Revue Électronique Francophone d'Informatique Graphique, Volume 1, Numéro 1, pp. 33–43, 2007

Vers une architecture pour la simulation microscopique de foule

Sébastien Paris[1,2], Stéphane Donikian[2] et Nicolas Bonvalet[1]

sgparis@irisa.fr, donikian@irisa.fr, nicolas.bonvalet@arep.fr
[1]AREP VILLE, 163 bis avenue de Clichy, impasse chalabre, 75847 Paris cedex
[2]IRISA, Campus de Beaulieu, 35042 Rennes Cedex, France

Résumé

Cet article présente une architecture complète pour effectuer des simulations de foule d'une manière microscopique, c'est à dire en simulant chaque entité de manière autonome. Nous abordons ainsi l'ensemble des étapes nécessaires à un tel processus. Premièrement, nous proposons une représentation de l'environnement efficace en termes de calculs, obtenue de manière automatique, et comportant un ensemble d'informations utilisables à moindre coût par les entités. Ensuite, nous proposons un moteur de description des objets interactifs de la simulation, permettant d'enrichir le potentiel comportemental de nos entités simulées. Nous mettons l'accent concernant ces interactions sur l'intégration des objets interactifs directement dans la procédure de planification de chemin de l'humanoïde. Cette dernière, basée sur une connaissance individuelle de l'environnement, prend en compte plusieurs critères de décisions simultanément, et comporte différentes conditions de fin, aussi bien géographiques que conceptuelles. Finalement, nous montrons comment l'ensemble de ces modèles s'intègrent dans notre architecture pour obtenir une plateforme de simulation de foules.

We present in this study an architecture for microscopic crowd simulation, i.e. which simulates each entity by an autonomous agent. We treat in this article all the necessary steps for such a process. First, we propose a representation of the environment which is computationally efficient, automatically calculated, and informed with pre computed data which can be used by the entities at low cost. Then, we propose a description engine for the interactive objects of the simulation, which enhances the behavioural potential of the simulated entities. For these interactions, we emphasise the fact that the interactive objects are directly integrated inside the path planning process of the humanoid. This path planning process is based on an individual knowledge base about the environment, takes into account several decision criteria at the same time, and manages different completion conditions which may be as well geographical as conceptual. Finally, we show how these models are integrated in our architecture to obtain a crowd simulation platform.

Mots clé : Informatique Graphique, Simulation de Foules

1. Introduction

L'évaluation des bâtiments publics, et plus particulièrement ceux comportant une forte affluence, est depuis longtemps une nécessité en matière de sécurité. Mais depuis peu, un nouveau cadre d'évaluation voit le jour, pour lequel un bâtiment n'est plus seulement jugé par rapport à certaines normes de sécurité (plus ou moins laxistes en ce qui concerne les capacités d'accueil des personnes), mais aussi expertisé par rapport à la qualité des services qu'il offre à ses usagers. C'est dans ce cadre que se placent nos travaux : quantifier, par la simulation, la qualité de service des lieux publics, potentiellement à forte affluence, et plus particulièrement les lieux d'échange que sont les gares. Cette étude est donc issue d'une étroite collaboration entre l'équipe *Bunraku* de l'IRISA, apportant son expertise en matière de simulation et d'environnements virtuels, et l'*AREP*, filiale du groupe SNCF en charge de l'aménagement et de la construction des lieux d'échange, apportant son expertise en matière d'étude de flux et d'organisation

des circulations de personnes. Ainsi, des techniques de simulation de foules ont été étudiées et réalisées, en retenant une approche peu utilisée dans ce domaine, de par sa complexité de mise en œuvre, qui est la simulation microscopique par agents autonomes. Cette technique simule chaque individu avec son comportement propre, en laissant le système évoluer pour former le comportement macroscopique émergeant dénommé *foule*. Le principal avantage de cette technique est de prendre en compte des comportements suffisamment évolués pour chaque entité, permettant ensuite de produire des données de sortie suffisamment réalistes pour être étudiées. Notons que lorsque nous évoquons le réalisme d'une simulation nous n'abordons pas le sujet du réalisme visuel, qui n'est que peu important dans notre cadre, mais plutôt du réalisme comportemental, à savoir la dynamique macroscopique des flux de population, leur capacité à réagir et à *comprendre* leur environnement. De plus, comme ce qui nous intéresse est de produire des données exploitables, la vitesse d'exécution d'une simulation ne doit pas nécessairement supporter le temps réel, bien qu'une exécution en temps interactif soit appréciable pour l'utilisateur.

Cet article présentera ainsi successivement l'ensemble des techniques nécessaires à l'élaboration d'une simulation de foule microscopique. Nous commencerons par la représentation de l'environnement (chapitre 2), y compris son abstraction et l'ensemble des informations qui y sont stoquées. Ensuite, sera présenté le moteur de description des objets (chapitre 3), nommé *BIIO* pour *Behavioural Interactive and Introspective Objects*, qui permet une description hiérarchique des composants interactifs de la simulation. Le modèle d'humanoïde, avec l'ensemble de ses composants (connaissance de l'environnement, procédure d'observation, planification de chemins multicritères, etc.), sera ensuite décrit (chapitre 4). Pour finir, nous présenterons une partie des résultats et conclurons avec les perspectives d'évolution de ce modèle.

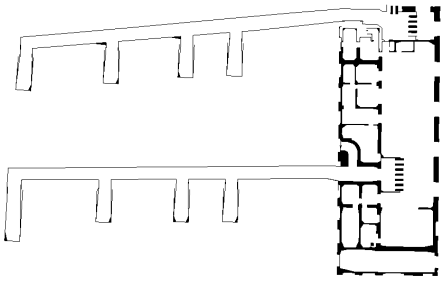
2. Représentation de l'environnement

2.1. Introduction

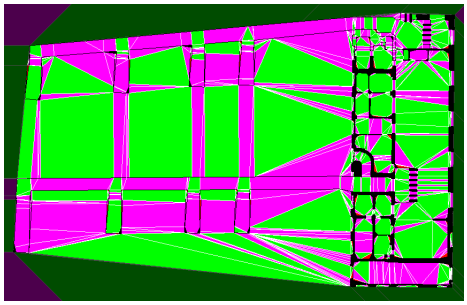
Afin de pouvoir raisonner sur son environnement, ne serait-ce que pour s'y déplacer, une entité doit forcément en avoir une représentation. Nous allons donc décrire ici la façon dont nous représentons l'environnement virtuel, ainsi que les informations que nous lui associons. Plusieurs méthodes de description existent déjà dans la littérature, provenant toutes de recherches sur la robotique [Lat91] : les champs de potentiel [GSN04], les cartes de cheminement [SKG05], ou encore la décomposition en cellules [AVF04]. C'est cette dernière méthode que nous avons privilégiée, les deux premières n'offrant pas assez de flexibilité dans notre cadre d'application. De plus, nous avons utilisé une décomposition exacte de l'environnement en cellules convexes, plutôt qu'une décomposition sous forme de grille jugée trop approximative. Notons finalement que cette représentation est globale pour toute la simulation, et donc partagée entre toutes les entités.

2.2. Subdivision

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



(a) Géométrie brute (murs en noir)



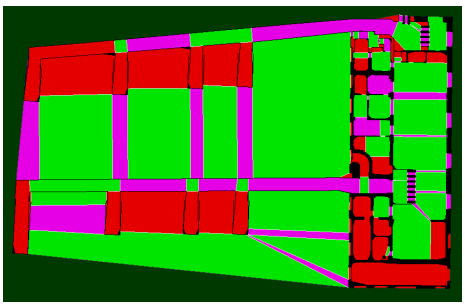
(b) Subdivision (*1526 cellules*)

Figure 1: *Etape de subdivision de la gare de St Denis*

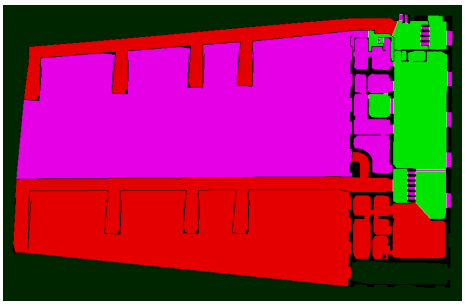
2.3. Abstraction

Comme nous pouvons le voir sur la figure 1, une subdivision produit un grand nombre de cellules, même pour un environnement relativement simple. C'est pourquoi nous avons mis en place un processus d'abstraction de l'environnement [PDB05] visant à faciliter l'exploitation d'une telle représentation. L'abstraction se déroule en deux étapes,

produisant deux graphes successifs de plus en plus condensés, organisés hiérarchiquement, i.e. dont les nœuds contiennent un sous-ensemble du graphe précédent. Nous nommerons les nœuds du premier niveau d'abstraction des *groupes*, et ceux du second niveau des *zones*. L'heuristique de regroupement des cellules se base à la fois sur leurs propriétés géométriques, maximisant l'aspect convexe des groupes pour obtenir un découpage plus naturel, et sur leurs propriétés topologiques, réunissant les nœuds de même catégorie afin de mettre en évidence des zones de navigation plus globales. Comme nous pouvons le voir sur la figure 2, l'abstraction produit une représentation plus intuitive de l'environnement, et a l'avantage de réduire drastiquement le nombre de nœuds à manipuler.



(a) 1ère abstraction (*153 groupes*)



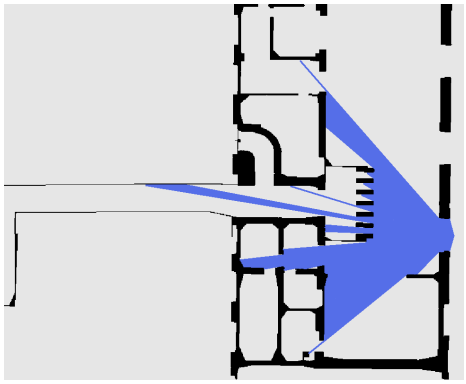
(b) 2ème abstraction (*40 zones*)

Figure 2: *Abstractions successives de la gare de St Denis*

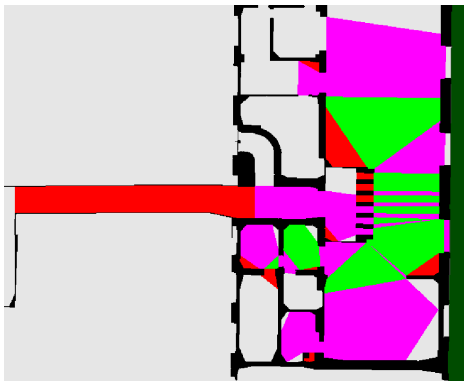
2.4. Graphe informé

Une fois ce graphe hiérarchique obtenu, nous associons un ensemble de données précalculées à certains de ses nœuds. Cette phase d'information a deux buts principaux : 1) accélérer l'accès aux données lors de la simulation ; 2) faciliter l'exploitation du graphe pour certaines fonctionnalités des entités, comme nous le verrons par la suite. Comme il serait trop coûteux en espace mémoire de stocker des précalculs pour l'ensemble des cellules de la subdivision, ceux-ci ne sont évalués que pour les groupes.

Ainsi, nous calculons tout d'abord les champs de visibilité potentiels (*Potential Visibility Sets* en anglais) au travers de l'environnement, qui nous servirons pour le comportement d'observation des entités. Ces PVS se présentent sous la forme d'arbres dont la racine est une frontière franchissable de l'environnement, et dont chaque nœud correspond à la prochaine frontière franchissable potentiellement visible. Sans rentrer dans les détails de l'algorithme, le procédé de calcul est une planification de chemin au travers de l'environnement. Nous pouvons voir sur la figure 3 un PVS représenté sous la forme de frustra composés, ainsi que la correspondance en matière de connectivité des cellules perçues. Un frustrum est ici la projection dans le plan d'un cône de vision, et est donc représenté par un triangle dont l'un des sommets est la position de l'entité, et les deux



(a) Frustra composés



(b) Connectivité perçue

Figure 3: *Exemple de champ de visibilité (PVS)*

autres sommets les limites du champ de vision dans l'environnement. La propriété la plus importante d'un tel frustrum est que l'ensemble de sa surface couvre une zone libre, i.e. sans obstacle, de l'environnement.

D'autre part, une grille orientée est affectée à chaque groupe, dont les cases vont stocker dynamiquement lors de la simulation le nombre d'entités qu'elles contiennent. Ce procédé nous permet ensuite de faire un calcul fin de densité pour chaque paire de connexion du groupe, et ainsi d'obtenir une valeur de densité tenant compte de la topologie des lieux (et ainsi des regroupements gênant la circulation ou non). La

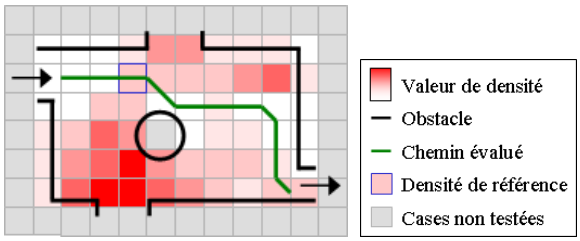


Figure 4: *Calcul fin de densité utilisant une grille orientée*

méthode de calcul utilisée prend la forme d'une planification de chemin au sein de la grille (figure 4), avec pour origine la case correspondant à l'entrée du groupe envisagée, et pour destination la sortie évaluée. Le chemin est évalué en cherchant à minimiser la distance parcourue, mais aussi la densité des cases traversées. Finalement, c'est la densité la plus importante des cases composant le chemin qui est retenue.

Nous précalculons aussi les plus courts chemins, ainsi que leur largeur de passage minimale, à l'intérieur de chaque groupe entre chacune de ses frontières franchissables, ce qui nous servira lors de la planification de chemin. Finalement, un compteur est associé à chaque frontière intergroupe, qui est mis à jour par chaque entité traversant le groupe en direction de cette frontière afin de permettre le calcul des flots de personnes dans le groupe.

3. Moteur de description des objets interactifs

3.1. Introduction

La représentation de l'environnement que nous venons de présenter permet certes une description topologique des lieux, mais a comme lacune principale de ne pas contenir d'information concernant les possibilités d'interactions. En effet, les lieux que nous voulons simuler, comme les gares ferroviaires, ont par nature un fort potentiel d'interactivité. C'est pourquoi nous proposons une architecture de description des interactions nommée *BIIO* pour *Behavioural Interactive and Introspective Objects* [PDB06a]. Comme son nom l'indique, cette architecture va permettre d'exprimer le potentiel d'interaction des objets simulés, un peu à la manière de la théorie écologique [Gib97]. Les Smart Objects [KT99] abordent déjà ce sujet, en se concentrant surtout sur la description d'une interaction physique avec l'objet, et donc de la description des contraintes et pré-conditions nécessaires pour effectuer cette interaction [ACT06].

3.2. Architecture de modélisation

Dans notre cadre d'application, nous sommes moins intéressés par cette interaction physique, locale dans le schéma comportemental d'un individu, que par le processus qui va pousser un individu à *choisir* un objet interactif parmi un ensemble d'objets dans son environnement. Donc, sans aller jusqu'à une planification rationnelle, nous avons besoin de pouvoir spécifier des procédures comportementales de haut niveau, et laisser les individus simulés résoudre ces procédures comportementales avec un niveau d'autonomie maximal. Ainsi, l'architecture BIIO propose un service d'introspection permettant à chaque objet simulé, que ce soit un équipement aussi bien qu'un agent autonome, d'accéder à l'ensemble des propriétés des objets présents dans le monde virtuel, en n'ayant que peu d'a priori sur leur composition. Une telle propriété nous permet de concevoir des procédures comportementales génériques, pouvant s'adapter automatiquement à l'introduction de nouvelles catégories d'objets dans la simulation.

La description des objets, quant à elle, se fait de manière hiérarchique, permettant de décrire des procédures comportementales atomiques, puis de les composer par héritage afin de produire des objets plus complexes. Ainsi, le moteur de BIIO propose par défaut une première architecture d'objets interactifs (figure 5), qui pourra servir de base pour toute description ultérieure. Les objets *utilisables* sont nativement capable de gérer leur file d'attente, composée d'objets *utilisateurs*, et de permettre à leur utilisateur courant d'exécuter un ou plusieurs de ses comportements interactifs. De la même manière, un objet *observable* peut mettre à disposition de ses *observateurs* un ensemble d'information qu'il contient.

Comme indiqué précédemment, la création de nouveaux objets interactifs se fait par héritage et spécialisation d'objets existants. Ainsi, pour créer un objet utilisable simple, comme un composteur dans une gare, l'utilisateur se contentera d'hériter de l'objet *utilisable* et de le spécialiser avec le comportement associé au composteur (comme le changement de l'état d'une variable *billet* de l'humanoïde). Si l'utilisateur veut aussi rajouter une information visuelle au composteur, comme l'affichage de l'heure, il lui suffit d'hériter aussi de l'objet *visualisable* en lui associant l'information visuelle de l'heure courante. Il est aussi possible de créer des objets héritant à la fois d'*utilisable* et d'*utilisateur*, afin par exemple de gérer une interaction sociale entre les humanoïdes, comme demander l'heure à quelqu'un.

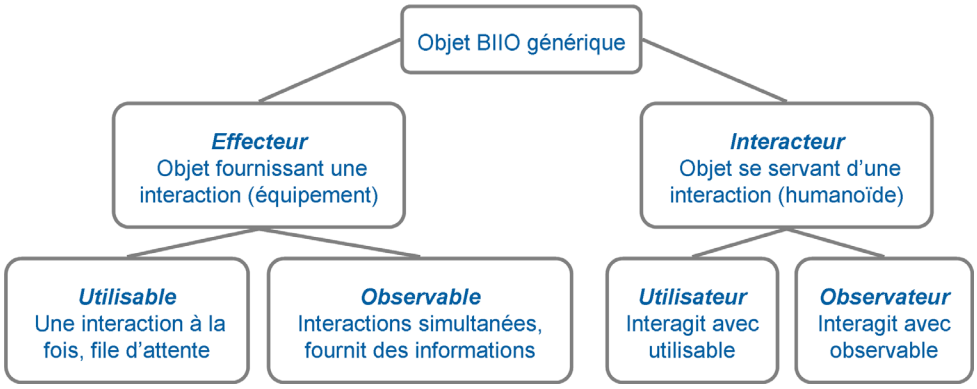
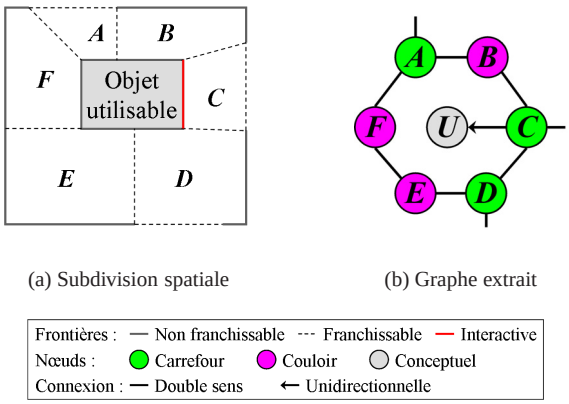


Figure 5: *Hiérarchie de base des objets interactifs*

3.3. Description des interactions

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



(a) Subdivision spatiale (b) Graphe extrait

Figure 6: *Intégration d'un objet utilsable statique dans le graphe de l'environnement*

3.4. Environnement informé

Le dernier aspect nécessaire à notre schéma de simulation est la prise en compte de la topologie des lieux dans le raisonnement des entités simulées, et donc de proposer un environnement informé. Pour ce faire, nous avons besoin d'intégrer totalement les objets interactifs à notre description de l'environnement. Ainsi, les objets BIIO peuvent être déclinés en deux catégories topologiques : objet statique (dont la position est figée durant la simulation), et objet dynamique (qui va pouvoir se déplacer).

Les objets statiques sont directement insérés dans le graphe d'abstraction de l'environnement sous la forme de nœuds sémantiques dits *conceptuels* (figure 6). Ainsi, les objets statiques bénéficient automatiquement de l'ensemble des propriétés des nœuds du graphe, que ce soit au niveau des pré-calculs effectués (*PVS*, etc.), ou au niveau des algorithmes de parcours applicables (comme la planification de chemin que nous verrons par la suite). De plus, une sous-catégorie des objets statiques est proposée, les objets traversables, qui pourront relier deux zones (comme pour un escalator) ou plus (un ascenseur) de l'environnement.

Les objets dynamiques sont quant à eux intégrés par référencement dans les groupes préexistants du graphe de l'environnement, permettant ainsi de conserver un lien avec la topologie. Nous proposons ainsi un certain nombre de procédures permettant d'obtenir un équivalent rapide et efficace aux procédures proposées par les noeuds du graphe environnemental. Ainsi, nous proposons les *dPVS* (pour *dynamic PVS*) qui permettent d'obtenir très rapidement le potentiel de visibilité des objets dynamiques en se basant sur les *PVS* de la zone de l'environnement où ils se trouvent (figure 7). Le calcul de ces *dPVS* se fait en intersectant le frustrum de l'objet avec l'ensemble des *PVS* du groupe qui le contient. Tant que l'intersection n'est pas vide, l'algorithme parcours les *PVS*, produisant ainsi rapidement les zones visibles par l'objet.

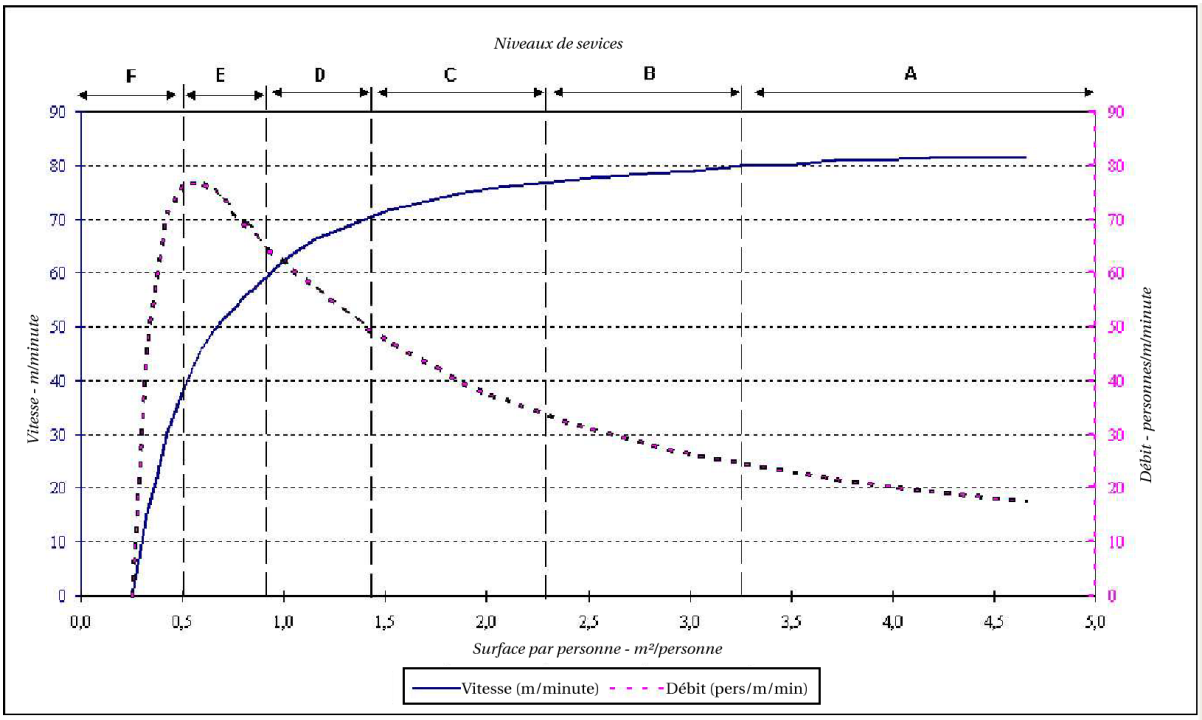


Figure 8: *Relation entre densité et débit pour chaque niveau de service*

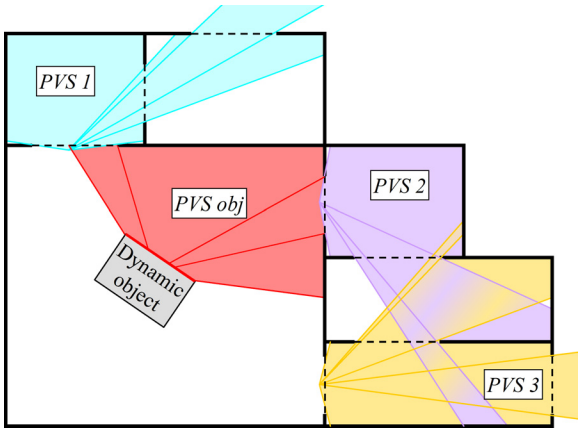


Figure 7: *Calcul du dPVS - le frustrum de visibilité de l'objet (en rouge) est connecté aux PVS statiques de l'environnement qui sont visibles*

4. Procédures de l'humanoïde virtuel

4.1. Introduction

Maintenant que nous avons défini notre description de l'environnement et des objets interactifs, voyons quelles procédures sont proposées par défaut pour l'objet interactif spécifique qu'est l'humanoïde virtuel. Très vite lors de sa conception, nous nous sommes rendus compte que des modèles simples, comme ceux à base de particules [HFV00], ne permettaient pas de prendre en compte la diversité des comportements individuels nécessaires. Néanmoins, certains modèles macroscopiques [Tog55, MB75, Pau84, PM78, TTK88] présentent des formules d'évaluation et des observations des flux de personnes qui peuvent être fortement utiles pour paramétrer ou faire une première validation d'un modèle plus fin. D'autres observations et expériences faites par des urbanistes sur les circulations de personnes sont aussi d'un intérêt majeur. Par exemple, J.J. Fruin [Fru71] est le premier à introduire la notion de niveau de service en se basant sur des observations réelles. Ainsi, il propose une table de classification en 6 niveaux de services (de *A* à *F*) quantifiant la vitesse de placement d'une personne ainsi que les débits de personnes pour des valeurs de densités croissantes (figure 8).

4.2. Connaissance et observation

La première nécessité pour notre modèle est d'introduire la notion de connaissance de l'environnement pour l'humanoïde, afin d'éviter de simuler des entités omniscientes. Nous pouvons trouver certaines approches dans la littérature [AM05, HK05] qui, bien qu'elles proposent de limiter la connaissance de l'entité simulée, restent plus restreintes

sur ses capacités à acquérir de nouvelles informations. C'est pourquoi nous proposons d'introduire cette notion de connaissance pour nos entités, mais aussi de fournir un modèle d'observation de l'environnement qui permet à l'entité de découvrir de nouvelles choses au cours du temps, ou encore de rafraîchir sa base de connaissances (dans le cas d'informations dynamiques) [PDB05].

Nous proposons de scinder la connaissance d'un humanoïde en deux parties. Tout d'abord, une connaissance topologique représentée par une table temporelle qui associe à chaque bordure intergroupe du graphe d'abstraction global une date d'observation locale à l'entité. Cette date sert ensuite de référence pour déterminer si les informations dynamiques correspondant à la localisation de la bordure sont disponibles pour l'humanoïde ou non : une durée de validité est associée à chaque type d'information dynamique, comme les densités de personnes, qui va ensuite être comparée à la dernière date d'acquisition de cette information, i.e. la date d'observation de la zone de l'environnement contenant cette information. De plus, la valeur spéciale *jamais* signifie que l'humanoïde ne connaît pas cette bordure de l'environnement, et ne peut donc pas en tenir compte dans son raisonnement, que se soit pour planifier un chemin ou pour se référer à une donnée dynamique associée. Notons que ce modèle de connaissance est volontairement simplifié, par rapport à des modèles de représentation plus abstraits de l'espace et de la mémoire [TD03], afin de conserver les performances pour un grand nombre d'entités.

L'autre connaissance prise en charge par l'humanoïde est beaucoup plus volatile, et concerne les objets interactifs présents dans son champ de vision. Ainsi, une liste des objets visibles est construite, où chaque objet peut prendre trois états :

Visible L'humanoïde sait qu'il y a un objet, mais ne connaît aucune de ses propriétés.

Identifié L'objet a été identifié, i.e. l'humanoïde a accès à l'ensemble des constituants de l'objet, mais pas à leurs valeurs, et ne peut donc pas effectuer une interaction avec cet objet.

Utilisable L'humanoïde a accès à l'ensemble des valeurs internes de l'objet, et peut ainsi effectuer une interaction avec celui-ci.

Ce type de connaissance peut ensuite être utilisée par l'humanoïde afin de déclencher certaines sous-procédures comportementales, comme se rapprocher d'un objet, ou encore lancer des interactions.

Afin de permettre à l'humanoïde d'acquérir de nouvelles connaissances, nous proposons un système basé sur l'observation. Ainsi, l'humanoïde dispose d'une procédure utilisant les *PVS* de l'environnement qui lui permet de rafraîchir aussi bien la date de sa base de connaissances topologique que sa liste d'objets visibles. La procédure parcours les *PVS* de l'environnement immédiat de l'humanoïde en intersectant les frustra correspondant avec le frustrum représentant son champ de vision. Chaque nœud traversé correspondant à une bordure intergroupe, la mise à jour de la connaissance topologique est donc triviale et implicite. Pour ce qui est de la connaissance associée aux objets, leur position dans l'environnement est simplement testée afin de déterminer s'ils se trouvent dans le frustrum courant, et si c'est le cas, leur état est déterminé suivant la distance relative à l'humanoïde.

4.3. Planification de chemin

Voyons maintenant comment l'humanoïde va gérer son déplacement au sein de l'environnement, et en premier lieu comment il peut planifier un chemin [PDB06b]. La planification de chemin consiste à prévoir son parcours au sein d'une représentation topologique de l'environnement, en prenant en considération divers critères conditionnant le coût du déplacement. La littérature fait état de bon nombre de ces critères, comme la distance [Gol95], mais aussi les changements de direction [HK05], l'impact cognitif [DK03], les densités de populations [ST05], ou encore un facteur de stress [Osa04]. Par contre, vraiment peu d'heuristiques de planification tiennent compte de plusieurs de ces critères à la fois.

4.3.1. Heuristique multicritères

Ce que nous proposons est une évaluation du chemin par une comparaison du temps de parcours ressenti (prévu) par l'entité. Ainsi, chaque critère va être converti en une évaluation temporelle quantitative (critères de distance, de densité de population) ou qualitative (critères de changements de direction, de largeur de chemin, de sens des flux de population, ou encore de potentiel de découverte). Cette évaluation temporelle peut être vue comme une succession de filtres :

$$coût = \sum_{i=1}^{n} P_i \times V_i$$

avec P_i le degré de préférence du filtre, et V_i la valeur du filtre ($V_i \in \{[0;1], infini\}$). La valeur du filtre V_i est ici obtenue par un calcul global (i.e. indépendant de l'entité) réalisé sur l'environnement. Le degré de préférence est quant à lui spécifique à chaque entité, et peut être positif (le critère est défavorable), négatif (le critère est favorable), ou nul (le critère est ignoré).

4.3.2. Algorithme hiérarchique

Maintenant que nous pouvons quantifier le coût de parcours le long du chemin, il nous est possible d'appliquer des algorithmes de parcours de graphe sur notre environnement, que ce soit le bien connu A^* [BMS04] pour une destination unique, ou encore le *flood-fill* [LRDG90] pour des destinations multiples. Comme notre graphe de représentation est hiérarchique, nous proposons une méthode d'évaluation par paliers. Tout d'abord le chemin est intégralement calculé sur la couche la plus abstraite du graphe, permettant ainsi une

évaluation certe approximative mais aussi trés rapide. Ensuite, ce chemin est raffiné localement dans les deux couches suivantes au fur et à mesure que l'entité en a besoin pour se déplacer (figure 9). Une telle technique a deux avantages

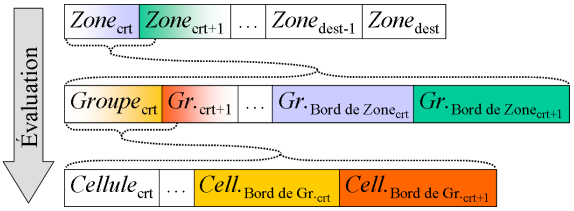


Figure 9: *Principe de planification hiérarchique*

principaux : elle permet de répartir la charge de calcul dans le temps, et aussi de limiter la perte de calcul lorsqu'un chemin est invalidé.

4.3.3. Conditions de fin multiples

En ce qui concerne l'expression de la condition de fin de la planification, nous proposons trois possibilités. Premièrement, une condition de fin classique donnant une localisation à atteindre dans l'espace. Deuxièmement, une condition de fin plus conceptuelle, visant à améliorer la connaissance de l'humanoïde sur son environnement, et donc exprimée comme étant n'importe quelle zone de l'environnement partiellement connue. Troisièmement, une condition de fin régissant la procédure de décision de l'humanoïde, exprimée comme étant l'obtention d'un comportement interactif spécifique. Cette dernière condition de fin va permettre à l'humanoïde de choisir dans son environnement l'objet interactif le plus approprié en tenant compte de sa localisation dans l'espace, ainsi que des temps d'attente éventuels. L'intérêt majeur de cette dernière condition est de corréler la représentation de l'espace, le temps, et l'interaction dans la procédure de décision de l'humanoïde. Notons que cette dernière condition de fin permet bien de choisir un objet interactif parmis plusieurs proposant une interaction déterminée, mais ne s'occupe pas de choisir l'enchaînement des interactions conduisant à un but de plus haut niveau. En résumé, cet algorithme est capable d'effectuer une tâche cognitive identifiée, mais nécessite encore les données d'une couche rationnelle externe pour ce qui est de la prise de décision de plus haut niveau.

5. Résultats

L'ensemble du modèle que nous avons présenté est intégré à une application de simulation (figure 10). Cette application est scindée en quatre parties.

Premièrement, la définition des types d'objets grâce à une interface dédiée pour BIIO. Cette partie, bien que prévue pour faciliter au maximum la tâche du programmeur, est

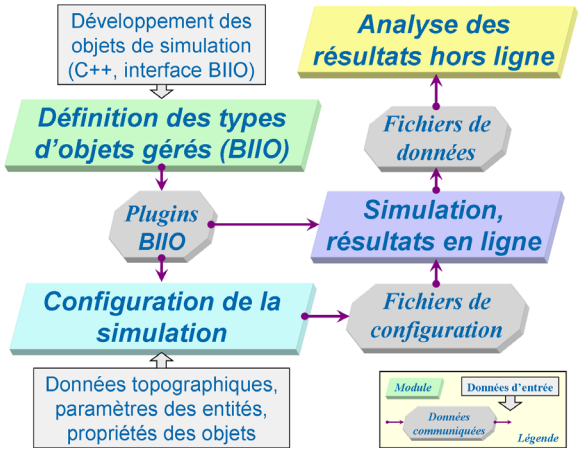


Figure 10: *Schéma de l'application de simulation*

tout de même dédiée à des informaticiens car elle nécessite la programmation des comportements. L'ensemble des objets décrits dans cette partie sont ensuite disponibles sous la forme de plugins pour une utilisation future. Pour le moment, les objets spécifiques aux procédures comportementales des voyageurs d'une gare sont disponibles : distributeur automatique de billets, composteur, valideur automatique de billets, tableau général des départs, et bien entendu le voyageur.

Deuxièmement, la configuration de la simulation permet à un utilisateur non initié de spécifier la topologie de l'environnement par l'import d'un fichier AutoCAD. Ce fichier AutoCAD permet à des architectes de spécifier non seulement la géométrie des lieux, qui est ensuite subdivisée et abstraite de façon totalement automatique, mais aussi de placer les objets statiques de l'environnement. L'utilisateur peut ensuite configurer l'ensemble des paramètres de ces objets interactifs.

Troisièmement, la simulation à proprement parler, ainsi que les résultats en ligne, tel que la visualisation 3D (figure 11) ou encore une carte de densité dynamique. Notons que l'utilisateur peut interagir avec la simulation en cours en modifiant les paramètres des objets.

Quatrièmement enfin, les résultats hors ligne sont calculés à partir de données récupérées lors de la simulation et permettent d'émettre des graphes statistiques sur des périodes données, ou encore de comparer plusieurs simulations entre elles.

Concernant les coûts mémoire associés à chaque partie du modèle, la représentation de l'environnement est de l'ordre du méga octet (20Mo pour la gare de St Denis), tandis qu'un humain virtuel est de l'ordre du kilo octet (1Ko pour St Denis). Pour ce qui est des coûts de calcul, l'initialisation de l'environnement est de l'ordre de la seconde (4s pour St Denis), les calculs dynamiques de l'environnement sont de l'or-

(a) Interaction automatique avec des valideurs de billet traversables

(b) File d'attente devant un composteur

Figure 11: *Visualisation 3D de la simulation*

dre de la milliseconde, et enfin la procédure de planification de l'ordre de la microseconde (500μs pour St Denis). Notons que tous ces coûts sont directement dépendants de la complexité géométrique (et donc du nombre de zones abstraites) de l'environnement.

Des études expérimentales sont en cours d'élaboration afin de valider les résultats obtenus. Ces études visent deux points. Premièrement, extraire par analyse du réel les degrés de préférence de l'heuristique de planification pour différentes catégories de populations, allant de la personne pressée au chaland. Deuxièmement, valider par comparaison au réel les résultats de nos simulations. Cette étape de validation ne se fera bien entendu pas par comparaison des trajectoires microscopiques, mais plutôt par une validation visuelle du comportement macroscopique émergent par des experts de l'étude de flux, ainsi que par une comparaison des données de flux de personnes avec des tables disponibles dans la littérature (commes celles de J.J. Fruin) et relevées sur le terrain.

6. Conclusion

Le modèle que nous avons présenté forme une architecture de simulation d'humains virtuels. Il couvre l'ensemble des domaines nécessaires à une telle simulation. Tout d'abord, nous avons proposé une représentation de l'environnement extraite de façon totalement automatique, et proposant un grand nombre de précalculs utilisables par la suite à faible coût. Nous avons ensuite doté cet environnement d'un ensemble d'informations visant à améliorer les possibilités comportementales de nos humanoïdes simulés, grâce à notre moteur de description des interactions qu'est BIIO. Ce dernier permet une modélisation hiérarchique des interactions, et aussi de faire le lien entre la représentation topologique et les procédures comportementales de l'humanoïde. Pour finir, notre modèle d'humanoïde est doté d'une connaissance topologique visant à supprimer l'aspect omniscient des entités, ainsi que d'une connaissance volatile lui permettant de caractériser les objets interactifs de sont environnement immédiat. Concernant la procédure de planification de chemins, elle associe un grand nombre de critères quantitatifs et qualitatifs permettant d'introduire une certaine variété dans la population simulée. De plus, elle a comme principal avantage de permettre une sélection des objets interactifs en tenant compte des aspects temporels et topologiques. Une telle sélection peut ainsi être utilisée comme base décisionnelle pour une procédure comportementale de plus haut niveau, telle qu'une procédure rationnelle. Concernant les résultats, notre modèle est en cours d'évaluation et de réglage. En effet, il dispose d'un nombre conséquent de paramètres, notament concernant les coûts de planification, rendant sa validation un sujet d'étude à part entière.

7. Perspectives

Nous avons plusieurs perspectives d'évolution concernant notre modèle de simulation. Tout d'abord, nous voudrions compléter le modèle de l'humanoïde en y intégrant une procédure d'évitement des collisions entre humanoïdes plus efficace. En effet, nous utilisons actuellement le modèle d'Helbing [HFV00] qui comporte trois points faibles d'un point de vue réalisme : l'oscillation de la direction de déplacement en réaction à une collision, l'absence totale de prédiction de collision, et enfin la non prise en charge de l'aspect social (du point de vue des règles de comportement aussi bien que de la formation de groupes). Ensuite, comme évoqué précédemment, nous voulons lancer une batterie de tests des simulations en identifiant des mises en situation permettant premièrement d'identifier certains critères de planification, afin de pouvoir configurer finement notre modèle, et ensuite de valider ce modèle par la comparaison avec le réel. Ainsi, nous pourrons caractériser les importances relatives de nos critères de simulation, et donc proposer des catégories de population pré-configurées (comme un homme d'affaire, un touriste, une personne pressée, ou

encore se promenant, etc.). Nous aimerions aussi connecter notre moteur de simulation à un moteur d'animation des interactions entre les humanoïdes et les équipements afin d'en améliorer l'aspect visuel, même si cela n'a que peu d'importance en ce qui concerne l'extraction de résultats. Pour finir, nous voudrions raffiner encore notre modèle d'agent autonome afin qu'il prenne en compte la signalétique dans l'environnement, que ce soit pour récupérer de l'information nécessaire à l'enchaînement de ses comportements, ou encore pour s'orienter.

Références

[ACT06] ABACI T., CIGER J., THALMANN D. : Planning with smart objects. In *International Conferences in Central Europe on Computer Graphics, Visualization and Computer Vision* (janvier 2006).

[AM05] ALI W., MOULIN B. : 2d-3d multiagent geosimulation with knowledge-based agents of customers' shopping behavior in a shopping mall. In *COSIT 2005*, Kohn A., Mark D., (Eds.), vol. 3693 de *LNCS*. Springer-Verlag, 2005, pp. 445–458.

[AVF04] ANDÚJAR C., VÁZQUEZ P., FAIRÉN M. : Wayfinder : guided tours through complex walkthrough models. *Computer Graphics Forum, Eurographics'04* (2004).

[BMS04] BOTEA A., MÜLLER M., SCHAEFFER J. : Near optimal hierarchical path-finding. *Journal of Game Development. Vol. volume 1* (2004).

[DK03] DUCKHAM M., KULIK L. : "Simplest Paths" : Automated route selection for navigation. In *Spatial Information Theory : Foundations of Geographic Information Science* (Berlin : Springer, 2003), Kuhn W., Worboys M. F., Timpf S., (Eds.), vol. 2825 de *Lecture Notes in Computer Science*, pp. 182–199.

[Fru71] FRUIN J. J. : Pedestrian planning and design. New-York : Metropolitan Association of Urban Designers and Environmental Planners, Inc., 1971.

[Gib97] GIBSON J. : The theory of affordances. In *Perceiving, acting and knowing - Towards an ecological psychology* (1997), et J. Bransford R. S., (Ed.), Houghton-Mifflin, pp. 67–82.

[Gol95] GOLLEDGE R. : Path selection and route preference in human navigation : A progress report. In *Spatial Information Theory : A Theoetical Basis for GIS* (Berlin : Springer, 1995), Frank A., Kuhn W., (Eds.), vol. 988 de *Lecture Notes in Computer Science*, pp. 207–222.

[GSN04] GLOOR C., STUCKI P., NAGEL K. : Hybrid techniques for pedestrian simulations. In *4th Swiss Transport Research Conference* (Monte Verità, Ascona, 2004).

[HFV00] HELBING D., FARKAS I. J., VICSEK T. : Simulating dynamical features of escape panic. *Nature. Vol. 407* (2000), 487–490.

[HK05] HOCHMAIR H. H., KARLSSON V. : Investigation of preference between the least-angle strategy and the initial segment strategy for route selection in unknown environments. *Lecture Notes in Computer Science. Vol. 3343* (2005), 79 – 97.

[KT99] KALLMANN M., THALMANN D. : Direct 3d interaction with smart objects. In *VRST '99 : Proceedings of the ACM symposium on Virtual reality software and technology* (New York, NY, USA, 1999), ACM Press, pp. 124–130.

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

[LD04] LAMARCHE F., DONIKIAN S. : Crowds of virtual humans : a new approach for real time navigation in complex and structured environments. *Computer Graphics Forum, Eurographics'04* (2004).

[LRDG90] LENGYEL J., REICHERT M., DONALD B. R., GREENBERG D. P. : Real-time robot motion planning using rasterizing computer graphics hardware. In *SIGGRAPH '90 : Proceedings of the 17th annual conference on Computer graphics and interactive techniques* (1990), ACM Press, pp. 327–335.

[MB75] MELINEK S. J., BOOTH S. : An analysis of evacuation times from buildings. In *CIB Symposium on the control of smoke movement in building fires* (Watford, U.K., 1975), Building research establishment, Fire research station.

[Osa04] OSARAGI T. : Modeling of pedestrian behavior and its applications to spatial evaluation. In *AAMAS '04 : Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems* (Washington, DC, USA, 2004), IEEE Computer Society, pp. 836–843.

[Pau84] PAULS J. : Development of knowledge about means of egress. In *Fire technology*, vol. 20. mai 1984, ch. 2, pp. 28–40.

[PDB05] PARIS S., DONIKIAN S., BONVALET N. : Towards more realistic and efficient virtual environment description and usage. In *First International Workshop on Crowd Simulation (V-Crowds'05)* (2005), VRlab, EPFL.

[PDB06a] PARIS S., DONIKIAN S., BONVALET N. : Behavioural interactive and introspective objects for crowd simulation. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation (poster session)* (sep 2006).

[PDB06b] PARIS S., DONIKIAN S., BONVALET N. : Environmental abstraction and path planning techniques for realistic crowd simulation. *Computer Animation and Virtual Worlds* (2006), 325–335.

[PM78] PREDTECHENSKII V. M., MILINSKII A. I. : *Planning for foot traffic flow in buildings*, National Bureau of Standards ed. Amerind Publishing CO Pvt Ltd, New Dehli, Inde, 1978.

[SKG05] SUNG M., KOVAR L., GLEICHER M. : Fast and accurate goal-directed motion synthesis for crowds. In *Eurographics/ACM SIGGRAPH Symposium on Computer Animation* (2005), Anjyo K., Faloutsos P., (Eds.), pp. 291–300.

[ST05] SHAO W., TERZOPOULOS D. : Environmental modeling for autonomous virtual pedestrians. *Digital Human Modeling for Design and Engineering Symposium* (2005).

[TD03] THOMAS R., DONIKIAN S. : A model of hierarchical cognitive map and human memory designed for reactive and planned navigation. In *4th International Space Syntax Symposium* (Londres, juin 2003).

[Tog55] TOGAWA K. : *Study on fire escapes based on observations of multitude currents*. Tech. rep., Ministry of construction, 1955.

[TTK88] TAKAHASHI K., TANAKA T., KOSE S. : An evacuation model for use in fire safety design of buildings. In *Fire Safety Science Proceedings of the Second International Symposium* (Tokyo, juin 1988), Hemisphere Publishing Company, New York, pp. 551–560.