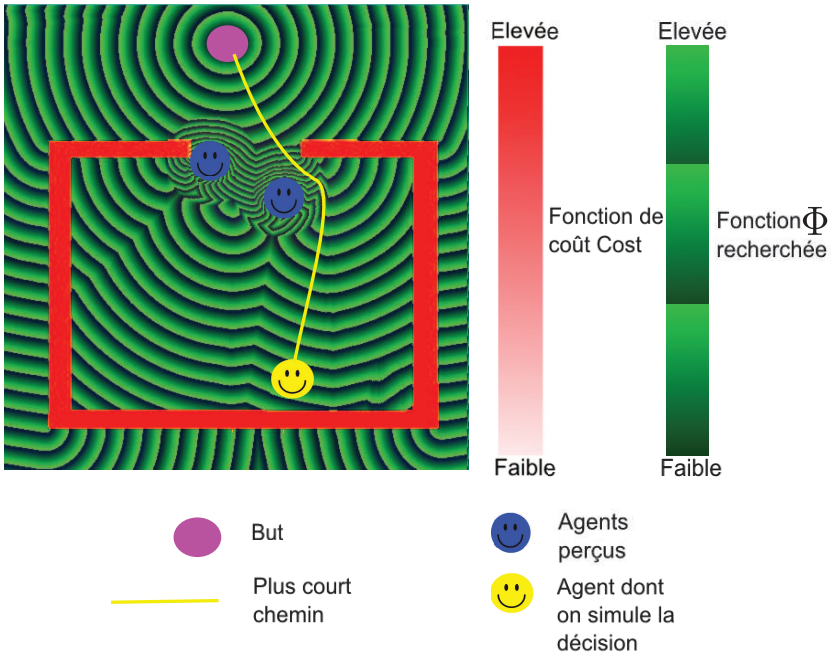


FIGURE 7.2 – Illustration de la Fast Marching Method : les vagues (en vert) représentent des iso-courbes de distance au but (en rose), matérialisant le champ de potentiel Phi. Le plus court chemin est alors obtenu en suivant l'inverse du gradient du champ Phi, autrement dit la plus forte pente de la fonction de distance. Pour des raisons de clarté, l'impact de la présence des deux autres agents (en bleu) sur la fonction de coût en rouge n'a pas été représenté.

7.5 Implémentation GPU

L'essentiel du coût en calcul de notre simulateur est dû à l'exécution de la FMM, aussi nous en proposons une implémentation sur GPU.

7.5.1 Processus de satisfaction de but

Le processus de satisfaction de but pour un agent s'effectue en plusieurs étapes :

1. Mise à jour des cellules buts G (processus de perception) : CPU,
2. Mise à jour de la grille de coût C (voir équation (6.2)) : CPU,
3. Calcul de la carte des distances (voir 7.5.2) : GPU,
4. Détermination du gradient puis de la vitesse de déplacement de l'agent (voir équation (7.1)) : GPU.

Ces différents traitements sont non seulement fortement parallélisables - traitement identique pour chaque cellule - mais doivent être effectués indépendamment pour chacun des agents.

7.5.2 Planification sur GPU

Nous décrivons d'abord notre implémentation de la Fast Iterative Method, variante de la FMM, sur GPU. Puis nous présentons le framework de résolution que nous avons conçu afin d'accélérer encore le calcul.

Algorithme 2 Fast Iterative Method

```
Phi : champ de potentiel
Pour chaque cellule i Faire {Initialiser le champ Φ}
  Si la cellule i appartient à un but Alors
    Φ[i] ← 0
  Sinon
    Φ[i] ← ∞
  Fin Si
Fin Pour
Répéter
  Stable ← Vrai
  Pour chaque cellule i Faire
    old ← Φ[i]
    Calculer Φ[i] {Réévaluation du potentiel}
    Si Φ[i] < old Alors
      Stable ← Faux
    Sinon
      Φ[i] ← old
    Fin Si
  Fin Pour
Jusqu'à Stable
```

7.5.2.1 Fast Iterative Method

La Fast Iterative Method [Jeong 2007] est une variante de la FMM particulièrement adaptée aux architectures de type GPU. Elle consiste à chaque itération à réévaluer la distance de toutes les cellules au front, par opposition à la FMM qui limite cette opération aux cellules 4-voisines du front. En conséquence la FIM n'a pas besoin de gérer une liste des cellules voisines mais effectue beaucoup plus d'opérations que la FMM. Par contre, déployée sur une architecture GPU capable de réaliser l'opération

en parallèle pour toutes les cellules en même temps, elle est plus efficace puisque la complexité au pire est en $O(N)$ contre $O(N \log(N))$ pour la FMM, le surcoût pour la FMM étant dû à la gestion du front. L'algorithme 2 est celui de la FIM [Manolios 2006].

D'un pas de temps au suivant durant la simulation, la connaissance sur laquelle raisonne un agent ne change pas beaucoup, sinon cela signifierait que le pas de temps est trop important et ne permet pas de capturer les évolutions avec suffisamment de précision. La conséquence est que la solution à la planification est elle aussi très ressemblante à celle calculée au pas de temps précédent. Afin d'exploiter cette cohérence temporelle, nous apportons deux modifications à l'algorithme de la FIM :

- on ne réinitialise plus le champ Φ à chaque pas de temps mais on repart avec la solution calculée au pas de temps précédent, ce qui permettra la plupart du temps de converger beaucoup plus rapidement,
- on modifie le test - ligne barrée dans l'algorithme - afin de valider systématiquement les changements de potentiel, y compris lorsque le potentiel augmente. Dans la version d'origine, le potentiel est systématiquement réinitialisé à l'infini et donc seules les diminutions de la distance aux objectifs sont validés. A partir du moment où nous repartons d'une solution existante, il est nécessaire d'envisager que la distance augmente. En effet, l'effort nécessaire pour atteindre un but peut augmenter, par exemple parce que l'agent vient de découvrir qu'un couloir qu'il comptait emprunter a été envahi par le feu. Tout changement de la distance est par conséquent validé.

L'algorithme 3 est notre algorithme de la FIM modifié pour les besoins de la simulation d'évacuation de bâtiment en feu.

Algorithme 3 Fast Iterative Method modifiée

Phi : champ de potentiel
Répéter
  $Stable \leftarrow Vrai$
  Pour chaque cellule i **Faire**
    $old \leftarrow \Phi[i]$
    Calculer $\Phi[i]$ {Réévaluation du potentiel}
    Si ~~$\Phi[i] < old$~~ $\Phi[i] \neq old$ **Alors**
      $Stable \leftarrow Faux$
    Sinon
      $\Phi[i] \leftarrow old$
    Fin Si
  Fin Pour
Jusqu'à *Stable*

Cet algorithme est implémenté en tant que Fragment Shader OpenGL, le champ de potentiel *Phi* étant stocké sur le GPU dans une mémoire de texture (Frame Buffer Object).
La boucle *Pour* de l'algorithme 3 disparait alors puisque chaque cellule de la grille est un fragment - ou texel - et que tous les fragments sont traités simultanément. Le test $\Phi[i] \neq old$ n'est pas implémenté au niveau de chaque fragment mais pour l'ensemble de la texture et consiste alors à tester si la texture a été modifiée par le traitement qui lui a été appliqué. La première implémentation de la FIM sur GPU que nous avons trouvée dans la littérature est basée sur la librairie CUDA fournie par NVidia et elle utilise une fonctionalité spécifique à cette librairie qu'est la réduction itérative [Jeong 2007]. Nous avons utilisé l'extension GLSL de OpenGL comme librairie afin de programmer le GPU et utilisé deux fonctionnalités matérielles accessibles depuis cette librairie afin d'implémenter le test de convergence :

le rejet de fragments du fragment shader et les requêtes d'occlusion [Manolios 2006].
Si le traitement ne modifie pas la texture, l'instruction *discard* est appelée dans le fragment shader et la géométrie "portant" la texture (GL_QUADS) est dessinée eu sein d'une requête d'occlusion. On peut ainsi déterminer le nombre de fragments dessinés et réécrire le test de convergence : $\#fragments \neq 0$.

La réévaluation du potentiel - ligne $Calculer\Phi[i]$ - est réalisée par un noyau d'interpolation quadratique permettant de déterminer un potentiel à partir des minimas/maximas des potentiels voisins dans les deux dimensions x et y.

Enfin nous devons souligner que cet algorithme est potentiellement sensible aux phénomènes d'oscillations puisque tout changement de potentiel est validé. Nous corrigeons cela grâce à l'introduction d'un ε_Φ proche de zéro et dépendant de la précision de l'implémentation. ε_Φ correspond alors à la plus petite différence perceptible avec la précision implémentée. Un exemple pratique pour des textures 16 bits est d'utiliser $\varepsilon_\Phi = \frac{1}{2\times 255} + \varepsilon$ avec ε proche de 0.

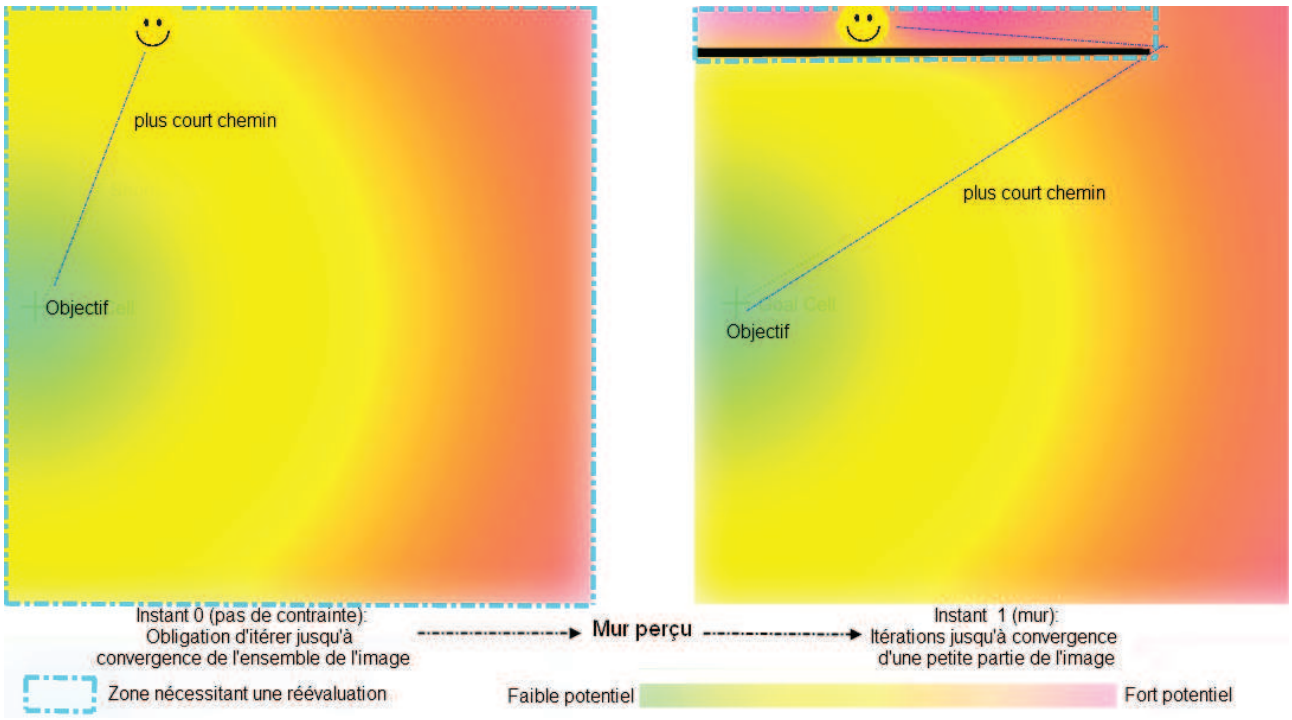


FIGURE 7.3 – Impact de la réparation locale sur la fonction Φ.

Le principal intérêt de cet algorithme est qu'il permet la réparation locale de chemin, autrement dit il permet de ne pas repartir de 0 à chaque itération mais de corriger la solution en fonction des évolutions détectées par l'agent dans son environnement. La figure 7.3 met en évidence l'avantage de la réparation locale du champ Φ lors de l'augmentation locale de la fonction de coût :

- à droite, l'individu n'a aucune connaissance du bâtiment, toute la fonction Φ à l'instant 0 doit être reconstruite (carré en pointillé bleu),
- à gauche, l'individu a modifié sa fonction de coût en y ajoutant une zone de coût élevé symbolisant un obstacle. Les répercutions sur la fonction Φ de l'instant 0 sont alors très localisées (carré en pointillé bleu). Cette localisation des changements engendre alors très peu d'itérations de la FIM pour converger vers la nouvelle solution.

7.5.2.2 Framework de résolution

Faire converger le processus nécessite de multiples itérations du fragment shader de résolution. Le shader présenté effectue un calcul force brute à chaque itération et traite alors beaucoup de fragments dont les potentiels ne peuvent être déterminés à cette itération car l'un au moins de ses voisins n'est pas à jour (progression du front).

Pour remédier à ce problème, nous avons développé un framework de contrôle basé sur la subdivision du problème et les requêtes d'occlusion afin de déterminer la convergence locale de la résolution :

1. Le CPU détermine les sous-surfaces de rendu où des fragments sont potentiellement candidats à l'expansion du front de propagation, grâce à des requêtes d'occlusion,
2. Le GPU ne traite que les fragments dont le potentiel a évolué.

Ainsi un contrôle statistique de l'évolution itérative de la résolution peut être mis en oeuvre. Le pipeline graphique est alors utilisé comme un système de filtrage des traitements, comme illustré sur la figure 7.4.

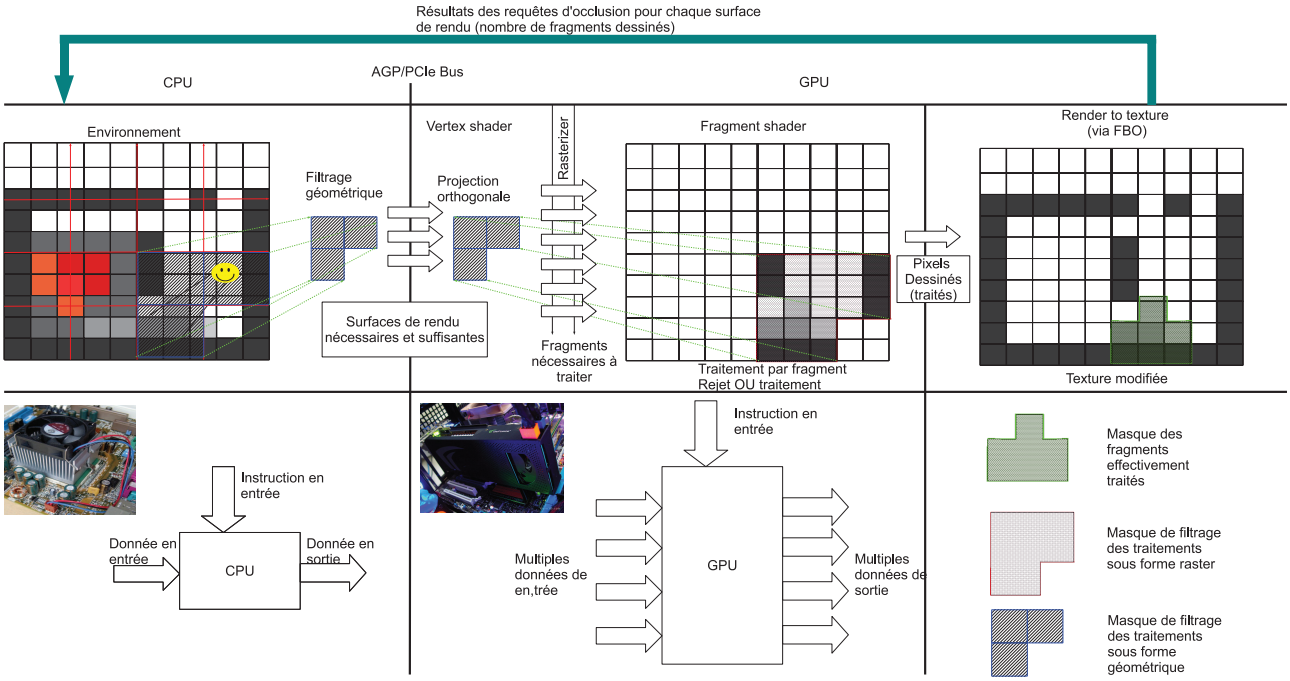


FIGURE 7.4 – Illustration du mécanisme de filtrage GPU mis en oeuvre.

Nous avons donc créé deux niveaux d'abstraction du problème :

1. les sous-surfaces, dont le rendu est contrôlé par la CPU via OpenGL,
2. les texels, rendus sans aucun contrôle par la GPU via GLSL.

A chaque itération, on ajoute des sous-surfaces de rendu à transmettre au GPU et chaque fragment de ces géométries est traité par le GPU.

Pour le cas de la FIM, chaque surface de rendu possède un drapeau de convergence mis à jour grâce au mécanisme de requête d'occlusion d'OpenGL : si le résultat d'une requête d'occlusion pour une surface est 0 fragment (convergence locale), toutes les surfaces voisines sont ajoutées à l'itération de rendu suivante.

Une partition de la surface de rendu en grille régulière de plus basse résolution que la texture qui lui

est attachée (voir figure 7.5) a été préféré à d'autres structures de subdivision, tel que le quadtree par exemple. Cependant, leur utilisation n'est pas à proscrire et pourrait faire l'objet de futurs travaux.

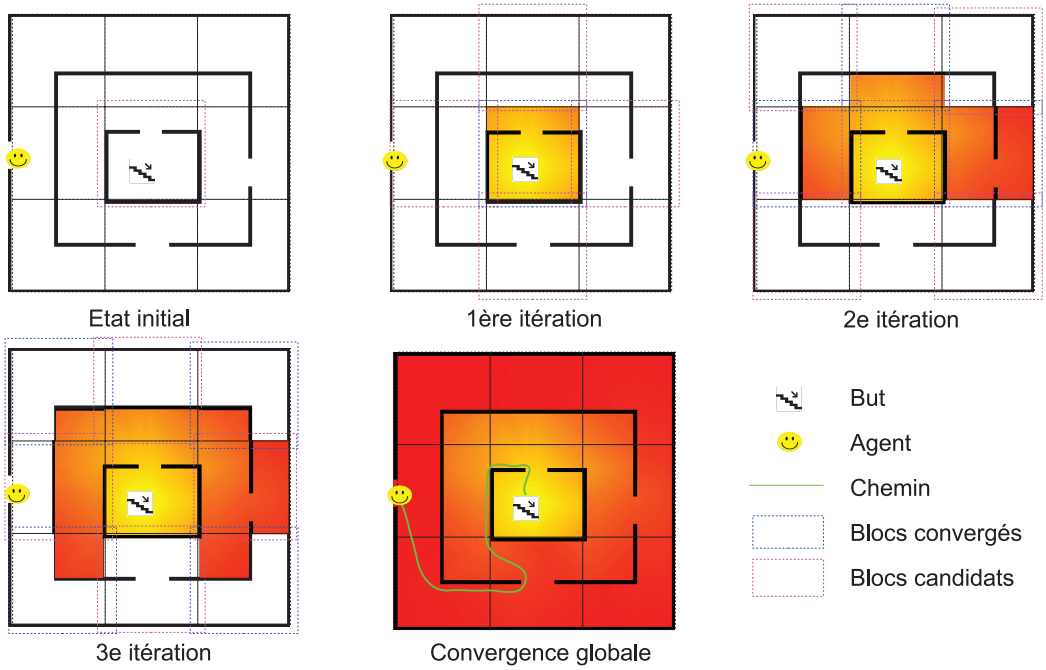


FIGURE 7.5 – Politique CPU de résolution de la Fast Iterative Method.

7.5.3 Calcul du champ de forces

Le moteur physique Bullet [7], que nous utilisons pour la résolution des collisions, admet en entrée un ensemble de corps physiques et les forces qui s'appliquent sur eux, pour calculer un état d'équilibre du système garantissant l'absence de collisions. Nous devons donc calculer la force à appliquer à chaque individu et nous procédons comme suit.
Une fois la carte des distances calculée, le calcul du gradient au centre des cellules est effectué par le GPU, ce qui évite de devoir rapatrier la texture en mémoire centrale.
Le gradient donne la direction du déplacement mais il nous faut de plus disposer de la vitesse de ce déplacement. Celle-ci serait donnée par la norme du gradient. Cependant, on peut admettre que dans une situation d'urgence, les individus adoptent systématiquement la vitesse maximale afin de se mettre hors de danger. Les seules limitations à la vitesse sont d'ordre physique : vitesse de course, présence d'obstacles qui obligent à ralentir, inertie, visibilité, voire toxicité si on voulait modéliser le fait qu'un manque d'oxygène dans l'air affecte les capacités respiratoires et donc d'effort des individus. Nous calculons la vitesse comme une fonction des contraintes qui l'affectent :

$$V(x) = \prod_i (1 - \beta_{C_i} C_i(x)) \tag{7.1}$$

avec $V(x)$ la vitesse à adopter dans la cellule x, $C_i(x)$ la valeur associée à la contrainte i dans la cellule x, normalisée sur l'intervalle [0,1] et β_{C_i} l'impact de la contrainte C_i sur la vitesse de déplacement.

7. http ://www.bulletphysics.com

La force résultante $\overrightarrow{F}$ est alors calculée ainsi :

$$\overrightarrow{F} = -\frac{\overrightarrow{\nabla\Phi(x)}}{||\overrightarrow{\nabla\Phi(x)}||}V_x \tag{7.2}$$

Chapitre 8

Résultats

Nous allons d'abord présenter, à travers quelques captures d'écran agrémentées d'explications, les principales possibilités offertes à l'utilisateur de notre simulateur afin de paramétrer la simulation. Puis nous allons, au travers d'un exemple d'une simulation d'évacuation de bâtiment en feu, pouvoir observer l'effet sur le comportement d'un individu de notre implémentation de la théorie de la rationalité limitée. Enfin nous allons donner quelques éléments permettant d'apprécier la gain obtenu par l'implémentation de plusieurs algorithmes sur GPU.

8.1 Utilisation du logiciel

Notre simulateur peut être paramétré via deux interfaces différentes :
- les fichiers de configuration de la simulation, qui sont lus par le simulateur et permettent d'initialiser la plupart des paramètres,
- l'interface graphique du logiciel, qui reprend tout ou partie de ces paramètres.

La configuration des scénarios d'évacuation est décrite dans un fichier XML, dont un exemple est donné en annexe 10. On y trouve notamment les fichiers de simulation au format FDS ainsi que le fichier contenant la description géométrique du bâtiment.

Nous allons maintenant passer en revue les fonctionnalités de notre logiciel selon une classification qui suit le processus de traitement, allant de la perception de son environnement par l'agent jusqu'à la prise décision et la mise en mouvement de l'avatar 3D qui le représente.

8.1.1 Perception

8.1.1.1 Distance de vision

Cette propriété de l'archétype comportemental contrôle jusqu'à quelle distance voit un agent.

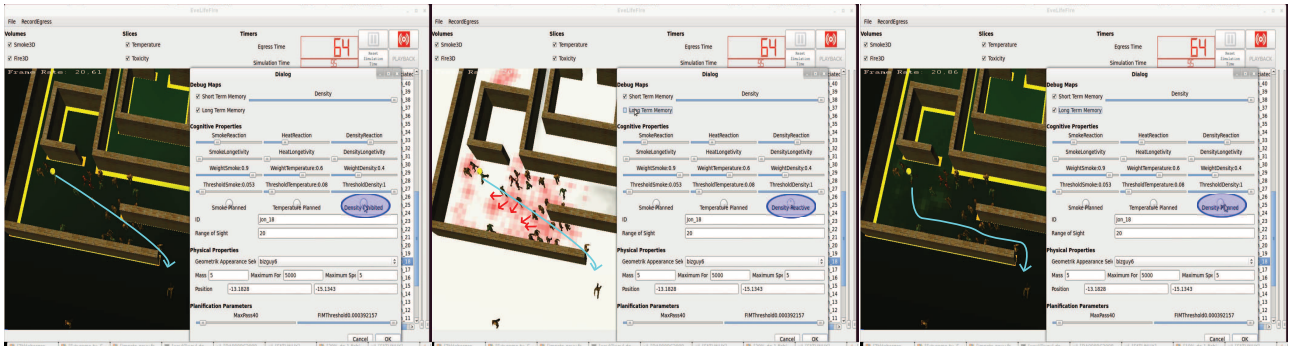
8.1.1.2 Connaissance initiale du bâtiment

Cette propriété permet de doter les agents d'hypothèses sur leur environnement avant le début de la simulation. Grâce à cette propriété, on peut par exemple décider de la connaissance partielle ou totale d'un évacuant.

8.1.2 Symbolisation des contraintes

8.1.2.1 Seuils de saturation des contraintes

Chaque contrainte possède un seuil de saturation au dessus duquel sa valeur est bloquée à son maximum. Cela permet par exemple d'empécher le déplacement vers un espace où la densité de fumée dépasse un certain seuil, simulant le fait que devant une fumée aussi épaisse, l'agent refusera de s'engager davantage.



contrainte de densité inhibée | prise en compte réactive de la contrainte de densité | prise en compte planifiée de la contrainte de densité

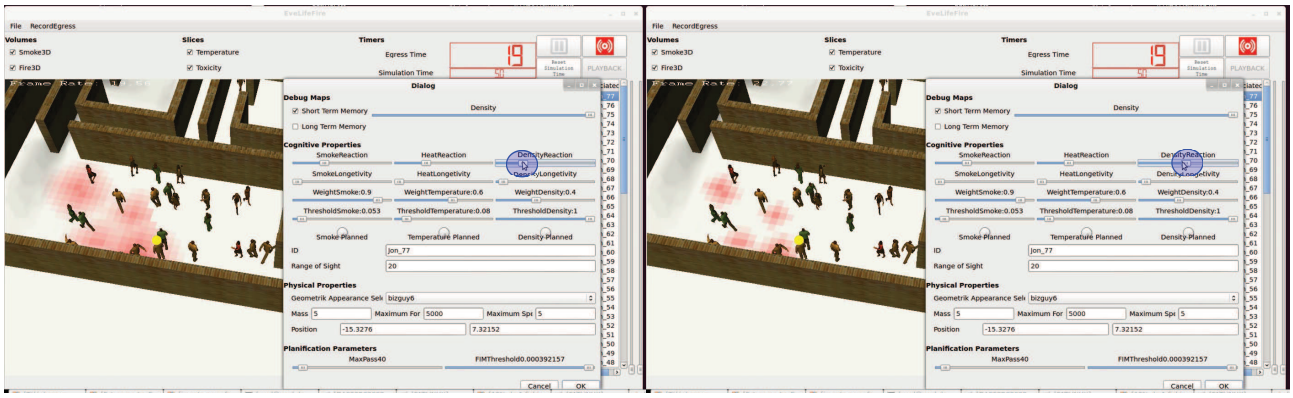
FIGURE 8.1 – Les trois niveaux de symbolisation de la contrainte de densité d'occupation et leur impact sur la prise de décision. Les flèches turquoises représentent la composante planifiée du comportement et les flèches rouges représentent des influences réactives locales. Dans le cas réactif (image du centre), ces contraintes vont faire dévier l'individu de sa trajectoire planifiée et celui-ci s'écartera donc de la trajectoire turquoise, sous l'effet des comportements réactifs résultant, lorsqu'il arrivera au contact des autres individus. Par contre, dans le cas planifié (image de droite), ces contraintes sont également symbolisées au niveau planifié et donc prises en compte dans le processus de planification, avec comme résultat leur prise en compte dans la définition de la trajectoire à emprunter, qui s'écarte donc des individus présents sur le chemin.

8.1.2.2 Niveaux de prise en compte des contraintes

La symbolisation de chaque contrainte est contrôlée par son niveau de prise en compte :

1. Inhibition : la contrainte n'est pas du tout prise en compte,
2. Prise en compte réactive : répulsion à la contrainte mais sans prise en compte dans le raisonnement,
3. Prise en compte rationnelle : intégration de la contrainte dans le processus de planification.

La figure 8.1 montre l'impact du niveau de symbolisation sur la trajectoire suivie par l'agent.



Amplitude de répulsion répulsion faible | Amplitude de répulsion répulsion forte

FIGURE 8.2 – Modification de l'amplitude de la répulsion associée à la contrainte de densité d'occupation.

8.1.2.3 Amplitude de répulsion des contraintes

Nous avons vu qu'une contrainte est l'objet de plusieurs symbolisations (voir section 5.4.2.1). L'amplitude de la réaction à la contrainte (voir figure 8.2) permet de quantifier les comportements qui lui sont associés, que ce soit pour sa prise en compte au niveau réactif - force de répulsion instantanée - ou lors du processus de planification.

8.1.2.4 Durée de vie d'une contrainte

Ce paramètre permet de contrôler la vitesse à laquelle la contrainte s'efface des cartes mentales des agents (voir figure 8.3). Ce paramètre permet de jouer sur la remise en cause des observations d'un agent.

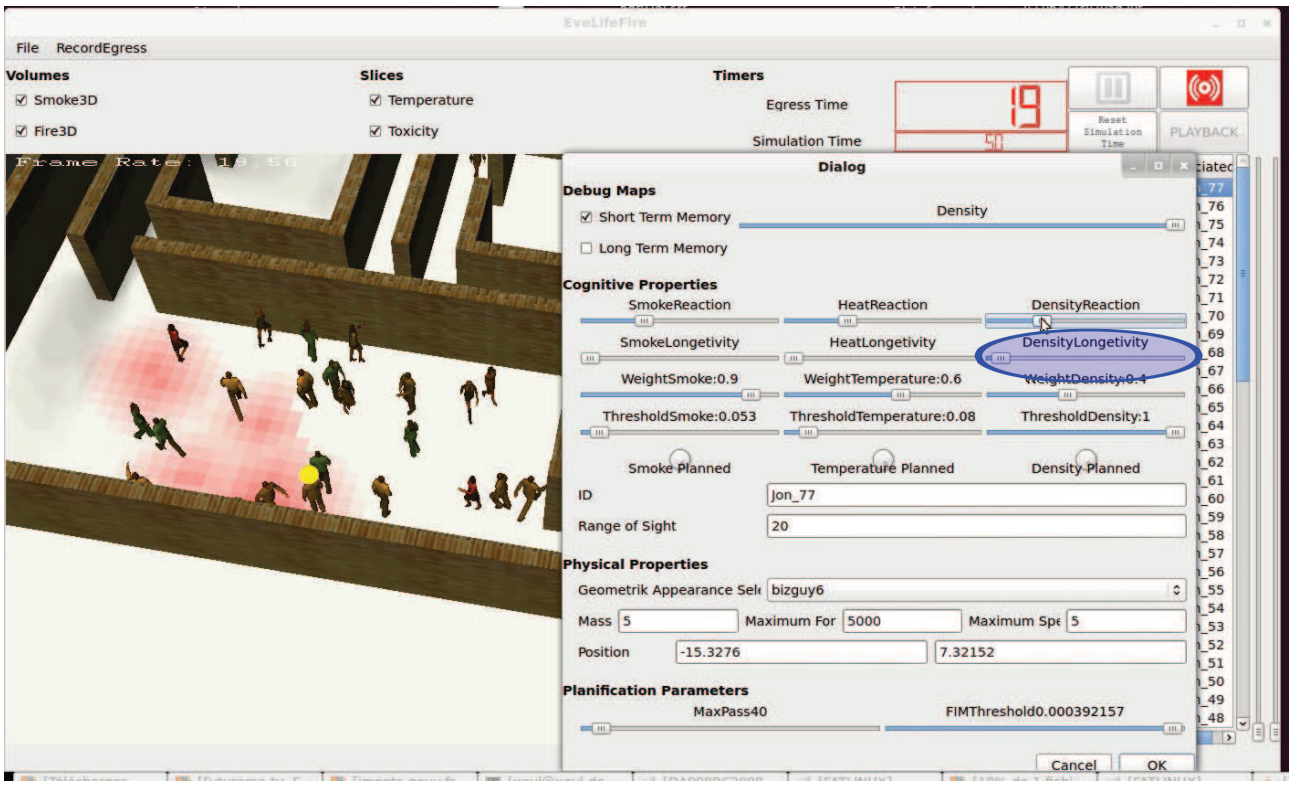
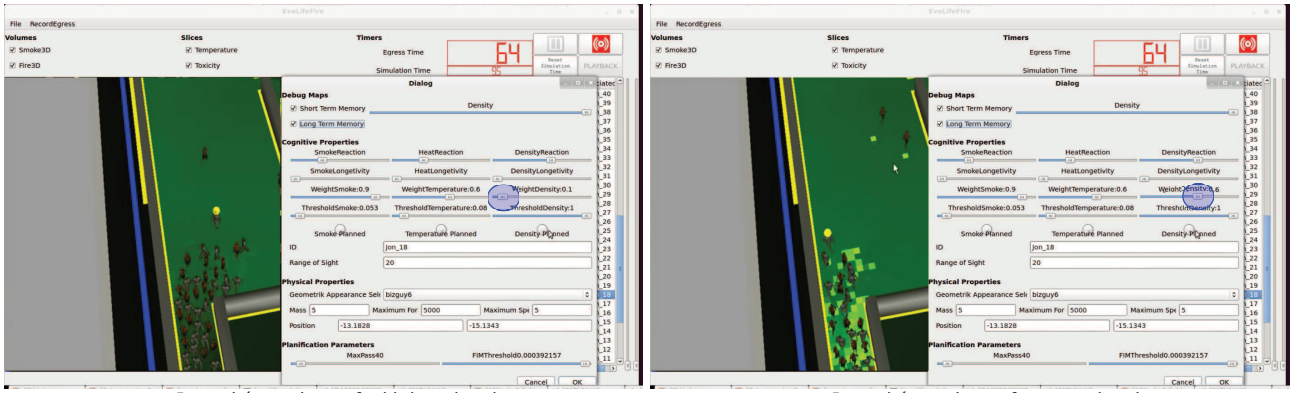


FIGURE 8.3 – Slider de modification de la durée de vie de la contrainte densité.

8.1.3 Prise de décision

8.1.3.1 Coefficients appliqués aux différentes contraintes

Nous avons vu que le raisonnement des agents est simulé par un algorithme de planification tendant à satisfaire l'objectif - sortir du bâtiment - en intégrant les différentes contraintes. Cet algorithme utilise une fonction de coût unique qui combine les symboles issus des contraintes en utilisant un jeu de coefficients propre à chaque agent et stocké dans son archétype comportemental (voir section 6.2.2). La figure 8.4 montre qu'il est possible de modifier ces coefficients interactivement via l'interface graphique du logiciel.



Pondération faible de la contrainte de densité

Pondération forte de la contrainte de densité

FIGURE 8.4 – Pondération de la contrainte de densité dans le raisonnement.

8.1.3.2 Contrôle de l'exécution

Deux paramètres supplémentaires permettent à l'utilisateur de contrôler la simulation (voir figure 8.5) :

- MaxPass : le nombre maximum d'itérations de la planification par pas de temps,
- FIMThreshold : le seuil de convergence de la FIM (le paramètre ε_Φ dans la section 7.5.2.1).

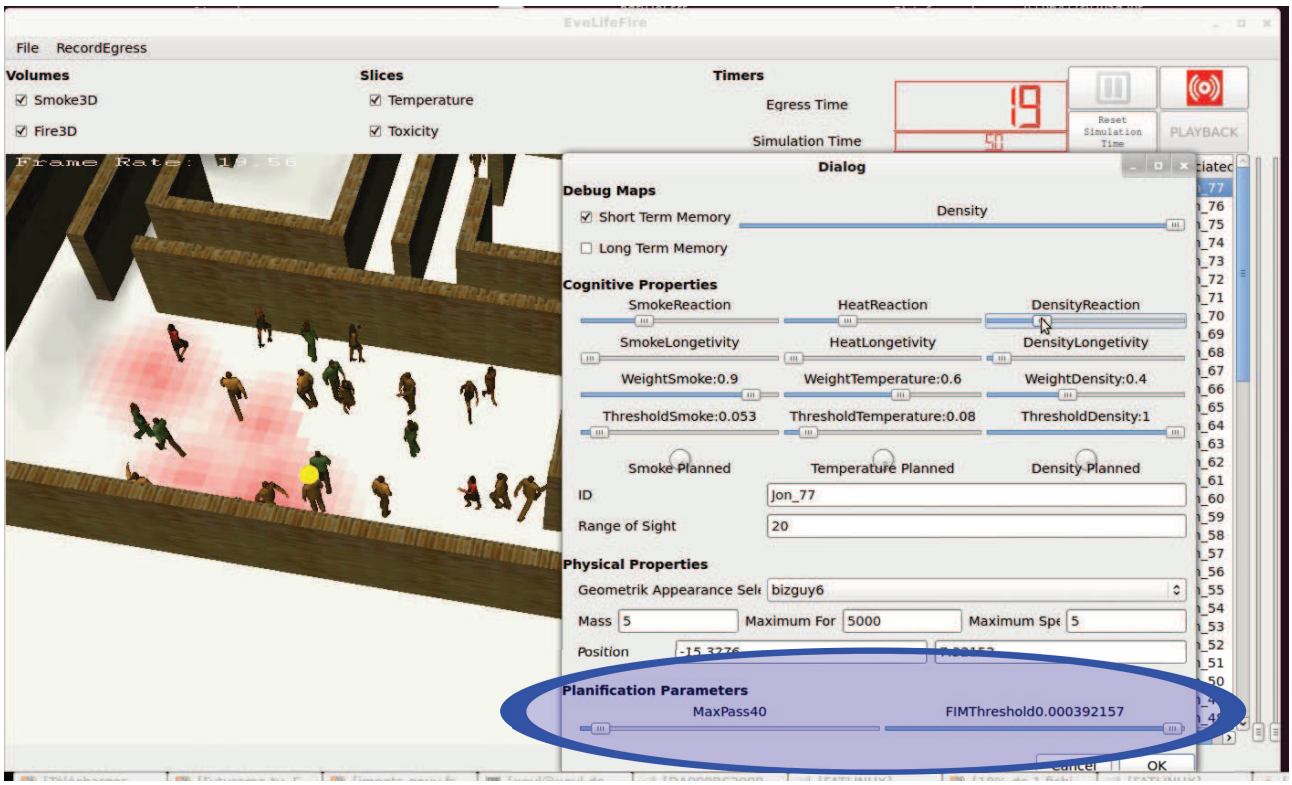


FIGURE 8.5 – Paramètres permettant de contrôler l'exécution.

8.1.4 Action

Nous avons vu que la décision ne reflète qu'un désir de mouvement de l'agent mais que le mouvement effectif dépendra des contraintes physiques qui s'appliquent, afin notamment d'éviter les collisions (voir section 7.5.3). Les propriétés physiques susceptibles de contraindre le déplacement sont :

- la masse de l'agent (en kg),
- la force maximale instantanée qu'il peut supporter (en N),
- la vitesse maximale à laquelle il peut se déplacer (en m/s).

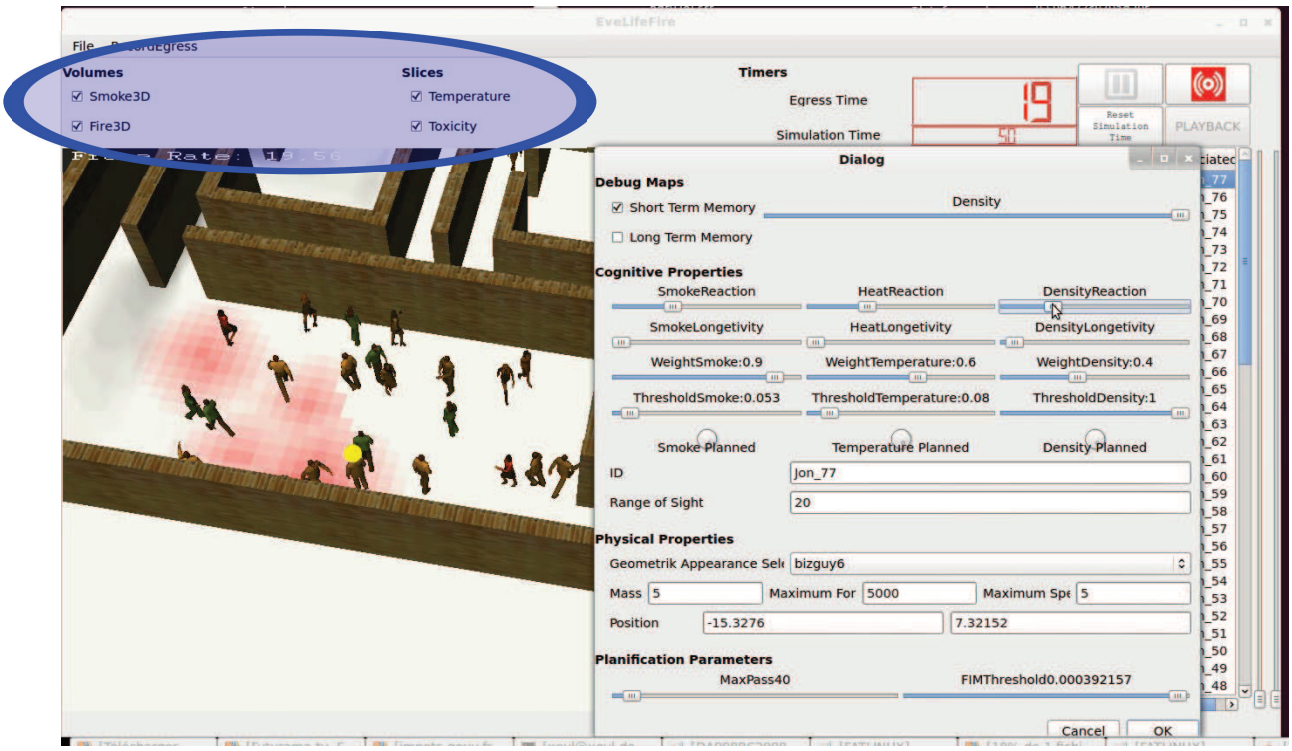


FIGURE 8.6 – Visualisation des phénomènes environnementaux.

8.1.5 Visualisation de l'évacuation

La visualisation de l'évacuation met en jeu plusieurs cartes 2D et volumes 3D. L'utilisateur a la possibilité d'activer ou désactiver chaque carte ou volume à tout moment.

8.1.5.1 Visualisation des phénomènes environnementaux

L'affichage des phénomènes environnementaux, tels que la température, la toxicité, le feu et la fumée, est contrôlé par la partie en haut à droite de l'interface, tel que le montre la figure 8.6.

8.1.5.2 Visualisation des cartes mentales des agents

Une carte mentale représente l'état de la mémoire d'un agent. La mémoire à long terme de l'agent sélectionné est visualisable : elle représente la connaissance qu'a l'agent à un instant donnée de l'état d'une contrainte aux différents points de l'espace (voir figure 8.7). Un interacteur permet de sélectionner la contrainte à visualiser.

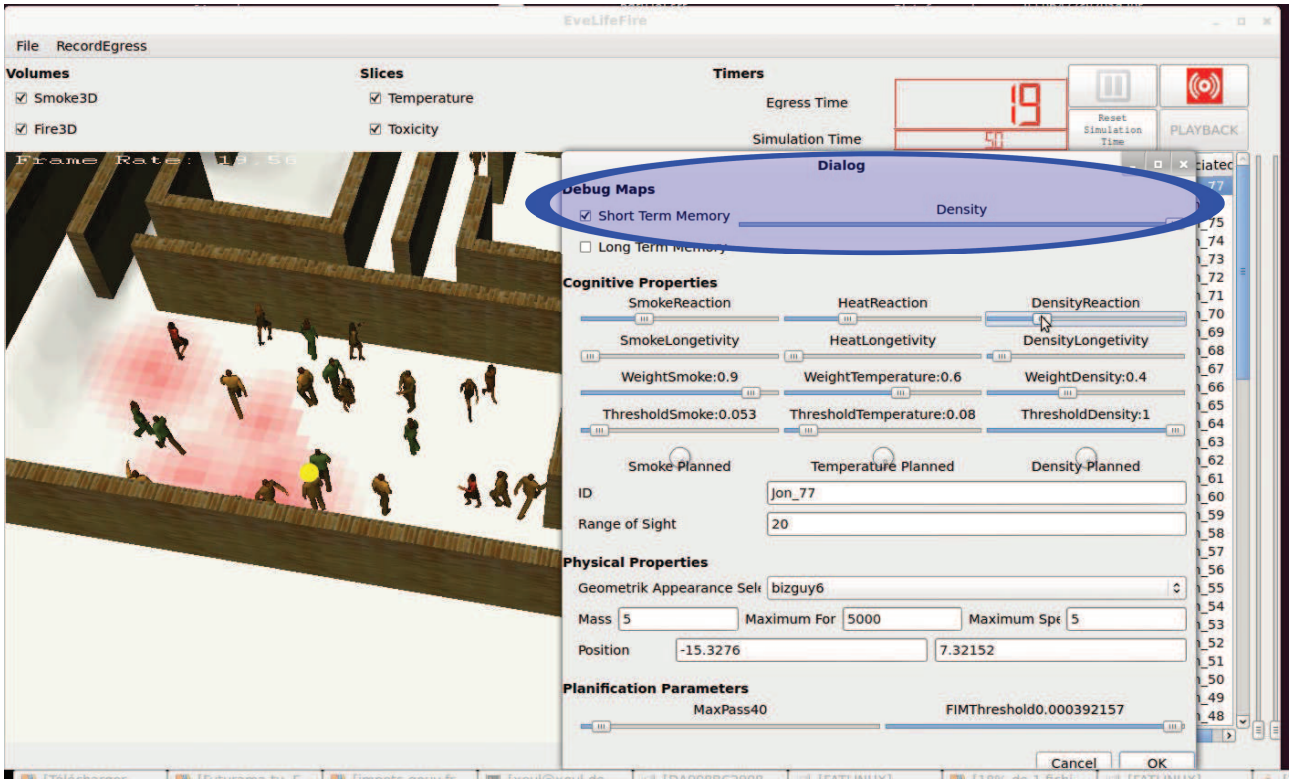


FIGURE 8.7 – Visualisation des cartes mentales de l'agent sélectionné. La contrainte est visualisée avec un mapping couleur allant du blanc (contrainte faible) au rouge (contrainte forte).

Le déroulement du processus de planification peut également être visualisé. La figure 8.8 montre la solution actuelle au problème d'évacuation. Le chemin le plus court menant de la position actuelle de l'agent au but est implicitement représentée en suivant le gradient négatif, c'est-à-dire lorsque l'agent se déplace systématiquement vers celle des cases voisines qui a le potentiel le plus faible.

8.2 Déroulement d'un scénario d'évacuation

Les figures 8.9 et 8.10 vont nous permettre de voir l'exécution d'un scénario d'évacuation en entier et d'apprécier notre implémentation de la rationalité limitée. Le bâtiment est vu de dessus, ses cloisons étant représentées en marron. Le sol du bâtiment est colorié avec un dégradé du bleu au vert représentant le champ de potentiel, au sens de la Fast Marching Method, ou pour faire plus simple la distance à l'issue la plus proche. Le feu est visualisé en rouge et la fumée en noir.
Afin de rendre les visualisations plus simples à analyser, nous avons joué une même simulation à plusieurs reprises, en ne modifiant que l'archétype d'un seul individu, matérialisé en jaune sur les figures. Chaque image correspond à un pas de temps particulier de la simulation, les images alignées verticalement correspondant au même pas de temps.
L'implémentation de la théorie de la rationalité limitée que nous avons proposée a conduit à considérer trois niveaux de prise en compte des contraintes :

– le niveau planifié, dans lequel toute contrainte est symbolisée jusqu'au niveau le plus haut et impacte donc à la fois le niveau planifié, par la prise en compte de la contrainte dans la planification de l'agent, et le niveau réactif, par la production d'un comportement réactif en réponse à

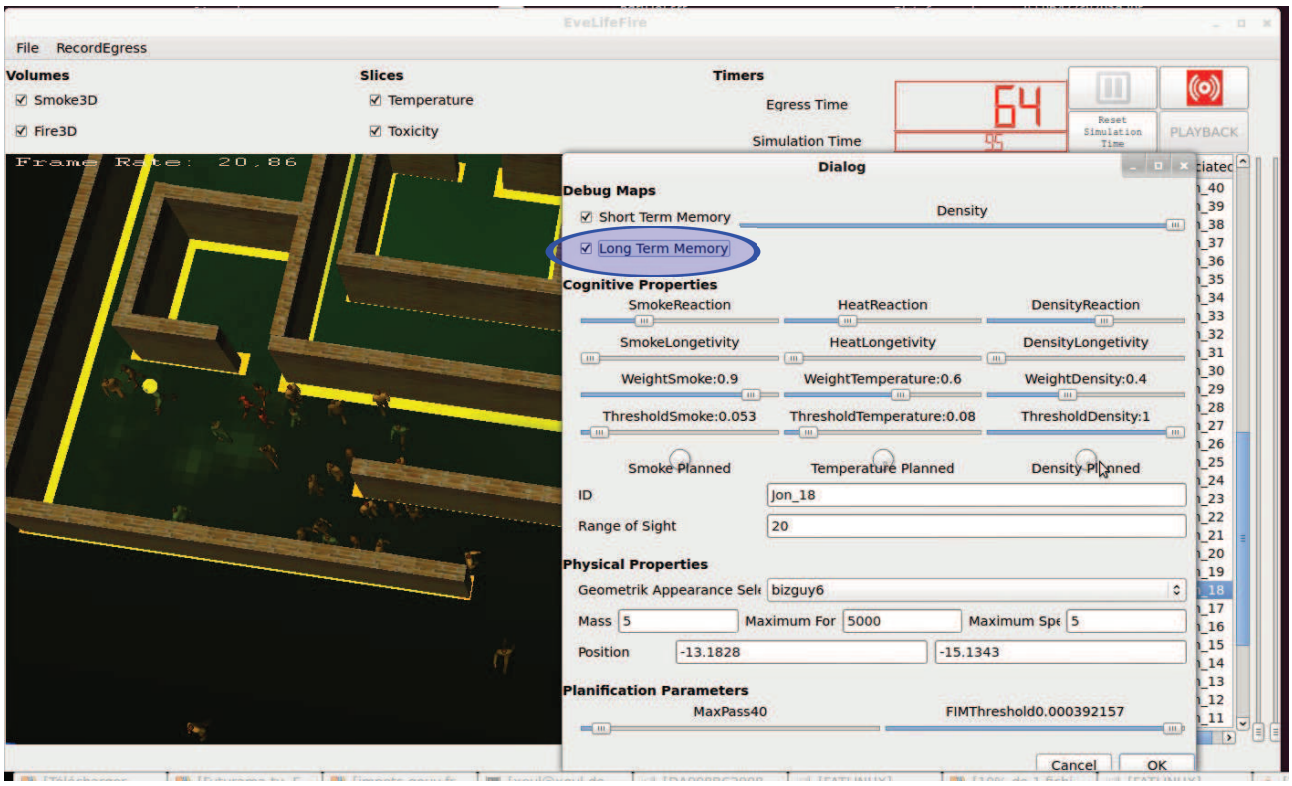


FIGURE 8.8 – Visualisation de la planification de l'agent sélectionné. Le champ de potentiel est visualisé avec un mapping couleur allant du noir (potentiel faible) au vert (potentiel fort).

la contrainte,

- le niveau réactif, dans lequel la contrainte n'est prise en compte que pour la production d'un comportement réactif mais n'est pas symbolisée au niveau planifié et dont les modifications n'ont par conséquent aucun impact sur la replanification éventuelle,
- le niveau inhibé, où les modifications de la contrainte ne sont pas du tout symbolisées et ne sont par conséquent pas du tout prises en compte par l'individu.

Les 3 lignes d'images qui sont présentées sur chaque figure correspondent respectivement aux 3 niveaux de prise en compte de la contrainte considérée, à savoir le niveau planifié en haut, le niveau réactif au milieu et le niveau inhibé en bas.

Dans notre implémentation le niveau de prise en compte peut être différent d'une contrainte à l'autre, ce qui nous a conduit, pour cet exercice, à présenter 2 séries d'images :

- sur la figure 8.9, c'est la contrainte *toxicité perçue* qui fait l'objet de niveaux de prise en compte différents,
- sur la figure 8.10, c'est la contrainte *densité de la foule*.

Commentaire de la figure 8.9 :

Colonne 0 (pas de temps 0) : Au pas de temps 0, les 2 issues du bâtiment sont accessibles et l'individu choisit donc la plus proche, qui se trouve être celle qui est vers la haut de l'image.

Colonne 1 (pas de temps 2) : Lorsqu'il aperçoit de la fumée au bout du couloir, 3 situations différentes se produisent selon le niveau de prise en compte de la contrainte. Au niveau planifié, la contrainte est symbolisée à tous les niveaux et donc prise en compte à tous les niveaux : l'individu replanifie

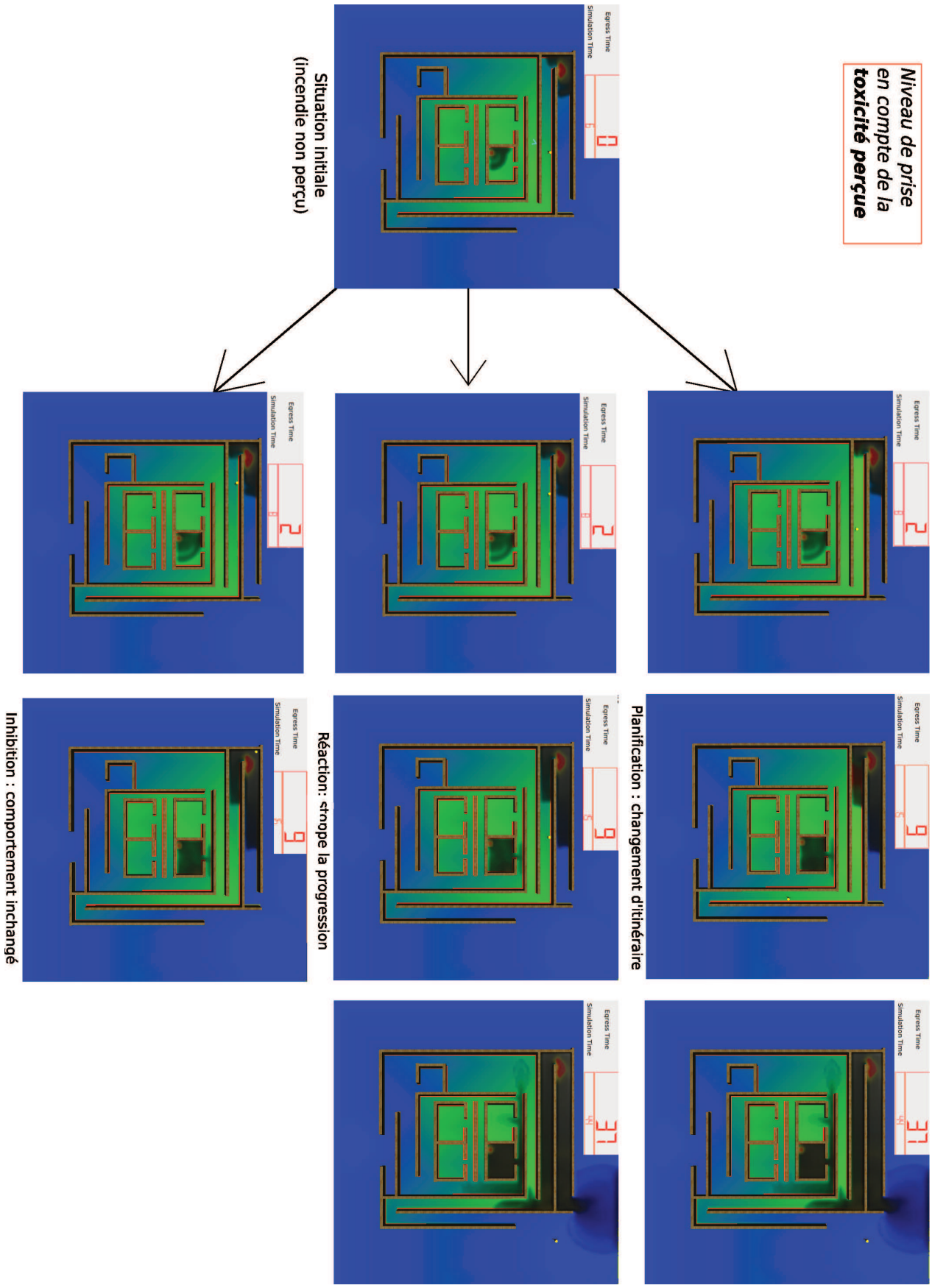


FIGURE 8.9 – Quelques captures de la fénêtre principale de notre simulateur, montrant un scénario dévacuation de bâtiment en feu et l'impact de notre implémentation de la rationalité limitée sur le comportement d'un individu (en jaune). Seule la contrainte *toxicité perçue* est concernée par l'application des axiomes de la rationalité limitée.

par conséquent et décide de rebrousser chemin. Au niveau réactif, la contrainte n'est symbolisée qu'à ce niveau mais pas au niveau planifié, ce qui a pour effet sa prise en compte pour le comportement réactif mais pas pour la planification. L'agent ne remet donc pas en cause sa décision d'aller en direction du feu mais est soumis à son comportement réactif qui lui intime de ne pas s'enfoncer dans le nuage de fumée : il stoppe donc sa progression. Au niveau inhibé, aucune symbolisation n'a lieu et la nouvelle information est tout simplement ignorée, l'agent continuant alors à avancer en direction du feu puisqu'il ignore celui-ci et n'hésite donc pas à s'en approcher dangereusement.

Colonne 2 (pas de temps 9) : Quelques instants plus tard, on constate que dans le cas où toutes les symbolisations sont autorisées, l'individu se dirige à sa vitesse de croisière vers la seconde sortie : il a donc pris de l'avance sur la fumée, qui progresse à un rythme plus lent. Dans le cas où le niveau planifié est inhibé, l'individu se tient toujours à distance respectable de la fumée car le comportement réactif lui intime l'ordre de se tenir à distance de sécurité, mais il n'a toujours pas remis en cause sa décision initiale et il continue de n'envisager que l'issue sélectionnée au départ. Enfin dans le cas où toute symbolisation est inhibée, l'agent s'est engagé dans la fumée comme si de rien n'était et a été carbonisé.

Colonne 3 (pas de temps 37) : On constate que dans les cas planifié et réactif, l'individu a évacué le bâtiment. Cependant il a évacué pour des raisons radicalement différentes. Dans le cas planifié l'individu, constatant que l'issue envisagée initialement n'était plus accessible du fait de la présence du feu, a recherché une autre issue et s'y est délibéremmment dirigé. Dans le cas réactif, l'individu n'a pas remis en cause le choix initial mais la subsistance d'un comportement réactif lui a fait éviter le contact avec le feu, celui-ci l'ayant repoussé jusqu'à le pousser hors du bâtiment. Même si le résultat final est identique, au temps d'évacuation prêt, c'est en quelque sorte une coïncidence ou plus exactement c'est la géométrie de ce bâtiment qui conduit à ce résultat. Mais il n'est pas difficile d'imaginer une configuration de bâtiment dans laquelle l'individu serait repoussé par le feu dans une voie sans issue et se retrouverait prisonnier des flammes. Il est important de bien comprendre cette différence de comportement. Dans le cas où la contrainte n'est pas prise en compte au niveau réactif, l'absence de comportement réactif au feu a pour effet que l'individu tente de se frayer un chemin à travers les flammes et finit par périr, d'où l'absence de la dernière image dans ce cas puisque l'individu est déjà décédé à l'image précédente.

Commentaire de la figure 8.10 :

Colonne 0 : Le pas de temps 0 correspond au moment où l'individu en jaune constate la présence de nombreuses personnes à l'entrée d'un couloir qu'il comptait emprunter pour accéder à l'issue du bâtiment.

Colonne 1 : Dans le cas planifié, on constate que l'agent a replanifié, ce qui l'a conduit à trouver un nouvel itinéraire consistant à passer de l'autre côté du mur. A l'inverse, dans le cas réactif, aucune replanification n'a eu lieu et l'information sur la densité de la foule n'a été prise en compte que pour la production d'un comportement réactif, qui va avoir pour effet de ralentir la progression de l'agent, celui-ci cherchant à respecter l'espace vital des autres agents. Enfin, dans le cas inhibé, l'agent continue sa progression comme si de rien n'était et on note que l'absence de prise en compte au niveau réactif entraîne l'absence de comportement réactif visant à préserver l'espace vital des autres agents : il en résulte une compétitivité physique et l'agent n'hésite pas à bousculer les autres afin de se frayer un chemin à travers la foule.

Colonnes 2 et 3 : On constate que lorsque la symbolisation de la contrainte est limitée au niveau réactif, l'individu aura attendu sagement son tour à l'arrière de la foule alors que lorsqu'elle est

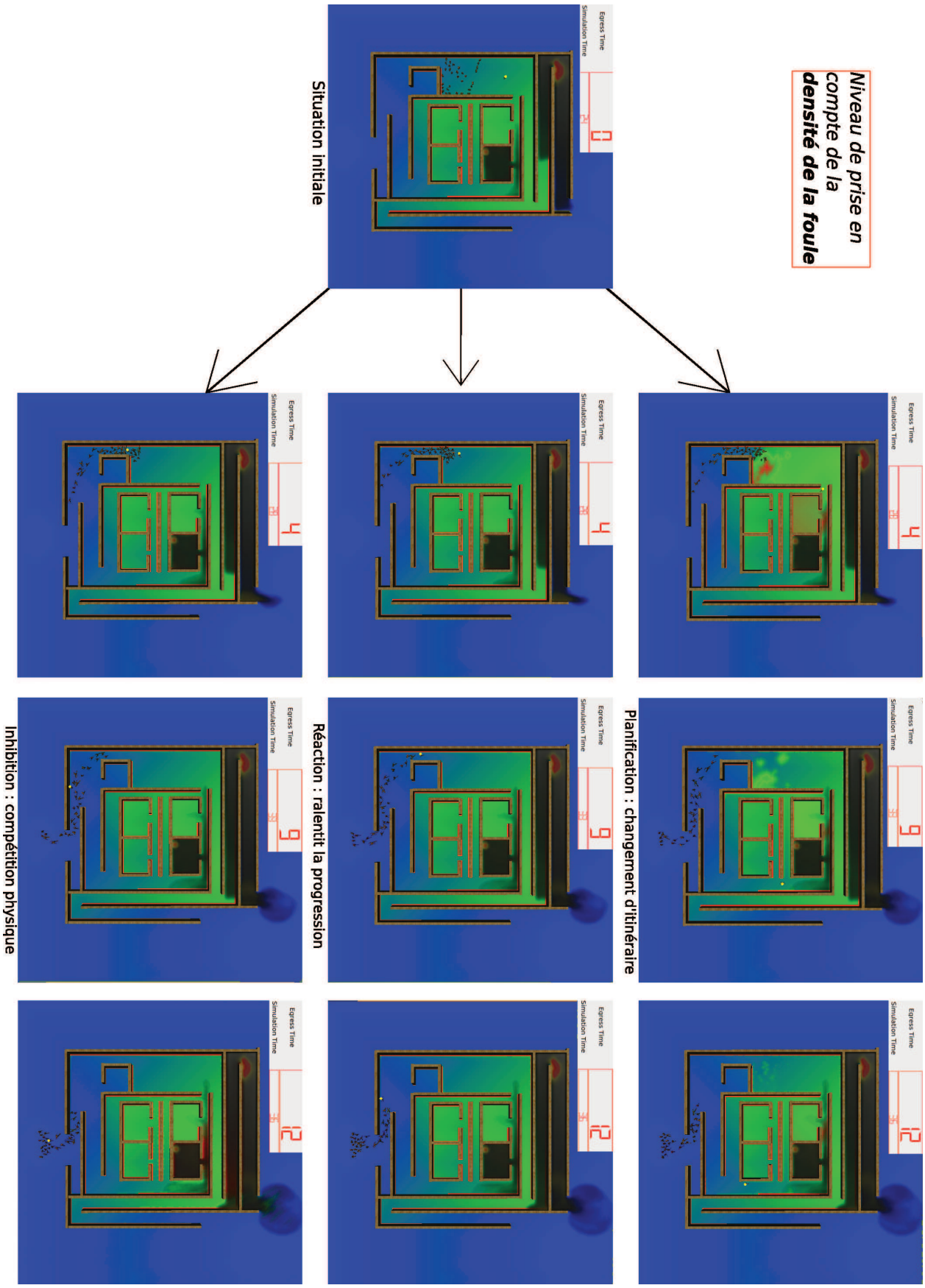


FIGURE 8.10 – Quelques captures de la fénêtre principale de notre simulateur, montrant un scénario dévacuation de bâtiment en feu et l'impact de notre implémentation de la rationalité limitée sur le comportement d'un individu (en jaune). Seule la contrainte *densité de la foule* est concernée par l'application des axiomes de la rationalité limitée.

prise en compte au niveau planifié, il s'est extrait par un autre chemin. On remarque que pour autant il n'est pas parvenu à sortir plus rapidement, ce qui est finalement logique car il est, dans une situation réelle, bien difficile d'anticiper sur le temps nécessaire pour qu'une congestion se dissipe, ou encore sur la vitesse de progression d'une telle congestion. Dans le cas de notre simulation, nous avons dit que chaque contrainte admettait un seuil au dela duquel elle était considérée comme totalement saturée (voir section 8.1.2.1). C'est le cas lorsque la densité de la foule est quasi-maximale, c'est-à-dire quand les individus recouvrent presque tout l'espace au sol (il est en réalité nécessaire de considérer qu'il y a saturation à un taux de recouvrement plus faible car les corps ne sont pas déformables). Lorsque la contrainte est saturée, le couloir est considéré comme bouché et comme il n'y a pas d'heuristique permettant d'anticiper sur la dissipation de la congestion, le processus de recherche de plus court chemin va trouver que tout chemin menant à une autre issue est plus court, au sens de la fonction de coût, que le chemin passant pas la congestion. Ce qui explique que l'individu a replanifié et choisi un chemin qui finalement va s'avérer plus long en temps de parcours. Enfin, lorsque la contrainte est totalement inhibée (images du bas), l'individu est parvenu à remonter la foule en « jouant des coudes » et à s'extraire parmi les premiers. A noter que même si la contrainte densité de la foule n'est pas prise en compte au niveau réactif, la progression de l'agent est néanmoins impactée par la présence des autres individus. Mais cet impact n'est alors pas du au comportement de l'agent, qui de lui-même éviterait de s'approcher trop près des autres, mais aux contacts physiques entre individus. Pour comprendre cela, il faut se rappeler que la simulation du raisonnement humain fournit une intention de déplacement mais que la tranformation de cette intention en déplacement effectif est soumise à validation par un moteur physique, qui gère les interactions physiques entre individus (voir section 7.3). Lorsque la contrainte densité de la foule n'est pas prise en compte au niveau réactif, les seules contraintes au déplacement souhaité sont donc dues aux interactions physiques, qui empèchent par exemple qu'un agent passe par dessus un autre.

8.3 Evaluation de l'implémentation GPU

Nous expliquons d'abord comment nous conditionnons notre framework de résolution afin d'obtenir les meilleures performances en fonction de l'architecture matérielle sur laquelle est exécutée la simulation. Puis nous faisons quelques considérations sur la qualité de l'implémentation proposée.

8.3.1 Conditionnement du framework de résolution

La taille des sous-surfaces est le paramètre crucial conditionnant les performances du processus de résolution. En effet, des sous-surfaces trop grandes peuvent nécessiter plusieurs échanges entre CPU et GPU - sous-surface excédant la taille des buffers du GPU - alors que des sous-surfaces trop petites conduisent à une sous-exploitation du GPU - noeuds n'ayant pas de données à traiter.
Afin de déterminer ce paramètre pour un problème de planification de trajectoire d'évacuation particulier, plusieurs résolutions sont effectuées dans la phase d'initialisation de la simulation en faisant varier ce paramètre. L'environnement de simulation étant fortement dynamique, seule la contrainte forte qu'est la contrainte spatiale est considérée. Les graines de résolution sont alors :

- les buts de l'évacuation de chaque étage en mode inductif (temps réel),
- une suite de positions aléatoires dans le bâtiment en mode déductif (simulation).

Le paramètre offrant les meilleures performances est alors choisi. Dans l'exemple considéré (bâtiment à sortie unique) qui est de résolution globale 2048^2, une taille de bloc de 128^2 donne les meilleurs performances (≈ 8 fois supérieures à l'implémentation brute force (voir figure 8.11).

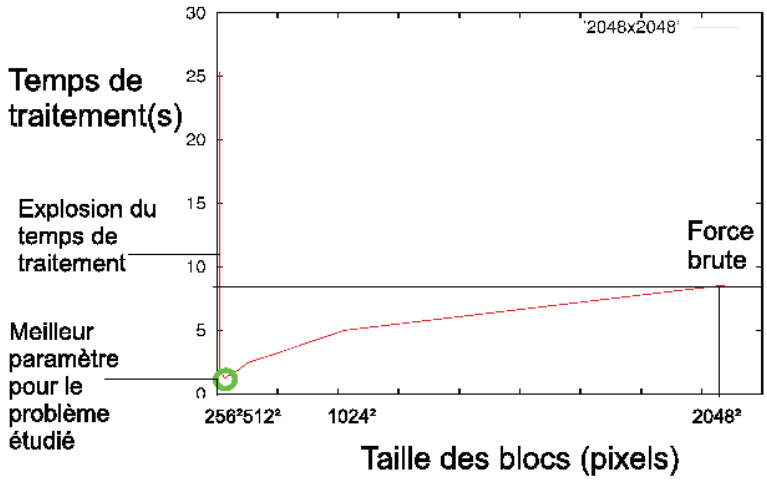


FIGURE 8.11 – Performance de la résolution en fonction de la taille des blocs géométriques.

Ainsi les requêtes d'occlusion permettent de localiser les changements dans la texture ayant été traitée et par suite de minimiser les échanges d'informations entre CPU et GPU. On obtient de cette façon des performances de résolution optimales. Cependant :

- pour des problèmes basses résolutions ($< 128^2$), l'implémentation force brute de l'algorithme reste la plus performante,
- les latences de réponse aux requêtes asynchrones d'occlusion peuvent mener à l'effondrement des performances pour des problèmes complexes.

8.3.2 Considérations qualitatives sur l'implémentation

Contrairement à l'implémentation originale en CUDA, le test de convergence ne nécessite pas un traitement lourd de réduction de la convergence de chaque pixel [Jeong 2007]. Cependant, ce choix d'implémentation entraînent plusieurs problèmes :

- les requêtes d'occlusion ne sont pas performantes sur des architectures à bus AGP,
- les requêtes d'occlusion sont généralement utilisées afin de déterminer la visibilité des objets lorsqu'ils arrivent dans le champ de vision, une application non critique, qui a motivé une adaptation dans les pilotes constructeurs. Ce adaptation a entraîné certains problèmes d'implémentation : les premières requêtes renvoient toujours 0 et un nombre d'itérations minimum de 2 a du être mis en place afin de corriger cet effet indésirable. Une fonctionnalité présente dans OpenGL 3.0 (ARB_Sync_Object) permettrait de régler ce problème, mais cette dernière n'étant pas disponible sur le matériel à notre disposition, elle n'a pas été mise en œuvre.

Malgré tout, le framework de localisation des traitements GPU décrit dans le cadre de la FIM peut être généralisé à la plupart des traitements locaux et représente un caractère fortement attractif pour la montée en charge de la simulation. De plus, il est fondé sur le processus de rastérisation et n'est donc pas portable vers une implémentation CUDA ou OpenCL, où la rastérisation n'est pas exposée. Enfin, cette implémentation a le mérite de n'utiliser que des fonctionnalités normalisées ARB OpenGL 1.5, elle est donc très portable et fonctionne sur les cartes graphiques ne proposant pas de fonctionnalités de rendu avancées.

Fisher et al. ont montré qu'une planification locale avec identification de buts intermédiaires permet de garder des performances très stables lors de changements de l'environnement [G. Fischer 2009]. Notre approche fait de même sans l'identification des buts : les potentiels non locaux sont toujours valides car la perception est un phénomène influant localement la planification inductive.

Une dernière propriété intéressante de notre implémentation est que le caractère local de la minimisation à partir des potentiels voisins inhérent à la FIM ne rend pas important la synchronisation de l'état global entre chaque itération de l'algorithme. En effet, si un potentiel voisin est lu par un thread avant d'avoir été mis à jour par un autre, cette erreur sera corrigée à l'itération suivante. Cette constatation rend obsoléte l'utilisation d'une stratégie de ping-pong de textures, qui est généralement utilisée par les implémentations GPU utilisant l'extension OpenGL, ce qui permet :

1. de diviser le nombre de passes de rendu par 2,
2. d'utiliser avantageusement l'instruction GLSL *discard* afin de signifier la convergence locale (invariance entre 2 itérations) du potentiel d'une cellule.

Chapitre 9

Conclusion et perspectives

Ce travail de thèse a consisté à modéliser, développer et visualiser les mouvements individuels des individus d'une foule en situation d'évacuation d'un bâtiment en feu. Il a donné lieu à un état de l'art des simulateurs existants qui a mis en exergue leurs faiblesses en matière d'expressivité des comportements et en particulier :

- des hypothèses cognitives invalides : omniscience, limitation à certains comportements prédéfinis, ...
- la non prise en compte de l'irrationalité et notamment du phénomène de panique, facteur clé afin de comprendre les comportements en situation d'évacuation de bâtiment en feu.

Un modèle de simulation microscopique de foule en mouvement a été proposé, comportant une couche agent située permettant de modéliser l'adaptation du comportement, sous forme d'implémentation des deux premiers axiomes de la rationalité limitée, tels qu'il ont été formulés par Herbert Simon. L'architecture proposée a ensuite été instanciée pour le problème de l'évacuation de bâtiment en feu. Puis son implémentation a produit un logiciel permettant la simulation et la visualisation de l'évacuation d'un bâtiment par des individus lors d'un incendie. Le recours aux ressources de calcul parallèle disponibles dans les GPUs a été exploitée afin d'obtenir des temps d'exécution compatibles avec une utilisation interactive du logicielle.

9.1 Propriétés du simulateur développé

Nous rappelons les principales propriétés « théoriques » du modèle proposé ainsi que celles qui ont pu être observées, en termes d'expressivité et de performances.

9.1.1 Modèle microscopique

Le simulateur développé intègre un modèle flexible du comportement humain autorisant la personnalisation de plusieurs propriétés intrinsèques à chaque agent, lui permettant ainsi de développer une stratégie d'évacuation qui lui est propre. Il permet de décrire le mécanisme du raisonnement humain selon plusieurs niveaux d'abstraction et offre ainsi un moyen de contrôle de l'adaptation du raisonnement d'un individu confronté à un problème rationnel spatial et soumis à différentes contraintes de mouvement. En particulier, nous proposons une implémentation de chacun des deux premiers axiomes de la théorie de la rationalité limitée proposée par Herbert Simon :

- informations disponibles limitées :
  - acquisition des informations via un module de perception simulant plusieurs sens - vue, toucher, ouie, odorat - individualisant la connaissance de l'environnement, qui de fait n'est plus que partielle,
  - gestion dynamique d'une mémoire individuelle à deux niveaux intégrant la péremption de la connaissance sous forme d'un indice de confiance limité dans le temps.

– moyens cognitifs limités : contrôle du niveau de symbolisation de chaque contrainte, associé à un jauge simulant l'état psychologique de l'agent.

Le processus de planification implémenté intègre l'influence positive des panneaux d'évacuation. Cette prise en compte rationnelle ou navigationnelle des panneaux permet de juger des deux comportements différents face à ces aides dans le bâtiment. A des fins de diagnostic, l'utilisateur final peut donc peupler un bâtiment virtuel décrit par sa géométrie et juger du comportement émergeant de ses occupants vis-à-vis des infrastructures de ce dernier et d'un scénario d'incendie. On peut en particulier déterminer le temps d'évacuation d'un bâtiment et ainsi détecter ses faiblesses de conception et reconsidérer par exemple le positionnement des issues ou simplement celui de la signalisation d'urgence.

Par rapport à la plupart des travaux existants, nous nous affranchissons des concepts d'omniscience et d'intention fanatique. Dans le contexte de situations d'urgence, telles que l'évacuation de bâtiment en feu, ce modèle nous permet de simuler la panique des occupants, élément clé pour comprendre les comportements et apporter des réponses appropriées.

9.1.2 Expressivité et comportements émergents

L'instanciation de l'architecture pour la simulation de l'évacuation de bâtiment en feu a mis en évidence plusieurs modèles de comportements émergents identifiés dans la littérature. Nous les listons ci-dessous en précisant la cause au niveau microscopique et le comportement émergeant au niveau macroscopique.

1. Phénomène de congestion aux issues :
   – microscopique : prise en compte bas niveau de la contrainte d'espace individuel des congénères.
   – macroscopique : congestion et collisions au niveau des portes.
2. Phénomène de répartition sur les issues :
   – microscopique : prise en compte à un haut niveau d'abstraction de la contrainte d'espace individuel des congénères.
   – macroscopique : changement de chemin si congestion au niveau des portes.
3. Phénomène de panique :
   – microscopique : contrainte inhibée au niveau planifié.
   – macroscopique : prostration (exemple : l'agent reste prostré devant la fumée qu'il trouve sur son chemin).

Mais d'autres comportements émergent également du fait de la gestion individuelle de la mémoire :

1. Phénomène d'exploration :
   – microscopique : connaissance partielle du bâtiment.
   – macroscopique : exploration du bâtiment en quête d'issues.
2. Phénomène de remise en question :
   – microscopique : péremption des symboles des contraintes dynamiques.
   – macroscopique : l'individu remet en question la viabilité des informations qu'il a collectées.

Enfin les hypothèses sur l'interprétation cognitive des panneaux d'aide à l'évacuation permet l'apparition de stratégies privilégiant les chemins passant à proximité des panneaux.

9.1.3 Performances

Cette architecture présente un fort potentiel de parallélisation, qui a été mis en exergue à travers l'illustration d'un système de planification individuel sur GPU. Son application pour le problème de la simulation d'évacuation de bâtiment montre de très bonnes performances : 16 itérations par seconde

pour 100 agents sur un GPU Nvidia GTX275.
Les performances temps réel de notre solution ne sont toutefois pas à comparer avec celles des simulateurs de foule utilisés pour les applications de jeux vidéo car l'individualisation des comportements, fondement de notre approche, nous prive naturellement de la possibilité de factoriser les traitements, base de l'optimisation des cadences de rendu par les projets concurrents [Treuille 2006].
Lorsqu'il n'est plus possible de stocker la grille entière en mémoire GPU, nous basculons vers une architecture hybride CPU-GPU, la grille étant découpée en plusieurs sous-grilles chargées tour à tour sur le GPU, ce qui conduit à une grille à 2 niveaux : une sur-grille gérée par le CPU et des sous-grilles gérées par le GPU. Le rôle du GPU est d'appliquer le calcul de la FIM modifiée à la sous-grille que lui envoie le CPU, qui doit lui optimiser l'ordre dans lequel il envoie les sous-grilles au GPU car la convergence globale de l'algorithme est alors très dépendante de cet ordre.

9.2 Perspectives

Les perspectives de travaux futurs s'articulent en deux parties : l'extension des possibilités d'expression de nouveaux comportements et l'extension à d'autres domaines de la simulation.

Plus d'expressivité et tout autant de performances L'architecture et son implémentation font émerger les prémisses d'un système multi-agents. L'intégration d'un modèle BDI (Belief-Desire-Intention) de la rationalité d'un agent serait une bonne perspective de recherche future afin d'augmenter l'expressivité de la plateforme en y intégrant des règles logiques faisant abstraction du problème spatial. On rendrait ainsi son utilisation accessible à la communauté multi-agents pour le développement de simulations performantes. De plus, l'implantation d'une logique surjacente à l'architecture offrirait une sémantique des symboles plus riche, ce qui permettrait notamment d'augmenter l'expressivité des communications échangées et ainsi de développer le modèle de collaboration entre agents.
Si notre modèle cognitif intègre la possibilité pour les individus d'utiliser des aides à l'évacuation, la signalisation notamment, nous n'avons pas pour l'instant considéré l'utilisation de dispositifs de lutte contre l'incendie, tels que les extincteurs. Plus tard, un lien avec les travaux de Paris et Donikian sur les BIIO pourra être fait dans cet objectif [Paris 2009].

Extension de l'implémentation à d'autres simulations D'un point de vue technique, le framework de localisation spatiale des traitements utilisé pour implémenter la Fast Iterative Method démontre une efficacité remarquable sur des grilles de hautes résolutions. Son utilisation pourrait être généralisée à beaucoup d'algorithmes itératifs. Par exemple pour la simulation de fluide, le terme de Poisson de l'équation de Navier Stokes se résout généralement à l'aide d'une méthode itérative comme la méthode des conjugués gradient. Le nombre d'itérations nécessaires afin de faire converger le potentiel de la cellule vers une solution acceptable a une grande cohérence spatiale. Le framework pourrait donc permettre une accélération conséquente de cette méthode pour des milieux hétérogènes de résolution conséquente.

Enfin, le framework de traitement développé peut être utilisé pour tout algorithme global nécessitant une convergence itérative locale. La généralisation de son utilisation pour la résolution de CFD haute résolution mériterait donc de futures investigations.

Bibliographie

[Association 2002] National Fire Protection Association et Society of Fire Protection Engin. Sfpe handbook of fire protection engineering. National Fire Protection Assn, 2002. 15

[Ben Amor 2003] Heni Ben Amor, Oliver Obst et Jan Murray. *Fast, Neat and Under Control : Inverse Steering Behaviors for Physical Autonomous Agents*. Fachberichte Informatik 12–2003, Universität Koblenz-Landau, Universität Koblenz-Landau, Institut für Informatik, Rheinau 1, D-56075 Koblenz, 2003. 18, 20

[Botea 2004] A. Botea, M. Müller et J. Schaeffer. *Near Optimal Hierarchical Path-Finding*. Journal of Game Development, vol. 1, no. 1, pages 7–28, 2004. 41

[Cohen 1995] Jonathan D. Cohen, Ming C. Lin, Dinesh Manocha et Madhav Ponamgi. *I-COLLIDE : an interactive and exact collision detection system for large-scale environments*. In Proceedings of the 1995 symposium on Interactive 3D graphics, I3D '95, pages 189–ff., New York, NY, USA, 1995. ACM. 36

[Cohen 1998] Jonathan Cohen, Marc Olano et Dinesh Manocha. *Appearance-preserving simplification*. In SIGGRAPH '98 : Proceedings of the 25th annual conference on Computer graphics and interactive techniques, pages 115–122, New York, NY, USA, 1998. ACM. 39

[D. Kuligowski 2005] Erica D. Kuligowski et Richard D.Peacock. *Review of Building Evacuation Models*, Juillet 2005. 41

[Dobbyn 2005] Simon Dobbyn, John Hamill, Keith O'Conor et Carol O'Sullivan. *Geopostors : a real-time geometry / impostor crowd rendering system*. In I3D '05 : Proceedings of the 2005 symposium on Interactive 3D graphics and games, pages 95–102, New York, NY, USA, 2005. ACM. 39

[Donikian 2008] Stéphane Donikian, Sébastien Paris et Delphine Lefebvre. *SIMULEM : Introducing Goal Oriented Behaviours in Crowd Simulation*. In Springer Berlin Heidelberg, editeur, Pedestrian and Evacuation Dynamics 2008, pages 479–490. Wolfram W. F. Klingsch, Christian Rogsch, Andreas Schadschneider and Michael Schreckenberg, 2008. 30, 55, 57

[Engel 2001] K. Engel, M. Kraus et T. Ertl. *High-Quality Pre-Integrated Volume Rendering Using Hardware-Accelerated Pixel Shading*. In Eurographics / SIGGRAPH Workshop on Graphics Hardware '01, Annual Conference Series, pages 9–16. Addison-Wesley Publishing Company, Inc., 2001. 40

[Ferber 1997] Jacques Ferber. Les systémes multi-agents : Vers une intelligence collective. Dunod, Décembre 1997. 13, 32, 49

[Fruin 1993] John J. Fruin. *The causes and prevention of crowd disasters*. Engineering for Crowd Safety, pages 99–108, 1993. 34, 70

[Funge 1999] John Funge, Xiaoyuan Tu et Demetri Terzopoulos. *Cognitive modeling : knowledge, reasoning and planning for intelligent characters*. In SIGGRAPH '99 : Proceedings of the 26th annual conference on Computer graphics and interactive techniques, pages 29–38, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co. 55

[G. Fischer 2009] Leonardo G. Fischer, Renato Silveira et Luciana Nedel. *GPU Accelerated Path-planning for Multi-agents in Virtual Environments*. In Games and Digital Entertainment (SBGAMES), 2009 VIII Brazilian Symposium, pages pp.101–110, Rio de Janeiro - Brasil, oct 2009. 95

[Gaskins 2006] Ryland C. Gaskins, Randall D. Spain et Carlotta Boone. *PMFserv Psychological and Software Engineering Analysis*. Rapport technique, Virginia Modeling Analysis and Simulation Center, Old Dominion University, 2006. 45

[Geraerts 2002] Roland Geraerts et Mark H. Overmars. *A comparative study of probabilistic roadmap planners*. In In workshop on the algorithmic foundations of robotics, pages 43–57, 2002. 24

[Helbing 2000] Dirk Helbing, P. Molnar, I. J. Farkas et K. Bolay. *Self-organizing pedestrian movement*. Environment and Planning B : Planning and Design, vol. 28, no. 3, pages 361–383, Mai 2000. 19, 42, 43, 46

[Heliövaara 2007] Simo Heliövaara. *Computational Models for Human Behavior in Fire Evacuations*. PhD thesis, Department of Engineering Physics and Mathematics, Helsinky University of Technology, 2007. 42

[III 2000] Kenneth E. Hoff III, Tim Culver, John Keyser, Ming C. Lin et Dinesh Manocha. *Interactive Motion Planning Using Hardware-Accelerated Computation of Generalized Voronoi Diagrams*. In ICRA, pages 2931–2937. IEEE, 2000. 24

[Jeong 2007] Won-Ki Jeong et Ross T. Whitaker. *A Fast Iterative Method for a Class of Hamilton-Jacobi Equations on Parallel Systems*. SIAM Journal on Scientific Computing, Vol 30, no 5, pages pp.2512–2534, 2007. 30, 76, 77, 94

[Kallmann 2003] M Kallmann, H Bieri et D Thalmann. *Fully Dynamic Constrained Delaunay Triangulations*. In Geometric Modelling for Scientific Visualization, G. Brunnett, B. Hamann, H. Mueller (Eds.), Springer-Verlag, 2003, 2003. 23

[Koenig 2002] Sven Koenig et Maxim Likhachev. *D^* Lite*. Proceedings of the national conference on artificial intelligence, pages 476–483, 2002. 27

[Le Bon 1982] Gustave Le Bon. The crowd : A study of the popular mind. Cherokee Publishing Company, Atlanta, GA 1982, Septembre 1982. 10

[Manolios 2006] Panagiotis Manolios et Yimin Zhang. *Implementing Survey Propagation on Graphics Processing Units*. In Armin Biere et CarlaP. Gomes, editeurs, Theory and Applications of Satisfiability Testing - SAT 2006, volume 4121 of *Lecture Notes in Computer Science*, pages 311–324. Springer Berlin Heidelberg, 2006. 77, 78

[Maslow 1943] A.H. Maslow. *A Theory of Human Motivation*. In Psychoanalytic Quarterly, pages 370–396. N. Reider, 1943. 10

[McDougall 1973] William McDougall. The group mind. Arno Press, Juin 1973. 10

[Newell 1972] Allen Newell et Herbert Alexander Simon. *Human problem solving*. Englewood Cliffs, 1972. 8, 17, 55

[Nguyen 2007a] Hubert Nguyen. *GPU Gems 3 - Chapter 29. Real-time Rigid Body Simulation on GPUs*. In GPU Gems 3. Addison-Wesley Professional, 2007. 37

[Nguyen 2007b] Hubert Nguyen. *GPU Gems 3 - Chapter 32. Broad-Phase Collision Detection with CUDA*. In GPU Gems 3. Addison-Wesley Professional, 2007. 36

[P. Müller 1993] Jörg P. Müller et Markus Pischel. *The agent architecture inteRRaP : Concept and application*. Rapport technique, German Research Center for Artificial Intelligence, 1993. RR 93-26. 32

[Pan 2006] Xiaoshan Pan. *Computational Modeling of human and social behavior for emergency egress analysis*. Civil and environmental engineering dept., Civil and Environmental Engineering Dept, Stanford University, Juin 2006. 9, 46, 47

[Paris 2009] Sébastien Paris et Stéphane Donikian. *Activity-Driven Populace : a Cognitive Approach for Crowd Simulation.* IEEE Computer Graphics and Applications special issue Virtual Populace, Juillet 2009. 99

[Parthenay 2005] Claude Parthenay. *Herbert Simon : rationalité limitée, théorie des organisations et sciences de l'artificiel* , 2005. iii, iv, 9

[Pelechano 2007] Nuria Pelechano, J. M. Allbeck et N. I. Badler. *Controlling individual agents in high-density crowd simulation.* In SCA '07 : Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation, pages 99–108, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association. 44

[Philippsen 2007] Roland Philippsen, Björn Jensen et Roland Siegwart. Autonomous navigation in dynamic environments, volume 35 of *Springer Tracts on Advanced Robotics (STAR) ISBN 978-3-540-73421-5*, chapitre Towards Real-Time Sensor-Based Path Planning in Highly Dynamic Environments. Edited by Christian Laugier and Raja Chatila, 2007. 29

[Procopiuc 2003] Octavian Procopiuc, PankajK. Agarwal, Lars Arge et JeffreyScott Vitter. *Bkd-Tree : A Dynamic Scalable kd-Tree.* In Thanasis Hadzilacos, Yannis Manolopoulos, John Roddick et Yannis Theodoridis, editeurs, Advances in Spatial and Temporal Databases, volume 2750 of *Lecture Notes in Computer Science*, pages 46–65. Springer Berlin Heidelberg, 2003. 23

[Proulx 2002] Guyléne Proulx. *Understanding Human Behaviour in Stressful Situations*, 2002. 3

[Quanrantelli 2001] E.L. Quanrantelli. *The Sociology of Panic.* Rapport technique, University of Delaware, Newark DE. 19716 USA, 2001. 10

[Reynolds 2006] Craig Reynolds. *Big fast crowds on PS3.* In Sandbox '06 : Proceedings of the 2006 ACM SIGGRAPH symposium on Videogames, pages 113–121, New York, NY, USA, 2006. ACM. 19, 44

[Rubini 2007] P. A. Rubini et Q. Zhang. *Simulation of visibility in smoke laden environments.* In 5th International Seminar on Fire and Explosion Hazards, pages 23–27, 2007. 15

[Sethian 1999] James A. Sethian. Level set methods and fast marching methods evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science. Cambridge University Press, 1999. 67, 75

[Sethian 2003] James A. Sethian et Alexander Vladimirsky. *Ordered Upwind Methods for Static Hamilton–Jacobi Equations : Theory and Algorithms.* SIAM J. Numer. Anal., vol. 41, no. 1, pages 325–363, 2003. 28

[Sime 1995] J. D. Sime. *Crowd psychology and engineering.* Safety Science, vol. 21, no. 1, pages 1 – 14, 1995. 3

[Simonin 2000] Olivier Simonin et Jaques Ferber. *Reactive Real-Time Cooperation as a Combination of Altruism and Self Satisfaction.* In Fourth International Conference on Multi-Agent Systems (ICMAS'00), pages pp 439–440, Boston, USA, 2000. 48

[Speitel 1996] Louise C. Speitel. *Fractional effective dose model for post-crash aircraft survivability.* Toxicology, vol. 115, no. 1-3, pages 167 – 177, 1996. International Colloquium on Advances in Combustion Toxicology. 74

[Staal 2004] M. A Staal. *Stress, cognition, and human performance : A literature review and conceptual framework.* NASA Technical Memorandum, vol. 212824, 2004. 9, 11

[Stentz 1994] Anthony Stentz. *Optimal and Efficient Path Planning for Partially-Known Environments.* In ICRA, pages 3310–3317. IEEE Computer Society, 1994. 27

[Thalmann 2008] Daniel Thalmann. Crowd simulation. Springer-Verlag London Limited, 2008. 37

[Treuille 2006] Adrien Treuille, Seth Cooper et Zoran Popović. *Continuum crowds.* ACM Trans. Graph., vol. 25, no. 3, pages 1160–1168, 2006. 29, 67, 99

[van den Berg 2008] Jur van den Berg, Ming C. Lin et Dinesh Manocha. *Reciprocal Velocity Obstacles for Real-Time Multi-Agent Navigation.* In IEEE international conference on robotics and automation, pages 1928–1935. IEEE, 2008. 20

[Van den Bergen 1999] Gino Van den Bergen. *A fast and robust GJK implementation for collision detection of convex objects.* J. Graph. Tools, vol. 4, no. 2, pages 7–25, Mars 1999. 36

[Veeraswamy 2009] A. Veeraswamy, E.R. Galea et P. Lawrence. *Implementation of Cognitive Mapping, Spatial Representation and Wayfinding Behaviours of People within Evacuation Modelling Tools.* In Proceedings of the 4th International Symposium on Human Behaviour in Fire, pages 501–512, Robinson College, Cambridge, UK, Juillet 2009. 41

[Wilkie 2009] David Wilkie, Jur Van Den Berg et Dinesh Manocha. *Generalized velocity obstacles.* In IROS'09 : Proceedings of the 2009 IEEE/RSJ international conference on intelligent robots and systems, pages 5573–5578, Piscataway, NJ, USA, 2009. IEEE Press. 19

[Xie 2007] Hui Xie, Lazaros Filippidis, Steven Gwynne, Edwin R. Galea, Darren Blackshields et Peter J. Lawrence. *Signage Legibility Distances as a Function of Observation Angle.* Journal of Fire Protection Engineering, pages 17–41, 2007. 15

Chapitre 10

Fichiers de configuration XML

Cette annexe contient un exemple de fichiers de configuration au format XML permettant de décrire tous les éléments d'une simulation, à savoir le bâtiment au sein duquel se déroule le scénario d'évacuation et les occupants du bâtiment.

10.1 Description de l'environnement

```
<?xml version="1.0" encoding="iso-8859-1" standalone="yes" ?>
<EveLifeXML cfg="EnvironmentManagement">
        <Environment ID="000" xmin="-32" xmax="32" ymin="-32" ymax="32" resolution="128"
                     generationfile="obstaclesprebuilt.temp" maxFBOsize="2048">
                <!--Effective environment goals-->
                        <Goals File="escape.bmp"/>
                <!--FDS input files-->
                        <Smoke File="FDSscenars/mybat/mybat_01.s3d.svz"  />
                        <Fire File="FDSscenars/mybat/mybat_02.s3d.svz"  />
                        <Temperature File="FDSscenars/mybat/mybat_01.sf.svz"  />
                        <FED File="FDSscenars/mybat/mybat_02.sf.svz"  />
                <!--scene containing vertical obstacles scene-->
                        <OSGObstacleGeom OSGgeometry="scene/mybat.osg"/>
                <!--agent vertical collision scene-->
                        <Ground OSGgeometry="scene/floor.osg"/>
        </Environment>
</EveLifeXML>
```

10.1.1 Configuration des agents

```
<?xml version="1.0" encoding="iso-8859-1" standalone="yes" ?>
<EveLifeXML cfg="AvatarManagement">

<!--Chargement d'un modèle visuel d'agent et de ses animations-->
<CoreModel NameDir="bizguy3" cal3dcfg="cal3d.cfg">
        <Animation ID="Idle" AnimNo="0" speedMin="0" speedMax="0.01"/>
        <Animation ID="Walk" AnimNo="1" speedMin="0.01" speedMax="2"/>
        <Animation ID="Run"  AnimNo="2" speedMin="2" speedMax="6"/>
</CoreModel>
```

```
<!--Définition de l'archétype de comportement qui sera affecté par défaut lors de l'instanciatio
<DefaultProperties>
        <!--Nombre max d'itérations de planification par pas de temps-->
                <Property Name="MaxNbFIMPass" Value="40"/>
        <!--Connaissance préalable/ Hypothèse sur l'environnement-->
                <Property Name="Basic Knowledge" Value="known.bmp"/>
        <!--Pondérations des contraintes-->
                <Property Name="AlphaBlockage" Value="10.0"/>
                <Property Name="AlphaSmoke" Value="0.9"/>
                <Property Name="AlphaDensity" Value="0.3"/>
                <Property Name="AlphaTemperature" Value="0.6"/>
        <!--Seuils de saturation des contraintes-->
                <Property Name="ThresholdSmoke" Value="0.053"/>
                <Property Name="ThresholdTemperature" Value="0.08"/>
                <Property Name="ThresholdDensity" Value="1.0"/>
        <!--Longevités des contraintes 0:volatile 1:persistant-->
                <Property Name="SmokeTTL" Value="1"/>
                <Property Name="TemperatureTTL" Value="1"/>
                <Property Name="DensityTTL" Value="0.0005"/>
        <!--Niveau de prise en comptes des contraintes(0:inhibition-1:reaction-2:Planification -
                <Property Name="SmokeLevel" Value="2"/>
                <Property Name="TemperatureLevel" Value="2"/>
                <Property Name="DensityLevel" Value="2"/>
        <!--Pondérations des réactions aux contraintes -->
                <Property Name="SmokeReaction" Value="0.3"/>
                <Property Name="TemperatureReaction" Value="0.3"/>
                <Property Name="DensityReaction" Value="0.7"/>
        <!--Propriétés physiques et pseudo physique -->
                <Property Name="Maxspeed" Value="5"/>
                <Property Name="Maxforce" Value="50000"/>
                <Property Name="Mass" Value="5"/>
                <Property Name="Rangeofsight" Value="30"/>
</DefaultProperties>
```

```
<!--Instanciation d'un seul agent-->
<Avatar CoreRef="bizgirl3" EnvID="000" ID="Jon_52" Mass="80" Maxforce="5000" Maxspeed="5" Radius
        <position x="-21.8771" y="-0.349569" z="0"/>

        <CognitiveProperties>
        <!--Nombre max d'itérations de planification par pas de temps-->
                <Property Name="MaxNbFIMPass" Value="40"/>
        <!--Connaissance préalable/ Hypothèse sur l'environnement-->
                <Property Name="Basic Knowledge" Value="Jon_52.bmp"/>
        <!--Pondérations des contraintes-->
                <Property Name="AlphaBlockage" Value="10.0"/>
                <Property Name="AlphaSmoke" Value="0.9"/>
                <Property Name="AlphaDensity" Value="0.3"/>
                <Property Name="AlphaTemperature" Value="0.6"/>
        <!--Seuils de saturation des contraintes-->
                <Property Name="ThresholdSmoke" Value="0.053"/>
                <Property Name="ThresholdTemperature" Value="0.08"/>
                <Property Name="ThresholdDensity" Value="1.0"/>
        <!--Longevités des contraintes 0:volatile 1:persistant-->
                <Property Name="SmokeTTL" Value="1"/>
                <Property Name="TemperatureTTL" Value="1"/>
                <Property Name="DensityTTL" Value="0.0005"/>
        <!--Niveau de prise en comptes des contraintes(0:inhibition-1:reaction-2:Planification -
                <Property Name="SmokeLevel" Value="2"/>
                <Property Name="TemperatureLevel" Value="2"/>
                <Property Name="DensityLevel" Value="2"/>
        <!--Pondérations des réactions aux contraintes -->
                <Property Name="SmokeReaction" Value="0.3"/>
                <Property Name="TemperatureReaction" Value="0.3"/>
                <Property Name="DensityReaction" Value="0.7"/>
        <!--Propriétés physiques et pseudo physique -->
                <Property Name="Maxspeed" Value="5"/>
                <Property Name="Maxforce" Value="50000"/>
                <Property Name="Mass" Value="5"/>
                <Property Name="Rangeofsight" Value="30"/>
        </CognitiveProperties>
</Avatar>
```

```
<!--Generation d'agents dans un périmètre avec des valeurs d'attributs compris dans les interval
<!--Note: les apparences sont séléctionnés au hazard-->
<GeneratedAvatars ID="001" EnvID="000" IdBeg="2" IdEnd="10"
        centerx="0" centery="0" centerz="0"
        perimeter="10"
        minRadius="0.3" maxRadius="0.3"
        minMass="60" maxMass="80"
        minRangeofsight="10" maxRangeofsight="10"
        minMaxspeed="5" maxMaxspeed="5"
        scale="0.03"
        minMaxforce="500" maxMaxforce="500"

        <CognitiveProperties>
        <!--Pondérations des contraintes-->
                <Property Name="AlphaBlockage" Value="10.0"/>
                <Property Name="AlphaSmoke" Value="0.9"/>
                <Property Name="AlphaDensity" Value="0.3"/>
                <Property Name="AlphaTemperature" Value="0.6"/>
        <!--Seuils de saturation des contraintes-->
                <Property Name="ThresholdSmoke" Value="0.053"/>
                <Property Name="ThresholdTemperature" Value="0.08"/>
                <Property Name="ThresholdDensity" Value="1.0"/>
        <!--Longevités des contraintes 0:volatile 1:persistant-->
                <Property Name="SmokeTTL" Value="1"/>
                <Property Name="TemperatureTTL" Value="1"/>
                <Property Name="DensityTTL" Value="0.0005"/>
        <!--Niveau de prise en comptes des contraintes(0:inhibition-1:reaction-2:Planification -
                <Property Name="SmokeLevel" Value="2"/>
                <Property Name="TemperatureLevel" Value="2"/>
                <Property Name="DensityLevel" Value="2"/>
        <!--Pondérations des réactions aux contraintes -->
                <Property Name="SmokeReaction" Value="0.3"/>
                <Property Name="TemperatureReaction" Value="0.3"/>
                <Property Name="DensityReaction" Value="0.7"/>
                <Property Name="Basic Knowledge" Value="known.bmp"/>
        </CognitiveProperties>
</GeneratedAvatars>
```