Projet algorithmique :
Jeu de l'ascenseur

William Kaczmarek Théo Julien AlexanderMazeres AlexandreHouhou

21 juin 2019

Table des matières

I Introduction

Notre projet du premier semestre en Algorithmique consistait donc à la création d'un jeu de cartes appelé l'Ascenseur en Pascal pour 2 à 5 joueurs avec un ordinateur capable de contrôler rentre 0 et le nombre total de joueur moins un. Nous nous sommes alors demandés : Comment et par quels outils informatiques allons-nous réaliser ce jeu de cartes de la manière la plus optimale et la plus graphique afin qu'il soit semblable à un jeu de cartes classique en ligne ? Pour répondre à cela, nous avions à notre disposition différentes indications sur la manière de procéder ainsi que les diverses étapes à réaliser pour y parvenir sur le document d'Arel : Projet 2019. En effet, tout d'abord nous devions réaliser une version "humaine" de ce jeu de cartes, en fixant différents paramètres et ainsi modéliser une partie complète que nous allons développer par la suite. Puis, nous devions faire une version ordinateur de l'Ascenseur, ce qui correspond plus concrètement à la création d'une IA. Et enfin, nous avons également tenté de réaliser l'interface graphique pour ce jeu, même si cela n'était qu'une extension.

Figure 1 – Jeu de 54 cartes

II Développement

L'Ascenseur est un jeu de cartes qui nécessite 2 à 5 joueurs, ainsi qu'un paquet de 52 cartes « normales » sans joker (pique, cœur, carreau, trèfle, et pour chaque couleur, dans l'ordre croissant de valeur : numéro 2 à 10, puis valet, dame, roi et as). Mais, nous avons préciser toutes les règles de ce jeu au début de notre programme avec la procédure Règles.

Tout d'abord, nous avons commencé par bien lire le sujet plusieurs fois afin de bien comprendre ce qui nous était demandé, dans le but d'avoir une idée de base sur notre sujet et par la suite de poser nos idées sur papier, de répartir les tâches afin de réaliser nos diverses fonctions. Afin de pouvoir bien attaquer le projet on a de suite créé nos constante, et la façon dont on allait gérer les différentes données relative aux joueurs. C'est donc dans l'unité *UBase.pas* que nous avons tout réunis. Les types, qui gèrent le tableau *TabJoueur*. Où sont stockées toutes les infos relatives aux joueurs de la partie. En effet, ce tableau contient des *Joueur* un type, qui contient : le prenom, les cartes de sa mains, ses points, son contrat, son nombre de pli gagné et enfin son statut (si il est humain ou si c'est un IA).

II.1 Version "Humaine" : Création d'une partie classique

Lors d'une création d'une partie composée seulement d'humain, on va tout d'abord choisir les nombres de joueurs (2 à 5), rentrer leurs prénoms. Tout cela est réalisé dans notre unité *UPartie.pas* plus précisement la Procedure *DefineParam()*. Ensuite grâce à la Procedure *CreationPartie(var LesJoueurs : TabJoueur)*, nous allons initialiser les infos relatives à cette partie, c'est-à-dire créer avec le fichier rempli précédemment par *DefineParam()* remplir le TabJoueur avec les prenoms et initialiser leurs points, contrat etc.

```
for i:=1 to 3 do //3 car il y a trois ligne dans le fichier
//A TESTER je pense qu'en enlever le 'case i of' ca devrait fonctionner pareil sar
        Begin
        case i of
                1:Begin
                readln(Fichier,ligne); //Premiere ligne qui est le nombre de joue
                setlength(LesJoueurs,StrToInt(ligne));//Creation d'un tableau du r

                End;
                2:Begin
                readln(Fichier,ligne);//Deuxieme ligne qui est le nombre d'IA
                InA:=StrToInt(ligne);//On transforme la chaine en entier et on met
                End;
                3:readln(Fichier,ligne);//Derniere ligne qui contient les nom, on
                else writeln('Y a pas de else non plus');
                End;
        End;
NbNom:=0; //Joueur que on configure actuellement
name:=''; //Nom qui contient rien au debut
for i:=1 to length(ligne) do
        Begin
        if (ligne[i]='*') then  //Des que un prenom entier est identifier on initi
                Begin
                LesJoueurs[NbNom].Prenom:=name;
                name:='';//On reset le nom pour qu'il prenne le nom d'apres
                LesJoueurs[NbNom].Points:=0;
                LesJoueurs[NbNom].Contrat:=-1;
                LesJoueurs[NbNom].PliGagne:=0;
                NbNom:=NbNom+1; //On augmente le numero du joueur a configurer
                End
        else name:=name+ligne[i]; //Si on a pas de * alors le nom n'est pas finit
        End;
        for i:=0 to high(LesJoueurs) do  //Ici on initialise le statut(IA ou humai
                Begin
                if (i<length(LesJoueurs)-InA) then LesJoueurs[i].Player:=Humain
                else  LesJoueurs[i].Player:=IA;
                End;
```

FIGURE 2 – Procedure CreationPartie

Pour le reste du déroulement du jeu, on va à travers nos différentes Unit, continuer la partie en respectant les différentes règles du jeu.

— *UBase.pas* regroupe toute les bases du jeu, nos constantes, nos types et les infos sur les joueurs.

— *UCarte.pas* regroupe toute les procedure et fonction concernant les cartes et leurs gestion. Comme le mélange du paquet, la distribution et tout ce qui concerne la main des joueurs.

— *UIA.pas*, comme son nom l'indique cette Unit regroupe les fonctions pour que l'IA puisse jouer.

— *UPartie.pas* est le coeur de notre programme, dans cette Unit on y trouve tout le déroulement d'une partie, de l'initialisation jusqu'au score final. Et utilise toutes les autres Unit.

— *Ugraph.pas*, surement notre Unit la plus grande avec près de 1250 lignes. Celle qui nous à aussi prit le plus du temps. On y retrouve une fonction ou une procedure (11 en tout) pour chaque fenêtre, que nous affichons pour notre jeu. Il y a une procedure, nommée *partiecours()*, au stade final de notre jeu nous n'en n'avons plus besoin mais cette procedure nous permettait d'indiquer que le jeu se jouait dans la console à l'époque où notre interface graphique était plein de bug et injouable.

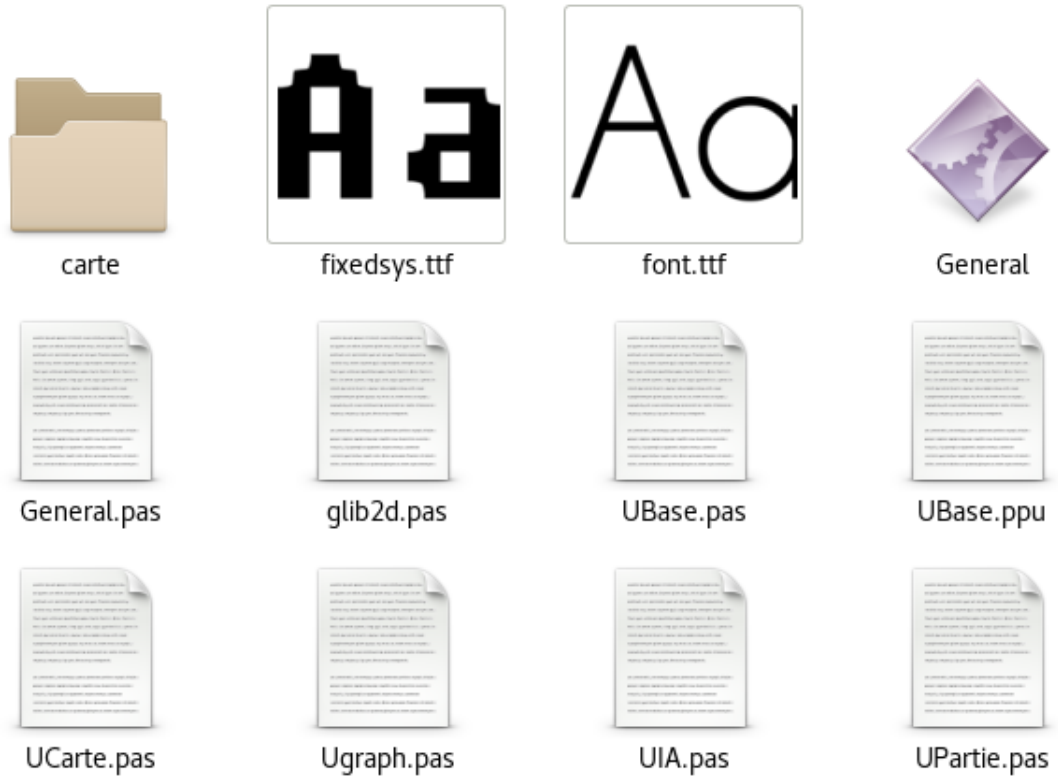

FIGURE 3 – Tout les fichiers nécessaire à la compilation

II.2 Version "Ordinateur" : Création d'une IA

Dans un deuxième temps nous devions réaliser. Nous avons donc décidé de le définir sur papier suivant plusieurs facteurs qui nous semblaient important après avoir joué à ce jeu avec un jeu de cartes classique. Les différents paramètres que nous avons posés pour optimiser l'IA sont :
- Concernant le contrat :
* Comme nous avons pu le préciser dans notre procédure des règles, notre premier paramètre est celui qu'à partir d'une valeur d'une certaine carte, ici toute carte supérieur à 10, l'IA possédant une telle carte décide de considérer qu'il va faire un pli.
* Et notre deuxième paramètre concerne cette fois-ci les cartes d'atout, en effet, peu importe la valeur d'une des cartes considérés d'atout, cette IA considère également qu'il va faire le pli.
- Concernant le déroulement d'une manche :
* Notre IA joue toujours au-dessus de la valeur et de la couleur de la carte précédente lorsque c'est possible.
* En revanche, si l'IA ne dispose pas de la couleur de la carte demandé au début du tour, il choisit de mettre une carte d'une autre couleur autre que l'atout mais la plus petite en terme de valeur. Et il décide donc ainsi, de garder ses atouts pour la fin.

Nous avions également réfléchi à trois différents styles de jeu ainsi que de niveau allant de facile à difficile, mais nous avons finalement choisi de coder un seul IA celui du niveau difficile ci-dessous afin de l'optimiser (nous avons juste fait une fonction pour le pli, la carte jouée est aléatoire).

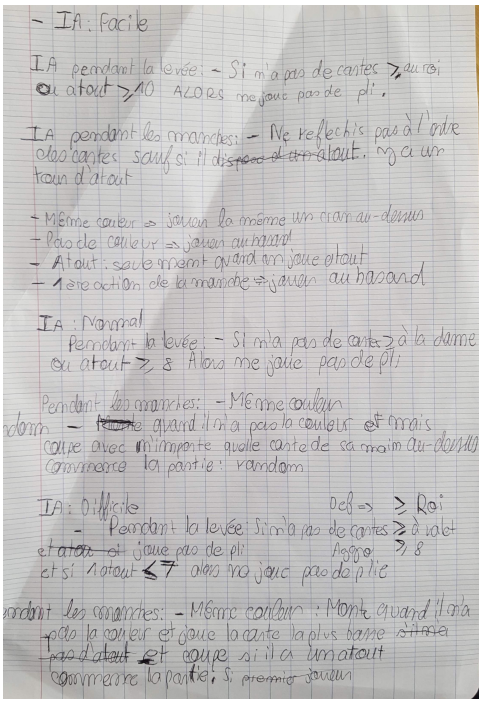

Figure 4 – Création des trois niveau d'IA : 1ère partie

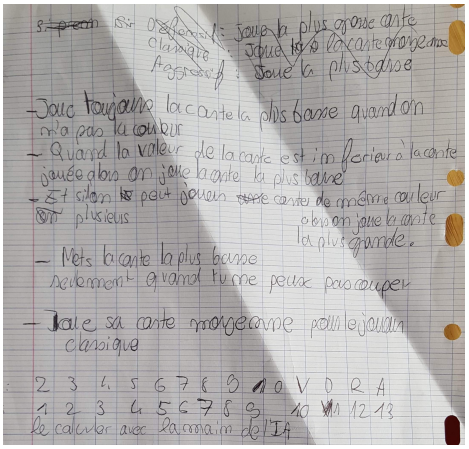

FIGURE 5 – Création des trois niveau d'IA : 2ème partie

Nous pouvons noter que nous avons conçu ces IA de manière à ce qu'il soit dans les mêmes conditions qu'un joueur humain, donc sans triche, c'est-à-dire qu'il ne connaît pas le jeu de ses autres adversaires. Et ainsi, à partir des différents paramètres que nous avons fixé lors de la création de notre IA principal, tels que son estimation de niveau de cartes et son style de jeu, il pourra alors, essayer de prévoir son nombre de plis, de remplir son contrat de la meilleure façon possible.

III Difficultés rencontrées (explication gLib2D

La principale difficulté rencontré lors de la mise en place de l'interface graphique. Fut de comprendre et découvrir une unité graphique : "gLib2D", sans aucune aide extérieure. Willik été le premier de la classe et de la promotion CPI1 à m'attaquer à ce travail. N'ayant rien en rapport avec cette unité sur internet j'ai vraiment dû pas à pas, essayer d'avancer. Le plus dur fut de comprendre comment ajouter une image, une zone de texte sans rencontrer de soucis entre chacune d'entre elles. Après des heures passées à tester on a fini par comprendre qu'il y a deux manières d'afficher quelques choses avec cette interface graphique :

1- "gBeginRect(Image) ; gBlit(0,0,Image,800,600) ; " qui prend directement les coordonnées, et la taille que l'on souhaite. Très pratique pour les grosses images en arrière plan, mais absolument pas pour les zones de texte.

2- " gBeginRects(ZoneDeTexte) ;
gSetCoord(515,176) ;
gSetColor(BLACK) ;
gAdd() ;
gEnd() ; "
Cette méthode-ci totalement manuel est la seule manière efficace pour afficher du texte, car ça nous permet de gérer chaque élément, sa couleur en plus des coordonnées et la taille comme gBlit.

De plus la commande "gFlip ;" à été un véritable enfer, car elle est indispensable à l'affichage de la fenêtre mais on ne sait pas exactement ce qu'elle fait mais elle est obligatoire pour le bon affichage de chacun des éléments présents dans la fenêtre. Si elle n'est pas mise au bon endroit il y aura plein de soucis d'affichage, si elle est pas mise le programme va totalement crasher pendant plusieurs dizaine de minutes.

IV Avantages et limites de la solution

En ce qui concerne, les avantages et les limites de notre programme, nous pouvons dire que l'avantage majeur de ce que nous avons réalisé est l'interface graphique puisqu'en effet, il est plus pratique de jouer à l'Ascenseur via cette interface que dans un terminal. De plus, grâce à la programmation de ce jeu il est possible de jouer face à des IA, ce qui est impossible dans la réalité pouvant donner également un aperçu de notre niveau. Et du point de vue, de la limite de notre solution, il s'agit du fait que nous n'avons pas pu codé l'entièreté de notre IA avec les différents niveaux et certaines règles aussi comme nous avons pu le mentionner dans notre deuxième partie concernant le déroulement de la manche (celle de la pose obligatoire de la carte de la même couleur que le premier à avoir jouer), notamment dû au manque de temps entraînant l'impossibilité de le tester et une certaine complexité pour le coder. Nous avons donc décidé de ne pas le réaliser pour ne pas rendre un projet qui ne fonctionne pas.

Mais maintenant c'est à vous de tester et de jouer !!!

V Excuter notre programme

Il est à noter qu'il faut compiler et éxécuter ***General.pas*** afin de pouvoir jouer.

Maintenant du point de vue de la jouabilité une fois que nous ne sommes plus sur le terminal et que l'on arrive sur l'interface graphique on clique sur le bouton "Jouer".

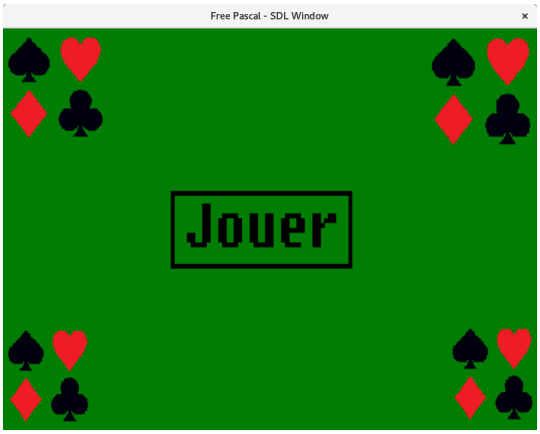

FIGURE 6 – Interface graphique lançant une partie

Puis, une fois avoir cliquer sur le premier bouton différentes fenêtres vont alors s'ouvrir les unes après les autres permettant de fixer les paramètres de jeu que l'on veut.
En premier lieu, l'ensemble des règles du jeu s'affichent et nous arrivons alors sur une nouvelle fenêtre permettant de sélectionner le nombre de joueurs "humains" qui souhaitent prendre part à la partie en cliquant sur le nombre choisi.

FIGURE 7 – Interface graphique sélectionnant le nombre de joueurs

Ensuite, une autre fenêtre s'ouvre donnant accès au choix du nombre de IA que les utilisateurs veulent affronter dans leur partie en cliquant également sur le nombre choisi par ces derniers.

FIGURE 8 – Interface graphique sélectionnant le nombre d'IA

Par la suite, nous arrivons sur la fenêtre pour définir le nom de chaque joueur "humain", ceux des IA étant déjà définis. Il suffit simplement de rentrer le nom et valider en appuyant sur la touche "Entrée".

FIGURE 9 – Interface graphique permettant de rentrer le nom des joueurs

Après cela, une dernière fenêtre apparaît permettant de fixer son contrat pour chaque joueur après avoir vu ses cartes et la carte d'atout, ceci en rentrant le nombre que l'on souhaite et validant grâce à la touche "Entrée"

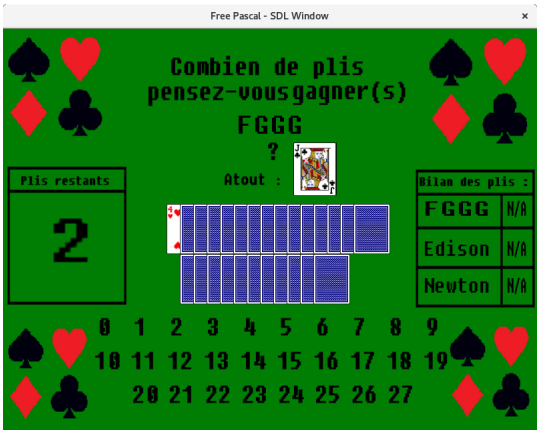

FIGURE 10 – Interface graphique pour déterminer son contrat

Et enfin, la partie peut commencer, il suffit seulement de sélectionner les cartes que l'on souhaite jouer en essayant de l'emporter et de suivre les indications qui suivront au fil de la partie.
Nous pouvons aussi préciser qu'à la fin de chaque manche, il y a un tableau récapitulatif des scores de chaque joueur, mais aussi un tableau du score final à la fin de la partie.

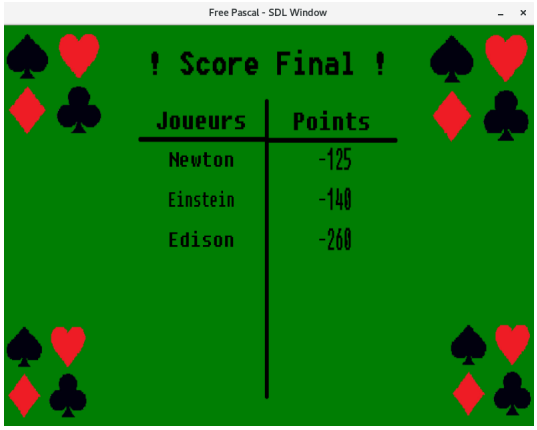

FIGURE 11 – Interface graphique récapitulant les scores des joueurs

Il est à noter aussi que les "clics" doivent être brefs, il ne faut donc pas rester appuyer trop longtemps dessus ou double-cliquer.

VI Conclusion

Pour conclure ce projet d'algorithmique, nous nous sommes organisés correctement afin d'arriver au résultat voulu pour ce projet. Qui étaient : la création d'une version "humaine" d'une partie classique de l'Ascenseur et la création d'une version "ordinateur" pour implémenter des IA (intelligence artificielle) dans le jeu.
Après avoir réaliser tout cela, nous avons tenté de le perfectionner en passant à la conception d'une interface graphique qui nous a posée des difficultés, mais après plusieurs heures nous sommes finalement arrivé à une version graphique fonctionnelle et facilitant grandement sa jouabilité . Nous avons ainsi un projet complètement abouti et en état de marche malgré le manquement d'une seule règle dans la partie code.

Nous pouvons préciser que nous nous sommes très bien entendus au sein de notre groupe avec un bon esprit d'équipe et beaucoup d'échanges et d'entre-aide. Ce projet nous a malgré tout apporté beaucoup en effet, en premier lieu nous avons pu davantage nous découvrir, nous perfectionner en langage pascal. Mais aussi nous avons pu approfondir l'expérience du travail en équipe, ce qui pourra être attendu de nous en entreprise, à travers l'élaboration, les recherches et la conception d'un projet. Cette expérience nous a donc été très bénéfique à tous et nous sommes prêt pour les nouveaux projets l'année prochaine.

Bibliographie

[1] *http ://www.regles.com/jeux-cartes/ascenseur.html*

[2] *https ://www.jeudecartes.be/ascenseur.html*

[3] *http ://52cartes.com.free.fr/Ascenseur.html*

[4] *https ://www.youtube.com/watch ?v=Vymf0VpUbUM*

[5] *https ://www.gameduell.fr/gd/jeux-de-cartes.html*

[6] Cours d'algorithmique *http ://arel.eisti.fr*