**REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE**

**MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE**

**Université Mohamed Khider-BIKSRA**

**Faculté des Sciences Exactes et des Sciences de la Nature et la Vie**

**-Département de l'informatique-**

**Mémoire**

Présenté en vue de l'obtention du diplôme de

**Master INFORMATIQUE**

**THEME**

# Simulation de foules d'humains virtuels en temps réel

Spécialité : Image et Vie Artificielle

Présenté par **: Bichari amine**

Supervisé par **: Mr.Mebarek boucetta**

Soutenu le 27/10/2020, devant le jury composé de :

Présidant **:**

Examinatrice :

Année Académique 2019-2020

# DEDICACE

A mes très chers parents adorés

Qui m'ont toujours soutenu et encouragé à aller de l'avant

Mes très chers frères et sœurs,

À toute ma famille,

À tous ceux que j'aime

Je tiens à remercier mon encadreur : *Mr.MEBAREK BOUCETTA* pour la confiance qu'il m'a accordée, ses encouragements, et ses précieux conseils, sa patience et sa persévérance dans le suivi.

Mes remerciements vont également aux membres du jury qui ont pris le temps nécessaire à

L'évaluation et au jugement de mon travail.

À tous ceux qui ont participé de loin ou de près à la réalisation de ce travail.

Avec l'expression de tous mes sentiments de respect,

Je dédie ce modeste travail.

# TABLE DES MATIERES

## CHAPITRE 3 : LA CONCEPTION



## CHAPITRE 4: IMPLEMENTATION ET RESULTATS

# LISTE DES FIGURES

## CHAPITRE 1



## CHAPITRE 2

## Chapitre 3



## Chapitre 4

# RESUME

l'animation comportementale est une méthode la plus utilisée dans la simulation de foule humaine dans un monde virtuel cette méthode donne de l'humain virtuel l'aspect d'autonomie , l'individu virtuel perçois son monde qui passe ensuite vers l'analyse les données entrante a la fin pour prendre une décision et de faire un action en rapport a la situation perçu , le résultats c'est de simulé un groupe d'humanoïdes qui compose une foule humaine virtuel avec des comportement proche a la réalité.

dans ce mémoire nous sommes intéressés à développer les aspects suivants :

- la représentation du monde

-la planification de chemin dans un environnement virtuel.

- donner un comportement individuel et de groupe

-l'évitement de collisions entre les individu pendant le croisement.

-

# ABSTRACT

behavioral animation is a method most used in the simulation of human crowds in a virtual world this method gives the virtual human the aspect of autonomy, the virtual individual perceives the there world and makes him process the incoming data at the end of making a decision and taking an action in relation to the perceived situation, the result is to simulate a group of humanoid which makes up a virtual human crowd with behavior close to reality.

in this thesis we are interested in developing the following aspects:

- the representation of the world

-path planning in a virtual environment.

- give individual and group behavior

-avoidance of collisions between individuals during crossing

# Introduction général

Les nouvelles technologies nous ont permis depuis quelques années de créer des humains virtuels et de les animer. Plus récemment, l'animation comportementale nous a amené la possibilité de s'immerger dans les mondes virtuels et d'y rencontrer des êtres virtuels. Enfin grâce aux recherches en intelligence artificielle et en vie artificielle, les humains virtuels sont capables d'une certaine autonomie. La simulation de comportements de foules humaine est un sujet d'actualité dans le domaine de l'infographie. L'importance de lancer des simulations de comportement de foules dans différentes situations réside dans l'impossibilité d'accomplir, d'une manière rapide et concluante, le comportement réel avec des humains. Pour cette raison, la simulation de comportement de la foule concerne beaucoup de domaines d'applications tels que la sécurité, le génie civil, l'urbanisme...etc. Les applications s'intéressent peu à la finesse et au réalisme des mouvements et des interactions locales. Au contraire, l'animation graphique cherche à simuler et animer des environnements virtuels qui sont généralement peuplés d'humains virtuels, et s'intéresse surtout au réalisme local et au réalisme visuel plutôt qu'au réalisme d'échelle

L'objectif de ce mémoire est de réaliser une simulation de foule humaine en temps réel dans plusieurs situation ( déplacement et l'étude des mouvement de groupe d'individu , évitement de collision entre individu , suivez de chef etc )

La nature de notre problématique nous a suggéré alors, de faire appel à un certain nombre d'approches et techniques :

- les différents techniques de représentation du monde virtuel
- planification de chemin
- algorithme de recherche
- navigation
- comportement d'individu et de groupe

# Chapitre 1

## 1. Introduction

Dans ce chapitre on va parler sur la notion des environnements virtuels, ce dernier est la base de toute planification de chemin, pour utiliser l'environnement virtuel, il faut qu'il soit bien représenté donc on va voir aussi un nombre de méthodes de représentations et un ensemble d'algorithmes de planification de chemin

## 2. Type d'environnements

Les environnements sont devisés en deux catégories, chaque catégorie à sa propre représentation est son type d'utilisation:

### 2.1. Les environnements statiques

Sont des environnements qui ne subissent pas de modifications au cours de temps. Les environnements statiques se caractérisent par leur structure qui n'est pas amenée à évoluer au cours du temps. [12] Cette propriété implique donc qu'il n'y a pas de nouvelles accessibilités ou obstructions créées pendant le déroulement de la simulation. Puisque la configuration de ces environnements ne change pas, il est donc possible d'effectuer des pré-calcules afin de proposer des structures de données performantes lors de requêtes de planification de chemin ultérieures.

### 2.2. Les environnements dynamiques

Pour leur part présentent des caractéristiques qui évoluent au cours de temps[15]. Nous appelons un environnement dynamique lorsque les positions occupées par certains obstacles peuvent changer dans le temps. Cela peut être dû à des objets qui : modifient la position dans le temps, modifient la forme dans le temps où apparaissent ou disparaissent dans l'espace de travail.

## 3 . Représentation de l'environnement

L'environnement dans lequel les entités évoluent est représenté par une géométrie statique. Cette géométrie traduit la structure de l'environnement et donc les contraintes imposées lors de la navigation. De manière générale, cette géométrie représente les obstacles sous la forme de polygones en 2Det sous la forme de polyèdres en 3D. Cette hypothèse sert de base à un certain nombre de méthodes de représentation de l'espace et s'avère être corrélée avec les méthodes de modélisation d'environnement. Que cet environnement ait été produit avec un logiciel de modélisation 3Dou un outil dédié, pour permettre une interprétation rapide du point de vue des entités le peuplant, il doit être représenté dans une structure de données appropriée.

Nous allons étudier deux méthodes : la représentation exacte de l'environnement, et la représentation approximative de l'environnement.

### 3.1 Représentation exacte de l'environnement

Les représentations exactes de l'environnement vont chercher à organiser les données spatiales tout en conservant intégralement les informations qu'elles contiennent à l'origine.

La méthode utilisée consiste généralement à découper l'espace navigable en cellules convexes de différentes formes (triangles, polygones, trapèzes...). Il existe plusieurs modèles pour effectuer cette subdivision.

- **Triangulation de Delaunay**

La triangulation de Delaunay [3][16] crée un ensemble de triangles en réunissant des points fournis en entrée. L'algorithme d'unification respecte pour contrainte que le cercle circonscrit à un triangle ne contienne aucun autre point que les trois sommets qui le composent. Une conséquence de cette propriété est que l'angle minimum d'un triangle produit est maximisé. La complexité de l'algorithme d'unification est en O (n. ln(n)), avec n le nombre de points fournis en entrée. La propriété la plus intéressante de cette triangulation est que chaque point y est relié à son plus proche voisin par l'arête d'un triangle. Ainsi, cette triangulation peut être utilisée pour représenter l'espace navigable, les points d'entrée étant issus des obstacles.

Une autre utilisation possible de cette triangulation est cette fois dynamique lors de la simulation , pour calculer un graphe de voisinage entre les entités mobiles, qui serviront cette fois-ci de point d'entrée.

- **Triangulation de Delaunay contrainte**

La triangulation de Delaunay contrainte [3] permet de modérer les contraintes de la version standard afin de conserver certaines arêtes de la définition graphique d'origine. Pour se faire, la contrainte du cercle circonscrit à un triangle est modifiée, pour spécifier que tout point inclus dans ce cercle ne peut être relié à tous les points du triangle sans intersecté un segment contraint. Cette triangulation étend la propriété du plus proche voisin en permettant d'y associer une notion de visibilité. Si l'on considère que les arêtes contraintes coupent la visibilité d'un point à l'autre, la propriété de plus proche voisin devient : chaque point est relié à son plus proche voisin visible. La triangulation de Delaunay contrainte est aussi utilisée pour obtenir une subdivision spatiale en triangles , en introduisant les obstacles à la navigation sous la forme d'autant de segments contraints (voir la figure 1.1), la discrétisation de l'espace 'à gauche' avec une triangulation de Delaunay contrainte 'à droite'). Il a été prouvé quelle nombre de triangles produits lors d'une telle subdivision est linéaire en fonction du nombre de points, indiquant que la discrétisation est donc directement proportionnelle à la complexité géométrique de l'environnement.

Cette propriété présente un avantage certain comparativement aux méthodes approximatives, où la discrétisation est fonction de la précision désirée de la représentation.

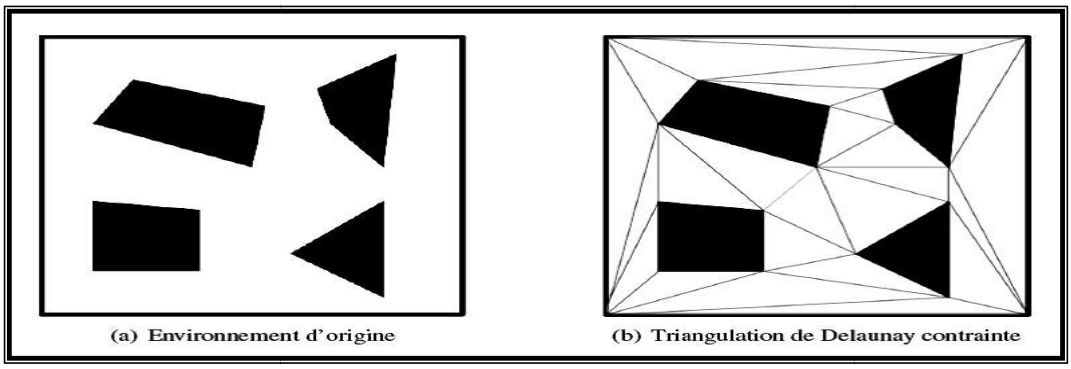



**FIGURE 1.1 :**EXEMPLE DE TRIANGULATION DE DELAUNAY CONTRAINTE

- **Triangulation de Delaunay filtrée**

La triangulation de Delaunay filtrée [4] est une extension de la version contrainte. Deux types de filtrages sont proposés, l'un ayant pour effet d'augmenter le nombre de triangles produits afin d'affiner la représentation de l'environnement, l'autre de le diminuer afin de tenir compte de la visibilité pour l'élaboration d'un graphe de voisinage .Premièrement, concernant son application à la subdivision spatiale, la triangulation de Delaunay est filtrée par l'ajout progressif de contraintes représentant les goulets d'étranglement (voir la figure 1.2). Ainsi, en considérant l'ensemble des arêtes produites, on est sûr de représenter tous les rétrécissements présents dans l'environnement.

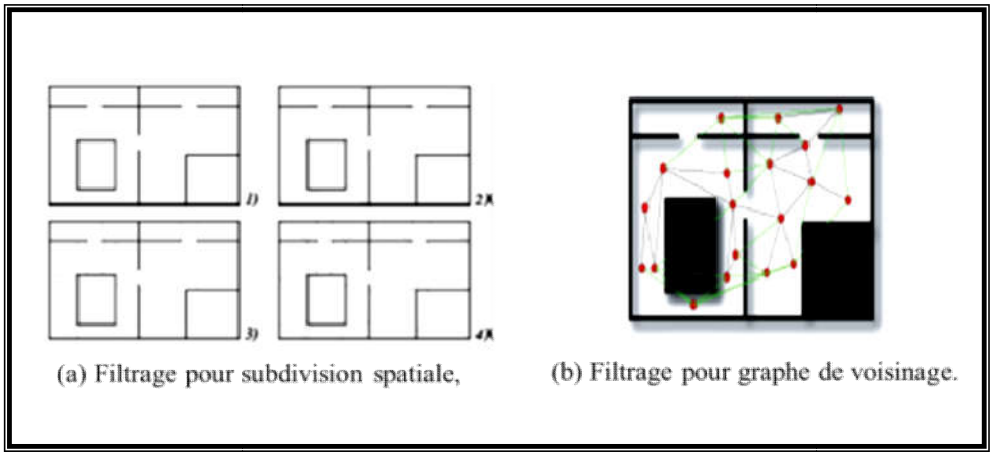



**FIGURE 1.2 :**EXEMPLES DE TRIANGULATIONS DE DELAUNAY FILTREES

Deuxièmement, concernant son application aux graphes de voisinage, la triangulation de Delaunay est filtrée sur un principe équivalent à la triangulation contrainte, en supprimant les arêtes intersectées des

obstacles de l'environnement (voir la figure 1.2). Ainsi, cette triangulation peut servir à déterminer trivialement quels sont les voisins directs visibles d'une entité, et avec un simple parcours de graphe peut sélectionner les entités visibles à une certaine distance.

### 3.2. Représentation approximative de l'environnement

Les représentations approximatives de l'environnement sont sans doute les plus utilisées en animation comportementale, du fait de leur facilité de mise en œuvre. Ces représentations vont décrire l'espace libre de navigation avec des formes géométriques simples telles que des carrés ou des segments. Deux modèles entrent dans cette catégorie : les modèles à base de grilles, et les cartes de cheminement.

- **Modèles à base de grilles**

Le premier modèle de représentation approximative utilise des grilles régulières, formées de cellules carrées en deux dimensions et cubiques en trois dimensions.

La précision de la représentation obtenue dépend directement de la taille des cellules utilisées (plus les cellules sont grandes, moins la représentation est précise).

Pour rappel, l'inconvénient de cette méthode réside dans l'augmentation proportionnelle de l'occupation mémoire due à l'augmentation de la précision. Ce qui influence sur la complexité de recherche de chemin à l'intérieur de l'environnement.

Pour réduire ce problème, une évolution de ce modèle est apparue sous la forme des grilles hiérarchiques (figure 1.3). Cette méthode décrit l'espace navigable par une succession de grilles de plus en plus précises, organisées sous forme d'arbre [2].

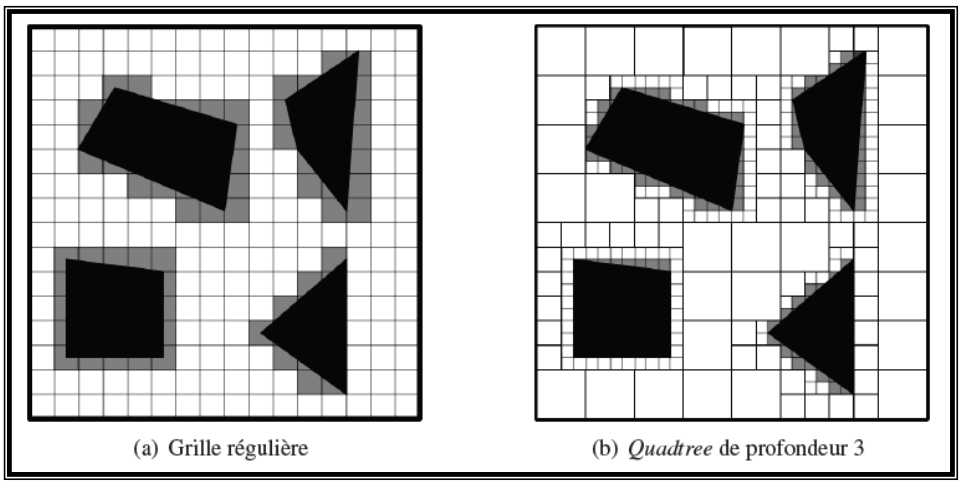



**Figure 1.3** Exemple De Grilles Régulières[2].

Les méthodes à base de grilles sont très utilisées en animation comportementale à cause de leur simplicité de mise en œuvre et de leur rapidité d'exploitation.

- **Cartes de cheminement**

Les cartes de cheminement discrétisent l'espace navigable sous la forme d'un réseau de chemins. Ce réseau est obtenu en reliant des points clefs répartis à l'intérieur de l'environnement (Figure 1.4)[2]. Plusieurs méthodes existent pour créer des cartes de cheminement, différentes dans la manière de créer et de relier ces points clefs[5].

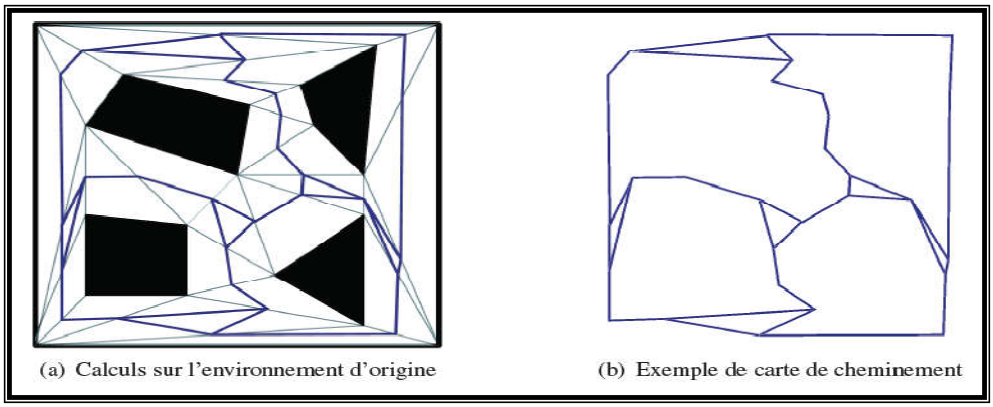



**Figure 1.4** Exemple De Carte De Cheminement[2] .

- **Graphe de visibilité**

Cette méthode utilise les sommets des polygones représentant les obstacles comme points clefs. Les points clefs sont ensuite reliés deux à deux s'ils sont mutuellement visibles, c'est à dire si l'on peut tracer une ligne droite passant par les deux points sans rencontrer d'obstacle (voir la figure 1.5).[17] Les graphes ainsi créés minimisent les distances parcourues , assurant d'obtenir des chemins de longueur minimale. La taille du graphe généré est ici dépendante du nombre de points mutuellement visibles, donc d'autant plus importante que l'environnement est ouvert avec des obstacles ponctuels et disparates.

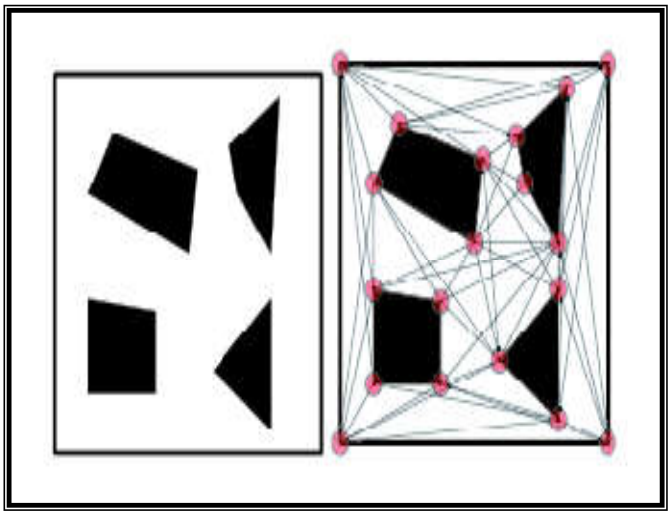


**Figure 1.5** Graphe De Visibilité[17].

## 4. Environnement virtuel informé

Un environnement virtuel informé est un environnement virtuel dont les modèles 3D contiennent non seulement la géométrie de la scène mais aussi toutes les informations pertinentes pour les entités comportementales à simuler comme les éléments symboliques ou sémantiques qui peuvent permettre à ces entités de percevoir, décider et agir.

Les environnements virtuel informé intègrent sur les objets des connaissances sur le savoir (description de l'objet) et le savoir-faire (interactions possibles avec l'objet) et ce de plusieurs manières.

### 4.1 Intégration des objets interactifs

Les objets définit dans l'environnement est géo-localisé, i.e. il connaît sas positions dans l'espace, mais aussi le nœud du graphe topologique qui lui correspond. Le but premier de cette géo-localisation est de rendre les objets accessibles aux algorithmes basés sur la définition topologique, dont notamment la planification de chemin et l'évitement de collision [4].

Il y a deux catégories d'objets interactifs sont ici à prendre en compte : premièrement, les objets statiques dont la localisation spatiale est figée au cours d'une simulation, deuxièmement, les objets dynamiques pouvant se déplacer au cours d'une simulation.

- **Les objets statiques**

Les objets statiques sont fixes dans l'environnement. Ces objets sont dans la plupart des cas des spécialisations de l'effecteur, représentant des équipements accessibles aux inter acteurs.

Trois sous catégories des objets statiques sont :

- Les objets obstacles :les objets obstacles statique doivent être évités par des entités se déplaçant dans l'environnement. Par exemple ; un guichet de vente, un mur[18] (figure 1.6).

**Figure 1.6** Objets obstacles (mur)[18].

- Les objets traversables : les objets traversables joignent plusieurs zones navigables de l'environnement, tout en étant eux-mêmes navigables lors de leur utilisation, ou obstacles dans le cas contraire. Par exemple ; un ascenseur, un escalier, un portique automatique(figure 1.7).

**Figure 1.8** Objets traversables (ascenseur)|18].

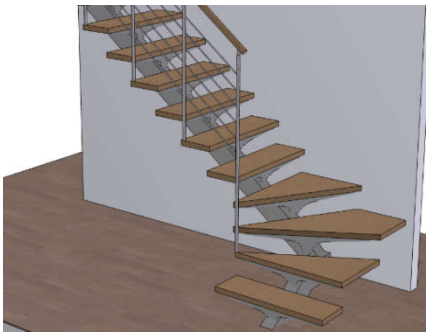


**Figure 1.7** objets traversables (escalier)[18].

- Les objets libres : les objets libres sont disposés de telle façon qu'ils ne gênent en rien la navigation, mais seraient considérés comme des obstacles s'ils étaient situés ailleurs. Par exemple : un panneau d'affichage accroché en hauteur sur un mur ou sur un pilier.

- **Les objets dynamiques**

Les objets dynamiques sont mobiles dans l'environnement. Ces objets peuvent indifféremment être des spécialisations de l'inter-acteur, comme l'individu virtuel, ou de l'effecteur, comme une valise, ou encore l'unification des deux, comme un agent de service dans un lieu publique.

N'étant pas fixes, ces objets ne peuvent être directement intégrés au graphe d'abstraction de l'environnement. Nous utilisons donc la même technique ici que pour les objets statiques libres : les objets dynamiques sont associés au nœud du graphe correspondant à leur géol-ocalisation. De plus, une procédure automatique est mise en œuvre afin de transférer l'objet d'un nœud à l'autre lors de son déplacement.

**FIGURE 1.9** objets mobile ( humains virtuel )

## 5. Humain virtuel

### 5.1 Modélisation des propriétés des humains virtuels

L'objectif majeur de la modélisation des comportements des acteurs est de construire des acteurs virtuels plus réalistes, ces acteurs sont intelligents et autonomes et avec une adaptation, perception et mémoire, capable d'agir librement et avec émotion, d'être conscient et imprévisible .perception et mémoire, capable d'agir librement et avec émotion, d'être conscient et imprévisible

- **La perception**

La perception [4] est définie comme la conscience des éléments dans l'environnement à travers des sensations physiques. Il est réalisé en équipant les agents avec des détecteurs visuels, tactiles et auditifs ainsi ils simulent le comportement quotidien humain (aspect visuel, mouvement, réaction, …). Le sous-système perceptuel le plus important est le système visuel. Une approche basée sur la vision est idéale pour le modelage d'une animation comportementale. Elle offre une approche universelle pour le passage d'information de l'environnement à l'acteur dans le contexte de recherche de chemin.

A un niveau plus haut, nous pouvons décomposer la perception comme suggérée par . La perception d'un acteur peut être limitée aux objets et à d'autres acteurs dans le voisinage. Mais cela limite le nombre de comportements possibles, parce que seules la présence et les caractéristiques d'un objet ou d'un acteur sont impliquées dans la sélection d'un comportement. Les actions des autres acteurs ne sont pas prises en considération.

Le module de perception produit trois types de perception :

- La perception de la présence d'objets et d'acteurs .
- La perception des actions d'acteurs .
- La perception d'acteurs exécutant des actions sur des objets.

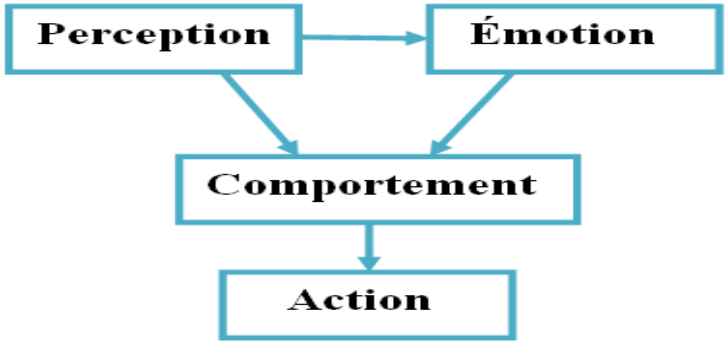



**Figure 1.10** : Structure du modèle comportemental[4].

- **L'émotion**

L'émotion peut être définie comme l'aspect affectif de la conscience : Un état de sentiment, une réaction psychique et physique (comme la colère ou la crainte), subjectivement expérimenté comme un sentiment fort et physiologiquement qui augmente les changements préparant le corps pour une action vigoureuse immédiate[19].

Les acteurs doivent être capables de répondre, avec émotion à leur situation et agissant physiquement à cela. Les émotions visibles fournissent des designers avec un moyen direct pour affecter à l'utilisateur un état émotionnel propre à lui. Les acteurs sont donc équipés d'un modèle informatique simple de comportement émotionnel, qui est lié au comportement comme les expressions de visage qui peuvent être employées pour influencer leurs actions.

Une émotion est la réaction d'une personne à une perception. Celle-ci est amenée à répondre par une expression du visage, un geste, ou à choisir un comportement spécifique. Une émotion arrive entre une perception et la réaction suivante. Deux personnes différentes peuvent avoir des réactions différentes à la même perception, selon la façon dont ils sont affectés par cette perception.

Les émotions sont causées par la réaction aux objets, les actions d'agents et les événements. Les émotions causées par des événements peuvent être classées selon trois types d'émotions :

- Les émotions causées par des événements potentiels .
- Les événements affectant le destin d'autres .
- Les événements affectant le bien-être de l'acteur.

Chaque classe est caractérisée par des conditions d'apparition pour chacune de ses émotions et variables affectant son intensité. Les émotions auxquelles est soumis un acteur sont dues à sa perception.

- **Le comportement**

Le comportement est souvent défini comme la voie dans lequel des animaux et les humains agissent. Il est usuellement décrit en termes de langage naturel qui a la signification sociale, psychologique ou physiologique, mais qui n'est pas nécessairement facilement réductible au mouvement d'un ou deux muscles.

Le comportement est aussi la réponse d'un individu, groupe, ou espèce à son environnement. Le comportement ne réagit pas seulement à l'environnement, mais inclut aussi le flux d'information par

lequel l'environnement agit sur la créature vivante aussi bien sur la façon dont la créature code et emploie cette information.

Le comportement peut être décrit d'une façon hiérarchique. Le modèle comportemental décompose un comportement en comportements plus simple qui peuvent être décomposés plus loin. Chaque niveau de cette décomposition hiérarchique contient un ou plusieurs comportements exécutés séquentiellement, ou bien concurremment. Un niveau de la hiérarchie contenant plusieurs comportements pour être exécuté séquentiellement, appelé comportement.

Chaque comportement d'un ordre de comportement est appelé, cellule comportementale. La cellule comportementale contient les comportements qui sont exécutés chacun concurremment, ou exclusivement quand les règles d'inhibition sont spécifiées. Les comportements contenus dans une cellule comportementale sont de nature composée ou élémentaire.

Un comportement permet une décomposition récursive dans la hiérarchie. Un comportement élémentaire est placé au fond de la décomposition hiérarchique et encapsules un comportement spécialisé qui contrôle directement une ou plusieurs actions. Un comportement est exécuté récursivement : en fait la hiérarchie du comportement permet d'exécuter les comportements élémentaires des entités à chaque niveau de la structure de la hiérarchie.

Pour contrôler le comportement global d'un acteur, on exploite une pile de comportements. Au début de l'animation, l'utilisateur pousse un ordre de comportements dans la pile de l'acteur. À la fin du comportement actuel le système d'animation passe le comportement suivant de la pile et l'exécute. Ce processus est répété jusqu'à ce que la pile de comportement de l'acteur se vide.

Avec Ce contrôle de comportement employant une pile, un acteur devient plus autonome et crée ses propres sous buts en exécutant le scénario original.

- **L'action**

Basé sur l'information perceptuelle, le mécanisme comportemental d'un acteur détermine les actions à exécuter. Les actions peuvent avoir plusieurs degrés de complexité. Un acteur peut se développer dans son environnement ou bien agir réciproquement avec l'environnement ou encore communiquer avec d'autres acteurs.

Les actions sont exécutées en employant une architecture de mouvement commune. Le module d'action gère l'exécution des actions employées par un comportement en animant un modèle d'homme générique basé sur une hiérarchie de nœud. Il permet l'exécution simultanée ou séquentielle d'actions

en gérant des transitions lisses entre des actions finales et des actions [6]. Une boucle comportementale conduit l'animation, son rôle est de mettre à jour l'état du monde virtuel.

A chaque itération, le temps est incrémenté, le monde virtuel est mis à jour avec en particulier une mise à jour de l'état de chaque objet et acteur. Dans le cas d'un acteur, la perception est d'abord exécutée, après ses émotions sont produites avant que son comportement et ses actions ne soient exécutés.

- **La Mémoire**

La mémoire est d'habitude définie comme le pouvoir ou le processus de reproduction ou de rappel de ce qui a été appris et conservé, particulièrement par les mécanismes associatifs. La mémoire est aussi le dépôt pour des informations apprises et à conserver, générées à partir de l'activité d'un organisme ou d'une expérience.

### 5.2 Le comportement d'individu

Bien que la capacité à se mouvoir soit un comportement primordial de l'être humain, ce qui le caractérise vraiment est sa capacité à raisonner[19]. En effet, un individu est capable de symboliser mentalement les actions qu'il veut exécuter, pour ensuite les organiser. Il est ainsi capable d'élaborer un plan comportemental lui permettant d'ordonner ses actions, et de gérer leurs liens de dépendance. Ce raisonnement va ainsi avoir un impact direct sur son déplacement.

Les tâches mentales prépondérantes concernant l'action d'un individu sur son environnement sont sans doute les tâches d'interaction. Celles-ci définissent la manière dont l'individu va pouvoir affecter d'autres entités, et ainsi modifier l'état du monde qui l'entoure.

Ces tâches d'interaction entre elles même dans un processus mental complexe, pouvant être directement désirées par l'individu, ou symboliser une étape de son raisonnement .

Nous entendons par comportements d'individus ceux qui permettent à l'individu de se déplacer dans l'environnement. parmi ces les comportements les suivants :

- **Comportement de recherche et fuite**

La recherche et la fuite sont deux comportements très simples qui déplacent un individu vers ou loin d'une position de cible avec une vitesse constante.

La différence entre la recherche et la fuite est que la recherche est l'acte d'orienter le caractère vers une position indiquée dans l'espace global. Ce comportement ajuste le caractère de sorte que sa vitesse soit radicalement alignée vers la cible. D'une manière simple, la fuite peut être considérée comme étant

l'inverse de la recherche. Elle agit en orientant le caractère de sorte que sa vitesse soit radicalement alignée loin de la cible. La vitesse désirée se dirige dans la direction opposée .

- **Comportement d'arrivée**

Le comportement d'arrivée est une extension du comportement de recherche. De même que pour le comportement de recherche, il est utilisé pour orienter l'individu vers une cible indiquée[6]. La différence importante se résume dans la manière selon laquelle l'individu atteint la destination.

Le comportement de recherche fait que notre individu atteint la cible à pleine vitesse. Il se déplace en fait davantage dans la direction actuelle et va ainsi générer plutôt un mouvement de danse semblable au comportement d'une mite autour d'une source lumineuse.

Cependant, le comportement d'arrivée doit être tel qu'il y'ait un ralentissement commandé conformément au cahier des charges édicté par l'utilisateur, le mouvement doit s'arrêter à la position désirée( figure1.11). Ceci fait provoque le ralentissement selon sa distance actuelle à la destination et le résultat est un objet qui s'arrête au niveau de la cible indiquée[6].

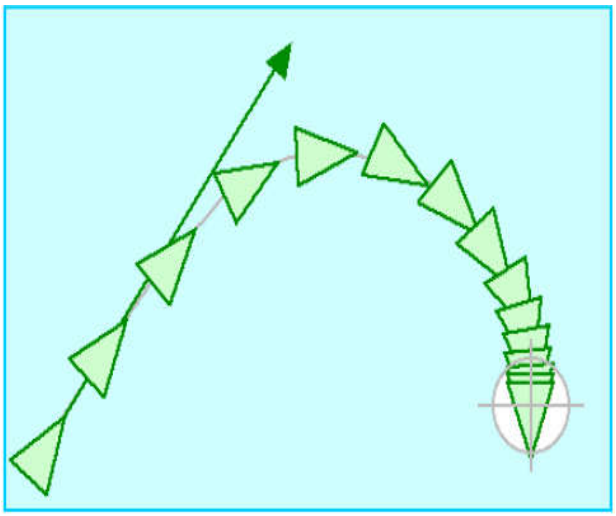


**Figure 1.11** Le Comportement D'arrivée.

- **Comportement d'évitement d'obstacles**

Le comportement d'évitement d'obstacles donne à un acteur la capacité de manœuvrer dans un environnement encombré en esquivant autour des obstacles[20][6]. Il y a une distinction importante entre l'évitement d'obstacles et le comportement de fuite.

La fuite a pour conséquence d'orienter l'acteur loin d'un emplacement donné, tandis que l'action d'évitement d'obstacles agit seulement quand un obstacle se trouve directement devant cet acteur. Par

exemple, si un individu se déplace sur une trajectoire parallèle à un mur, l'évitement d'obstacles ne prendrait aucune mesure corrective de direction, mais la fuite essaierait de s'éloigner du mur, en suivant la perpendiculaire à ce mur.

Le but du comportement est de garder un cylindre imaginaire de l'espace libre devant l'acteur. Le cylindre se trouve le long de l'axe vers l'avant l'acteur, à un diamètre égal à la sphère de bondissement de l'acteur, et s'étend du centre de l'acteur pour une distance basée sur la vitesse et l'agilité de l'acteur.

Le comportement d'évitement d'obstacles considère chaque obstacle alternativement et détermine si ces obstacles possèdent des intersections avec le cylindre. En localisant le centre de chaque obstacle sphérique, le test d'intersection avec le cylindre est ainsi effectué très rapidement [6]

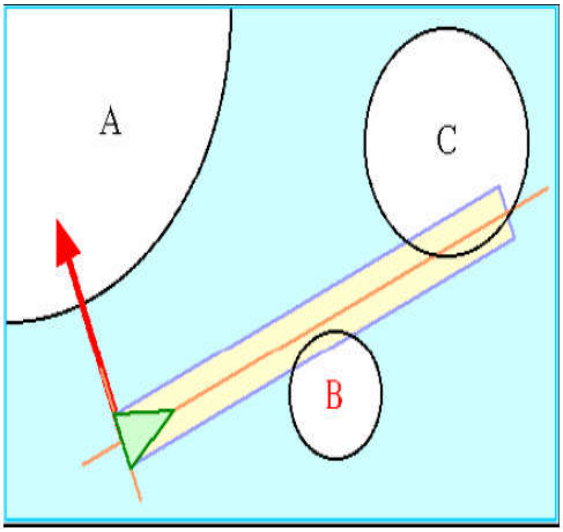



**Figure 1.12** L'évitement D'obstacle[6].

- **Comportement de suivi de chemin**

Ce type de comportement permet à un acteur de s'orienter le long d'une voie d'accès prédéterminée, telle qu'une chaussée, un couloir ou un tunnel. C'est en fait très distinct de l'action de contraindre un individu à suivre d'une manière rigide une voie d'accès tel que le roulement d'un train le long d'un rail, par exemple. Le comportement de suivi d'une voie d'accès est plutôt destiné à la production de mouvements tels que des personnes empruntant un couloir : les différentes voies d'accès restent proches et sont souvent parallèles à l'axe du couloir, elles sont néanmoins libres[20] .

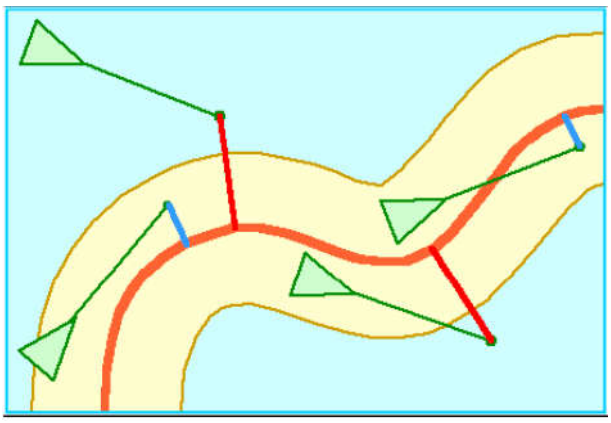


**Figure 1.13** Le Comportement De Suivre De Chemin[20].

Autrement dit, l'acteur vire loin de la voie d'accès, ou en est trop loin de celle-ci. Pour l'orienter en arrière vers la voie d'accès, le comportement de recherche est exploité pour une orientation en arrière vers la projection de la voie d'accès de la future position prévue, de la même manière que pour l'action d'évitement [5].

- **Evitement de collision non-alignée**

Le comportement d'évitement de collision non alignée se doit de prévenir les collisions entre des acteurs se déplaçant dans une direction arbitraire. Considérez votre propre expérience de marche à pied à travers une place ou une entrée

pleine d'autres marchants : L'évitement de collisions implique la prévision de collisions potentielles et le changement de votre direction et de votre vitesse pour les prévenir

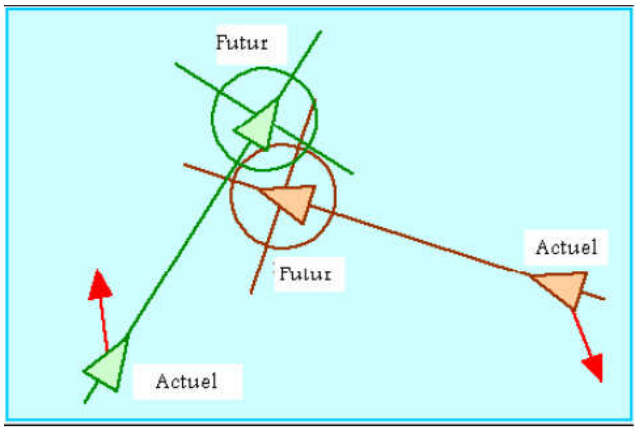



**FIGURE** 1.14 Evitement de collision non-alignée [6]

## 6. La planification de chemin

La planification de chemin dépend en grande partie de l'environnement dans lequel l'agent va évoluer. La formulation des problèmes de planification de chemin va donc généralement reposer sur la représentation qui est faite de cet environnement, aussi appelée "espace de travail" ou workspace. Les différentes méthodes de planification proposées explorent une représentation duale de l'espace de travail afin d'identifier le chemin désiré.

### 6.1 algorithme de planification de chemin

algorithmes suivants peuvent être adaptés, modifiés pour d'autres types de structures comme des triangles, des hexagones également. Ce sont des algorithmes abstraits.

- **L'algorithme de Dijkstra (Dijkstra, 1959)**

parcourt toutes les cases possibles en tenant compte de leurs coûts respectifs. Lorsque tous les chemins sont visités, l'algorithme compare les coûts. Il trouvera ainsi tous les chemins minimums possibles ou le seul chemin minimum. C'est l'un des algorithmes les plus connus dû à son efficacité pour trouver les chemins les plus courts voir figure(1.15)[21]. Il a été conçu par Edsger Dijkstra en 1959.

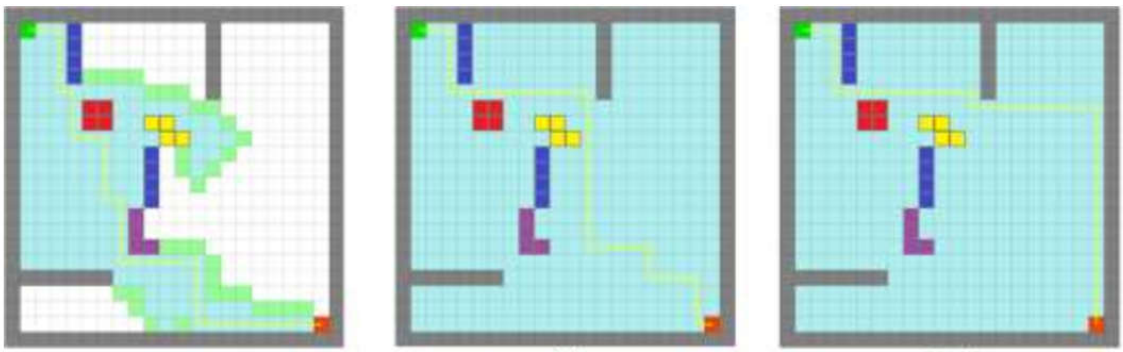


**Figure 1.15** Dijkstra L'algorithme dans mazerunner game

- **Breadth-First Search (BFS)**

Il s'agit d'un algorithme de parcours en largeur. Il parcourt tous les chemins d'une seule case à chaque itération. Il s'arrête lorsqu' il a trouvé un chemin entre le départ et l'arrivée voir exemple(figur1.16)[22].

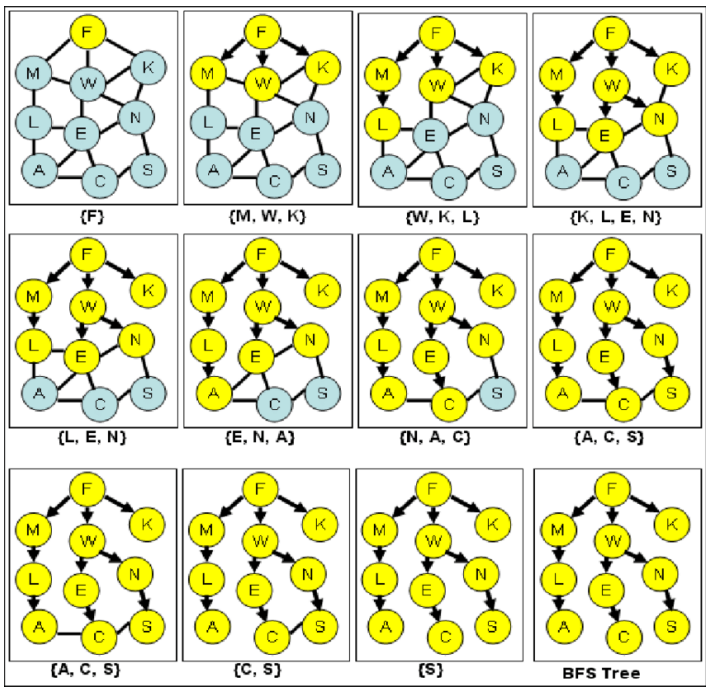



**Figure 1.16** : exemple pseudo code BFS [22]

➢ **Depth-First Search (DFS)**

Cet algorithme est traduit par « parcours en profondeur » en français. Il va au bout de chaque chemin. Il les parcourt un à un au fur et à mesure. Ce n'est pas un algorithme très efficace lorsqu'il y a beaucoup de chemins et que ceux-ci sont longs (figure 1.17) . Il s'arrête lorsqu'il a trouvé un chemin (Cormen, 2001)[22].

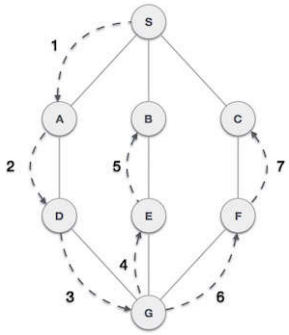



**FIGURE 1.17** data structure Depth-First Search[22]

➢ **Iterative-Deepening Depth-First Search (IDDFS)**

Il s'agit également d'un parcours en profondeur à la différence près que le nombre de cases explorées dans chaque chemin va augmenter à chaque itération (Demyen, 2007). Il va explorer le chemin 1 jusqu' à une profondeur seuil, puis le chemin 2 jusqu'à la même profondeur et ainsi de suite pour tous les chemins. À la prochaine itération, la profondeur seuil augmente et l' algorithme parcourt chaque chemin de la même manière. Il s' agit d' un mélange entre parcours en largeur et parcours en profondeur. L' algorithme s' arrête lorsqu' il a trouvé un chemin.

➢ **Best-First-Search Best First Search (Dechter et Pearl, 1985)**

est un type d'algorithme de planification de chemin, qui calcule d' abord le chemin le plus prometteur avant tous les autres. Pour 1 estimer quel est le chemin le plus intéressant, ce type d'algorithme utilise une heuristique, c'est-à-dire une estimation de la distance jusqu'à l'arrivée. A* et B* sont donc des algorithmes de type Best First Search.

➢ **B * (B Star) B Star**

est un algorithme de type Best First Search (Berliner, 1979). Il utilise un arbre pour répertorier tous les chemins possibles. Ensuite il va se concentrer sur la branche avec le coût le moins important. Si celle-ci ne parvient pas à l'arrivée, il prendra une autre branche avec un coût peu important.

## 7. Conclusion

Dans la première partie de ce chapitre Nous avons étudié la représentation de l'environnement virtuel. Nous avons ainsi exposé les techniques permettant la représentation de l'environnement. Ensuite, nous avons présenté les environnements informés. Enfin nous avons exposé l'intégration des objets interactifs. Notamment concernant les représentations de l'environnement, on a pu voir que les approches approximatives offrent un accès rapide à la description topologique, alors que les approches exactes permettent la conservation maximale de l'information géométrique.

# Chapitre 2

## 1. l'Animation comportemental

L'animation comportementale s'attache à décrire la raison d'un mouvement plutôt que le mouvement lui-même. L'idée est de doter les entités de comportements et d'une certaine autonomie[13].

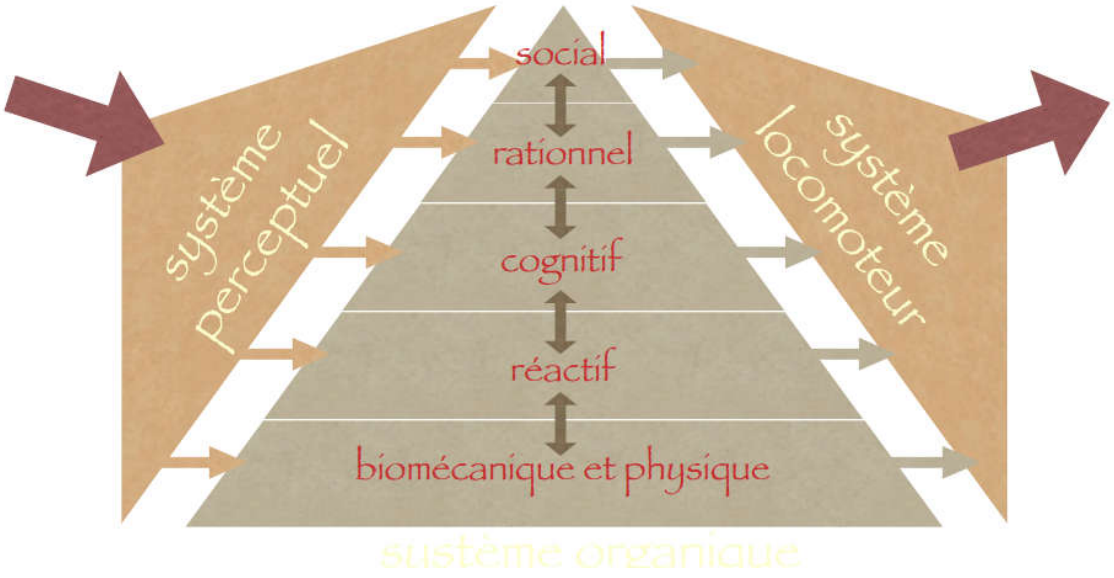



figure (2.1) Niveaux de comportements[13]

### 1.1 La boucle de l'animation comportementale

L'animation comportementale prend comme base la boucle **Perception-réaction-Action** :

une entité perçoit l'environnement, décide de la prochaine action à exécuter et agit sur le monde[14].

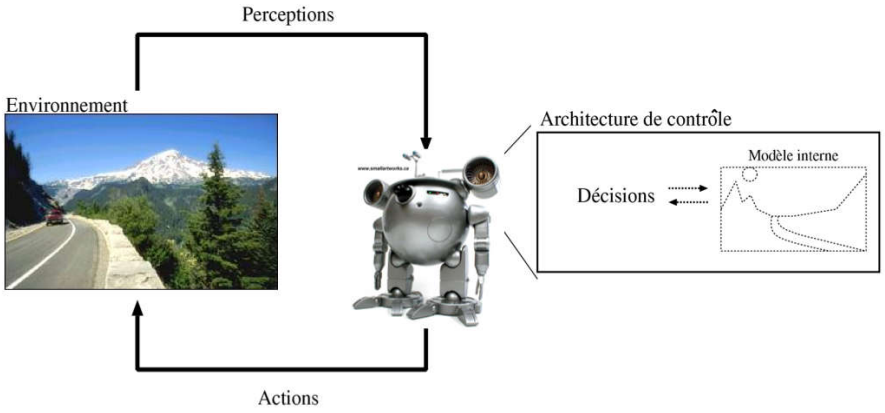



**Figure 2.2**La boucle de l'animation comportementale[14].

- **Module de perception**

La fonction de perception permet à l'individu virtuel d'une part, de percevoir l'état de son environnement (les autres individus virtuels, les obstacles), et d'autre part, de percevoir ses états internes (les actions internes). Le module de perception de l'individu virtuel extrait les informations de l'environnement, pour traiter et utiliser par les autres modules (voir la figure 2.3)[12] .

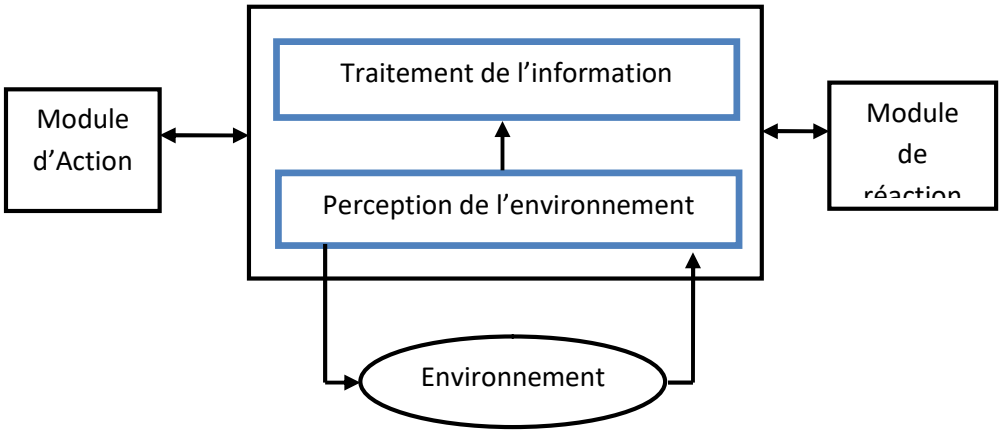



**FIGURE 2.3:** STRUCTURE DE MODULE PERCEPTION[12][14].

- **Module de réaction**

Après la phase de perception, dans laquelle nous avons vu les différents types de traitement de détections d'obstacles et de collisions, nous passons au module de réaction qui se manifeste essentiellement dans les techniques d'évitement d'obstacle et de collisions. Donc le module de réaction est un module superviseur qui permet de contrôler l'action globale de l'individuvirtuel. Il permet au l'individu virtuel de réagir aux différents événements internes et externes en déclenchant un comportement correspondant à l'événement capté selon le mode stimulus/réponse comme illustré dans la figure suivante :


**FIGURE 2.4:** LE MODE STIMULUS/REPONSES DU MODULE DE REACTION[13].

- **Module d'action**

L'action est la dernière phase dans le cycle d'animation comportementale, l'action est présentée par le déplacement des individus dans la scène Alors on peut dire que le module d'action est

une collection de comportements prédéfinis, tel que déplacement, arrêt, évitement d'obstacle. Ces comportements sont nécessaires pour la navigation de l'individu virtuel et qui sont :

- **Sélection chemin** : il permet de trouver un chemin, plus ou moins optimal pour une animation réaliste.
- **Éviter obstacle** : il permet d'éviter les obstacles qui se trouvent dans le chemin de piéton virtuel.
- **Éviter collision** : il permet d'éviter les collisions avec les autres piétons virtuels.

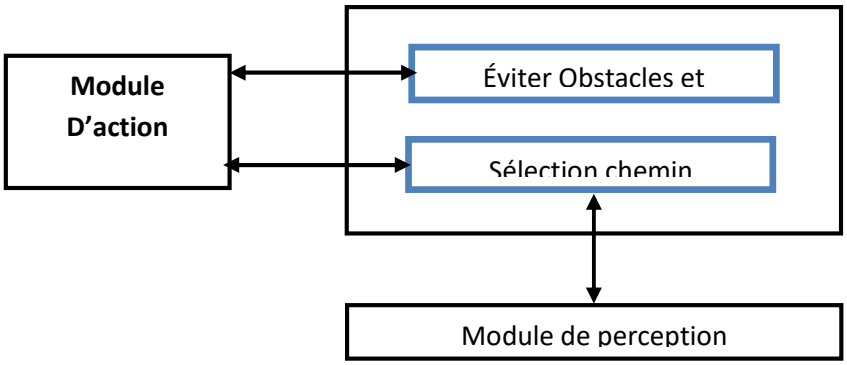



**Figure 2.5** : Module d'action et ses Interactions[13]

## 1.2 Caractéristiques du déplacement des piétons

### ➢ Comportement des piétons

À l'échelle microscopique On peut distinguer plusieurs caractéristiques du comportement individuel des piétons :

un piéton se déplace à une vitesse dite ≪ de confort ≫ qui dépend de ses caractéristiques personnelles (son âge, son genre, sa taille, son état de santé ...), de ses caractéristiques de déplacement (une visite touristique, la longueur du chemin,...) et les caractéristiques de l'environnement (les conditions météorologiques par exemples).

### ➢ Comportement des piétons à l'échelle macroscopique

Dans de nombreuses situations de la vie quotidienne, une foule de piétons en déplacement présente des capacités d'organisation, dans cette section nous aborderons la notion du phénomène d'auto-organisation (l'émergence spontanée d'une structure globale dans un système, à partir d'interactions locales entre les agents qui composent ce système). Ces phénomènes sont :

– Si un blocage s'installe et une arche se forme autour d'un espace étroit tel qu'une porte quand une foule dense veut la traverser (figure 2.6), ce phénomène est dite le phénomène d'arche .

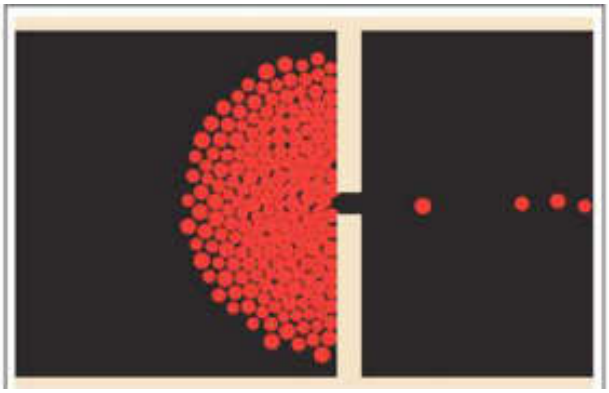


**Figure 2.6**– Le phénomène d'arche.

– Une autre forme d'organisation apparait lorsque deux flux de piétons se déplacent en sens inverses, il se crée naturellement des lignes de piétons qui se suivent (figure2.7), ce qui permet de réduire les collisions entre piétons et d'augmenter leur vitesse de déplacement, ce phénomène prend le nom la formation de files

**Figure 2.7** – Formation de files.

## 2. Réalisme de simulation

L'animation comportementale s'attache à décrire le principe d'un comportement plutôt que le comportement lui-même, c'est-à-dire que l'objectif n'est pas d'obtenir une visualisation réaliste mais un comportement qui va permettre de comprendre les résultats de la simulation Le choix est basé sur des

actions effectuées par les agents en fonction de l'environnement, Ce qui nous mène à décrire différents de modèles :

## 2.1. Modèles macroscopiques

Les modèles macroscopiques sont les premiers modèles de simulation de foule à être apparus. Ceci est notamment dû à leur faible consommation en terme de ressources de calcul. Ainsi, ils simulent un grand nombre de piétons sans s'intéresser à leur comportement individuel figure(2.8) . Ces modèles sont généralement dédiés aux simulations d'évacuation de bâtiments et à la mise en place des conditions de sécurité qui sont liées à l'évacuation .Ils ont aussi pour but de simuler de manière réaliste les phénomènes macroscopiques sans que l'individu ne soit représenté dans le modèle.

**Figure 2.8** example de foule humaine de haute densité

**-Modèles statistiques**

Les modèles statistiques s'intéressent directement aux débits de piétons. Certains sont basés sur la dynamique des fluides. Henderson propose un modèle gazeux permettant de simuler des foules de piétons de faible densité et assimile les états de déplacement (arrêt, marche, course) aux différents niveaux d'énergie des gaz. propose quant à lui un modèle hydraulique pour simuler des foules de forte densité.

Plusieurs modèles s'intéressent aux débits de piétons quant à l'évacuation de bâtiments , et notamment le débit de piétons que peuvent permettre les portes et les escaliers. Ceux-ci proposent des formules calculant le temps d'évacuation en fonction de divers paramètres comme le nombre de piétons, la largeur des zones de circulation, la pente ou encore l'espace occupé par une personne.

Plus récemment, Colombo et Rosini proposent un modèle qui s'inspire de modèles de trafic routier . Il s'appuie sur deux suppositions : le nombre total de piétons reste constant au cours de la simulation et la loi de vitesse est fonction de la densité. Leur modèle introduit la notion de débit de piétons à très haute

densité, dans des situations exceptionnelles de type panique où le débit serait quasi-nul en situation normale .De par leur nature, ces modèles macroscopiques sont capables de simuler des foules de piétons avec un très faible coût calculatoire et sont capables d'intégrer facilement des observations réelles.

Ils ont en revanche besoin d'informations issues d'observations réelles pour fonctionner correctement et sont peu adaptables à de nouvelles situations .De plus, ils ne simulent pas les piétons à l'échelle individuelle.

**-Modèles de flots**

Les modèles de flots sont inspirés de la dynamique des fluides. Des champs de potentiel agissent sur les piétons qui sont considérés comme des particules soumises à des champs Dans le cadre de la simulation de fluides, [11] représente des champs de vitesse à travers un carrelage de flots. Chaque carreau contient son propre champ et leur assemblage produit des flots plus grands. Les arêtes et les coins des carreaux sont élaborés de manière à assurer une continuité entre les carreaux. Non seulement des fluides peuvent être dirigés ainsi mais cette méthode peut également être appliquée aux foules. Les carreaux sont stationnaires (leur champ n'est pas modifié au cours du temps)mais peuvent générer des flots dynamiques en combinant le carrelage en fonction du temps.

Inspiré des travaux de Hughes et , Treuille et al. ont simulé des foules de piétons à partir d'un modèle qui s'appuie sur les quatre hypothèses suivantes :

– chaque personne tente d'atteindre un objectif géographique.

– les personnes cherchent à marcher à la vitesse la plus élevée possible.

– il existe des endroits dits inconfortables que les piétons vont chercher à éviter

le piéton cherchera à emprunter le chemin le moins "coûteux", en fonction de sa longueur, de son temps de parcours et de son niveau d'inconfort.

Un champ de potentiel dynamique intègre simultanément la navigation globale avec des obstacles mobiles tels que d'autres piétons, permettant de simuler de grandes foules en temps réel, les collisions étant résolues de manière implicite. Narain et al [10]. proposent quant à eux une approche hybride basée sur le principe des flots continus où les interactions locales sont résolues à partir d'une approche géométrique. Maury et Venel [9] proposent un modèle déterministe d'évacuation de bâtiments dans lequel chaque personne est assimilée à un disque. Son approche repose sur deux principes :

– chaque personne à une vitesse souhaitée, vitesse qu'elle aurait en l'absence des autres. L'espace est considéré comme un flot de champs de vitesse qui détermine le plus court chemin entre chaque point et la sortie.

– une contrainte d'encombrement empêche les personnes (représentées par leur dis que) de se chevaucher ou de chevaucher des obstacles.

Les modèles de flots permettent de simuler des foules de manière efficace en terme de coût calculatoire et de reproduire certains phénomènes macroscopiques. Les piétons étant considérés comme des particules, ils font souvent preuve d'un manque d'individualité et le flot prend souvent le pas sur leurs objectifs propres. Ces modèles sont intéressants lorsque tous les piétons ont un objectif commun. A l'échelle locale, on peut aussi observer quelques artéfacts et des mouvements impossibles d'un point de vue biomécanique, affectant le réalisme à l'échelle microscopique [9][10].

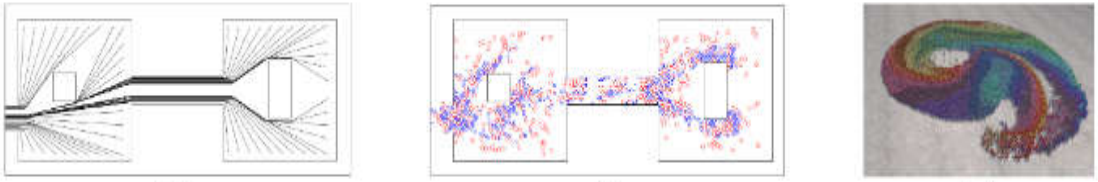



**Figure 2.9** –Modèles de flots. (a). Représentation des flot des champs de vitesse dans [9] (b). Exemple d'évacuation [9] (c). Exemple de simulation dans [10] : 10000 agents placés en cercle et devant aller à l'endroit diamétralement opposé.

### 2.2 Modèles microscopiques

L'approche microscopique consiste à identifier les interactions entre chaque individuet son voisinage pour ensuite influencer le comportement de l'individu. Ces modèles sedifférencient par leur manière de prendre en compte les informations provenant duvoisinage d'un piéton et par la réaction de l'individu compte tenu de ces informations.

#### -Automates cellulaires

Certains modèles, appelés automates cellulaires, sont basés sur une discrétisation de l'espace en cellules, souvent carrées mais parfois hexagonales ou octogonales. Leurprincipe repose sur le fait que chaque piéton occupe une cellule et ne peut se déplacervers une autre cellule que si celle-ci est libre.

Un automate cellulaire consiste en une grille régulière de « cellules » contenant chacune un « état » choisi parmi un ensemble fini et qui peut évoluer au cours du temps. L'état d'une cellule au temps t+1 est fonction de l'état au temps t d'un nombre fini d'état de cellules appelé son (voisinage). À chaque nouvelle unité de temps, un ensemble de règles locales sont appliquées simultaément à toutes les cellules de la grille, produisant une nouvelle (génération) de cellules dépendant entièrement de la génération précédente. Ces règles décrivent le comportement (intelligent) de prise de décision des automates, de ce fait créant et émulant le comportement réel. Chaque automate évalue ses occasions au cas par cas. Le comportement émergent global de groupe est un résultat des interactions des règles locales comme chaque piéton examine les cellules disponibles dans son voisinage.

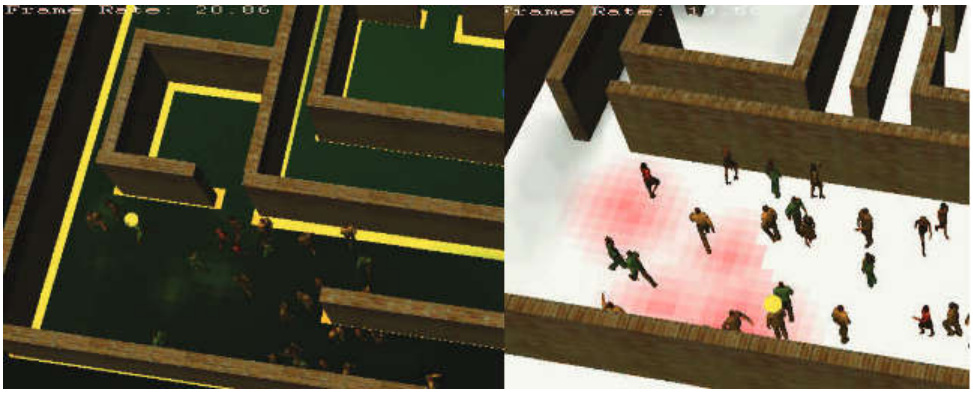


**Figure 2.10**: Simulation de foule : AC (à gauche) et (à droite) structure 2D à la base de grille (Tecchia et al 2001)

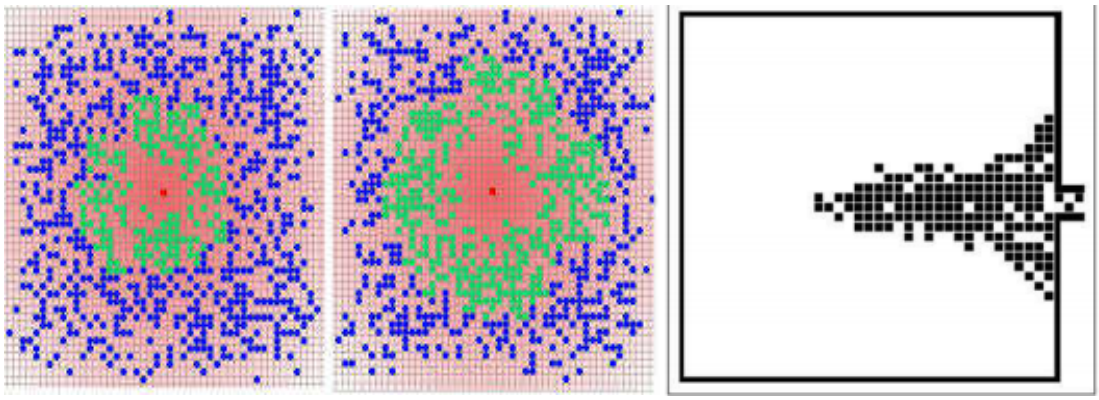


**Figure 2.11** – Simulation d'´evacuation de salle avec le modèle automate cellulaire .

Un modèle proposé par Burstedde et al [8]. Qui est présenté dans , où l'environnement est une grille 2D. A chaque pas de temps, une matrice 3*3 (représentant la cellule courante et chaque cellule adjacente, par le coin ou l'arête) qui permet de déterminer la prochaine cellule sur laquelle l'agent va se déplacer selon une direction de préférence (figure 2.12)[8].

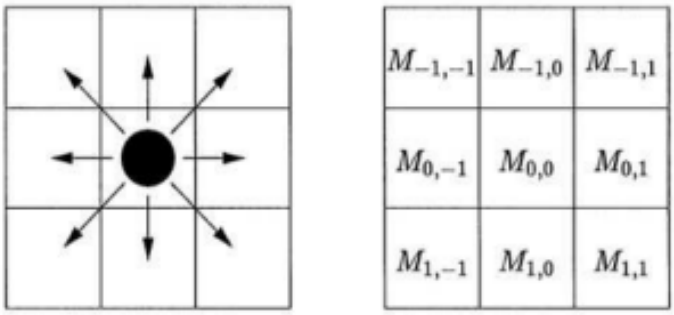



**Figure 2.12** – Matrice 3*3 de la prochaine cellule d'un agent[8] .

**-Modèles à base de règles**

Les modèles à base de règles ≪rule-based≫ ont été introduits par Reynolds , il est représenté par des nuées d'oiseaux pour modéliser la foule (figure 2.13). Les membres d'un groupe sont considérés comme des agents individuels ou chacun de ces agents est soumis aux règles suivantes :

- la séparation : les agents ne collisionnent pas.
- l'alignement : ils essayent de garder une direction et une vitesse de mouvement commune.
- ils essayent de rester unis

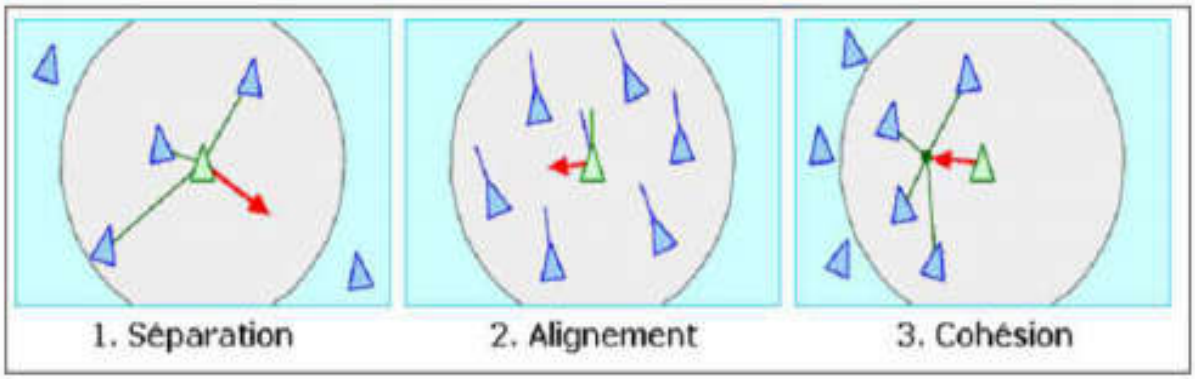



**Figure 2.13**– Les trois règles caractérisant le comportement des oiseaux[6] .

**- Les modèles à base de forces**

Développé principalement par Helbing qui a réalisé une analogie avec la physique newtonienne. Ce modèle permet de gérer le mouvement de chaque piéton, représenté par un disque, est soumis à des forces d'attraction et de répulsion qui agissent sur son accélération suivant la deuxième loi de Newton ( $\sum F = m.a$ )

F = force , m= masse , a = accélération.

La force d'attraction (la force motrice) permet au piéton de se déplacer vers sa destination souhaitée. La force de répulsion (la force d'interaction) résolvent les problèmes de collision et gère les contacts piéton-piéton et piéton-obstacle (**figure 2.14).**

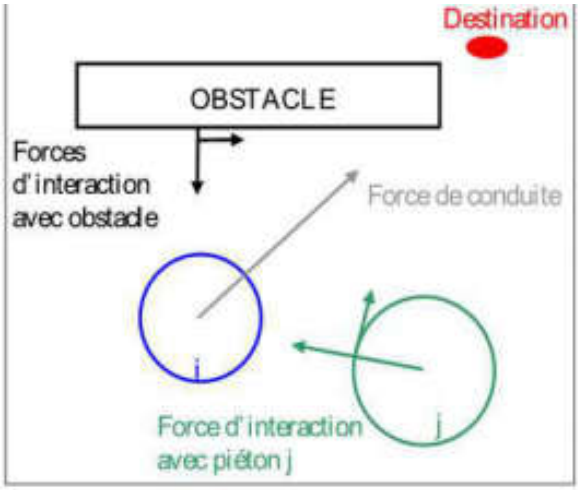



**Figure 2.14** – Modèle à base de force.

## 3. Conclusion

Différents modèles ont été présentés dans ce chapitre à travers leur type d'approche. Nous remarquons que la plupart des modèles présentés restreignent la simulation de foule à la résolution d'un problème d'évitement de collision. Peu d'entre eux s'intéressent au comportement de suivi. Parmi ceux-ci, rares sont ceux proposant un haut niveau de réalisme et aucun d'entre eux n'a été évalué à partir d'observations réelles Dans le chapitre 3, nous proposons de réaliser Création d'un environnement de déplacement par la méthode de triangulation de Delaunay, avec une configuration qui permette à une entité virtuelle de naviguer dans cet environnement , utilisant la l'Algorithme A ( planification de chemin ) et la méthode à base de règles pour la représentation et de la simulation de la foule humaine en temps réel

# Chapitre 3

## 1. Introduction

Dans n'importe quel système, la conception est l'étape la plus importante, car une bonne démarche du système nécessite une bonne conception. Le rôle de cette phase est de décrire une architecture interne du système, par la présentation des modules qui constituent ce dernier. Dans ce chapitre nous allons proposer une étude détaillée concernant toutes les étapes et les éléments nécessaires dans notre système.

## *2. Objectifs*

Les principaux objectifs auxquels doit répondre notre application sont :

- Création d'un environnement de déplacement par la méthode de triangulation de Delaunay, avec une configuration (zone navigable, zone non navigable, point de départ, point destination), et permettre à une entité virtuelle de naviguer dans cet environnement, cette entité doit suivre un chemin planifié.
- Planification d'un chemin en utilisant l'algorithme A*.
- Navigation des entités virtuelles dans un environnement virtuel basant sur la lois a base de règles ( séparation , alignement , cohésion ) ainsi la détection de collision entre eux.
- Le rendu en utilisant Ogre 3D.

## 3. Conception global

Conception global est La phase de description de l'architecture du système à réaliser, dans la quel on va motionner les modules de notre système qui sont des étapes complémentaire pour avoir le résultat final.

1. la représentation de l'environnement via la triangulation de Delaunay
2. la planification de chemin utilisant A*.
3. la navigation des entités virtuel en faisons les comportements individuels et des groupe

-le schéma si dessous l'architecture de notre système est illustré.

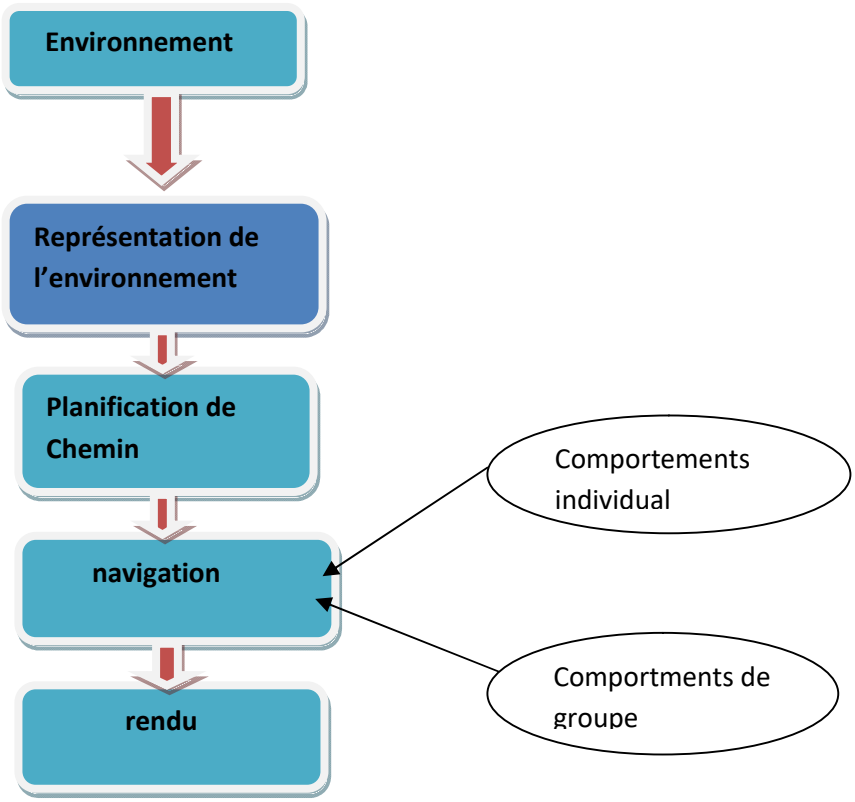



**Figure 3.1**: Schéma de la conception globale du système

## 4. La conception détaillée

La conception détaillé sert à montrer l'architecture détaillé du système en discutant les modules qui composent le système, notre travail est composé de quatre modules ,le premier module s'agit sur la représentation de l'environnement par Triangulation de Delaunay ,le deuxième est l'Algorithme de A* ( algorithme de recherche ) , le troisième est le module de l'humain virtuel (agent) qui est caractérisé par son état physique et le quatrième qui est la navigation à base de règles , pour cela , dans cette partie on va parler et discuter chaque module de système en détail .

## 5. Représentation de l'environnement

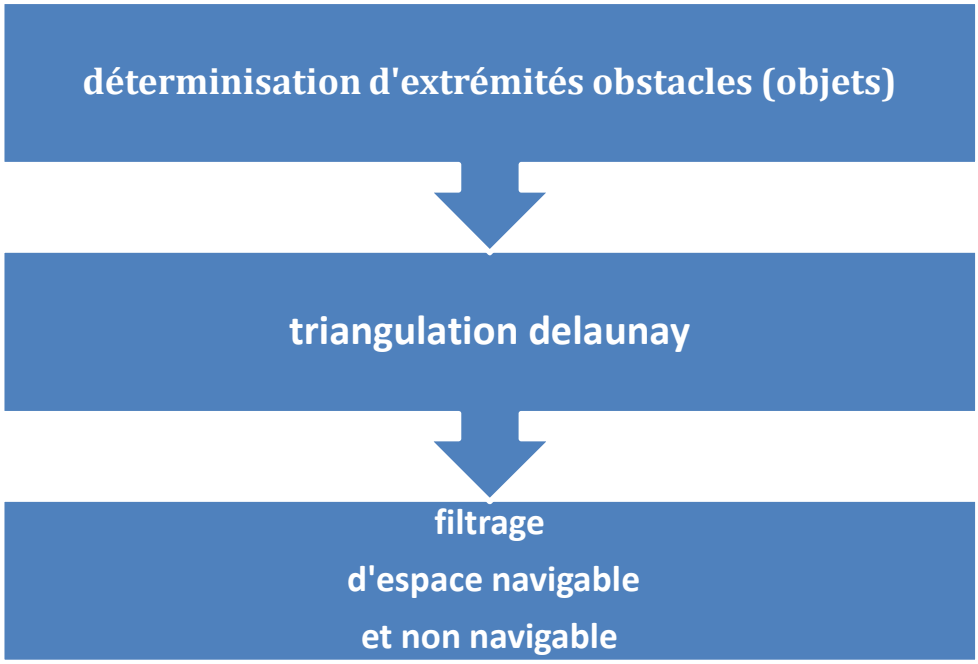



**Figure3.2** : schéma illustratif pour les étape effectuer pour la R.E utilisant T.

### 5.1. Représentation de l'environnement par Triangulation de Delaunay

Il existe un très grand nombre de méthodes théoriques et pratiques pour générer une triangulation à partir d'un ensemble de points. Mais toutes ne sont pas exploitables. En effet, on cherche à construire un ensemble de triangles qui soient arrangés de manière assez uniforme et régulière .Par exemple, les triangles générés doivent être les moins allongés possibles et former un ensemble homogène. Dans le cas contraire, les calculs effectués par la méthode des éléments finis seraient moins représentatifs et les résultats moins proches de la réalité.

Cela implique une construction assez naturelle du maillage, et « arrange » les triangles pour uniformiser leurs dimensions. Les seuls triangles plats qui peuvent advenir se situent aux frontières de l'ensemble de points considéré.

- **Propriétés**

Cette triangulation possède des propriétés intéressantes pour construire un algorithme de génération de maillage . On se restreint ici aux propriétés de la triangulation dans un espace à deux dimensions.

Soit n le nombre de points :

- La triangulation est unique s'il n'y a pas trois points alignés ou quatre points cocycliques.
- La triangulation contient au plus n simplexes.
- Chaque sommet est connecté en moyenne à 6 triangles.
- L'union des simplexes de la triangulation forme l'enveloppe convexe de l'ensemble des points

**-Algorithmes**

Il existe deux types d'algorithmes pour générer une triangulation : les algorithmes itératifs d'une part, et récursifs d'autre part. Dans les deux cas, on part d'un ensemble de points du plan et le résultat obtenu est un ensemble de triangles (la triangulation).

L'approche itérative tend à partir d'un ensemble constitué d'un unique triangle et à ajouter à cet ensemble les nouveaux triangles obtenus pour finir avec l'ensemble désiré. L'approche récursive adopte une stratégie diviser pour régner. Une division par dichotomie de l'ensemble de points est effectuée avant fusion des ensembles.

- **Méthode incrémentale**

L'ensemble de points étant fini dans le plan, il est inclus dans un intervalle de $|R^2$. On peut ainsi définir deux triangles initiaux formant cet intervalle et englobant l'ensemble des points (Illustration 2-1)[23]. Puis, on choisit un point dans l'ensemble initial (Illustration 2-2). On peut localiser le triangle dans lequel il se trouve et créer les nouveaux triangles formés entre le point et les sommets de ce triangle (Illustration 2- 3 et 4). On continue de la sorte jusqu'à ne plus avoir de point à choisir. On obtient la triangulation de Delaunay en sortie.

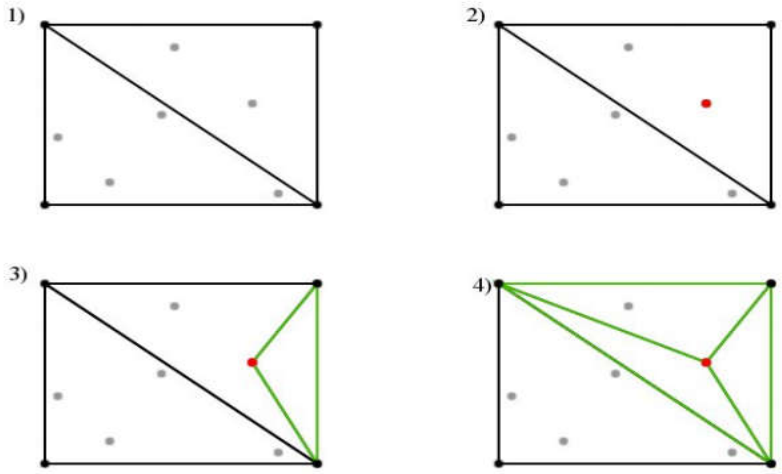



**Figure3.3**: Première itération de l'algorithme incrémental[23]

La complexité théorique de ce genre d'algorithme est quadratique.

**Algorithme**

L'algorithme général se penche fortement sur une propriété importante des triangulations de Delaunay, qui est: Outre les sommets, il n'y a pas d'autres points sur ou à l'intérieur du cercle circonscrit d'un triangle quelconque. En d'autres termes, tous les cercles circonscrits sont vides.
Sachant cela, et pensant dur, vous pouvez imaginer la façon suivante pour ajouter un sommet à une triangulation déjà existante (en pseudo-code).

```
1.    add_vertex(vertex)
2.    {
3.    for (each triangle)
4.    {
5.    if (vertex is inside triangle's circumscribed circle)
6.    {
7.    store triangle's edges inedgebuffer
8.    remove triangle
9.    }
10.   }
11.   remove all double edges fromedgebuffer,
12.   keeping only unique ones
13.   for (each edge inedgebuffer)
14.   {
```

```
15.     form a new triangle between edge and vertex
16.     }
17.     }
```

L'algorithme complet revient donc à la suivante en pseudo-code.

```
1.      triangulate()
2.      {
3.        create supertriangle and add it to the triangulation
4.        for (each vertex)
5.        {
6.      add_vertex(vertex)
7.        }
8.        for (each triangle)
9.        {
10.         if (one or more vertices stem fromsupertriangle)
11.         {
12.           remove triangle
13.         }
14.       }
15.     }
```

## 6. La recherche et la planification du chemin

Les personnages doivent interagir dans des environnements complexes, dynamiques et possédant une géographie étendue, par exemple ils doivent parcourir différents types de dispositions spatiales (chambres, labyrinthes, plateformes, etc ) d'une façon intelligente et réaliste ; c'est `a dire, en prenant des raccourcis si ils existent ou en évitant des obstacles . Un personnage capable de traverser des objets ou qui arrête son déplacement car il a trouvé un obstacle, qui est franchissable depuis la perspective de l'utilisateur, va créer une brèche dans l'illusion de comportement intelligent et adaptatif que nous essayons de produire. Il existe diverses solutions `a ce problème, certaines sont proposées par l'intelligence artificielle et les autres correspondent aux méthodes empiriques et approximatives utilisées par l'industrie des jeux vidéo. Ces dernières se sont montrées plus efficace en temps de calculs dans des environnements moins complexes et avec moins d'obstacles . Dans le cas des

environnement très larges et peuplés par des obstacles, des personnages non joueurs et des joueurs, la méthode la plus utilisé est l'algorithme A* (Figure 3.3).

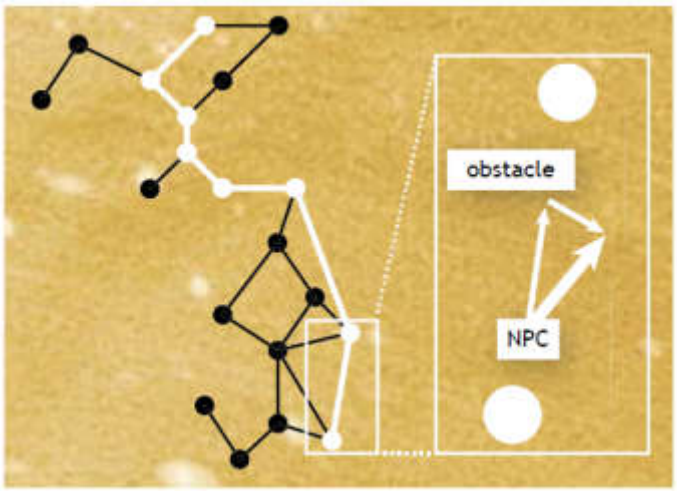



**Figure 3.4** – Exemple algorithme A* et pilotage du mouvement

Cet algorithme cherche le chemin optimal, dans le cas de jeux vidéo c'est le chemin le plus court, entre un nœud initial et un nœud destin prédéfini. Parmi les titres de jeux les plus connus, des jeux comme Half Life, Descent 3, Quake III l'utilisent pour le déplacement des personnages non joueurs (bots ) .

### 6.1 Le pilotage et contrôle du mouvement

Pour suivre le chemin trouvé par l'algorithme A*, ainsi que pour donner aux personnages la capacité de parcourir leur environnement d'une manière très proche à celle perçut par l'esprit humain, nous avons besoin de ce qu'on appelle le pilotage et contrôle du mouvement. La technique la plus utilisé a été proposée par C. Reynold [6][7] et consiste `a calculer un vecteur avec une direction ainsi qu'une vitesse et une force pour chaque type de mouvement que nous souhaitons inclure dans notre personnage (éviter des obstacles, suivre un leader, suivre un groupe, échapper d'un ennemi, etc.). Les calculs faits pour chaque comportement seront finalement additionnés pour produire un seul vecteur final qui sera appliqué au mouvement et pilotage du personnage.

La figure suivante illustre la tâche de la planification de chemin par l'algorithme A*:

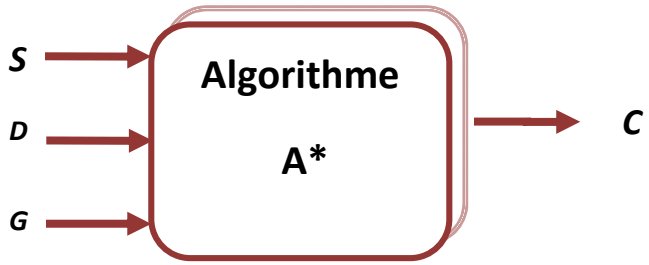



***Figure3.5 :*** *la planification de chemin par A**

Tel que :

**S** (Source) : le point de départ.

**D** (Destination) : le point d'arrivé.

**G** (Graphe) : le graphe à parcourir.

**C** (Chemin) : le chemin optimal reliant le nœud de départ et le nœud destination.

A* permet la recherche d'un chemin optimal entre un nœud de départ et un nœud d'arrivé, il s'agit d'une extension de l'algorithme de Dijkstra. Il utilise une évaluation heuristique qui estime la distance entre un nœud quelconque du graphe et le nœud cible et visite ensuite les nœuds par ordre de cette évaluation heuristique.

Cette fonction peut être calculée comme suit :

Soit un nœud N de coordonnés (X, Y), et un nœud destination D de coordonnés (X', Y').

- Distance Euclidienne (théorie de Pythagore) :

$H(N)=\sqrt{(X \quad X')^2+(Y \quad Y')^2}$

- Distance maximum :

H(N)=max (|X-X'|, |Y-Y'|)

- Distance de Manhattan :

H(N)= (|X-X'|+|Y-Y'|)

Donc on calcule un chemin optimal par A* de la façon suivante:

1. créer deux listes vides Ouverte et Fermée.
2. insérer dans la liste Ouverte le nœud de départ (S).
3. tant que la liste Ouverte n'est pas vide et ne contient pas le nœud destination :
    a. chercher dans la liste Ouverte le nœud dont le coût estimé (F) est minimal.
    b. ajouter le nœud à la liste Fermée.
    c. supprimer le nœud de la liste Ouverte.
    d. pour chaque nœud voisin V du nœud actuel N :
    i. calculer G' comme la somme de cout G de N et du coût associé au nœud adjacent V.
    ii. si V est dans la liste ouverte :
        si le coût G' de V est inférieur à G
            Mettre à jour le Nœud V (cout G, F, Parent).
    iii. sinon ajouter V à la liste ouverte.

1. si le nœud d'arrivé n'est pas dans Fermée, terminer l'algorithme sur un échec : il n'existe pas de chemin depuis le nœud de départ vers le nœud d'arrivé.
2. sinon, reconstruire le chemin en suivant l'information parent dans la liste Fermée.

On peut simplifier l'algorithme A* par le pseudo code suivant

```
Fonction A *(départ, destination)
 Initialise Liste ouverte et Liste fermée à liste vide
Ajoute départ à Liste ouverte
Coût g(départ) = 0
Coût f = Coût g( départ) + Coût h (départ, destination)
 Tant que la liste ouverte n'est pas vide
NoeudActuel =Nœud de la liste ouverte avec Coût f minimum
 Si (NoeudActuel = destination)
     Retourne chemin
```

```
    Supprime NoeudActuel de la liste ouverte
    Ajoute NoeudActuel à la liste fermée
Pour chaque Voisin dans noeuds voisins(NoeudActuel)
     Si le Voisin est dans la liste fermée
          Continue
         Coût g temporaire Voisin = coût g (NoeudActuel) + distance(NoeudActuel,Voisin)
    Si Voisin n'est pas dans la liste ouverte ou i le coût g temporaire< coût g
        (voisin)
    Si voisin n'est pas dans la liste ouverte
            Ajoute voisin à la liste ouverte
        Coût g(voisin) =Coût g temporaire
        Coût [(voisin) =Coût g (voisin) +Coût h (voisin, destination)
retourne "impossible de trouver le chemin"
```

Algorithme Pseudo Code de A Star (A*)

## 6. la Navigation

La navigation est la phase qui se fait après la fin de la planification de chemin, son but est de faire naviguer les entités virtuels dans l'environnement on faisant des comportement précis qui se basent sur la capacité physique de l'entité virtuel ,dans cette phase l'environnement est bien représenté et le chemin est planifié .Dans notre travail la navigation est basé sur les comportements réalisés par l'agent virtuel , c'est grâce à ces comportement et la capacité physique de l'agent que ce dernier peut se déplacer librement dans l'espace de travail pour que la navigation soit bien faite elle doit baser sur la perception pour aider à déterminer la prochaine action a accomplir . Dans notre système la navigation est basée sur le comportement

Cette figure illustre la tâche de navigation

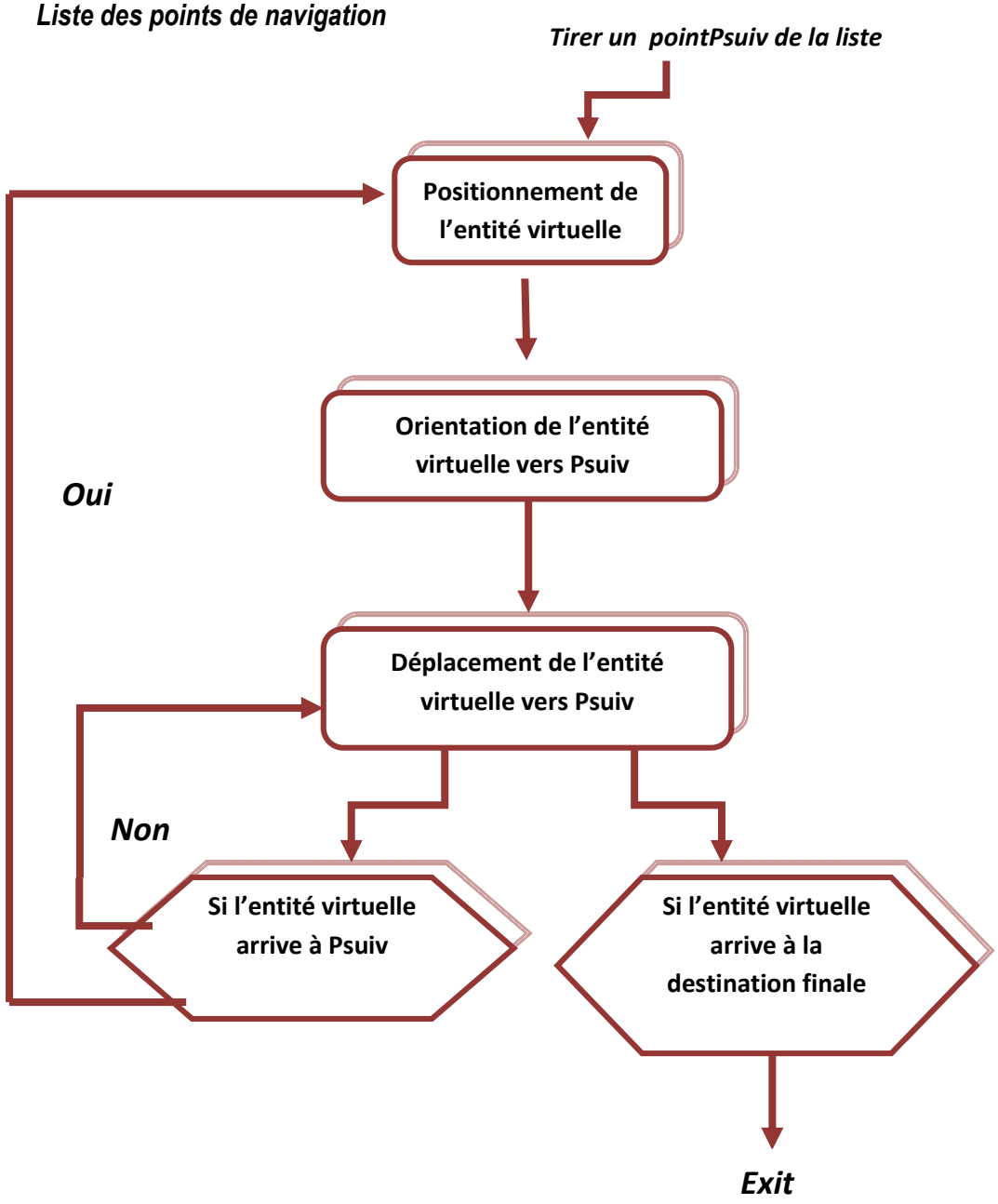



***Figure3.6 :*** *la tâche de navigation*

***Positionnement:*** permet de positionner l'entité virtuelle dans les coordonnés des points de navigation.

***Orientation:*** permet de calculer la direction de l'entité virtuelle pour aller au point suivant, le vecteur de direction $\bar{D}$ peut être calculé comme suit :

Soit la position de l'entité virtuelle ayant comme coordonnés(X, Y) et les coordonnées du point destination (X', Y') :

$\bar{D}=\overline{XX'}$donc $Dx$= (X'- X) et $Dy$= (Y'- Y)

3.2.1 ***Déplacement :***On a ***: d*=v*** **t**, tel que :

**d** : le déplacement de l'entité.

**v** : la vitesse de l'entité virtuelle.

**t**: le temps écoulé depuis le dernier frame.

**a. la perception**

La perception dans l'environnement constitue un critère essentiel du mécanisme de prise de décision. Chaque agent effectue des actions via un mécanisme de prise de décision, Cette opération est très importante dans quelques comportements comme la détection de collision et l'évitement d'obstacle. La perception se fait d'une manière mathématique par lancer de rayon.

**b. Les comportements individuels**

Les comportements individuels sont proposés par Craig Reynolds [6] ces comportements simple sont réalisé par chaque entité on utilisant l'information locale

```
Début r.
 Lancer rayon
Si ( r .intersecté par un (Object)
changement de direction
Sinon Avancement ;
 Fin
```

### b.1. l'évitement de collision

- **Collision de face:** Si les deux agents se déplacent dans une direction opposée alors l'agent doit dévier avec un angle approprié (éviter à gauche ou bien éviter à droite).

- **Collision en arrière:** Si les deux agents ont la même direction alors on distingue deux cas de réactions possible :

  l'agent adapte sa vitesse à l'agent qui le précède et marcher derrière lui.

  l'agent dépasse l'agent qui le précède en choisissant le type d'évitement approprie.

- **Collision de côté:** on distingue deux cas de réactions possible :

  Ralentir : Si la distance entre l'agent et la position attendue de la collision est assez suffisante.

  Arrêter : Si la distance entre l'agent et la position attendue de la collision est moins suffisante.

Le pseudo code de l'évitement :

```
Début
r. Lancer rayon ;  // (  vecteur3.direction )
Si ( r .intersecte avec  (zone non navigable ou obstacle mobile ) ;
 Changer de direction ;    (évitement appliquant un algo s'éloigner ou eviter )
Sinon Avancement ;
Fin.
```

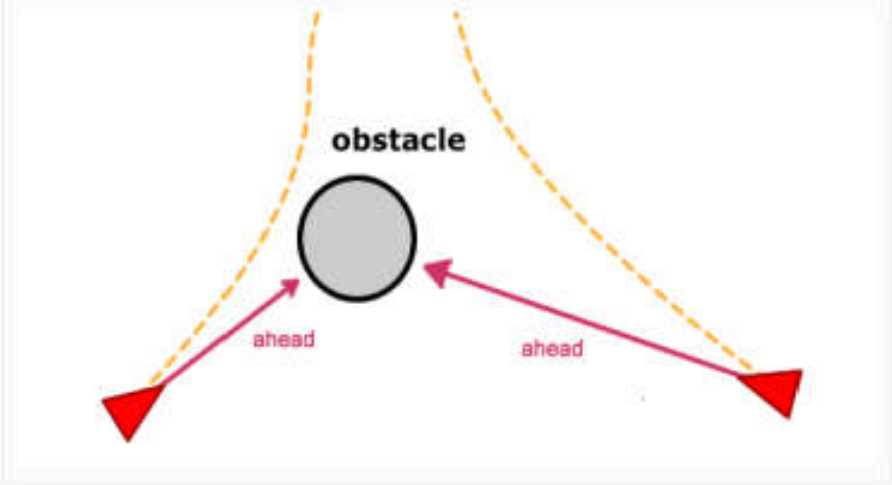



**FIGURE 3.7** exemple d'évitement de collision lancer de rayon / changement de direction

### b.2. Poursuite et fuite

Une poursuite est le processus de suivre une cible visant à l'*attraper*. Si un agent ne fait que suivre une cible, tout ce qu'il a à faire est de répéter les mouvements de la cible et, par conséquent, il sera sur sa piste. Lors de la poursuite, le poursuivant doit ajuster son chemin avec celui-ci de la cible, mais aussi d'avoir anticipé où la cible sera dans un proche avenir. Si l'entité ou l'agent peut prédire (ou estimer) où sera la cible dans les prochaines secondes, il est possible d'ajuster la trajectoire actuelle pour éviter les routes inutiles:

**pseudo code :**

Poursuite (pos_agent à suivre)

Début Suivre le but avec une force de recherche

Fin

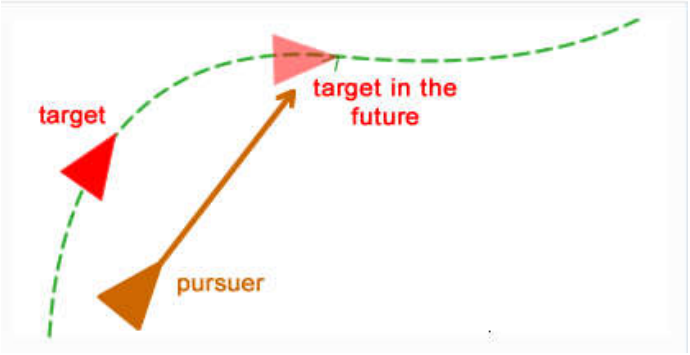



**FIGURE 3.8** exemple de pour agent-agent[5]

### b.3 L'évasion(Evade)

Ce comportement produit une force de direction similaire à celle de la fuite mais reste que l'agent ne connait pas la prochaine position de but ou l'agent qu'on va éviter. Dans ce processus, âpres l'estimation de la position de but dans la prochaine seconde on l'évite. . [6]

**Pseudo code :**

Début

Dists = position –position de but ;

Max vélocité = Dist.normalize () ;

Mise à jour de position=dist /Max_vilocité ;

Fuite(Agent) ;

Fin.

### b.4. Comportement d'errance

Errer est un type de direction aléatoire. Une mise en œuvre facile devrait produire une force de direction aléatoire à chaque encadrement, mais cela produit un mouvement plutôt non intéressant. C'est une situation inconfortable qui ne permet de produire aucune tournure supportée. Une approche plus intéressante est de conserver l'état de la direction et d'y intégrer de petits déplacements aléatoires pour chaque encadrement. Cette idée peut être mise en œuvre selon plusieurs voies, mais celle ayant produit de bons résultats doit contraindre la force de direction à la surface d'une sphère placée légèrement devant l'acteur. Pour produire la force de direction pour l'encadrement suivant : Un déplacement aléatoire est ajouté à la valeur précédente et la somme est contrainte de nouveau à la surface de la sphère[5].

### C. Les comportements de groupe

sont des comportements qui sont produit par de simples comportement, ces comportements sont dictent comportements émergent.

### C.1.1 La séparation

Le comportement de séparation donne à l'acteur la capacité de maintenir une certaine distance de séparation des autres caractères qui sont proches de lui.

Les forces répulsives pour chaque acteur voisin sont additionnées ensemble pour produire la force de direction[6]

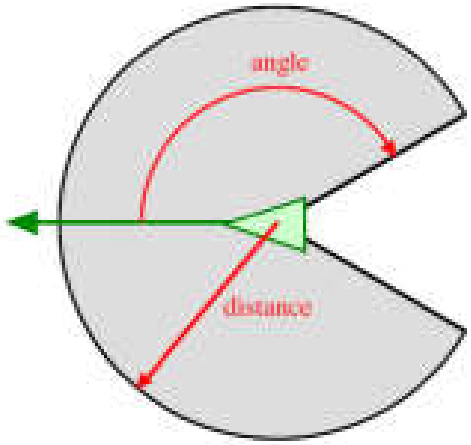



**FIGURE 3.9** notion de voisinage (Reynolds, 1999)[6][7].

```
PSEUDO CODE

Début

Pour (chaque groupe) faire

Pour (tout  agents du groupe ) faire

Si (distance (position actuel, position  voisin.x de mm groupe < espace de personnel)alors

Calculer la force d'éloignement ;

action();

Finsi

Finpour

Finpour
```

### C.2. La cohésion

Le comportement de cohésion donne aux acteurs la capacité de générer une situation de cohérence dans le groupe qu'ils forment (se rapprocher et former un groupe) avec d'autres acteurs voisins (figure 3.9)[5]. La direction de la cohésion peut être calculée en déterminant tous les acteurs dans le voisinage local (comme décrit plus haut pour la séparation) et en calculant 'la position moyenne' (ou centre de gravité) des acteurs voisins. La direction de la force peut ainsi être appliquée dans la direction de 'la position moyenne'.

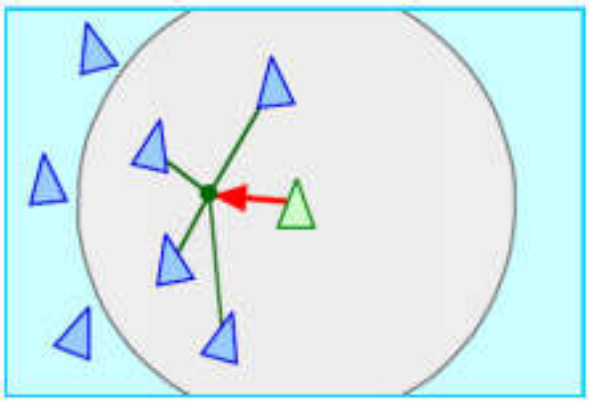


**FIGURE 3.10** Le comportement de cohésion[6]

Début

Pour (chaque groupe) faire

Pour (chaque membre de groupe) faire

Si (voisin n'est pas soit même) alors

Calculer le centre de masse de groupe ;

Finsi Calculer la direction vers le centre de groupe ;

Finpour;

finpour;

fin.

## 7 . Combinaison de comportements

Les différents comportements de direction décrits ci-dessus peuvent servir de modules à des modes de comportement plus complexes. Ainsi, des comportements plus complexes et plus en rapport avec la réalité peut être construit à l'aide de comportements élémentaires. voici un exemple pseudo code

d'une combinaison des trois comprtement simples de groupes (l'alignement + la séparation + la cohésion) ,( Flocking) :

Pour (chaque groupe) faire

Pour (chaque membre de groupe) faire

Cohésion(i) = calculer cohésion (agent) ;

Alignement(i ) = calculer alignement(agent) ;

Separation (i) = calculer separation (agent) ;

Flocking =cohesion(i) + alignment(i) + separation(i) ;

Return force (Flocking)

## 8. Conception générale de notre système

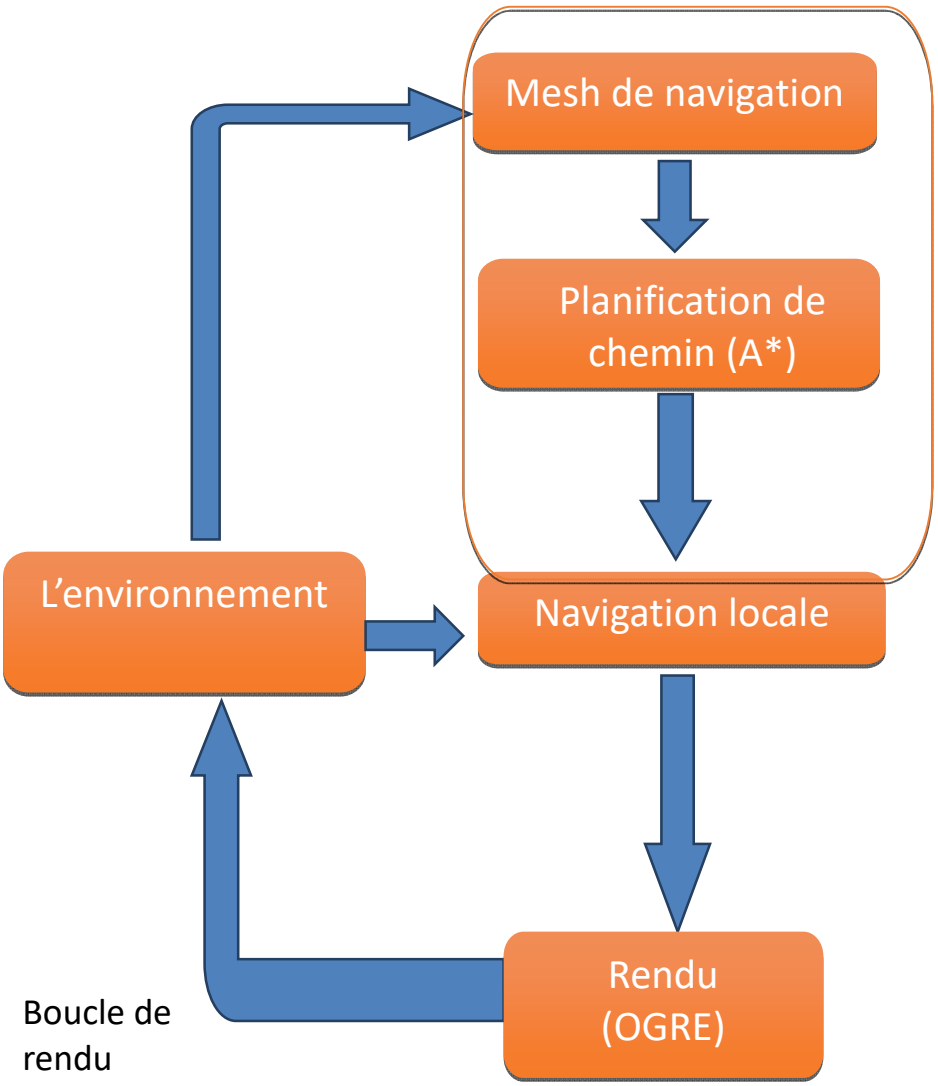



**Figure.3.11** Conception générale du système

## 9. Conclusion

Dans ce chapitre on a défini les tâches de chaque module dans notre système, consiste à faire une simulation comportementale dans un environnement bien planifié ou le monde virtuel contient un nombre d'humain virtuel aussi d'obstacles statique et mobile puis nous avons détaillé dans les différents modules qui sont nécessaires pour la réalisation de notre système a la fin pour assembler tous ces modules pour faire la conception générale. Dans le chapitre suivant, nous allons présenter la phase de la réalisation de cette conception ainsi les outils utilisés pour implémenter notre système.

# Chapitre 4

## 1. Introduction

L'implémentation de notre modèle proposé nécessitera la disponibilité d'un environnement physique réaliste (3D) parce qu'il nous permet de réaliser un monde virtuel proche du monde réel qui de plus devra permettre à nos agents virtuel de réaliser des mouvements réalistes proches des mouvements de l'humain réel. Il y'a lieu, enfin de conclure que dès lors l'étape du choix de l'environnement réalisée, il sera question de trouver le moyen ou bien l'outil qui assure la simulation de toutes les caractéristiques nécessaires pour cet environnement, c'est le moteur physique.

## 2. Les moteurs physiques

Plusieurs travaux cherchent à faire une simulation réelles des comportements des humains virtuels dans des environnements physiques réalistes et en leur faisant reproduire des mouvements réalistes également. De ce fait, le seul moyen qui leur permet de garantir ce réalisme réside dans la simulation où l'on a à reproduire les lois de la physique. La simulation crédible du comportement des humains virtuels sur un environnement 3D incluant la physique était alors un travail relativement complexe à faire. Actuellement et depuis les années 2000, nous disposons des outils (moteurs physiques libre) et du matériel (puissance des ordinateurs) nécessaires pour une bonne simulation d'humains virtuels quelque soit leur formes ou les mouvements qu'on voudra leur faire reproduire. Parmi les moteurs physiques qui sont à l'usage public (ou libre) nous pouvons citer, Breve1 , ODE2 , Newton Dynamics, OGRE et bien d'autres, ceux là étant les plus répandus. Notre choix c'est retourné vers le moteur physique le plus populaire qui est Ogre (Ogre Engine).

## 3-Environnement OGRE 3D

OGRE est un moteur de rendu 3D libre et gratuit qui permet, à partir d'objets à facettes, de réaliser un environnement en 3D. Celui-ci sera perçu par un rendu 2D au travers d'une caméra.

OGRE est une surcouche utilisant les APIs DirectX et OpenGL, qui permettent l'utilisation

De cartes accélératrices 3D (OGRE ne fournit pas de moteur de rendu 3D logiciel, il faut

Une carte 3D ou un émulateur de cartes 3D).


**Figure 4.1** Ogre 3D

➢ **Avantage**

- Multi plateforme : OGRE fonctionne aussi bien sous Windows, Linux ou MAC.
- Compatible OpenGl et DirectX.
- Très performant : il gère les effets graphiques les plus récents et affiche un Frame Rate très satisfaisant.
- Offre une architecture "professionnelle" : son design est tel qu'il peut parfaitement s'intégrer a des applications graphiques professionnelles. Le travail en équipe ne pose pas de problème grâce à la POO et sa modularité offre de grandes possibilités de personnalisations.
- Très bien documenté : OGRE offre une API de très bonne qualité, un manuel complet et des tutoriaux tous très bien fait. Le forum est aussi très actif et peu beaucoup aider.

➢ **Inconvénients :**

- Difficile a prendre en main : en effet même si il est très bien documenté et bien pensé, OGRE reste un moteur très complet et par conséquent complexe, tant dans le mode de fonctionnement que dans l'utilisation. Il vous faudra de nombreuses heures avant de le prendre en main. Mais une fois cet effort effectué, il sera votre meilleur outil !
- Trop fouillis : OGRE permet de faire tellement de choses qu'il comporte beaucoup de dossiers et de fichiers qui paraissent « ésotériques » lors des premières utilisations. Mais au fur et à mesure, on se rend compte à quel point cette organisation est logique

### 3.1. Fonctionnalités d'OGRE

- **Affichage 3D**

Le but principal d'une application 3D est d'afficher un agencement d'objets 3D (des modèles) à l'écran. OGRE gère divers formats d'objet, qui peuvent être préalablement modélisés sous différents logiciels, open source ou payant, tels que 3DStudio Max, Blender, Maya, etc. Il offre de nombreuses fonctions afin de gérer ces objets : déplacement, rotation, agrandissement… Un format de fichier spécifique est utilisé par OGRE pour ces modèles, ainsi que pour leurs textures.

Il est également possible d'afficher les polygones qui composent un objet, généralement dans le but de repérer facilement les erreurs dans une application.

- **Animation**

Si certains objets doivent être immobiles, d'autres en revanche nécessitent des mouvements. Pour cela, OGRE supporte différents types d'animations. Les animations de type « squelettes » sont prévues dès la modélisation de l'objet. Les mouvements sont préalablement créés par le modeleur et calculés par le logiciel de modélisation, puis exportés dans un fichier. OGRE se contentera alors de charger ce fichier, et de lancer la ou les animations choisies. Il est également possible de programmer des animations directement avec OGRE. En effet, il suffit d'indiquer à un objet un mouvement (translation, rotation), ainsi que le temps qu'il lui est accordé pour effectuer ce mouvement. Le moteur se chargera de mettre à jour la position de notre objet pour chaque frame.

- **Caméra**

Afin d'effectuer le rendu de la scène, OGRE a besoin que le programmeur lui spécifie un, Ou plusieurs angles de vue. Pour cela, des objets spéciaux sont insérés dans l'application : Les caméras. Une application peut contenir une seule caméra et tout de même changer D'angle de vue durant son exécution. De même, plusieurs caméras peuvent rendre différents points de vue à l'écran simultanément : dans le cas d'un jeu vidéo, le cas le plus courant est l'écran divisé en deux lors d'une partie multi-joueurs sur une seule machine.

Dans la première image, une caméra est utilisée pour regarder le sol, tandis que dans la seconde, on regarde à la fois le sol et le ciel, via deux caméras.

- **Éclairage**

OGRE offre quatre types d'éclairage pour les scènes finales (à noter que la couleur de la lumière est laissée à l'appréciation du développeur).

- **Entité**

Une entité, c'est la représentation dans la scène d'un modèle 3D, aussi appelé mesh (mesh signifie maillage en anglais).

Le mesh est l'ensemble des polygones élémentaires (des quadrilatères et des triangles, le plus souvent) que l'on crée dans un logiciel de modélisation, comme 3DS Max ou Blender ou Maya, et qui constituent votre modèle 3D complet. Tous ces polygones sont reliés entre eux par leurs sommets. Plus il y a de sommets, et donc de polygones, plus le mesh est précis. Généralement, le mesh possède aussi une ou plusieurs textures (une image plaquée sur les polygones) qui lui donnent un aspect plus réaliste qu'une couleur unie lors du rendu. Enfin, pour certains modèles comme les personnages, le mesh possède un squelette qui permet de créer des animations pour son mouvement.

Les mesh sont gérés par le gestionnaire MeshManager. Ils sont généralement chargés à partir du format de fichier spécifique à OGRE : le format ".mesh". Ces fichiers sont créés grâce aux outils d'exportation (section 3.2.3) disponibles dans la plupart des logiciels 3D, comme ils peuvent être crées manuellement en appelant la méthode MeshManager::createManual.

- **Scène-manager**

Afin de pouvoir gérer l'ensemble des objets de notre scène tout au long du déroulement du programme, OGRE introduit la classe de gestion de la scène "SceneManager".

Concrètement, le SceneManager s'occupe de garder une trace de tous les modèles, lumières, caméra et autres objets que peut contenir la scène. C'est à lui que revient la tâche de créer tous ces objets et de nous permettre ensuite d'y accéder. Par conséquent, l'insertion d'objets dans la scène passe toujours par lui (ou par l'un des éléments déjà insérés) et par les méthodes qu'il propose pour cela.

Il en existe diverses variantes en fonction de la scène que l'on veut réaliser ; selon que celle-ci sera en intérieur ou en extérieur, par exemple, on pourra utiliser un SceneManager différent.

## 4. QT CREATOR

**Figure 4.2** Qt Creator

Qt est principalement dédiée au développement d'interfaces graphiques en fournissant des éléments pr édéfinis appelés widgets(pour windows gadgets) qui peuvent être utilisés pour créer ses propres fenêtr es et des boîtes de dialogue complètement prédéfinies (ouverture / enregistrement de fichiers, progres sion d'opération, etc), fournit également un ensemble de classes décrivant des éléments non graphiqu es :

accès aux données, connexions réseaux (socket), gestion des fils d'exécution (thread), analyse XML, etc

### 5. Langage C++

C'est le langage de programmation le plus utilisé ,notamment en ce qui concerne les applications. Il permet d'aborder le développement sous plusieurs paradigmes : programmation générique, procédurale et orientée objet. C'est un langage compilé, ce qui signifie que le code source est traduit en code objet, ou binaire pour que la machine puisse l'exécuter

### 6. Classes

**Class**

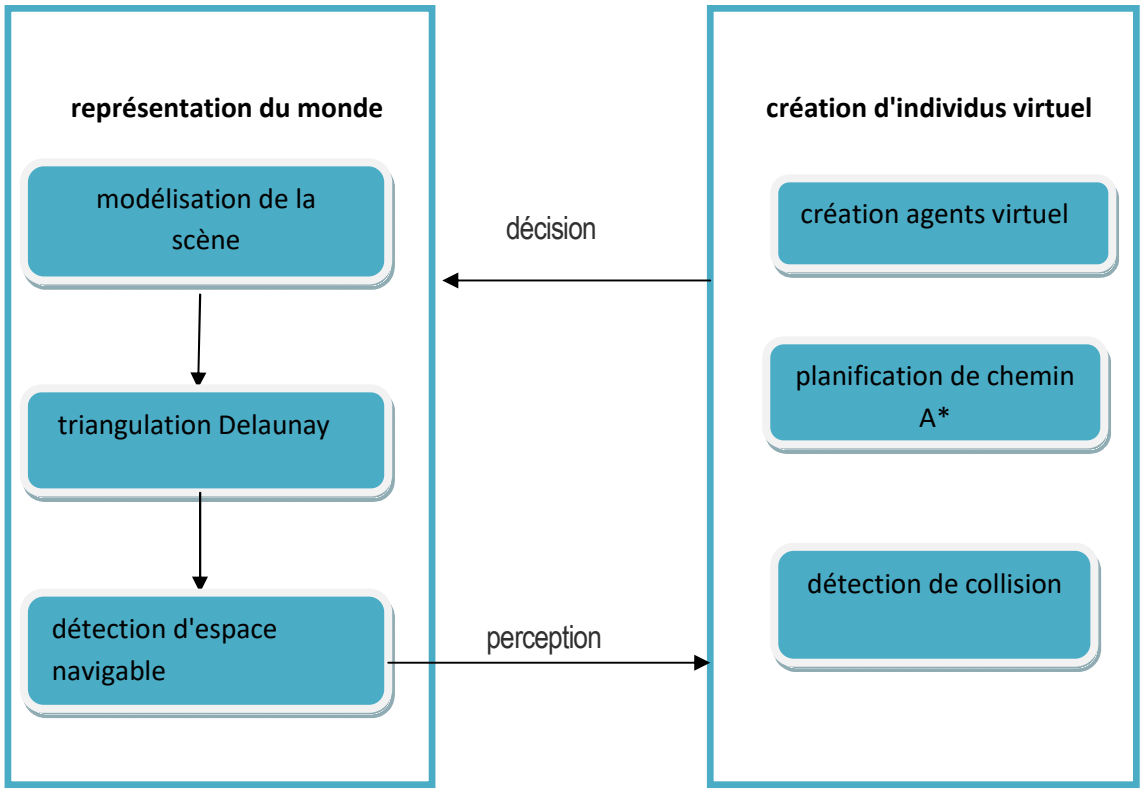
- agent : contient les différents méthodes alignement() séparation() cohésion()
- Delaunay : structure de donnée de Delaunay
- planA: l'implimentation de A*
- qtogrewindow : gestion de fenêtre Ogre avec QT
- mainwindow :
- SdkQtCameraMan :

**- comportement individuel utilisée :**

- suivez de chef
- évitement de collision
- suivez le chemin le plus court
- **- comportement de groupe utilisée :**
- séparation() , cohésion() , alignement
- Flocking

## 7. Applications et résultats

l'architecture de la première application et sous la forme suivante :



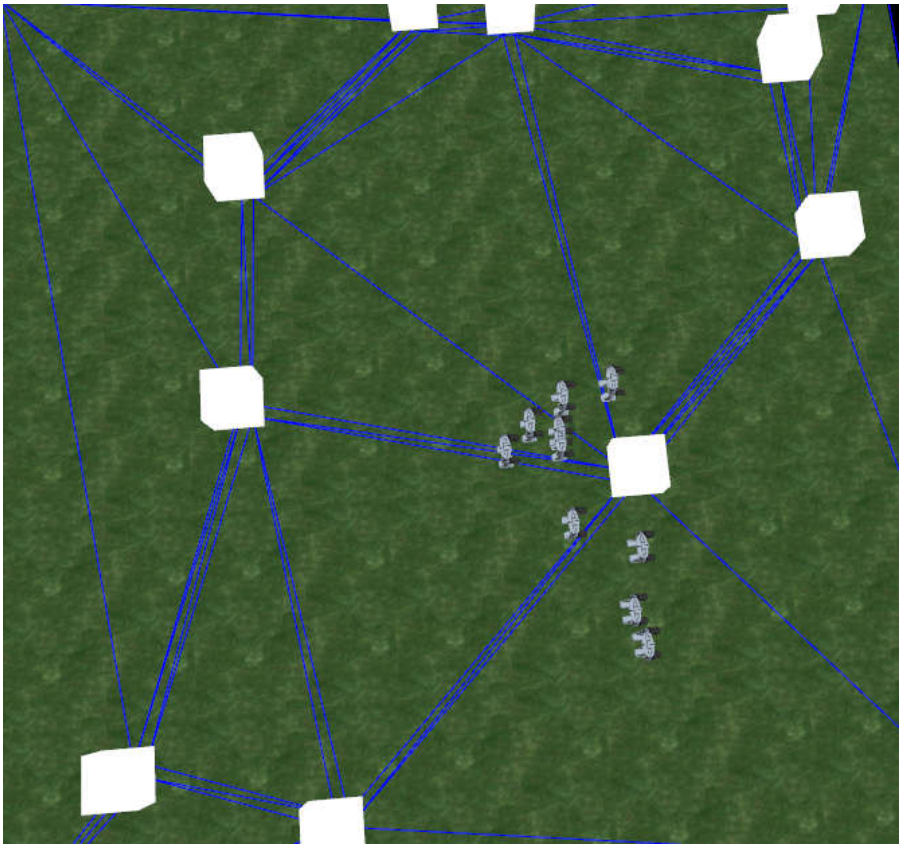


**Figure4.3** navigation d'entités utilisant triangulation Delaunay et A*

**la deuxième application** détection de collision

déplacement de deux groupe d'individus composée de faible et haute densité vers de point différents données

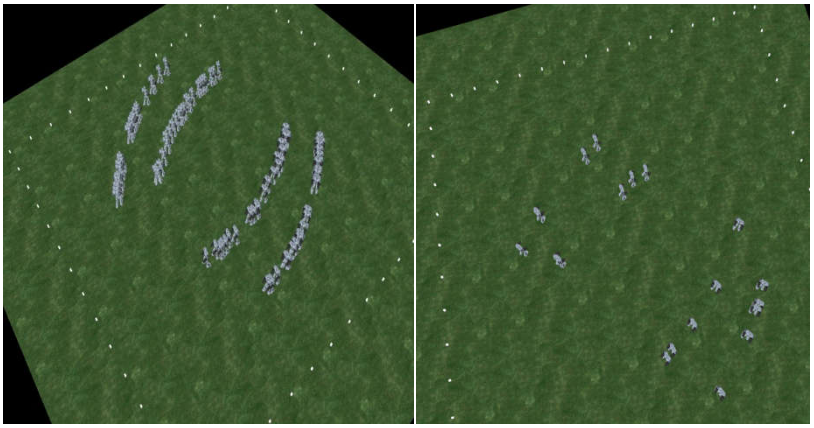


**Figure4.4** déplacement d'individu ( faible , forte densité)

**la troisième application Flocking**

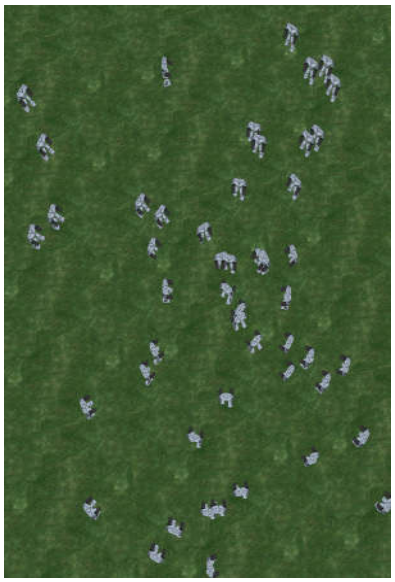


**Figure 4.5** cohésion

**Figure 4.6** séparation

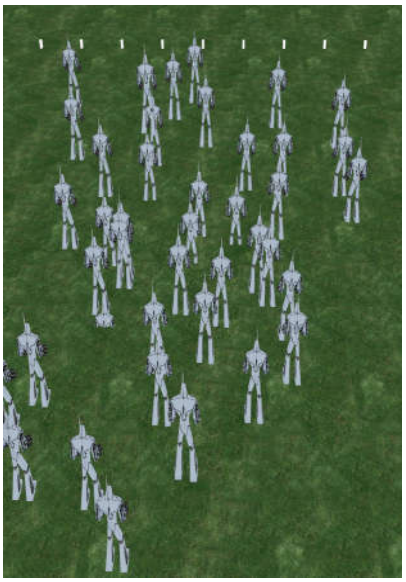


**Figure 4.7** alignement

**application suivez de chef**

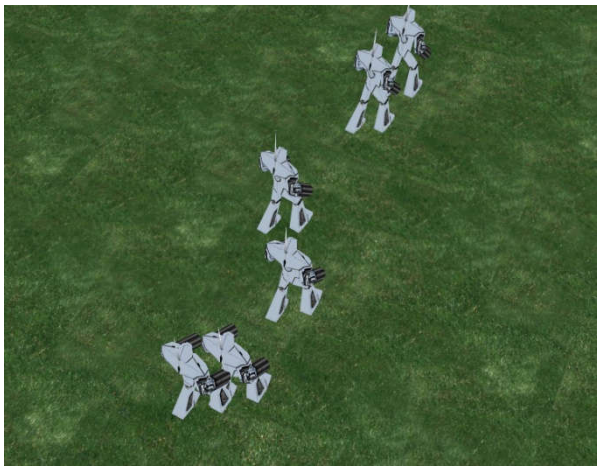


**Figure4.8** suivez de chef

## 8. Conclusions

Nous avons abordé tout au long de ce chapitre les résultats obtenus lors de ce mémoire. Ces résultats prennent tout d'abord la forme d'une architecture dédiée à la simule de croisement de groupe d'individus virtuel La première application concentre sur les techniques d'évitement de collision entre les acteurs, ainsi que sur la recherche de chemin dans un environnement statique.

la deuxième application basé sur le mouvement et le déplacement de deux groupe d'un point source a autre cible donnée dans l'environnement

la troisième et la quatrième application est basé sur le comportement du groupes et d'individu séparation alignement cohésion et les différents cas entre eux et a la fin comportement suivez de chef ( leader follow)

## Conclusion générale

Nous avons intéressé de simuler l'animation des foules humaine en temps réel, on exploitant le principe de l'animation comportementale pour animer les individus. Nous avons concentré notre effort sur les techniques d'évitement de collision ainsi que sur les algorithmes de recherche de chemin dans un environnement statique.

Nous avons proposé pour chaque individu un ensemble de règles comportementales ( C.reynolds) pour résoudre le problème d'évitement de collision entre les individus et Nous avons choisi l'algorithme de triangulation Delaunay pour la représentation de l'environnement et A* pour trouver un chemin ou un itinéraire initial. Ces deux techniques rehaussent le réalisme de la simulation d'animation d'une foule des individus.

Les résultats Obtenus sont satisfaisants mais il reste quelque perspectives afin d'améliorer le travail.

- Optimiser la structure de donnée de la triangulation de Delaunay.
- l'étude de la haute densité d'individus virtuel est trop couteuse en temps de calcule le parallélisme est une solution pour avoir un rendu réaliste et en temps réel

# BIBLIOGRAPHIE


[1] A. Schadschneider. Cellular automaton approach to pedestrian dynamics-theory. In Pedestrian and evacuation dynamics, pages 75–86, 2001.

[2] S. Paris, “Caractérisation des niveaux de services et modélisation des circulations de personnes dans les lieux d’échanges”, thèse de doctorat, Université de Rennes 1, 2007.

[3] Fabrice Lamarche et Stéphane Donikian, « Crowds of Virtual Humans: a New Approach for Real Time Navigation in Complex and Structured Environments ». Computer Graphics Forum, Eurographics, volume 23(03), 2004.

[4] Stéphane Donikian, “Modélisation, contrôle et animation d’agents virtuels autonomes évoluant dans des environnements informés et structurés”, Université de Rennes 1. (26/08/2004).

[5] Jean-Daniel Boissonnat et Mariette Yvinec, « Algorithmic Geometry ». Cambridge University Press, 1998.

[6] C. W. Reynolds. Steering behaviors for autonomous characters. In Game Developers Conference. http ://www. red3d. com/cwr/steer/gdc99, 1999

[7] C. W. Reynolds. Flocks, herds and schools : A distributed behavioral model. In SIGGRAPH '87

[8] C. Burstedde, K. Klauck, A. Schadschneider, and J. Zittartz. Simulation of pedestrian dynamics using a two-dimensional cellular automaton.

[9] B. Maury and J. Venel. Un modèle de mouvements de foule. ESAIM : Proc., 18 :143–152, 2007.

[10] R. Narain, A. Golas, S. Curtis, and M. C. Lin. Aggregate dynamics for dense crowd simulation. In SIGGRAPH Asia '09 : ACM SIGGRAPH Asia 2009 papers, pages 1–8, New York, NY, USA, 2009. ACM.

[11] S. Chenney. Flow tiles. In SCA '04 : Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation, pages 233– 242, Aire-la-Ville, Switzerland, Switzerland, 2004. Eurographics Association.

[12] L. Henderson. The statistics of crowd fluids. Nature, 229 :381–383, 1971.

[13] S. Donikian. Modélisation, contrôle et animation d'agents virtuels autonomes évoluant dans des environements informés et structurés. Documents d’habilitation, 2004.

[14] A. Newell. Unied theories of cognition. Harvard University Press, Cambridge, MA, USA, 1994

[15] THÈSE INSA Rennes par Thomas LOPEZ Planification de chemin et adaptation de posture en environnement dynamique

[16] J.-D. Boissonnat et M. Yvinec. Algorithmic Geometry. Cambridge University Press, 1998.

[17] O. Arikan, S. Chenney, et D. A. Forsyth. Efficient multi-agent path planning. Dans Computer Animation and Simulation ’01, pages 151–162. Springer-Verlag, 2001.

[18] Septseault (C.). Représentation d'environnements virtuels informés et de leur dynamique par un personnage autonome en vue d'une crédibilité comportementale. PhD thèse. Université de Brestagne Occidentale. 2007.

[19] Becheiraz (P.) et Thalmann, (D.). A Behavioral Animation System for Autonomous Actors personified by Emotions. Dans: Proceedings of the First Workshop on Embodied Conversational Characters (WECC '98), Lake Tahoe, California. 1998.

[20] Beer (R.D.). Intelligence as Adaptive Behavior: Experiments in Computational Neuroethology. New York: Academic Press. (1990).

[21] Fabrice Lamarche and Stephane Donikian. Crowd of virtual humans : a new approach for real time navigation in complex and structured environments. Computer Graphics Forum, 23(3) :509-518, 2004.

[22] Thomas H. Cormen  Introduction to Algorithms, Second Edition 2001 by The Massachusetts Institute of Technology

[23] Triangulation de delaunay ,telechargé le 22/06/,2019, https//fr.wikipedia .org , consulté le 24/06/2019.