VI51: VIRTUAL LIFE SIMULATION

Hadrien Beaufils
Hubert Lacôte
Guillaume Stuber

Simulation de l'évacuation d'un bâtiment

Rapport de Projet : Sujet 4

Responsable de l'UV : Stéphane Galland

P2010

Sommaire

Introduction

Dans le cadre de l'UV VI51 (Virtual Life Simulation), nous avons eu l'opportunité de mettre en pratique les connaissances acquises tout au long du semestre à travers un projet simulant des exemples de cas réels où la simulation de vie artificielle pouvait être appliquée. C'était pour nous l'occasion de coder nos propres comportements, de réfléchir à des structures intelligentes pour gérer notre simulation ou encore d'utiliser des outils développer dans le laboratoire du SET, à savoir le couple Jasim/Janus.

Nous avons choisi de traiter le sujet concernant l'évacuation d'un bâtiment. Il s'agit effectivement de présenter une simulation d'un bâtiment d'une entreprise/d'un laboratoire (ici, le modèle 3d n'était rien d'autre que les locaux du SET).

I. Pré-requis

1. Contexte du projet

Les buildings sont communs dans les villes modernes. Ce projet s'intéresse à l'usage professionnel des buildings. Les gens qui se trouvent à l'intérieur ont des tâches à réaliser : aller dans une pièce, à un endroit, faire un travail, … Mais lorsqu'un événement dangereux se produit, ces personnes seront tentées d'évacuer les lieux aussi vite que possible. Elles essayeront de se diriger vers la sortie la plus proche, si elles en perçoivent une

2. Buts du projet

Le but de ce projet est de concevoir une application permettant de simuler les comportements basiques humains ainsi que les activités qui y sont liées. Il faudra pour cela que l'application soit en mesure de simuler différents types de personnes (par exemple des personnes âgées, jeunes, aveugles ou encore à mobilité réduite …).

3. Rappel des éléments obligatoires du cahier des charges

a. Modèle environnemental

Le modèle environnemental de notre simulation sera continu et en 3D. Les seuls objets présents dans notre monde seront des murs. Il n'y aura pas de mobilier, ni de portes dans un premier temps. Chaque entité se déplacera suivant un modèle cinématique. Les entités possèderont un certain nombre d'attributs qui reflètera son état à un instant t :

Santé (entre 0 et 100)
Mobilité (entre 0 et 100)
Vue (entre 0 et 100)
Ouïe (entre 0 et 100)
Social (entre 0 et 100)
Panique (non paniquée, modérée, en panique)

Toutes les personnes seront dans l'état « non paniqué » lors de leur création et le niveau de leur caractéristique « Social » suit une loi normale.

Type de personne générée	**Caractéristiques physiques**	**Probabilité d'apparition**
Personne en bonne santé	Santé, Mobilité, Vue et Ouïe à 100	85%
Personne à mobilité réduite	Santé, Vue et Ouïe à 100	8% =

	Mobilité = {20;50}	{3%;5%}
Personne souffrant de cécité partielle	Santé, Mobilité et Ouïe à 100 Vue = 50	3%
Personne souffrant de troubles de l'audition	Santé, Mobilité et Vue à 100 Ouïe = 50	4%

b. Comportements

Les probabilités données ci-après sont les probabilités choisies dans le cadre de notre simulation. L'utilisateur pourra modifier des variables dans le programme si il le souhaite. Les événements extérieurs seront déclenchés par l'utilisateur du logiciel.

Pour plus de réalisme, chaque entité aura un certain nombre de tâches à effectuer qui évolueront au fil de la journée. Ces tâches pourront en induire d'autres (aller voir un collègue pour des tâches difficiles).

Standards

Ce comportement est celui d'une personne avec la caractéristique « panique » à l'état « non paniqué ». Il y a deux types de personnes qui ont des comportements différents : les employés et les visiteurs. Les employés ont un lieu de travail qui leur est affecté aléatoirement parmi des zones définies. Les visiteurs auront pour objectif de rencontrer une ou plusieurs personnes.

Chaque personne aura des besoins : faire des pauses, manger, aller aux toilettes. Chacune d'elle satisfera ses besoins selon les probabilités suivantes :

Activité	**Probabilité**
Faire une pause	0 / j : 5% 1 ou 2 / j : 60% 3 ou 4 / j : 35%
Aller aux toilettes	0 / j : 15% 1 ou 2 / j : 60% 3 à 10 / j : 25%
Aller manger si planning vide et quand pas influencé	[11h30; 12h [: 10% [12h; 12h30 [: 40% [12h30; 13h [: 40% [13h; 13h30 [: 10%
Suivre les autres (pour les pauses ou pour aller manger)	Dépend de la valeur de la caractéristique sociale et de l'importance de la tâche actuelle

D'évacuation

Des éléments déclencheront des changements de comportement impliquant l'évacuation du bâtiment. Dès qu'une personne sera confrontée à un élément déclencheur, sa caractéristique « panique » passera à l'état « modéré » si elle est à l'état « non paniquée ».

Le feu sera choisi comme élément déclencheur obligatoire.

Voici un tableau présentant les probabilités de passer d'un état « modéré » à un état « en panique» en fonction de la proximité des événements obligatoires :

Événement	**Probabilité de passer de l'état « modéré » à l'état « en panique »**
Feu	De 0 à 5m : 25% De 6 à 20m : 5% Hors de vue (alarme) : 0,1%
Fumée	Pas d'altération de la vue : 5% Visibilité inférieur à 5m : 30%

Voici un tableau présentant la propagation des événements obligatoires :

Événement	**Propagation**
Feu	Propagation radiale ne tenant pas compte des murs, de vitesse constante = 4m / min
Fumée	Propagation radiale ne tenant pas compte des murs, de vitesse constante = 10m / min

- **Modérée**

Ce comportement est celui d'une personne avec la caractéristique « panique » à l'état « modéré ». C'est le comportement d'évacuation calme, c'est-à-dire que les personnes se dirigent diligemment vers la sortie la plus proche. Ils sont maîtres d'eux mêmes, et réagissent en fonction de leur caractéristique « sociale ».

- **En panique**

Ce comportement est celui d'une personne avec la caractéristique « panique » à l'état «panique ». A ce moment là, les personnes sont prêtes à tout pour sauver leur vie. Elles pourront bousculer ou encore blesser des gens. Ces blessures pourront diminuer les caractéristiques « santé », « vue », « mobilité » ou « ouïe ».

4. Rappel des éléments facultatifs du cahier des charges

a. Ajout de nouveaux éléments déclencheurs

De nouveaux événements déclencheurs pourront être introduits. Cela permettra de créer de nouveaux modes de propagation de la panique et de simuler de nouveaux comportements.

Événement	Description	Comportements induits
Attaque à main armée dans le cas d'une prise otage	Une ou plusieurs personnes imposent un comportement à d'autres individus.	Comportement d'agresseur et d'agressé.
Alerte à la bombe	Une personne extérieure préviendra par téléphone un employé. Suivant ses caractéristiques « sociales » et son rôle, il préviendra différemment ses collaborateurs ce qui influera sur leur caractéristique « panique ».	
Infection par des zombies	Un visiteur pourra être porteur d'une maladie. Des personnes deviendront porteuses de cette maladie lorsqu'ils entreront en contact avec une personne contaminée.	Comportement zombie.
Attaque biochimique		

Déclenchement d'événements en chaîne

Les événements déclencheurs pourront provoquer une modification de l'environnement (configuration du terrain), et même engendrer d'autres événements (feu + bouteille de gaz = bombe).

Ajout de nouveaux comportements

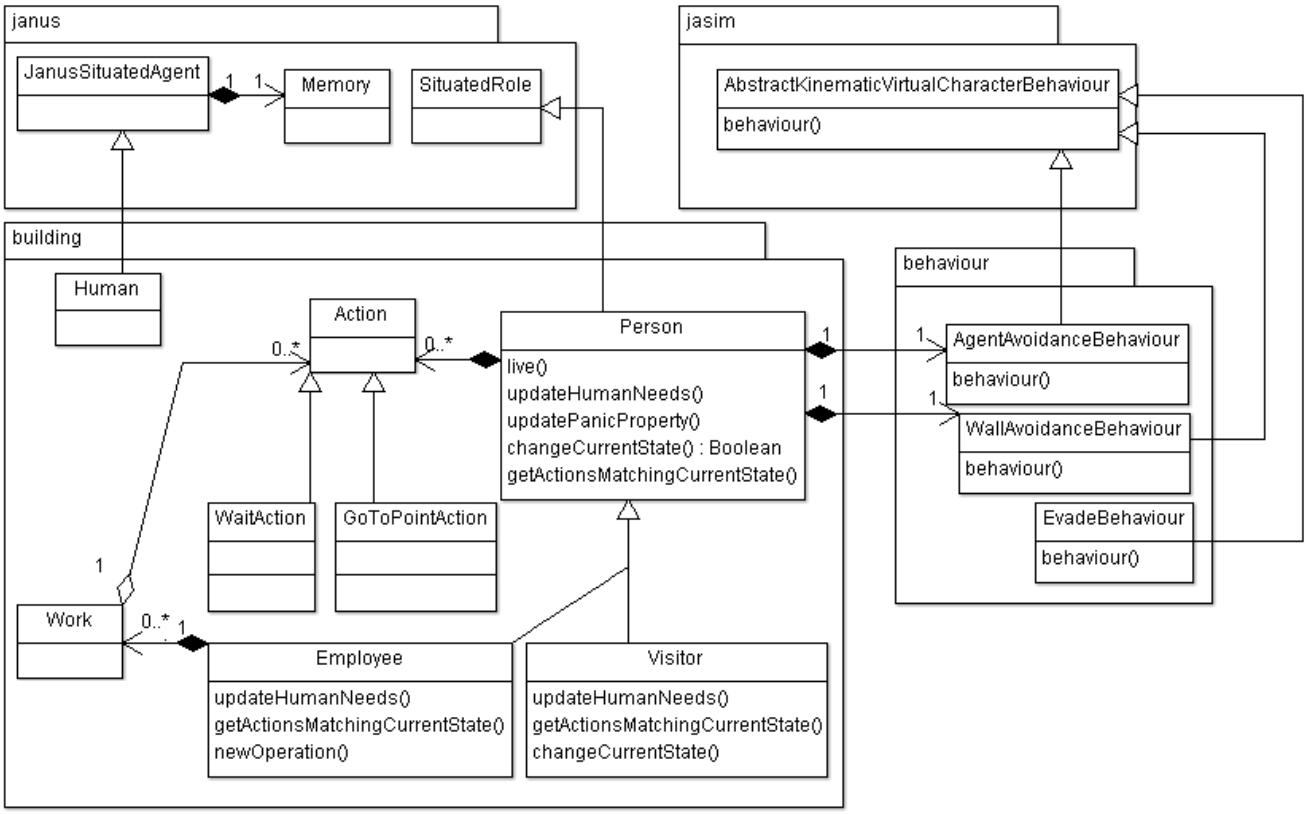
- Évolution du comportement standard : ajout de différents rôles pour les employés qui auront de nouvelles tâches ou de nouvelles obligations plus précises à effectuer (secrétaire, responsable, livreur, clients) ;
- Évolution du comportement de panique :
  - Personnes deviennent agressives et peuvent être amenées à tuer des personnes volontairement (des affinités seront ajoutées entre les personnes).
  - Personnes bloquées par le feu : tentative suicidaire (passer à travers le feu). Sauter par les fenêtres si possible, utiliser un ascenseur qui peut se bloquer.
- Portes verrouillées dans un bureau : des personnes altruistes se trouvant à l'extérieur du bureau pourront aller chercher un outil (hache, ...) pour ouvrir la porte.
- Intervention de pompiers dans le cas d'un feu :
- Les pompiers sont altruistes, sauvent des personnes,
- Ils se déplacent par binôme et rasent les murs.
- Intervention de démineurs dans le cas d'une bombe : déplacement lent. Le démineur peut être un robot.

b. Interface graphique

- « Capteurs » de statistiques (nombre de personnes passant à un certain point) ;
- Affichage de l'état de panique moyen/ individuel (code couleur sur les personnages);
- Sélection d'une personne et affichage des caractéristiques ou des actions prévues.

II. Conception

1. Description de l'architecture logiciel



2. Explications des choix de conception

Dans un premier temps, nous allons vous expliquer comment fonctionne notre simulation. Janus est une plateforme multi-agent qui fonctionne sur le schéma Rôle/Organisation. En effet chacun de nos agents a au moins un rôle qui appartient à une organisation.

Nous avons choisi une approche simple : nous savons que peu importe le type de personne que nous avons dans notre simulation, elles ont toutes des besoins simples (faim, besoin de pause, envie d'aller aux toilettes) en communs et des besoins plus spécifique propre au « rôle » qu'ils joue dans la simulation. C'est pourquoi nous avons deux rôles, Employee et Visitor qui toutes deux dérivent d'une classe mère Person. Cette classe gère tous les aspects basiques d'une personne, que les Employee et les Visitor ont en commun (les besoins communs énoncés plus haut). Elle possède des méthodes simples pour exécuter des actions simples : goTo, wait. Nous reviendrons sur les actions tout à l'heure. Quant aux classes filles, elle ne rajoute que les actions « spécifiques » qui sont traduites en actions « basiques » par Person. Exemple : si mon

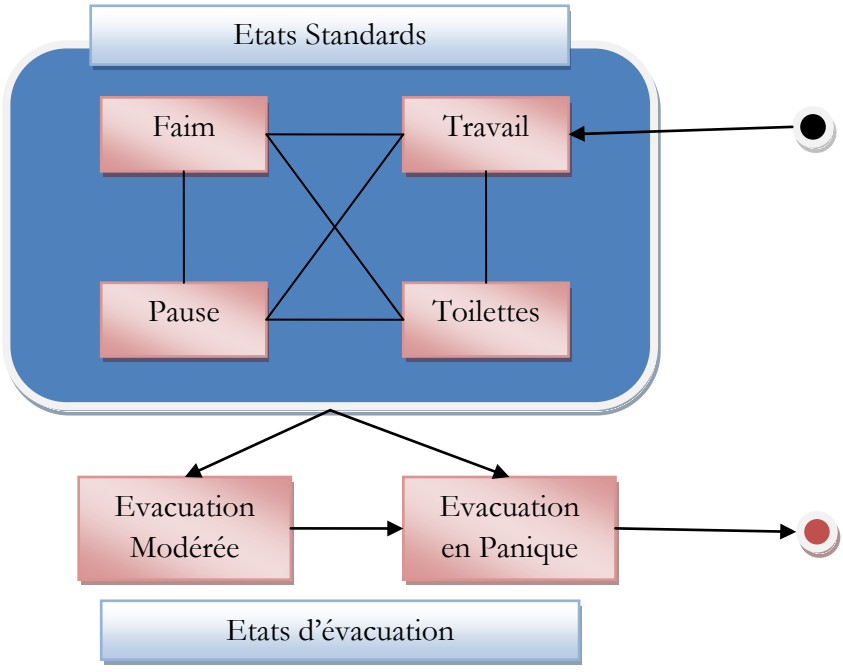
Agent qui a le rôle d'Employee veut effectuer l'action spécifique « travailler à mon bureau», cette action sera découpée en actions de base : « aller à mon Bureau », « attendre le temps que mon travail soit fini ».

Ainsi c'est la classe fille Employee (ou Visitor) qui va s'occuper de déterminer si le travail est prioritaire vis-à-vis des autres besoins de l'agent. Au contraire, c'est la classe Person qui s'occupe de gérer la panique.

Les actions effectuées par les agents dépendent de l'état dans lequel ils se trouvent. Le schéma ci-dessous montre que l'ont peut passer dans tous les états basiques des uns vers les autres, ils sont étroitement liés. Et de chacun de ces états on peut passer dans un des deux états d'évacuation. Le changement d'un état à un autre est soumis à un système de priorité quant aux besoins de l'agent. Ces besoins sont au nombre de quatre :

- La faim : dépend de l'heure habituelle pour manger, ce besoin possède trois stades : pas de besoins quand on se trouve avant l'heure habituelle de l'agent pour manger/ besoin lorsqu'on est dans son heure / Gros besoin quand on est après cette heure.

- Les pauses : une jauge de 0 à 100% évoluant au cours du temps (plus ou moins vite suivant le nombre de pauses que l'agent à besoin en moyenne par jour). Cette jauge est traduite en deux paliers pour effectuer les priorités avec les autres besoins. (pas besoin/besoin)



- Les toilettes : également une jauge de 0 à 100 % qu'on traduit en 3 paliers pour la comparer aux autres besoins.(pas besoin/besoin/très besoin)

- Enfin le travail : chaque travail possède une priorité également sur 3 paliers (pas besoin/besoin/très besoin)

A chaque changement d'état, on charge les nouvelles actions(actions dérivés de la classe PersonAction) le concernant. Attention pour un retour à l'état « WORK », nous avons créé une pile de travail qui est reprise après une interruption (classe Work), et permet de terminer un travail commencé.

3. Explications du choix de l'interface graphique

Comme nous avons utilisé pour ce projet le couple Jasim/Janus, il nous a été fourni une démo avec Jasim-Viewer utilisant Live3d.

4. Techniques utilisées

a. Recherche de chemin

Afin que nos agents puissent se déplacer de manière naturelle dans le bâtiment, nous avons choisi de discrétiser l'environnement en pièces. Les coordonnées de chaque pièce sont définies manuellement dans un fichier Graph XML. Chaque pièce peut être reliée à plusieurs autres pièces. Le fichier est ainsi chargé en mémoire et convertit en graphe afin que nous puissions lancer un algorithme de recherche du plus court chemin. L'agent n'a ainsi plus qu'à suivre le chemin général indiqué par l'algorithme en utilisant le comportement Seek pour se déplacer de portes en portes.

Algorithme utilisé :

Nous devions utiliser un algorithme qui permette aux agents d'avoir des comportements naturels (ne pas faire tout le tour du bâtiment pour se rendre dans le bureau voisin par exemple) tout en étant assez performant pour ne pas ralentir la simulation.

Nous avons opté pour l'algorithme A* qui allie rapidité et efficacité. Cela aurait été trop coûteux et inutile dans notre cas d'utiliser un algorithme tel que Dijkstra pour obtenir la meilleure solution. L'algorithme A* que nous utilisons permet d'obtenir des chemins très logiques sans avoir besoin d'effectuer un parcours coûteux de tout le graphe.

Implémentation :

Nous avons retenu pour l'implémentation de l'algorithme une des méthodes proposées par Ian Millington, dans son livre Artificial Intelligence for Games, 2008.
Pour la structure de données, nous utilisons pour la liste ouverte une file d'attente prioritaire qui permet très facilement d'obtenir de très bonnes performances. Comme conseillé, nous n'utilisons pas de liste fermée mais chaque nœud contient directement une valeur énumérée qui permet de trouver dans quelle liste il se trouve. Cela évite plusieurs recherches coûteuses.

Pour la fonction heuristique, nous avons après plusieurs essais choisis d'utiliser la distance de Manhattan qui semble donner de bons résultats. En effet, pour aller d'un point à un autre dans un bâtiment, un agent se déplace rarement en ligne droite, donc la distance euclidienne donnait de trop faibles résultats

Il nous fallait aussi une fonction de coût pour calculer le coût de déplacement d'une pièce à une autre. Pour cela, nous avons aussi choisi d'utiliser la distance de Manhattan par rapport au centre des deux pièces. Le coût de déplacement d'une pièce à une autre est pondéré si la pièce de destination est une pièce où l'agent a déjà perçu du feu.

b. Graph XML

Pour créer notre graphe, nous avons construit un fichier xml. Celui-ci nous permet de définir nos nœuds (les pièces, définies par un Oriented Bounding Rectangle), nos liens entre chaque nœuds (les waypoints par lesquels passer pour arriver à la pièce suivante) ainsi que des points clés (position des toilettes, de l'endroit où manger,etc..). Ces différentes informations sont définies comme suit :

```
<nodes>
        <node id="1"
            x1="-131.2" y1="50.2"
            x2="-126.1" y2="49.5"
            x3="-130.6" y3="54.1"
            isAnOffice="true">
        </node>
        …
</nodes>
<links>
        <link id1="27" id2="32">
            <waypoint name="pt1"
                x="-130.9"
                y="57.4"/>
        </link>
        …
</links>
<keypoints>
        <keypoint name="relaxing_place" id="27" />
        …
</keypoints>
```

Explications:

On définit nos nœuds (node) dans la première partie via un id et trois points, qui formeront deux vecteurs pour notre OBR. La variable isAnOffice permet de stocker ce noeud en tant que lieu de travail. Les liens (link) se définissent entre deux ids (référençant des

nœuds/pièces du paquet des nodes). Ensuite entre les balises <link>, il faut définir les waypoints s'il y en a. Enfin les keyPoints associent un id (node) à un nom.

c. Evitement des murs

En supplément à la recherche de chemin, un comportement permettant d'éviter les murs est intéressant. Le comportement en question a été implémenté sous le nom **WallAvoidanceBehaviour**. Pour le réaliser, nous avions à notre disposition différentes techniques, que nous avons testé.

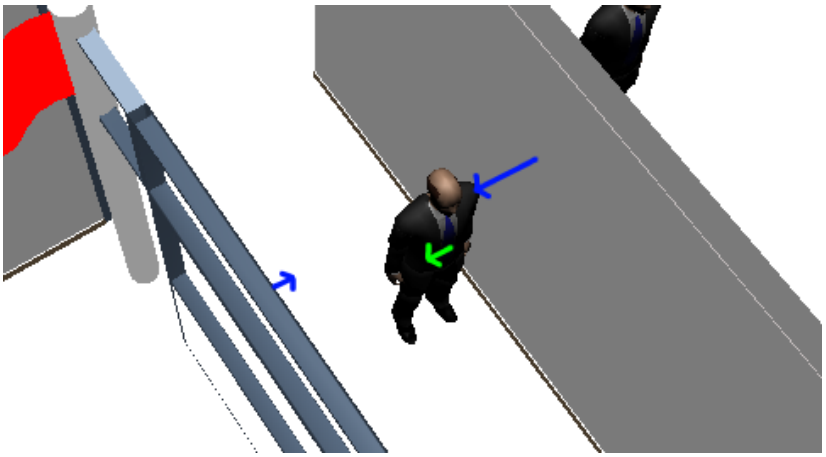
Finalement, nous avons choisit un évitement de murs assez simple : une répulsion juste assez forte pour permettre aux agents de ne pas raser les murs en permanence.

Le principe est le suivant :

L'évitement des murs et configuré à l'aide de deux paramètres qui sont la distance (entre le mur et l'agent) en dessous de laquelle l'influence du mur sera prise en compte, et un facteur de répulsion qui sera appliqué à la fin du calcul de répulsion (afin de pondérer celle-ci).

Les forces de répulsions sont calculées puis amplifiées ou réduites en fonction de la distance au carré entre l'agent et le mur. Ensuite elles sont additionnées pour donner une force unique de répulsion.

Sur la capture d'écran suivante, nous avons représenté les forces de répulsion en bleu et la force de répulsion finale (composée de toutes les forces) en vert :



Seuls les murs proches sont pris en compte. Sur chacun de ces murs, la force est calculée à partir du point le plus proche de la position de l'agent, et plus les forces éloignées, moins elles ont d'influence sur l'agent. Dans ce cas, la force de répulsion éloigne l'agent du mur dont il est le plus proche.

d. Evitement des agents

Le comportement d'évitement des agents vient naturellement après celui des murs. Aussi, nous avons utilisé la même approche, d'autant plus que les obstacles en question sont mobiles. Pour rester cohérents, nous l'avons nommé **AgentAvoidanceBehaviour**.

Les seules différences notables sont le fait considérer les perceptions dynamiques, et celui de pondérer la force de répulsion calculée à l'aide d'un facteur plus faibles que dans le cas de l'évitement des murs. La distance en dessous de laquelle les forces de répulsions sont calculées est, elle aussi, plus faible.

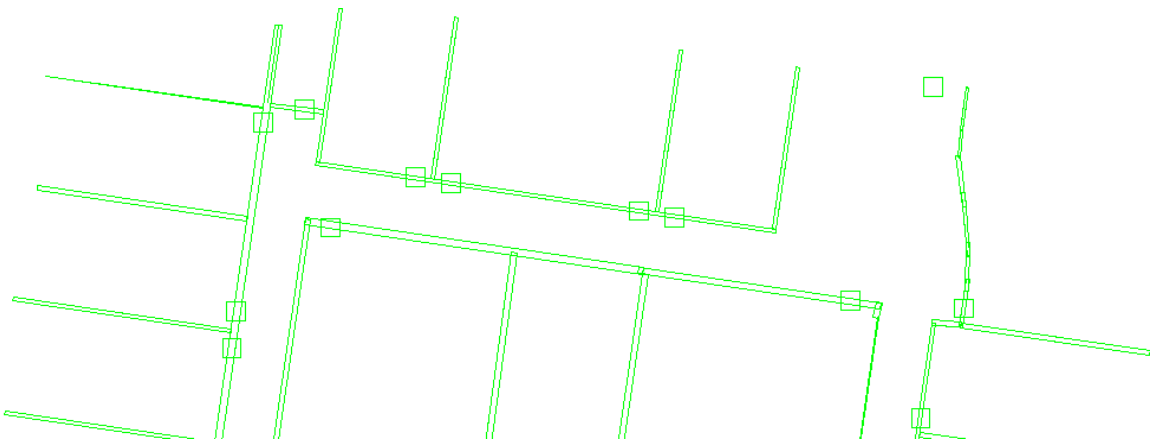
5. Problèmes rencontrés

a. Evitement des murs

Malgré la simplicité finale du comportement permettant d'éviter les murs, nous avons rencontré beaucoup de difficultés provenant des perceptions notamment.

Il existe des murs de différents types : « INSIDE » et « SPANNING ». Au départ, nous ne le savions pas et nos agents développaient des comportements peu compréhensibles dans certains cas. En effet, les murs de types *SPANNING* ne correspondent pas aux murs visibles dans l'application, mais à des murs « couvrants » qui ne devaient pas entrer en compte dans le calcul. En pensant que l'erreur provenait plutôt de l'algorithme d'évitement des murs, nous faisions fausse route. Ca n'est qu'après plusieurs tentatives que nous avons compris d'où venait le problème.

Un deuxième point problématique était la conception même du modèle 3D. Nous pensions que les portes étaient des espaces « ouverts », comprenons par là qu'aucune perception n'intervenait à ces endroits. Or, l'image de perception des agents ci-dessous montre bien que les portes sont « murées » :



Nous avons réussi à contourner le problème en désactivant l'évitement des murs lorsque l'on souhaite passer d'une pièce à l'autre.

b. Des idées non mises en œuvre

Nous n'avons pas pu implémenter tout ce que nous aurions aimé implémenter, faute de temps. Parmi les idées que nous avions, l'utilisation de la mailbox de Jasim pour faire communiquer les agents entre eux. Par exemple, pour une personne aveugle, des agents altruistes auraient pus envoyer un message à cette personne afin de lui indiquer la direction de la sortie. Pareil pour prévenir un agent ayant des troubles auditifs que l'alarme retentie.

III. Guide d'Utilisation

Documentation des contrôles de l'interface graphique

a. Contrôle du flux de la simulation

Le flux de la simulation est géré à l'aide de 3 boutons, dans l'ordre : Play/Pause, Avancer d'un pas de simulation et Stop :

- Le bouton **Play/Pause** permet de lancer la simulation en mode « continu » et de la figer à volonté.
- Le bouton **Avancer d'un pas de simulation** a aussi pour fonction de lancer la simulation mais il permet avant tout de n'avancer que d'une itération (un pas de temps), donc en mode « pas-à-pas ».
- Le bouton **Stop** arrête définitivement la simulation.

b. Changement du type d'interaction

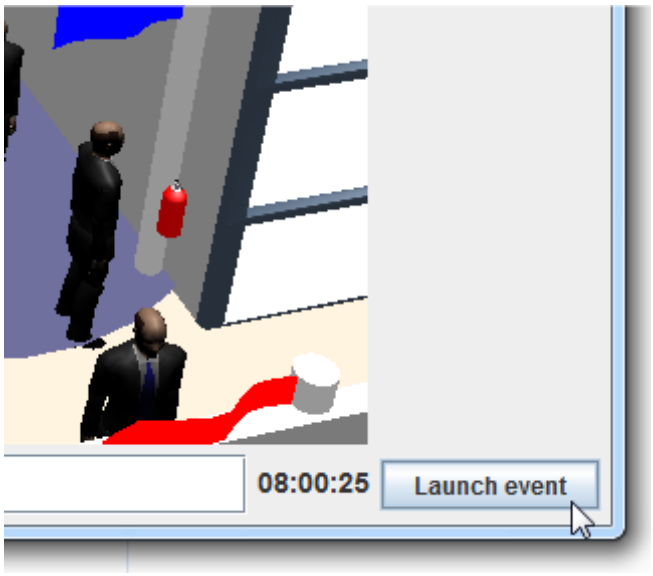
Le **picking** permet à l'utilisateur de lancer des actions ou de prélever des informations en cliquant sur la scène. Techniquement parlant, un événement particulier est généré et permet notamment de retrouver le point tridimensionnel correspondant à l'emplacement du clic dans la scène 3D.

Deux interactions sont possibles : récupérer des informations sur un agent, ou encore créer un événement déclenchant l'évacuation du bâtiment. Pour changer le type d'interactions utilisable, nous avons créé un bouton permettant de passer du prélèvement d'informations, au lancement d'événement. Ce bouton est situé en bas à droite de l'interface.

Il peut se trouver dans 2 états différents.

- **Lorsqu'il affiche « Pick info »** (vue de gauche), cela signifie que nous avons actuellement la possibilité de lancer des événements. Dans ce cas, la scène est entièrement affichée dans la zone centrale. Si on souhaite récupérer des informations, il faut cliquer sur le bouton pour passer en mode « Pick info ».
- **Lorsqu'il affiche « Launch event »** (vue de droite), cela signifie qu'on a actuellement la possibilité de récupérer des informations. Dans ce cas, un panneau latéral vient couvrir la

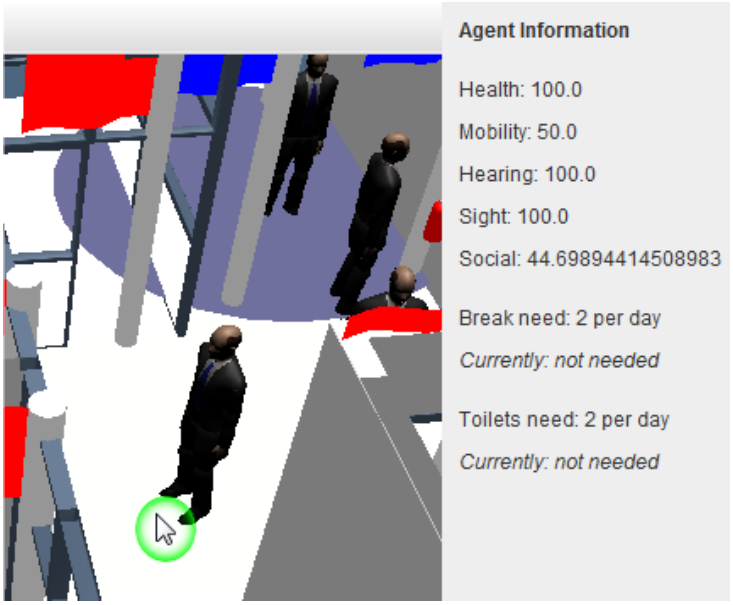
partie droite de la scène. De la même manière, un clic sur ce bouton aura pour effet de passer en mode « Launch event » pour lancer des événements.



c. Récupérer des informations sur les agents

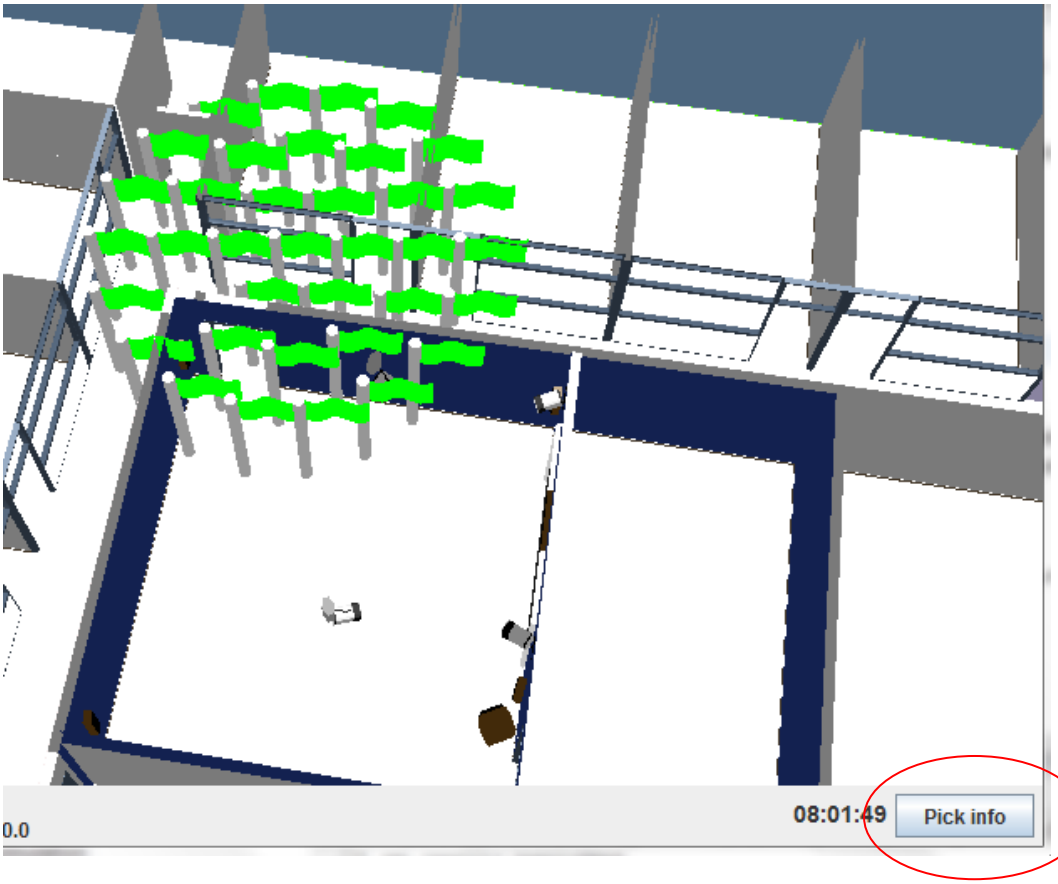
En mode « Pick info », nous avons la possibilité de récupérer les propriétés internes de l'agent. Pour ce faire, il suffit de cliquer sur le sol juste à ses pieds sur la scène 3D, comme le montre l'emplacement de la souris (entourée d'un cercle vert) sur la capture d'écran ci-dessous.

Nous avons ainsi accès aux informations sur les propriétés actuelles de l'agent :



d. Lancer des événements d'évacuation

En mode « Launch Event », il suffit de cliquer au sol pour déclencher le feu (représenté par des drapeaux) dont la surface va alors s'élargir au fur et à mesure de sa propagation.



Conclusion

1. Eléments abordés et performances du logiciel

Pour résumer, nous avons traité les éléments suivant :

- La génération des entités selon des règles de probabilités définies pour des personnes en bonne santé, à mobilité réduite, souffrantes de cécité partielle ou encore de troubles de l'audition.
- Les comportements standards pour les 2 types d'agent (employé et visiteur) selon des besoins évoluant en fonction des attributs des agents en question.
- L'attribution de tâches à réaliser pour ces 2 types d'entités (travailler, manger, aller au toilettes, faire une pause, etc.)
- La création d'événements déclenchant l'évacuation du bâtiment (à l'aide d'un lanceur d'événement simplement au clic), et la propagation selon des règles définies dans le cahier des charges. Le feu a été totalement intégré à la solution.
- L'évolution des états de panique en fonction de la proximité d'un danger.
- La récupération des informations des agents évoluant dans la simulation
- Une gestion des déplacements améliorée avec prise en compte des obstacles (murs et agents), et génération de chemin.

2. Critique des résultats

Les résultats en eux-mêmes sont satisfaisants. La plus part des points traités permet de réaliser ce pourquoi elle a été crée. Le côté négatif et que par manque de temps, nous n'avons pas réussi à aller plus loin dans le perfectionnement des comportements. Seuls quelques points facultatifs ont pu être traités. D'autres auraient étés très intéressant à intégrer à notre solution tels que l'altruisme (personnes particulières ou pompiers), le déclenchement d'événements en chaîne (le feu qui déclenche une explosion) ou encore des événements spéciaux (attaques biochimiques). L'implémentation de ces éléments aurait été un vrai plus dans notre projet.