

Rapport projet ADA

« jeu TIC-TAC-TOE »

S O M M A I R E

1. Introduction..... 3

2. Présentation du langage Ada..... 4

3. Description de l’environnement de réalisation.....5

4. Gestion du projet5

 Diagramme de GANTT.....5

5. Réalisations6

6. Programmation de l’intelligence artificielle.....11

7. Les difficultés rencontrées.....13

8. Conclusion.....13

1.Introduction

En début d'année de Licence 3 MIAGE, il nous a été confié le développement d'un mini projet « Tic Tac Toe » afin de mettre en pratique les connaissances de programmation acquises en ADA.

Le Tic Tac Toe (ou Morpion) est un jeu de réflexion se pratiquant à deux joueurs au tour par tour et dont le but est de créer un alignement horizontal, vertical ou diagonal sur une grille.

Ainsi, nous articulerons le rapport autour de quatre parties : La première partie consiste en une rapide présentation du langage de programmation utilisé. La deuxième partie aura pour but de définir l'environnement de travail. La troisième partie présente la gestion du projet. Finalement, nous présenterons dans la dernière partie le programme développé lors du projet (les packages et les fonctions).

2.Présentation du langage ADA

Ada est un langage de programmation qui représente l'aboutissement de la lignée des langages "classiques", impératifs et procéduraux. Il constitue essentiellement un effort de synthèse des meilleurs éléments figurant dans les langages qui l'ont précédé, intégrés dans un ensemble cohérent.

Ada a été conçue d'après un cahier des charges, dont l'idée directrice est de diminuer le coût des logiciels, en tenant compte de tous les aspects du cycle de vie. Il a été utilisé avec succès dans des domaines aussi variés que le temps-réel, la gestion, la CAO, le traitement linguistique.

Ada est une norme internationale. Tous les compilateurs actuellement sur le marché ont été validés selon une procédure extrêmement rigoureuse qui assure leur conformité à la norme. Aucun sur-ensemble ni sous-ensemble n'est admis, afin de garantir la portabilité des applications.

Des études économiques ont montré que les projets en Ada coûtaient moins chers en développement, que leurs phases d'intégration étaient plus courtes, et qu'ils restaient moins d'erreurs résiduelles que dans des projets équivalents développés dans d'autres langages.

Passer à Ada est avant tout une décision justifiée sur le plan économique comme sur celui de la sécurité.

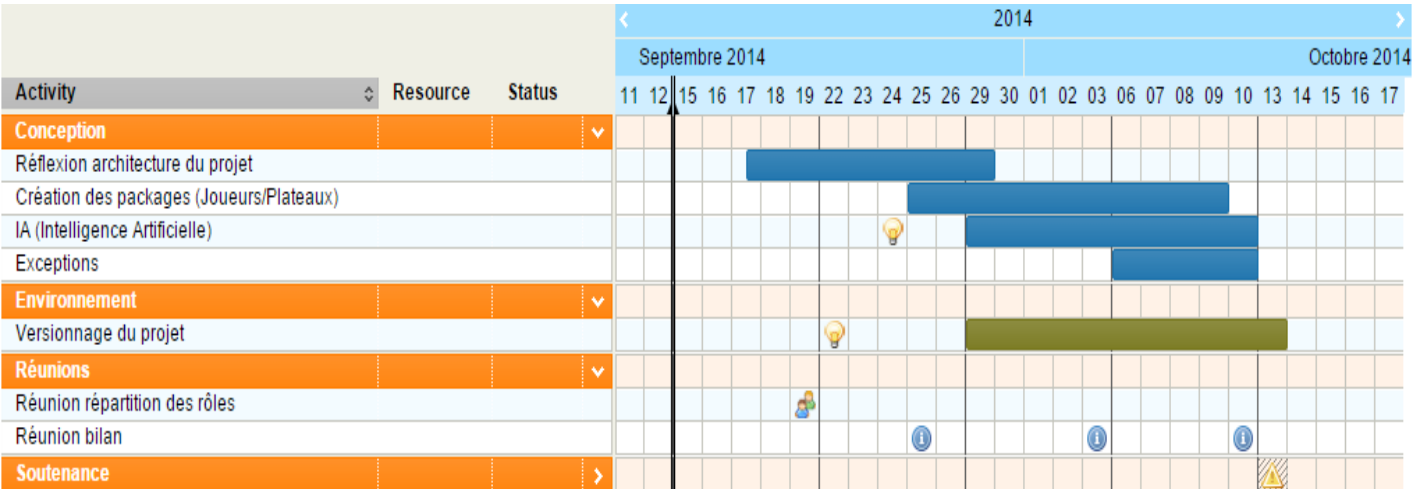
3. Environnement de travail

Le langage de programmation utilisé pour ce projet est Ada. Comme tout langage, Ada a besoin de deux types de logiciels : un éditeur de texte avancé (appelé IDE) et un compilateur. Nous avons utilisé l'IDE appelé GPS (Gnat Programming Studio) et le compilateur GNAT.

En premier lieu, nous avons donc téléchargé et installé le compilateur GNAT. Ensuite, nous avons fait de même pour le logiciel GPS. Afin de mettre en commun notre code, nous avons utilisé le site web relatif à Google « Google Code ». Nous codions chacun de notre côté et faisons une réunion hebdomadaire afin de mettre en commun nos travaux

4. Gestion de projet

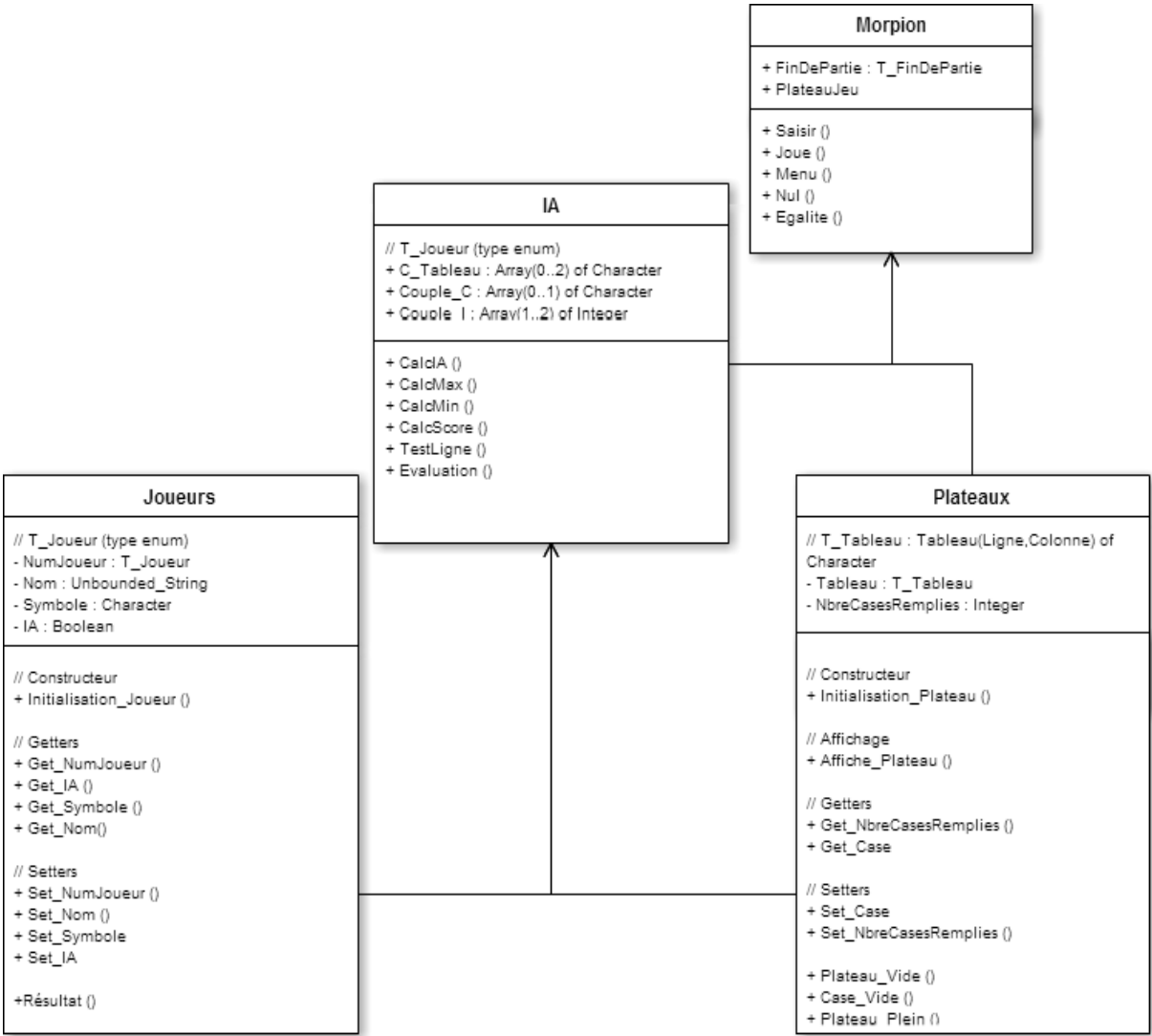
4.1 Diagramme de GANTT



5.Réalisation

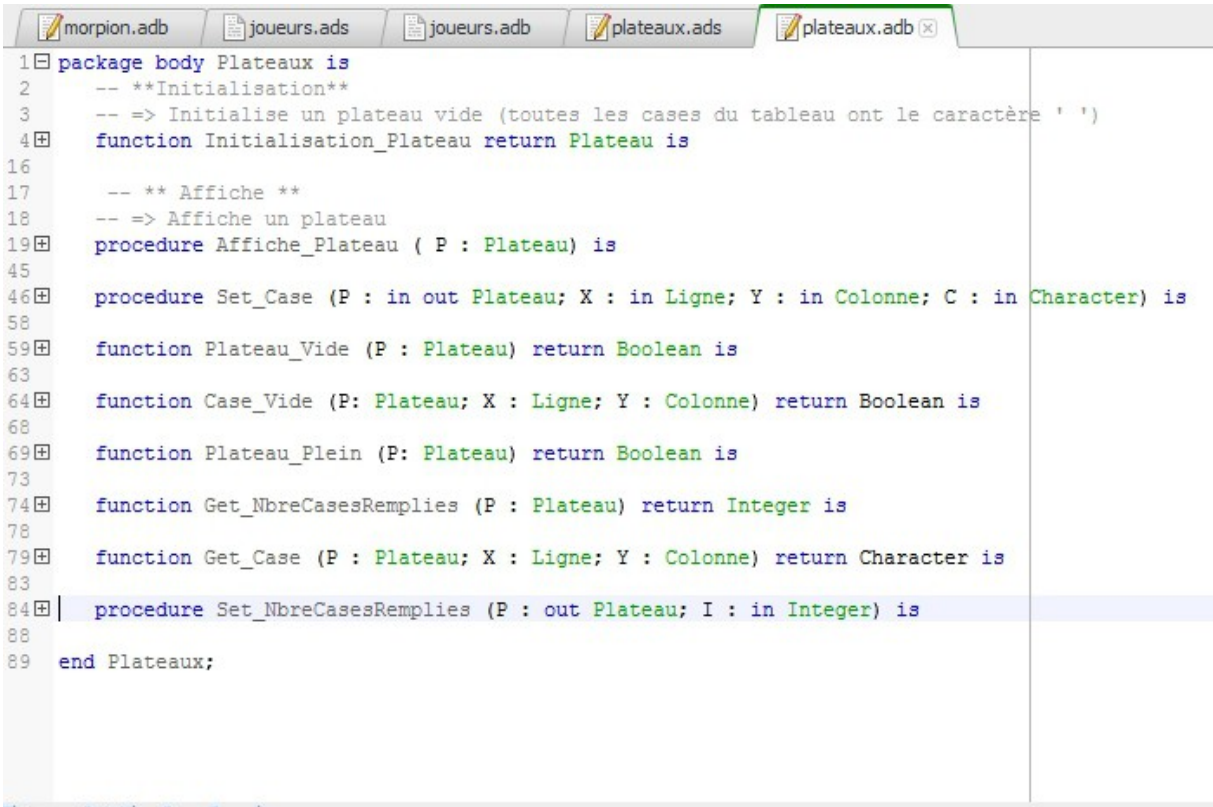
A) Architecture

Le plan de construction du jeu ainsi que son architecture sont représentés ci-dessous par un diagramme de classes :



B) Les différents packages

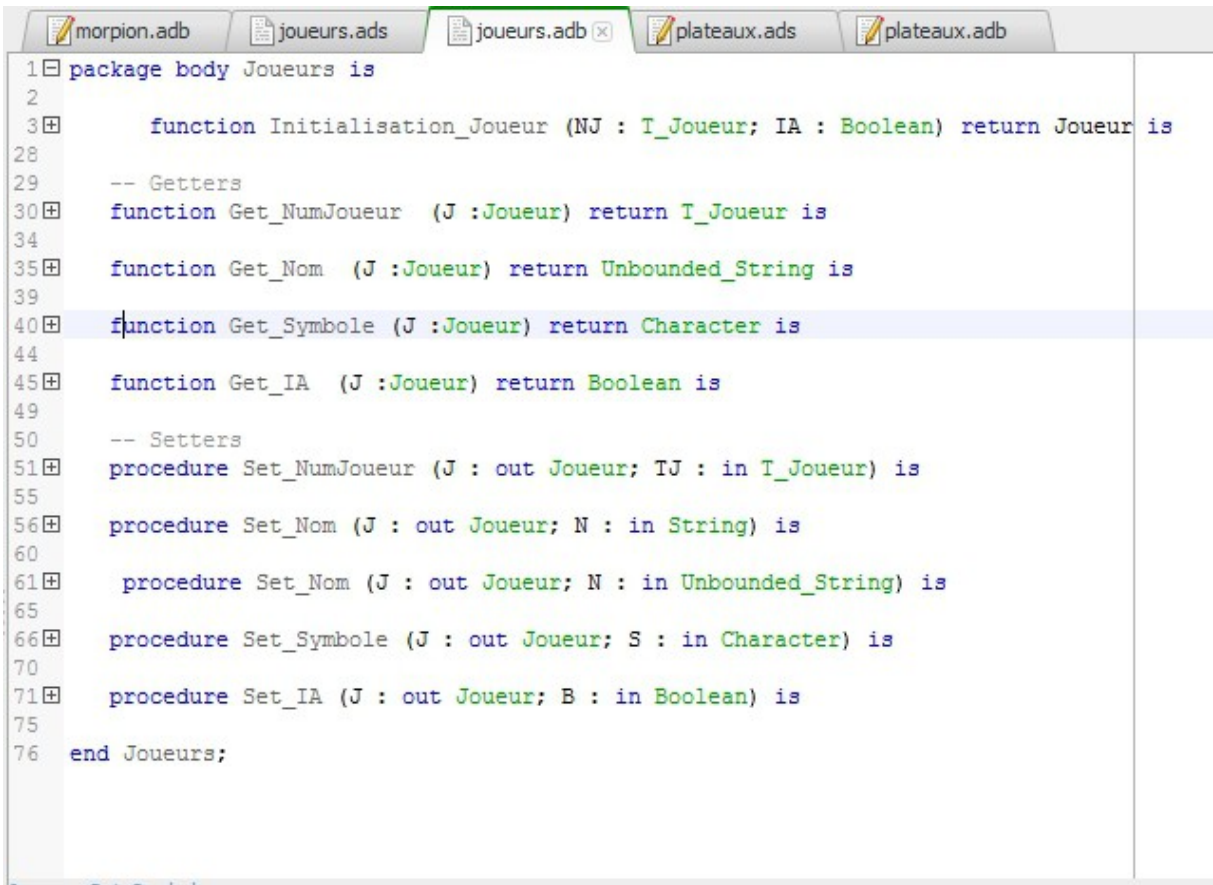
- *Package d’affichage (plateaux)*



```
1 package body Plateaux is
2   -- **Initialisation**
3   -- => Initialise un plateau vide (toutes les cases du tableau ont le caractère ' ')
4   function Initialisation_Plateau return Plateau is
16
17   -- ** Affiche **
18   -- => Affiche un plateau
19   procedure Affiche_Plateau ( P : Plateau) is
45
46   procedure Set_Case (P : in out Plateau; X : in Ligne; Y : in Colonne; C : in Character) is
58
59   function Plateau_Vide (P : Plateau) return Boolean is
63
64   function Case_Vide (P: Plateau; X : Ligne; Y : Colonne) return Boolean is
68
69   function Plateau_Plein (P: Plateau) return Boolean is
73
74   function Get_NbreCasesRemplies (P : Plateau) return Integer is
78
79   function Get_Case (P : Plateau; X : Ligne; Y : Colonne) return Character is
83
84   procedure Set_NbreCasesRemplies (P : out Plateau; I : in Integer) is
88
89 end Plateaux;
```

Le package plateaux permet d’initialiser et d’afficher le plateau de jeu en mettant toutes les cases à la valeur « vide », et permet de tester si le tableau est plein ou pas compter le nombre des cases remplies et récupère la position de la case jouée par l’utilisateur afin de placer le pion du joueur en question.

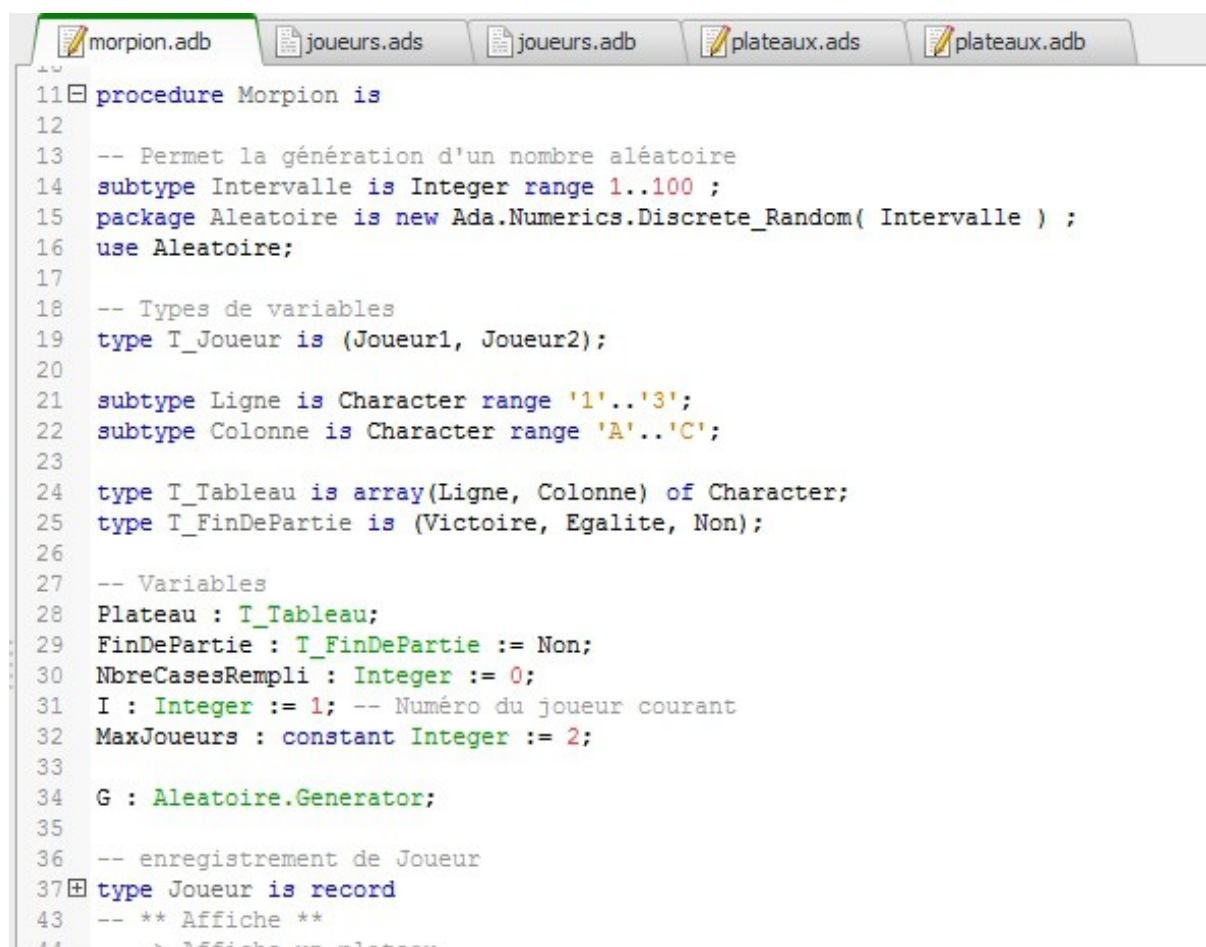
- *Package Joueurs*



```
1 package body Joueurs is
2
3     function Initialisation_Joueur (NJ : T_Joueur; IA : Boolean) return Joueur is
28
29     -- Getters
30     function Get_NumJoueur (J : Joueur) return T_Joueur is
34
35     function Get_Nom (J : Joueur) return Unbounded_String is
39
40     function Get_Symbole (J : Joueur) return Character is
44
45     function Get_IA (J : Joueur) return Boolean is
49
50     -- Setters
51     procedure Set_NumJoueur (J : out Joueur; TJ : in T_Joueur) is
55
56     procedure Set_Nom (J : out Joueur; N : in String) is
60
61     procedure Set_Nom (J : out Joueur; N : in Unbounded_String) is
65
66     procedure Set_Symbole (J : out Joueur; S : in Character) is
70
71     procedure Set_IA (J : out Joueur; B : in Boolean) is
75
76 end Joueurs;
```

Ce package permet au joueur de rentrer son nom et de lui définir un pion.

- Le main ou la classe principale (morpion)



```

11 procedure Morpion is
12
13   -- Permet la génération d'un nombre aléatoire
14   subtype Intervalle is Integer range 1..100 ;
15   package Aleatoire is new Ada.Numerics.Discrete_Random( Intervalle ) ;
16   use Aleatoire;
17
18   -- Types de variables
19   type T_Joueur is (Joueur1, Joueur2);
20
21   subtype Ligne is Character range '1'..'3';
22   subtype Colonne is Character range 'A'..'C';
23
24   type T_Tableau is array(Ligne, Colonne) of Character;
25   type T_FinDePartie is (Victoire, Egalite, Non);
26
27   -- Variables
28   Plateau : T_Tableau;
29   FinDePartie : T_FinDePartie := Non;
30   NbreCasesRempli : Integer := 0;
31   I : Integer := 1; -- Numéro du joueur courant
32   MaxJoueurs : constant Integer := 2;
33
34   G : Aleatoire.Generator;
35
36   -- enregistrement de Joueur
37 type Joueur is record
43 -- ** Affiche **
44 -- Affiche un plateau

```

La classe (morpion) est la classe principale du jeu c'est là où on appelle toutes les fonctions et procédure des autres packages

C) La gestion des erreurs (Exceptions)

```
Put_Line("Bienvenue dans le jeu du (Super) Morpion !...");
Put_Line("Faites [Entrée] pour commencer !");
Ch := Get_Line;
-- tester le choix pour diriger le joueur vers son choix
While (Choix /= 1 AND Choix /= 2 AND Choix /= 3) loop
  TRY:begin
    Choix := Menu;
    case Choix is
      when 1 => Put_Line("ICI faut appeler l'IA");
      when 2 => Put_Line("Cliquez sur [Entree] pour continuer");
      when 3 => Put_Line("Au revoir !");exit;
      when others => raise CHOIX_INCORRECTE;
    end case;
  exception
    when CHOIX_INCORRECTE => Put_Line ("Votre choix est incorrecte. Cliquez sur la touche [Entree] pour reessayer");Skip_Line;New_Line;
    Reessayer := Get_Line;
    when others => New_Line; Put_Line ("ERREUR ! ");New_Line;
    Put_Line("Cliquez sur la touche [Entree] pour reessayer");
    Reessayer := Get_Line;
  end TRY;
end loop;

Skip_Line;
-- ne pas commencer à jouer tant que choix n'est pas egale à 2
... ..
```

Plusieurs exceptions ont dû être implémentées dans ce mini projet. Dès que l'utilisateur saisisait quelque chose au clavier, nous devions gérer les erreurs. Ici, par exemple lorsque l'on demande à l'utilisateur de saisir les coordonnées d'une case tant que la saisie n'est pas correcte le programme continuera à lui demander de réessayer.

6.Intelligence artificielle

L’algorithme utilisé pour programmer l’IA est l’algorithme du Min-Max.

Cet algorithme se présente comme un arbre des possibilités.
Chaque étage représente toutes les possibilités à un état N du plateau pour un joueur.
L’étage N+1 représente donc toutes les possibilités du joueur adverse.

La construction d’une branche de l’arbre s’arrête lorsque qu’il y a match nul, victoire, défaite ou lorsque qu’on a fixé une profondeur maximale à l’arbre.

Une profondeur élevée permettra de simuler profondeur! (Factorielle) possibilités.
Ainsi plus la profondeur est élevée plus l’IA anticipera les coups adverses.
Néanmoins pour le Tic-Tac-Toe, le nombre de cases n’étant que de neuf, une profondeur de 9 est suffisante pour explorer tous les coups et ainsi rendre l’IA imbattable.

Une profondeur inférieure peut simuler un niveau de difficulté moins élevée.

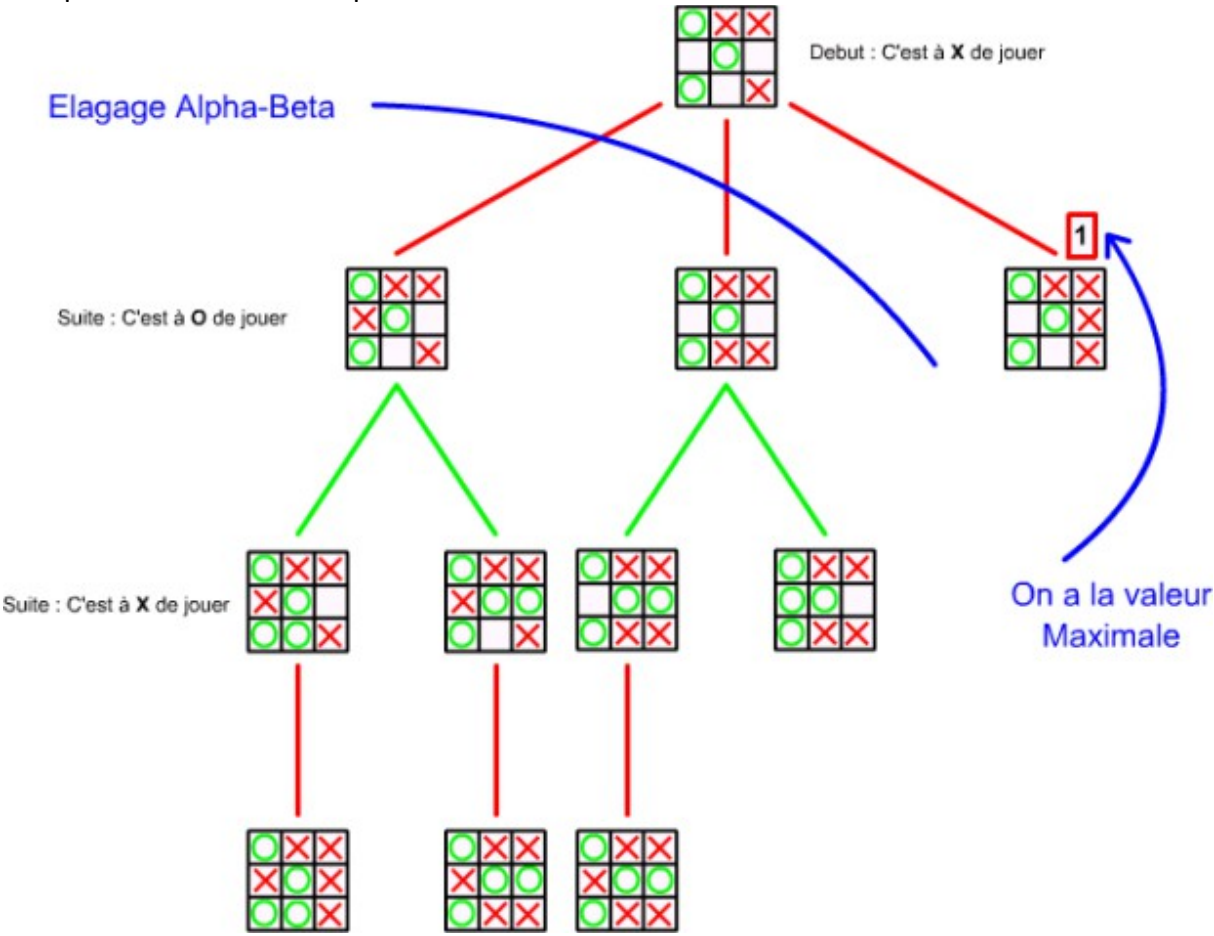


Schéma de l’algorithme du min-max appliqué au Tic-Tac-Toe.

Arrivé à une feuille l'algorithme retournera une valeur qui correspondra à l'état du plateau.

C'est le rôle de la fonction « évaluation » d'évaluer le poids d'une feuille.

En cas de victoire de l'IA, la fonction renverra une très forte valeur (1000), en cas de défaite une valeur très faible (-1000) et en cas de match nul, une valeur nulle (0).

Dans un cas intermédiaire elle renverra une valeur plus ou moins forte si le jeu lui est favorable à l'IA ou défavorable.

Par exemple 2 pions adverses alignés renverront une forte valeur négative (défavorable) pour montrer que le jeu est en défaveur de l'ordinateur.

Les fonctions annexes calcScore et TestLigne permettent de fixer un poids sur une ligne.

Les fonctions calcMax et calcMin sont relativement similaires :

- La fonction calcMax cherche à maximiser les coups de l'ordinateur sélectionnant les coups qui ont le poids le plus élevé. (Donc les coups les plus judicieux)
- A l'inverse calcMin cherche à minimiser les coups, sélectionnant les cases susceptibles de faire perdre l'IA. Cette fonction simule un joueur humain.

Ces deux fonctions s'appellent mutuellement, simulant le tour par tour entre 2 joueurs.

La fonction calcIA quant à elle permet de construire l'arbre et représente son sommet. Cette fonction a la tâche de choisir la case de plus forte valeur et renvoie ses coordonnées.

Amélioration possible :

- Incorporer une difficulté en choisissant la profondeur de l'arbre.
- Établir des stratégies prédéfinies permettant à l'IA de faire des fourchettes.

7. Difficultés rencontrées

Nous avons essayé de coder l'interface graphique, mais on a eu quelques difficultés avec la configuration de Gtkada avec Gps à cause de plusieurs problèmes rencontrés lors de l'installation du gtk.

8. Conclusion

En conclusion, ce projet fut très instructif que ce soit sur un plan scolaire ou personnel. Il nous a amené à découvrir un nouveau langage de programmation. Ce travail de groupe nous a incité à nous fixer des objectifs et à savoir s'organiser pour permettre l'accomplissement de ce projet.