Dédicace

Tous les mots ne sauraient exprimer la gratitude, l'amour, le respect, la reconnaissance, c'est tous simplement que je dédie ce modeste travail à :

À mon chèr père,Aucune dédicace ne saurait exprimer l'amour, l'estime, le dévouement et le respect que j'ai toujours eu pour vous. Ce travail est le fruit de tes sacrifices que tu as consentis pour mon éducation et ma formation.

À ma chère mère, Tu représentes pour moi le symbole de la bonté par excellence, la source de tendresse et l'exemple du dévouement qui n'a pas cessé de m'encourager et de prier pour moi. Ta prière et ta bénédiction m'ont été d'un grand secours pour mener à bien mes études.

A mon mari houssem, Aucun mot ne saurait t'exprimer mon profond attachement et ma reconnaissance pour l'amour la tendresse et la gentillesse dont tu m'as toujours entouré. Cher mari j'aimerai bien que tu trouves dans ce travail lexpression de mes sentiments de reconnaissance les plus sincères car grâce à ton aide et à ta patience avec moi que ce travail a pu voir le jour. Que dieu le tout puissant nous accorde un avenir meilleur.

A mes tres chers petits iyad et zineb C est à vous mes adorables anges ma joie mon tresor que maman dédie ce travail pour vous dire que vous resterez pour toujours le rayon du soleil de ma vie. Je vous aime mes bébés et je vous souhaite tout le bonheur du monde.

A ma soeur hadjer, Ta présence à mes cotés m'a toujours donné l'impression d'être proche de toute la famille. Sans toi ma vie ne serait que simple.Je voudrais t'exprimer à travers ces quelques lignes tout l'amour et toute l'affection que j'ai pour toi. Je t'aime petite sœur !

A mon frère ocba Merci d'être toujours à ma coté .par votre présence .par votre amour,pour donner du gout et du sens à notre vie de famille Je te souhaite une vie pleine de bonheur et de succès

À tous mes collègues de l'Université de biskra .

Remerciement

Tout d'abord, je remercie le Dieu, notre créateur de m'avoir donné la force, la volonté et le courage afin d'accomplir ce travail modeste.

Je veux adresser mes remerciements les plus sincères à mon encadreur madame Bouguetitche Amina qui a proposée le thème de ce mémoire, pour ses conseils et ses dirigées du début à la fin de ce travail pour l'attention et la disponibilité dont elle a su faire preuve au cours de la réalisation de ce mémoire.

Je tiens également à remercier messieurs les membres de jury pour l'honneur qu'ils m'ont fait en acceptant de siéger à notre soutenance.

Finalement, je tiens à exprimer ma profonde gratitude à ma famille qui m'a soutenue et encouragée tout au long de mes études.Ainsi que l'ensemble des enseignants qui ont contribué à ma formation.

TABLE DES MATIÈRES

TABLE DES FIGURES

LISTE DES TABLEAUX

INTRODUCTION GÉNÉRALE

Grâce aux progrès de la synthèse d'images, la modélisation de mondes en trois dimensions et l'automatisation de leur peuplement par des entités autonomes ouvrent de nombreux champs d'applications. Les réalisateurs utilisent largement ces outils pour leurs effets spéciaux, les jeux vidéo sont de plus en plus réalistes et les architectes peuvent valider la conception de leurs maquettes.

La planification de chemin est une problématique de recherche qui a été largement abordée dans le domaine de la robotique. Que ce soit pour des robots se déplaçant dans des environnements réels ou pour des agents se déplaçant dans des environnements virtuels, le fond de cette problématique reste le même. Dans les deux cas, l'entité (robot ou agent virtuel) doit construire ou posséder une représentation de l'environnement l'entourant afin de pouvoir naviguer de manière autonome au sein de cet environnement. Cette capacité de navigation est cruciale puisqu'elle va permettre à un agent d'aller à la découverte de son environnement, augmentant ainsi ses possibilités d'action avec cet environnement ainsi qu'avec les autres agents qui le peuplent.

La navigation des agents virtuels est donc un centre d'intérêt commun à l'ensemble des domaines applicatifs utilisant des environnements virtuels. Toutefois, les besoins et les contraintes imposées ne sont pas les mêmes en fonctions des applications considérées. Ainsi, ces domaines applicatifs peuvent être divisés en deux parties : les applications interactives, telles que les jeux vidéo ou encore la réalité virtuelle, et les applications de production, telles que le cinéma d'animation ou la gestion de cycles de vie des produits. Dans le cadre des applications interactives, l'accent est mis sur les temps de réponse des méthodes proposées afin qu'un utilisateur agissant avec l'environnement virtuel ait une réponse directe à ses actions. À l'inverse, dans le cadre des méthodes de production, le résultat final sera en général produit hors-ligne à l'aide de méthodes coûteuses en temps de calcul [1].

Cependant, une représentation de l'environnement sous la forme d'une géométrie 3D (triangles, polygones) n'est pas bien adaptée pour le peuplement.

La première difficulté consiste donc à structurer l'environnement pour gérer ef-

ficacement les requêtes, comme par exemple trouver un chemin ou détecter les obstacles proches. Une fois le décor planté, il s'agit de le peupler avec des entités pouvant interagir entre elles et avec l'environnement.

La gestion du déplacement d'une entité autonome au sein d'un environnement résulte de l'utilisation conjointe de plusieurs modèles. Tout d'abord, nous calculons un chemin permettant de rejoindre notre point cible en prenant en compte la topologie et les obstacles statiques, comme les marches et les poutres. Lors du déplacement, les trajectoires sont affinées pour tenir compte de certains obstacles dynamiques (par exemple une autre entité), Donc :

Le deuxième problème : Ces trajectoires ne sont que des lignes sur le sol que l'entité doit suivre au mieux. Le calcul de la position des empreintes de pas pose notamment problème dans le cas où il est nécessaire de franchir des obstacles, gravir des marches ou tourner.

Le troisième problème est comment améliorer le temps de calcul.

Ce mémoire comporte quatre chapitres organisés de la manière suivante :

le premier chapitre est consacré aux définition des jeux videos, les concepts et notions de la sumilation comportementale, ainsi que, ses domaines d'application.

Le deuxième chapitre expose les approches de représentation de L'environnement , avec une comparaison entre les représentations.

Le troisième chapitre présente la méthode proposée pour la sélection des gènes, puis, la conception du système à réaliser et son architecture globale et détaillée.

L'implémentation du système et les expérimentations et les résultats sont abordés dans le quatrième et le cinquième chapitre.

Le mémoire est terminé par une conclusion générale avec les perspectives envisagées.

CHAPITRE

1

LES JEUX VIDÉO ET LA SIMULATION COMPORTEMENTALE

1.1 Introduction

Les environnements virtuels sont depuis toujours très présents dans les jeux vidéo où le joueur doit généralement évoluer à l'aide d'un avatar le représentant. Ces mondes virtuels sont généralement peuplés d'agents virtuels autonomes avec qui le joueur va pouvoir interagir et qui vont permettre, par leurs actions, d'enrichir le jeu. Grâce aux améliorations des performances des processeurs et des cartes graphiques, ces environnements virtuels se complexifient afin d'être toujours plus riches, que ce soit par des rendus des décors toujours améliorés ou par l'intégration de plus en plus courante de moteurs de simulation physique. Afin de planifier leur chemin au sein de tels environnement, les agents virtuels vont devoir être capables de réagir rapidement aux actions de l'utilisateur ou du moteur physique afin de prendre en compte les modifications topologiques créées et de proposer des interactions riches au joueur sans que celui-ci n'ait à attendre.

1.2 Les jeux vidéos

1.2.1 Définition de jeu

On peut définir le jeu comme une activité d'ordre psychique ou bien physique pensée pour divertir et improductive à court terme. Le jeu entraîne des dépenses d'énergie et de moyens matériels, sans créer aucune richesse nouvelle. La plupart des individus qui s'y engagent n'en retirent que du plaisir, bien que certains puissent en obtenir des avantages matériels.

Le mot « jeu » vient du mot latin jocus signifiant « plaisanterie » ou « badinage », qui a aussi donné en français jouet. Par métonymie, on appelle « jeu » l'ensemble des cartes à jouer, et par extension, tout assortiment ou ensemble. En informatique, on a ainsi des jeux de données. Le « jeu » peut aussi désigner la façon personnelle de pratiquer un jeu ou de jouer d'un instrument.

1.2.2 Définition de jeu vidéo

Un jeu vidéo est un jeu électronique doté d'une interface utilisateur permettant une interaction humaine ludique en générant un retour visuel sur un dispositif vidéo. Le joueur de jeu vidéo dispose de périphériques pour agir sur le jeu et percevoir les conséquences de ses actes sur un environnement virtuel. Le mot « vidéo » dans le jeu vidéo fait traditionnellement référence à un dispositif d'affichage de trame, mais à la suite de la vulgarisation du terme il implique désormais tout type de dispositif d'affichage.

Les systèmes électroniques utilisés pour jouer à des jeux vidéo, connus sous le nom de plates-formes, peuvent être aussi bien de gros ordinateurs, que de petits appareils portables, tels la borne d'arcade, la console de jeux vidéo, l'ordinateur personnel ou encore le smartphone. Les jeux vidéo spécialisés tels que les jeux d'arcade, auparavant communs, ont vu leur usage progressivement diminuer. Le jeu vidéo est aujourd'hui considéré comme une industrie, et parfois envisagé comme une forme d'art.

1.2.2.1 Dispositif informatique

Plates-formes Différents types de systèmes sur lesquels le jeu vidéo se pratique coexistent, et de nombreux jeux sont dorénavant disponibles sur ces plates-formes. Les consoles de jeux, **les bornes d'arcade et les ordinateurs**, en sont les trois principaux vecteurs. Les plates-formes portables ont débuté avec le jeu électronique individuel sur Game and Watch, de petites consoles portables dédiées à un seul jeu, aujourd'hui pratiquement disparues, supplantées par les consoles portables. Plus récemment, les téléphones portables, et notamment les smartphones, mais aussi les tablettes tactiles, sont devenus des supports adaptés à la pratique du jeu vidéo, certains étant conçus pour répondre aux besoins des joueurs.

— **a) Les consoles de jeux vidéo**
Les consoles de jeux vidéo sont des systèmes informatiques dédiés au jeu vidéo. À la différence d'un ordinateur, une console utilise un matériel dédié, qui ne peut être que rarement amélioré. Communément, les consoles de salon se branchent sur un téléviseur et sont vendues en standard avec une manette de jeu, bien qu'il soit possible d'adjoindre d'autres périphériques voire des jeux dans des paquetages promotionnels. Les consoles portables, en plus de leur autonomie d'énergie, disposent de l'ensemble des périphériques interactifs intégrés dans le boîtier nomade. Les jeux sont développés en tenant compte des capacités de la machine, et sont ensuite mis à disposition sur le support numérique qu'elle utilise.

FIGURE 1.1: Exemple de jeu vidéo.

b) Les bornes d'arcade

Les bornes d'arcade sont des systèmes prévus pour fonctionner dans des lieux en libre accès. Une borne se compose classiquement d'un monnayeur et de périphériques robustes.

FIGURE 1.2: Exemple de borne d'arcade.

b) Les ordinateurs

Les ordinateurs sont des plates-formes informatiques hétérogènes qu'il est possible de trouver et de faire évoluer vers différentes puissances. Ils ne sont pas spécialement prévus pour jouer, mais de par leur modularité certaines configurations se prêtent aux jeux, parfois par l'adjonction de matériel dédié comme une carte graphique ou un périphérique de contrôle particulier.

Périphériques d'entrée Après les premières tentatives de périphérique de contrôle, la plupart des jeux vidéo sur console de salon et ordinateur se sont tournés respectivement vers les manettes de jeu et le duo clavier/souris qui resteront pendant longtemps les périphériques les plus utilisés.
Certains jeux peuvent également utiliser des contrôleurs dédiés, sans toutefois restreindre leur utilisation par le biais des trois contrôleurs courants, le clavier, la souris, et la manette de jeu. Par exemple, les joysticks et les volants, certains à retour de force, sont utilisés pour améliorer l'expérience de jeu des simulateurs de vols et des jeux de courses, sans être

indispensables.

Périphériques de sortie Les jeux vidéo restituent l'information par le biais de l'image et du son. L'affichage s'effectue principalement sur du matériel existant, comme la télévision pour les consoles de salon, ou les moniteurs d'ordinateur, éventuellement au moyen d'une sortie vidéo (pour affichage sur grand écran, par exemple). Le rendu sonore du jeu est retransmis via des haut-parleurs externes, ou une sortie audio vers un dispositif d'amplification externe (chaîne Hi-Fi, par exemple).

1.2.3 Type de jeux vidéo

Les jeux vidéo sont des programmes qui permettent au joueur d'interagir avec un environnement de jeu virtuel pour le divertissement et le divertissement. Il existe de nombreux types de jeux informatiques disponibles, allant des jeux de cartes traditionnels aux jeux vidéo plus avancés tels que les jeux de rôle et les jeux d'aventure.
Bien que les jeux vidéo offrent principalement du divertissement, ils améliorent également la coordination main / œil et la résolution de problèmes. Chaque jeu a sa propre stratégie, action et fantaisie qui rend chaque jeu unique et intéressant. En règle générale, nous pouvons classer les jeux informatiques dans les types suivants : jeux de cartes, jeux de société, puzzles, labyrinthe, combat, action, aventure, jeux de rôle, stratégie, sport et jeux de simulation. Cependant, la classification est un concept flou, car beaucoup de jeux sont des hybrides qui appartiennent à plus d'une classe. Par exemple, Doom peut être classé comme un jeu de labyrinthe ou un jeu d'action, tandis que Monopoly peut être classé comme un jeu de société ou un jeu de stratégie. Les différents types de jeux informatiques sont brièvement décrits comme suit :
Les systèmes électroniques utilisés pour jouer à des jeux vidéo, connus sous le nom de plates-formes, peuvent être aussi bien de gros ordinateurs, que de petits appareils portables, tels la borne d'arcade, la console de jeux vidéo, l'ordinateur personnel ou encore le smartphone. Les jeux vidéo spécialisés tels que les jeux d'arcade, auparavant communs, ont vu leur usage progressivement diminuer. Le jeu vidéo est aujourd'hui considéré comme une industrie, et parfois envisagé comme une forme d'art.

1.2.3.1 Jeux de cartes

Ce sont des versions informatisées des jeux de cartes traditionnels, ou des jeux qui ressemblent essentiellement à des jeux de cartes en ce sens qu'ils sont principalement basés sur des cartes (comme le solitaire).

1.2.3.2 Jeux de plateau

Ce sont des adaptations de jeux de société classiques. Des exemples de jeux de société incluent les échecs, les dames, le backgammon, le scrabble et le monopole.

1.2.3.3 Des puzzles

Les jeux de réflexion visent à trouver une solution, ce qui implique souvent de résoudre des énigmes, de naviguer, d'apprendre à utiliser différents outils et de manipuler ou de reconfigurer des objets. Mastermind et Tetris sont des exemples de jeux de réflexion.

1.2.3.4 Labyrinthe

Les jeux de labyrinthe nécessitent la navigation réussie d'un labyrinthe. Les labyrinthes peuvent être vus de différentes manières. Par exemple, ils peuvent apparaître dans une vue de dessus (comme dans Pac-Man) ou dans une perspective à la première personne (comme dans Doom).

1.2.3.5 Combat

Les jeux de combat impliquent des personnages qui se battent généralement au corps à corps, dans des situations de combat à deux. Les combattants sont généralement représentés par des humains ou des personnages animés. Les jeux de combat incluent Street Fighter, Avengers et Body Slam.

1.2.3.6 Action

Les jeux d'action impliquent que le joueur humain tire sur une série d'adversaires ou d'objets. Les jeux d'action traditionnels incluent Space Invaders, Asteroids, etc. Les jeux d'action populaires récents sont Doom, Quake, Descent, Half-Life et Unreal, impliquant le joueur humain à contrôler un personnage dans un environnement virtuel afin de sauver le monde des forces du mal en utilisant une force meurtrière.

1.2.3.7 Aventure

Les jeux d'aventure sont différents des jeux d'action. Ils mettent davantage l'accent sur l'histoire, l'intrigue et la résolution de casse-tête que de simplement attraper, tirer, capturer ou s'évader. Le joueur humain doit résoudre des énigmes en aventurant. Les personnages sont généralement capables de transporter des objets, tels que des armes, des clés, des outils, etc.

1.2.3.8 Simulation

Il existe deux types de jeux de simulation : la simulation de gestion et la simulation de formation. Les jeux de simulation de gestion font référence aux jeux dans lesquels les joueurs doivent gérer l'utilisation de ressources limitées pour créer ou développer un type de communauté, une institution ou un empire. Des exemples de jeux de simulation de gestion incluent Railroad Tycoon, SimAnt et SimCity. Pour les jeux de simulation

d'entraînement, il s'agit de jeux qui tentent de simuler une situation réaliste à des fins d'entraînement. Grâce à la simulation de jeu, elle aide le joueur à développer certaines aptitudes physiques, telles que la direction dans les jeux de simulation de conduite et de vol. Des exemples de jeux de simulation d'entraînement incluent Police Trainer, Gunship et Flight Unlimited.

1.2.4 Histoire du jeu vidéo

L'expression « jeu vidéo », a évolué au long de son histoire, d'une définition purement technique vers un concept général définissant un nouveau type de divertissement interactif. Techniquement, un jeu vidéo doit comporter la transmission d'un signal vidéo vers un tube cathodique affichant des images rastérisées sur un écran1. Cette définition éclipse cependant les premiers jeux sur ordinateur qui délivrent leurs résultats sur papier via des imprimantes ou des téléscripteurs, plutôt que par le biais d'affichage sur un écran. Elle écarte également tous les jeux sur écran à affichage vectoriel, ceux sur écran moderne en haute définition et la plupart de ceux fonctionnant sur consoles portables. Du point de vue technique, ces jeux n'entrant pas dans cette définition peuvent être désignés par les expressions « jeu électronique » ou « jeu sur ordinateur ».
À partir de la fin des années 2000, l'expression « jeu vidéo » perd complètement l'aspect technique de sa définition et englobe tout jeu fonctionnant via un circuit électronique logique qui comporte un aspect d'interactivité, et affiche sur un écran le résultat des actions d'un joueur. Prenant en compte cette définition plus large, les premiers jeux vidéo de l'histoire apparaissent au début des années 1950, en grande partie liés à des projets de recherches universitaires et de grandes entreprises. Cependant, ces jeux n'ont que peu d'influence les uns sur les autres à cause de leur but initial, qui consiste à réaliser la promotion de nouvelles technologies ou de matériels universitaires, plutôt que de proposer un divertissement.
L'Histoire du jeu vidéo débute dans les années 1950, où l'idée du jeu vidéo naît au sein des universités lors de recherches sur l'informatique. Les jeux vidéo ne se font connaitre du grand public qu'à partir des années 1970 avec la commercialisation des premières bornes d'arcade ainsi que les consoles de jeu vidéo, pouvant faire tourner une dizaine de jeux simplistes. Le jeu vidéo devient alors une industrie, évoluant avec la technologie et marqué par un krach en 1983, à la suite de l'inondation des modèles sur le marché après le boom des années 1970.
Les premiers jeux sur ordinateur reliés à un écran font leur apparition en 1952 via des recherches académiques : un jeu de dames par Christopher Strachey sur Ferranti Mark I et OXO, un jeu de tic-tac-toe par Alexander S. Douglas sur EDSAC. Strachey est alors mathématicien et physicien à Harrow School et tente dès 1951 de faire un jeu de dames afin de s'entraîner à la programmation sur le Pilot ACE du National Physical Laboratory. Il échoue toutefois à faire tourner le programme à cause de plusieurs erreurs de programmation. Ce n'est qu'après avoir été encouragé par Alan Turing de continuer qu'il réussira à faire marcher le jeu sur Ferranti Mark 1 en juillet 1952. La même année, Douglas développe OXO dans le cadre de son doctorat sur l'interaction homme-machine à l'université de Cambridge sur l'EDSAC. Le jeu consiste en un morpion similaire à Bertie the Brain, mais cette fois-ci sur un ordinateur non affecté à cette tâche initialement.

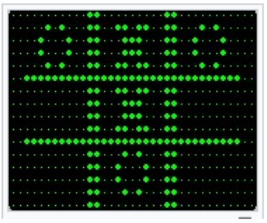

FIGURE 1.3: Reproduction d'un des premiers écrans de jeu sur ordinateur.

Ces deux jeux restent toutefois statiques : l'écran ne sert qu'à afficher un état fixe du jeu, avant que le joueur ne joue le coup suivant. En 1958, William Higinbotham crée le jeu Tennis for Two pour distraire les visiteurs du laboratoire national de Brookhaven à New York. En reliant un oscilloscope à un ordinateur, Higinbotham propose une sorte de tennis simplifié où deux adversaires doivent renvoyer une balle au-dessus d'un filet à l'aide de deux manettes.
En 1962, un groupe d'étudiants du MIT mené par Steve Russell programme un jeu nommé Spacewar ! Sur un PDP-1, le premier mini-ordinateur de la société DEC15. Le but n'était pas de créer un jeu en lui-même, mais d'expérimenter les possibilités du nouvel ordinateur

FIGURE 1.4: Spacewar.

Les années 1970 ont vu l'apparition des jeux vidéo en tant que produits commercialisés en masse, tout d'abord sous la forme de bornes d'arcade, puis de consoles de salon. La plupart des genres de jeux vidéo furent également conçus pendant cette période. En 1970 naît le jeu Darwin, renommé Core War en 1984, aux laboratoires Bell, où des bouts de programme informatique auto-reproducteurs s'affrontent... Toute une littérature cyberpunk, à commencer par Sur l'onde de choc de John Brunner (1974), naîtra de ce jeu pour programmeurs.

FIGURE 1.5: Borne d'arcade Computer Space.

Pac-Man est publié sur borne d'arcade en 1980 et devient le jeu le plus populaire du moment. En 1983, Nintendo, société japonaise qui s'est fait un nom dans les cartes à jouer et le domaine des jeux, sort la Famicom, console de jeux vidéo individuelle. Son démarrage est lent, mais les jeux proposés font augmenter la demande.
En 1992, id Software sort Wolfenstein 3D, un jeu d'action vu entièrement par les yeux du héros que l'on incarne. Il est généralement crédité pour avoir créé les mécanismes au genre du jeu de tir à la première personne, sur lesquels les jeux suivants seront basés.
En 1993, Westwood Studios sort D'une II, considéré comme le premier jeu de stratégie en temps réel (ou RTS pour Real Time Strategy en anglais) moderne, car introduisant des concepts originaux encore d'actualité.
En 1994, Sega sort la Sega Saturn, console de jeu de cinquième génération. Véritable succès au Japon, ce ne fût pas le cas aux États-Unis et en Europe, sa commercialisation est arrêtée quatre années après sa sortie.

FIGURE 1.6: Borne d'arcade Computer Space.

En 1996, Apple sort la Pipp !n, console ayant certaines caractéristiques communes avec un PC (possibilité de jouer via Internet par exemple). Ce fut un échec commercial, mais

ce principe fut repris par Microsoft avec sa Xbox.
En 1997, Origin Systems et Electronic Arts créent Ultima Online, un des premiers jeux heroic fantasy à monde persistant. L'univers du jeu continue à évoluer même lorsque le joueur n'est pas connecté.
En 1998, LucasArts renouvelle le genre en produisant Grim Fandango, un jeu d'aventure où l'affichage des lieux et des personnages est semblable à Alone in the Dark (personnages en 3D sur décors pré-calculés fixes). Toutefois, malgré quelques réussites comme The Longest Journey, le jeu d'aventure semble être sur la pente descendante.
En 2000, Sony lance la PlayStation 2. Forte de la popularité de son prédécesseur, la console se fait très vite une réputation grâce à sa puissance et sa ludothèque riche et variée comprenant de nombreuses exclusivités. Leader de sa génération, la PlayStation 2 est aussi la console de salon la plus vendue de l'histoire avec plus de 150 millions d'unités écoulées à travers le monde.
En 2001, Microsoft entre sur le marché des jeux vidéo sur console en lançant la Xbox aux États-Unis.
En 2002, la Xbox sort au Japon, puis en Europe. La GameCube sort deux mois plus tard en Europe.
Nokia sort la N-Gage en 2003, un téléphone mobile incluant des fonctionnalités de console de jeu. Nintendo sort la Game Boy Advance SP. L'entreprise 3DO dépose le bilan tandis qu'Infogrames rachète le nom de la marque Atari et Square fusionne avec Enix pour devenir Square Enix le 1er avril 2003.
L'année 2004 voit Nintendo sortir la DS au Japon et aux États-Unis. La même année, Sony sort la PSP au Japon, un échec face à la Nintendo DS malgré des caractéristiques techniques supérieures.
En 2005, Nintendo sort la DS en Europe ainsi que le successeur de la Game Boy Advance SP, la Game Boy Micro. Microsoft sort la Xbox 360 cette année-là.
La PlayStation 3 sort le 23 mars 2007 en Europe. Sony sort aussi la PSP Slim and Lite en Europe cette année-là.
En 2009, Nintendo sort la Nintendo DSi en Amérique du Nord et en Europe. Square Enix rachète Eidos et Sony annonce la PSP GO et l'officialise lors du salon de l'E3. À cette occasion, Microsoft dévoile le Projet Natal, nouveau support permettant de jouer et de commander la console Xbox 360 sans aucun périphérique, grâce au mouvement du corps et à la voix. Le développement de ce projet se voit clairement être une réponse à Nintendo et sa Wii.
En juillet 2009, après 48 heures de commercialisation au Japon, le très célèbre Dragon Quest IX a été vendu à près de 2,3 millions d'exemplaires. En août de cette même année, Sony confirme la PS3 Slim qui était alors à l'état de rumeur avancée.
En 2010, Nintendo annonce la console Nintendo 3DS et sort la Nintendo DSi XL. Sony annonce le PlayStation Move, nouveau contrôleur à détection de mouvement pour la PlayStation 3 tandis que Microsoft lance la Xbox 360 Slim, version plus fine de la Xbox 360 et utilisant les mêmes jeux. Microsoft annonce le Projet Natal renommé Kinect (interaction animée haptique humain-écran).
Nintendo sort la Nintendo 3DS en 2011 en jeu vidéo et annonce la Wii U ainsi qu'un accessoire pour la Nintendo 3DS permettant de jouer avec 2 sticks. Sony sort cette année la PlayStation Vita, première console portable avec 2 sticks. Nintendo baisse le prix de la Nintendo 3DS.

Fin juillet 2012 Nintendo sort la Nintendo 3DS XL et la Wii U quelques jours avant Noël 2012. Cette même année, Sony sort la PlayStation Vita et Square Enix présente un moteur graphique le Luminous Engine avec sa démo technique Agni's Philosophy, un avant-gout des capacités des consoles nouvelle génération. Call of Duty : Black Ops II marque un record avec 1 milliard de dollars de chiffre d'affaires en 15 jours. Sony sort la PlayStation 3, version slim, plus petite que les anciennes versions.
Fin 2013 sortent les consoles de Sony et Microsoft, respectivement la PlayStation 4 et la Xbox One. Avant même leur adaptation sur ces consoles, le jeu Grand Theft Auto V dépasse le milliard de dollars de recettes 3 jours après sa sortie le 17 septembre 2013, sur les plates-formes Xbox 360 et PlayStation 3.
En 2014, la saison 3 du championnat de League Of Legends bat un nouveau record avec 32 millions de spectateurs au total.
La PS4 Pro sort en 2016 puis la Nintendo Switch et la Xbox One X en 2017.

1.3 La simulation comportementale

Les méthodes de simulation comportementale peuvent être appliquées dans tous les domaines qui font face 'à la problématique de génération d'agents autonomes capables de montrer des comportements pertinents dans un contexte donné, réalistes et intelligents. Un de ces domaines est l'industrie des jeux vidéo où, depuis l'apparition de processeur graphiques de plus en plus puissants et où, du fait de la nature compétitive propre au secteur, l'inclusion des ennemis, des collègues ou en général des personnages capables de montrer des actions et comportements plus intelligents et en rapport avec les actions du joueur est devenu un facteur clé dans le succès ou l'échec des nouveaux jeux.
Le but final est toujours de développer ou d'utiliser une méthode de simulation de comportements capable de produire des agents crédibles et aussi amusants que possible.

1.3.1 Définition de la simulation comportementale

La simulation comportementale consiste à tenter de faire reproduire par des entités virtuelles les mécanismes décisionnels qui mènent à l'action, en prenant en compte un objectif et les contraintes imposées par leur environnement.

La simulation

Représentation du comportement d'un processus physique, industriel, biologique, économique ou militaire au moyen d'un modèle plus simple dont les paramètres et les variables sont les images de ceux du processus étudié [1].

FIGURE 1.7: City car simulator.

FIGURE 1.8: Le film d'animation « Mulan » De Disney.

Méthode de mesure et d'étude consistant à remplacer un phénomène, un système par un modèle plus simple mais ayant un comportement analogue. Le système ou phénomène analysé peut être schématisé sous forme d'un modèle mécanique, électronique ou logico-mathématique. On s'intéresse ici uniquement à la représentation du système sous la forme d'un modèle informatisable. L'objectif d'un modèle de simulation peut être :

— Simplement descriptif : pour étudier le comportement d'un système sous différentes hypothèses d'évolution de l'environnement.

— Normatif (décisionnel) : en simulant plusieurs décisions envisagées afin de choisir la meilleure ou la moins mauvaise.

1.3.2 L'animation comportementale

Le comportement est ce qui caractérise l'activité d'un individu ou d'un animal à l'intérieur d'un environnement, et se traduit par un certain nombre d'interactions. Cela souligne les deux concepts importants sous-jacents à la simulation comportementale : l'être vivant simulé est donc doté de ses propres moyens de perception et d'action, qui

sont reliés par une composante décisionnelle. Son état interne et l'état perçu de l'environnement déterminent directement ou indirectement le comportement qu'il adopte. Ces concepts permettent de définir l'autonome d'une entité au même titre que pour un être vivant : il s'agit de sa capacité à décider seule de ses actions, en fonction d'une situation perçue et éventuellement de l'état de l'état de ses connaissances sur le monde.
L'animation comportementale fait allusion aux différentes méthodes dont l'informatique se sert pour modéliser les comportements des entités qui peuplent un monde virtuel et qui ont comme caractéristiques communes la capacité de percevoir ce monde ainsi qu'un certain niveau d'autonomie. En général, toutes les approches proposées par l'animation comportementale ont comme finalité de sélectionner la meilleure action à réaliser dans une certaine situation parmi l'ensemble de toutes les actions possibles. Si on est capable de bien réaliser cette sélection, on peut modéliser toute sorte d'individus vivants (plantes, animaux et êtres humains) [2].

1.3.3 Domaines d'application

Validation ergonomique de sites : La création de lieux publics pose des problèmes d'ordre ergonomique, autrement dit, relatifs à la qualité de leur utilisation. Des problèmes peuvent se poser quant à la bonne navigation à l'intérieur des lieux, la lisibilité des divers panneaux de direction, ou lors de situation de panique. L'animation comportementale, au travers de modèles se basant sur l'analyse du comportement humain, peut être utilisée pour effectuer des validations sur des maquettes virtuelles. Les éventuels problèmes peuvent alors être détectés et corrigés avant la construction des divers aménagements. Dans ce cadre, son utilisation offre des atouts qui sont de l'ordre de la sécurité, de l'ergonomie mais aussi d'ordre économique.
Robotique : Un robot est une machine équipée de capacités de perception, de décision et d'action, qui lui permettent d'agir de manière autonome dans son environnement en fonction de la perception. La simulation d'un robot consiste à déterminer une suite de cordonnées que ce dernier doit suivre pour atteindre une destination donnée. La simulation réactive est l'ensemble des comportements fonctionnant, en parallèle, qui contrôle le robot. L'animation supprime les problèmes dus aux déférences entre la réalité et le modèle de l'environnement du robot.

FIGURE 1.9: Robot de la NASA.

Médecine : La simulation comportementale en santé correspond « à l'utilisation d'un matériel (comme un mannequin ou un simulateur procédural), de la réalité virtuelle ou d'un patient standardisé, pour reproduire des situations ou des environnements de soins, pour enseigner des procédures diagnostiques et thérapeutiques et permettre de répéter des processus, des situations cliniques ou des prises de décision par un professionnel de santé ou une équipe de professionnels.

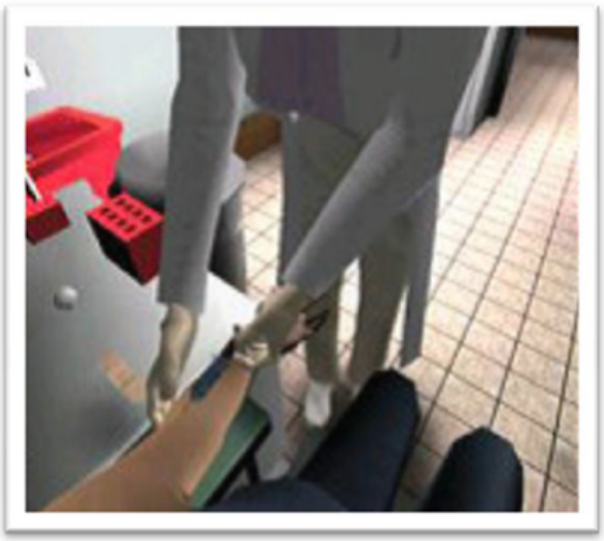

FIGURE 1.10: Un centre médical virtuel.

Les jeux vidéo : L'évolution de la puissance de calcul des ordinateurs permet aux concepteurs de jeux vidéo de concentrer sur le comportement des personnages par exemple : les jeux créateurs de la société cyber life, et Black and white, de la société Lion Head, qui proposent des mondes peuplés de joueur, qui peut cependant, dans certaines circonstances influer sur leur comportement et tenter de les faire apprendre des choses. Le développement de l'animation comportementale a fait naitre une nouvelle tendance :

la fiction interactive l'idée ici de fournir les grandes lignes d'un scénario, tout en laissant le joueur libre d'interagir à son gré avec des entités semi-autonomes. Ces entités suivent la ligne directrice du scénario mais peuvent prendre des discisions pour interagir avec le joueur. Cela donne une grande sensation de liberté, tout en pouvant générer des situations imprévues faces aux actions de l'utilisateur.

FIGURE 1.11: Lion Head studios –Black and White.

1.3.4 Les entités autonomes

L'entité autonome doit avoir un comportement plausible vis-à-vis de la situation dans laquelle il se trouve. Nous commencerons par étudier quelques travaux sur le comportement des organismes vivants ayant inspirés les architectures comportementale pour Les entités autonomes que nous présenterons par la suite.

1.3.4.1 Comportements des organismes vivants

Le comportement d'une Organisme vivant est un phénomène très complexe, des disciplines sont entièrement dédiées à son étude ; aussi nous ne présenterons ici que certaines des théories qui font références dans les travaux d'animation comportementale.

— **Entité et son environnement**

L'entité évolue dans son environnement ; il est continuellement en interaction consciente ou non avec celui-ci. Ces relations entre l'entité et le monde extérieur sont présentées sous la forme de **la boucle perception-décision-action.**

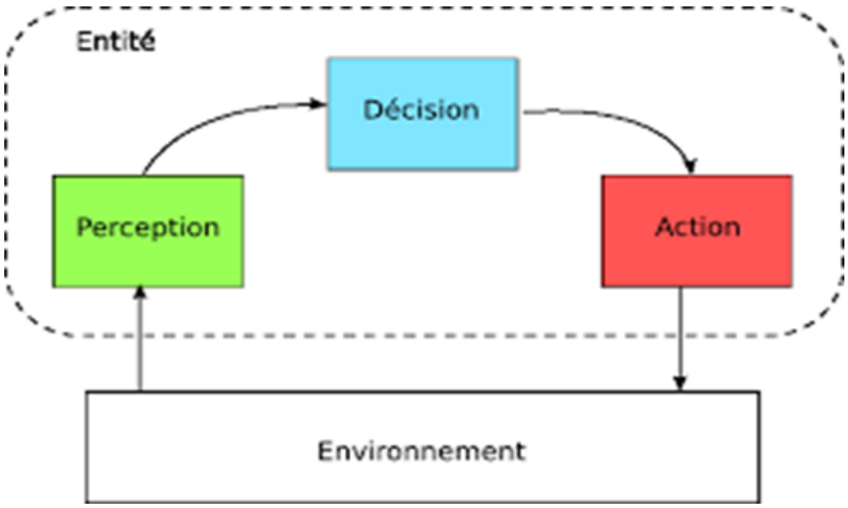

FIGURE 1.12: la boucle perception-décision-action.

L'entité dispose d'une interface avec le monde extérieur comprenant notamment des capteurs ; il s'agit des cinq sens traditionnels (vue, ouïe, odorat, toucher et goût) et des capteurs internes du corps (équilibre, proprioception...). Les informations reçues sont analysées, combinées et stockées par le centre décisionnel : l'unité de traitement. Celle-ci fonctionne de manière autonome, elle est isolée du reste du monde. La décision prise est ensuite exécutée grâce aux effecteurs. On regroupe sous ce terme les organes capables d'interagir avec l'environnement (membres, voix...). Von Uexkull se base sur cette interaction pour définir l'environnement de l'entité comme la partie du monde extérieur avec laquelle il peut naturellement interagir. L'environnement n'est donc que la partie locale à l'entité du monde extérieur, celle à portée de ses capteurs et de ses effecteurs.
La figure présente, en outre, trois boucle de retour :

— **Les interactions avec environnement :** il s'agit de la boucle de retour concernant les actions et la perception de l'environnement.

— **L'homéostasie :** elle représente la régulation interne du corps, c'est-à-dire les maintiens des constantes biologiques (la température interne par exemple).

— **Les comportements d'acquisition :** il s'agit des comportements permettant d'affiner la perception d'une partie de l'environnement.

L'entité est donc autonome dans son environnement, il dispose de moyen de se renseigner et d'agir sur celui-ci. Nous allons maintenant étudier plus en détail une vision de l'architecture décisionnelle d'une entité.

— **Un mécanisme de cognition hiérarchique**

Les travaux de Newell présentent la « machine à décider » d'une entité sous la forme de plusieurs couches organisées de manière hiérarchique. Chaque couche ajoute un niveau

d'abstraction supplémentaire et plus on monte dans la hiérarchie plus les tâches gérées sont complexes et s'étalent dans le temps. Chaque couche utilise les « fonction » de la couche précédente sans connaitre le Fonctionnement des capteurs et donner des ordres aux effecteurs. Cette vision de l'architecture décisionnelle se rapproche du modèle informatique, ce qui explique son intérêt dans le cadre de l'animation comportementale.

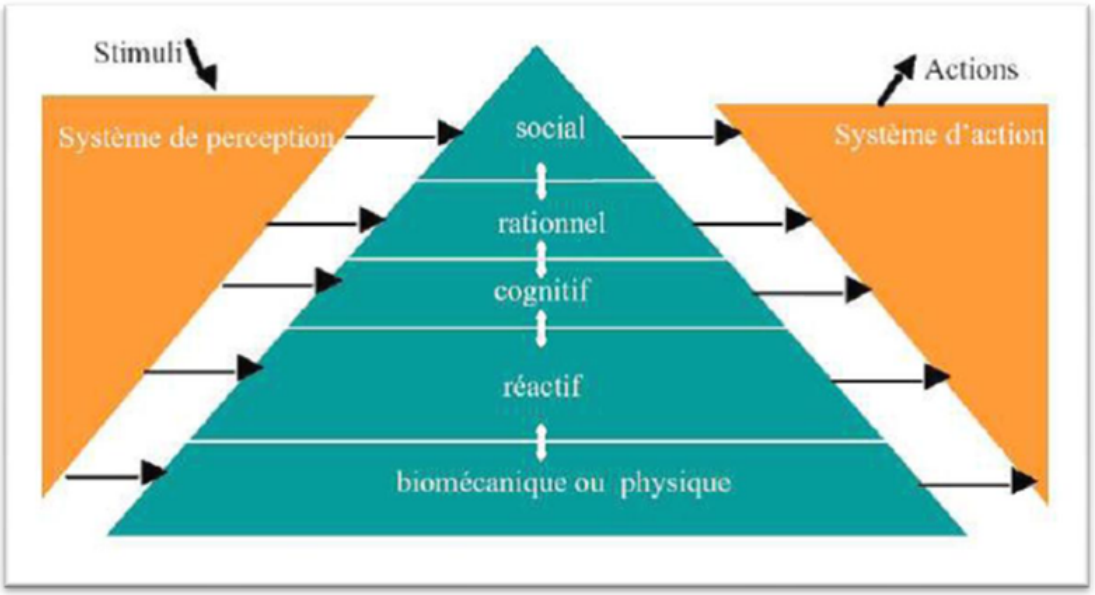

FIGURE 1.13: Architecteur prise de décision.

Newell distingue quatre domaines participant à la prise de décision ce qui va correspondre aux quatre couches de son modèle : biologique, cognitif, rationnel et social.
Elles se définissent de la manière suivante :
Couche biomécanique ou physique : La couche la plus basse de la pyramide, elle contrôle les organes du corps notamment la contraction et le relâchement des muscles. Les actions qu'elle ordonne sont très courte (quelques centième de seconde).
Couche cognitif : Cette couche gère les actions délibérées simple ou composées en introduisant la notion de choix. Elle gère les actions d'une durée de quelques dixièmes de seconde à une minute. La couche cognitif peut accéder à la connaissance et en introduit une représentation symbolique.
Couche rationnelle : Cette couche gère les taches trop longues et trop complexes pour être gérées par la couche cognitive. Elle intègre la notion de buts ainsi que la projection dans le futur. La durée des actions gérées par cette couche peut être supérieure à l'heure.
Couche sociale : C'est la couche supérieure de la pyramide, elle gère les relations entre l'humain et ses pairs. La durée de ses actions peut s'étaler sur plusieurs heures voire plusieurs jours.
Ces quatre couches agissent donc à des niveaux différents du mécanisme de prise de décision. Chaque couche de base sur les fonctionnalités de la précédentes ; les déplacements seront gérés par la couche cognitive qui fera appel à la couche biomécanique pour gérer

les mouvements des jambes. Ces couches fonctionnent aussi de manière parallèle ; on peut marcher (couche cognitive et biomécanique) tout en discutant avec son voisin (couche social et inférieures). Lors de la marche les bras ont tendance à avoir un mouvement de balancier, mais la discussion peut amener à faire des signes. Il peut donc y avoir concurrence entre plusieurs comportements pour l'accès aux effecteurs mais aussi aux capteurs.
Les avantages d'un agent autonome adaptatif : L'utilisation d'agents autonomes adaptatifs permet la simplification du travail des concepteurs, car la conception d'un monde virtuel se résumera 'a la création d'un environnement (une maison, une ville, une forêt, une usine, etc.) et au placement d'agents possédant des règles de comportement parmi lesquelles sélectionner chacune de ses actions. Cela produira des animations et simulation plus crédibles et riches pour l'utilisateur avec moins d'interventions des concepteurs. Egalement, les mondes virtuels peuplés par ce type d'agents vont profiter de l'émergence de comportements plus complexes grâces aux interactions autonomes entre l'agent et l'environnement.
L'entité autonome a pour but de ressembler à l'organisme vivant d'un point de vue extérieur, c'est-à-dire par son comportement émergent. L'animation comportementale a pour rôle de peupler les environnements virtuels à l'aide d'agents autonomes en interaction avec leur environnement local. Afin de pouvoir animer une entité, diverses méthodes ont été proposées pour simuler un comportement plus ou moins élaboré selon le but désiré. Il est possible de dissocier les comportements réactifs qui ne nécessitent pas de connaissance, des comportements cognitifs qui, eux, utilisent une abstraction de la connaissance.
Cette différence permet de classer ces modèles en deux parties : modèles réactifs et modèles cognitifs orientés but.

1.3.4.2 Modélisation de comportement

Pour pouvoir animer de manière réaliste une entité, il est nécessaire de disposer d'un modèle de comportement d'un organisme vivant réel dans son environnement, il est possible de dissocier les comportements réactifs qui ne nécessitent pas de connaissance, des comportements cognitifs qui, eux, utilisent une abstraction de la connaissance. Cette différence permet de classer ces modèles en deux parties :
Modèle de comportements réactifs : sont des modèles décrivant pour l'entité Autonome des comportements reliant directement la perception à l'action, sans modélisation abstraite des connaissances et son raisonnement, il existe trois catégories de modèle réactif qui sont :

— **Système stimuli-réponses :** une des premières modélisations inspirées de l'architecture d'un cerveau. De façon générale, la représentation s'effectue sous la forme de réseaux de nœuds interconnectés. Les nœuds en entrée du réseau propagent l'information perçue, alors que les nœuds de sortie sont connectés aux effecteurs. L'idée est donc de traduire par l'intermédiaire de ce réseau une fonction mathématique permettant de corréler directement la perception à l'action [3]

— **Système à base des règles :** Ces systèmes utilisent des capteurs pour percevoir les informations provenant de l'environnement et/ou l'état interne de l'entité virtuelle ; ces perceptions sont prises en compte pour sélectionner la prochaine ac-

tion à effectuer. Dans ces systèmes un comportement correspond à une règle sous la forme (Conditions => action), ainsi l'action à effectuer sera choisie parmi l'ensemble des actions dont les conditions sont satisfaites par les perceptions provenant des capteurs. La principale différence entre deux ou plusieurs systèmes à base de règles est le mode utilisée pour représenter ces règles [4].
En général, les représentations les plus utilisées sont :
* Un ensemble de règles simples de type (si .....alors).
* Un arbre de décision où les feuilles correspond aux actions, et les nœuds sont des experts en charge d'effectuer un choix entre tous leur sous nœuds

— **Les automates :** Cette approche modélise les comportements comme des enchainements conditionnels d'actions. Pour cela elle se sert de deux composants :
Des états qui correspondent à une tâche ou une action unitaire.
Des transitions qui correspondent aux conditions d'enchainement entre deux états, ces conditions équivalentes au contexte et au dernier état attendu. Comme l'a exprimée il existe plusieurs typologies de systèmes permettant de décrire et gérer l'exécution de comportements via des automates :
•**Les piles d'automates :** lorsqu'un automate demande l'exécution d'un autre, ce premier est empilé et le dernier est exécuté. Une fois que le dernier arrive à son état terminal, le premier est dépilé et reprend son exécution.
•**Automates parallèles :** plusieurs automates peuvent être exécutes simultanément donnant comme résultat des comportements complexes.
•**Automates parallèles hiérarchiques :** un comportement correspond à une hiérarchie d'automates qui fonctionnent de manière simultanée. Ici l'automate père peut soit superviser le fonctionnement de ses automates fils soit filtrer et/ou fusionner les résultats provenant de l'exécution de ceux-ci.

Modèle de comportements cognitif : est contrairement aux systèmes réactifs utilisent une représentation abstraite du monde (croyances, intention) afin de pouvoir planifier une suite d'action qui leur permet d'atteindre un but souhaité [5].
Situation calculs : Proposée par « McCarthy » et « P.J Hayes » en 1969, il permet de décrire des mondes complexes ainsi que les comportements (actions) des entités qui les peuplent. Pour cela, il se base sur les éléments suivants :

— Situations : états complets du monde virtuel à un instant donné.
— Les fluents : fonctions qui permettent de décrire une propriété dynamique du monde.
— Les causalités : relations cause éffets utilisées pour déduire l'état d'un fluent (propriété) en fonction de l'état d'autres fluents.
— Les actions : sous la forme pré conditions => effets elles modifient une situation.
— Les stratégies : enchainements d'actions en fonction de certains raisonnements.
— Les connaissances : elles modélisent si une entité connaît ou non l'état de certains fluents du monde.

1.4 Conclusion

permettant de générer une situation du monde qui satisfait un but donnée. Pendant cette phase tous les mondes possibles sont explorés, cela suppose un temps de recherche considérable dans les cas ou l'ensemble des actions possibles est très large. La modélisation de comportements pour les entités virtuels nécessite une étude préalable des comportements des organismes vivants que l'on cherche à imiter. En ce qui concerne la navigation, l'entité devra être dotée de comportement réactif et cognitif. Réactifs afin de s'adapter aux évolutions de l'environnement, le déplacement des autres entités par exemple. Cognitif pour être capable de planifier un chemin vers un but qu'il s'est fixé.

CHAPITRE

2

LA RECHERCHE DE CHEMIN ET REPRÉSENTATION DE L'ENVIRONNEMENT

2.1 Introduction

La recherche de chemin est utilisée pour résoudre le problème de trouver un chemin traversable dans un environnement avec des obstacles. Il s'agit d'un problème qui se pose dans de nombreux domaines d'études, notamment la robotique, le contrôle de la circulation et les jeux vidéo. Tous ces domaines reposent sur des algorithmes de recherche de parcours rapides et efficaces. Cela signifie également que le problème de recherche de chemin apparaît sous différentes formes et tailles. Les applications nécessitant une recherche de chemin donneront une priorité différente aux choses et poseront des exigences différentes pour les algorithmes. Il est donc utile d'explorer et de comparer une grande variété d'algorithmes afin de déterminer lesquels sont les meilleurs pour une situation donnée.

Les personnages doivent interagir dans des environnements complexes, dynamiques et possédant une géographie étendue, par exemple ils doivent parcourir différents types de dispositions spatiales (chambres, labyrinthes, plateformes, etc.) d'une façon intelligente et réaliste ; c'est 'a dire, en prenant des raccourcis si ils existent ou en évitant des obstacles. Un personnage capable de traverser des objets ou qui arrête son déplacement car il a trouvé un obstacle, qui est franchissable depuis la perspective de l'utilisateur, va créer une brèche dans l'illusion de comportement intelligent et adaptatif que nous essayons de produire. Il existe diverses solutions à ce problème, certaines sont proposées par les méthodes de pathfinding 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.
Les personnages doivent interagir dans des environnements complexes, dynamiques et possédant une géographie étendue, par exemple ils doivent parcourir différents types de dispositions spatiales (chambres, labyrinthes, plateformes, etc.) d'une façon intelligente et réaliste ; c'est 'a dire, en prenant des raccourcis si ils existent ou en évitant des obstacles. Un personnage capable de traverser des objets ou qui arrête son déplacement car il a trouvé un obstacle, qui est franchissable depuis la perspective de l'utilisateur, va créer une brèche dans l'illusion de comportement intelligent et adaptatif que nous essayons de produire. Il existe diverses solutions à ce problème, certaines sont proposées par les méthodes de pathfinding 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 environnements très larges et peuplés par des obstacles, des personnages non joueurs et des joueurs, la méthode la plus utilisé est l'algorithme A*.

Nous allons maintenant présenter quelques travaux concernant la navigation des entités virtuels. Ces travaux traitent des structures de données employées pour décrire l'environnement virtuel et de la manière dont l'entité va planifier son chemin dans cet espace.

2.2 Représentation de L'environnement

« Von Uexkull » se base sur l'interaction humain-environnement pour définir ce dernier comme « la partie du monde extérieure avec laquelle il peut naturellement interagir ». L'environnement n'est donc que la partie locale à l'humain du monde extérieure celle à la portée de ses capteurs et ses effecteurs. [6].
Dans le cas général, les graphistes et les architectes fournissent l'environnement sous la forme d'une base de données et trois dimensions. L'objectif est de gérer un déplacement cohérent des entités sur la base de ces informations géométriques. Pour cela, des représentations plus simples sont calculées, éliminant les informations superflues et rendant possible des calculs rapides. L'environnement est ramené à une carte en deux dimensions contenant des informations sur l'espace navigable, c'est à dire l'espace dans lequel l'humanoïde peut se déplacer. Sur la base de cette représentation, on planifie un chemin permettant à un entité d'atteindre un point défini.

L'environnement de déplacement des entités se compose de deux parties :

— Des obstacles : tous ce qui empêche d'avancer, ou qui s'oppose à la marche.
— Un espace libre : dans lequel l'humanoïde peut se déplacer.

L'environnement de déplacement des entités est une scène 3D, la structure de donnée utilisée la plupart du temps est un ensemble de points placés en trois démentions. Mais ces données brutes sont très couteuses à parcourir pour prédire les collisions et planifier un chemin. La solution est de découper et simplifier cet espace pour extraire les caractéristiques nécessaires à la gestion de la navigation des entités. La figure 2.1 représente une scène en 3D, l'image de gauche montre la représentation géométrique, l'image de droite

est obtenu après simplification de cette représentation

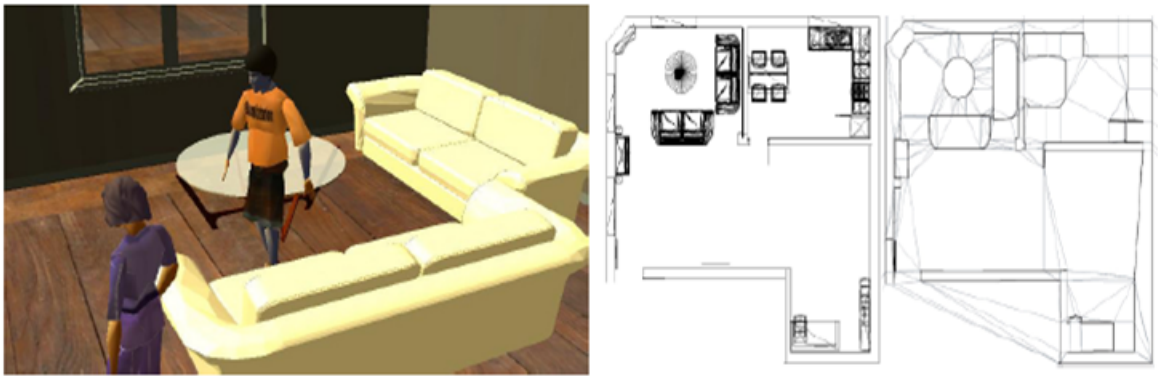

FIGURE 2.1: Exemple d'environnement 3D.

L'objectif de ces méthodes est de représenter l'espace navigable, c'est à dire l'espace dans lequel l'entité peut de déplacer. Cette notion est donc reliée à la morphologie de l'entité : sa taille, sa largeur, sa capacité à se baisser, à sauter. Un objet est un obstacle dans le cas où il est impossible de passer par-dessus ou par dessous. Beaucoup de techniques se basent sur un volume englobant de l'entité pour faciliter la détection des collisions. Ce volume correspondant à l'espace nécessaire pour tenir debout, des configurations où il est possible de faire passer l'entité sont extraites. La planification a pour objectif de déplacer le volume englobant dans l'espace libre. Il existe trois grandes familles de techniques de représentation de l'espace libre : le champ potentiel, la décomposition en cellules, et les cartes de cheminement.

2.2.1 Le champ de potentiel :

Elle a été introduite par Latombe [7], et est à l'origine une technique appliquée à la planification de mouvement en robotique. Cette méthode consiste en la construction d'une "carte" associant à chaque "point" du terrain un potentiel attractif ou répulsif pour une entité. Cette carte est générée avant le début de la simulation. Typiquement, un obstacle physique aura un potentiel maximal, un endroit sans obstacle une valeur moyenne, et un lieu étant défini comme attractif pour l'entité aura un potentiel minimal (typiquement la destination d'une entité). Le problème inhérent à cette méthode est l'existence potentielle de minimum locaux, empêchant alors l'entité de rejoindre correctement sa destination. L'idée a été reprise par Gloor et al pour la simulation de navigation de piétons dans [8]. Ici, les champs de potentiels sont calculés pour chaque point de l'environnement par la distance séparant ceux-ci d'une destination donnée.

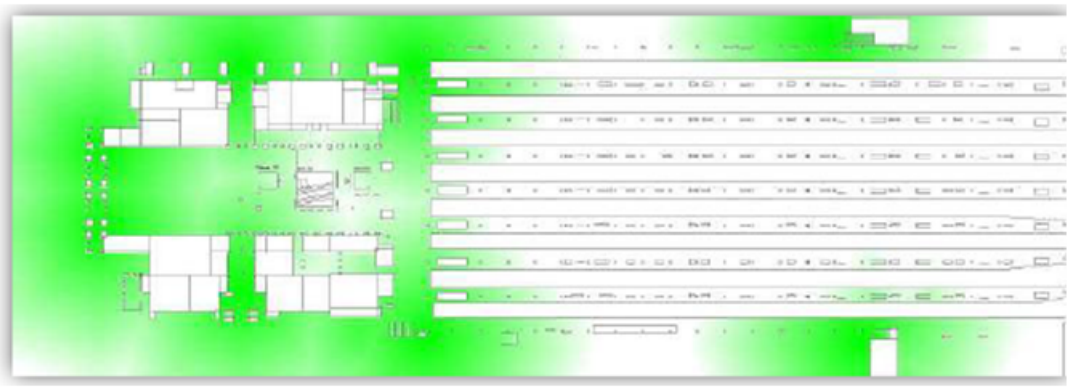

FIGURE 2.2: Représentation des champs de potentiels des sorties de la gare de Zurich.

2.2.2 Subdivision spatiale :

Il existe deux familles de subdivision spatiale :

2.2.2.1 Décompositions approchées :

il s'agit dans ce cas de découper l'environnement en cellules de forme fixée, la plupart du temps hyper cubique (carrés ou cubes). Ces cellules seront soit totalement navigables, soit totalement obstruées par un obstacle, soit partiellement obstruées, dans ce dernier cas, ils sont considérées, à la plupart du temps, comme totalement obstruées.
Il y a deux approches concernant le découpage en cellule :

Grilles de cellules régulières : ici, une cellule correspond soit à un espace sans obstacle de l'environnement, soit à un espace partiellement obstrué. Ainsi, on peut construire un graphe de connexité, chaque cellule étant représentée par un nœud de ce graphe, et chaque connexion étant un arc de celui-ci.
Grilles hiérarchiques : la grille a alors la forme d'un arbre, les cellules représentant le même type d'informations étant regroupées hiérarchiquement. Il existe plusieurs techniques, de décomposition, parmi lesquelles les quad-trees (2D). La Figure 16 représente une décomposition d'une scène (à gauche), par une grille régulière (au centre) et une grille hiérarchique de type quad-trees (à droite) :

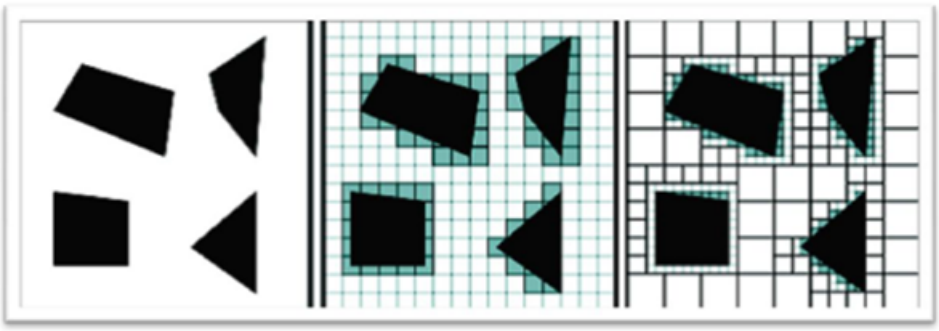

FIGURE 2.3: Décomposition d'une scène (à gauche) par une grille régulière (au centre) est une grille hiérarchique de type quad-tree (à droite).

2.2.2.2 Décomposition exacte :

une autre méthode pour la décomposition en cellule est la décomposition exacte. Elle consiste en la génération de cellules de façons à ce que l'union de celles-ci décrive exactement l'espace non contraint de l'environnement (l'espace libre). Cette décomposition est obtenue grâce à une triangulation de Delaunay [9].

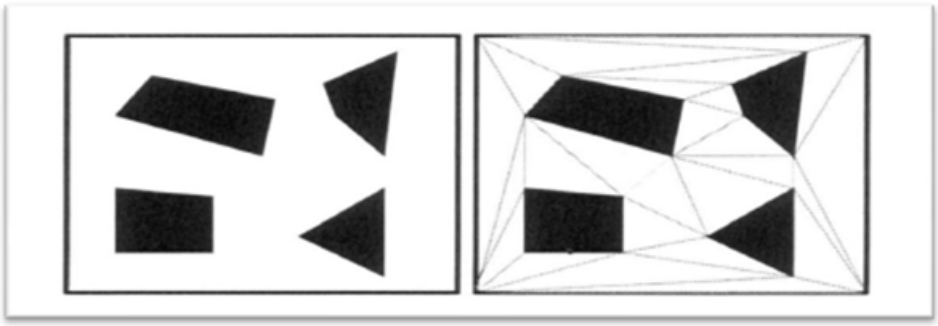

FIGURE 2.4: La décomposition exacte (triangulation de Delaunay).

Les décompositions en cellules exactes sont géométriquement plus simples mais il est nécessaire de trouver un bon compromis entre la précision et la taille mémoire. Avec la décomposition exacte, la complexité des cellules dépend directement de la géométrie de départ.

Les représentations exactes de l'environnement vont chercher à organiser les données spatiales tout en conservant intégralement les informations qu'elles contiennent à l'origine. Les méthodes de cette décomposition ont pour but de représenter exactement l'espace libre, généralement sous la forme de cellules convexes de différentes formes : triangle, polygones ou trapèzes...etc. Nous décrivons ici trois modèles pour effectuer cette subdivision : la triangulation de Delaunay, la méthode de décomposition en trapèzes et la

méthode du graphe topologique.

1. **La triangulation de Delaunay** La triangulation de Delaunay prend en entrée un ensemble P de points du plan et fournit en sortie un ensemble de triangles dont les sommets sont formés par les points fournis en entrée. Ces triangles respectent la contrainte suivante : le cercle circonscrit au triangle ne contient pas de point de P autre que les sommets de ce même triangle.
   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 sont extrait à partir les obstacles.

2. **Triangulation de Delaunay contrainte** La triangulation de Delaunay contrainte permet d'effectuer une subdivision de l'environnement en cellules triangulaire dont les bords peuvent être de deux types : contraints ou ajoutés par la triangulation. Les bords contraints expriment les arêtes des polygones délimitant les obstacles constituant l'environnement. Pour ce faire, la contrainte du cercle circonscrit à un triangle est modifiée, pour spécifier que tout point inclus dans ce cercle ne peut être relié à tous les points de ce même triangle sans entrer en intersection avec une contrainte. L'ajout de cette notion étend la propriété de plus proche voisin en permettant d'y associer une notion de visibilité. Si l'on considère que les arêtes contraintes coupent la visibilité d'un point à l'autre, la propriété du plus proche voisin devient : chaque point est relié à son plus proche voisin visible.

3. **Décomposition en trapèzes** Le principe de cette méthode est de décomposer l'environnement en cellules trapézoïdales. Elle utilise un algorithme de balayage tel que les points délimitant la géométrie de l'environnement sont triés suivant l'axe des ordonnées pour que chaque segment ait pour origine le point sélectionné et pour origine le point sélectionné et pour extrémité l'intersection avec le bord d'un obstacle. Les segments générés (maximum 2 segments) ont pour origine le point sélectionné et pour extrémité la prochaine intersection avec le bord d'un polygone délimitant un obstacle.

4. **Graphe topologique** F.Lamarche et S.Donikian proposent ce type de décomposition, afin d'optimiser l'organisation de l'espace. La première étape consiste à ramener un environnement 3D et un environnement 2D par projection sur un plan 2D. Grâce à une triangulation de Delaunay, les autres cités ci-dessus subdivisent l'espace une première fois en cellules triangulaires. Une caractéristique importante pour la navigation d'humanoïdes est la présence de goulots d'étranglement dans l'environnement ainsi que leurs positions. Pour connaitre ces emplacements, il faut calculer la distance minimale entre les coins et les murs, ils vont ainsi avoir une carte 2d de l'environnement avec une subdivision en cellules efficaces.

2.2.3 La carte de cheminement :

Une carte de cheminement est une représentation implicite de l'environnement, particulièrement bien adaptée pour la recherche de chemins. Ces techniques utilisent un graphe dont les sommets correspondent à des points clés, c'est à dire à des positions où l'humanoïde peut tenir debout sans entrer en collision avec l'environnement. Si une arrête existe entre deux sommets, il est possible de naviguer d'un point à l'autre.

2.2.3.1 Cartes de cheminement déterministes :

Dans un graphe de visibilité, les points correspondent aux coins des obstacles. Les sommets du graphe sont reliés par une arrête s'il est possible d'aller d'un point à l'autre en ligne droite, sans rencontrer d'obstacle. Cette technique a l'inconvénient de produire un grand graphe si l'environnement contient des espaces très ouverts. L'utilisation de diagramme de Voronoï constitue une autre approche intéressante. L'objectif est de partitionner l'espace grâce à des lignes équidistantes des obstacles les plus proches. Les intersections de ces lignes constituent les points clés de notre carte de cheminement. Les chemins calculés à partir de cette représentation seront assez éloignés des obstacles, contrairement à ceux obtenus à partir du graphe de visibilité. Dans le cas où l'environnement a été subdivisé en cellules, il est possible de se ramener à une carte de cheminement. Un point clé est placé sur le milieu de chaque segment libre des cellules, puis un chemin est établit entre les points clés de chaque cellule.

2.2.3.2 Carte de cheminement probabiliste :

Plutôt que de définir les points clés de manière déterministe en s'appuyant sur les caractéristiques de l'environnement, il est possible de prendre ces points au hasard dans l'espace [13]. Ils sont reliés s'il est possible d'aller de l'un à l'autre en ligne droite sans rencontrer d'obstacle. Il est nécessaire de trouver un compromis sur le nombre de points. En effet, prendre trop de points aboutirait à alourdir les calculs, alors que trop peu de points créerait des zones isolées.

FIGURE 2.5: Carte de cheminement probabiliste, et points associés à un chemin.

2.2.4 Comparaison entre les représentations

La planification de chemin signifie la capacité d'une personne pour déterminer son chemin vers une destination particulière, et être capable de reconnaître la destination une fois qu'elle est atteinte dans un environnement.
La carte de cheminement est une technique populaire pour représenter une carte Emplacements. Les cartes de cheminement sont un ensemble de points faisant référencée aux coordonnées dans l'espace physique. Il est conçu pour navigation et a été appliquée à un large éventail de des zones. La plupart des moteurs de jeu prennent en charge la recherche sur un graphique de carte de cheminement. Bien que cela fonctionne bien pour la plupart des applications 2D jeux, il faut des tonnes de points de passage pour atteindre un mouvement adéquat dans un jeu 3D en raison de la complexité de l'environnement. Ainsi, une nouvelle technique appelée Décomposition exacte (NavMesh) est créé. Décomposition exacte nécessite seulement quelques polygones pour représenter la carte. Il en résulte beaucoup plus rapidement pathfinding parce que moins de données sont examinées. Il utilise la ville de Halaa dans World of WarCraft en tant que exemple présenté à la Figure 2.6, la figure 2.7. Sur la Figure 2.6, il utilise 28 points de passage pour représenter les emplacements possibles pendant De l'autre main, comme le montre la figure 2.7, seulement 14 convexes les polygones sont utilisés. Le mouvement de la figure 2.7 agit également beaucoup plus comme un humain réel que le mouvement Figure 2.6.

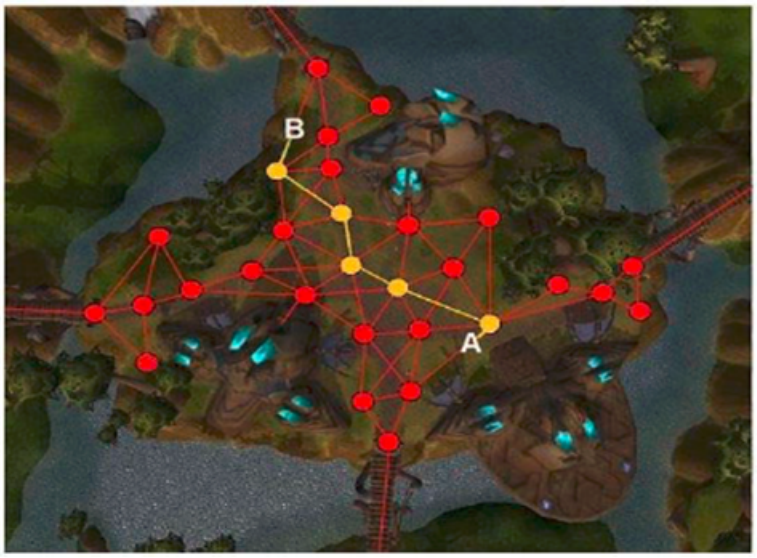

FIGURE 2.6: Navigation de A à B à l'aide d'un graphique de carte de cheminement.

FIGURE 2.7: Navigation de A à B sur Décomposition exacte.

2.3 Planification de chemin « Pathfinding »

En règle générale, la recherche de chemin consiste à déterminer un ensemble de mouvements d'un objet d'une position à une autre, sans se heurter à aucun obstacle sur son parcours. Évidemment, choisir un chemin raisonnable pour chaque objet en mouvement est souvent considéré comme la tâche d'intelligence artificielle la plus fondamentale d'un jeu commercial.
Un chemin "raisonnable" doit avoir deux propriétés. La première propriété est appelée validité, qui est la mesure la plus courante pour indiquer si le chemin est exempt de collision. La deuxième propriété est appelée optimalité. Elle est mesurée normalement par une métrique de distance ou par le temps nécessaire pour parcourir le chemin. En utilisant une métrique de distance, un chemin optimal est simplement le chemin le plus court. Cela signifie que la distance entre le début et la fin d'un tel chemin n'est pas supérieure à celle d'autres itinéraires. C'est une exigence intuitive. Par exemple, si quelqu'un se rend de Londres à Pairs, un itinéraire passant par New York ne serait pas considéré comme optimal.
Le temps est une autre mesure largement utilisée. Il définit un chemin optimal comme itinéraire le plus rapide. Cela signifie simplement que le temps requis pour un chemin optimal à parcourir est toujours inférieur à tous les autres itinéraires. Dans la plupart des cas, le chemin le plus court est souvent le plus rapide. . Cependant, il existe des cas particuliers. Par exemple, lorsque vous voyagez entre deux endroits, le temps de trajet pour suivre une route est évidemment inférieur à celui nécessaire pour conduire sur une route accidentée, même si la distance peut être plus courte. Dans le contexte des jeux commerciaux, il est préférable de mesurer l'optimalité en fonction du temps, en particulier dans les jeux de stratégie en temps réel tels que StarCraft et Age of Empires, où le temps nécessaire pour atteindre une destination est plus important que la distance parcourue.
En ce qui concerne l'efficacité d'une solution de détermination de trajectoire, elle est

généralement mesurée par le temps d'exécution et l'utilisation de la mémoire. Selon les recherches existantes, trouver un chemin presque optimal ne nécessite qu'une petite partie des ressources nécessaires pour trouver un chemin parfaitement optimal. Les jeux commerciaux imposant souvent des exigences strictes en termes de temps d'exécution et d'utilisation de la mémoire, il n'est pas toujours utile de trouver un chemin optimal. Tant que le chemin semble «raisonnable», une approche sous-optimale est généralement acceptable.

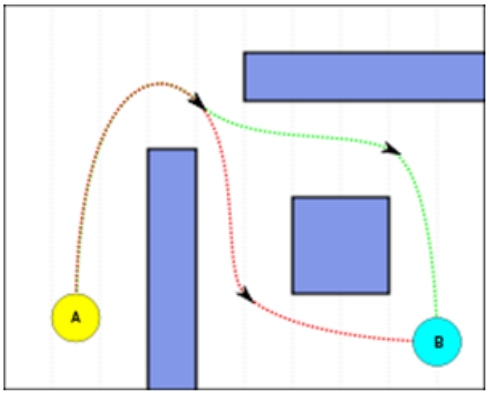

FIGURE 2.8: Exemple de pathfinding.

2.3.1 Machine d'apprentissage et Pathfinding

Pour permettre à un agent de naviguer efficacement dans un monde dynamique, il faudrait lui donner accès en temps réel. Prise de conscience de l'environnement qui l'entoure. Pour y parvenir avec les méthodes traditionnelles, il faudrait l'exécution de l'algorithme de recherche de chemin à chaque mouvement, cela coûterait cher en calcul, en particulier pour la mémoire limitée disponible pour les consoles de jeux d'aujourd'hui. Par conséquent, l'agent devra être donné une sorte de capteurs qui peuvent obtenir des informations sur ses environs. Ce n'est pas difficile à mettre en œuvre, cependant, un problème clé se pose lorsque l'agent doit déchiffrer des informations utiles à partir de ces informations. Capteurs et réagir en conséquence en temps réel sans trop solliciter les ressources des ordinateurs. Les réseaux de neurones artificiels et les algorithmes génétiques sont des techniques connue qui fournit une solution potentielle à la ce problème.

2.3.1.1 Réseaux de neurones et les algorithmes génétiques pour la recherche de chemins en temps réel

Un réseau neuronal artificiel est un système de traitement de l'information qui présente certaines caractéristiques de performance communes aux réseaux neuronaux biologiques [Fausett94]. Chaque entrée dans un neurone a une valeur de poids qui lui est associée ; ces poids constituent le principal moyen de stockage pour les réseaux de neurones. L'apprentissage se fait en modifiant la valeur des poids. Le point clé est qu'un réseau de neurones (NN)

formé doit être capable de généraliser des situations qu'il n'a jamais rencontrées. C'est une fonctionnalité particulièrement utile qui devrait considérablement aider les scènes dynamiques.
Évolution du poids d'un réseau de neurones :

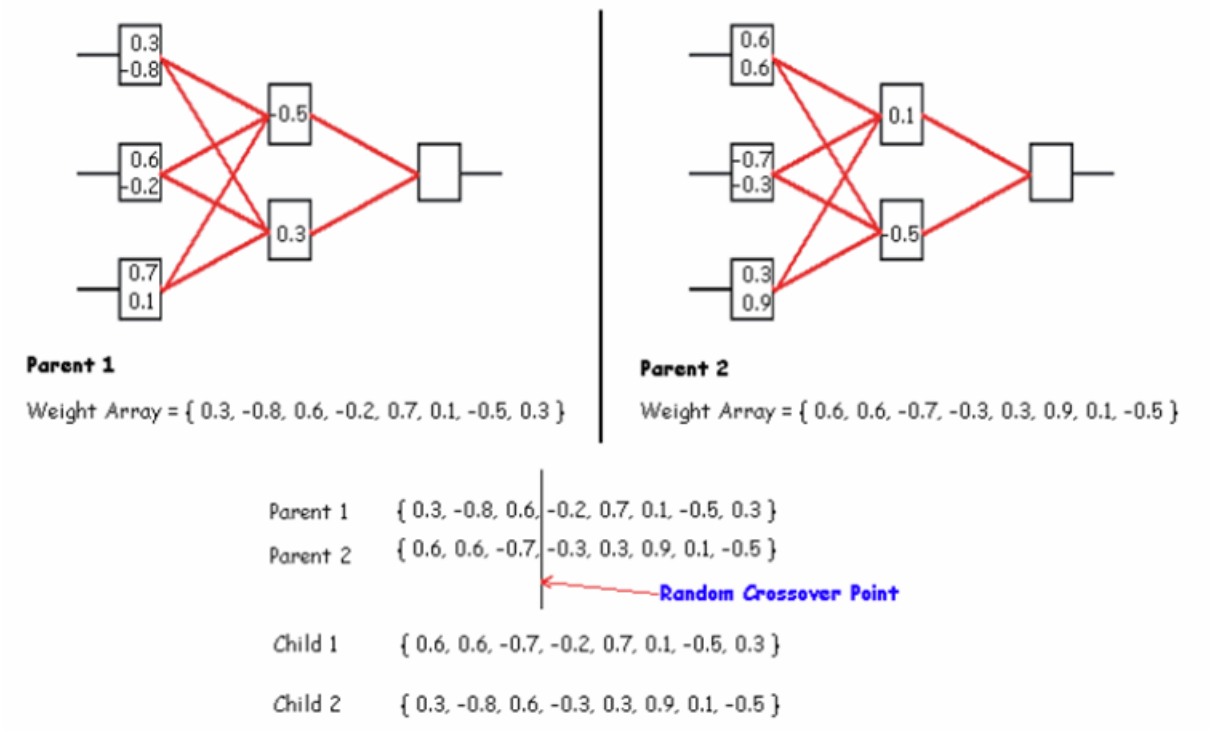

FIGURE 2.9: Exemple de real-time pathfinding.

Le codage d'un réseau de neurones devant être développé par un algorithme génétique est très simple. Ceci est réalisé en lisant tous les poids de ses couches respectives et en les stockant dans un tableau. Ce tableau de poids représente le chromosome de l'organisme, chaque poids individuel représentant un gène. Pendant le croisement, les tableaux des deux parents sont alignés côte à côte. Ensuite, en fonction de la méthode de croisement, l'algorithme génétique choisit les poids des parents respectifs à transmettre au progéniture comme le montre la figure 2.9.
Pour former le réseau de neurones à la recherche de chemin de base en temps réel, il a d'abord fallu apprendre à (1) se diriger vers direction de l'objectif, puis (2) naviguer autour des obstacles qui pourraient gâcher le chemin. Par conséquent, la première étape sera de voir si le NN peut apprendre ces deux tâches séparément et enfin apprendre les deux à la fois.

2.3.1.2 La limitation de Real-Time pathfinding

La raison pour laquelle les développeurs de jeux n'ont pas étudié le machine learning pour La découverte est qu'il faudrait beaucoup de temps pour le faire et le temps, c'est de l'argent ! Jeux Les développeurs sont également très réticents à expérimenter avec l'apprentissage automatique, imprévisible. Black 1 est un jeu qui utilise l'apprentissage automatique non supervisé. White (www.lionhead.com) où le joueur humain a pu former son propre créature. Un réseau de neurones a été utilisé pour contrôler la créature, mais tandis que l'humain joueur a été en mesure de former la créature par le renforcement de l'apprentissage, il était étroitement contrôlée pour éviter tout comportement totalement imprévisible / irréaliste. Donc, il semble que jusqu'à ce que les développeurs de jeux sont montrés la preuve que l'apprentissage automatique peut surmonter la limitations des approches standard, ils l'éviteront.

2.3.2 Les approches de pathfinding

— • La théorie des graphes
Les algorithmes de recherche de chemin peuvent être utilisés une fois que la géométrie d'un monde de jeu a été codée sous forme de carte et pré-traitée pour produire un maillage de navigation ou un ensemble de points de cheminement. Étant donné que les polygones du maillage de navigation et les points du système de points de cheminement sont reliés entre eux, ils ressemblent en quelque sorte aux points ou aux nœuds d'un graphique. Ainsi, tout l'algorithme de recherche de chemin à parcourir est transversal au graphique jusqu'à ce qu'il trouve le point final recherché. Conceptuellement, un graphe G est composé de deux ensembles et peut être écrit ainsi :
G = (V, E) où :
— V - Sommets : ensemble de points discrets dans le n-espace, mais correspondant généralement à une carte 3D.
— E – Les arêtes : ensemble de connexions entre les sommets, qui peuvent être orientés ou non.

Parallèlement à cette définition structurelle, les algorithmes de recherche de chemin doivent également connaître les propriétés de ces éléments. Par exemple, la longueur, le temps de trajet ou le coût général de chaque bord doivent être connus. (À partir de ce moment, le coût se rapportera à la distance entre deux nœuds).

— • Les Algorithmes de pathfinding
Pathfinding peut être utilisé pour répondre à la question «Comment puis-je obtenir de la source à destination ? ". Dans la plupart des cas, le chemin de la source (point actuel) à la destination (point suivant) pourrait éventuellement inclure plusieurs solutions différentes, mais si possible, la solution doit couvrir les objectifs suivants :
— Le chemin pour aller de la source A à la destination B.
— Le moyen de contourner les obstacles sur le chemin.
— Le moyen de trouver le chemin le plus court possible.
— Le moyen de trouver le chemin rapidement.

Pathfinding est une étude visant à déterminer comment aller de la source à la destination. Le problème du plus court chemin est parmi les plus étudiés dans la littérature informatique. À partir d'un graphique pondéré, le problème est de rechercher le chemin de poids total minimum dans le graphique entre les paires de nœuds. Il existe plusieurs algorithmes développés pour les variantes du problème. Les variantes incluent des bords dirigés et non dirigés. Un graphique est un nombre de nœuds et d'arcs qui les connectent, et un graphique étiqueté a une ou plusieurs descriptions attaché à chaque nœud qui distingue le nœud de tout autre nœud dans le graphique. La recherche de graphique est divisée en recherche aveugle et recherche heuristique.
La recherche à l'aveuglette est parfois appelée recherche en uniforme car elle n'a aucune connaissance son domaine. La seule option qu'une recherche aveugle est capable de faire est de distinguer un état non objectif à partir d'un état objectif. La recherche aveugle n'a pas de préférence quant à quel état (nœud) cela peut être étendu ensuite ; tandis que la recherche heuristique est l'étude des algorithmes et règles de découverte et de développement. Les heuristiques sont des règles de base qui peuvent résoudre un problème donné, mais ne garantit pas une solution. L'heuristique est la connaissance du domaine cela peut aider à guider la recherche et le raisonnement dans le domaine.
L'algorithme de recherche de chemin concerne le problème de trouver le chemin le plus court de la source à la destination et d'éviter les obstacles. Plusieurs algorithmes de recherche, y compris l'algorithme de recherche A *, djikstra, largeur d'abord, profondeur-premier, ont été créés pour résoudre le problème de plus court chemin.

2.3.2.1 Algorithme de recherche en profondeur d'abord

L'algorithme de recherche Depth-First utilise la pile Last-In-First-Out et présente un algorithme récursif. Cet algorithme approfondit l'espace de recherche à tout moment, lorsque cela est possible. Il est simple à mettre en œuvre, mais le problème majeur de cet algorithme est qu'il nécessite grande puissance de calcul pour une petite augmentation de la taille de la carte.

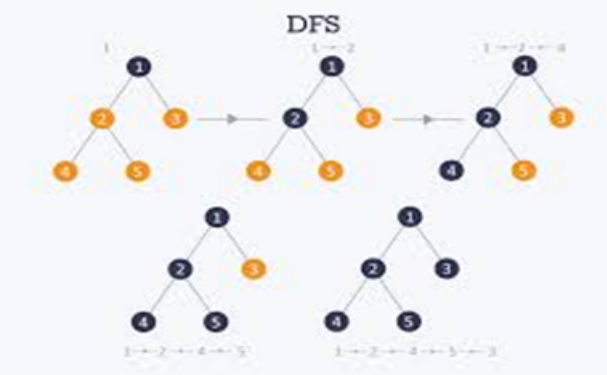

FIGURE 2.10: Algorithme de recherche en profondeur d'abord.

2.3.2.2 Algorithme de recherche de largeur d'abord

L'algorithme de recherche largeur-premier utilise la file d'attente première entrée, premier sorti. Cet algorithme implique la visite des nœuds un à la fois. Cet algorithme de recherche largeur-premier visite les nœuds dans l'ordre de leur distance au nœud source, où la distance est mesurée en tant que nombre des arêtes traversées.

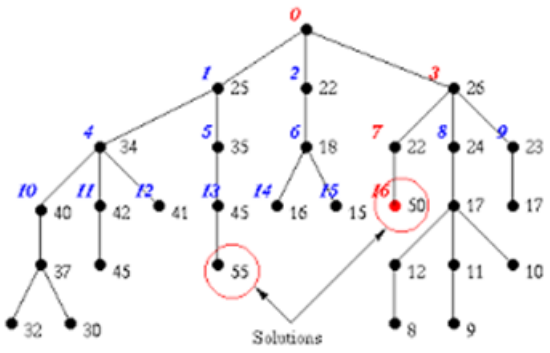

FIGURE 2.11: Algorithme de recherche de largeur d'abord

2.3.2.3 Algorithme de Dijkstra

L'algorithme de Dijkstra visite les sommets du graphe un par un en commençant par le point de départ de l'objet. Il examine ensuite le sommet le plus proche qui doit encore être examiné et ce processus se déroule dans une boucle externe qui se termine lorsque le sommet examiné se trouve être la cible ou bien si la cible n'est pas trouvée même après que tous les sommets ont été examinés. Sinon, les sommets les plus proches du sommet examiné sont ensuite ajoutés à la collection de sommets à examiner. De cette façon, il s'étend du point de départ jusqu'à atteindre l'objectif. Lorsque la cible est trouvée, la boucle se termine, puis l'algorithme revient au début en se souvenant du chemin requis. L'algorithme de Dijkstra trouve toujours le chemin le plus court du point de départ au but. Toutefois, lors de la recherche d'une cible ou d'un objectif unique, l'utilisation de cet algorithme n'est pas recommandée car il consomme du temps et des ressources supplémentaires en raison du nombre supplémentaire de nœuds que cet algorithme inspecte. D'autre part, s'il y a plusieurs cibles à rechercher, cet algorithme est l'option la plus rapide.

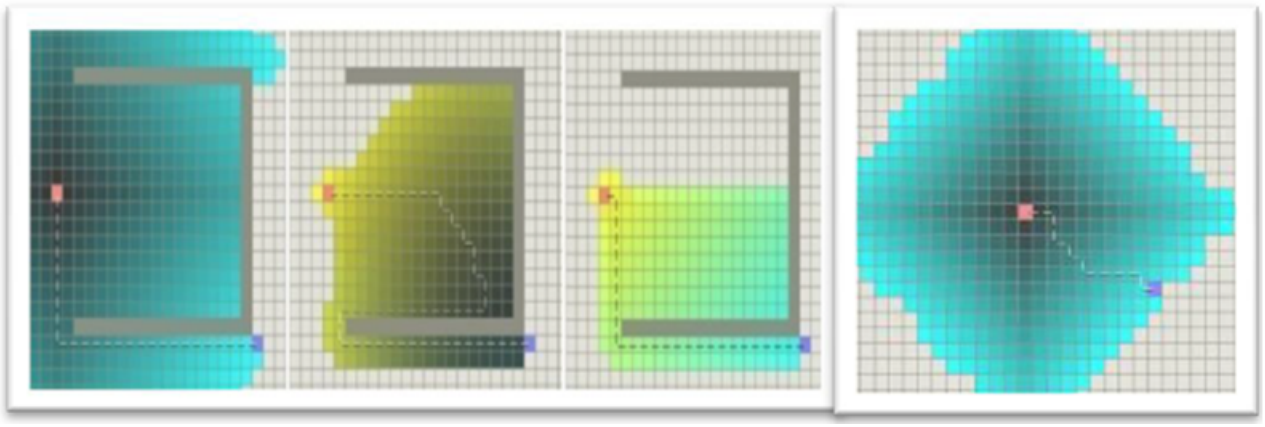

FIGURE 2.12: Le fonctionnement de l'algorithme de Djikstra.

2.3.2.4 Algorithme A *

Cet algorithme a été proposé pour la première fois par Peter E. Hart, Nils Nilsson et Bertam Raphael en 1968 [14], c'est un algorithme de recherche de chemin dans un graphe entre un nœud initial et un nœud final prédéfinis. Il est basé sur l'algorithme de Djikstra associé à une heuristique qui estime la distance entre chaque nœud pour trouver le meilleur chemin, et visite ensuite les nœuds par ordre selon cette fonction heuristique. C'est un algorithme simple et plus rapide pour trouver la solution optimal.

7	6	5	6	7	8	9	10	11		19	20	21	22
6	5	4	5	6	7	8	9	10		18	19	20	21
5	4	3	4	5	6	7	8	9		17	18	19	20
4	3	2	3	4	5	6	7	8		16	17	18	19
3	2	1	2	3	4	5	6	7		15	16	17	18
2	1	0	1	2	3	4	5	6		14	15	16	17
3	2	1	2	3	4	5	6	7		13	14	15	16
4	3	2	3	4	5	6	7	8		12	13	14	15
5	4	3	4	5	6	7	8	9	10	11	12	13	14
6	5	4	5	6	7	8	9	10	11	12	13	14	15

FIGURE 2.13: Exemple de algorithme A*.

A * est un algorithme de recherche générique qui peut être utilisé pour trouver des solutions à plusieurs problèmes, Pathfinding est essentiellement l'un d'entre eux. Cet algorithme rassemble des fonctionnalités de recherche à coût uniforme et recherche heuristique. Pour pathfinding, à nouveau l'algorithme A * et examine à nouveau le lieu non exploré le plus offrant qu'il a vu. Lorsqu'un emplacement est examiné, l'algorithme A * est terminé lorsque cet emplacement est l'objectif. Dans tous les autres cas, il est utile de noter tous les voisins d'emplacement pour une exploration supplémentaire. A * pourrait être l'algorithme de recherche de chemin le plus populaire dans le jeu AI. La complexité temporelle

de cet algorithme dépend de l'heuristique utilisée.
Algorithme A *

1. Soit P le nœud source.
2. Attribuez les valeurs g, h et f à P.
3. Ajoutez le nœud source à la liste ouverte.
4. Répétez les étapes suivantes :
   (1) Recherchez le nœud qui a le plus bas f sur la liste ouverte. Faites référence à ce nœud en tant que nœud actuel.
   (2) Basculez-le sur la liste fermée.
   (3) c) Pour chaque nœud accessible à partir du nœud actuel.

Cela fonctionne en maintenant deux listes ; la liste ouverte et la liste fermée. L'objet de la liste ouverte est de contenir les meilleurs nœuds de chemin qui n'ont pas encore été pris en compte, à commencer par le nœud de départ. Si la liste ouverte devient vide, il n'y a pas de chemin possible. La liste fermée commence vide et contient tous les nœuds qui ont déjà été considérés / visités.
La boucle principale de l'algorithme sélectionne le nœud dans la liste ouverte avec le coût estimé le plus bas pour atteindre l'objectif. Si le nœud sélectionné n'est pas l'objectif, il place tous les nœuds voisins valides dans la liste ouverte et répète le processus.
Une partie de la magie de l'algorithme réside dans le fait que tous les nœuds créés conservent une référence à leur parent. Cela signifie qu'il est possible de revenir en arrière au nœud de départ à partir de n'importe quel nœud créé par l'algorithme.
Nœud

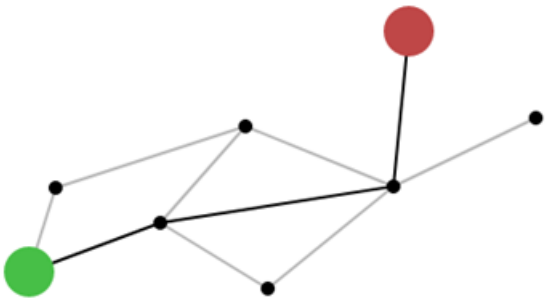

FIGURE 2.14: Représentations de chemin de A* par un graphe

Un nœud a une valeur de positionnement (par exemple x, y), une référence à son parent et trois "scores" qui lui sont associés. Ces scores indiquent comment A * détermine quels nœuds il faut prendre en compte en premier.

- **Score G :** Le score g est le score de base du nœud et représente simplement le coût incrémentiel du déplacement du nœud de départ vers ce nœud.
- **Score H :** l'heuristique : L'heuristique est une estimation facile à calculer de la distance entre chaque nœud et l'objectif. Cette facilité de calcul est très importante car le score H sera calculé au moins une fois pour chaque nœud considéré avant d'atteindre l'objectif.
- **Le score f :** est simplement l'addition des scores g et h et représente le coût total du chemin via le nœud actuel. f (n) = g (n) + h (n)

Limitations de A * A * nécessite une grande quantité de ressources de la CPU, s'il y a beaucoup de nœuds à explorer, comme c'est le cas dans les grandes cartes qui deviennent populaires dans les nouveaux jeux. Dans les programmes séquentiels, cela peut entraîner un léger retard dans la partie. Ce délai est aggravé si A * recherche des chemins pour plusieurs agents IA et / ou lorsque l'agent doit se déplacer d'un côté à l'autre de la carte. Cette perte de ressources du processeur peut provoquer le gel du jeu jusqu'à ce que le chemin optimal soit trouvé. Les concepteurs de jeux surmontent ces problèmes en peaufinant le jeu afin d'éviter ces situations.
L'inclusion d'objets dynamiques dans la carte est également un problème majeur lors de l'utilisation de A *. Par exemple, une fois qu'un chemin a été calculé, si un objet dynamique bloque le chemin, l'agent n'en aurait aucune connaissance, continuerait normalement et entrerait directement dans l'objet. Réappliquer simplement l'algorithme A * chaque fois qu'un nœud est bloqué provoquerait une surcharge excessive du processeur. Des recherches ont été menées pour étendre l'algorithme A * pour traiter ce problème, notamment l'algorithme D * (qui est l'abréviation de A * dynamique). Cela tient compte du fait que les coûts de nœud peuvent changer à mesure que l'agent se déplace sur la carte et présente une approche pour modifier les estimations de coûts en temps réel. Cependant, l'inconvénient de cette approche est qu'elle alourdit davantage le processeur et impose aux objets dynamiques une limite pouvant être introduite dans le jeu.
Un problème clé qui limite l'avancement de l'industrie des jeux est sa trop grande confiance en A * pour la recherche de parcours. Cela a amené les concepteurs de jeux à contourner les limitations dynamiques associées en peaufinant leurs conceptions plutôt qu'en développant de nouveaux concepts et approches pour résoudre les problèmes d'un environnement dynamique. Cette modification entraîne souvent la suppression / réduction du nombre d'objets dynamiques dans l'environnement et limite donc le potentiel dynamique du jeu. Une solution potentielle consiste à utiliser des réseaux de neurones ou d'autres techniques d'apprentissage automatique pour apprendre les comportements de recherche de cheminement qui pourraient s'appliquer à la recherche de cheminement en temps réel.

2.4 Conclusion

L'approche traditionnelle pour trouver un itinéraire entre deux emplacements sur un terrain de jeu consiste à modéliser ce problème en tant que problème de planification dans un modèle d'état et à appliquer une sorte d'algorithme de recherche. Chaque emplacement possible sur la carte est considéré comme un état différent avec un ensemble d'actions possibles à effectuer (déplacements vers d'autres emplacements) qui serviront à représenter la topologie du terrain.
Le graphique des états peut être exploré de l'état initial à l'état de l'objectif en utilisant différentes stratégies pour trouver la meilleure façon (ce qui réduit les coûts).
Les différentes manières d'explorer les nœuds du graphique motivent l'existence de différents algorithmes de recherche, comme recherche par la largeur d'abord, recherche par la profondeur, Dijkstra's, profondeur itérative recherche ou le préféré dans la communauté de jeu, A *. Une fois que nous avons trouvé un chemin disponible entre l'état initial et l'objectif, nous pouvons commencer à avancer. Pour surmonter le problème du déménagement entités qui pourraient interrompre notre chemin, nous pouvons mettre à jour notre plan en recalculant le chemin chaque certain le temps ou juste chaque fois le quelque chose bloque notre chemin.
Il y a des situations où A * peut ne pas fonctionner très bien, pour diverses raisons. Le plus ou moins besoins en temps réel des jeux, ainsi que les limites de la mémoire disponible et du temps de processeur certains d'entre eux peuvent rendre difficile même pour A * de bien travailler. Une grande carte peut nécessiter des milliers des entrées dans la liste ouverte et fermée, et il ne peut pas être assez de place pour cela. Même s'il y a assez de mémoire pour eux, les algorithmes utilisés pour les manipuler peuvent être inefficaces.

CHAPITRE

3

CONCEPTION

3.1 Introduction

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

3.2 Objectifs

Les principaux objectifs auxquels doit répondre notre application sont :

— • Représentation d'un environnement de déplacement par la méthode de triangulation de Delaunay, avec une configuration (espace navigable, les obstacles, points de départ, le but).
— • Planification de chemin en utilisant l'algorithme A*.
— • Améliorer le temps de calcul de plus court chemin avec l'algorithme A*.

3.3 Conception générale de notre système

Notre système est résumé par le schéma de la figure 3.1, La première étape consiste à représenter l'environnement sous forme d'un maillage triangulaire : La triangulation de Delaunay est un type de partitionnement d'un ensemble de points E positionnés dans un plan formé de triangles dont les sommets sont des objets, et qui à eux tous constituent

une partition de l'enveloppe convexe de ces objets.
La deuxième étape consiste à améliorer le temps de calcul : nous avons proposé de convertir les triangles en polygones afin de réduire la zone de mouvement, de sorte que l'homme virtuel se déplace sur un polygone plutôt que sur deux ou trois triangles. L'étape suivante est l'élaboration de la planification du chemin en appliquant l'algorithme A*.

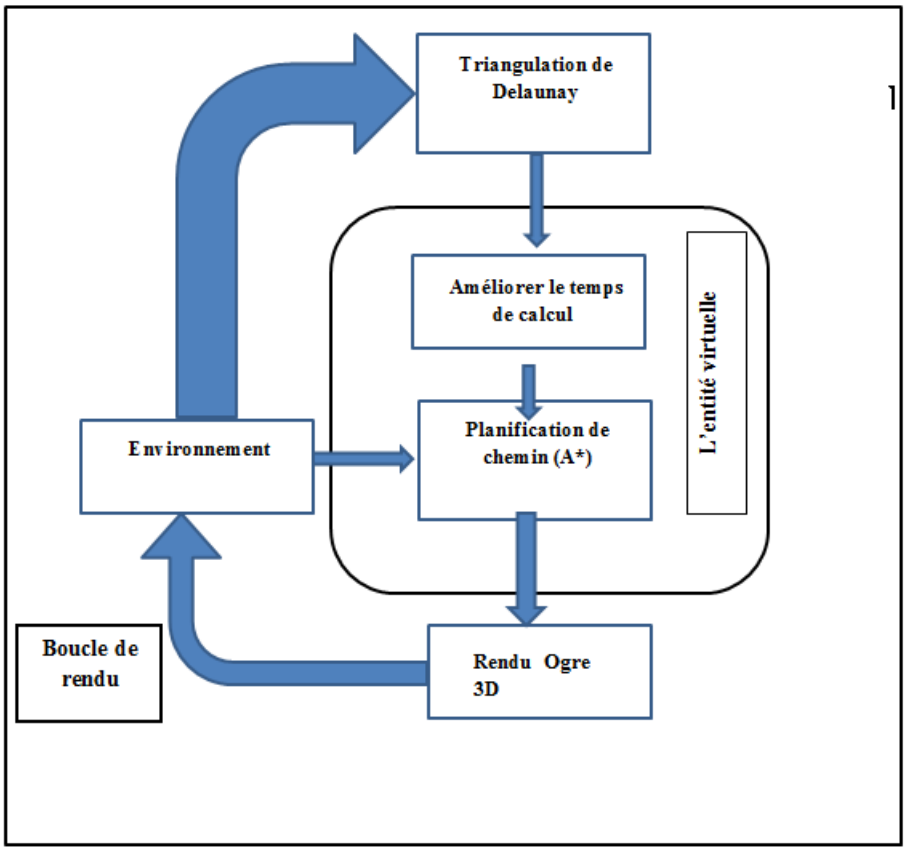

FIGURE 3.1: Conception générale de notre système.

L'entité virtuelle : Les humains virtuels assimilés à des agents qui jouent presque le même rôle (comportements humains), c'est la création de personnages dans un environnement virtuel guidés par des commandes programmés, se transforme en être humain. Deux concepts sont nécessaires pour la création d'humanoïde :

— L'apparence : Ce qui apparaît à la surface des choses, par opposition à ce qui est en profondeur, essentiellement. L'apparence est réalisée par des logiciels de modélisation.
— Le comportement : Le comportement est une activité d'individu au sein d'un environnement, et se traduit par un certain nombre d'interactions. Le comportement humain est un phénomène très complexe, celui qui nous intéresse dans notre étude est bien le comportement de recherche de chemin (le fait de se déplacer dans l'espace).

Un moteur de rendu Les moteurs de rendu 3D peuvent être des logiciels classiques ou des algorithmes intégrés dans des cartes graphiques spéciales qui calculent une ou plusieurs images 3D en y restituant non seulement la projection 3D, les textures mais surtout tous les effets d'éclairage. C'est ce que l'on appelle le rendu.

3.4 Conception détaillée de notre système

Notre système est composé des modules suivants :

3.4.1 Représentation de l'environnement

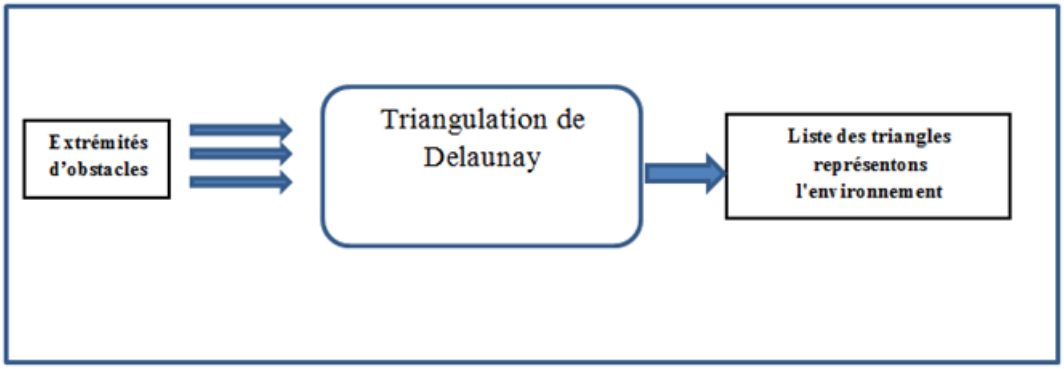

FIGURE 3.2: Processus de représentation de l'environnement.

Parmi les modèles de représentation des environnements virtuels, nous avons choisi la triangulation de Delaunay. La triangulation de Delaunay est une structure de donnée utilisée en synthèse d'image permettant de représenter les zones d'un environnement 3D. Couplé à certains algorithmes comme A*, il permet à des agents informatiques de trouver leur chemin dans l'environnement. Cette structure de données est particulièrement adaptée à la description des larges espaces.
Les avantages :

— Possibilité de trouver le chemin le plus court réel.
— Réduction du temps de calcul et de l'empreinte mémoire dans les environnements.
— Gestion plus aisée d'obstacles dynamiques.

Environnement et Triangulation de Delaunay :

3.4.1.1 Environnement et Triangulation de Delaunay :

La première étape consiste à représenter l'environnement sous forme d'un maillage triangulaire.
Soient N Points. Par définition, on appelle triangle de Delaunay un triangle qui a comme sommets trois des points, et tel que son cercle circonscrit ne pas entourer Aucun point de

tels triangles existent, et l'on démontre que :

— L'assemblage de tels triangles de Delaunay pour la triangulation des N points, appelée triangulation de Delaunay. Une telle triangulation unique, sous réserve qu'on n'ait jamais trois points alignés, ni quatre points sur le même cercle.

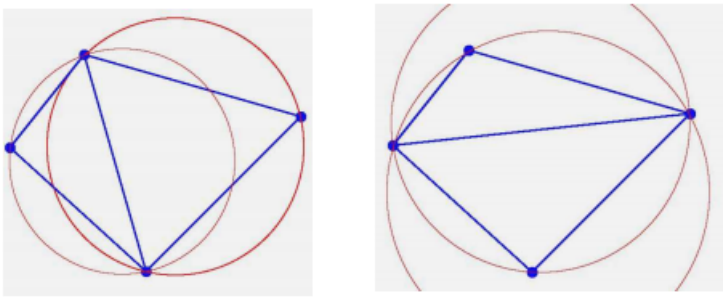

FIGURE 3.3: la figure à gouche triangulation de Delaunay, la figure à droite n'est pas.

Deux triangulations de N = 4 points. Celle de gauche est une triangulation Delaunay, celle de droite ne l'est pas (les cercles circonscrits aux triangles contiennent le quatrième site).

3.4.1.2 Exécution de l'algorithme :

Il existe un très grand nombre de méthodes théoriques et pratiques pour générer une triangulation à partir d'un ensemble de points. Mais toutes ne sont pas exploitables. En effet, on cherche à construire un ensemble de triangles qui soient arrangés de manière assez uniforme et régulière [15].
Par exemple, les triangles générés doivent être les moins allongés possibles et former un ensemble homogène. Cas contraire, les calculs effectués par la méthode des éléments finit seraient moins représentatifs et les résultats moins proches de la réalité moins proches de la réalité.
Le mathématicien russe Boris Delaunay (1890-1980) a énoncé pour la première fois la définition de la triangulation qui porte son nom en 1934 dans son œuvre <Sur la sphère vide> [16].
La triangulation de Delaunay impose une contrainte : pour chaque triangle du maillage, son cercle circonscrit ne doit contenir aucun élément de l'ensemble des sommets comme le montre la figure suivante :

FIGURE 3.4: Représentation de la propriété de Delaunay.

Cela implique une construction assez naturelle du maillage, et <arrange> les triangles pour uniformiser leur dimension. Les seuls triangles plats qui peuvent advenir se situent aux frontières de l'ensemble de points considéré. Cette triangulation possède des propriétés intéressantes pour construire un algorithme de génération de maillage [17]. On se restreint ici aux propriétés de la triangulation dans un espace à deux dimensions.
Soit n le nombre de points :

- La triangulation est unique s'il n'y a pas trois points alignés ou quatre 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.

3.4.1.3 Les algorithmes :

Il existe deux types d'algorithmes pour générer une triangulation : les algorithmes itératifs d'une part, et récursifs d'autre part. Dans les deux cas, on part d'un ensemble de points du plan et le résultat obtenu est un ensemble de triangle (la triangulation).
L'approche itérative tend à partir d'un ensemble constitué d'un unique triangle et à ajouter à cet ensemble les nouveaux triangles obtenus pour finir avec l'ensemble désiré.
L'approche récursive adopte une stratégie divisée pour régner. Une division par dichotomie de l'ensemble de points est effectuée avant fusion des ensembles.

- a) **Méthode incrémentale :** l'ensemble de points étant fini dans le plan, il est inclus dans un intervalle de |R2. On peut ainsi définir deux triangles initiaux formant cet intervalle et englobant l'ensemble des points. Puis, on choisit un point dans l'ensemble initial. On peut localiser le triangle dans lequel il se trouve et créer les nouveaux triangles formés entre le point et les sommets de ce triangle. On continue de la sorte jusqu'à ne plus de point à choisir. On obtient la triangulation de Delaunay en sortie [18].

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

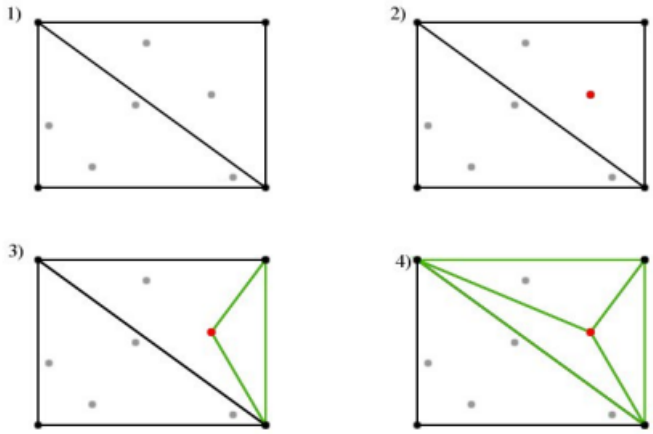

FIGURE 3.5: Première itération de l'algorithme incrémental.

— b) **Méthode récursive :** La créature de cet algorithme sont Lee et Schachter. Plus tard, il a été amélioré par Guibas et Stolfi et finalement par Dwyer. On part de l'ensemble initial de points, on impose un ordonnancement aux points selon axe des ordonnées. Puis on divise cet ensemble en deux sous ensemble en gardent le même nombre de points dans les deux sous ensemble. On réitère la division sur ces deux ensemble et ainsi de suite jusqu'à obtenir au final des ensemble d'au plus trois points. Puis on fusionne chaque couple d'ensembles pour obtenir la triangulation de Delaunay relative à ces deux ensembles en supprimant certaines arêtes en cas de nécessité. On obtient ainsi un nouvel ensemble constitué de triangle. On réitère la fusion entre deux ensemble et cela jusqu'à ce qu'il n'y ait plus d'ensemble à fusionner. Par construction, on obtient la triangulation de Delaunay de l'ensemble de point initial [19].

La complexité théorique de cet algorithme est quasi-linéaire.

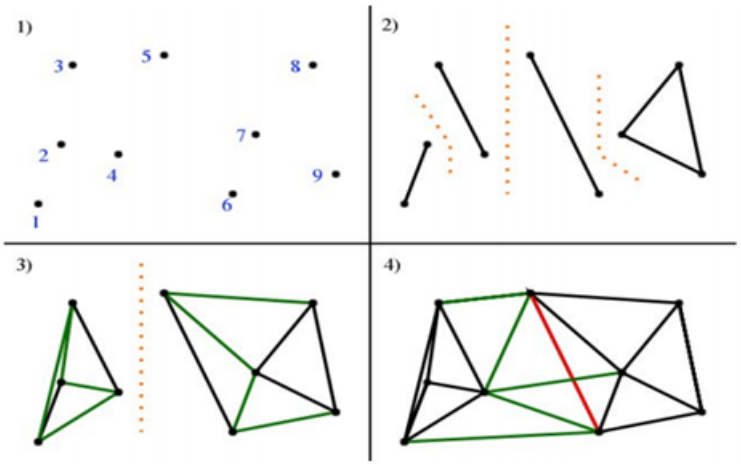

FIGURE 3.6: Division des ensembles puis fusion.

Après avoir achevé l'opération de représentation de l'environnement par la triangulation de Delaunay, dans laquelle nous avons décrit ses propriétés et utilisé la méthode incrémentale, nous allons aborder l'étape de planification de chemin qui exploite cette triangulation.

3.4.2 Planification de chemin

Il existe plusieurs algorithmes de recherche de chemin, parmi lesquels nous avons choisi algorithme A*,
La figure suivante illustre la tâche de la planification de chemin par algorithme A*

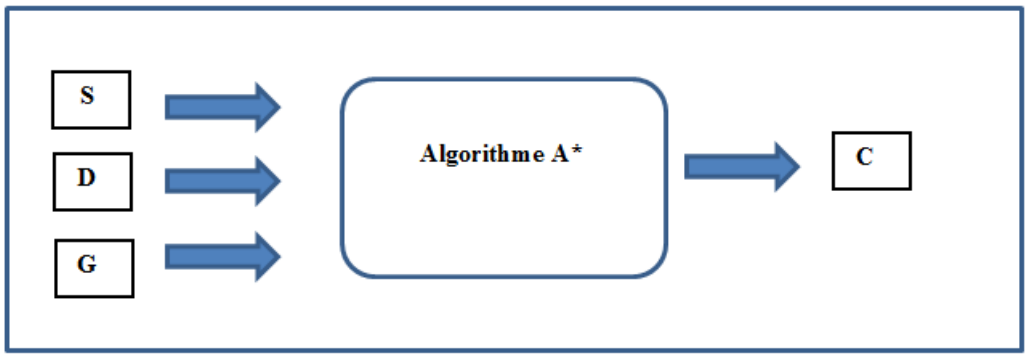

FIGURE 3.7: Planification de chemin par A*.

Avec :

— **S (Source)** : le point de départ.

— **D (Destination)** : le point d'arrivé.
— **G (Graphe)** : le graphe à parcourir.
— **C (Chemin)** : le chemin optimal reliant le nœud de départ et le nœud de destination.

A-Star Search (A*) (Hart, Nilsson et Raphael, 1968) est un des algorithmes classiques les plus robustes car il trouve toujours un chemin s'il en existe un [20] Contrairement à la majorité des algorithmes, il utilise une heuristique de la distance entre la position du personnage et le but, c'est une des caractéristiques qui permet de le différencier. Son seul désavantage notable est le fait qu'il ne trouve pas toujours le meilleur chemin si l'heuristique n'est pas admissible alors que l'algorithme de Dijkstra le trouvera toujours. Pour obtenir une heuristique admissible, il faut que l'heuristique ne surestime dans aucun cas la distance jusqu'à la destination.
A* utilise deux listes de d'états : une liste «ouverte » et une «fermée». La liste ouverte contient toutes les cases où le personnage peut éventuellement se déplacer, alors que la liste fermée contient les cases déjà parcourues.
L'heuristique évalue la distance jusqu'à l'arrivée à chaque case qu'elle traverse. Généralement notée h, elle peut diminuer à chaque pas si elle s'approche du point de destination par rapport au précédent pas. Une autre variable est g, mesurant le chemin parcouru. Ainsi plus le nombre de déplacements est élevé, plus la variable est importante.
L'algorithme additionne les variables g et h pour donner la variable f. Toutes les variables f de chacun des chemins pris par l'algorithme sont ensuite comparées afin de trouver le chemin le plus court. L'algorithme s'arrête dès qu'il a trouvé le point d'arrivée et retourne le chemin parcouru pour s'y rendre.
Les étapes :

— On commence par le nœud de départ, c'est le nœud courant
— On teste tous ses nœuds voisins
— si un nœud voisin est un obstacle, on l'ignore.
— si un nœud voisin est déjà dans la liste fermé, on l'ignore.
— sinon, on ajoute le nœud voisin dans la liste ouverte avec comme parent le nœud courant.
— On cherche le meilleur nœud de toute la liste ouverte. Si la liste ouverte est vide, il n'y a pas de solution, fin de l'algorithme
— On le met dans la liste fermée et on le retire de la liste ouverte
— On réitère avec ce nœud comme nœud courant jusqu'à ce que le nœud courant soit le nœud de destination.

Un nœud n'étant rien de plus que le regroupement des informations suivantes :

— le cout G (le cout pour aller du point de départ au nœud considéré)
— le cout H (le cout pour aller du nœud considéré au point de destination)
— le cout F (somme des précédents mais mémorisé pour ne pas le recalculer à chaque fois)

La figure 29 représente une illustration d'une itération de l'algorithme. On souhaite aller du point orange au point bleu. Les nœuds voisins de la case orange sont les cases marquées en vert, ils passent en liste ouverte. Et de chacune d'elles on calcule les couts G et H pour aller à la case orange et pour aller à la destination. Et la case qui aura le cout le plus

faible sera la case verte du dessous, elle va donc passer en liste fermée. L'algorithme va réitérer à partir de cette case.

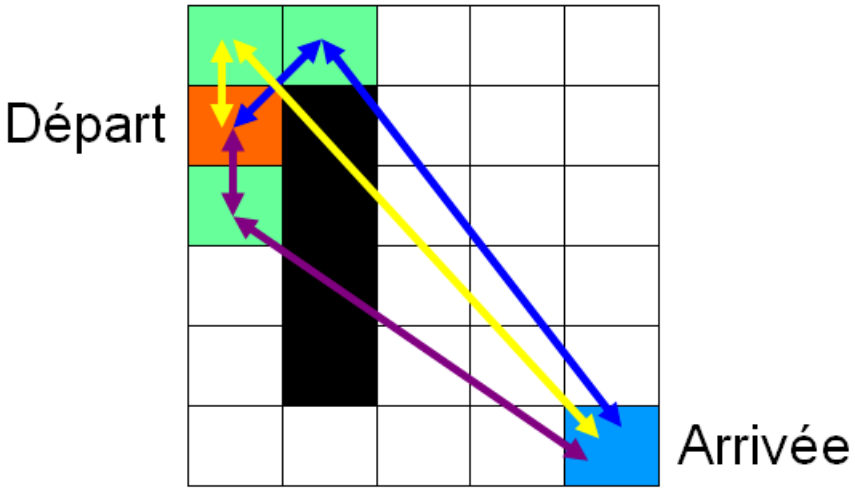

FIGURE 3.8: Exemple d'implémentation de A*.

On peut simplifier l'algorithme A* par le schéma suivant :

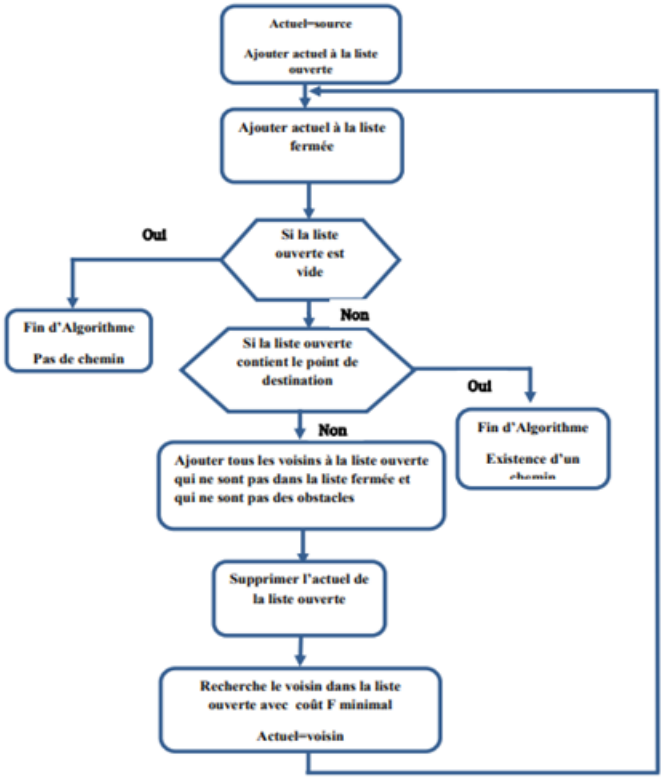

FIGURE 3.9: L'organigramme de l'algorithme A*.

3.4.3 Amélioration du temps de calcul :

3.4.3.1 Transformation les triangles de Delaunay vers des polygones :

Les sommets des polygones sont les centres des cercles circonscrits des triangles de la triangulation de Delaunay. Les arêtes du polygone sont sur les médiatrices des arêtes de la triangulation de Delaunay (cependant les arêtes des polygones sont des segments ou demi-droites qui n'ont pas nécessairement d'intersection sur le centre de ces médiatrices, comme on peut le voir sur la figure ci-contre).

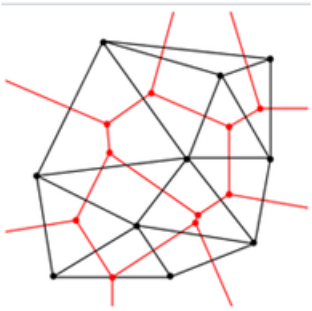

FIGURE 3.10: Superposition d'une triangulation de Delaunay (en noir) et son polygonisation (en rouge.

Soit G un graphique avec P polygones pouvant être parcourus et Polygones B qui sont bloqués comme indiqué ci-dessous

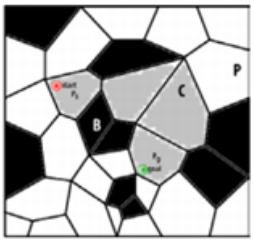

FIGURE 3.11: Exemple des polygones.

Premièrement, nous devons trouver un ensemble de polygones $\subseteq Popasseraunchemin optimal.SiP_s$ est le polygone dont le début est, alors P_s doit être le premier polygone de C. Si P_g est le polygone où se trouve l'objectif, alors P_g doit être le dernier polygone de C. Pour trouver les parties restantes de C, nous devons faire des changements dans la carte. Chaque polygone dans G est mappé sur un nœud dans G'. Par exemple, P_s est mappé sur N_s et P_g sur Ng. Chaque arête partagée par deux polygones dans G est mappée sur une arête reliant deux nœuds dans G '.

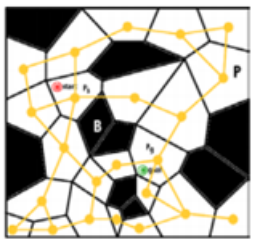

FIGURE 3.12: Transformer des polygones en nœuds.

Ensuite, un chemin optimal P de N_s à N_g en G' peut être trouvé avec A *. Comme chaque nœud de G' correspond à un polygone de G, chaque nœud de P correspond à un polygone de C.

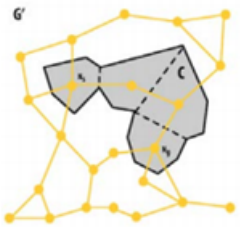

FIGURE 3.13: Chemin obtenu par A*.

C n'est pas un vrai chemin ; au lieu de cela, c'est un ensemble de polygones. La figure 3.1.3 trois manières différentes de trouver un chemin réel en C. Quelle que soit la méthode utilisée, aucune d'entre elles ne peut garantir un chemin optimal. Ainsi, un processus supplémentaire est requis.

FIGURE 3.14: Le long d'un polygone.

FIGURE 3.15: Au milieu du bord.

FIGURE 3.16: Le long de l'obstacle.

3.5 Conclusion

Dans ce chapitre, on a globalement défini les objectifs assignés à notre système, présenté la conception générale de ce dernier, puis détaillé les tâches de chacun de ses modules Dans le chapitre suivant, nous allons présenter la phase de réalisation de cette conception ainsi que les outils utilisée pour implémenter notre système.

CHAPITRE

4

IMPLEMENTATION

4.1 Introduction

L'implémentation est une activité de transformation d'une conception détaillée vers la programmation de notre système. Dans ce chapitre, nous allons décrire la mise en œuvre des différentes étapes de notre système conçu dans le chapitre précédent. Nous allons commencer par justifier l'environnement de développement qu'on a choisi pour implémenter notre système, puis présenter les fonctions utilisées. Nous présentons à la fin de ce chapitre un aperçu des animations d'humain virtuel.

4.2 Outils de développement

Dans cette partie nous allons spécifier les outils utilisés afin de mettre en œuvre notre système. Nous avons commencé par choisir le langage de programmation C++ et nous avons trouvé que Visual Studio 2010 comme une convenable plateforme.

4.2.1 Langage de programmation

Après étude de nos besoins, nous avons choisi l'environnement de programmation Visual Studio 2010 et programmé notre application en utilisant le langage C++, notre choix est motivé par les faits suivants :

— Le langage de programmation C++ est un langage très puissant.
— Le langage C++ offre la possibilité de programmer avec les classes et les objets

Microsoft Visual Studio est une suite de logiciels de développement pour Windows et mac OS conçue par Microsoft. La dernière version s'appelle Visual Studio 2019. Visual Studio est un ensemble complet d'outils de développement permettant de générer des applications web ASP.NET, des services web XML, des applications bureautiques et des applications mobiles. Visual Basic, Visual C++, Visual C utilisent tous le même environnement de développement intégré (IDE), qui leur permet de partager des outils et facilite la création de solutions faisant appel à plusieurs langages.

4.2.2 Moteur de rendu

OGRE 3D (Object-Oriented Graphics Rendering Engine) est un moteur 3D libre multiplateforme (Linux, Win32, OS X, iOS, Android et Windows Phone 8) orienté scène qui permet à partir d'objets à facettes de réaliser un environnement tridimensionnel qui sera perçu par un rendu bidimensionnel au travers d'une ou plusieurs caméra virtuelle. OGRE est une couche d'abstraction supplémentaire au-dessus des API Direct3D et OpenGL, qui permet l'utilisation des cartes accélératrices 3D (OGRE ne fournit pas de moteur de rendu 3D logiciel, il faut une carte 3D ou un émulateur de cartes 3D).

4.3 Les algorithmes de base

4.3.0.1 Algorithme A*

Pour la représentation de la liste ouverte, la liste fermée et la liste qui contient les nœuds du chemin final on a utilisé la classe Nœud. Et on a implémenté ces trois listes comme étant des tableaux dynamiques en utilisant la classe vector de la bibliothèque standard de C++. Cette classe possède plusieurs fonctions qui facilitent la manipulation sur les tableaux dynamiques.

- **Push () :** Ajouter un élément.
- **Top () :** Consulter le dernier élément ajouté.
- **Pop () :** Supprimer le dernier élément ajouté.
- **Sort () :** Pour trier le contenu de la liste

La déclaration des listes est faite comme suite :

```
std::vector <Noeud*> liste_O;
std::vector<Noeud*> liste_F;
std::vector<Noeud*> m_resultat;
```

FIGURE 4.1: déclaration en C++.

La fonction distance peut être calculée comme suit :
Soit un nœud N de coordonnés (X, Y), et un nœud destination D de coordonnés (X', Y').

La distance de Manhatan
Return (abs (A->getx ()-mdest->getx ()) +abs (A->gety ()-mdest->gety ())) ; **La distance ecludienne**
return((A->getx()-mdest->getx())*(A->getx()-mdest->getx())+(A->gety()-mdest->gety())*(A->gety()-mdest->gety())) ;

Fonction A*(départ, destination)

```
Initialise Liste ouverte et Liste fermée à la liste vide
Ajoute départ à liste ouverte
Coût g (départ)=0
Coût f= Coût g(départ)+ Coût h (départ, destination)

Tant que la liste ouverte n'est pas vide

        NoeudActuel= Nœud de la liste ouverte avec Coût f minimum
        Si NoeudActuel= destination
            Retourne chemin

        Supprime NoeudActuel de la liste ouverte
        Ajoute NoeudActuel de la liste fermée
        Pour chaque voisin dans noeudvoisins (NoeudActuel)

            Si le Voisin est dans la liste fermée
               Continue

            Coût g temporaire = Coût g(NoeudActuel) + distance (NoeudActuel, voisin)

            Si Voisin n'est pas dans la liste ouverte ou si le Coût g temporaire < Coût g (voisin)

                Si voisin n'est pas dans la liste ouverte
                      Ajoute voisin à la liste ouverte
                Coût g (voisin) = Coût g temporaire
                Coût f (voisin) = Coût g (voisin) + + Coût h (voisin, destination)
    Retourne "impossible de trouver le chemin"
```

FIGURE 4.2: Pseudo Code de A Star (A*).

4.3.0.2 Triangulation de Delaunay

Extraction des informations géométrique à partir de l'environnement
Une fois les obstacles sont définis dans l'environnement, ils sont englobés dans des parallélépipèdes et les points d'extrémités des obstacles sont calculés en utilisant la fonction **getBoundingBox ()** qui est prédéfinie dans Ogre 3D. Cette fonction nous aide à extraire les quatre sommets (coordonnées) de la base de chaque parallélépipède englobant l'obstacle. Chaque sommet est ajouté pour réaliser la triangulation.
La Création d'objet obstacle

```
//Create objects as obstacl

    Entity *ent;
    SceneNode *node;
    srand(time(NULL));
     Ogre::Vector3 v,v1;

    for(int i=0; i<1;i++)
    {


    ent = mSceneMgr->createEntity("Tete"+StringConverter::toString(i), "ogrehead.mesh");
    node = mSceneMgr->getRootSceneNode()->createChildSceneNode("ogreheadlNode"+StringConverter::toString(i),Vector3(rand()%
    node->attachObject(ent);
    node->setScale(1.5f, 1.5f, 1.5f);

     v= ent->getBoundingBox().getHalfSize();
     v=v*25;

     v1=node->getPosition();

     p->AddVertex(v1.x-v.x,v1.z-v.z,0,i+1);
     p->AddVertex(v1.x+v.x,v1.z-v.z,0,i+1);
     p->AddVertex(v1.x+v.x,v1.z+v.z,0,i+1);
     p->AddVertex(v1.x-v.x,v1.z+v.z,0,i+1);

     }
''
```

FIGURE 4.3: Création d'objet obstacle.

Création des sommets (Triangulation) :
Enfin la création des obstacles l'étape d'ajout des sommets a été lancée :
Au début, il suffit de trouver un moyen de commencer l'ensemble du processus. D'une certaine manière, on doit créer une triangulation de Delaunay valide pour commencer avec. On ajoute ensuite successivement tous les sommets.
Cette triangulation initiale valide est facile à trouver. Il suffit juste de calculer un grand triangle "super triangle" qui englobe tous les sommets. Bien sûr, cela signifie que les triangles superflus seront formés, mais ils peuvent être enlevés après facilement.
L'algorithme complet s'écrit donc comme suit :

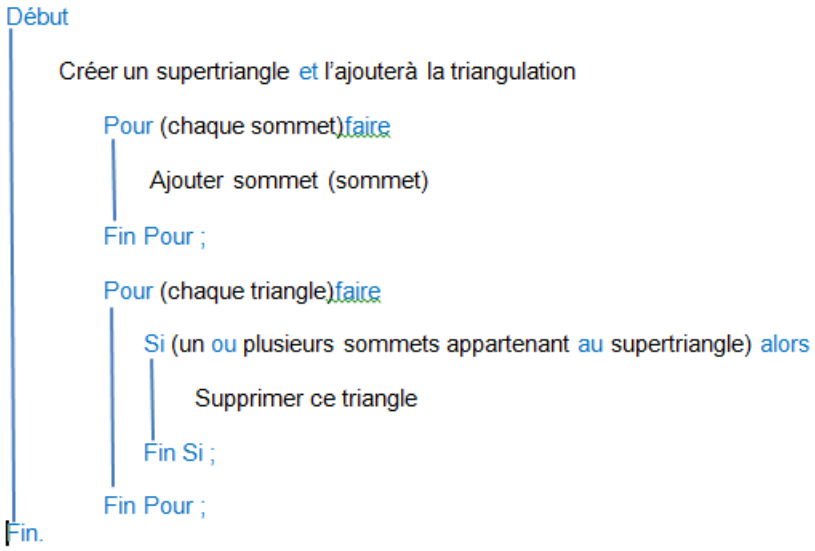

FIGURE 4.4: Algorithme supertriangle.

Pour notre application, on utilise l'algorithme suivant pour ajouter un sommet à une triangulation déjà existante.

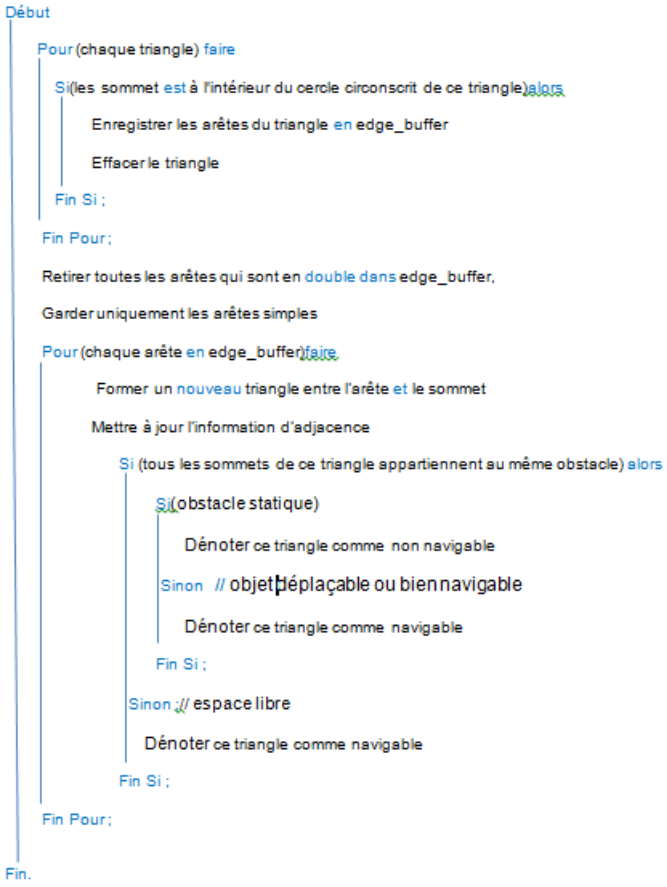

FIGURE 4.5: Algorithme ajouter un sommet.

L'application de cet algorithme est expliquée comme suit :

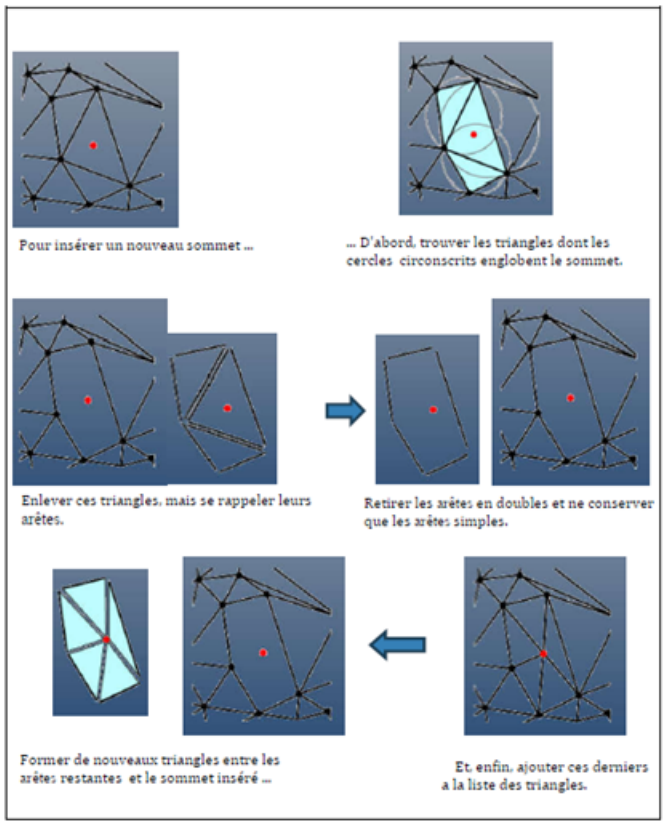

FIGURE 4.6: Processus d'insertion d'un sommet à une triangulation déjà existente.

Les sommets, les segments, et les triangles générés sont sauvegardés dans les tableaux de structure approprie à chaque type.

4.3.0.3 Algorithme de Passage de la triangulation de Delaunay à un ensemble de polygones :

Pour améliorer le temps de calcul on a proposé de transformer les triangles vers des polygones :
Algorithme pour trouver des polygones :

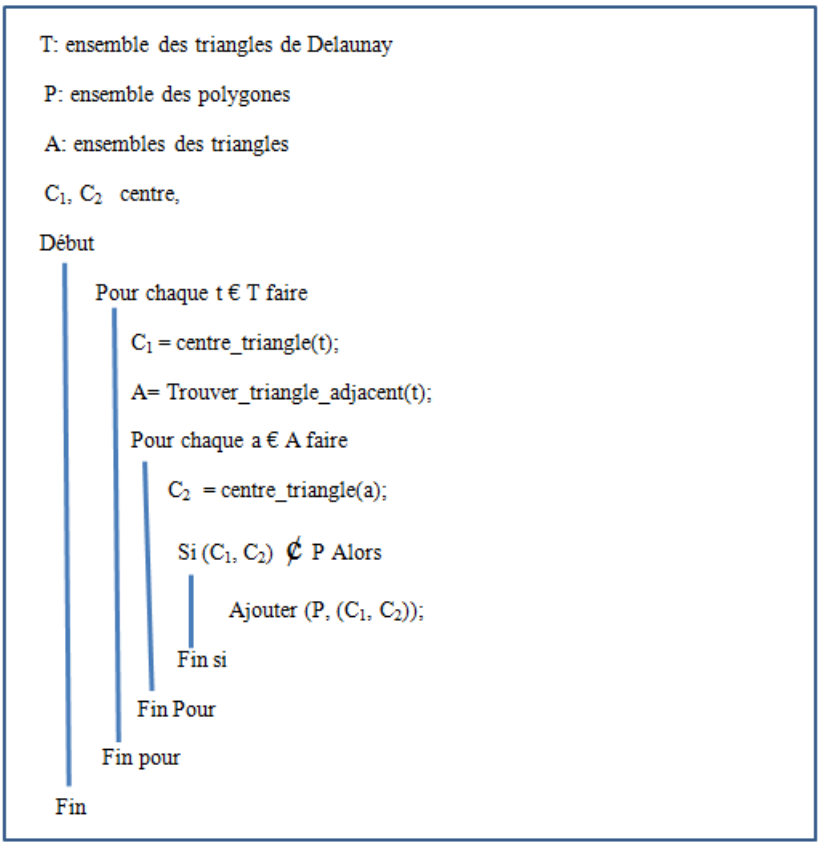

FIGURE 4.7: Algorithme pour trouver des polygones.

Les obstacles sont traités comme des espaces non navigables.

4.4 L'interface du notre application

Pour nos simulations, nous avons utilisé la même configuration de la scène, les points de départ et de destination sont gardés fixes, lorsqu'on crée aléatoirement des obstacles et sans fixer ces deux point, des conflits sont souvent rencontrés. Chaque obstacle dans la scène est représenté par quatre points.

— L'exécution de notre application sans obstacle et avant la triangulation (environnement d'origine) :

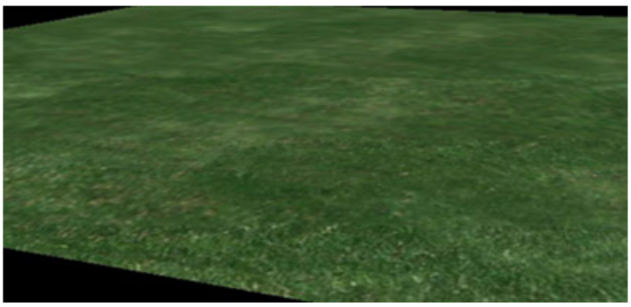

FIGURE 4.8: L'environnement d'origine.

4.5 Les résultats obtenus de notre application

— L'exécution de notre application sans obstacle :

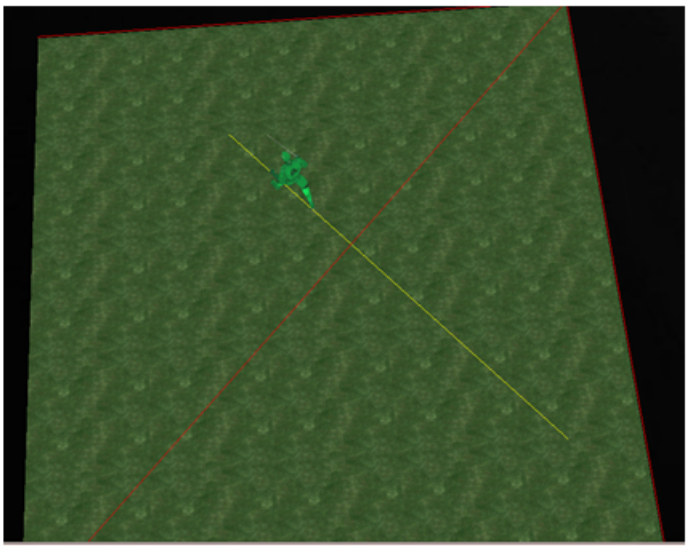

FIGURE 4.9: Plan sans obstacle.

Figure 4.9 : Après l'exécution avec un plan sans obstacles on obtient deux triangles et nous obtenons un chemin sans alias car le plan ne présente aucun obstacle.

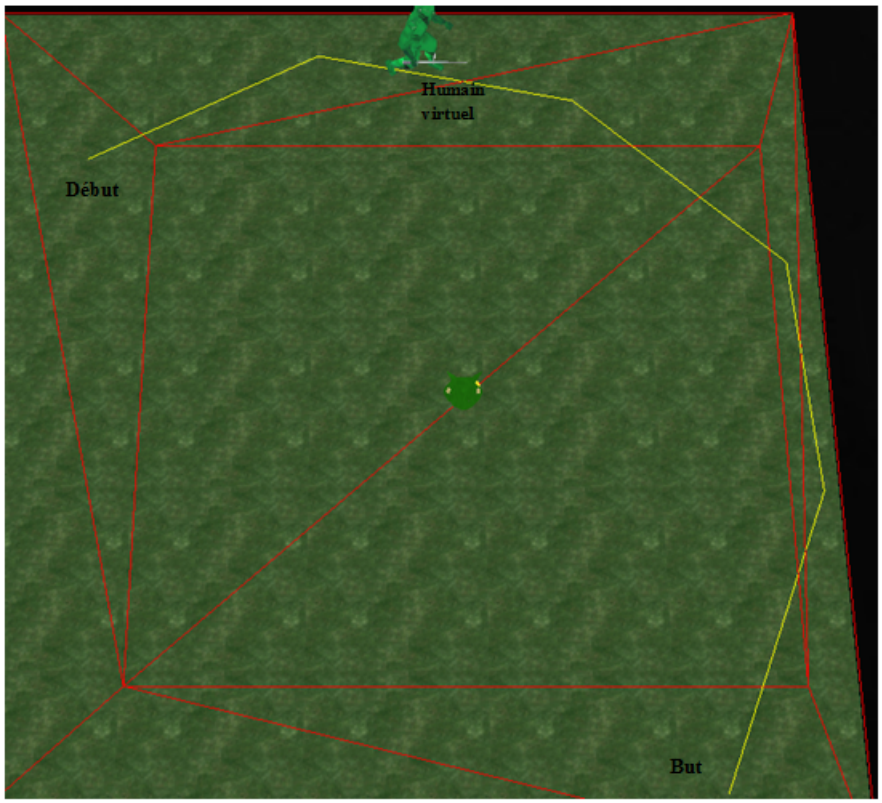

FIGURE 4.10: Triangulation obtenu.

Figure 4.10 : Après l'exécution de la planification de chemin en utilisant l'algorithme A*, on obtient la liste des triangles par lesquels l'entité peut passer pour atteindre son but.
La triangulation de Delaunay est représentée par les lignes en couleurs rouge. Le chemin obtenu par algorithme A* est représenté par la ligne jeune, La création d'obstacles ce fait aléatoirement tel que le point de départ et le point d'arriver sont fixé. Si on répète l'exécution avec un obstacle on obtient le résultat suivant :

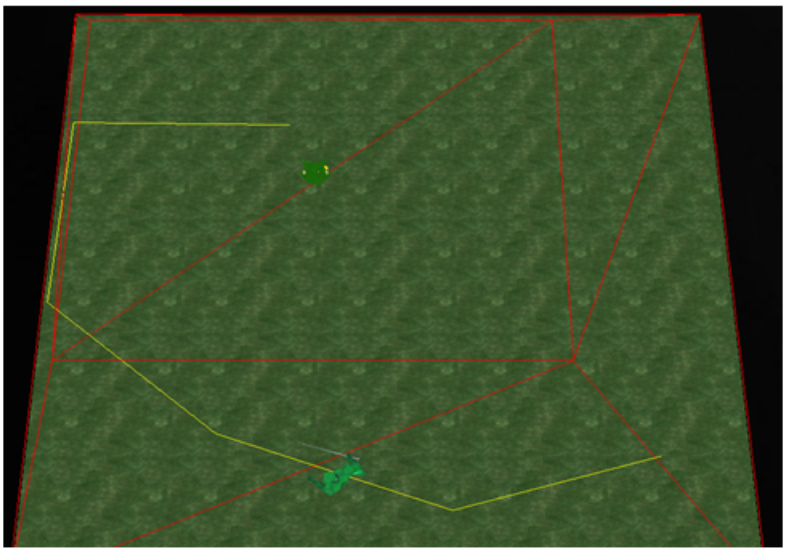

FIGURE 4.11: Plan avec un obstacle.

Figure 4.11 : à chaque fois on a fait l'exécution et lancer l'animation le chemin obtenu est défirent car la position d'obstacle a été changer, donc le changement des positions des triangles et le chemin obtenu par A* est lié au changement de création des obstacles.

— L'exécution de notre application avec 5 obstacles :

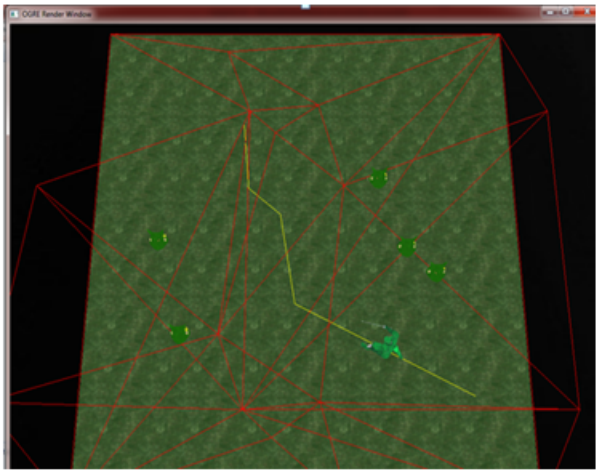

FIGURE 4.12: Plan avec cinq obstacle.

Figure 4.12 : Si nous augmentons le nombre d'obstacles à cinq, cela conduit à une augmentation du nombre de triangles, ce qui rend difficile la recherche du chemin.

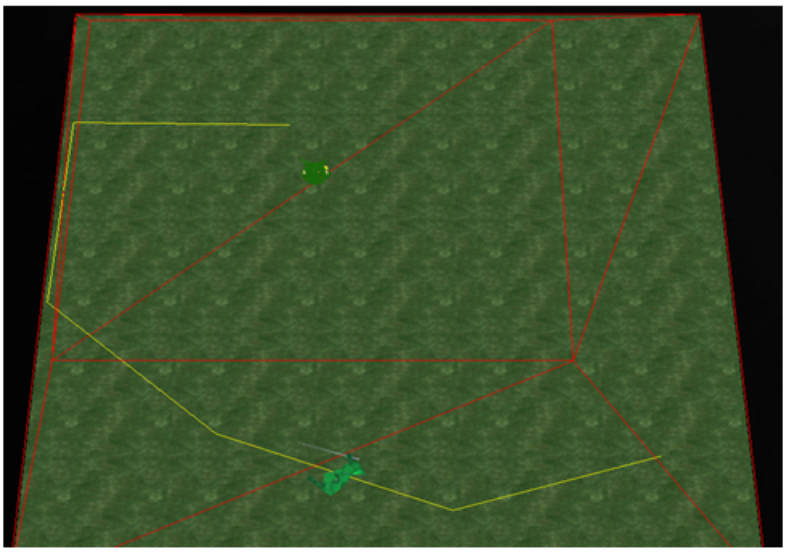

FIGURE 4.13: Conflit.

Figure 4.13 : Cette situation représente un obstacle en position de départ qui empêche l'humain virtuel de se déplacer.
Dans le cas de 25 obstacles ou plus, nous ne trouvons pas le chemin :

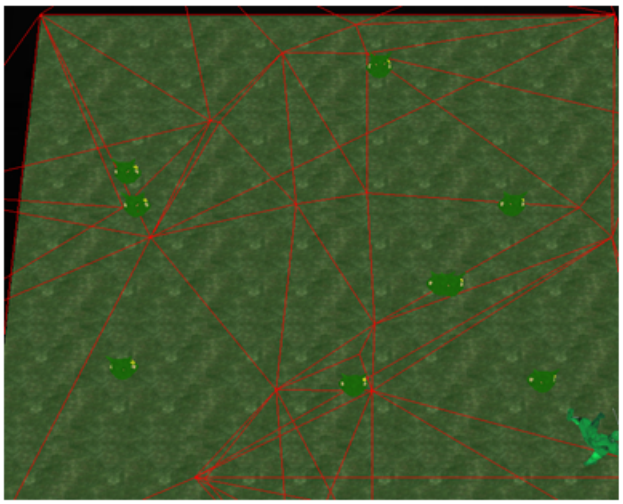

FIGURE 4.14: Cas de 10 obstacles.

4.6 Conclusion

Dans ce chapitre, nous avons présentés le choix de l'environnement ainsi que le langage de programmation utilisés pour implémenter notre système. Nous avons traité au sein de ce travail la problématique de navigation en environnements virtuels. Cette problématique est renforcée dans notre cadre d'application par la nécessité de tenir compte des besoins d'interaction des humains virtuels avec l'environnement ou ils évoluent, afin de remédier aux inconvénients des méthodes basés uniquement sur la représentation géométrique.

CONCLUSION GÉNÉRALE

Le maillage de navigation est une méthode de représentation importante pour les scènes de jeu en 3D, et sa technologie de génération a directement influencé l'efficacité de recherche du rôle du jeu. Nous avons tout d'abord présenté plusieurs méthodes de représentation communes pour les scènes de jeu en 3D et choisit une méthode du maillage de navigation utilisant la technologie de triangulation de Delaunay. Deuxièmement, on a terminé la planification du chemin du rôle de jeu en fonction de algorithme A*. Enfin, compte tenu des exigences spécifiques du cheminement du jeu.

Pour réaliser ce mémoire, nous avons d'abord rassemblé le plus de publications concernant le sujet. La planification de chemin étant un domaine relativement récent. Heureusement, des thèses, articles et autres papiers sont disponibles à ce sujet en ligne.

Pour conclure, la planification de chemin A* fonctionne désormais correctement dans les jeux vidéo dans un environnement triangularisé. Nos recherches dans ce domaine nous ont permis de découvrir et d'implémenter les techniques concernons ce domaine.

La solution que nous avons réalisée se base sur :

— Représenter l'environnement de navigation par la triangulation de Delaunay.
— Augmenter cette représentation par des informations relatives aux objets de la scène.
— Planifier le chemin avec l'algorithme A*.
— Permettre à l'entité virtuelle de naviguer vers sa destination.

Les résultats obtenus sont satisfaisants du point de vue temps de calcul et qualité de simulation obtenue. Néanmoins, notre projet pourra être amélioré par l'ajout d'autre fonctionnalités comme :

— Utiliser une fonction de lissage de chemin obtenu.

— Prendre en considération le volume de l'entité durant le processus de planification.
— Prendre en considération d'autres entités mobiles dans la scène.
— tiliser des règles de comportement pour améliorer le processus de navigation.
— Adapter la solution pour le cas d'environnements complexes.

BIBLIOGRAPHIE

[1] Clodéric Mars. Navigation piétonnière en environnement informé pour des humanoïdes virtuels. IRISA7,2,2006

[2] « Classification des jeux de Roger Caillois » , sur Revue électronique des écoles doctorales ED LISIT et ED LETS (Maison des sciences de l'homme de Dijon) (consulté le 2 octobre 2017)

[3] Kevin Hottot, « Le jeu vidéo entre au Museum of Modern Art de New York » [archive], sur Next INpact, 30 novembre 2012 (consulté le 8 janvier 2019).

[4] https ://www.larousse.fr/dictionnaires/francais/simulation/72824

[5] Stephane Donikian. Modelisation, contr^ole et animation d^aagents virtuels autonomes evoluant dans des environnements informes et structures. Habilitation 'a diriger des recherches, University of Rennes 1, 2004.

[6] F. Lamarche, Humanoïdes virtuels, réaction et cognition : une architecture pour leur autonomie (2003).

[7] F. Lamarche and S. Donikian, Crowd of virtual humans : a new approach for real time navigation in complex and structured environments, Eurographics 23(3) (2004).

[8] Modélisation, contrôle et animation d'agents virtuels autonomes évoluant dans des environnements informés et structurés (2004).

[9] Jakob Von Uexküll : environnement et monde - Christian Fauré (1956)

[10] J.Latombe : Robot Motion Planning , Boston : Kluwer Academic Publishers, Boston, 1991.

[11] C. Gloor, P. Stucki, Hybrid techniques for pedestrian simulations, 4th Swiss Transport Research Conference, 2004.

[12] F. Lamarche and S. Donikian, Crowd of virtual humans : a new approach for real time navigation in complex and structured environments, Eurographics 23(3) (2004).

[13] J. Pettré, J.-P. Laumond and T. Siméon, a 2-stages locomotion planner for digital actors, Eurographics (2003).

[14] P. E. Hart, N. J. Nilsson, B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". IEEE Transactions on Systems Science and Cybernetics SSC4 4 (2) : 100–107. 1968.

[15] Frey P. et George P., « Le maillage facile », Hermès Science Publications Paris, Paris, 2003

[16] B. Delaunay, « Sur la sphère vide », Izvestia Akademii Nauk SSSR, Otdelenie Matematicheskikh i Estestvennykh Nauk, 7 :793800, 1934

[17] Coatelen Jérémy et Falk Kevin "Implémentation de la triangulation de Delaunay en C++" Vendredi 2 Avril 2010

[18] Coatelen Jérémy et Falk Kevin "Implémentation de la triangulation de Delaunay en C++" Vendredi 2 Avril 2010

[19] Coatelen Jérémy et Falk Kevin "Implémentation de la triangulation de Delaunay en C++" Vendredi 2 Avril 2010

[20] L'auteur : khayyam90, L'article Publié le 23 août 2006 - Mis à jour le 26 février 2008