Modélisation de l'évacuation d'un immeuble en cas d'incendie

FAULHABER Guillaume
ROY Vincent

June 17, 2018

Abstract

When an issue arises in a building, people's evacuation is a recurring challenge. We wondered whether we could make a realistic simulation of people's evacuation based on a simple physical model. First, we elaborated this model and then we simulate the people's comportment on MATLAB. We could conclude that our simulation is enough to describe the general comportment of people.

Keywords— evacuation, MATLAB, multi-agent, comportment, building

Nous adressons nos remerciements à notre professeur de Signaux et Systèmes ainsi que notre chargée de projet, Madame Cristina Maniu, qui nous a donné de précieux conseils pour la réalisation de ce projet.

Table des matières

1 Introduction

A la fin du 20ème siècle, l'Etat français a commencé à s'intéresser à l'évacuation des bâtiments. En 2008, un décret est d'ailleurs signé afin de rendre obligatoire les exercices d'évacuation de ces derniers. Néanmoins, cela prend du temps aux entreprises et les exercices sont souvent peu efficaces : les personnes ne se sentent pas concernés. En effet, il n'y pas de fumée, pas de flamme et par conséquent les personnes ne ressentent pas de stress. Il est donc impossible de mesurer par la pratique le temps d'évacuation d'un bâtiment avec les moyens actuels.
Cependant, la mise en sécurité des occupants d'un bâtiment vers un lieu sûr, à l'abri des effluents du feu, est un objectif essentiel en cas d'incendie. Aujourd'hui, l'informatique permet de simuler le comportement humain. De plus, la création d'un environnement complexe est aussi possible avec des logiciels informatiques.

Des étudiants de l'école CentraleSupélec avaient déjà simulé un système multi-agents sur une carte [1]. Le système était composé de deux types de personnes, les yakuzas et les policiers. Les yakuzas devaient déplacer la drogue entre deux points et la police devait attrapait les yakuzas. Il utilisait la méthode des potentiels. Dans un premier temps, nous avons utilisé ce principe, mais après des recherches bibliographiques, nous avons remarqué que certains auteurs avaientt travaillé sur des sujets similaires en utilisant une équation du mouvement pour décrire le comportement des agents [2]. Nous avons alors utilisé une méthode sans potentiels mais avec des forces d'intéractions.

Dans ce projet, nous désirons simuler informatiquement, à l'aide du logiciel MATLAB, une évacuation de personnes dans un immeuble lors d'un incendie. Il s'agit donc d'un système multi-agents en faisant intervenir le comportement humain. C'est pourquoi nous prendrons en compte des phénomènes tel que le stress des agents ainsi que l'interaction entre les agents et le feu. Nous allons nous appuyer sur les travaux réalisés par Dirk Helbing [3] pour réaliser l'équation du mouvement d'un agent. Dans son modèle, il fait intervenir des forces sociales. Après avoir simulé l'évacuation pour un seul agent, nous allons nous intéresser à une simulation fondée sur un système multi-agents, c'est-à-dire entre agents pouvant interagir entre eux. Ce sujet fait intervenir des notions d'informatique, de signaux et systèmes ainsi que de mécanique.

Pouvons-nous, à partir d'un modèle physique simple, simuler de manière réaliste une évacuation d'un immeuble ?

Notre objectif est ainsi de simuler une réelle évacuation et d'estimer sa durée. Dans un premier temps, nous exposerons le modèle physique utilisé. Grâce à ce modèle, nous pourrons coder sous MATLAB l'évacuation des agents, avec une programmation orientée fonction. Enfin, nous simulerons notre algorithme, d'abord avec un seul agent puis avec un système multi-agents.

2 Modélisation du problème

2.1 Cahier des charges

Nous désirons faire une modélisation en 2 dimensions du problème, c'est-à-dire représenter une évacuation sur un étage d'un bâtiment. Nous allons dans un premier temps ne pas différencier les individus, c'est-à-dire attribuer les mêmes caractéristiques physiologiques à chaque agent. Ensuite, nous essayerons de prendre en compte de les différencier. De plus, nous souhaitons prendre en compte les interactions entre les agents, interactions modélisées par des forces répulsives. Les agents ne devront également pas traverser les murs, ces derniers doivent donc être rigides. Enfin, nous essayerons de modéliser la fumée dégagée par l'incendie ainsi que le stress des agents, afin d'être le plus réaliste possible.

2.2 Modèle physique utilisé

Nous allons appliquer le principe fondamental de la dynamique sur chaque agent i (équation 1). Nous utilisons le modèle de forces sociales expliqué par Helbing [3]. Le modèle suggère que le mouvement des agents peut être décrit comme s'ils étaient soumis à des "forces sociales". Dans le modèle présenté, la force motrice est une force interne à l'individu, elle est crée par nos muscles, elle se traduit par un terme décrivant l'accélération de l'agent vers la vitesse de mouvement désirée. La force liée aux interactons entre agents indique que l'agent veut se tenir à une distance raisonnable des autres agents. La force d'interaction avec les murs représente le fait que les agents veulent éviter les murs.

$$m_i \frac{\partial \vec{v}_i(t)}{\partial t} + \vec{F}_{motrice}(t) = \vec{F}_{autres\ Agents}(t) + \vec{F}_{mur}(t) \tag{1}$$

Plus précisément, l'équation du mouvement s'écrit ainsi :

$$m_i \frac{\partial \vec{v}_i(t)}{\partial t}(t) = \frac{\vec{v}_i^d(t) - \vec{v}_i(t)}{\tau_i} + \sum_j^{n-1} \vec{f}_{ij}(t) + \sum_w^W \vec{f}_{iw}(t) \tag{2}$$

Terme	Description	Valeur
m_i	masse de la particule i	80 kg
$\vec{v}_i^d(t)$	vitesse désirée	1.5 $m.s^{-1}$
$\vec{v}_i(t)$	vitesse de la particule i à t	
τ_i	paramètre de relaxation qui fixe l'intensité de la force motrice	$0.5s$
$\vec{f}_{ij}(t)$	force d'interaction entre i et j	
n	le nombre d'agents	
$\vec{f}_{iw}(t)$	force d'interaction entre i et le mur w	
W	le nombre de murs	

Table 1: Description des termes de l'équation du mouvement.

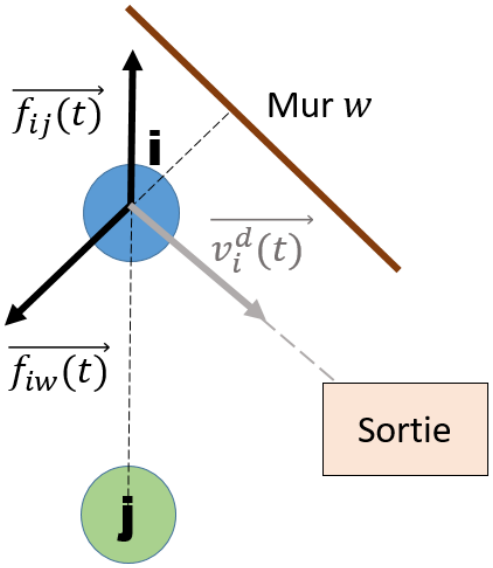

Figure 1: Représentation des forces qui s'exercent sur l'agent i.

On explicite à présent la forme de la force $\vec{f}_{ij}$ (équation 3). A noter que l'expression de la force $\vec{f}_{iw}$ est similaire.

$$\vec{f}_{ij}(t) = \vec{F}_{socio_psychologique}(t) + \vec{N}_{compression}(t) + \vec{T}_{compression}(t) \tag{3}$$

Plus précisément,

$$\vec{f}_{ij}(t) = A \exp \frac{r_{ij} - d_{ij}}{B} \vec{n}_i j + C\, g(r_{ij} - d_{ij})\, \vec{n}_{ij} + D :\ g(r_{ij} - d_{ij}) \Delta v_{ij}^t\, \vec{t}_{ij} \tag{4}$$

Le premier terme représente la force sociale qui agit de telle sorte que l'agent i a tendance à rester à une certaine distance des autres agents j. Les deuxième et troisième termes représentent respectivement la résistance normale et tangentielle à la compression.

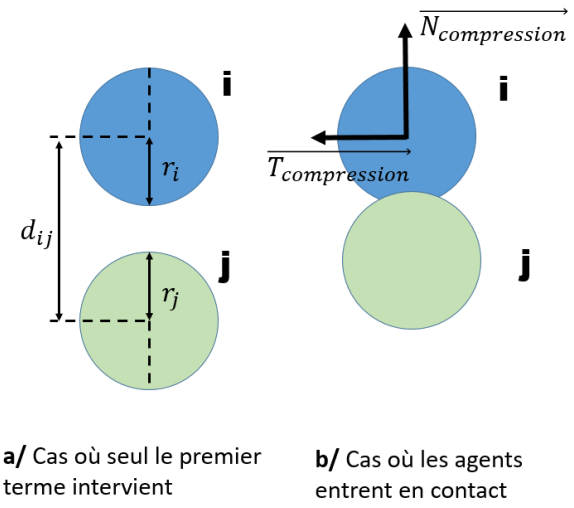

Figure 2: Représentation de la force $\vec{f}_{ij}$.

Terme	Description	Valeur
A	constante	$2e3 \quad N$
$r_i = r_i + r_j$	somme des rayons des agents i et j	$0.6 \quad m$
d_{ij}	distance entre le centre de i et j	
B	constante	$0.008 \quad m$
$\vec{n}_{ij}$	vecteur unitaire allant de j vers i	
C	constante	$1.2 \quad kg.s^{-2}$
D	constante	$2.4e5 \quad kg.m^{-1}.s^{-1}$
Δv_{ij}^t	la différence de vitesse de j par rapport à i	
$\vec{t}_{ij}$	vecteur unitaire tangent à $\vec{n}_{ij}$	
$g(x) = \begin{cases} 0 & si \quad x \leq 0 \\ x & sinon \end{cases}$	la fonction "Macauly bracket"	

Table 2: Description des termes intervenant dans $\vec{f}_{ij}$.

2.3 Prise en compte du stress

Nous sommes partis de la théorie de Henri Laborit sur le syndrome général d'adaptation. Cette théorie décrit la résistance au stress en 3 phases. La phase d'alerte où l'adrénaline monte et où la stress est positif, celle de résistance dans laquelle l'individu contient le stress et enfin celle d'épuisement : l'individu craque jusqu'au burn-out. La durée de ces phases dépend de l'intensité du stress.

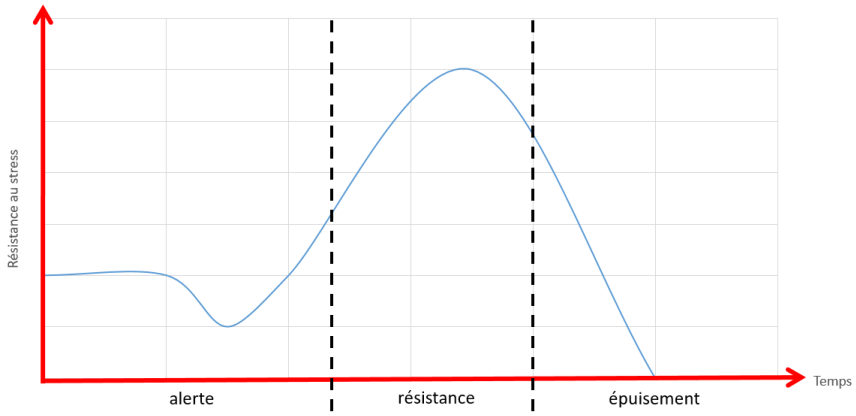

Figure 3: Représentation de la résistance au stress en fonction du temps.

Nous avons décidé de traduire la Figure 3 par l'équation 5, où f_{stress} est la fonction stress dépendant du temps et est représentée sur la figure 4.

$$\vec{v}_i^d(stress) = \vec{v}_i^d(sans\ stress)\ f_{stress}(t)\ \vec{Rot}(\vec{v}_i^d(sans\ stress), t) \tag{5}$$

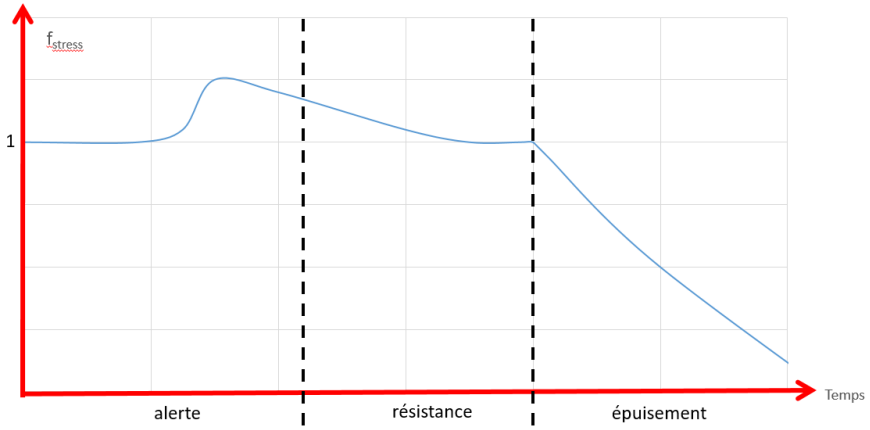

Figure 4: Représentation de la fonction f_{stress}.

Le stress agit sur la direction de l'individu, c'est-à-dire sur les choix qu'il fait quand il court entre deux chemins, ainsi que sur sa vitesse. Dans un premier temps, le stress améliore les capacités physiques et de concentration de l'individu : il court plus vite et dans la bonne direction. Ensuite, ses capacités commencent à décroire jusqu'à son état initial. Au moment où l'individu entre dans la phase d'épuisement, il commence à faire des choix déraisonnés et court de plus en plus lentement. A l'épuisement total, il finit par marcher dans une direction aléatoire.

2.4 Prise en compte de la fumée

Prenons maintenant en compte la fumée de l'incendie. Le modèle de Frantzich et Nilsson [5] montre que la vitesse de marche des agents diminue avec la présence de la fumée. La fumée est modélisée par un coefficient d'extinction noté K_s. La vitesse des agents diminue avec K_s jusqu'à atteindre une vitesse critique. Cette vitesse est arbitraire mais on peut considérer qu'elle vaut 10% de la vitesse initiale. En effet, nous voulons que, pour notre modélisation, les agents ne cessent d'avancer.

Terme	Description	Valeur
$\vec{v}_{i,min}$	la vitesse initiale	$0.10\,\vec{v}_{initiale}$
α	paramètre expérimental	$0{,}706\ ms^{-1}$
β	paramètre expérimental	$-0{,}057\ m^2s^{-1}$

Table 3: Description des termes intervenant dans $\vec{v}_i^d(fumee, t)$.

$$\vec{v}_i^d(fumee, t) = \max\{\ \vec{v}_{i,min}\ ,\ \vec{v}_i^d(t)\ (1 + \frac{\beta}{\alpha}K_s)\ \} \tag{6}$$

3 Programmation sous MATLAB

3.1 Présentation générale de l'algorithme

Nous avons opté pour une programmation orientée fonction mais nous avons quand même choisi de faire une représentation de diagramme de packages UML (figure 5) pour faciliter la compréhension du programme.

Ce dernier initialise la matrice position et vitesse des agents $serverMat$, puis initialise la carte grâce à la fonction $setEnvironment$.

Ensuite, nous pouvons coder les termes de droite de l'équation du mouvement, à savoir f_{ij}, f_{iw} et $vitesseDesiree$. A noter que $vitesseDesiree$ se code en plusieurs fonctions, qui dépendent des murs de la carte.

Une fois les différentes forces codées, nous pouvons implémenter l'équation du mouvement grâce à la méthode d'approximation d'Euler puis celle d'Heun. Nous utilisons pour cela les fonctions $UpdatePosEuler$ et $UpdatePosHeun$.

Dans le programme $Main$, nous initialisons les différentes constantes et nous itérons $UpdatePosHeun$ jusqu'à la disparition des agents sur la carte.

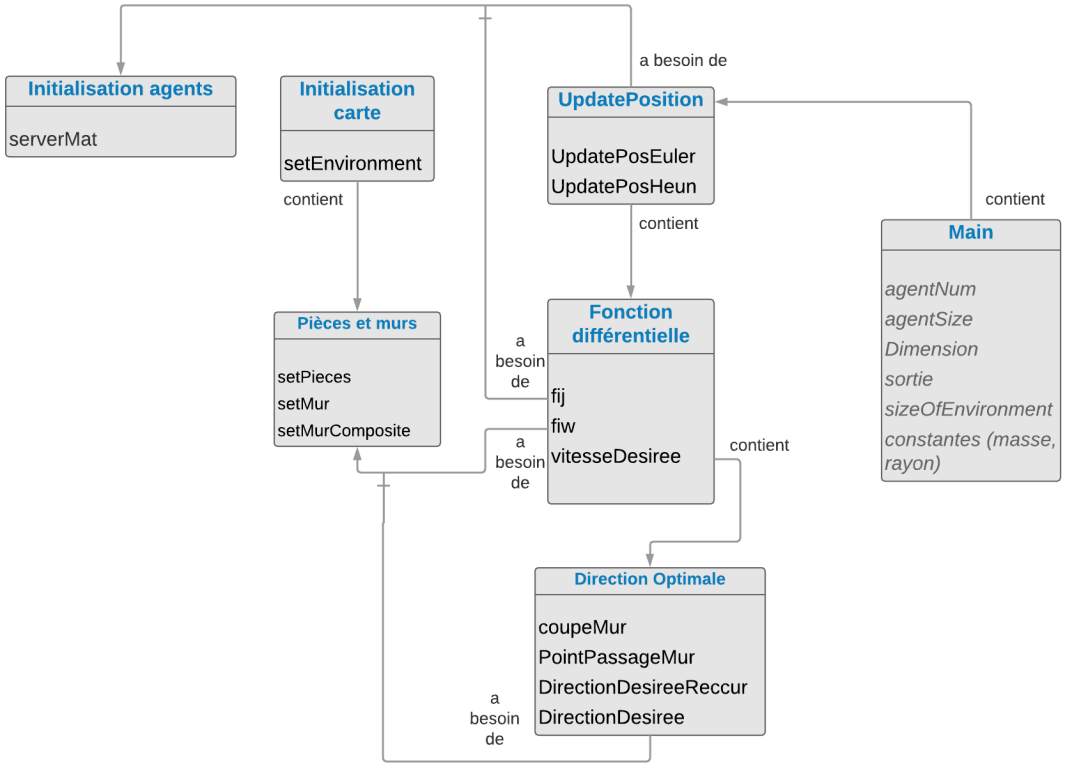

Figure 5: Explication sous forme de diagramme de package UML du programme MATLAB

3.2 Méthodes d'approximation

Nous utilisons deux méthodes d'approximation : la méthode d'Euler et la méthode de Heun (Table 5). La méthode de Heun nous permets d'être plus précis pour une complexité comparable à celle d'Euler. De plus, la méthode d'Euler est instable, ce qui n'est pas le cas de celle de Heun. Nous présenterons un tableau comparant les résultats de nos deux méthodes dans la section suivante. Néamoins voici le détail sur ces méthodes (h étant le pas de discrétisation) :

$t_{i+1} = t_i + h$		
	Méthode d'Euler	Méthode de Heun
Vitesse	$\vec{v}_{i+1}^{\,Euler} = \vec{v}_i + h\, f(t_i, \vec{v}_i)$	$\vec{v}_{i+1} = \vec{v}_i + \frac{h}{2} \{ f(t_i, \vec{v}_i, \vec{r}_i) + f(t_{i+1}, \vec{v}_{i+1}^{\,Euler}, \vec{r}_{i+1}^{\,Euler}) \}$
Position	$\vec{r}_{i+1}^{\,Euler} = \vec{r}_i + h\, \vec{v}_i$	$\vec{r}_{i+1} = \vec{r}_i + \frac{h}{2} \{ \vec{v}_i + \vec{v}_{i+1}^{\,Euler} \}$

Table 4: Description des 2 méthodes d'approximation

3.3 Vitesse désirée

Positionnement du problème :

Soit en rouge la sortie et considérons un agent i (représenté en bleu sur la figure 6). En premier lieu, on a implémenté la vitesse désirée comme ceci : la direction de cette vitesse était égale à chaque itération à celle de la sortie. Seulement, si il y a présence d'un mur entre l'agent et la sortie, le programme ne marche pas car l'agent reste coincé derrière le mur (figure 6).

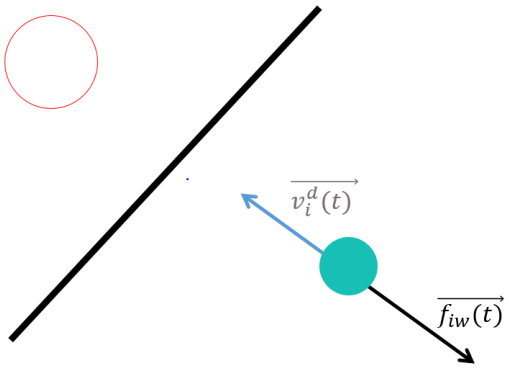

Figure 6: Présence d'un mur entre l'agent et la sortie : disfonctionnement du programme

Nous allons donc construire un programme récursif qui permet à $VitesseDesiree$ de changer de direction à chaque itération du programme. Les figures ci-dessous expliquent le principe du programme, en présence d'un mur.

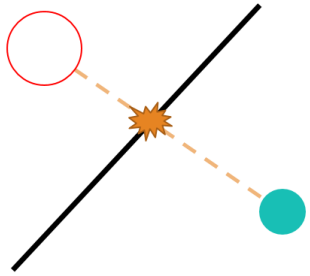

Figure 7: L'agent rencontre un mur

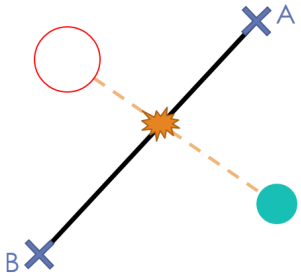

Figure 8: Le programme mémorise les 2 extrémités de ce mur

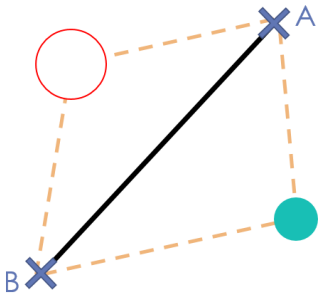

Figure 9: Tracé des 2 chemins possibles

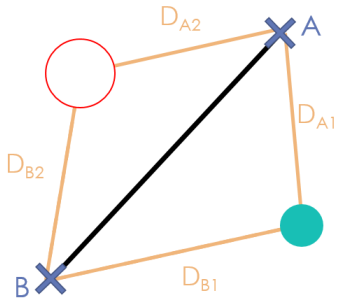

Figure 10: Calcul des 2 distances

Ici, on garde ainsi le trajet A car il est plus court que le trajet B :

$$D_A = D_{A1} + D_{A2} \tag{7}$$

$$D_B = D_{B1} + D_{B2} \tag{8}$$

$$D_A < D_B \tag{9}$$

Essayons à présent la nouvelle méthode pour $VitesseDesiree$ cette fois avec 2 murs.

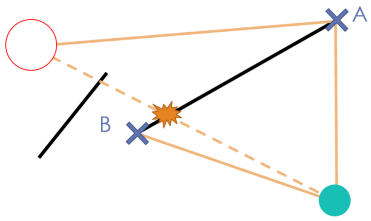

Figure 11: La première étape est la même

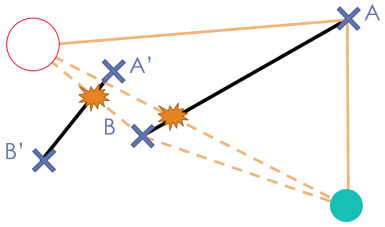

Figure 12: Le programme mémorise les 2 extrémités du 2^{me} mur

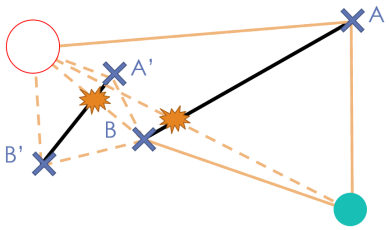

Figure 13: On trace les chemins possibles

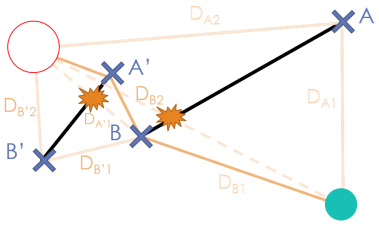

Figure 14: Calcul de la distance la plus courte

Lorsque l'agent arrive au point B, il ne peut plus cette-fois-ci se diriger directement vers la sortie (figure 11). Il est en effet heurté par un second mur. On recommence alors notre programme de $VitesseDesiree$. Le programme mémorise alors 2 autres sommets, A' et B' (figure 12). Ensuite, à partir de A' et de B', il peut enfin se diriger vers la sortie (figure 13). A la fin, le programme choisit le chemin le plus court parmi tous les chemins possibles. Ce chemin est représenté en couleur plus foncée sur la figure 14).

4 Simulation sous MATLAB

4.1 Simulation avec un agent

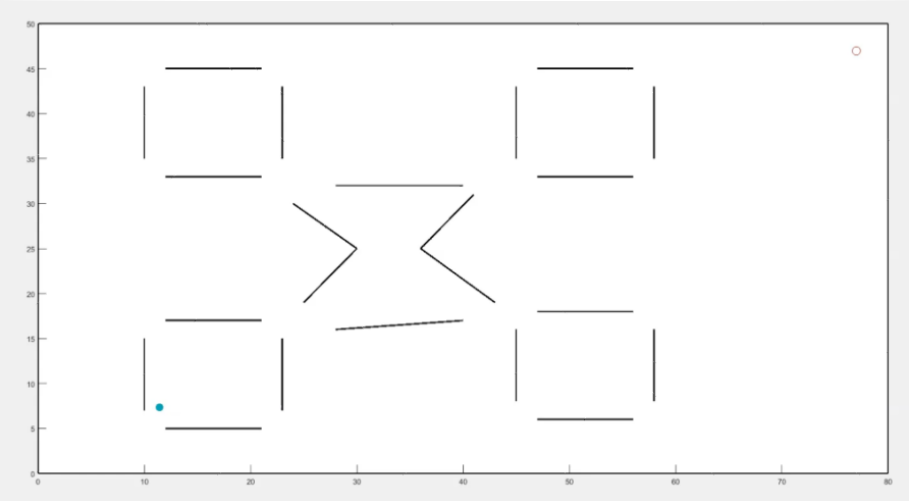

Figure 15: Représentation de la carte avec un agent

Lors de la simulation ci-dessus, nous remarquons que l'agent se dirige bien vers la sortie et passe par des points remarquables au bout des murs. La simulation nous a aussi permis de mettre en évidence les effets dynamiques, en particulier l'accélération de l'agent et l'interaction entre agent et murs. Cette simulation représente le *scnario*1 du programme.

4.2 Simulation multi-agents

Nous faisons à présent une simulation multi-agents. Nous avons représenté un étage d'un bâtiment quelconque avec plusieurs pièces et des couleurs. La sortie est en rouge. Il s'agit du *scnario*2 du programme.

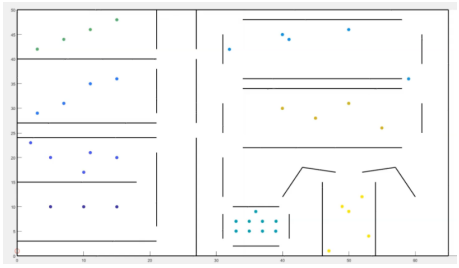

Figure 16: Début de simulation

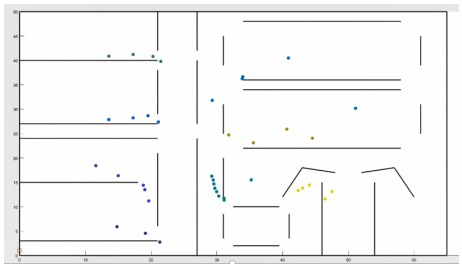

Figure 17: Les agents sortent des pièces...

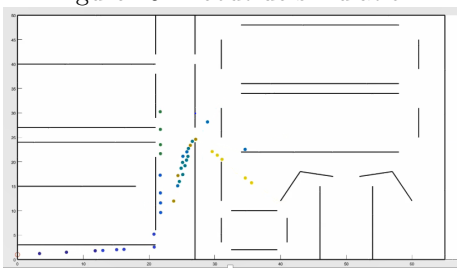

Figure 18: ...En se dirigeant vers la sortie

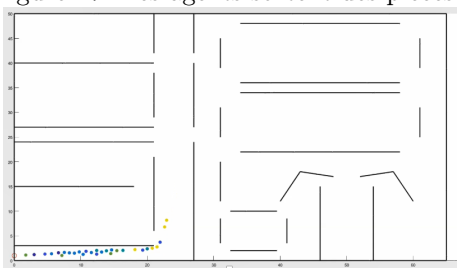

Figure 19: Quasiment tous les agents sont sortis

Comparons à présent les durées d'évacuation des 2 méthodes : celle d'Euler et celle d'Heun. Pour la simulation, nous prenons un pas $h = 0.002\ s$, un nombre d'agent $n = 39$ et une carte similaire à celle représentée en figure 16.

	Méthode d'Euler	Méthode de Heun
Temps d'évacuation (en s)	342.15	341.85
Itérations		4 itérations de moins que Euler

Table 5: Comparaison de la durée d'évacuation pour les méthodes d'Euler et de Heun

5 Conclusion et ouverture

Nous avons donc vu qu'il était possible de transcrire le comportement global des agents. Néanmoins, nous n'avons pas réussi à implémenter sous MATLAB tout ce que nous avion prévu - à savoir le stress et l'interaction avec la fumée - dans la modélisation en raison d'un manque de temps. De plus, pour retranscrire le comportement individuel et la diversité des agents, il nous faudrait repenser le programme. En effet, nous devrions passer d'une programmation fonctionnelle à une programmation orientée objet. A l'aide de classe, nous arriverions de manière simple à ces résultats.A la fin du projet, nous avons compris que la programmation orientée objet serait plus simple, mais nous avions bien avancé de manière fonctionnelle et ne voulions pas changer. Nous envisageons donc de faire une version β du programme par la suite.

Le projet permet, en outre, d'arriver à certaines conclusions notamment par l'estimation du temps de sortie d'individu dans un bâtiment. Un prochain objectif pourrait être de programmer la carte d'un étage de la résidence de l'école CentraleSupélec et calculer ainsi le temps d'évacuation de cet étage, en fonction du nombre d'individus qui s'y trouvent. La création d'une interface graphique nous paraît d'ailleurs nécessaire pour faciliter la génération de cartes.

Ce projet, à la croisée de l'automatique, du génie logiciel et de la mécanique, nous a permis d'utiliser nos connaissances dans un cas réel. L'expérience de travail en binôme en autonomie dans un temps contraint a été formatrice. Nous remercions Madame Maniu pour les quelques libertés qu'elle nous a laissé, qui nous ont permis de développer notre curiosité et stimuler notre créativité.

References

[1] Valentin Baillard, Alexandre Goy, Nicolas Vasselin, Cristina Stoica Mani : Potential field based optimization of a prey-predator multi-agent system, (Centralesupélec, 2018)

[2] Laure Bourgois, Haifa Rabai, Jean-Michel Auberlet : Adaptation dynamique du comportement: vers un modèle guidé par la perception, Page 1 et 4, (Lyon, 2012)

[3] Benjamin Lavis, Yasuyoshi Yokokohji, Tomonari Furukawa : Estimation and Control for Cooperative Autonomous Searching in Crowded Urban Emergencies.

[4] Les trois comportements fondamentaux du Professeur Laborit : https://www.vexilla-galliae.fr/royaute/idees/118 les-trois-comportements-fondamentaux-du-pr-laborit

[5] Eric Guillaume : Modélisation de l'évacuation en cas d'incendie, (Saint-Denis, 2013)