REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mohamed Khider – BISKRA
Faculté des Sciences Exactes, des Sciences de la Nature et de la Vie
Département d'informatique

N° d'ordre : IVA01/M2/2024

Mémoire

Présenté pour obtenir le diplôme de master académique en

Informatique

Parcours : **Images et Vie Artificielle (IVA)**

Adaptation de posture pour la navigation d'entités virtuelles dans un environnement 3D

Par :
Salma Lansab

Soutenu le 23 Juin 2024 devant le jury composé de :

Meadi Mohamed Nadjib	MMA	Président
Mr Mebarek boucetta	MCB	Rapporteur
Babahenini Sara	grade	Examinateur

Année universitaire 2023-2024

Dédicaces

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 sesprécieux conseils, sa patience et sa persévérance dans le suivi.*

Résumé

L'évolution rapide des technologies de réalité virtuelle et augmentée a rendu la navigation dans des environnements 3D de plus en plus immersive et interactive. Dans ce contexte, l'adaptation de la posture des entités virtuelles constitue un défi majeur pour améliorer l'expérience utilisateur et la fluidité des interactions. Ce mémoire explore les différentes techniques et algorithmes utilisés pour ajuster la posture des avatars et autres entités virtuelles en fonction de la topographie, des obstacles et des objectifs de navigation dans un environnement tridimensionnel. Nous analysons l'impact de ces adaptations sur l'ergonomie et le réalisme perçu, en mettant l'accent sur l'importance de la synchronisation entre les mouvements de l'utilisateur et la réponse des entités virtuelles. De la modélisation 3D et de l'ergonomie, notre recherche vise à proposer des solutions innovantes pour améliorer la réactivité et la pertinence des ajustements posturaux. Les résultats obtenus montrent que l'intégration de ces techniques peut considérablement améliorer la satisfaction utilisateur et la précision des interactions dans des environnements virtuels complexes.

Mots clés : environnements 3D, adaptation de la posture, entités virtuelles, navigation,.

Abstract

The rapid evolution of virtual and augmented reality technologies has made navigation in 3D environments increasingly immersive and interactive. In this context, adapting the posture of virtual entities represents a significant challenge for enhancing user experience and the fluidity of interactions. This thesis explores various techniques and algorithms used to adjust the posture of avatars and other virtual entities based on topography, obstacles, and navigation objectives within a three-dimensional environment. We analyze the impact of these adaptations on ergonomics and perceived realism, emphasizing the importance of synchronization between user movements and the responses of virtual entities. 3D modeling, and ergonomics, our research aims to propose innovative solutions to improve the responsiveness and relevance of postural adjustments. The results obtained show that integrating these techniques can significantly enhance user satisfaction and the accuracy of interactions in complex virtual environments.

Keywords: 3D environments, adaptations, virtual entities, navigation.

ملخص

لقد أدى التطور السريع لتقنيات الواقع الافتراضي والمعزز إلى جعل التنقل في البيئات ثلاثية الأبعاد أكثر غامرة وتفاعلية بشكل متزايد. وفي هذا السياق، يشكل تكييف وضع الكيانات الافتراضية تحدًيا كبيرًا لتحسين تجربة المستخدم وسيولة التفاعلات. تستكشف هذه الأطروحة التقنيات والخوارزميات المختلفة المستخدمة لضبط وضعية الصور الرمزية والكيانات الافتراضية الأخرى وفًقا للتضاريس والعقبات وأهداف التنقل في بيئة ثلاثية الأبعاد. نقوم بتحليل تأثير هذه التعديلات على بيئة العمل والواقعية المدركة، مع التركيز على أهمية التزامن بين حركات المستخدم واستجابة الكيانات الافتراضية. من النمذجة ثلاثية الأبعاد وبيئة العمل، يهدف بحثنا إلى تقديم حلول مبتكرة لتحسين استجابة وملاءمة التعديلات الوضعية. تُظهر النتائج التي تم الحصول عليها أن دمج هذه التقنيات يمكن أن يحسن بشكل كبير من رضا المستخدم ودقة التفاعلات في البيئات الافتراضية المعقدة.

الكلمات المفتاحية: البيئات ثلاثية الأبعاد، التكيفات، الكيانات الافتراضية، التنقل.

Table des matières

Table des figures

Liste des tableaux

Introduction Générale

Les avancées technologiques récentes ont permis l'intégration croissante des mondes virtuels dans divers domaines d'application. Ces environnements immersifs sont souvent animés par l'agents autonome, contribuant à rendre ces univers plus interactifs et réalistes. Un aspect crucial pour maximiser l'immersion et l'interaction est l'adaptation de la posture l'agent en environnement virtuel, permettant une interaction plus naturelle et crédible avec leur entourage virtuel. Dans le cadre de ce travaille, nous nous sommes penchés sur les problématiques liées à la navigation de ce agent au sein des monde virtuel.

L'adaptation de la posture l'agent autonome est essentielle pour garantir une interaction fluide et crédible avec l'environnement virtuel et les utilisateurs humains. Cette adaptation repose sur des techniques avancées d'animation et de contrôle, permettant aux agent de répondre en temps réel aux changements de l'environnement et aux interactions avec les utilisateurs. Elle est particulièrement pertinente dans les environnements dynamiques où les conditions évoluent de manière imprévisible, nécessitant des ajustements continus de la part l'agent. Les principales problématiques rencontrées incluent la réactivité en temps réel, où l'agent doivent adapter leur posture instantanément en réponse aux changements de l'environnement, ce qui exige des algorithmes de contrôle et d'animation hautement réactifs, ainsi que des temps de calcul suffisamment rapides pour ne pas affecter la fluidité de l'interaction.

La crédibilité et le naturel des mouvements sont également des aspects critiques, nécessitant une modélisation précise des comportements humains et une animation réaliste. Les techniques de capture de mouvement et les algorithmes de machine Learning peuvent être utilisés pour améliorer la qualité de l'animation. La planification de chemin dans des environnements dynamiques pose un autre défi, car le agent doivent être capables de planifier leur chemin de manière efficace dans des environnements où la configuration peut changer de manière imprévisible. Les algorithmes de planification doivent être robustes face aux obstacles mouvants et aux changements soudains dans l'environnement. L'interaction avec les utilisateurs humains nécessite que l'agent puissent comprendre et répondre aux intentions et aux actions humaines, impliquant la reconnaissance des gestes et des commandes vocales, ainsi que la capacité à maintenir une distance personnelle appropriée.

La gestion des conflits et des priorités est une autre problématique majeure. Lors de la navigation, l'agent peuvent rencontrer des conflits d'intérêts, comme éviter des collisions tout en respectant des priorités d'objectifs, nécessitant des mécanismes de résolution de conflits efficaces pour garantir une navigation fluide et sans heurts. L'optimisation de la performance est également cruciale, car les algorithmes de planification et d'adaptation de posture doivent être suffisamment performants pour être intégrés dans des applications interactives en temps réel, telles que les jeux vidéo. L'optimisation des ressources de calcul est essentielle pour maintenir la performance globale du système. De plus, les agents doivent être capables d'apprendre de nouvelles situations et de s'adapter en conséquence, ce qui peut impliquer l'utilisation de techniques d'apprentissage automatique. La capacité à s'adapter aux préférences et aux comportements des utilisateurs individuels peut améliorer l'expérience utilisateur.

Enfin, l'interopérabilité et l'intégration des solutions développées sont essentielles pour une adoption large. Les solutions doivent être facilement intégrables dans différentes plateformes et environnements virtuels, et l'interopérabilité avec les systèmes existants et les normes de l'industrie est essentielle. En conclusion, l'adaptation de la posture d'un agent autonome pour la navigation dans des environnements 3D virtuels pose des défis complexes qui nécessitent des solutions avancées en matière de planification de chemin, d'animation, de contrôle en temps réel, et d'interaction humaine. Les progrès dans ces domaines contribueront à des environnements virtuels plus immersifs et interactifs, offrant une expérience utilisateur enrichie.

Chapitre 1

L'environnement virtuelle , Navigation et Adaptation posture

Introduction

Ce chapitre aborde la notion des environnements virtuels, qui constitue la base de toute planification de chemin.Afin d'utiliser efficacement un environnement virtuel, il est crucial qu'il soit bien représenté. Par conséquent, nous examinerons également diverses méthodes de représentation ainsi qu'un ensemble d'algorithmes de planification de chemin.

L'environnement virtuel

Le terme « environnement virtuel » (en anglais « virtual environment ») a été introduit par les chercheurs du MIT (Massachusetts Institute of Technology) au début des années 90 comme synonyme de RV (Heim et al., 1995). Il est considéré comme le lieu suggéré par la RV pour accueillir un ou plusieurs utilisateurs et leur permettre d'accomplir certaines tâches avec l'impression d'être dans un cadre spécifique. L'environnement virtuel est représenté par un modèle 3D de données réelles ou imaginaires qu'on peut visualiser et avec lesquelles on peut interagir en temps réel (Hachet, 2003). Il existe différents types d'environnements virtuels selon le degré d'immersion qu'ils procurent à l'utilisateur : environnement virtuel non-immersif (Non-Immersive Virtual Environment " NIVE "), environnement virtuel semi-immersif (Semi-Immersive Virtual Environment " SIVE "), environnement virtuel totalement immersif (FullyImmersive Virtual Environment " FIVE ") (Kalawsky, 1996).

Type d'environnements

Les environnements peuvent être classés en deux catégories distinctes, chacune ayant sa propre représentation et son propre mode d'utilisation :

Les environnements statiques

Les environnements statiques sont des environnements qui ne subissent pas de modifications au fil du temps. Ils se caractérisent par une structure stable, qui ne change pas avec le temps [9]. Cette particularité implique l'absence de nouvelles accessibilités ou obstructions pendant le déroulement de la simulation. Étant donné que la configuration de ces environnements demeure constante, il est possible d'effectuer des pré-calculs afin de proposer des structures de données performantes pour les requêtes de planification de chemin futures.

Les environnements dynamiques

Pour leur part, ces environnements présentent des caractéristiques évolutives au fil du temps[10]. Nous qualifions un environnement de dynamique lorsque les positions occupées par certains obstacles peuvent varier. Cela peut être attribué à des objets qui modifient leur position, changent de forme au cours du temps, ou apparaissent et disparaissent dans l'espace de travail.

La réalité virtuelle

Définir la réalité virtuelle est une tâche indispensable. On trouve encore dans la littérature et dans les médias des définitions qui mélangent malencontreusement la finalité de la réalité virtuelle, ses fonctions, ses applications et les techniques sur lesquelles elle repose, telles que le visiocasque. Il faut rejeter ces approches, à la fois parce qu'elles sont trop centrées sur une technologie particulière et parce qu'elles sont trop fortement restrictives quant aux enjeux scientifiques et commerciaux. Nous proposons des définitions à plusieurs niveaux pour clarifier le domaine de la réalité virtuelle [11].

La réalité augmentée

À la différence de la réalité virtuelle, la réalité augmentée se rapproche du réel. Historiquement, c'est en 1992 que le terme « Réalité augmentée » (RA ou AR pour Augmented Reality) est introduit par [12] pour désigner des systèmes interactifs combinant la visualisation en temps réel d'objets virtuels et d'éléments physiques dans un environnement réel. L'auteur prototype un casque de type « Head-up display » permettant de connaître la position de la tête dans le monde réel et d'afficher des contenus virtuels à travers un système see-through (auquel on peut voir au travers) en temps réel. Puis, en 1994, Milgram propose une définition plus précise pour la réalité augmentée [13] en émettant l'hypothèse d'un continuum entre le réel et le virtuel qu'il appelle la « réalité mixte » (Figure I.1) composé de la réalité, la réalité augmentée, la virtualité augmentée et le virtuel.

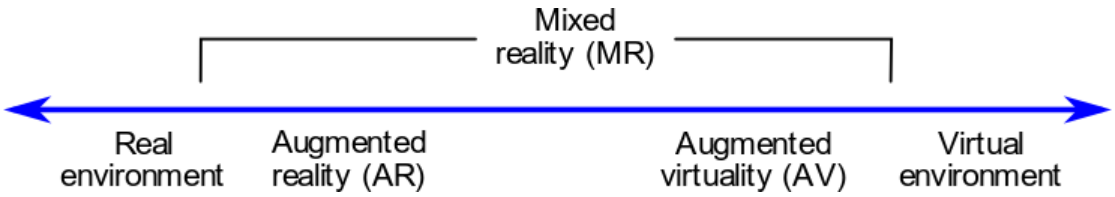


FIGURE I.1 – Le monde réel et le monde virtuel [1].

En 1997,[14] a voulu définir les différences opposant la réalité virtuelle et la réalité augmentée. Il propose ainsi une définition plus structurée de la réalité augmentée qui ne dépend pas de la technologie d'affichage employée. Il met en avant 3 caractéristiques que doivent respecter tous systèmes de réalité augmentée :

— Le système doit pouvoir combiner le réel et le virtuel.

— Le système doit être interactif en temps réel.

— Le système doit être synchronisé en 3D.

Aussi, pour qu'un système de réalité augmentée soit fonctionnel, il est nécessaire de connaître la position et l'orientation de la caméra filmant la scène, mais aussi de pouvoir traquer l'objet réel que l'on souhaite augmenter (image, marqueur, reconnaissance de l'objet 3D . . .). La réalité augmentée, permettant de mixer

du contenu réel et virtuel trouve de nombreux domaines d'applications (divertissement, médecine, éducation, jeux vidéo et maintenance…).

Les domaines d'application

jeu vidéo

Les environnements virtuels occupent une place prépondérante dans l'univers des jeux vidéo, où les joueurs évoluent généralement à l'aide d'un avatar les représentant.

Ces mondes virtuels sont peuplés d'agents autonomes avec lesquels les joueurs peuvent interagir, enrichissant ainsi l'expérience de jeu grâce à leurs actions.

La complexification croissante de ces environnements virtuels, rendue possible par l'amélioration des performances des processeurs et des cartes graphiques, se manifeste par des rendus de décors toujours plus sophistiqués et l'intégration de plus en plus courante de moteurs de simulation physique.

Dans ce contexte, les agents virtuels doivent être en mesure de planifier rapidement leur chemin au sein de ces environnements complexes, en réagissant promptement aux actions de l'utilisateur ou du moteur physique, afin de prendre en compte les modifications topologiques et de proposer des interactions riches aux joueurs, sans que ceux-ci aient à attendre [15].

Robotique

En robotique planifier un déplacement devient encore plus complexe. En effet, il s'agit de se déplacer dans un environnement réel. Tout d'abord, le robot ne dispose que d'une estimation de sa position, car ses capteurs ne sont pas parfaits. De la même façon, il doit se déplacer au moyen d'effecteurs qui ne peuvent l'amener là où il décide qu'avec une certaine imprécision [16].

Le cinéma

Le domaine cinématographique a recours depuis longtemps à des environnements virtuels pour créer des films d'animation, des effets spéciaux intégrés à des scènes réelles, ou encore pour incruster des acteurs virtuels dans des scènes réelles ou inversement, des acteurs réels dans des scènes virtuelles.

Cette technologie a été utilisée dans certains films pour générer des armées d'acteurs virtuels s'affrontant sur des champs de bataille, ce qui témoigne de l'ampleur des possibilités offertes par les environnements virtuels [15].

(a) Le film d'animation « Mulan » de Disney.

(b) Le jeu vidéo « Les Sims 2 » de Electronic Arts.

(c) L'environnement virtuel du jeu en ligne « *Second Life* ».

(d) La simulation d'entraînement *Firefighter Training* (IRIT).

(e) Visite virtuelle de la ville de Grenoble (Artesia).

(f) Visite virtuelle de la cité des sciences et de l'industrie (IRISA).

FIGURE I.2 – Les application de l'environnement virtuelle.

Représentation de l'environnement

L'environnement dans lequel les entités évoluent est représenté par une géométrie permet de traduire la structure de l'environnement et donc les contraintes imposées lors de la navigation, cette géométrie représente les obstacles sous la forme de polygones en 2d et sous la forme de polyèdre 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, plusieurs méthodes consistent à discrétiser l'espace de manière à identifier les zones dans lesquelles l'entité peut naviguer. Cette discrétisation peut ensuite être utilisée pour représenter la topologie des lieux au travers de la notion d'accessibilité entre zones. Cette propriété s'avère nécessaire pour la recherche d'un chemin entre deux points [17].

En effet, disposant de bases de données 3D de l'environnement, le besoin d'une représentation adéquate pour la détection des obstacles et la planification de chemin est primordial. Il est également important de pouvoir attacher une sémantique à cet environnement, permettant par exemple d'adapter le comportement au fait que l'humanoïde soit sur un trottoir ou sur un passage piéton [18].

Représentation des parties statiques de l'environnement

Pour établir une navigation efficace au sein des environnements virtuels, les humanoïdes doivent être capables de planifier leurs trajets de manière à optimiser leurs déplacements. Cependant, les environnements de ce type, souvent dépeints à travers des bases de données 3D, ne sont pas intrinsèquement conçus pour les calculs de navigation spécifiques aux humanoïdes. Ainsi, une organisation structurée des données géométriques

s'avère nécessaire pour faciliter ces calculs. Divers modèles ont été développés dans le but de représenter l'espace libre de ces environnements, offrant ainsi des solutions pour une navigation fluide et efficace [18].

1. Décomposition en cellules

Cette catégorie de méthodes de discrétisation vise à obtenir une carte topologique de l'environnement en procédant à la décomposition de l'espace en cellules qui représentent des nœuds topologiques. Une fois cette décomposition réalisée, il devient possible d'extraire un graphe topologique dans lequel chaque nœud correspond à une cellule et chaque arc illustre une relation de connexité et d'accessibilité entre deux cellules [17].

(a) La triangulation de Delaunay

La triangulation de Delaunay, comme décrite par [1], se caractérise par la création d'un ensemble de triangles à partir des points d'entrée donnés. L'algorithme sous-jacent garantit que chaque triangle formé possède un cercle circonscrit contenant uniquement ses trois sommets, une contrainte essentielle qui entraîne une maximisation de l'angle minimum dans chaque triangle généré. Cette propriété fondamentale de la triangulation de Delaunay aboutit à une représentation précise et complète de l'environnement en deux dimensions.(Figure I.3) [18].

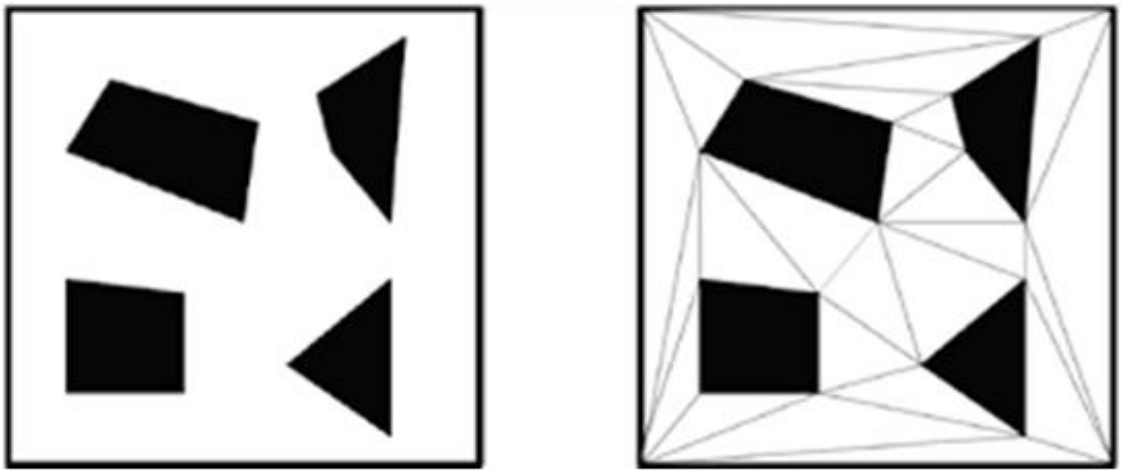


FIGURE I.3 – La triangulation de Delaunay d'un environnement [2]

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 [19]. F.Lamarche propose une version filtrée de cette triangulation [20] permettant de choisir une augmentation ou une diminution du nombre de triangles.

(b) Grille

La méthode de décomposition en grille uniforme, fondée sur la notion de cellules carrées en deux

dimensions et cubiques en trois dimensions [17], permet d'obtenir une représentation approximative de l'environnement dans un espace bidimensionnel ou tridimensionnel [18] (Figure I.4). Cette approche repose sur le principe de quadrillage uniforme de l'espace, qui consiste à diviser l'environnement en cellules régulières. Une évolution de ce modèle est apparue sous la forme des grilles hiérarchiques [21]. Cette méthode décrit l'espace navigable par une succession de grilles de plus en plus précises, organisées sous forme d'arbre. Selon leur position par rapport aux obstacles, ces cellules peuvent être considérées comme vides si elles appartiennent à la zone de navigation, partiellement obstruées si elles contiennent une partie d'un obstacle, ou obstruées si elles sont incluses à l'intérieur d'un obstacle [19].

L'algorithme de construction :

— Va commencer par paver l'environnement avec les cases les moins précises,

— puis découper récursivement les cases partiellement obstruées, en quatre parties pour la 2D (Figure I.4), ou en huit pour la 3D (formation d'un outrée).

— L'algorithme arrête les subdivisions dès qu'une précision fixée est atteinte, ou si la case produite n'est plus partiellement obstruée.

La précision de ce mode de représentation de l'environnement dépend de la taille de la cellule de base, plus les cellules sont grandes moins la représentation est précise et inversement [19].

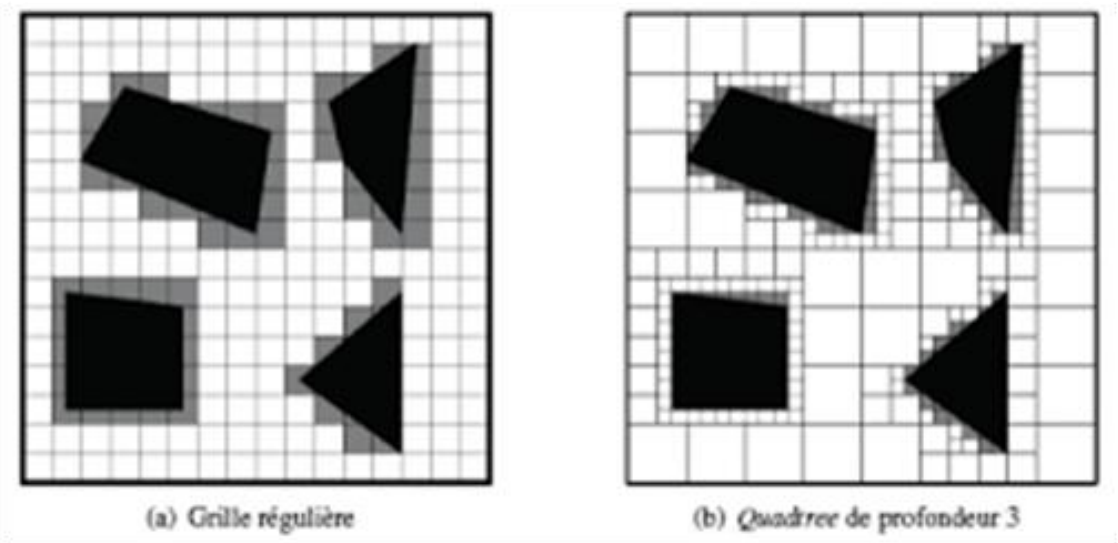


FIGURE I.4 – Exemple de grilles

2. **Cartes de cheminements** Dans le domaine de la navigation autonome, les cartes de cheminement jouent un rôle crucial en discrétisant 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 [19]. Cette méthode permet de calquer un environnement 3D en 2D, rendant ainsi les calculs moins longs car les déplacements y sont désormais plans. Les plans 2D ainsi extraits de l'environnement peuvent être légendés de façon à

faire apparaître les bâtiments comme obstacles pour une navigation urbaine et de marquer les zones navigables [22]. Ces zones peuvent également être divisées en les numérotant, permettant ainsi d'extraire un graphe spatial hiérarchique liant les zones entre elles [20]. Le choix de navigation entre les zones peut donc s'effectuer en parcourant ce graphe [18]. Plusieurs méthodes existent pour créer des cartes de cheminement, différentes dans la manière de créer et de relier les points clefs [17].

(a) Graphe de visibilité

Dans le cadre de la planification de cheminement, une méthode efficace consiste à utiliser les sommets des polygones représentant les obstacles comme points clefs de la carte de cheminement [17]. 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. Les graphes ainsi créés minimisent les distances parcourues [21], assurant d'obtenir des chemins de longueur minimale. La taille du graphe généré est ici dépendante du nombre de points mutuellement visibles, et est donc d'autant plus importante que l'environnement est ouvert avec des obstacles ponctuels et disparates [19].

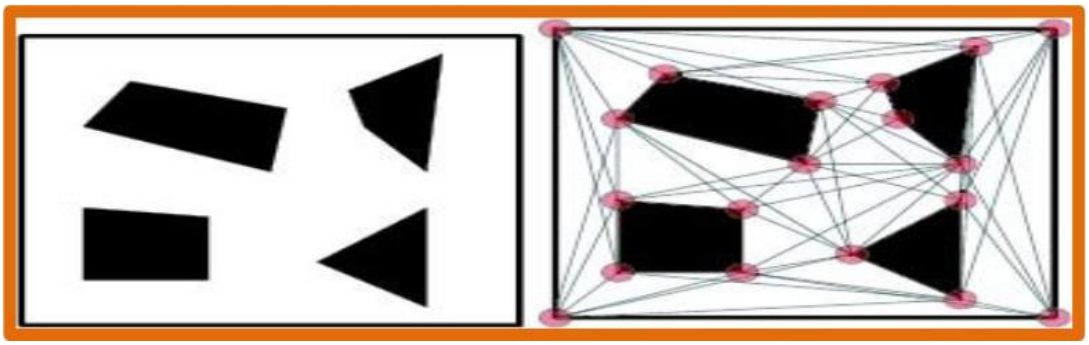


FIGURE I.5 – Graphe de visibilité (à droite) calculé sur l'environnement (à gauche). Chaque cercle (rouge) représente un sommet de la géométrie de l'environnement et chaque segment une relation de visibilité entre ces sommets. Il est à noter que les bordures des obstacles appartiennent au graphe de visibilité.

(b) Diagramme de Voronoï généralisé

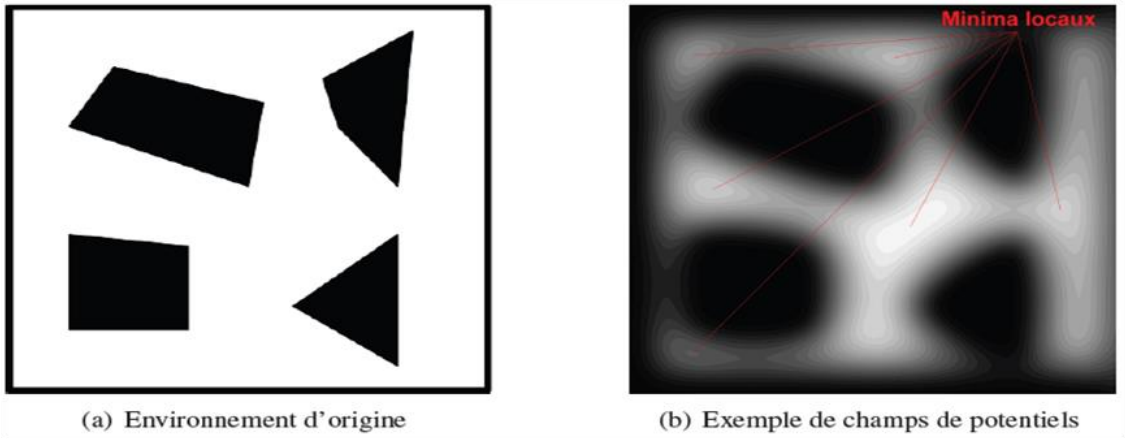
Cette méthode, nous introduisons le concept d'équidistance aux obstacles de l'environnement. Pour ce faire, un ensemble de sites représentant les obstacles est évalué au sein de l'environnement. Les intersections de ces sites forment les nœuds clés, à partir desquels sont générés des chemins maximisant la distance aux obstacles. Bien que le calcul de cette méthode soit complexe, une approximation rapide peut être obtenue grâce à des méthodes utilisant les cartes graphiques [23]. Une autre approche consiste à utiliser la triangulation de Delaunay pour déterminer les points clés du diagramme de Voronoï généralisé, en se basant sur les centres des triangles calculés. La carte de cheminement produite peut alors être considérée comme une synthèse des informations de la triangulation, sans la définition géométrique des obstacles de l'environnement [19].

(c) Cartes de cheminement probabilistes

Avec cette méthode, la définition géométrique de l'environnement n'est pas utilisée comme support direct de la construction. En effet, les points clefs sont ici obtenus par une distribution aléatoire au sein de l'environnement navigable [24]. Deux points sont ensuite reliés s'il existe un chemin libre de collision entre eux. Cette méthode est plus largement utilisée en planification de mouvement qu'en navigation, afin de générer des déplacements assez complexes (sauter, se baisser, marcher sur des piliers).

3. Champs de potentiel

Le modèle à base de champs de potentiels consiste en une définition de l'environnement permettant directement de résoudre les déplacements de personnes [19]. Un potentiel est donc définit pour chaque point de l'environnement comme la somme des potentiels de répulsion liés aux obstacles avec le potentiel d'attraction lié au but. L'entité étant localisée dans l'environnement, la direction à prendre pour converger vers son but est alors opposée au gradient de potentiel définit en ce point [18]. Un gradient de forces, ou champ de potentiel, est alors déduit en chaque point de l'environnement comme étant une somme pondérée, le plus souvent par la distance, des potentiels de répulsion et du potentiel lié au but (Figure I.6). La navigation d'une entité est alors simulée par une descente de gradient depuis sa position dans l'environnement [19].



(a) Environnement d'origine (b) Exemple de champs de potentiels

FIGURE I.6 – Carte de champs de potentiels

Les méthodes basées sur les champs de potentiels s'avèrent simples et efficaces, mais posent le problème des minima locaux : zones de l'environnement où un potentiel minimal isolé apparaît (Figure I.6). Ainsi, la méthode de navigation associée va pousser l'entité à se déplacer vers le minimum local le plus proche, qui ne représente pas forcément son but. Pour pallier ce type de problème, des méthodes à base de marche aléatoire sont utilisées. Par exemple, dans RPP (Random Path Planner)[25], lorsqu'un minimum local est atteint, un ensemble de configurations aléatoires sont tirées puis testées avec une phase de sortie du minimum et une phase de convergence vers le prochain minimum. Les informations sont alors stockées dans un graphe dont

les nœuds sont les minima locaux et les arcs traduisent des chemins entre deux minima. De manière générale, les méthodes à base de champs de potentiels ne sont pas applicables directement à la simulation d'humains, du fait du manque de souplesse dans la méthode de contrôle. Notons tout de même que cette technique a deux grands avantages qu'il serait bon de conserver dans une évolution permettant une décision individuelle plus importante. Premièrement, cette méthode permet la fusion de l'information au sein de l'environnement, traduisant tout ce qui intervient dans la navigation par des potentiels qui sont mélangés. Deuxièmement, et c'est une conséquence directe du premier point, ces méthodes introduisent une abstraction très forte de l'environnement, grâce à laquelle l'individu ne raisonne plus sur des entités perçues indépendantes (que ce soit la topologie des lieux ou les autres individus), mais directement sur une abstraction spécialisée pour sa tâche de déplacement [19].

Environnement virtuel informé

Un environnement virtuel informé se définit comme un environnement virtuel où les modèles 3D ne se limitent pas à la simple représentation géométrique de la scène, mais intègrent également toutes les informations pertinentes pour les entités comportementales à simuler, telles que les éléments symboliques ou sémantiques permettant à ces entités de percevoir, décider et agir. Ces environnements virtuels informés enrichissent les objets de connaissances spécifiques portant à la fois sur le savoir (description de l'objet) et le savoir-faire (interactions possibles avec l'objet) de diverses manières.

Intégration des objets interactifs

L'importance de la géo-localisation des objets dans l'environnement se manifeste par leur connaissance précise de leurs positions spatiales ainsi que des nœuds du graphe topologique qui leur correspondent. Cette géo-localisation trouve sa raison d'être principale dans la facilitation de l'accès des objets aux algorithmes basés sur la définition topologique, notamment pour des applications telles que la planification de trajectoires et l'évitement de collisions [26]. En considérant cette dynamique, il convient de distinguer deux catégories d'objets interactifs : les objets statiques, dont la localisation spatiale reste constante tout au long d'une simulation, et les objets dynamiques, capables de se déplacer au cours de celle-ci.

1. **Les objets statiques**

   Les objets statiques, définis comme des entités fixes au sein de l'environnement, constituent générale-ment des spécialisations de l'effecteur, symbolisant des équipements qui sont à la disposition des inter acteurs.

   Dans cette perspective, il est possible d'identifier trois sous-catégories de ces objets statiques :

   — **Les objets obstacles :** représente un défi significatif pour les entités en mouvement au sein de leur environnement. Parmi ces objets, on retrouve notamment les obstacles statiques tels que les guichets de vente, les murs [3], comme illustré dans la (figure I.7).

FIGURE I.7 – Objets obstacles (mur) [3].

— **Les objets traversables :** constituent des interfaces entre différentes zones navigables de l'environnement, tout en offrant eux-mêmes des possibilités de navigation lors de leur utilisation ou en présentant des obstacles lorsqu'ils sont inactifs. À titre d'exemple, on peut citer un ascenseur, un escalier ou encore un portique automatique (figure I.8).

FIGURE I.8 – Objets traversables (escalier) [3].

FIGURE I.9 – Objets traversables (ascenseur) [3].

— **Les objets libres :** La disposition des objets libres, caractérisée par leur placement stratégique visant à éviter toute entrave à la navigation, révèle leur importance dans la conception de l'espace. Cette précaution s'avère essentielle, car ces éléments pourraient potentiellement être perçus comme des obstacles s'ils étaient positionnés différemment. À titre illustratif, l'installation d'un panneau d'affichage en hauteur, que ce soit sur un mur ou un pilier, démontre la nécessité d'une réflexion préalable quant à leur emplacement afin d'assurer une fluidité dans l'environnement architectural.

2. **Les objets dynamiques** Les objets dynamiques, par leur nature mobile au sein de l'environnement, englobent une diversité de manifestations. Ils peuvent se présenter sous la forme de spécialisations de l'inter-acteur, telles que l'individu virtuel, ou de l'effecteur, tel qu'une valise, voire combiner ces deux aspects, comme c'est le cas d'un agent de service évoluant dans un espace public.

   Du fait de leur caractère non statique, ces objets ne peuvent être directement intégrés au graphe d'abstraction de l'environnement. Par conséquent, nous appliquons une méthode similaire à celle utilisée pour les objets statiques libres : les objets dynamiques sont associés au nœud correspondant à leur géo localisation dans le graphe. De plus, une procédure automatique est mise en place pour gérer le transfert de ces objets d'un nœud à un autre lors de leurs déplacements.

FIGURE I.10 – Objets mobile (humains virtuel)

Planification de chemin

La planification de chemin en environnement virtuel est une tâche cruciale pour les agents autonomes qui peuplent ces environnements. Elle consiste à trouver un chemin libre de collisions entre deux configurations du robot au sein de l'environnement. Cette problématique est définie ainsi : étant donné un robot possédant un nombre variable de degrés de liberté, et une description de l'environnement où ce robot est immergé, comment trouver un chemin libre de toutes collisions entre deux configurations du robot au sein de l'environnement [27].

Algorithmes de parcours

Dans le cadre de la recherche du plus court chemin, différents algorithmes de parcours de graphe ont été conçus pour minimiser un coût, souvent lié à la distance spatiale [19].

— **Dijkstra**

Au sein du domaine de l'informatique et de la théorie des graphes, l'algorithme de Dijkstra se révèle être un outil fondamental pour la détermination des plus courts chemins entre deux nœuds d'un graphe. Cet algorithme, qui repose sur l'utilisation d'une file de priorité, stocke pour chaque nœud exploré deux informations clés :

— la distance accumulée depuis le nœud de départ jusqu'au nœud en cours d'exploration.

— le nœud précédent, c'est-à-dire le nœud visité avant l'actuel.

— L'algorithme de Dijkstra prend initialement un nœud source, généralement correspondant à la position actuelle d'une entité, sans prédécesseur et avec un compteur de distance initialisé à zéro.

Il fonctionne directement sur le graphe topologique, comme les cartes de cheminement ou les grilles [27].

— Cet algorithme se distingue par sa robustesse, dans le sens où il trouvera systématiquement un chemin si celui-ci existe, et ce, sans aucune connaissance préalable de la destination, si ce n'est comme condition d'arrêt.

— Dans la figure suivant, le carreau rose est le point de départ, le carreau bleu représente le but et l'exposition de secteurs de sarcelle (d'élimination) les secteurs que l'algorithme de Dijkstra a parcouru. Les secteurs de sarcelle les plus légers sont les plus éloignés du point de départ et forment ainsi "la frontière" de l'exploration (Figure I.11).

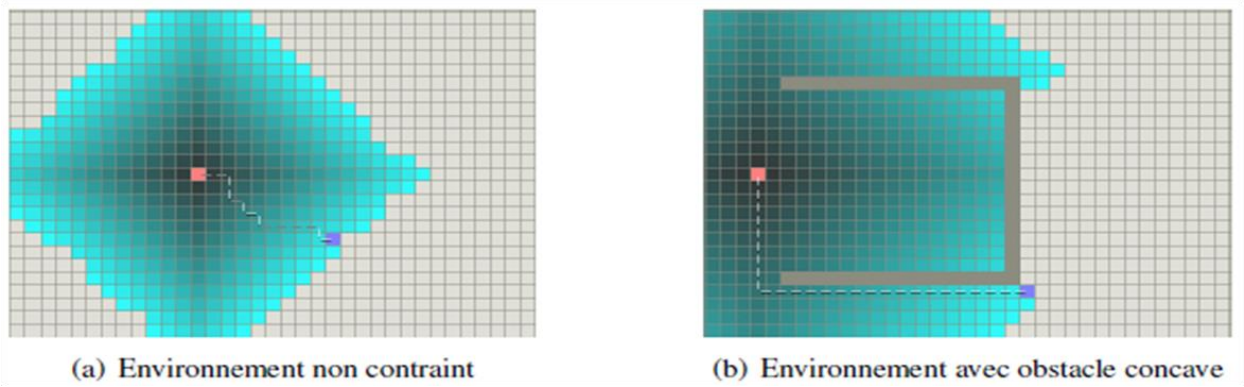


FIGURE I.11 – L'algorithme de dijkstra

— **Breadth-First Search (BFS)**

Dans le contexte de la théorie des graphes, également connu sous le nom de Breadth-First Search (BFS). Ce dernier permet de parcourir un graphe en explorant tous les chemins d'une seule case à chaque itération, jusqu'à ce qu'un chemin entre le départ et l'arrivée soit trouvé [28].

— **Depth-First Search (DFS)**

"Depth-First Search (DFS)", plus communément traduit en français par "parcours en profondeur". Cette méthode de parcours de graphe se caractérise par le fait qu'elle explore chaque chemin jusqu'à son terme, les abordant un à un de manière séquentielle. Cependant, il est important de souligner que le parcours en profondeur n'est pas réputé pour son efficacité lorsque la quantité de chemins est importante et que ces derniers sont particulièrement longs. En effet, l'algorithme DFS cesse son exploration dès lors qu'il a identifié un chemin [28].

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

Cette approche, qui combine les principes du parcours en largeur et en profondeur, se distingue par une exploration progressive des chemins . Plus précisément, elle consiste à explorer le premier chemin jusqu'à une profondeur donnée, puis à réitérer ce processus pour chaque chemin suivant, et ce, jusqu'à ce que tous les chemins aient été parcourus à cette profondeur. À chaque itération subséquente, la profondeur

d'exploration est augmentée, et l'algorithme répète le processus pour chaque chemin. L'algorithme prend fin lorsqu'il a identifié un chemin menant à la solution.

— **Best-First-Search**

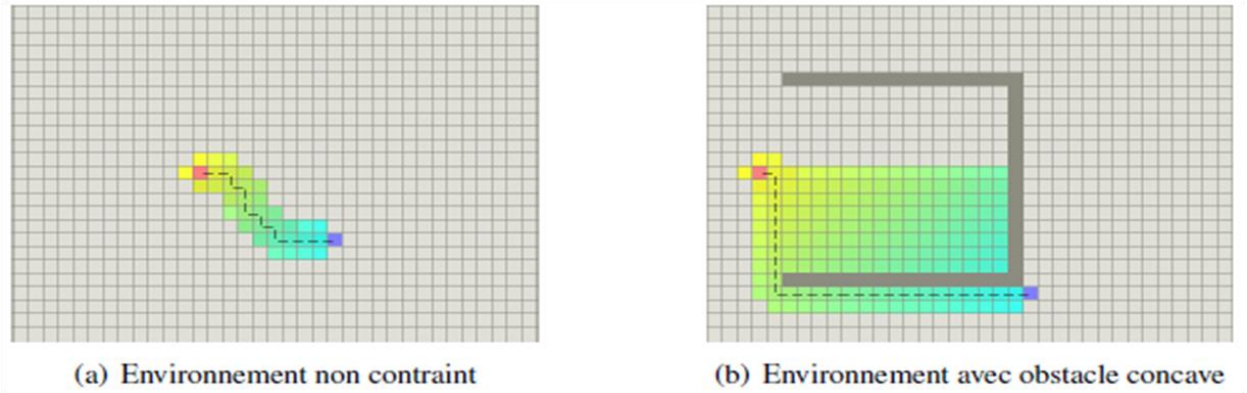
Dans le domaine de la planification de chemin, l'algorithme de recherche Best First Search occupe une place prépondérante en raison de sa capacité à identifier le chemin le plus prometteur avant d'explorer les autres options. Fondé sur l'utilisation d'une heuristique, qui estime la distance jusqu'à l'arrivée, ce type d'algorithme permet d'évaluer l'intérêt relatif des différents chemins possibles. Il est intéressant de noter que les algorithmes A* et B* sont des exemples de Best First Search, qui ont démontré leur efficacité dans divers contextes.

— **B*(B Star)**

L'algorithme B Star, développé par Berliner en 1979, est une variante de la méthode de recherche Best First Search. Cette approche utilise un arbre pour recenser exhaustivement tous les chemins possibles. Par la suite, l'algorithme se concentre sur la branche présentant le coût le moins élevé. Si cette branche ne permet pas d'atteindre la destination, l'algorithme sélectionne une autre branche avec un coût relativement faible.

— **A*(A Star)**

Dans les cas où la performance de la recherche est primordiale, comme en animation comportementale (dans le cas des jeux vidéos par exemple), l'algorithme A* [29] est plus utilisé. Celui-ci fonctionne d'une façon similaire à l'algorithme de Dijkstra mais en ajoutant un coût prédictif aux nœuds correspondant au reste du chemin à parcourir. Le coût total est donc la longueur réelle du chemin jusqu'au nœud, incrémentée par une valeur prédite pour le chemin menant au but, calculée par une heuristique. L'algorithme va ainsi parcourir les nœuds dans l'ordre croissant de leurs coûts associés. La rapidité de convergence de cet algorithme est directement dépendante de la qualité de l'heuristique employée. Dans le cas de la recherche de plus court chemin, l'heuristique employée est une estimation de distance, généralement la norme géométrique séparant le nœud courant du but [27].



(a) Environnement non contraint (b) Environnement avec obstacle concave

FIGURE I.12 – L'algorithme A*

— **D*(D Star)**

Le concept de D* est basé sur l'algorithme Dynamic A* [41], qui s'inspire directement de l'algorithme A*. Cette approche dynamique permet de prendre en compte les changements de coût des chemins pendant l'exécution de l'algorithme, et plusieurs variantes ont été développées pour répondre à des besoins spécifiques.

— **Alternative-Deepening A-Star Search (IDA*)**

Le but de réduire les coûts associés aux chemins, un algorithme a été conçu pour parcourir les chemins trouvés par A* et au lieu de stopper le parcours des chemins avec une profondeur seuil, il est stoppé lorsque le coût du chemin calculé par A* dépasse un seuil précisé. Ainsi les chemins avec un coût trop important sont écartés. Si les chemins sont tous écartés, alors le coût du seuil est naturellement augmenté .

— **Hierarchical Pathfinding A* (HP A*)**

Dans le cadre de l'optimisation des déplacements dans les jeux vidéo, l'algorithme proposé par Botea, Müller et Schaeffer en 2004 offre une approche intéressante.

Ce dernier divise la carte de jeu en plusieurs sections distinctes, dans le but de simplifier les déplacements entre les grandes zones.

Les trajectoires à l'intérieur de ces zones sont calculées de manière conventionnelle, tandis que les déplacements entre plusieurs zones sont effectués par les personnages sur des chemins prédéfinis et préalablement calculés.

L'illustration de cette division de la carte en de nombreux points est présentée dans la (Figure I.13).

Les chemins entre ces points sont donc précalculés, offrant ainsi une optimisation significative des déplacements.

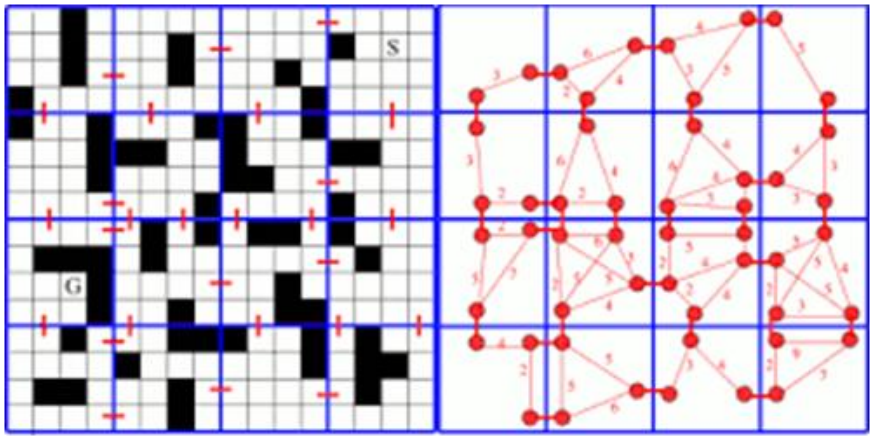


FIGURE I.13 – HPA*

I.9.1.1 Comparaison des algorithmes

Algorithm	Complixité	structure de donées	heuristique
Dijkstra	N2	Liste	Pas d'Heuristique
Best-firs-search	Nlog(n)	Arbre/File	Heuristique
B+	Nlog(n)	File	Manhattan/euclidienne
A+	Nlog(n)	File	Manhattan/euclidienne
Algorithme funnel	Nlog(n)	File	Manhattan/euclidienne

TABLE 1 – comparaison des algorithme

Humain virtuel

Au cours des deux dernières décennies, le développement de modèles réalistes du corps humain a suscité un intérêt croissant dans différents domaines, grâce à l'évolution continue des techniques de modélisation et de simulation assistées par ordinateur. En ce qui concerne le contexte industriel, les applications possibles de ces modèles sont nombreuses et particulièrement importantes sont l'ergonomie dans la conception et la maintenance des produits, la simulation des tâches, la formation du personnel, etc. Les premières applications de modèles humains ont débuté dans les années 1980 dans le domaine du cinéma ; les applications en ergonomie ont immédiatement suivi, comme le démontrent les activités de recherche illustrées dans [30]. Aujourd'hui, plusieurs cadres avec des modèles humains de différentes complexités sont disponibles ; ils peuvent être utilisés pour définir des scènes complexes avec des mannequins virtuels et des objets et pour simuler de nombreuses tâches. Les modèles humains peuvent être classés [31] sur la base des caractéristiques suivantes : apparence (graphiques bidimensionnels, fil de fer tridimensionnel, polyédrique tridimensionnel, modèle cinétique, limites de force, modèle psychologique, etc.), réponse dans le temps (simulation hors ligne, manipulation interactive, etc.), autonomie (interaction, prise de décisions, etc.) et être humain (sexe et âge, données anthropométriques, profil psychologique, etc.). Pour chaque aspect, il est possible d'établir une liste d'attributs qui rendent le mannequin virtuel plus avancé ; selon l'application, des caractéristiques de niveau plus ou moins élevé sont nécessaires.

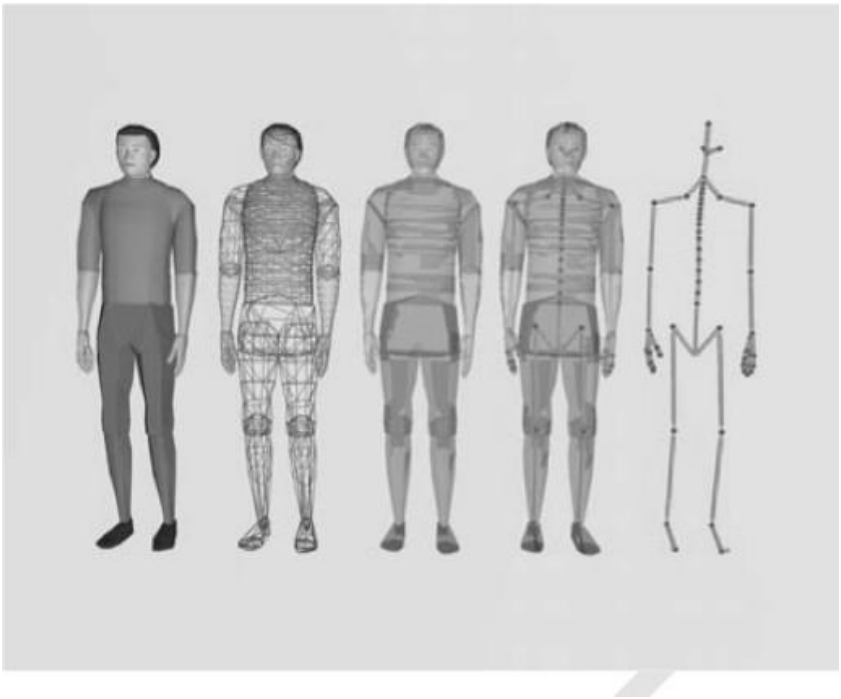


FIGURE I.14 – Apparence différente pour un mannequin virtuel

Par exemple, la simulation et l'analyse ergonomiques ne nécessitent pas de performances de haut niveau en ce qui concerne l'apparence, le temps, l'autonomie et la personnalité, tandis qu'un niveau fonctionnel élevé est nécessaire (grande mobilité, amplitude et limites des articulations, réponse au stress, etc.) [31]. Un humain virtuel est une chaîne cinématique composée d'un certain nombre de maillons rigides reliés par des articulations ; chaque articulation virtuelle reproduit les degrés de liberté de celle physique, avec des limites correspondant au mouvement autorisé d'un être humain. Des algorithmes pour la cinématique directe et inverse ont été développés afin qu'un modèle virtuel puisse reproduire les mouvements du corps humain ; ils sont dérivés de ceux appliqués dans l'analyse multi-corps, décrite par exemple dans[32]. Du point de vue du contrôle du modèle, deux catégories d'humains virtuels peuvent être identifiées : les agents et les avatars. Un agent est un humain virtuel créé et contrôlé par l'ordinateur, tandis qu'un avatar est contrôlé par un humain réel (participant en direct). Un humain virtuel peut avoir des sens virtuels, tels que la vue ; certaines analyses décrites ci-dessous exigent expressément cette fonction. Des caméras virtuelles situées en correspondance avec les yeux peuvent simuler le sens de la vue. Un autre aspect, très important pour les analyses ergonomiques, est lié au sexe et à la taille de l'humain virtuel, car il est nécessaire de modéliser une grande variété de populations.

l'Animation comportemental

L'animation comportementale se concentre sur l'explication des motivations sous-jacentes à un mouvement, plutôt que sur le mouvement lui-même. Cette approche vise à conférer une autonomie aux entités

concernées [4].

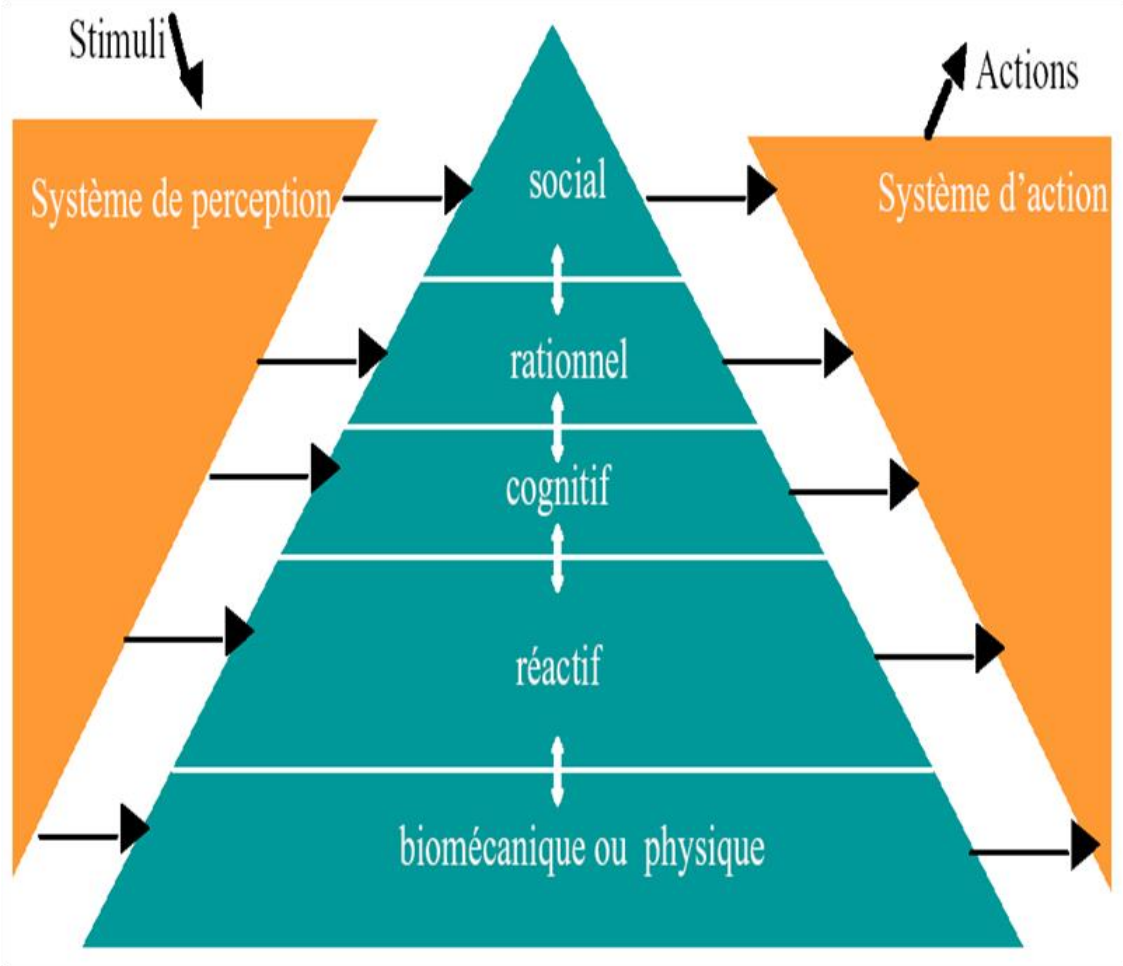


FIGURE I.15 – Niveaux de comportements [4]

La boucle de l'animation comportementale

L'animation comportementale repose fondamentalement sur le modèle de la boucle **Perception-Réaction-Action**. Ce modèle stipule qu'une entité, après avoir perçu son environnement, décide de la prochaine action à exécuter et agit en conséquence sur le monde qui l'entoure [5].

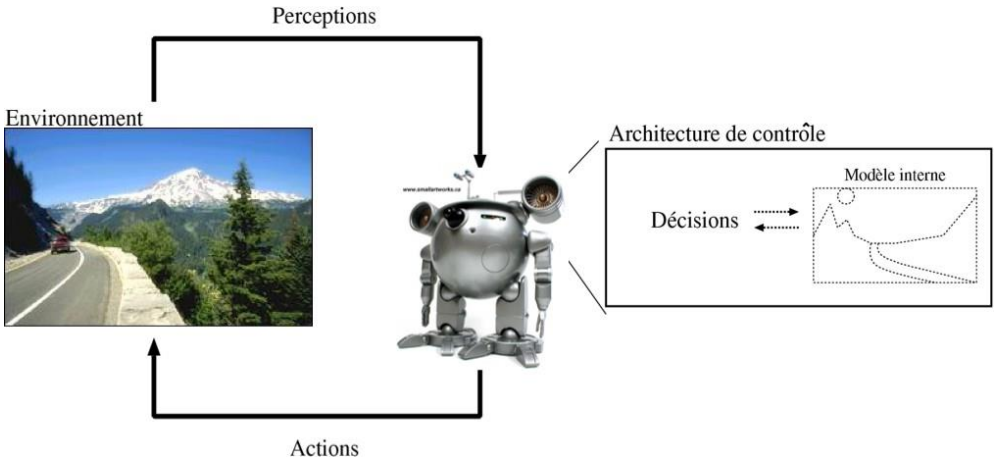


FIGURE I.16 – La boucle de l'animation comportementale [5].

Module de perception

Dans le cadre de l'étude des interactions et du comportement des individus virtuels, la fonction de perception joue un rôle crucial. Elle 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, afin de les traiter et de les utiliser par les autres modules (voir la figure I.17) [6].

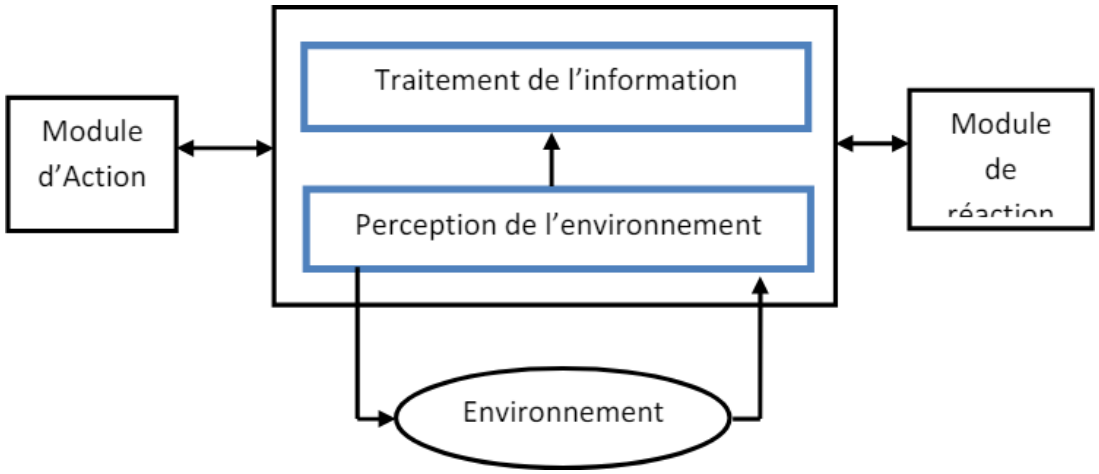


FIGURE I.17 – Structure de module perception [6].

Module de réaction

Après avoir abordé la phase de perception, qui englobe les divers types de traitement de détections d'obstacles et de collisions, nous nous focalisons désormais sur le module de réaction. Ce module est primordial dans

les techniques d'évitement des obstacles et des collisions. En effet, le module de réaction joue le rôle de superviseur en contrôlant l'action globale de l'individu virtuel. Il permet à ce dernier de réagir aux divers événements internes et externes en déclenchant un comportement approprié en fonction du stimulus perçu, conformément au modèle stimulus/réponse, comme l'illustre la figure suivante :

FIGURE I.18 – Structure de module perception [4].

Module d'action

La phase d'action constitue l'étape finale du cycle d'animation comportementale. Elle se manifeste par le déplacement des individus dans la scène. Ainsi, le module d'action peut être défini comme une collection de comportements prédéfinis, tels que le déplacement, l'arrêt et l'évitement d'obstacles. Ces comportements sont indispensables pour la navigation de l'individu virtuel, et incluent :

— **Sélection chemin :** il permet de trouver un chemin, plus ou moins optimal pour une animation réaliste.

— **Eviter obstacle :** il permet d'éviter les obstacles qui se trouvent dans le chemin de piéton virtuel.

— **Eviter collision :** il permet d'éviter les collisions avec les autres piétons virtuels.

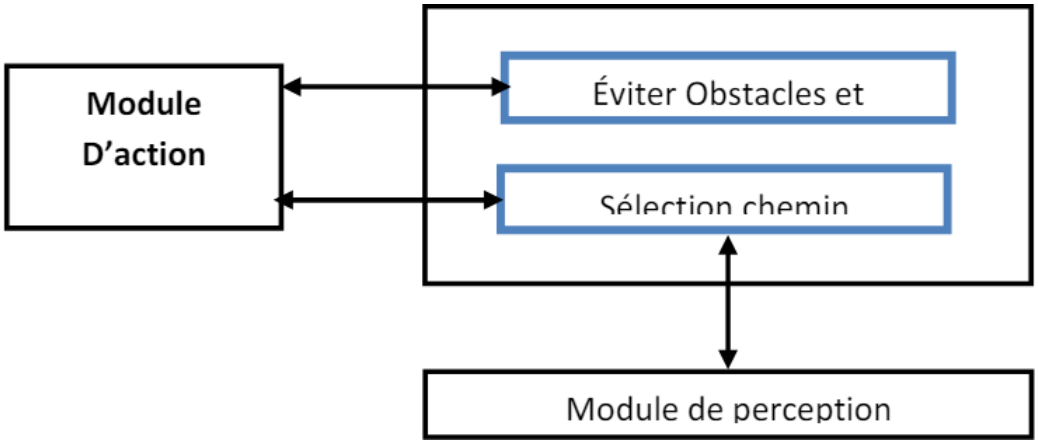


FIGURE I.19 – Module d'action et ses Interactions [4]

Caractéristiques du déplacement des piétons

Comportement des piétons

À l'échelle microscopique, il est possible d'identifier plusieurs caractéristiques distinctes du comportement individuel des piétons. En premier lieu, un piéton se déplace à une vitesse dite « de confort », laquelle varie en fonction de ses caractéristiques personnelles (âge, genre, taille, état de santé, etc.), de ses caractéristiques de déplacement (motif du déplacement tel qu'une visite touristique, longueur du trajet, etc.) et des caractéristiques de l'environnement (comme les conditions météorologiques, par exemple).

Comportement des piétons à l'échelle macroscopique

Dans de nombreuses situations de la vie quotidienne, les foules de piétons en mouvement démontrent des capacités d'organisation sophistiquées. Dans cette section, nous examinerons le phénomène d'auto-organisation, qui désigne l'émergence spontanée d'une structure globale au sein d'un système, résultant des interactions locales entre les agents constituants de ce système. Parmi ces phénomènes, on peut distinguer :

— Le phénomène d'arche, où un blocage se forme autour d'un espace restreint, tel qu'une porte, lorsque une foule dense tente de le traverser (figure I.20).

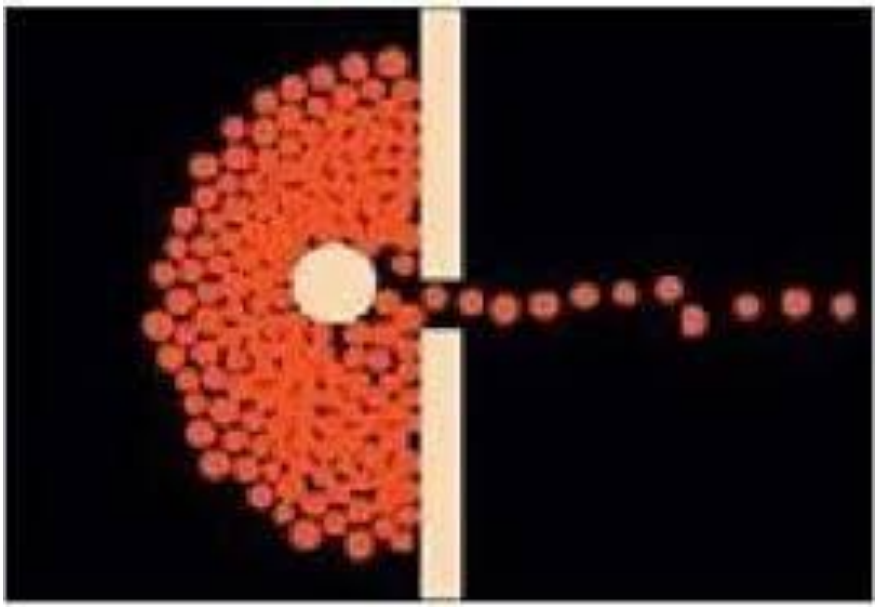


FIGURE I.20 – Le phénomène d'arche

— La formation de files, une autre forme d'organisation, apparaît lorsque deux flux de piétons se déplacent en sens inverse. Cette dynamique engendre naturellement des lignes de piétons suivant la même direction (figure I.21), ce qui réduit les collisions et augmente la vitesse de déplacement des piétons.

FIGURE I.21 – Formation de files.

Réalisme de simulation

L'animation comportementale vise à décrire les principes sous-jacents d'un comportement plutôt que le comportement lui-même. En d'autres termes, l'objectif principal n'est pas d'obtenir une visualisation réaliste, mais de représenter un comportement permettant de comprendre les résultats de la simulation. Ce choix repose sur les actions réalisées par les agents en fonction de leur environnement, ce qui nous amène à décrire différents modèles :

Modèles macroscopiques

Les modèles macroscopiques représentent les premiers types de simulations de foule développés dans le domaine de l'ingénierie des systèmes. Cette prééminence est principalement attribuable à leur faible exigence en termes de ressources de calcul. En effet, ces modèles permettent de simuler un nombre élevé de piétons sans se concentrer sur le comportement individuel de chaque agent (figure I.22). Ils sont couramment utilisés pour les simulations d'évacuation de bâtiments et pour l'élaboration des conditions de sécurité associées à ces évacuations. Par ailleurs, ces modèles visent à reproduire de manière réaliste les phénomènes macroscopiques, tout en négligeant la représentation individuelle des piétons.

FIGURE I.22 – Example de foule humaine de haute densité

— **Modèles statistiques**

Les modèles statistiques, dans leur essence, se concentrent directement sur les débits de piétons, intégrant diverses approches théoriques pour simuler ces flux. Parmi ces modèles, certains se basent sur la dynamique des fluides. Par exemple, Henderson propose un modèle gazeux permettant de simuler des foules de piétons de faible densité, où les états de déplacement (arrêt, marche, course) sont assimilés aux différents niveaux d'énergie des gaz. D'autres chercheurs proposent quant à eux des modèles hydrauliques pour simuler des foules de forte densité.

De plus, plusieurs modèles se focalisent spécifiquement sur les débits de piétons lors de l'évacuation de bâtiments, en s'intéressant notamment aux capacités des portes et des escaliers. Ces modèles fournissent des formules pour calculer le temps d'évacuation en fonction de divers paramètres tels que le nombre de piétons, la largeur des zones de circulation, la pente, ou encore l'espace occupé par chaque personne.

Plus récemment, Colombo et Rosini ont proposé un modèle inspiré des modèles de trafic routier. Ce modèle repose sur deux hypothèses principales : 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, particulièrement pertinent dans des situations exceptionnelles de type panique, où le débit serait quasi-nul en situation normale.

En raison de leur nature macroscopique, ces modèles sont capables de simuler des foules de piétons avec un coût calculatoire très faible, tout en intégrant facilement des observations réelles. Toutefois, ils nécessitent des données 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**

Dans le domaine de la simulation de la dynamique des fluides, les modèles de flots s'inspirent profondément de la mécanique des fluides. Ces modèles reposent sur l'utilisation de champs de potentiel qui influencent le mouvement des piétons, traités comme des particules répondant à ces champs. Par exemple, dans le cadre de la simulation de fluides, une méthode [33] représente des champs de vitesse à travers un carrelage de flots, où chaque carreau contient son propre champ, permettant de créer des flots plus larges par leur assemblage. Les arêtes et les coins de ces carreaux sont conçus pour assurer une continuité, facilitant ainsi la direction non seulement des fluides, mais également des foules humaines. Bien que ces carreaux soient stationnaires, ils peuvent générer des flots dynamiques en fonction du temps. Inspirés par les travaux de Hughes, Treuille et ses collègues ont modélisé les foules de piétons à partir de quatre hypothèses principales :

— Chaque individu vise à atteindre un objectif géographique.

— Cherche à marcher à la vitesse maximale possible.

— évite les zones inconfortables, et choisit le chemin le moins coûteux en termes de distance, temps de parcours, et niveau d'inconfort.

Un champ de potentiel dynamique permet d'intégrer simultanément la navigation globale avec les obstacles mobiles, tels que d'autres piétons, simulant ainsi de grandes foules en temps réel avec une résolution implicite des collisions.

Par ailleurs, Narain et al [8] proposent une approche hybride basée sur les flots continus, où les interactions locales sont résolues par une méthode géométrique, tandis que Maury et Venel [7] présentent un modèle déterministe pour l'évacuation des bâtiments, assimilant chaque individu à un disque. Ce modèle repose sur le principe que :

— Chaque personne a une vitesse souhaitée et que l'espace est considéré comme un flot de champs de vitesse déterminant le chemin le plus court vers la sortie.

— Tout en évitant les chevauchements grâce à une contrainte d'encombrement.

Les modèles de flots offrent une simulation efficace des foules en termes de coût calculatoire et permettent de reproduire certains phénomènes macroscopiques. Cependant, en considérant les piétons comme des particules, ces modèles peuvent manquer de réalisme à l'échelle individuelle, générant parfois des mouvements irréalistes du point de vue biomécanique et des artéfacts au niveau local. Ces modèles sont particulièrement pertinents lorsque les objectifs des piétons sont communs, mais peuvent perdre en réalisme pour des comportements individuels distincts [7].

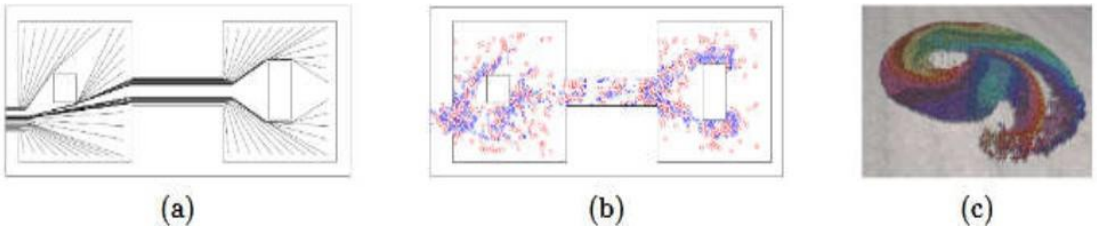


FIGURE I.23 – Modèles de flots. (a). Représentation des flot des champs de vitesse dans [7](b). Exemple d'évacuation [7] (c). Exemple de simulation dans [8] : 10000 agents placés en cercle et devant aller à l'endroit diamétralement opposé.

Modèles microscopiques

Dans le cadre de l'étude des dynamiques de groupe, l'approche microscopique se distingue par sa capacité à identifier les interactions individuelles entre chaque piéton et son voisinage immédiat, influençant ainsi le comportement de l'individu. Ces modèles se distinguent par leur manière de prendre en compte les informations provenant du voisinage d'un piéton et par la réaction de l'individu compte tenu de ces informations.

— **Automates cellulaires**

Certains modèles d'étude, notamment les automates cellulaires, se fondent sur une approche de discrétisation de l'espace en cellules, généralement carrées bien que parfois hexagonales ou octogonales. L'idée centrale de ces automates réside dans le fait qu'à chaque instant, chaque piéton occupe une cellule et ne peut migrer vers une autre que si celle-ci est libre. Un automate cellulaire se compose donc d'une grille régulière de cellules, chacune caractérisée par un état choisi parmi un ensemble fini, susceptible d'évoluer au fil du temps. L'évolution de l'état d'une cellule à l'instant t+1 est déterminée par l'état des cellules voisines à l'instant t, formant ainsi un voisinage fini. À chaque itération temporelle, des règles locales sont appliquées simultanément à toutes les cellules de la grille, générant ainsi une nouvelle génération de cellules en fonction de la génération précédente. Ces règles reflètent le comportement de prise de décision des automates, contribuant ainsi à la création et à l'émulation de comportements réels. Chaque automate évalue ainsi ses opportunités de manière individuelle, et le comportement global émergent du groupe résulte des interactions entre ces règles locales, tout comme chaque piéton examine les cellules disponibles dans son voisinage pour prendre ses décisions.

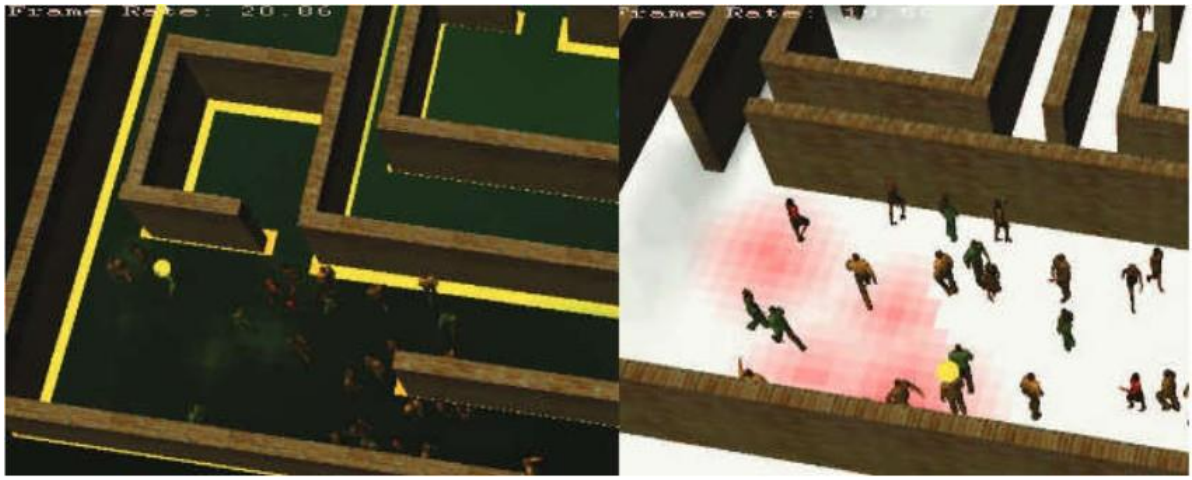


FIGURE I.24 – Simulation de foule : AC (à gauche) et (à droite) structure 2D à la base de grille

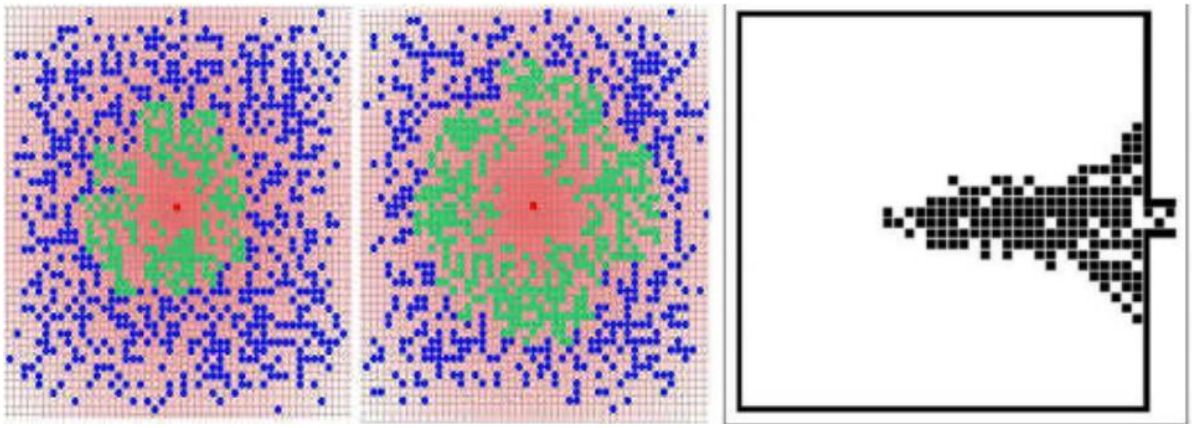


FIGURE I.25 – Simulation d'´evacuation de salle avec le modèle automate cellulaire.

— **Modèles à base de règles**

Les modèles de type règle, initialement proposés par Reynolds, sont illustrés par l'utilisation d'ensembles d'oiseaux pour simuler le comportement des foules (voir figure I.26). Dans ce cadre, les individus d'un groupe sont conceptualisés en tant qu'agents autonomes, chacun se conformant à un ensemble prédéfini de règles :

— **la séparation :** les agents ne collisionnent pas.

— **l'alignement :** ils essayent de garder une direction et une vitesse de mouvement commune.

— **cohésion :** ils essayent de rester unis.

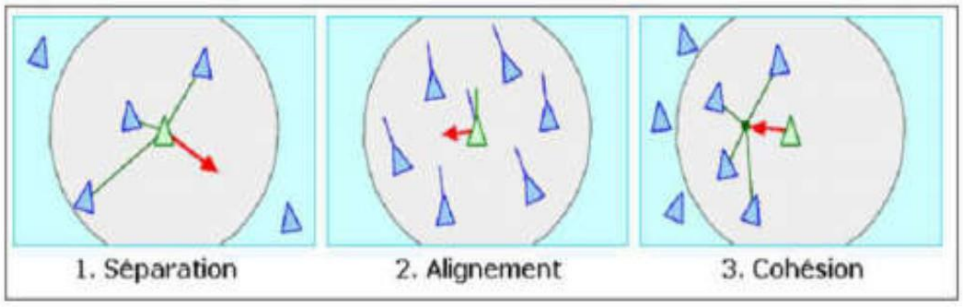


FIGURE I.26 – Les trois règles caractérisant le comportement des oiseaux.

— **Les modèles à base de forces**

Le modèle développé par Helbing, qui s'appuie sur une analogie avec la physique newtonienne, constitue une approche significative pour la gestion du mouvement des piétons. En effet, chaque piéton, représenté par un disque, est soumis à des forces d'attraction et de répulsion agissant sur son accélération conformément à la deuxième loi de Newton **($\sum F = m.a$)**

F : représente la force, **m :** la masse , **a :** l'accélération.

La force d'attraction, également appelée (force motrice) , permet au piéton de se diriger vers sa destination désirée, tandis que la force de répulsion, ou (force d'interaction) , résout les problèmes de collision et gère les interactions entre piétons ainsi qu'entre piétons et obstacles (figure I.27).

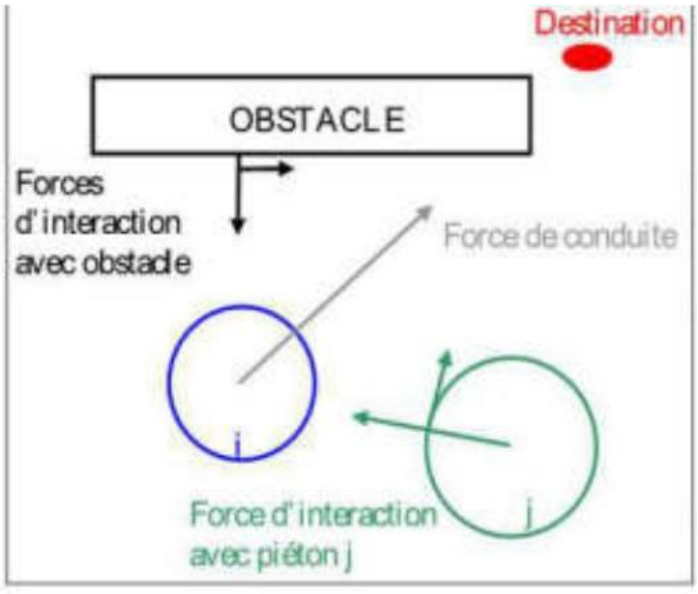


FIGURE I.27 – Modèle à base de force.

L'adaptation posture

L'adaptation de la posture dans un environnement virtuel se réfère à la manière dont les utilisateurs ajustent leur position corporelle et leurs mouvements en réponse aux stimuli et aux exigences d'un environnement numérique simulé. Cette adaptation est cruciale pour améliorer l'immersion, l'interaction, et l'expérience globale dans les environnements de réalité virtuelle (VR). Elle peut inclure des ajustements subconscients pour maintenir l'équilibre, éviter les collisions avec des objets virtuels, et naviguer efficacement dans l'espace virtuel.

Objectif l'adaptation posture

L'objectif de l'adaptation de posture en environnement virtuel est multiple et peut varier selon le contexte et l'application :

1. **Immersion et Réalisme** : Assurer que les mouvements et les postures de l'utilisateur soient correctement reflétés dans l'environnement virtuel pour augmenter le sentiment de présence et de réalisme [34].
2. **Ergonomie et Confort** : Adapter les interfaces et les interactions de manière à prévenir la fatigue et les inconforts physiques, en optimisant les postures pour des sessions prolongées en réalité virtuelle (RV) [35].
3. **Performance et Efficacité** : Améliorer les performances des utilisateurs en optimisant les postures pour des tâches spécifiques, ce qui peut être crucial dans des applications professionnelles ou de formation [36].
4. **Accessibilité** : Adapter les environnements virtuels pour être accessibles à tous les utilisateurs, y compris ceux ayant des limitations physiques, en permettant des ajustements de posture et des contrôles personnalisés [37].
5. **Prévention des Troubles Musculo-Squelettiques (TMS)** : Réduire le risque de développement de TMS par une conception attentive aux postures et aux mouvements [38].

Les méthodes existantes

Voici quelques méthodes couramment utilisées pour l'adaptation de la posture dans ces environnements.

1. **Capture de Mouvement**

   Les systèmes de capture de mouvement utilisent des capteurs pour enregistrer les mouvements corporels et adapter la posture de l'avatar en temps réel. Cela inclut l'utilisation de capteurs inertiels, de systèmes optiques basés sur des caméras, et de technologies de capture de mouvement magnétique [39].

2. **Kinect et Capteurs Profondeur**

L'utilisation de capteurs de profondeur comme Microsoft Kinect permet de suivre les mouvements corporels sans nécessiter de marqueurs, en utilisant des techniques de reconnaissance de formes et de suivi de squelette [40].

3. **Suites Haptique et Exosquelettes**

   Les dispositifs haptiques et les exosquelettes fournissent des retours sensoriels et moteurs pour simuler des sensations tactiles et adapter la posture en fonction des interactions avec l'environnement virtuel [41].

4. **Modèles Biomécaniques et Squelettiques**

   Les modèles biomécaniques utilisent des données anthropométriques et des principes de la cinématique inverse pour estimer et ajuster la posture de l'avatar en fonction des mouvements humains [42].

5. **Réseaux de Neurones et Intelligence Artificielle**

   Les réseaux de neurones et l'IA sont de plus en plus utilisés pour prédire et adapter la posture en temps réel, en apprenant des vastes ensembles de données de mouvements humains [43].

Conclusion

Dans un premier temps, ce chapitre aborde la représentation de l'environnement virtuel. Plus précisément, nous avons examiné les techniques permettant cette représentation, en exposant notamment les environnements informés et l'intégration des objets interactifs. Concernant la représentation de l'environnement, il a été démontré que les approches approximatives offrent un accès rapide à la description topologique, tandis que les approches exactes garantissent la conservation maximale de l'information géométrique.

Chapitre 2

Conception de système

Introduction

Dans le cadre de tout système, la conception est une étape préalable et déterminante, car une bonne démarche du système repose sur une bonne conception. En effet, cette phase joue un rôle crucial dans la description de l'architecture interne du système, en présentant les différents modules qui le composent. Dans ce chapitre, nous nous proposons de réaliser une analyse approfondie des différentes étapes et des éléments nécessaires pour concevoir notre système.

Objectif

Pour répondre aux exigences de notre projet, notre application doit remplir plusieurs objectifs principaux :

— Créer un environnement de déplacement en utilisant la méthode de triangulation de Delaunay, avec une configuration définissant les zones navigables et non navigables, ainsi que les points de départ et de destination.

— Cet environnement doit permettre à une entité virtuelle de naviguer en suivant un chemin planifié.

— Mettre en place la planification du chemin en appliquant l'algorithme A*.

— Assurer la navigation de l'entité virtuelle dans l'environnement ainsi créé.

— Adapter la posture de l'entité virtuelle en fonction des variations du terrain, assurant ainsi une navigation réaliste et cohérente.

— Procéder au rendu graphique en utilisant Ogre 3D.

Conception global

La conception globale représente la phase de description de l'architecture du système à réaliser, au cours de laquelle nous allons détailler les modules de notre système. Ces modules constituent des étapes complémentaires pour obtenir le résultat final, à savoir :

1. La représentation de l'environnement via la triangulation de Delaunay.
2. La planification de chemin utilisant l'algorithme A*.
3. La navigation des entités virtuelles.

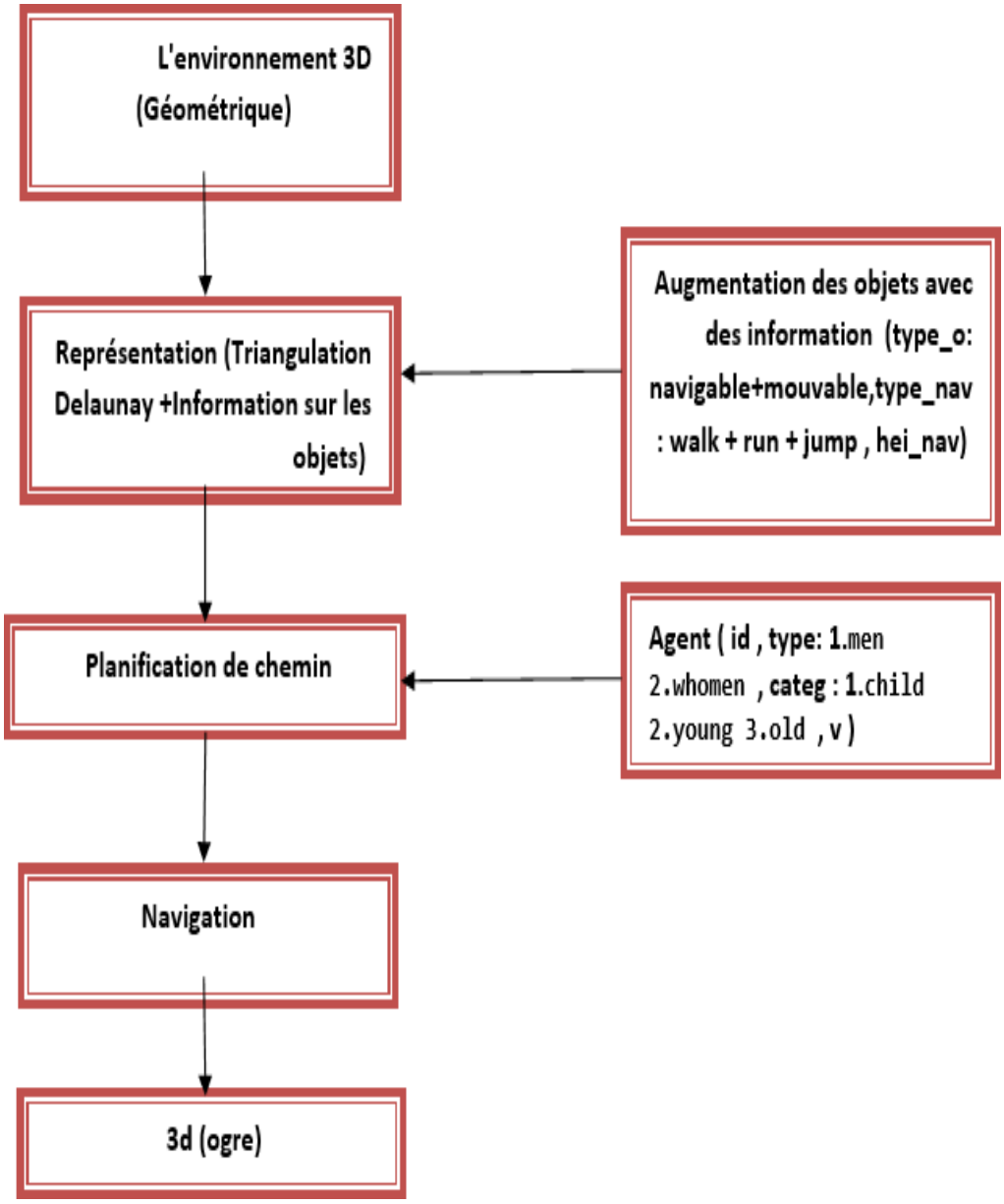


FIGURE II.1 – Schéma de la conception globale du système

La conception détaillée

La présente section se consacre à la conception détaillée du système, mettant en lumière son architecture par le biais d'une analyse approfondie des modules qui le composent. Notre travail s'articule autour de trois modules essentiels. Tout d'abord, le premier module aborde la représentation de l'environnement via la triangulation de Delaunay. Ensuite, le deuxième module concerne planification de chemin, une méthode de recherche fondamentale.Enfin, Le troisième la navigation basée sur des règles spécifiques. Cette approche

permettra une discussion approfondie de chaque module du système.

Représentation de l'environnement

La représentation de l'environnement et la planification de chemin dans notre système pour but de le raffiner et l'enrichir en précisant exactement d'où l'entité virtuel va passer dans chaque sous-espace de l'environnement.

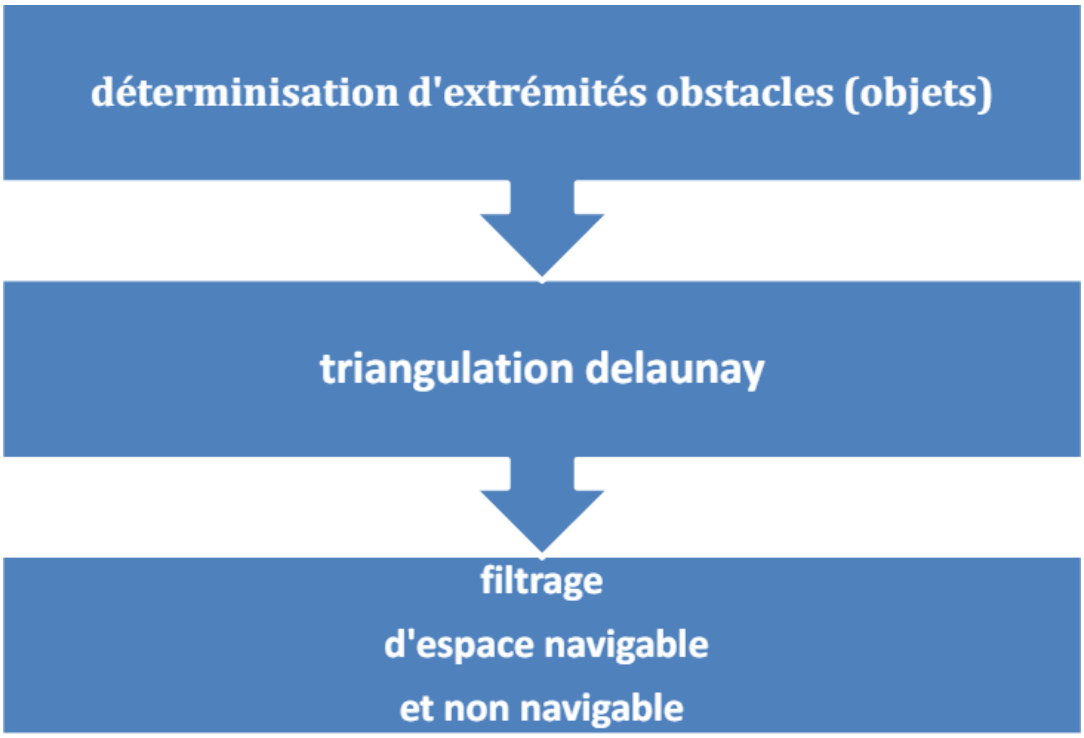


FIGURE II.2 – Schéma illustratif pour les étape effectuer pour la R.E utilisant T

Représentation de l'environnement par Triangulation de Delaunay

Il existe une multitude de méthodes théoriques et pratiques permettant de générer une triangulation à partir d'un ensemble de points, bien que toutes ne soient pas exploitables [9]. Par exemple, les triangles générés doivent être les moins allongés possible 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. Pour la première fois, le mathématicien russe Boris Delaunay (1890 – 1980) a énoncé la définition de la triangulation portant son nom en 1934 dans son œuvre intitulée « Sur la sphère vide » . La triangulation de Delaunay impose une contrainte : pour chaque triangle du maillage, son cercle circonscrit ne doit contenir aucun autre sommet de l'ensemble.

1. Propriétés

La triangulation étudiée présente des propriétés intéressantes pour la conception d'un algorithme de génération de maillage . Dans cette étude, nous nous concentrons sur les propriétés de la triangulation dans un espace bidimensionnel.

- 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.

2. Algorithmes

Pour générer un triangulations, deux principaux types d'algorithmes sont identifiables :

Les algorithmes itératifs et les algorithmes récursifs. Les deux méthodes partent d'un ensemble de points dans le plan pour produire un ensemble de triangles, constituant ainsi la triangulation. L'approche itérative commence typiquement par un unique triangle, auquel sont progressivement ajoutés de nouveaux triangles jusqu'à obtention de l'ensemble souhaité. En revanche, l'approche récursive adopte une stratégie de division et de conquête, divisant l'ensemble des points de manière dichotomique avant de fusionner les sous-ensembles obtenus.

Méthode incrémentale

Un ensemble fini de points dans le plan, cet ensemble peut être contenu dans un intervalle de $|R^2$. Par conséquent, il est possible de définir deux triangles initiaux délimitant cet intervalle et englobant l'ensemble des points (1 et 2). Ensuite, un point est choisi parmi l'ensemble initial (2). Ce point permet de localiser le triangle dans lequel il se trouve et de créer de nouveaux triangles en reliant ce point aux sommets du triangle identifié (2,3 et 4). Ce processus est répété jusqu'à ce qu'il n'y ait plus de points à sélectionner, aboutissant ainsi à la triangulation de Delaunay.

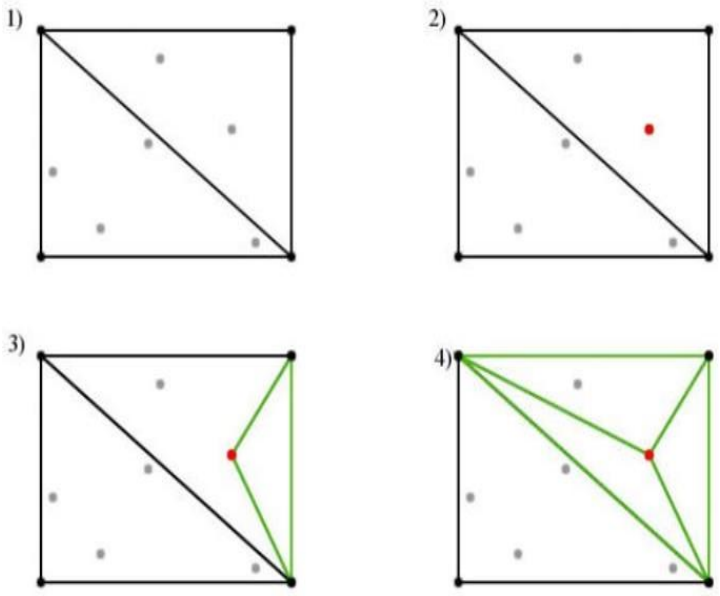


FIGURE II.3 – Première itération de l'algorithme incrémental

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).

```
add_vertex(vertex)
{
    for (each triangle)
    {
        if (vertex is inside triangle's circumscribed circle)
        {
            store triangle's edges in edgebuffer
            remove triangle
        }
    }
    remove all double edges from edgebuffer,
    keeping only unique ones
    for (each edge in edgebuffer)
    {
        form a new triangle between edge and vertex
    }
}
```

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

```
triangulate()
{
    create supertriangle and add it to the triangulation
    for (each vertex)
    {
        add_vertex(vertex)
    }
    for (each triangle)
    {
        if (one or more vertices stem from supertriangle)
        {
            remove triangle
        }
    }
}
```

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 à 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 à 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 II.4).

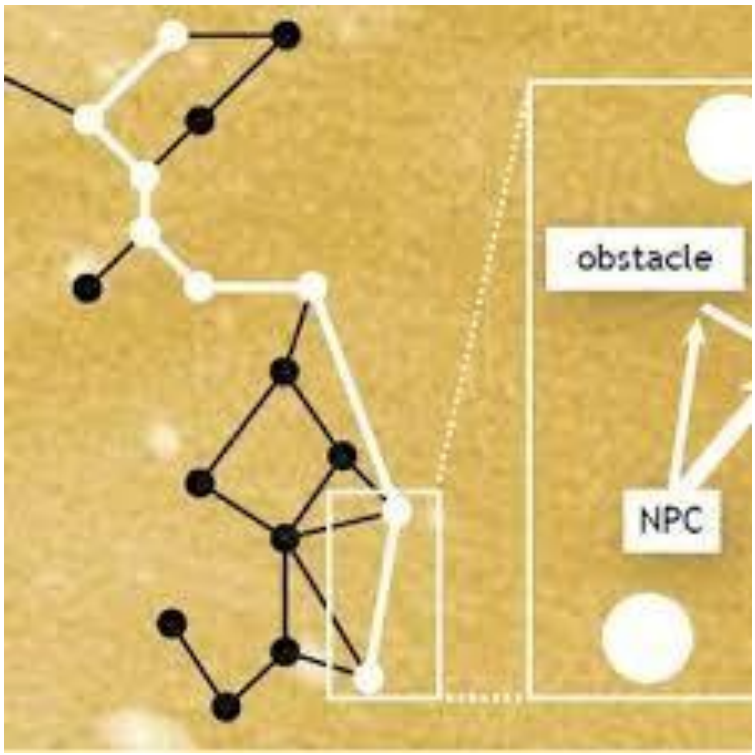


FIGURE II.4 – Exemple algorithme A* et pilotage du mouvement

Cet algorithme cherche le chemin optimal.

Algorithme A*

La figure ci-dessous présente la tâche de la planification de chemin à l'aide de l'algorithme A* :

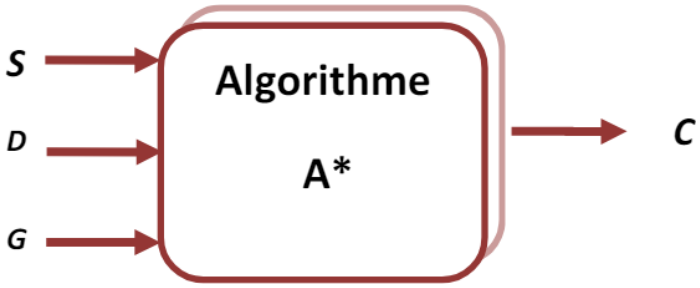


FIGURE II.5 – La planification de chemin par A*

Tel que :

S (Source) : le point de départ.

D (Destination) : le point d'arrivée.

G (Graphe) : le graphe à parcourir.

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

Algorithme A* :

Cet algorithme est conçu pour trouver le chemin optimal entre un nœud de départ et un nœud d'arrivée. Il s'agit d'une extension de l'algorithme de Dijkstra. En utilisant une évaluation heuristique qui estime la distance entre un nœud quelconque du graphe et le nœud cible, l'algorithme visite les nœuds en fonction de cette évaluation heuristique.

Cette fonction peut être déterminée de la manière suivante :

Pour un nœud N de coordonnées (X, Y), et un nœud destination D de coordonnées (X', Y')

La distance Euclidienne, dérivée du théorème de Pythagore, est définie comme suit :

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

Ensuite, on peut examiner la distance maximum, qui est déterminée par :

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

Enfin, la distance de Manhattan, également connue sous le nom de distance taxicab, se calcule par la formule suivante :

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

Chacune de ces distances présente des caractéristiques spécifiques et des applications particulières dans divers domaines mathématiques et informatiques.

Pour déterminer un chemin optimal,nous utilisons l'algorithme A* de la manière suivante :

1. Créer deux listes vides, désignées respectivement comme Ouverte et Fermée.
2. Insérer dans la liste Ouverte le nœud de départ (S).
3. Tant que la liste Ouverte ni vide ni contenant le nœud de destination :
   (a) il est primordial de chercher dans la liste Ouverte le nœud dont le coût estimé (F) est minimal.
   (b) ce nœud est ajouté à la liste Fermée, tandis que sa suppression est effectuée de la liste Ouverte.
   (c) Pour chaque nœud voisin V du nœud actuel N, il convient alors de calculer G' comme la somme du coût G de N et du coût associé au nœud adjacent V

   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).

sinon ajouter V à la liste ouverte.

4. si le nœud d'arrivée ne se trouve pas dans l'ensemble Fermée, la finalisation de l'algorithme s'opère sur un constat d'échec, signifiant qu'aucun chemin ne relie le nœud de départ au nœud d'arrivée.
5. sinon, il convient de reconstruire le cheminement en se référant à l'information parente présente dans la liste Fermée.

Navigation

La navigation constitue une étape qui suit la phase de planification du chemin. Son objectif principal est de permettre aux entités virtuelles de se déplacer dans l'environnement en exécutant des comportements spécifiques basés sur leurs capacités physiques. Au cours de cette phase, l'environnement est fidèlement représenté et le chemin est déjà planifié. Dans notre étude, la navigation repose sur les comportements adoptés par l'agent virtuel. C'est grâce à ces comportements et aux capacités physiques de l'agent que ce dernier peut se déplacer librement dans l'espace de travail. Pour que la navigation soit efficace, elle doit se fonder sur la perception afin de déterminer la prochaine action à entreprendre. Dans notre système, la navigation est basée sur le comportement. Cette figure illustre la tâche de navigation :

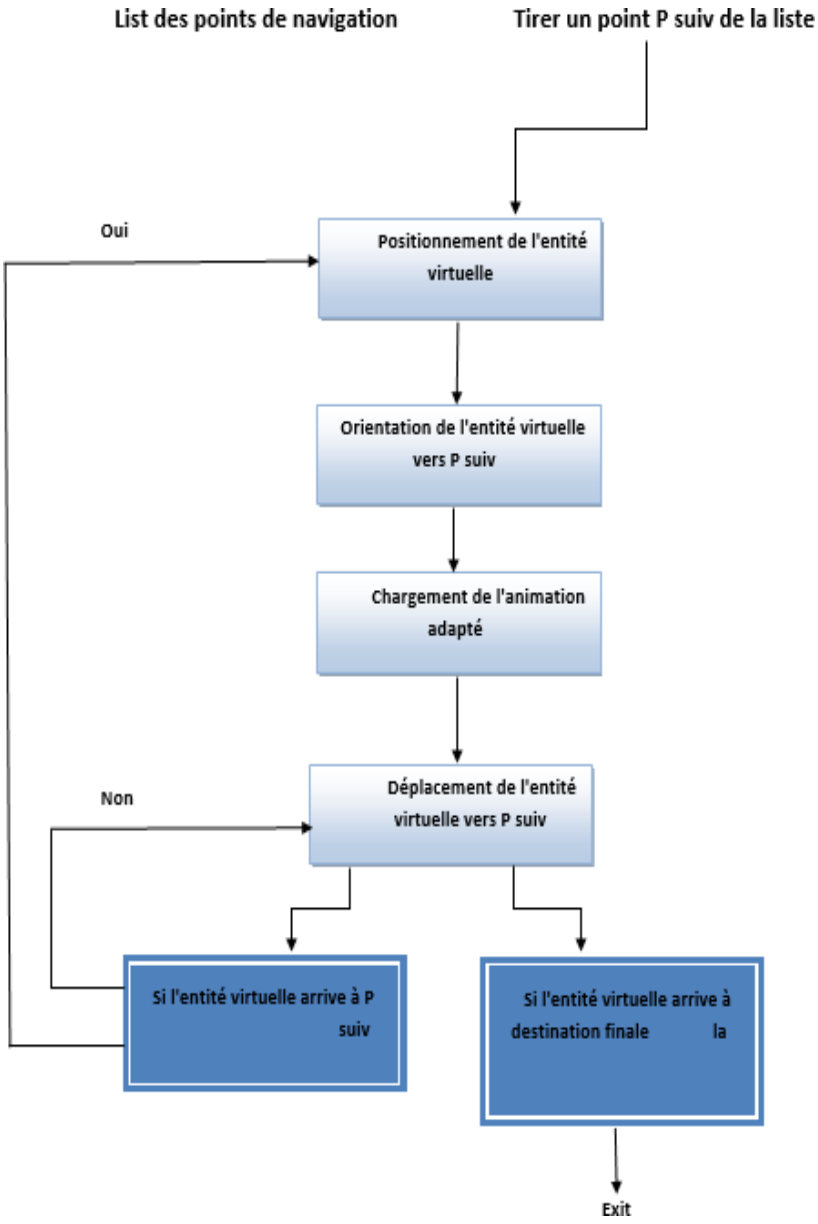


FIGURE II.6 – La tâche de navigation

— **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 D peut être calculé comme suit :

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

$$D = Xx'$$

donc

$$Dx = (X' - X) \quad \text{et} \quad Dy = (Y' - Y)$$

— **item :**

— **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.

]

Conclusion

Dans ce chapitre, nous avons élaboré une spécification détaillée de la conception de notre système, qui constitue une étape clé dans le processus de développement. La phase de conception est en effet l'une des étapes les plus critiques de ce processus, car elle permet de définir les spécifications fonctionnelles et techniques du système, ainsi que les exigences à satisfaire pour garantir sa réussite. Enfin, dans le chapitre suivant, nous présenterons la mise en œuvre de notre système, en illustrant certaines interfaces et résultats obtenus, ainsi que les structures de données et les mécanismes d'implémentation retenus pour sa réalisation.

Chapitre 3

Implémentation et résultat

Introduction

Le chapitre précédent ayant présenté la conception de notre système, nous allons maintenant décrire les étapes de sa mise en œuvre, en nous attachant notamment à l'environnement de développement, aux structures de données et aux fonctions utilisées, ainsi qu'aux résultats expérimentaux obtenus.

L'environnement de développement

Visual studio 2010

5 Après une analyse approfondie de nos besoins et exigences, il a été décidé d'adopter l'environnement de développement Visual Studio 2010 et de mettre en œuvre notre application en utilisant le langage de programmation C++. Ce choix est motivé par plusieurs facteurs déterminants :

— Le langage de programmation C++ est réputé pour sa puissance et sa flexibilité, ce qui le rend idéal pour le développement d'applications performantes et robustes.

— C++ offre des fonctionnalités avancées de programmation orientée objet, telles que les classes et les objets, permettant une conception et une structure de code optimales pour notre application.

FIGURE III.1 – Microsoft Visual studio 2010

moteur de rendu

Ogre 3D est un moteur graphique open source (Object Oriented Graphics Rendering Engine), ce qui signifie un moteur de rendu graphique orienté objets. Un moteur graphique est essentiel pour l'affichage en temps réel de formes 3D, constituant la base de toute application graphique, qu'il s'agisse de jeux vidéo, de simulateurs ou de réalité virtuelle.

FIGURE III.2 – Moteur de rendu ogre

— **Avantage**

1. Multi-plateforme : OGRE est compatible avec les systèmes d'exploitation Windows, Linux et macOS.
2. Il prend en charge à la fois OpenGL et DirectX, ce qui le rend très polyvalent. 3- Très performant : OGRE gère les effets graphiques les plus récents et offre un taux de rafraîchissement (Frame Rate) très satisfaisant.
3. Architecture professionnelle : Grâce à son design, OGRE peut s'intégrer parfaitement dans des applications graphiques professionnelles.Sa programmation orientée objet (POO) facilite le travail en équipe et sa modularité permet de nombreuses possibilités de personnalisation.
4. Documentation de qualité : OGRE propose une API de haute qualité, un manuel complet et des tutoriels bien conçus. Le forum actif constitue également une ressource précieuse pour les utilisateurs.

— **Inconvénients**

1. 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 !
2. 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

Affichage 3D

Dans le cadre de la conception d'applications 3D, l'objectif principal réside dans l'affichage d'un agencement d'objets 3D (appelés modèles) à l'écran. OGRE, en tant que moteur graphique, supporte divers formats d'objet pouvant être préalablement modélisés avec différents logiciels, qu'ils soient open source ou payants, tels que 3DStudio Max, Blender, Maya, etc. Il offre une multitude de fonctionnalités permettant de manipuler ces objets : déplacement, rotation, agrandissement, entre autres. OGRE utilise un format de fichier spécifique pour ces modèles, ainsi que pour leurs textures.

En outre, il est possible d'afficher les polygones composant un objet, ce qui s'avère particulièrement utile pour identifier facilement les erreurs dans une application.

Animation

Dans le domaine de la modélisation 3D, la distinction entre objets immobiles et objets nécessitant des mouvements est cruciale. À cet égard, OGRE (Object-Oriented Graphics Rendering Engine) propose divers types d'animations pour répondre à ces besoins variés. Les animations de type « squelettes » sont intégrées dès la phase de modélisation de l'objet. Ces mouvements sont préalablement créés par le modeleur et calculés par le logiciel de modélisation, puis exportés dans un fichier. OGRE se charge ensuite de charger ce fichier et de lancer les animations sélectionnées. En outre, il est possible de programmer des animations directement au sein d'OGRE. Il suffit de spécifier à un objet un mouvement (translation, rotation) ainsi que le temps imparti pour effectuer ce mouvement, après quoi le moteur met à jour la position de l'objet à chaque frame.

Camera

La caméra constitue un élément fondamental dans la détermination de la position de notre point de vue au sein d'une scène, orientant la direction de notre regard et définissant la distance maximale à laquelle les objets éloignés peuvent être affichés. La fonction suivante permet de créer une caméra :

```
mCamera = mScenemgr -> createCamera ("Ma_camera");
```

le déplacement de la camera se fait par la fonction **setPosition()** en déterminant.

coordonnés x,y,z :

```
mCamera->setPosition(vector3(x, y, z));
```

lookAT(), comme son nom l'indique ,permet de déterminer le point de la scène observé par notre caméra

mCamera->lookAt

(verctor3(0.0,100.0,0.0)) ;

Entité

Dans le contexte de la modélisation 3D, une entité est la représentation d'un modèle tridimensionnel, également appelé mesh. Le mesh est composé de polygones élémentaires créés dans un logiciel de modélisation, formant ainsi le modèle 3D complet. Tous ces polygones sont connectés par leurs sommets. Plus il y a de sommets, et donc de polygones, plus le mesh est précis. En général, le mesh possède également une ou plusieurs textures qui lui confèrent un aspect plus réaliste comparé à une couleur unie lors du rendu .On peut créer une entité comme suit :

```
Entity*head=mScenemgr->createEntity("Tete","ogrehead.mesh");
```

Le premier paramètre de la fonction createEntity correspond au nom de l'entité, tandis que le second paramètre désigne le nom du fichier à charger. Ce fichier, ayant l'extension «.mesh» , Et contient les modèles reconnus par Ogre.

ScèneNode

Dans le cadre de la modélisation d'une scène, un sceneNode se définit comme un objet invisible auquel il est possible d'attacher un nombre indéfini d'entités. Ces entités, une fois rattachées à ce nœud, deviennent solidaires de celui-ci et subissent les mêmes transformations. Ainsi, le sceneNode joue le rôle d'un conteneur qui conserve les informations de positionnement de chacune des entités qui lui sont associées.

Scène ménager

Scène Manager est un composant essentiel d'Ogre, chargé de l'organisation et du rendu des objets dans une scène. Il contrôle tout ce qui apparaît dans une scène. Lorsqu'un objet est placé dans la scène, la classe Scène Manager est responsable de mémoriser sa position.

Langage de développement

III.2.3.1 C++(Langage de programmation)

En tant que langage de programmation le plus couramment utilisé, notamment pour le développement d'applications, ce langage permet d'aborder divers paradigmes de programmation, tels que la programmation générique, procédurale et orientée objet. Étant un langage compilé, il traduit le code source en code objet ou binaire, permettant ainsi à la machine de l'exécuter efficacement.

Application et résultats

Dans cette partie, nous présenterons les résultats obtenus à travers une série de captures d'écran décrivant les principales étapes de l'application.

Scène initiale

Déterminer l'espace de l'environnement

FIGURE III.3 – création de l'environnement

Insertion des objet dans la scène

Ajouté des objets dans la scène et déterminé la nature de chacun (2 obstacle et 2 objet navigable).

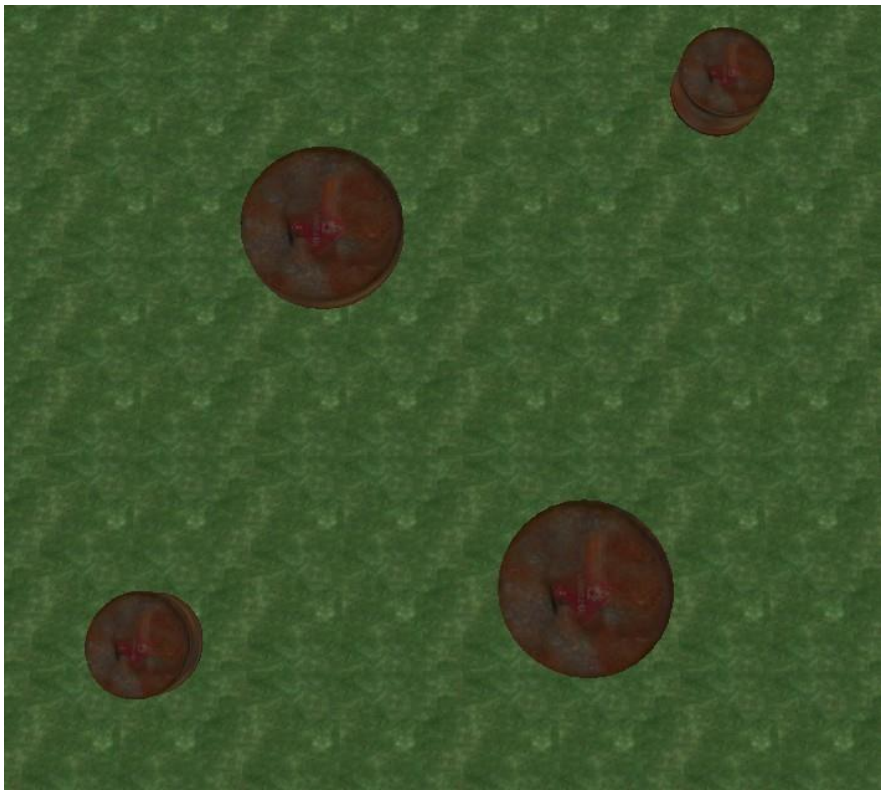


FIGURE III.4 – L'environnement d'origine

Représentation l'environnement

La triangulation de Delaunay est une méthode géométrique pour diviser un espace en triangles, avec des propriétés qui rendent ces triangles particulièrement utiles pour diverses applications. La triangulation de Delaunay est souvent utilisée pour représenter l'environnement de manière efficace et précise.

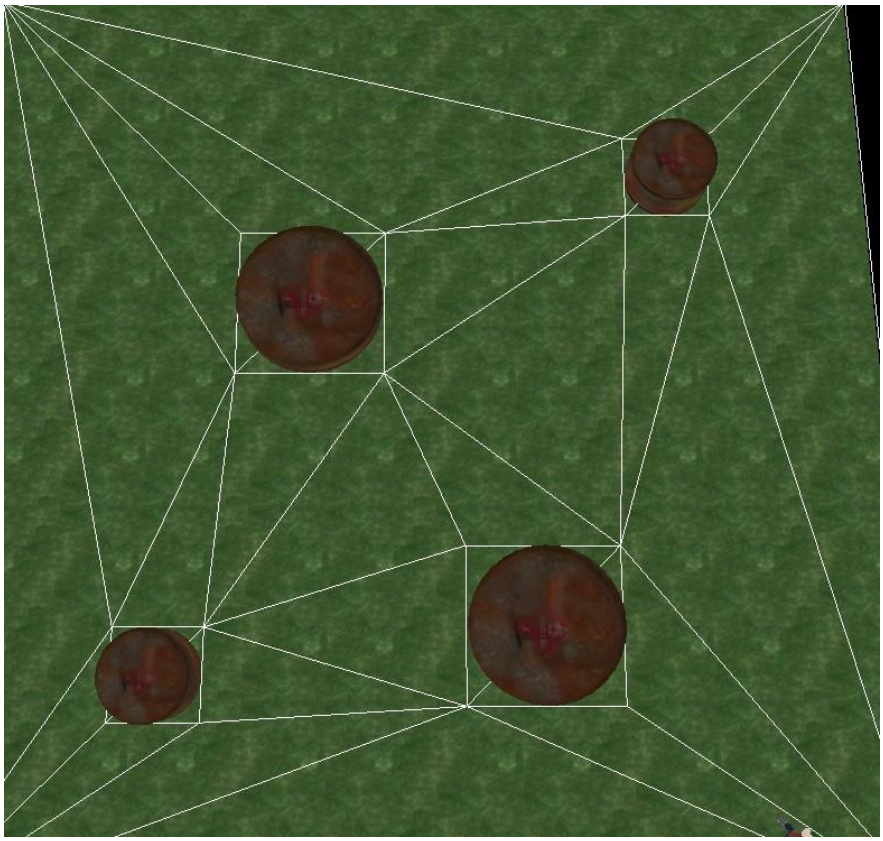


FIGURE III.5 – Représentation par la triangulation Delaunay

Le chemin planifier

L'objectif de l'algorithme de planification de chemin par A* est de trouver le chemin le plus court ou le plus optimal entre un point de départ et un point d'arrivée dans un espace donné. Cet algorithme est particulièrement utile dans les situations où il existe des obstacles ou des contraintes spécifiques qui rendent la recherche de chemin non triviale.

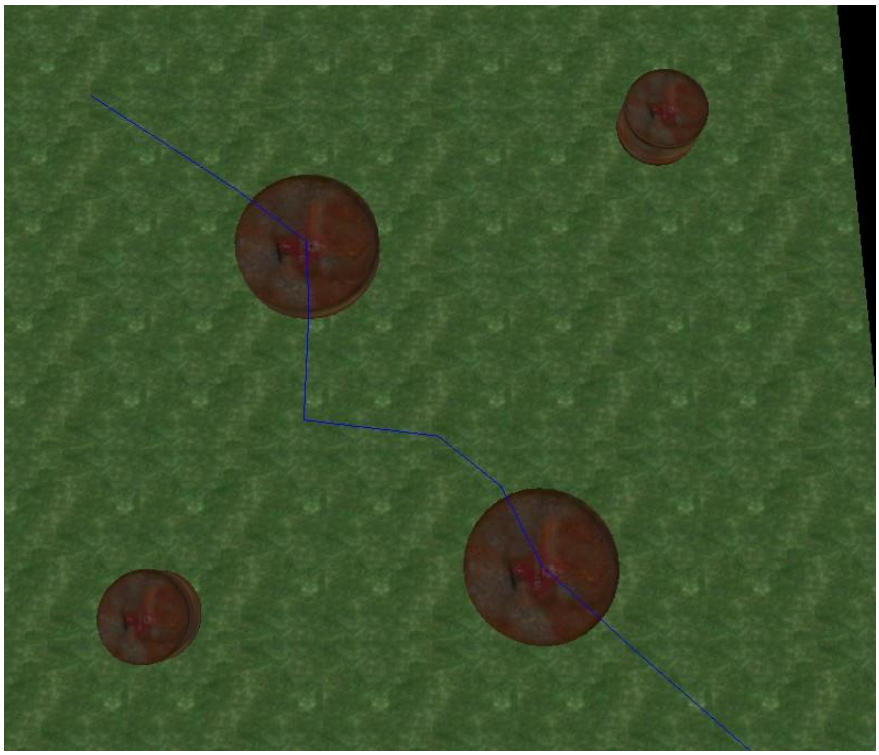


FIGURE III.6 – Le chemin planifier par A*

La navigation de l'entité virtuelle

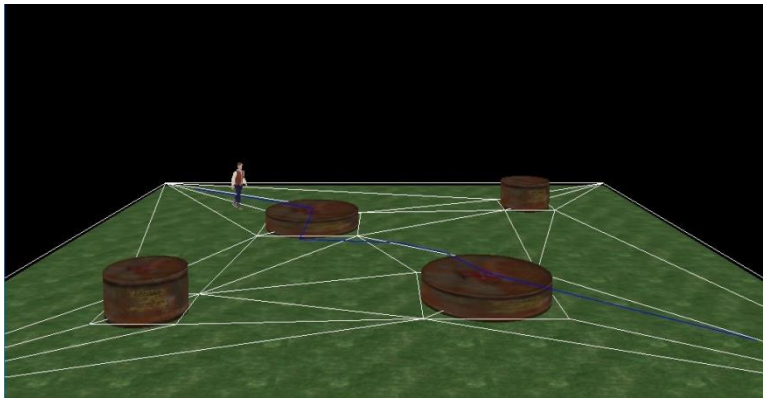


FIGURE III.7 – Navigation d'entité utilisant triangulation Delaunay et A*(a)

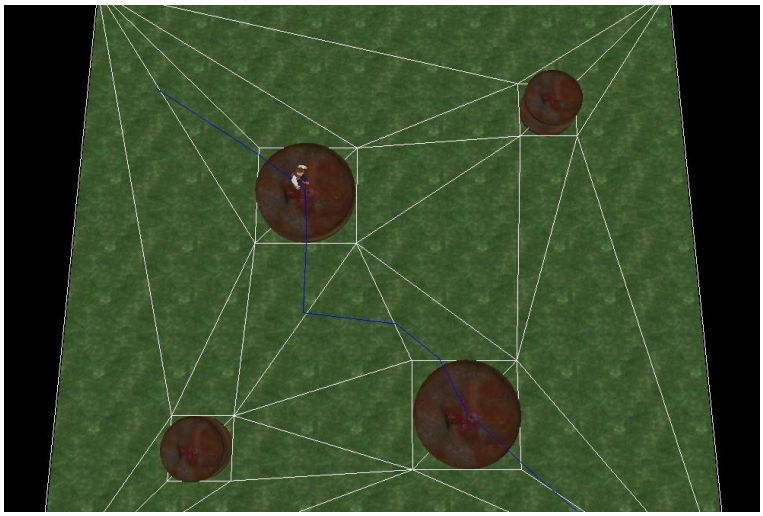


FIGURE III.8 – Navigation d'entité utilisant triangulation Delaunay et A*(b)

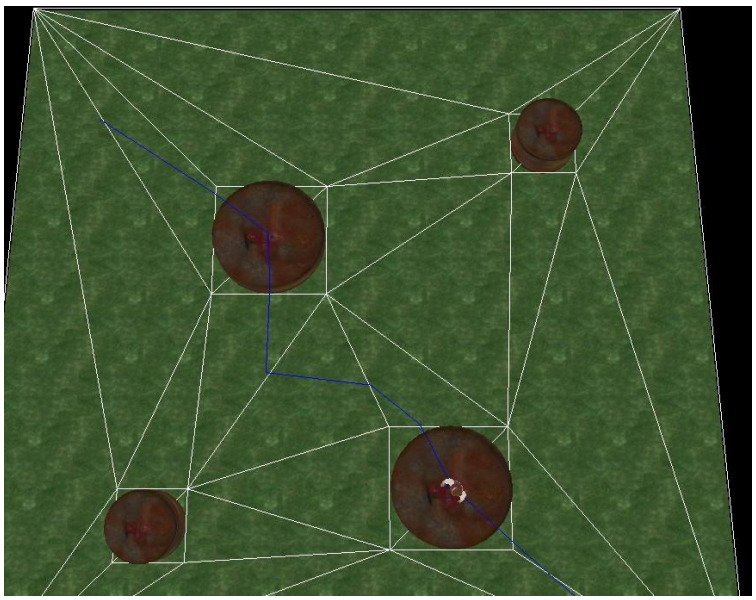


FIGURE III.9 – Navigation d'entité utilisant triangulation Delaunay et A*(c)

Conclusion

Dans ce chapitre, nous avons exposé le choix de l'environnement de développement et du langage de programmation utilisés pour mettre en œuvre notre système. Ensuite, nous avons présenté les résultats expérimentaux du système, en débutant par la configuration initiale de l'environnement de simulation, en passant par sa représentation par une triangulation, pour finir par la planification de chemin permettant à l'entité de se déplacer et de s'adapter d'une manière réaliste.

Conclusion Générale

L'adaptation de posture pour la navigation d'entités virtuelles dans un environnement 3D consiste à ajuster la posture de l'entité virtuelle en fonction de son environnement et de sa trajectoire de déplacement. Cela permet à l'entité de se déplacer de manière plus réaliste et naturelle, en prenant en compte des contraintes telles que les obstacles, les objets traversables, les pentes, ou les surfaces glissantes.

Dans ce travail, nous avons abordé le processus de planification de chemin en tenant compte de la posture de l'entité virtuelle. L'adaptation de posture peut inclure des changements dans la vitesse de déplacement, l'orientation du corps, ou même des mouvements spécifiques pour contourner des obstacles. ce qui permet une navigation plus réaliste et fluide. Nous avons également présenté les résultats illustrant l'efficacité de cette approche dans un environnement 3D. Cependant il y a encore des perspectives à explorer pour améliorer le travail.

— Optimiser les structures de données de la triangulation de Delaunay.

— Intégrer les approches de l'intelligence artificielle, en particulier dans la perception et l'identification des objets, à la solution.

Références

[1] J.D. Boissonnat and M. Yvinec. *Algorithmic Geometry*. Cambridge University Press, 1998.

[2] H. Jiang, W. Xu, T. Mao, C. Li, S. Xia, and Z. Wang. A semantic environment model for crowd simulation in multilayered complex environment. In *Proceeding of the 16th ACM Symposium on Virtual Reality Software and Technology, VRST '09*, pages 191–198, New York, NY, USA, 2009. ACM.

[3] Stéphane Donikian. *Modélisation, contrôle et animation d'agents virtuels autonomes évoluant dans des environnements informés et structurés*. PhD thesis, Université de Rennes 1, août 2004.

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

[5] A. Newell. *Unified theories of cognition*. Harvard University Press, Cambridge, MA, USA, 1994.

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

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

[8] 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.

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

[10] Thomas Lopez. *Planification de chemin et adaptation de posture en environnement dynamique*. Thèse, INSA Rennes.

[11] P. Fuchs, G. Moreau, J.-M. Burkhardt, and S. Coquillart, 2006.

[12] T. P. Caudell and D. W. Mizell, 1992.

[13] P. Milgram and F. Kishimo. A taxonomy of mixed reality. *IEICE Transactions on Information and Systems*, 77(12) :1321–1329, 1994.

[14] R. T. Azuma. A survey of augmented reality. *Presence : Teleoperators and Virtual Environments*, 6(4) :355–385, 1997.

[15] Marc Shakour. Conception et implémentation d'un algorithme de planification de chemin dans un jeu vidéo comportant en environnement triangularisé. Mémoire de maitrise en informatique, Université du Quebec à Montréal, septembre 2012.

[16] A. Bouguetitiche. Modélisation topologique et sémantique de l'environnement. Mémoire de magister en informatique, option synthèse d'image et vie artificielle, Université de Mohamed Khider Biskra, 2011.

[17] F. Lamarche. *Humanoïdes virtuels, réaction et cognition : une architecture pour leur autonomie*. Thèse de phd, Université de Rennes I, décembre 2003.

[18] Q. Avril. Peuplement automatisé de bases géométriques urbaines. Rapport de stage de master recherche, Université de Rennes I, juin 2008.

[19] S. Paris. *Caractérisation des niveaux de services et modélisation des circulations de personnes dans les lieux d'échanges*. Thèse de doctorat, Renne1, octobre 2007.

[20] F. Lamarche and S. Donikian. Crowd of virtual humans : a new approach for real time navigation in complex and structured environments. *Computer Graphics Forum*, 23 :509–518, 2004.

[21] O. Arikan, S. Chenney, and D.A. Forsyth. Efficient multi-agent path planning. In *Computer Animation and Simulation*, pages 151–162. Springer-Verlag, 2001.

[22] B. Moulin, W. Chaker, J. Perron, and P. Pelletier. Mags project : Multiagent geosimulation and crowd simulation. In W. Kuhn, M.F. Worboys, and S. Timpf, editors, *COSIT 2003*, pages 151–168. Springer-Verlag, 2003.

[23] K. E. Hoff, J. Keyser, M. Lin, D. Manocha, and T. Culver. Fast computation of generalized voronoi diagrams using graphics hardware. *Computer Graphics*, 33 :277–286, 1999.

[24] M. H. Overmars. Recent developments in motion planning. In *International Conference on Computational Science (3)*, pages 3–13, 2002.

[25] J. Barraquand and J.C. Latombe. Robot motion planning : a distributed representation approach. *International Journal of Robotics Research*, pages 628–649, 1991.

[26] Stéphane Donikian. *Modélisation, contrôle et animation d'agents virtuels autonomes évoluant dans des environnements informés et structurés*. PhD thesis, Université de Rennes 1, août 2004.

[27] F. Lamarche. *Humanoides virtuel, réaction et cognition : une architecture Pour leur autonomie*. Thèse de doctorat, Université de Rennes I, décembre 2003.

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

[29] N.J. Nilsson. *Principles of artificial intelligence*. Springer-Verlag, 1982.

[30] W. Karwowski, A.M. Genaidy, and S.S. Asfour. *Computer Aided Ergonomics*. Taylor Francis, London, 1990.

[31] N. Badler. Virtual humans, in virtual humans : Behaviors and physics, acting and reacting course notes, 1998.

[32] A.A. Shabana. *Computational Dynamics*. Wiley-Interscience, New York, 1994.

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

[34] M. Slater and M. V. Sanchez-Vives. Enhancing our lives with immersive virtual reality. *Frontiers in Robotics and AI*, 3 :74, 2016.

[35] T. Dingler, K. Kunze, and K. Wolf. The future of ubiquitous virtual reality : Challenges and research directions. In *Proceedings of the 19th International Conference on Human-Computer Interaction with Mobile Devices and Services*, pages 1–5, 2017.

[36] G. Caggianese, L. Gallo, and P. Neroni. An overview of virtual reality technologies in education : Applications, benefits, and challenges. *Future Internet*, 12(11) :208, 2020.

[37] K. H. Cheng and C. C. Tsai. A case study of immersive virtual field trips in an elementary classroom : Students' learning experience and academic achievement. *Interactive Learning Environments*, 27(7) :1014–1027, 2019.

[38] H. Paterson and S. Graham. An ergonomics approach to assessing postural adjustments in virtual reality environments. *International Journal of Industrial Ergonomics*, 66 :72–80, 2018.

[39] O. Boubaker. Motion capture technology : A survey. *Journal of Computer Science and Technology*, 28(5) :849–864, 2013.

[40] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, ..., and A. Blake. Real-time human pose recognition in parts from single depth images. In *CVPR 2011*, pages 1297–1304, 2011.

[41] C. Pacchierotti, S. Sinclair, M. Solazzi, A. Frisoli, V. Hayward, and D. Prattichizzo. Wearable haptic systems for the fingertip and the hand : Taxonomy, review, and perspectives. *IEEE Transactions on Haptics*, 10(4) :580–600, 2017.

[42] J. Lee and S. Y. Shin. A hierarchical approach to interactive motion editing for human-like figures. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 39–48, 1999.

[43] D. Holden, T. Komura, and J. Saito. Phase-functioned neural networks for character control. *ACM Transactions on Graphics (TOG)*, 35(4) :1–13, 2016.