Rapport de projet Infographie
Ace Attorney 3D

Auteurs :
Vincent Metton,
Grégoire Pommier

Responsable :
M. Yannick Guesnet

2 mai 2016

Table des matières

Table des figures

Remerciements

Nous remercions :
M.GUESNET de nous avoir soutenu et d'avoir répondu à nos différentes questions sur l'OpenGL lors de ce projet.

Introduction

En lisant le titre de ce rapport, vous avez pu vous demander : "Mais qu'est-ce que Ace Attorney ?" .
Ace Attorney est une série de jeux vidéos développés par Capcom et édités par Nintendo depuis 2001 sur GameBoy Advance puis porté sur DS et 3DS. Ce type de jeu est le visual novel, c'est-à-dire que l'aventure du jeu se défile sous nos yeux, en superposant du texte, qui sera alors le script du jeu, sur une image de fond. Le joueur doit, à certains moments du jeu, agir afin de faire avancer l'action du jeu.
Dans ces jeux, nous incarnons un avocat à la défense, qui doit prouver l'innocence de ses clients en découvrant la vérité sur l'affaire en cours (généralement un meurtre).
Ces jeux se déroule en deux phases : une d'enquête, où on cherche des preuves afin de disculper notre client, et une phase de procès où nous confrontons les preuves accumulées aux différents témoignages pour déméler le vrai du faux et ainsi résoudre l'affaire.

Dans ce projet, nous n'avons implémenté que la première affaire du premier jeu (The First Turnabout) qui ne contient qu'une phase de procès.

Nous décrirons dans un premier temps les raisons qui nous ont poussé à faire ce projet, puis nous détaillerons les structures et algorithmes utilisés pour la partie graphique (OpenGL), ensuite nous expliciterons les structures et algorithmes de la partie script. Nous finirons par exprimer les différents problèmes rencontrés, les idées d'améliorations et nous conclurons.

Figure 1 – Un aperçu du jeu original

1 Raisons du projet

Dans cette partie nous allons détailler les raisons qui nous ont amené à choisir ce sujet.

1.1 Une passion commune

Tout d'abord, nous sommes tous les deux des passionés de jeux vidéos et tout particulièrement de la série Ace Attorney. De ce fait, cela nous a paru comme une évidence de choisir cet univers et de vouloir le représenter en 3D. En effet, les jeux originaux sont en 2D et offre une variété graphique assez faible au niveau des détails sur la profondeur et des mouvements de caméra. Pouvoir recréer entièrement cet univers en utilisant la biliothèque OpenGL nous a donc offert une totale liberté de de mouvement créant des angles de vue totalements inédits.

2 Les Fonctionalités Graphiques

2.1 Les Fonctions Utisant OpenGL

2.1.1 Le sol

Le sol et constitué de 6 pavés mis côte à côte, afin d'avoir une plus belle reflexion de la lumière. Ces pavés sont eux même composés de sous pavés, qui ne sont au final que de simples GL_ QUADS, sur lesquels on plaque une texture de pavage à neuf cases.

2.1.2 Les meubles

Les meubles sont créer grâce à la fonction creer pave, qui prends en paramêtre les coordonées du centre du pavé, ainsi que sa largeur, sa profondeur et sa hauteur. Cette fonction crée correctement toutes les faces vers l'exterrieur, et fixe leurs normales, afin que la lumière et les textures ajoutées s'affichent correctement.

2.1.3 Les murs

Les murs sont crées par la fonction creer_ mur, qui ressemble beaucoup à creer_ pave, mais ne crée pas les faces en z et -z, et qui de plus, crée les faces vers l'interieur du solide.

2.1.4 Le blason, et la balance

Les éléments situés au dessus du juge ont été crées à l'aide des fonctions de la glu, gluCylinder, et gluDisk. Elles sont tournées et translatées sur le fond de la pièce, et des texture réalistes sont plaquées dessus.

2.1.5 Le marteau et son support

Le marteau, tout comme les précédents éléments, est crée à l'aide d'un quadric, avec les fonctions de la glu. Il a aussi été ajouté un glRotatef dedans, afin de pouvoir faire varier la position du marteau. Cela simule un coup au rendu convainquant.

2.1.6 Le Stand des Témoins

Ce stand est constitué en bas d'un ensemble de pavés, crée par la fonction décrite si dessus, repartis de façon homogène sur un demi-cercle. Il est

surmonté par des gluPartialDisk enpilés les un sur les autres.

2.1.7 Les personnages

Pour créer les personnages, une nouvelle fonction à due être definie, c'est la fonction creer_ pave_ with_ texture, cette fonction permet d'associer à chaques faces d'un pavé, une texture diférente, fournie en argument de la fonction. De plus, certains personages ont subit quelques modification, permetant des animations, au grés du scénario. En effet, les bras du personage principal sont affublés de deux glRotatef, ce qui leurs permet de bouger dans deux directions différentes, le bras du juge, à l'aide d'un glTranslate, bouge au rythme de son marteau, et les têtes des spectateurs de la foules peuvent, à l aide d'un glRotatef tourner de gauche à droite.

2.1.8 Les arcades

Des arcades ont été appliquées sur les murs afin d'améliorer leur esthétisme. Deux pavés créer normalements surmontés de deux autres, qui sont tournés à 45 degrés et déplacés au dessus des prémiers.

2.2 Le modèle de vu

Afin d'afficher la scene, un gluPerspective est fait à l'initialisation. Les arguments associés sont un fovy de 60, ce qui nous a semblé rendre une scène correcte, non déformée. Le rapport d'aspect reste à 1, near est à 10 et far est à 3000 ce qui permet, quelque soit notre enplacement dans la scène, de pouvoir l'afficher en entier.
Un gluLookAt est fait à chaques fois que la scène est affichée, il prend en paramètre des variables globales ce qui permet de pouvoir facilement deplacer la camera dans la scene. Il y deux façons de se déplacer dans la scène

2.2.1 Le mode "Vue libre"

Grace à la fonction PollEvent de le SDL, on peut recuperer les inputs faient au clavier quand la fenêtre a le focus, cela permet de pouvoir modifier les globales whereiamx,y,z et whereialookx,y,z offrant une totale liberté de mouvement dans la scène

2.2.2 Le mode automatique

Certaines touches, ou certaines balises dans le script, lancent les fonction de déplacement, qui elle même appellent movecamera, qui permet de deplacer

de manière fluide la camera dans la scène, en faisant un display de la scène tous les 1/100 ème de la distance entre où je suis et l'endroit où je dois être.

2.3 Les animations

Plusieurs fonctions permettent que créer de petites animations, et sont lancées par des évements clavier crées manuellement ou par le biais du script. Ces fonctions modifient la valeur des variables globales mentionnées plus haut, et affiche la scène plusieurs fois afin de pouvoir voir l'animation complète.

2.4 La musique

Grace à SDLmixer, de la musique à été ajoutée au projet, reprenant les sonorités classique du jeu. Ces musiques sont jouées lors d'évènement de clavier, ou bien lors d'animation (comme par exemple celle du marteau est associé au son du marteau). Il y deux types de musiques.

2.4.1 Les musiques standarts

Ce sont celles courament jouées en fond sonore, pendant la lecture du script. Une seule peut etre jouée à la fois, et certaines boucles alors que d'autres ne sont jouées qu'une fois.

2.4.2 Les "morceaux" de musique

Ce sont les bruitage que ne couperont pas la musique de fond quand ils seront joués. Ce sont principalement les objections des avocats, et autres interjections.

3 Interface Homme Machine

Dans toute application où les interactions avec l'utilisateur sont fréquentes, et notamment dans un jeu tel que la bataille navale, il est bon que cette application possède une Interface Homme Machine (IHM) [1] qui va prendre en compte les actions de l'utilisateur et modifier ce qu'il voit en conséquence.

Nous avons donc développé une IHM permettant à un joueur de pouvoir jouer aisément et sans difficultés. Notre application se découpe en deux fenêtres principales. La première est la fenêtre de démarrage où l'utilisateur pourra définir les différentes options liées à la nouvelle partie et la seconde est l'interface principale du jeu où toute la partie se déroulera. Nous allons donc exposer ces deux fenêtres dans deux parties distinctes où nous expliquerons comment les fenêtres sont construites et détaillerons les différents actions à effectuer. Enfin, nous en profiterons pour mettre en avant, dans une troisième partie, les composants graphiques permettant l'affichage des Boards.

3.1 La fenêtre de démarrage

Comme dit plus haut, la fenêtre de démarrage est la première fenêtre que verra le joueur au lancement de l'application. Pour nous aider à décrire cette fenêtre, voici une capture d'écran :

Comme on peut le voir, la fenêtre de démarrage est séparée en plusieurs morceaux. Pour s'y repérer, nous allons d'abord nous intéresser aux différents champs à remplir et actions à effectuer pour pouvoir créer une partie locale contre une IA, puis nous réitérerons cette logique pour soit créer, soit rejoindre une partie en ligne.

3.1.0.1 Une partie en local

Pour effectuer une partie en local, il faut premièrement indiquer, dans le champ de texte en haut à droite symbolisé par un "A", les dimensions de notre plateau de jeu, chaque dimension étant séparée par le symbole ",". Puis, nous devons choisir, grâce à la liste déroulante symbolisé par un "B" à gauche, le niveau de difficulté choisi pour l'IA qui sont, à l'heure actuelle, au nombres de trois. Enfin, Nous pouvons lancer la partie en cliquant sur le bouton "commencer une partie contre l'IA, qui se situe en bas à gauche symbolisé par un "C".

1. Voir lexique

3.1.0.2 Une partie en réseau

Pour pouvoir jouer en ligne, il faut bien entendu indiquer les dimensions du plateau de jeu, cependant cette information n'est nécessaire que pour le joueur qui hébergera la partie. Comme pour la partie en local, il faut remplir le champ de texte marqué d'un "1" en haut à droite de la fenêtre. Ensuite, chaque joueur doit rentrer son adresse IPv4 dans le champ numéroté "2" et l'adresse IPv4 de l'adversaire dans le champ numéro "3" au centre de la fenêtre.
Il est également possible de récupérer une adresse mémorisée au préalable dans un fichier texte, apparaissant dans les listes déroulantes notées "4" mais aussi de mémoriser l'adresse, qui est écrite dans les champ de textes sus-mentionné, grâce aux boutons sur la droite.
Enfin, le joueur hôte lance la partie en cliquant sue le bouton "créer une partie en ligne", noté "6" en bas au centre de la fenêtre et le joueur invité peut rejoindre une partie après avoir cliqué sur le bouton indiqué par un "7".

3.2 L'interface du jeu

3.2.1 Le placement des navires

Le joueur doit avant tout cliquer sur une des cases du plateau de jeu, ce composant graphique un peu spécial sera explicité dans une autre sous-partie. La case cliquée représentera alors la proue du navire("1") et, en cliquant sur une seconde case, nous choisissons la poupe("2") de celui-ci. Il est à noter que, sur la gauche de la fenêtre, il est possible d'entrer les coordonnées des deux cases en format textuel, c'est à dire "X,X"("3").

Il faut ensuite cliquer sur le bouton "Placer ce navire"("4"), en dessous des champs de texte permettant de renseigner les coordonnées. Le fait de cliquer sur ce bouton a pour effet de faire passer le navire à placer au suivant dans la liste déroulante en haut à gauche("7"). Naturellement, il est aussi possible de retirer un navire précédemment placé en le choisissant dans cette même liste et en cliquant dur le bouton "retirer ce navire" ("6").

Enfin, une fois tous les navires placés, le bouton "prêt"("8") est activé et nous pouvons cliquer dessus pour débuter une partie.

Toutes ces étapes mènent au commencement de la partie. Nous allons donc maintenant décrire ce que l'utilisateur voit et peut faire lors d'un tour de jeu sur une partie déjà avancée.

3.2.2 Une partie typique

Pour comprendre l'interface mise en œuvre ici, il est nécessaire de pouvoir la visualiser correctement. Pour cela, une capture de la fenêtre est donnée ci-après :

Sur cette image, nous pouvons tout d'abord parler du plateau de jeu. En effet, celui ci voit ses cases changées au cours de la partie suivant si la case a été ciblé on non ("1"). Si elle a été ciblé, cette case peut marqué comme "manqué" ("A"), "touchée" ("B") ou ecnore "coulée" ("C").

Lors du tour du joueur, celui-ci doit renseigner la case ou les coordonnées sur lesquelles il souhaite tirer. Il a donc à sa disposition la possibilité de soit cliquer directement sur la case voulue("1") ou alors de rentrer les coordonnées de la case au format textuel comme défini plus haut ("2"). Le joueur fait ensuite feu grâce au bouton "Feu !" ("3") et obtient le résultat de son tir qui affiche donc une image différente pour les trois cas possibles("A", "B" ou "C") ainsi qu'un petit texte décrivant la nature du résultat ("4"). Le joueur a également la possibilité de regarder son plateau de jeu, en cliquant sur l'onglet "Mon Board" ("5").

Voici typiquement comment l'IHM s'arrange et les différentes actions que le joueur peut effectuer au cous d'une partie.

Nous allons maintenant évoquer et expliquer le composant graphique majeur de l'application, le plateau de jeu qui se décline en plusieurs composants suivant quel plateau nous souhaitons regarder : soit le nôtre avec la position de nos navires, soit celui de l'adversaire contenant tous les résultats de nos tirs.

3.3 La représentation du plateau de jeu

La représentation d'un espace à deux, trois ou plus de dimensions se complexifie avec le nombre de dimensions. Nous avons choisi de représenter les plateaux à deux dimensions de manière classique, une surface avec un quadrillage. La représentation des plateaux à trois dimensions utilise une succession de plateaux à deux dimensions empilés entre lesquels on peut naviguer facilement.

Pour les nombres de dimensions supérieures, nous avons fait le choix de ne pas tout représenter étant donné la difficulté de visualiser un espace à quatre dimensions pour les humains et l'absence de conventions communé-

ment partagées sur la représentation de cet espace.

3.3.1 Explications sur les axes

Plutôt que de montrer une représentation qui ne serait pas comprise par les joueurs, nous avons opté pour un système de composantes fixes pour les dimensions non représentées. Ainsi, pour représenter un Board de dimensions (5, 5, 5, 5), on peut fixer la quatrième composante de chaque case représentée à la valeur t : les cases représentées seront celles dont les coordonnées sont (x, y, z, t) pour x, y et z entre 0 et 4, et pour t fixé. Les cases de coordonnées dont la quatrième composante n'est pas égale à t ne sont pas représentées.

Nos composants graphiques utilisent des objets Coordinates d'une façon spéciale pour déterminer les axes et les composantes fixées des coordonnées.

Dans ces coordonnées, les composantes de valeurs positives ou nulles représentent des composantes fixées et les valeurs -1, -2, -3 désignent respectivement qu'il faut représenter les composantes correspondantes sur l'axe X (horizontal, de gauche à droite), l'axe Y (vertical, de haut en bas) ou sur l'axe Z (en profondeur, vers l'écran).

Exemple : Dans une représentation en 3 dimensions d'un Board de dimensions (5, 4, 5, 3, 6), si on fixe l'axe à (-1, 2, -3, 1, -2). Les cases représentées seront celles de coordonnées (a, 2, b, 1, c) avec a, b et c des valeurs entières dans les limites des dimensions du Board. Chaque case représentée sera située à la a-ième place sur l'axe des X, la c-ième place sur l'axe des Y et la b-ième place sur l'axe des Z.

Dans la suite et dans le code source, nous appelons "axes" ces objets qui déterminent quelles composantes sont représentées sur quels axes et lesquelles sont fixées.

3.3.2 Représentation en deux dimensions : les GraphicBoards

Un objet GraphicBoard est un JComponent permettant la représentation en deux dimensions, il connaît le Board qu'il représente ainsi qu'un axe pour déterminer quelles cases du Board il doit représenter et dans quel sens.

En plus de ça, le GraphicBoard contient deux modes d'interaction : un passif et un actif. Le mode passif répond aux clics de souris en envoyant un

événement simple dont la sémantique est tout simplement "on m'a cliqué dessus".

Le mode actif répond aux clics de souris par un CoordinatesEvent contenant les coordonnées de la case cliquée dans le modèle du Board. Ce ne sont pas les coordonnées "graphiques" mais bien celles correspondantes à la case du Board dont on a cliqué sur la représentation graphique.

3.3.3 Représentation en trois dimensions : Les GraphicBoardLayers

Un GraphicBoardLayer est un JPanel contenant plusieurs GraphicBoards en couches superposées. Il permet de naviguer entre ses couches de plusieurs manières : en cliquant sur la couche qu'on veut voir entièrement (elles sont toujours cliquables car la superposition est en décalé), en utilisant le slider sur le coté ou avec la molette de souris.

Le GraphicBoardLayer s'assure qu'un seul GraphicBoard est en mode d'interaction actif et qu'il soit entièrement visible (en décalant ceux du dessus).

Il contient également des options de changement de la représentation, comme les changements d'axes, avec des boutons radios, et dans le cas où le Board qu'il représente contient plus de trois dimensions, il permet de fixer les composantes non représentées sur les axes.

Le GraphicBoardLayer relaie les CoordinatesEvents du GraphicBoard en mode actif.

3.3.4 Délégation de l'affichage spécifique : les BoardDrawer

GraphicBoard et GraphicBoardLayer sont des classes génériques, aptes à représenter les plateaux sur lesquels sont placés les navires comme les plateaux de tir. L'affichage spécifique de chaque type de plateau ne peut donc pas être pris en charge.

Cet affichage est donc délégué à des objets de type BoardDrawer, qui est une interface elle aussi générique, mais que nous avons implanté en deux classes non génériques, ShipDrawer et ShootDrawer.

3.3.5 La représentation des tirs : GraphicBoardShooter

Le GraphicBoardShooter contient un GraphicBoardLayer dont il récupère les CoordinatesEvents pour remplir un champ de coordonnée de tir. Il est doté d'un bouton "feu" qui s'active si les deux conditions suivantes sont remplies :

- La case correspondante aux coordonnées actuelles n'a pas encore été visée.
- C'est le tour du joueur (attribut booléen myTurn qui passe systématiquement à Faux après un tir, et que le contrôleur de jeu peut passer à VRAI ou FAUX) : C'est ce contrôleur (LocalController ou SuperController) qui donne donc l'autorisation de tirer avant chaque tir.

Comme tout n'est pas forcément visible, un label indique la dernière coordonnée de tir et son effet sur l'adversaire (s'il a touché ou coulé un navire).

3.3.6 La représentation des navires : GraphicShipBoard

Le GraphicShipBoard contient lui aussi un GraphicBoardLayer dont il récupère les CoordinatesEvents pour remplir alternativement deux champs de coordonnées (c'est-à-dire qu'il remplit le premier au premier événement, le second au second événement, puis de nouveau le premier, etc).

Avec ces deux champs de coordonnées qui représentent les coordonnées potentielles des extrémités d'un navire, le joueur peut essayer de placer le navire actuellement sélectionné dans la liste déroulante, s'il n'est pas déjà placé. Si les conditions d'un bon placement de navires sont réunies, le navire sera alors placé. Sinon, un message s'affichera, expliquant pour quelle raison le navire n'a pas pu être placé.
Lorsque tous les navires sont placés et que le joueur clique sur le bouton "prêt", le panneau de placement des navires disparaît. Le GraphicShipBoard représente les tirs subis en plus des navires.

Nous avons donc vu ensemble la partie graphique de l'application, de son démarrage jusqu'au déroulement d'une partie classique. Il nous reste une dernière partie à présenter, qui est la partie "humaine" du projet, c'est à dire les choses que nous avons acquis au cours du projet, les divers problèmes rencontrées, les idées d'améliorations potentielles ainsi que les technologies que nous avons utilisées tout au long du projet.

4 Autres

Dans cette partie, nous allons présenter les problèmes que nous avons rencontrés au cours de ce projet, les améliorations possibles pour pouvoir le continuer et enfin les compétences que nous avons acquise lors de la réalisation des différentes parties du projet.

4.1 Problèmes rencontrés

Dans ce projet nous avons rencontré différentes difficultés. Tout d'abord, il a fallut s'initier à l'utilisation d'OpenGL. Certains aspects furent compliquer à réaliser, notamenent réussir une projection au bon endroit avec volume de vue adéquat. Mais aussi créer de solides avec des faces dans le bon sens, des textures corectement appliquées. Il nous a fallut aussi gerer l'éclairement, la décompositon de certains éléments en pavage pour un rendu plus lisse. La lumière diffuse a aussi été dûre à gérer, longtemps les faces opposées à la lumière sont restées noires. Il a aussi fallut gérer un grand nombre de textures, leur application dans le bon sens sur chaqu'unes des figures. De plus, il a fallut comprendre l'utilisation des quadriques et l'appliquation des textures sur eux. Le witness stand a requis l'usage formules de trigonometrie afin d'obtenir une disposition homogène des piliers. Du coté de la lecture du script, les complications ont été d'encapsuler les accès mémoire dans un thread, pour ne pas ralentir la fenetre graphique. Enfin, il modularisation du programme en fichiers disjoints a été longue à faire à cause des étroits liens entre les globales et leurs initialisation, et leur usage dans des fonctions.

4.2 Idées d'améliorations

Notre projet est loin d'être exhaustif, et il y a beaucoup d'amélioration qui peuvent être faites. En premier lieu, certaine des musique du jeu n'ont pas été implentées, pour ne pas trop allourdir le projet, ce qui laisse quelques blanc dans le jeu. Ensuite, les textures et même les formes des personage ne sont pas superbes, et mériteraient d'être refaites et améliorées. Un plafond, manquant pour l'instant, sera ajouté. La possibilité de sauvegarder sa progression, et la fait que les réponses érronées soit pénalisée et mênent à l'echec du jeu, si on en fait trop, et l affichage du dossier des preuves, sont des fonctionalités présente dans le jeu d'origine et qui seront aussi implémentées. Enfin, le netoyage du code, et des évenements personalisés rendrons le jeu plus propre.

4.3 Compétences acquises

Les compétences acquise par la réalisation de ce projet sont la manipulation des bibliotheques OpenGL, et d'une grande partie de leurs fonctionalités. L'usage de la SDL, version 2, notemment de la file d'êvenement qui a permise de gerer de manière efficace le deroulement du jeu. Le renforcement de notre conaissance des thread POSIX. La manipulation des chaines de caractères en C afin de pouvoir lire un script, depuis plusieur fichier source, et effectuer des actions en conséquant.

Conclusion

Ce projet s'est révélé très enrichissant dans la mesure où nous avons pu utiliser toutes nos compétences afin de produire une application la plus fonctionnelle, la plus propre et la plus adaptée au besoin de notre client M.GUESNET.
Ce projet nous a aussi permis de mettre en corrélation les parties théorique et pratique vues en cours afin de produire un projet complet, reliant divers aspect de l'informatique, nous pensons notamment à la partie réseau, couplée à l'IHM et au modèle MVC que nous avons étudié jusqu'ici. Nous avons acquis nombre de nouvelles compétences qui nous seront très utiles dans nos projets futurs, nous pensons notamment à GIT, qui est un outil extrêmement puissant et efficace quand il s'agit de travailler en groupe, ou encore à LaTeX, qui nous permettra de rendre des rapports très beaux et fonctionnels.
Tout en restant positif, nous n'oublions pas les nombreuses difficultés que nous avons rencontrées tout au long du projet, à savoir principalement la difficulté de travailler dans un groupe, être capable d'uniformiser le code, ou l'apprentissage de nouvelles technologies.
Pour conclure, nous souhaitons que notre production soit appréciée par tous et qu'un jour, des développeurs aient envie de reprendre le projet afin de l'améliorer.

Annexe A Lexique

— EDT : EDT est un acronyme signifiant Event Dispatch Thread. Concrètement, EDT est le thread principal s'occupant de la partie graphique d'une application Java, c'est-à-dire le dessin de l'IHM ou la gestion des interactions avec l'utilisateur.

— Itérateur : En programmation, un itérateur est un objet permettant de parcourir un tableau ou une collection, non pas nécessairement depuis le début du tableau, mais depuis n'importe quel index du tableau. Dans le cadre de notre projet, notre itérateur nous permet de parcourir notre Board.

— Pair à pair : Le pair à pair (ou peer-to-peer en anglais), est un modèle de réseau permettant à deux machines de discuter d'égale à égale. Dans les faits, cela s'explique par le fait qu'une machine se connecte à une autre machine et inversement afin que celles-ci puissent s'échanger des informations sans passer par un serveur distant.

— Protocole réseau : Un protocole est une méthode standard qui permet la communication entre des processus (s'exécutant éventuellement sur différentes machines), c'est-à-dire un ensemble de règles et de procédures à respecter pour émettre et recevoir des données sur un réseau. Il en existe plusieurs selon ce que l'on attend de la communication. Certains protocoles seront par exemple spécialisés dans l'échange de fichiers (le FTP), d'autres pourront servir à gérer simplement l'état de la transmission et des erreurs (c'est le cas du protocole ICMP), ...

— Protocole TCP : Acronyme de Transmission Control Protocol, le protocole TCP/IP est le protocole standard utilisé sur internet, pour la liaison entre deux ordinateurs. Le protocole TCP vérifie la validité des paquets après leur réception afin d'être sûr de la validité de celle-ci. Le protocole TCP est située sur la couche 4 (couche de transport) du modèle OSI.

— Protocole UDP : Acronyme User Datagram Protocol, le protocole UDP est un des protocoles standards utilisé sur internet. La différence avec TCP est que les paquets sont reçus sous forme de datagrammes qui doivent être vérifiés pour valider la qualité du paquet reçu. Le protocole UDP est très utilisé, notamment, dans le cadre du jeu en ligne, ou encore le streaming, car la perte de paquet influe peu sur la

quantité reçus. Le protocole UDP est située sur la couche 4 (couche de transport) du modèle OSI, au même titre que le protocole TCP.

— Socket : Une socket est une interface de connexion bidirectionnelle permettant l'échange de données entre deux processus (distants ou non).

— Socket de Berkeley : Les sockets de Berkeley, sont un ensemble normalisés de fonctions de communications lancé par l'université de Berkeley au début des années 1980. De nos jours, elle est la norme utilisé par quasiment l'ensemble des langages de développement (C, Java, Python, ...).

— Socket d'écoute : Une socket d'écoute est une socket présente uniquement dans le protocole TCP. En effet, son rôle consiste, comme son nom l'indique, à écouter les demandes de connexion de socket externe sur un port prédéfini, afin de créer une socket qui permettra ensuite l'échange de données avant de reprendre son rôle d'écouteur.

— Socket de service : La socket de service est la socket crée par la socket d'écoute lorsque celle-ci reçoit une demande de connexion. La socket de service permet la communication entre le serveur et le client ayant fait une demande de connexion. C'est par cette socket que transitent toutes les données émises par le client et le serveur.

— Thread : Un thread est une sorte de processus, dit "léger". Le rôle d'un thread est d'exécuter une suite d'instruction précise que l'on peut nomme routine. Le fait de lancer une application informatique lance automatiquement un thread, celui-ci peut alors créer d'autres threads afin de délégué par exemple une tâche longue a un autre thread, afin que le thread principal (main thread), puisse continuer son fil d'exécution à lui.

Annexe B Webographie

— Fonctionnement des intelligences artificielles : `http://www.datagenetics.com/blog/december32011/index.html`

— API Java : `https://docs.oracle.com/javase/7/docs/api/`

— Règles de la bataille navale : `http://www.regles-de-jeux.com/regle-de-la-bataille-navale/`