École Polytechnique de l'Université de Tours
64, Avenue Jean Portalis
37200 TOURS, FRANCE
Tél. (33)2-47-36-14-14
Fax (33)2-47-36-14-22
polytech.univ-tours.fr

Rapport
Projet de Programmation et Génie Logiciel

Simulation de mouvement de foule à échelle microscopique basé sur le modèle des force sociales de D. Helbing

Auteurs

Maxime EDELINE
[maxime.EDELINE@etu.univ-tours.fr]
Hicham MOUSTAQIM
[hicham.moustaqim@etu.univ-tours.fr]
Mathis MOYSE
[mathis.moyse-2@etu.univ-tours.fr]

Encadrant

Emmanuel NERON
[emmanuel.neron@univtours.fr]

Polytech Tours
Département Informatique

Version du 19 Janvier 2022

Avertissements

Ce document a été rédigé par Maxime EDELINE, Hicham MOUSTAQIM et Mathis MOYSE susnommés les auteurs. L'École Polytechnique de l'Université de Tours est représentée par Emmanuel NERON encadrant susnommé le tuteur académique. Par l'utilisation de ce modèle de document, l'ensemble des intervenants du projet acceptent les conditions définies ci-après.

Les auteurs reconnaissent assumer l'entière responsabilité du contenu du document ainsi que toutes suites judiciaires qui pourraient en découler du fait du non-respect des lois ou des droits d'auteur.

Les auteurs attestent que les propos du document sont sincères et assument l'entière responsabilité de la véracité des propos.

Les auteurs attestent ne pas s'approprier le travail d'autrui et que le document ne contient aucun plagiat.

Les auteurs attestent que le document ne contient aucun propos diffamatoire ou condamnable devant la loi.

Les auteurs reconnaissent qu'ils ne peuvent diffuser ce document en partie ou en intégralité sous quelque forme que ce soit sans l'accord préalable du tuteur académique et de l'entreprise.

Les auteurs autorisent l'école polytechnique de l'université de Tours à diffuser tout ou partie de ce document, sous quelque forme que ce soit, y compris après transformation en citant la source. Cette diffusion devra se faire gracieusement et être accompagnée du présent avertissement.

Table des matières

Introduction

Ce Rapport rend compte de notre travail pour notre de projet de programmation et génie logiciel réalisé lors du semestre 7 de notre quatrième année du cycle ingénieur en informatique à Polytech Tours.

Le sujet de ce projet est la Simulation de mouvement de foule à échelle microscopique basée sur le modèle des forces sociales de D. Helbing.

Le but est de créer une application conviviale qui permet à l'utilisateur de lancer et d'analyser une simulation d'évacuation de population à l'échelle microscopique (bureau, couloir…) et pouvoir naviguer dans le temps. On souhaite représenter les personnes par des points. On veut également que l'accélération de chaque personne dépende de forces exercées sur elles, forces appelées forces sociales. Chaque personne aurait sa propre morphologie, sa vitesse de marche, ses affinités et d'autres attributs. Ce n'est donc pas une simulation discrète. La seule variable discrète exploitée serait le temps avec un pas d'une ou deux secondes par itération. On pourra choisir l'environnement dans lequel lancer notre simulation, les calculs se feront avant que l'animation graphique se lance pour optimiser la navigation dans le temps. La pression que supporte chaque personne sera représentée avec une couleur plus foncée si la pression augmente.

A la fin de la simulation on accédera à un bilan de l'évacuation avec le temps total de la simulation. Il serait également bien important de représenter une carte de chaleur avec des couleurs vives aux endroits les plus fréquentés.

On a réalisé ce projet sur python, on a procédé aux tests unitaires nécessaires et on comparera nos résultats avec des résultats existants trouvables sur internet.

I-Modèle des Force Sociale du D. Helbing

Cette partie va vous expliquer le principe du modèle des forces sociales ainsi que son fonctionnement. Cependant, ces explications ne seront qu'un condensé d'informations et toutes les formules utilisées ne seront pas présentes. Par conséquent, pour une compréhension plus fine du modèle nous vous invitons à aller consulter le document PDF « HelbingSocialForceModel » joint à ce rapport ou accessible depuis ce lien.

I.1-Principe générale

Le principe du modèle des forces sociales de D.Helbing est plutôt simple à comprendre. Helbing dit que lors d'un rassemblement, des forces sont exercées sur toutes les personnes en fonction de plusieurs facteurs. A un instant t, les personnes exercent des forces de répulsions entre elles lorsqu'elles sont assez proches mais des obstacles peuvent également exercer des forces de répulsions sur les personnes. Lors d'une évacuation les personnes sont attirées vers la sortie, attirance que l'on peut modéliser avec une force d'accélération vers la sortie. Si plusieurs personnes appartiennent à un même groupe, une force d'attraction peut possiblement être exercée, on peut prendre l'exemple d'une mère et son fils. La somme de ces forces nous permet de déterminer la nouvelle position du piéton à l'instant t+1. Voilà ce que signifie l'équation ci-dessous :

$$\overrightarrow{F_\alpha}(t) := \overrightarrow{\mathrm{F}_\alpha^0}(\overrightarrow{v_\alpha}, v_\alpha^0\overrightarrow{e_\alpha}) + \sum_\beta \overrightarrow{F_{\alpha\beta}}(\overrightarrow{e_\alpha}, \overrightarrow{r_\alpha} - \overrightarrow{r_\beta}) + \sum_B \overrightarrow{F_{\alpha B}}\left(\overrightarrow{e_\alpha}, \overrightarrow{r_\alpha} - \overrightarrow{r_B^\alpha}\right) + \sum_i \overrightarrow{F_{\alpha i}}(\overrightarrow{e_\alpha}, \overrightarrow{r_\alpha} - \vec{r_i}, t)$$

I.2-Force D'accélération

La force d'accélération permet de déterminer la force d'attraction vers la sortie. Pour calculer cette force, nous avons besoin du vecteur vitesse du piéton α, de sa vitesse actuelle v_α^0 ainsi que de son vecteur direction $\overrightarrow{e_\alpha}$. Cette force est calculée à partir de la formule que vous pouvez observer juste en dessous. Cette force est calculée en supposant que le piéton n'est pas perturbé par d'autres événements comme par exemple les piétons hors du champ de vision qui peuvent avoir une faible influence.

$$\overrightarrow{\mathrm{F}_\alpha^0}(\overrightarrow{v_\alpha}, v_\alpha^0\overrightarrow{e_\alpha}) := \frac{1}{\tau_\alpha}(v_\alpha^0\overrightarrow{e_\alpha}, \overrightarrow{v_\alpha})$$

I.3-Force de Répulsion

1.3.1 Force de Répulsion entre personnes

Lors d'un événement les piétons gardent une distance par rapport aux autres piétons. Cette influence exercée par les autres piétons sur un piéton peut être représentée par une force de répulsion modélisée par cette formule mathématique :

$$\overrightarrow{F_{\alpha\beta}}(\overrightarrow{e_\alpha}, \overrightarrow{r_\alpha} - \overrightarrow{r_\beta}) = w(\overrightarrow{e_\alpha}, -\overrightarrow{f_{\alpha\beta}})\vec{f}_{\alpha\beta}(\vec{r}_\alpha - \overrightarrow{r_\beta})$$

$\overrightarrow{r_\alpha}$ = Position du piéton alpha
$\overrightarrow{r_\beta}$ = Position du piéton beta
$\overrightarrow{f_{\alpha\beta}}$ = Effet de répulsion entre le piéton alpha et le piéton Beta

$la\ fonction\ w$: Permet de déterminer l'influence des autres piétons hors du champ de vision du piéton en question.

1.3.2 Force de Répulsion entre une personne et un obstacle

Le même type de comportement peut être observé avec des obstacles. D.Helbing a modélisé cette influence par un autre type de forces de répulsions qui est modélisé par la formule mathématique suivante :

$$F_{\alpha B}(\overrightarrow{r_{\alpha B}}) := -\nabla_{\overrightarrow{r_{\alpha B}}} U_{\alpha B}(||\overrightarrow{r_{\alpha B}}||)$$

$\overrightarrow{r_{\alpha \mathrm{B}}} = \overrightarrow{r_\alpha} - \overrightarrow{r_B}$, respectivement la position du piéton et la position du point de l'obstacle qui applique la force.

$-\nabla_{\overrightarrow{r_{\alpha B}}}$ = Opérateur mathématique Nabla appliquer sur le vecteur $\overrightarrow{r_{\alpha \mathrm{B}}}$

$U_{\alpha B}$ = Effet répulsif avec un potentiel décroissant constant

I.4-Force d'attraction

Enfin lors d'événement comme une évacuation, il peut exister des personnes qui veulent rester ensemble et ainsi former un groupe ou alors des personnes se dirigeant dans la même direction ce qui peut créer aussi des groupes. Ainsi les personnes peuvent aussi avoir un effet d'attraction entre elles, c'est ce que le D. Helbing a modélisé avec cette formule mathématique

$$\overrightarrow{F_{\alpha \iota}}(\overrightarrow{e_\alpha}, \overrightarrow{r_\alpha} - \vec{r_\iota}, t) = w\left(\overrightarrow{e_\alpha}, -\overrightarrow{f_{\alpha \iota}}\right)\vec{f}_{\alpha i}(\vec{r}_\alpha - \overrightarrow{r_\iota}, t)$$

$\overrightarrow{r_\alpha}$ = Position du piéton alpha

$\overrightarrow{r_\beta}$ = Position du piéton i

$\overrightarrow{f_{\alpha\beta}}$ = effet d'attraction entre le piéton alpha et le piéton i

t = Instant t à laquelle

II-Modélisation de la solution

Lors de la modélisation de la solution, nous avons réalisé parallèlement 3 diagrammes UML (diagramme de classe, diagramme de séquence et diagramme d'activité) ainsi que 2 maquettes pour les IHM que nous devons réaliser au cours de ce projet.

II.1-Les Diagramme UML

II.1.1-Diagramme de classe

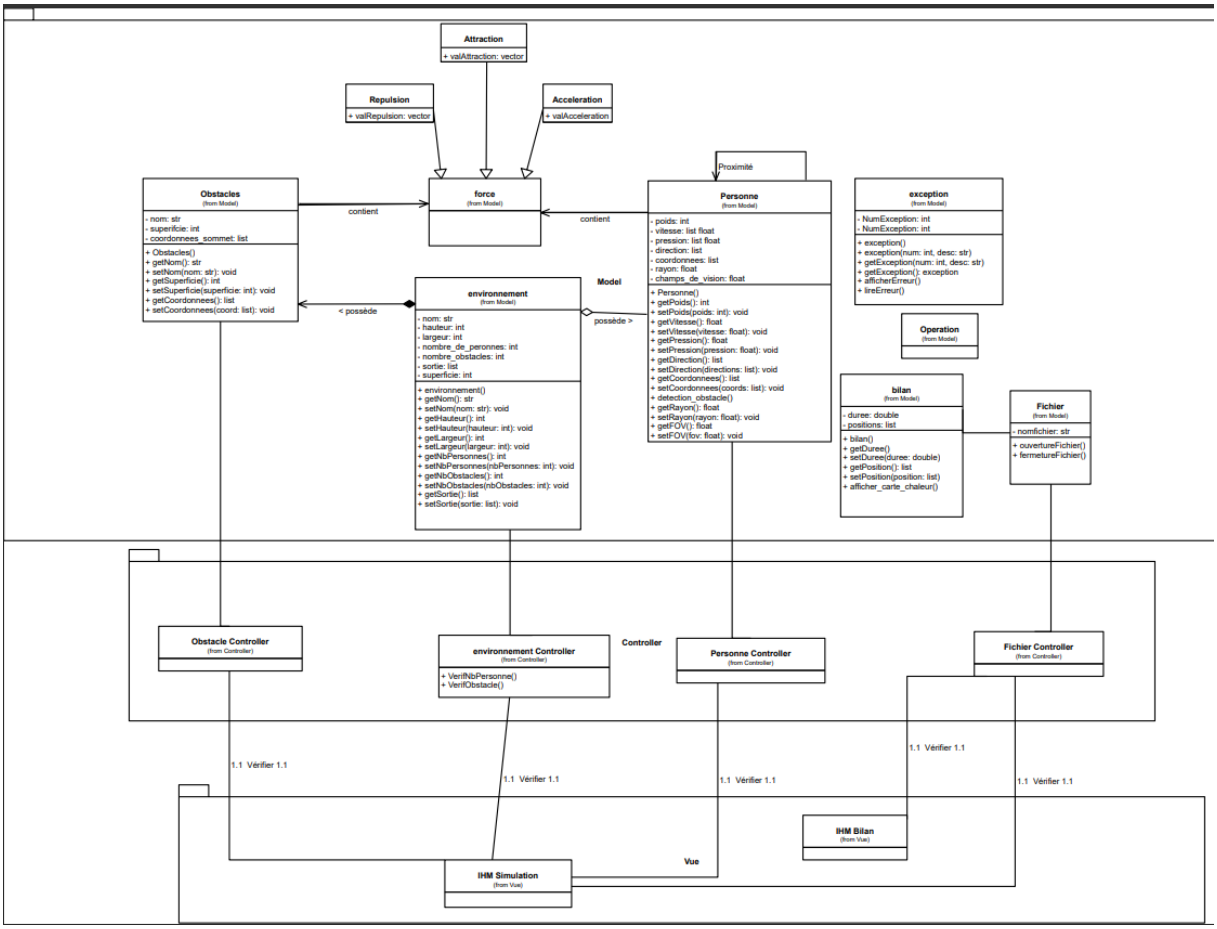


Figure 1 : Diagramme des classes lors de la modélisation

Comme nous pouvons le voir sur l'image du diagramme de classes ci-dessus, nous avons décidé d'implémenter une architecture MVC (Modèle-Vue-Contrôleur) pour notre solution. Nous allons vous expliquer l'utilité de chacune de ces classes :

Classe du Modèle :

- Force : Cette classe est un classe mère qui contient les constantes utiles aux 3 forces du modèle des forces sociales ainsi que les fonctions communes aux 3 forces comme le calcul du vecteur sorti.

- Repulsion : Fille de la classe Force, cette classe contiendra les attributs et fonctions spécifiques aux calculs de la force de la Force de répulsion entre piéton/piéton et piéton/obstacle
- Acceleration : Fille de la classe Force, cette classe contiendra les attributs et fonctions spécifiques aux calculs de la force de la Force d'accélération

- Obstacle : Cette classe nous permettra de représenter les différents obstacles de l'environnement utilisé dans la simulation. Nous caractérisons un obstacle par son nom et un attribut « coordonees_sommet » qui contiendra les coordonnées des sommets de l'obstacle, Ainsi qu'une superficie qui est calculé à partir des sommets. On a aussi créé une relation entre Obstacle et Force car un obstacle pourra appliquer une force de répulsion sur un piéton.

- Environnement : Cette classe nous permettra de modéliser un environnement. Celle-ci contient différents attributs qui permettent de le caractériser, notamment les attributs « nom », « hauteur », « largeur », « sorties » qui est une liste car un environnement peut contenir plusieurs sortie et Superficie qui est calculé à partir des attributs hauteur et largeur. Cette classe contiendra notamment un attributs « nombre_de_personnes », et « nombre_obstacles » qui correspondent à la relation de composition entre Environnement/Personnes et environnement/Obstacle

- Personnes : Cette classe est assez importante car le principe des forces sociale s'applique sur une personne. Elle fait donc partie des classes centrales de notre modélisation et nous permettra donc de représenter une personne dans notre solution. Pour cela, nous avons décidé de lui attribuer différents attributs, qui correspondent aux besoins de la simulation. Nous avons donc mis un attribut poids et vitesse qui permettra de calculer le vecteur vitesse. Un attribut « pression » pour modéliser la pression subit par une personne comme demandé dans le cahier des charges. Un attribut « direction » qui est une List pour savoir quelle chemin la Personne empruntera. L'attribut « coordonnées » enregistre les coordonnées de la personne ce qui nous permettra de calculer les forces ainsi que de la positionner dans l'environnement. L'attribut « rayon » permet de connaitre la « taille » de la personne dans l'IHM, personne qui sera représentée sous forme de cercle. Et enfin le champ de vision utilisé dans le calcul des forces.

- Exception : Cette classe nous permettra de gérer les exceptions dans notre code. L'attribut NumException correspond aux numéros de l'exception et ainsi qu'un second attribut qui permet de donner plus d'information sur l'exceptions levée.

- Operation : Cette classe contiendra toutes les opérations non élémentaires à une classe (comme des opérations par exemple)

- Bilan : Cette classe contiendra les informations liées aux bilans comme notamment le temps de la simulation (l'attribut « duree ») ou encore une liste de

positions (l'attribut « positions ») qui permettra de réaliser la carte de chaleur qui s'affichera dans l'IHMBilan

- Fichier : Cette classe nous permettra de gérer la lecture, l'écriture et le parsing des fichiers qu'on utilisera lors de la simulation.

Classes du Contrôleur :

Comme vous pouvez le voir sur l'image du diagramme de classes, nous avons 4 classes contrôleurs qui sont :

- ObstacleController
- EnvironnementController
- PersonneController
- FichierController

Nous n'allons pas les décrire une par une comme pour les classes du modèle car tous contrôleurs ont la même fonctionnalité, c'est-à-dire vérifier la cohérence des données par rapport aux modèles auxquels ils sont associés.

Classes de la Vue :

En ce qui concerne la vue, on a prévu de réaliser 2 classes :

- IHMSimulation
- IHMBilan

Ces 2 classes auront pour tâche d'afficher la simulation et le bilan le tout dans une interface graphique correspondant aux maquettes présentées dans la II.2.

II.1.2-Diagramme d'activités

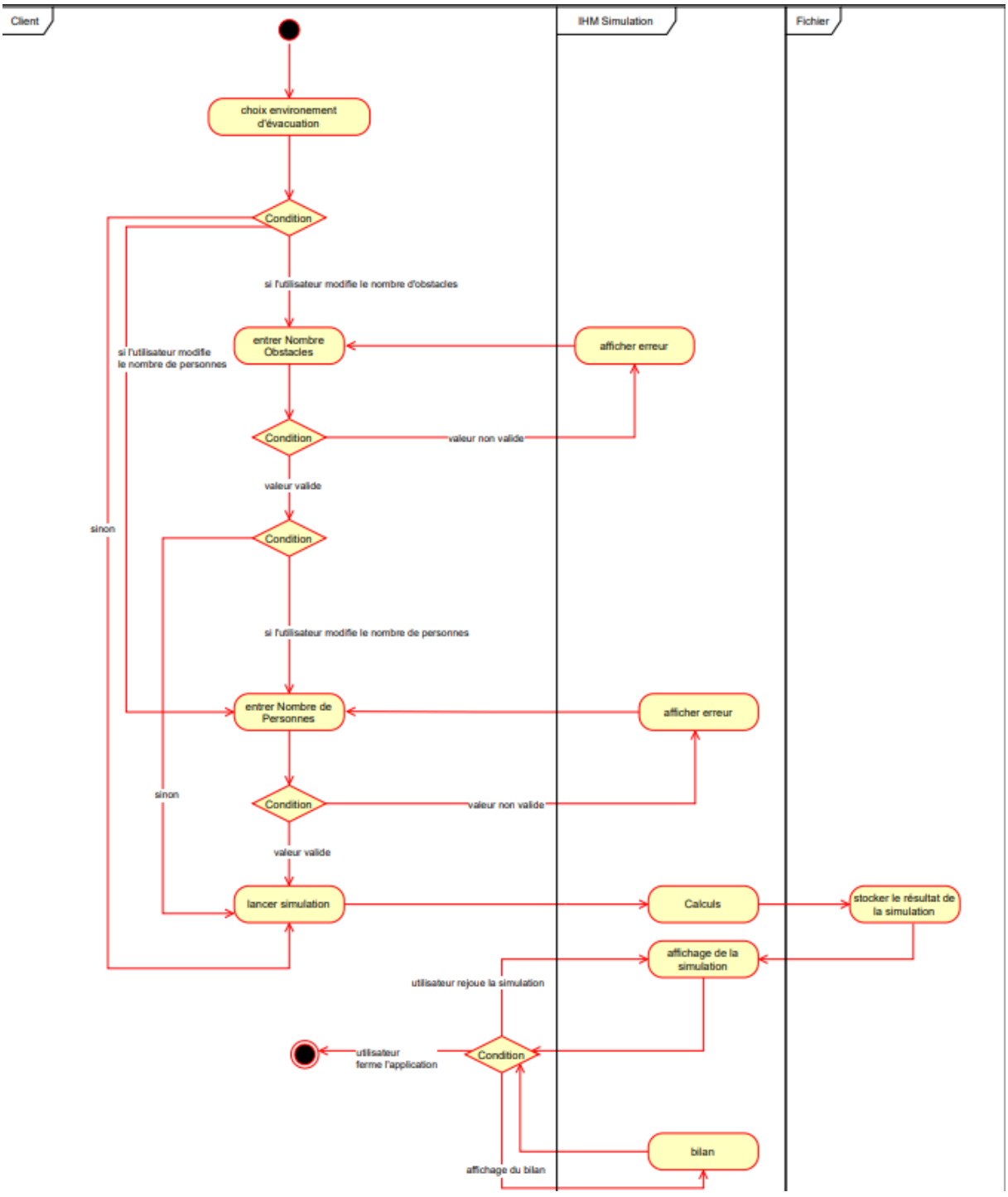


Figure 2 : Diagramme d'activités

Dans ce diagramme d'activités, nous pouvons voir qu'après avoir choisi le fichier on peut choisir le nombre d'obstacle et le nombre de personnes dans la simulation. Cependant, cette possibilité visible sur notre diagramme ne fut pas implémentée lors du développement de la solution. En effet, notre encadrant M. NERON nous a déconseillé de faire ceci pour plusieurs raisons dont la raison principale est le placement aléatoire des personnes et des objets, qui pourrait être compliqué à gérer et nous causer des soucis. A part cela, le diagramme d'activité correspond au fonctionnement général de notre application, c'est-à-dire qu'une fois le fichier choisis, l'utilisateur peut lancer la simulation. Et lorsque la simulation est lancée celle-

ci réalise d'abord les calculs pour les enregistrer dans un fichier csv que l'IHM lira pour afficher la simulation. Une fois la simulation terminée, l'utilisateur a alors le choix entre soit fermer l'application, relancer la simulation ou afficher le bilan.

II.1.3-Diagramme de séquence

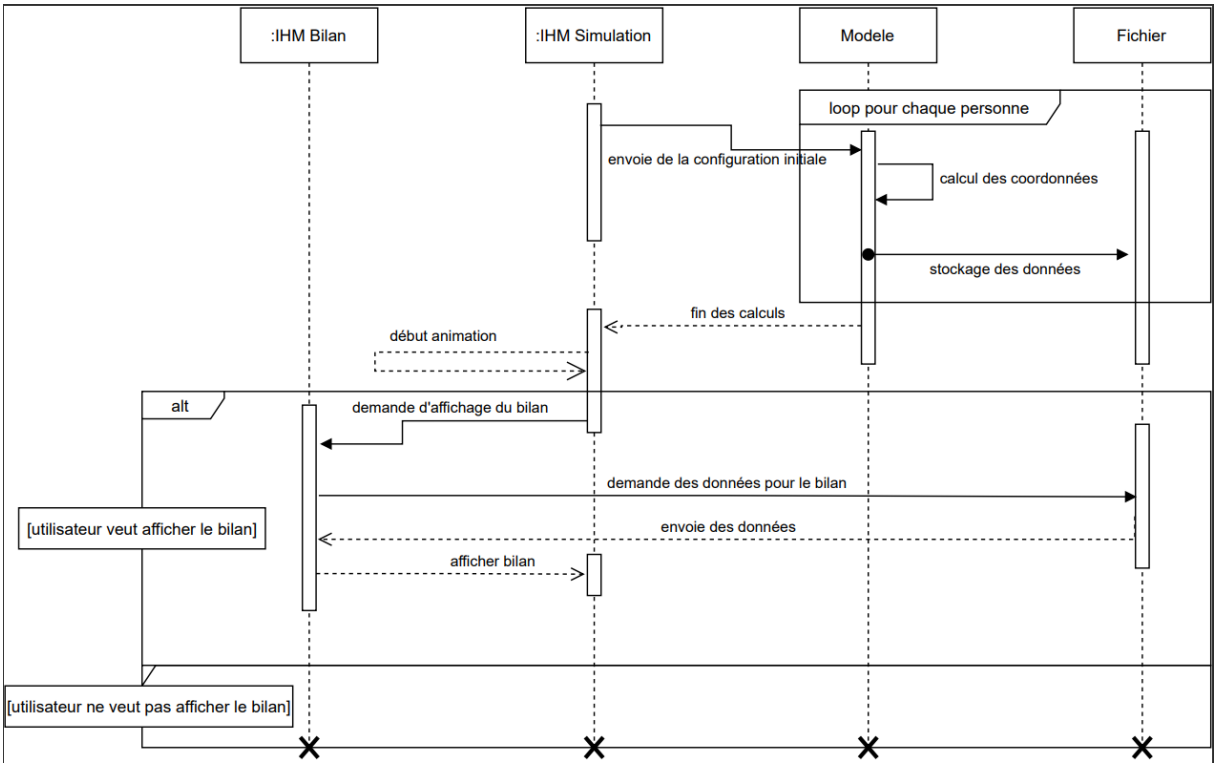


Figure 3 : Diagramme de séquence

En termes de scénario, le diagramme de séquence raconte la même chose que le diagramme d'activité décrit précédemment mais permet de visualiser temporellement ces différentes actions. En effet, certaines actions prennent plus de temps que d'autres à se réaliser, comme la phase de calculs des coordonnées des piétons par exemple, qui prendra plusieurs secondes. Ces phases sont représentées sur le diagramme grâce à des flèches en éclair.

Un des objectifs du projet était de réaliser une application conviviale, ainsi nous avons conçu une IHM simple et intuitive pour faciliter l'utilisation du simulateur. Nous avons réalisé ces maquettes en utilisant le logiciel QT, puis nous les avons faites valider par M. NERON. Pour ce projet, deux IHM ont été conçues, l'une pour la simulation et une autre pour afficher le bilan.

II.2.1 Maquette Simulation

En ce qui concerne l'IHM de la simulation, voici ce que nous avons conçu :

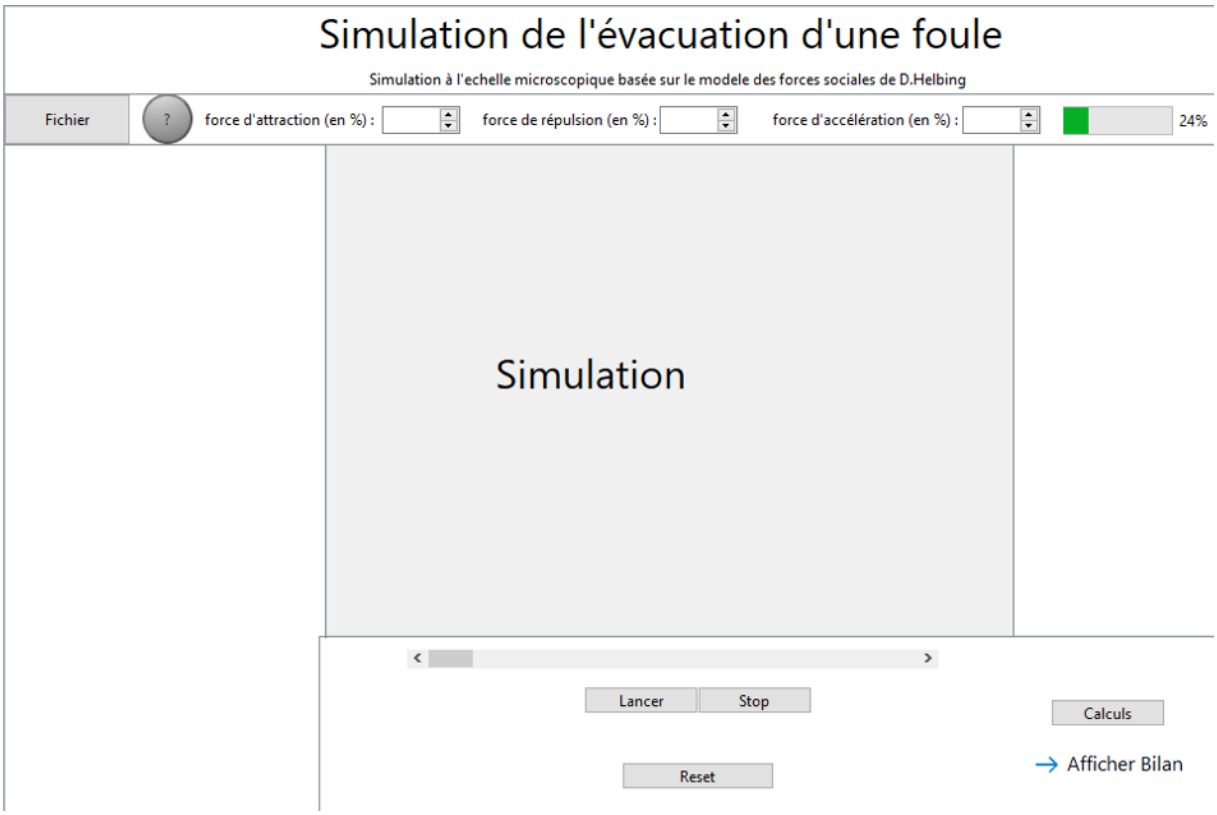


Figure 4 : Version Finale de la Maquette de l'IHM Simulation

Un bandeau supérieur est présent pour y contenir les différents éléments utiles pour le déroulement de la simulation. Un menu déroulant qui contient les différents fichiers permettant de sélectionner l'environnement souhaité. Ensuite, nous avions prévu un bouton « ? » qui afficherait un mode d'emploi de l'application. Et enfin différents champs de saisis pour appliquer des poids plus moins importants sur les forces.

II.2.2 Maquette Bilan

L'IHM bilan a pour rôle d'afficher la carte des chaleurs ainsi que le temps d'exécution de la simulation. La carte des chaleurs représente les zones de l'environnement où les personnes ont le plus circulé. Voici la maquette que nous avons conçu :

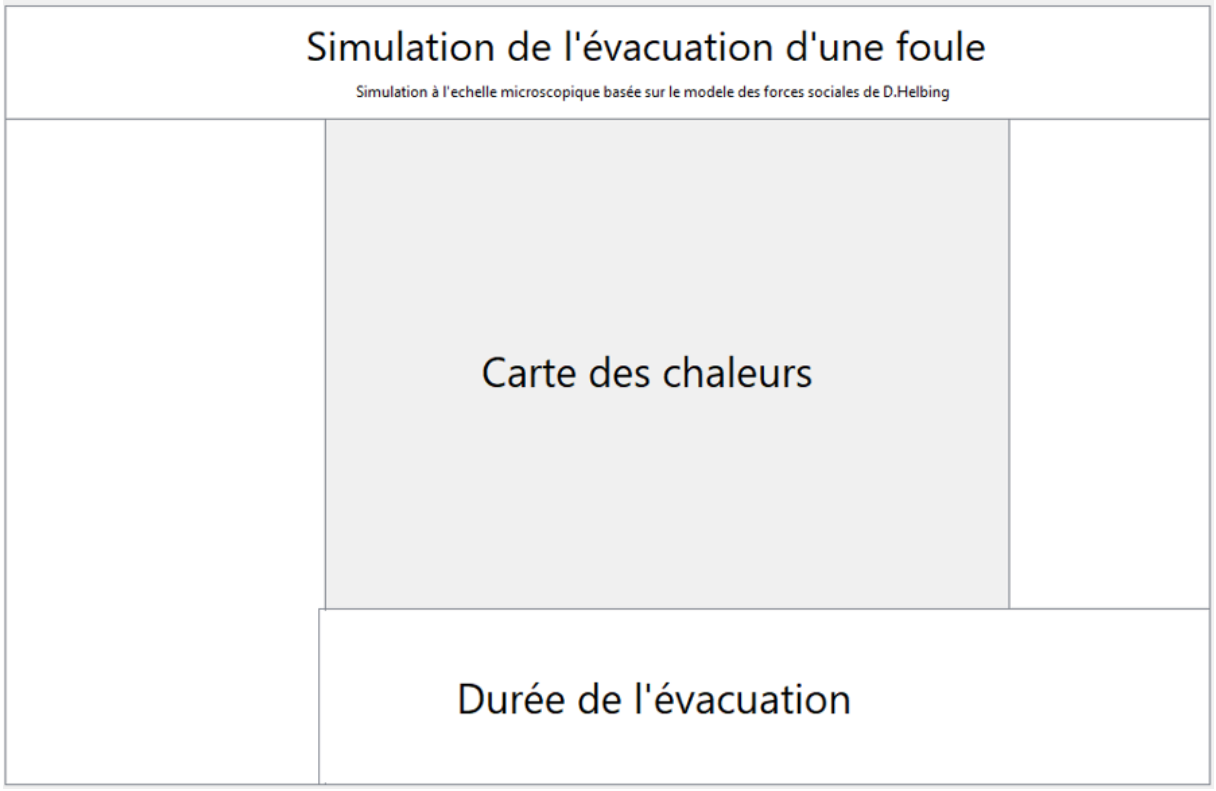


Figure 5 : Version Finale de la Maquette de l'IHM Bilan

III-Développement de la solution

La phase de développement a commencé relativement tard à savoir début décembre pour la simple et unique raison que nous avons voulu mener à bien la phase de modélisation et d'échanges avec notre encadrant M. Néron dans le but de cerner correctement les enjeux et les besoins auxquels nous devions répondre. Cependant, nous avons tout de même débuté une sorte de veille technologique en parallèle de nos échanges afin de reprendre en main les mécanismes et différentes librairies et ne pas perdre de temps.

III.1- Technologies utilisées

Nous avons décidé de coder en python puisque nous avions déjà pris en main son interface graphique TKinter, ce qui n'était pas forcément le cas avec le C++ par exemple. Notre enseignant préférait également ce langage donc cela nous a conforté dans notre choix. Nous savions que nous aurions affaire à des calculs complexes également comme des calculs de normes de vecteurs ou bien le fait d'utiliser des symboles mathématiques complexes comme le nabla ou bien la racine carrée. La bibliothèque numpy nous a beaucoup aidé sur ce coup-là. La bibliothèque matplotlib, servant à afficher toutes sortes de diagrammes, nous a permis de générer notre carte de chaleur. Finalement, il était important de pouvoir lire plusieurs informations issues de fichiers csv, la bibliothèque du même nom nous était donc indispensable.

III.2-Architecture de la solution

Bien que l'architecture de la solution se base sur le diagramme des classes montré et expliqué dans la précédente partie, celle-ci est bien différente que ce soit au niveau des relations entre des classes ou des attributs de classes. Vous pouvez voir le diagramme des classes correspondant à notre développement en annexe avec ce compte Rendu sous format PDF. Cf : Diagramme_Des_ClassesVFinale.pdf.

III.2.1-Modèle

Au niveau des classes beaucoup de nouvelles classes dans le modèle ont été modifiées et certaines ont même été supprimées.

III.2.1.1 Classes non développées

Au niveau des classes supprimées, nous n'avons pas développé :

- Bilan : Cette classe a tout simplement été fusionnées avec l'IHMBilan lors du développement. Avoir fusionné les 2 lors du développement est une mauvaise chose en termes de pratique de génie logicielle puisqu'il vaudrait mieux séparer la partie Modèle et Vue, surtout quand on applique une structure MVC.

- CException : Nous n'avons pas développé de classe CException car nous étions habitués aux systèmes de gestion d'exception en C++ ou l'on gérait les exceptions à l'aide de cette classe. Or en python le système de gestion des exceptions est différent, nous n'avons donc pas développé cette classe puisqu'elle n'est pas nécessaire.

III.2.1.2 Classes ajoutées

Pour faciliter la réalisation de certaines tâches ou alors par une réflexion plus profonde de la modélisation, nous avons ajouté certaines classes lors du développement. Voici la liste de ces classes :

- CObstacleQuadrilatere : Nous avons ajouté cette classe car nous nous sommes rendu compte lors du développement de la solution que les variétés d'obstacle réalisable pouvaient nécessiter des attributs ou des opérations propres à eux mais aussi des attributs et méthodes communes. C'est pour cela que nous avons décidé de transformer la classe « CObstacle » en une classe mère et de créer une classe fille «CObstacleQuadrilatere » qui nous permet donc de modéliser des quadrilatères comme son nom l'indique. Actuellement notre solution n'est capable que de modéliser des carrés ou des rectangles.

III.2.1.1 Optimisation du développement

L'optimisation de notre code était censée avoir une part importante lors du développement de la solution. Celle-ci devait prendre place lorsque la première version de la solution fut effective et fonctionnelle avec tous les objectifs du cahier des charges réalisés. Malheureusement, nous n'avons jamais réussi à réaliser cette première version et n'avons pas réalisé cette phase d'optimisation. Cependant conscient que l'on devait optimiser notre code, notamment pour les phases de calculs, nous avons pris quelques décisions pour les optimiser. Voici ces décisions :

- La première optimisation n'est pas une optimisation de performance mais plutôt une optimisation d'espace. En effet, lors du développement de la Classe « CPersonne », nous avons mis un attribut qui contient la liste des cordonnées du piéton. Mais dans cette liste nous n'avons besoin à chaque nouvelle itération que de la position à l'instant t et à l'instant t-DeltaT (itération précédente). C'est pour cela que nous avons décidé de faire une liste de taille 2 contenant ces 2 coordonnées. Ainsi, Si on prend par exemple une simulation qui à un totale de 300 itérations et de 20 personnes, nous ne stockons que 600 coordonnées à la place de 6000.
- La deuxième optimisation est une optimisation de performance. Dans la fonction qui s'occupe de la partie calculatoire de la simulation, nous avons fait en sorte de minimiser le plus possible le nombre de boucle pour optimiser les performances de notre solution. Cependant, nous pensons que notre partie calculatoire que ce soit les calculs des nouvelles coordonnées, les fonctions mathématiques implantées ainsi que l'implémentation du modéle des forces sociales est optimisable.

III.2.2-Contrôleur

En ce qui concerne le contrôleur, nous avons, sans le vouloir, fusionné cette partie dans le modèle. En effet, les calculs de forces se font dans le modèle mais auraient dû se trouver dans le contrôleur, de même pour la gestion des erreurs. Cela est principalement dû à notre manque d'expérience avec cette structuration de code. Néanmoins, nous avons uniquement développé la classe CEnvironnementController. Cette classe contient des fonctions permettant de contrôler que les éléments à afficher ont bien des coordonnées compatibles avec le canvas. En ce qui concerne les personnes, on redéfinit leurs coordonnées si elles sont en dehors du canvas afin qu'elles soient bien affichables. Pour les obstacles, si le coin en bas à gauche est en dehors du canvas on ne l'affiche pas.

III.2.3-Vue

La vue contient bien les classes que l'on peut apercevoir sur le diagramme de classe mais nous nous sommes aperçus qu'il fallait également des classes pour la majorité des objets qui se devaient d'être visibles lors de la simulation comme les obstacles ou les personnes. Par conséquent nous avons créé 3 classes supplémentaire « CObstacleQuadrilatereVue », « CSortiesVue » et « CPersonneVue » qui permettent de modéliser l'affichage d'une personne et d'un quadrilatère dans l'IHMSimulation.

En ce qui concerne les IHM nous avons réalisé une classe mère CIHM dont hérite les classes CIHMSimulation (l'IHM de la fenêtre de la simulation) et CIHMBilan (l'IHM de la fenêtre bilan).

Lors de la réalisation des IHM, nous avons fait quelques changements par rapport aux maquettes. La barre de progression a été remplacé par un label « chargement… » qui apparait lors des calculs et disparait quand ils se termine. De plus, pour la navigation nous avons choisis des boutons pour aller en arrière et pour aller en avant. Nous avons décidé d'abandonner l'idée de la scrollbar puisque nous avions des problèmes d'affichage avec. En effet il n'était pas possible de voir la barre de défilement, ce qui rendait la navigation impossible. Nous avons également ajouté un menu pour choisir la vitesse de lecture de la simulation.

Le bouton « bilan » permet d'ouvrir une nouvelle fenêtre pour afficher la carte des chaleurs. Cette carte est réalisée à partir d'une matrice qui est construite lors de l'ouverture de la fenêtre.

Pour le menu de sélection de l'environnement, on récupère le nom des fichiers CSV présent dans le dossier « environnements » que l'on stock dans une liste. Le menu « Fichier » affiche ces éléments puis lorsque l'on sélectionne un des éléments on construit un objet CFichier puis à partir de ce dernier on peut construire notre objet CEnvironnement grâce au parser. Cette liste est actualisée à chaque lancement de l'application.

III.3- Test Unitaire

Nous avons réalisé tout au long du projet des tests unitaires notamment pour tester si les calculs des forces étaient corrects. Cependant, pour les tests unitaires sur le modèle des forces sociales, nous avons réalisé des tests unitaires seulement sur la force d'accélération et la classe « CForce » car nous nous sommes rendu compte que lors du développement des tests unitaires, les fonctions permettant de calculer la Force de Répulsion ou les effets de répulsion et d'attraction était dépendante d'autre fonction pour les tests unitaire. Par exemple le calcul de Nabla ou du vecteur direction était calculé dans la fonction et non donnée en paramètre comme sur le document PDF d'explication du modèle des forces sociales. Mais ceci n'est pas le seul problème, car en ne donnant pas la valeur numérique des paramètres, comme sur le document présentant les principes des forces sociales, mais les coordonnées des piétons ainsi que d'autres paramètres, on doit réaliser tous les calculs à la main à partir de coordonnées pour savoir si notre méthode fonctionne. On ne peut donc pas prendre des valeurs aléatoires qu'on applique directement dans la formule. Les formules étant complexes à appliquer, nous avons préféré ignorer le développement de ces tests unitaires pour assigner ce temps gagné à des tâches plus importantes.

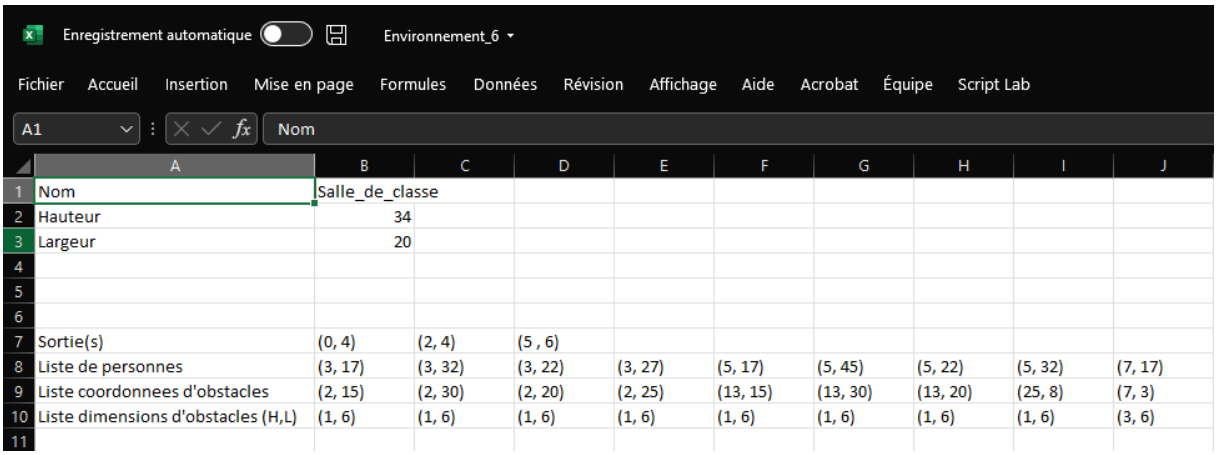
Nous avons également réalisé des tests sur les différentes méthodes de CPersonne et sur COperation afin de vérifier si nous obtenons bien les résultats voulus.

III.4-Simulation

Nous allons ici expliquer le mécanisme complet de la simulation, des calculs à l'affichage de cette dernière.

III.4.1 - Parsing

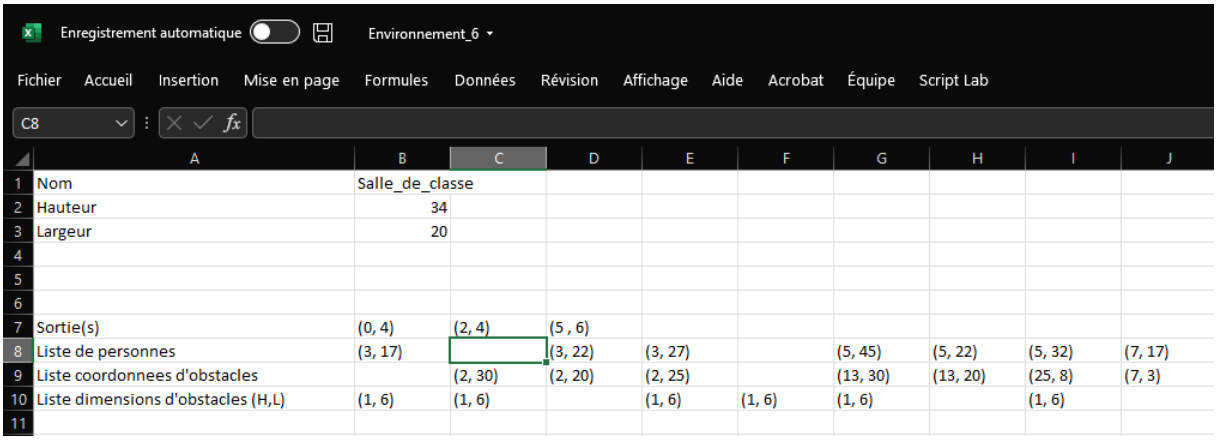
Pour réaliser le parsing nous avons fait une méthode dédiée dans la classe CFichier. Cette méthode est ensuite utilisée dans la méthode de lecture de fichier CSV. Puis les données récupérées sont utilisées pour créer l'objet CEnvironnement. Nous avons décidé d'un format à respecter dans le CSV pour permettre la lecture du fichier de manière efficace, voici le format à respecter le plus fidèlement possible :



	A	B	C	D	E	F	G	H	I	J
1	Nom	Salle_de_classe								
2	Hauteur	34								
3	Largeur	20								
4										
5										
6										
7	Sortie(s)	(0, 4)	(2, 4)	(5 , 6)						
8	Liste de personnes	(3, 17)	(3, 32)	(3, 22)	(3, 27)	(5, 17)	(5, 45)	(5, 22)	(5, 32)	(7, 17)
9	Liste coordonnees d'obstacles	(2, 15)	(2, 30)	(2, 20)	(2, 25)	(13, 15)	(13, 30)	(13, 20)	(25, 8)	(7, 3)
10	Liste dimensions d'obstacles (H,L)	(1, 6)	(1, 6)	(1, 6)	(1, 6)	(1, 6)	(1, 6)	(1, 6)	(1, 6)	(3, 6)
11										

Figure 6 : Fichier CSV pour modéliser un environnement

Les valeurs numériques correspondent aux coordonnées (x, y) en mètre où nous souhaitons positionner les éléments. La conversion des valeurs en mètre en px pour les positionner dans le canvas est faite directement dans la méthode de lecture du fichier CSV. En ce qui concerne les obstacles, les coordonnées correspondent au point en bas à gauche, puis à partir de la hauteur et de la largeur renseignés en dessous nous calculons les autres points. Si aucune dimension n'est précisée la valeur de 1 sera attribué comme largeur et hauteur. De plus, même si certaines infos ne sont pas précisées, le parser s'effectue sans problèmes car des valeurs par défaut sont prévu. Par exemple si on ne mentionne pas la ligne « Sortie(s) », une sortie par défaut sera initialisée. Notre parser est assez robuste pour gérer les sauts de lignes en trop et les cases vides entre les différents éléments. Par exemple ce format est parfaitement lisible par notre parser :



	A	B	C	D	E	F	G	H	I	J
1	Nom	Salle_de_classe								
2	Hauteur	34								
3	Largeur	20								
4										
5										
6										
7	Sortie(s)	(0, 4)	(2, 4)	(5 , 6)						
8	Liste de personnes	(3, 17)		(3, 22)	(3, 27)		(5, 45)	(5, 22)	(5, 32)	(7, 17)
9	Liste coordonnees d'obstacles		(2, 30)	(2, 20)	(2, 25)		(13, 30)	(13, 20)	(25, 8)	(7, 3)
10	Liste dimensions d'obstacles (H,L)	(1, 6)	(1, 6)		(1, 6)	(1, 6)	(1, 6)		(1, 6)	
11										

Figure 7 : exemple fichier CVS robuste

III.4.2 – Calculs

La partie calculatoire voici l'algorithme appliqué pour réaliser les calculs :

On initialise une liste de la taille du nombre de personnes avec toutes les valeurs à True. On l'appellera « ListePersonneSortie ».

```
1.  Fini <- False
2.  listPersonne <- environnement.getListPersonne()
3.  listObstacle <- environnement.getListObstacle
4.  t<-0
5.  Tant que Fini == False and t < 40000 :
6.  Pour chaque personne P1 dans listPersonne:
7.    Calcul de la Force D'accélération
8.    Clear de la liste contenant les personnes à proximité de P1
9.    Pour chaque personne P2 !=P1 dans listPersonne :
10.             Si P2 est dans la zone d'influence de P1
11.                       Alors on ajoute P2 dans la liste des personnes à proximité de P1
12.             Fin Si
13.   Fin Pour
14. On calcule la force de Répulsion entre Personne pour P1
15.   On calcule la force de D'attraction pour P1
16.   Pour chaque Obstacle Ob1 dans listObstacle :
17.             Si Ob1 est dans la zone d'influence de P1
18.                       Alors on ajoute Ob1 dans la liste des obstacles à proximité de P1
19.             Fin Si
20.   Fin Pour
21.   On calcule la force de Répulsion Personne/Obstacle
22.   Calcul de la nouvelle position du piéton
23.   Si la personne est sortie
24.   Alors passer sa valeur dans ListePersonneSortie à False
25.   Si toutes les personnes sont sorties (soit toutes les valeurs de ListePersonneSortie  à
    False)
26.             Alors Fini <- True
27.   Fin Si
28. Fin Si
29.   Fin Pour
30. t <- t + DeltaT
31. Fin Tant Que
32.
```

III.4.3- Affichage

Les positions de chaque personne, ayant été au préalable stocké dans un fichier csv, sont récupérées dans une matrice qui va s'avérer très utile pour la suite. On a donc sur chaque ligne de la matrice, les positions de toutes les personnes à l'itération correspondante. Lors de l'affichage de la simulation, on va donc parcourir cette matrice ligne par ligne, afficher les personnes grâce à la classe « CPersonneVue » qui prendra les coordonnées de chaque personne. On met à jour la fenêtre de simulation à chaque itération pour avoir une impression de mouvement. Pour naviguer dans la simulation, nous avons choisi de créer une variable qui va stocker l'avancement dans la matrice, si l'on veut revenir en arrière dans la simulation, on a juste à repartir de l'itération stockée et retirer 1 autant de fois que l'on veut et que l'on peut. Cela fonctionne avec la même idée dans l'autre sens, pour avancer dans la simulation.

IV-Résultat

IV.1- Résultat Global

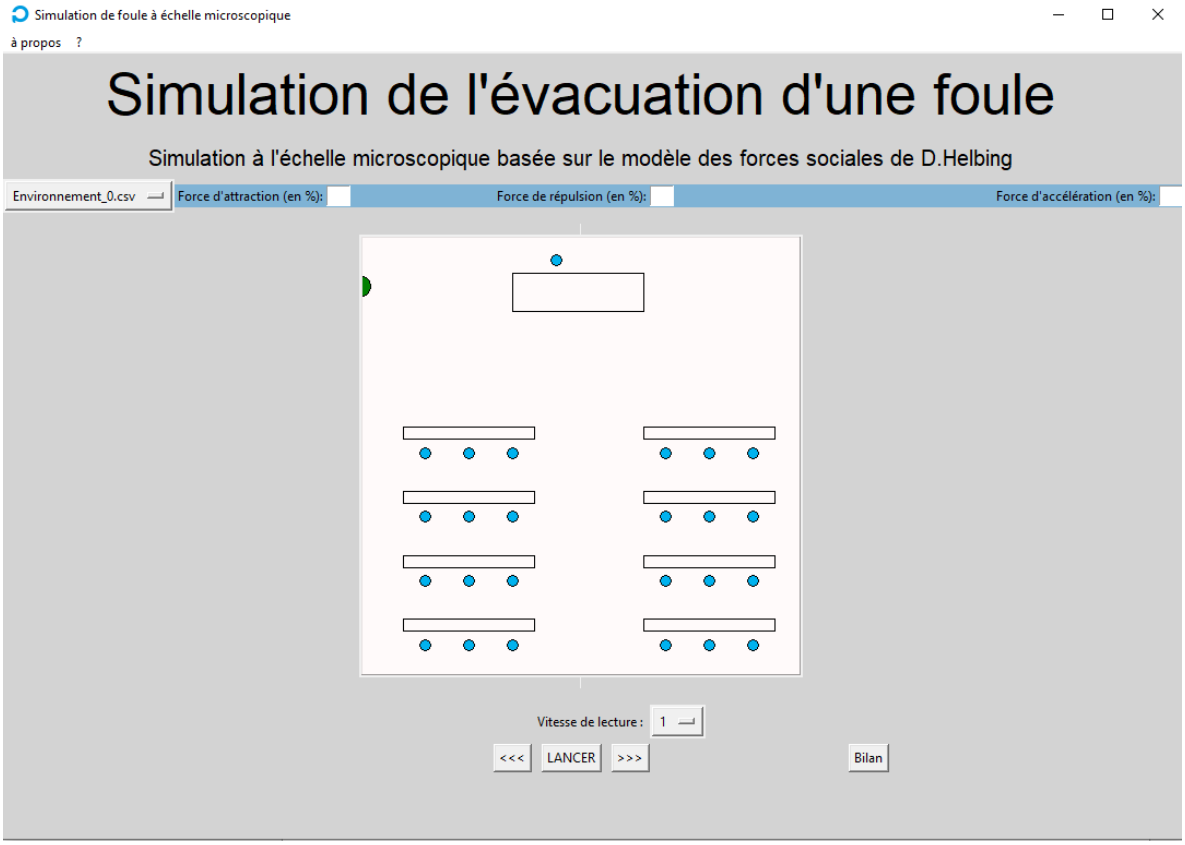


Figure 88 : Version finale de l'IHM avec un environnement chargé

IV.1.1 – Objectifs Réalisés

- Lancer une simulation fonctionne. Une fois la simulation lancée, l'utilisateur doit attendre la fin de celle-ci pour pouvoir utiliser les boutons « avancer » et « reculer ». En revanche, le menu de vitesse de lecture reste accessible si l'utilisateur veut augmenter ou diminuer la vitesse. Lors du choix de la vitesse, cela mettra en pause la simulation et la fera reprendre avec la vitesse sélectionnée.
- Nous pouvons donc naviguer dans la simulation avec deux boutons "<<<" et ">>>" sur lesquels nous pouvons rester appuyé pour revenir en arrière ou avancer dans la simulation.
- Appuyer sur le bouton « Bilan » ouvrira une fenêtre secondaire dans laquelle se trouvera la carte de chaleur ainsi que la durée de la simulation.
- Les forces de répulsion entre les personnes, d'attraction et d'accélération fonctionnent correctement.

IV.1.2 – Objectifs Non Réalisés

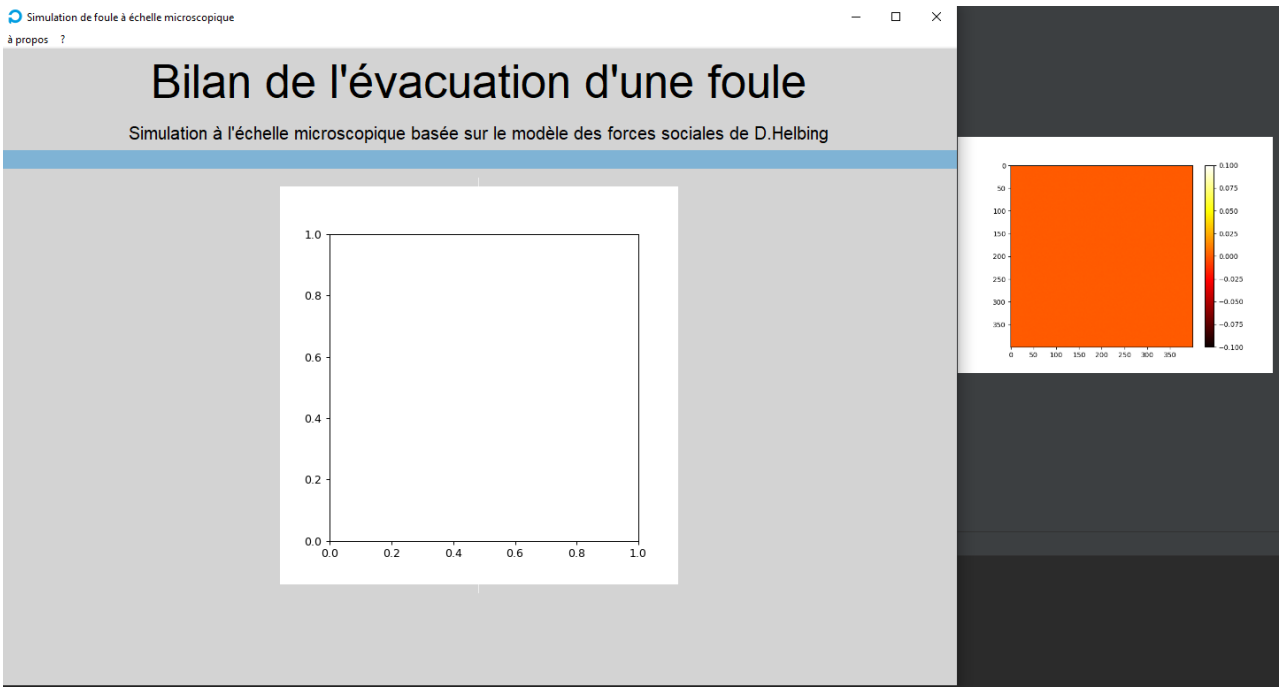


Figure 99 : Version Finale de l'IHM Bilan

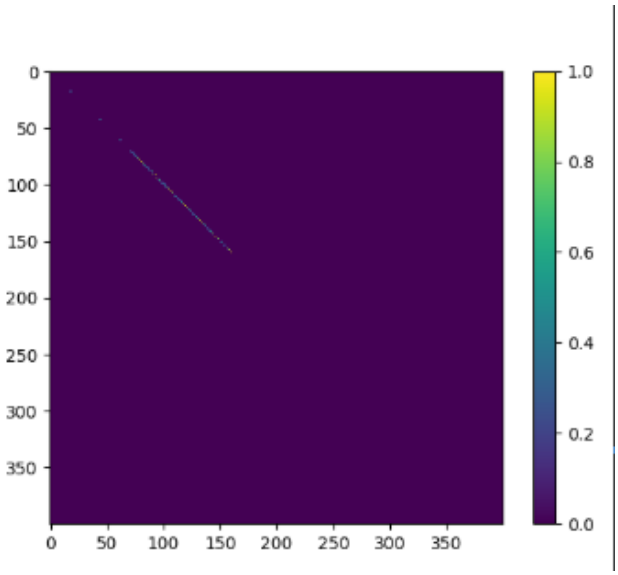


Figure 1010 : Résultat de la Carte de chaleur pendant les tests

Malheureusement, lorsque l'on fait la même manipulation dans le projet, l'échelle à droite n'est pas la même. Et le nombre de passage des personnes reste négligeable et donc rien ne s'affiche.

- Les forces de répulsion entre les obstacles et les personnes ne fonctionnent malheureusement pas mais nous comptons régler ce problème avant de rendre le code source à notre enseignant ultérieurement. Nous pensons que le problème vient soit de la détection du point qui applique la force de répulsion ou des constantes utilisées lors des calculs qu'il faut changer. Peut-être ces 2 possibilités.

- Les poids présents sur l'IHM ne sont pour l'instant pas fonctionnels.

IV.2 – Les évolutions Possibles

Différentes évolutions sont possibles en partant de notre projet :

- Il sera possible de créer de nouveaux obstacles de différentes formes (circulaire par exemple).
- De plus, on pourra rajouter une bordure physique au Canvas pour contenir les personnes à l'intérieur en exerçant une force de répulsion.
- On pourra également changer la détection des personnes, qui est pour le moment un cercle de détection autour de la personne, en un arc de cercle représentant le champ de vision de la personne afin de ne détecter seulement les personnes en face d'elle.
- Une autre évolution possible est que le Canvas s'adapte selon l'environnement généré. En effet, pour l'instant qu'importe l'environnement chargé celui-ci sera visualisée et convertit dans un Canvas de taille 400x400 pixel. Cela est dommage car si par exemple l'on veut modéliser l'évacuation d'un couloir, sur l'IHM on n'aura pas l'impression que l'évacuation se passe dans un couloir mais plutôt une salle. C'est pour cela qu'ajouter cette fonctionnalité pourrait être intéressante.

IV.3- Comparaison de notre modèle par rapport à des situation réelles

Nous ne trouvons pas notre modèle très représentatif de la réalité malgré le fait que nous ayons implémenté le modèle des forces sociales de D.Helbing. En effet, il suffit qu'une trop grosse pression soit exercée sur une personne pour que celle-ci se retrouve expulsée à l'extérieur, faussant tous les résultats. De plus sans force de répulsion entre les obstacles, nous ne pouvons pas réaliser toutes les situations réellement possibles. Nous ne pouvons pas, par exemple, afficher la situation où il se forme un bouchon en arc de cercle autour de la sortie.

Conclusion

Nous sommes fiers du travail que nous avons réalisé malgré certains objectifs qui n'ont pas été atteint. La phase de modélisation du problème de recherche sur le modèle des forces sociales de D.Helbing été très intéressante et enrichissante pour nous. Les réunions régulières avec notre encadrant M. NERON furent très plaisantes car cela nous permettait de rester sur la bonne voie pour que notre réalisation soit bien conforme aux attentes. Cette méthode de travail nous a permis de mettre en œuvres toutes nos compétences et connaissances en génie logiciel et programmation python ce qui fus très formateur. Malgré des difficultés rencontrées lors de ce projets nous avons su y faire face et prendre les bonnes décisions en réfléchissant en équipe, nous en ressortons avec un très bon souvenir. Au moment de la réalisation de ce rapport, notre code n'est pas totalement terminé, nous allons continuer de travailler dessus afin de remplir le maximum d'objectifs possible.

Remerciements

Nous tenons à remercier M.NERON pour son encadrement et son expertise sur le sujet. Il a su nous guider lors de nos interrogations et son encouragement a été un réel moteur nous permettant de nous surpasser.

Simulation de mouvement de foule à échelle microscopique basé sur le modèle des forces sociales de D. Helbing

Résumé : Ce document est un rapport sur la réalisation du projet « Simulation de mouvement de foule à échelle microscopique basé sur le modèle des forces sociales de D. Helbing ». L'objectif était de créer une application conviviale qui permet à l'utilisateur de lancer et d'analyser une simulation d'évacuation de population à l'échelle microscopique (bureau, couloir…) et pouvoir y naviguer dans le temps. Pour réaliser cette simulation nous nous sommes basés sur le modèle des forces sociales de D.Helbing. Les méthodes et matériels utilisés pour répondre aux objectifs sont présentés tout le long de ce rapport réalisé en 2021 en quatrième année du cycle ingénieur Polytech au département informatique à Polytech Tours.

Mots clé : Helbing, Forces Sociales, python, Simulation, évacuation

Abstract : This document is a report on the realization of the project "Simulation of crowd movement on a microscopic scale based on the social forces model of D. Helbing". The goal was to create a user-friendly application that allows the user to launch and analyze a population evacuation simulation on a microscopic scale (office, corridor ...) and be able to navigate through it. To carry out this simulation we took D. Helbing's social forces model as a base. The methods and materials used to meet the objectives are presented throughout this report, produced in 2021 in the fourth year of the Polytech engineering cycle in the IT department at Polytech Tours.

Key words : Helbing, social forces, python, Simulation, evacuation