Projet Algorithme
Ascenseur

Victor ALAIN & Arthur BLAISE & Amélie GUEDES
& Loann POTTIER

16 juin 2019

Table des matières

I Introduction

I.1 Démarrage et explication

Tout d'abord nous avons décidé de jouer concrètement au jeu, afin d'essayer de comprendre les règles pour mis cerner les problèmes que l'on peut rencontrer en créant les algorithmes. Notre principal problème de compréhension dans le jeu était comment fonctionnent les atouts quels sont-ils et quelles cartes jouer et à quel moment. Ainsi, nous avons fait une partie et nous sommes parvenus à comprendre les principales règles de l'Ascenseur sans comprendre totalement comment fonctionnait le décompte des points. Cela nous a également permis de nous imaginer une schématisation de ce qu'on nous demandait de faire dans une grande globalité. Puis on a pris conscience des consignes qui était les suivantes :

I.2 Problèmes à résoudre

Le but de ce projet est de réaliser un algorithme permettant d'exécuter le jeu l'Ascenseur. Pur ce faire, plusieurs consignes nous ont été donné(?) :

— Tout d'abord, nous devont réaliser un algorithme nommé "100% humain" où il faut initialiser certains paramètre notamment le nombre de joueurs (et donc le nombre de cartes maximal et par manche) puis l'utilisation des atouts et enfin les modalités de victoires et de défaites, en fixant le nombre de points gagnés par défaut, par plis remportés et les points perdus dans les deux mêmes catégories. Ensuite, il faut réaliser l'algorithme qui permet de jouer au jeu de l'ascenseur avec les deux phases : ascendente et descendante. Pour ce faire, il faut indiquer le nombre de manche par phase (Qui varie en fonction du nombre de joueur) et du nombre de tours par manches. Il faut également prendre en considération le fait que chaque cartes jouées est remis en bas du packet de cartes initiales c'est à dire qu'une carte sera jouées plusieurs fois dans la même partie. De plus, il faut que la distribution suivent un nombre logique (Dans le sens des aiguilles d'une montre notamment). En distribuant on finit toujours de distribuer par le joueur situé juste avant le joueur qui a reçu la 1ère carte. On doit également compter le nombre de points réaliser par chaque jour et donc, il faut, après chaque distribution avant de commencer la manche, que le joueur dise à haute voix le parie qu'il soit réalisé.

FIGURE 1 – Jeu de carte utilisé lors d'une partie d'ascenseur

De surcroît, chaque jour ayant remporté le tour précédement, commence au tour suivant. Il est important également que chaque joueur jouent une carte de la couleur demandée si il en possède une, sinon il doit jouer un atout. Cependant, un joueur n'est pas obligé un atout si il n'a pas de carte de la même couleur, il peut déposer une carte quelconque mais il perdra le tour. A la fin de la partie, le joueur possédant le plus de point à l'aide des contracts/paris gagnent la partie. Attention, les jokers ne font pas partie du jeu.

— Dans un second temps, on va créer un algorithme nommé "Version'Ordinateur'" qui reprendra l'algorithme précédent mais qui va implémenter des joueurs artificiels, qui devront chercher à gagner à tout prix (Il s'agit donc d'une IA qui aura les mêmes pensées qu'un humain). On devra considérer une IA comme un humain, c'est-à-dire qu'il jouera exactement la même chose que ce que l'homme jouerai.

II Explication des algorithmes

Pour commencer voici comment est construit le code . tout part d'un programme principal*start.pas* c'est ce programme qui lance toutes les fonctions secondaire et les appellenjt dans le bon ordre et qui permettent le bon déroulement du jeux . De surcroit ce programme appel des unités il faudra donc le compilé que tout fonctionne correctement

II.1 Programmeur

programme principal : start.pas

Le programme principal *start.pas* ne demande au joueur que de se crée "un joueur " pour la machine et ainsi savoir combien d'utiisateur joueront. Dans ce programme il y a aura également la présence de **uses** dont seuls des fonctions nécessaire sont utilisées. Ce programme appelant des *unit* ce dernier n'est pas excisement long (c'est le but des units et de voir plus clair dans son code et également de savoir d'où vient le probléme en cas d'erreur plus facilement
Ensuite durant le déroulement du programme les choses se complexifie car le deroulement du jeu est définie dans 2 unité l'une demande si le joueur soir afficher les régles *intro.pas* puis dans une autre chaque étapes du jeu *deroulement.pas*. Afin d'assurer le déroulement visuelle du jeu l'aspect graphique est contenu d'un seul unit *graph.pas*.

II.2 unités

auparavant nous avons parlé d'unité nous allons donc a présent détaillé celle-ci

intro.pas

Ce unit est particulièrement simple car il ne demande a l'utilisateur que si il souhaite lire les régles du jeu tout simplement

déroulement.pas

Ce unit est initalement le coeur du programme il fait donc appel à 11 fonctions et 9 procédure . Ces dernières sont :

III Diffultés et conclusion