SEVILLA Mathieu

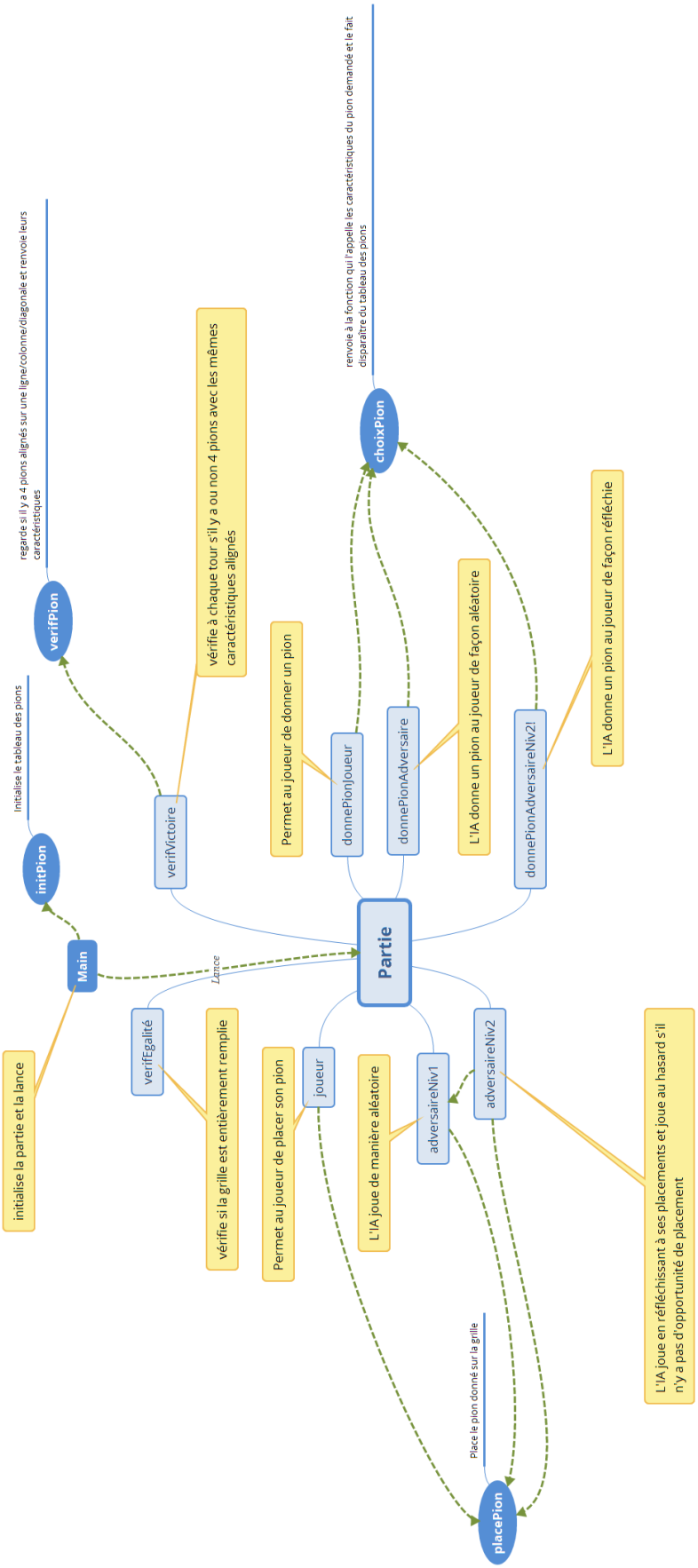
HASTOY Jean

# IPROG PROJECT : QUARTO

# JUSTIFICATIONS TECHNIQUES

Bordeaux INP ENSC

# I. Organisation du programme.



# II. Choix des principales variables

## A. Affichage

Nous avons choisi de réaliser un affichage permettant de représenter les pions de manière visuelle grâce à des caractères. Nous avons donc décidé de stocker chacune des chaines de caractères permettant de représenter le pion dans des tableaux :

- ***string affichagePion[]***

Ce tableau d'une taille de [9] permet de stocker toutes les chaines de caractères permettant de représenter un pion.

- Exemple **:**

Représentation du pion « Carré, grand, vide, foncé ».

| | |
|---|---|
| 0 | « @@@@@@@@@@@@@@@@@@ » |
| 1 | « @ @ » |
| 2 | « @ @ » |
| 3 | « @ @ » |
| 4 | « @ @ » |
| 5 | « @ @ » |
| 6 | « @ @ » |
| 7 | « @ @ » |
| 8 | « @@@@@@@@@@@@@@@@@@ » |

- ***string grille[,]***

Ce tableau d'une taille de [36,14] est la variable principale de l'affichage, c'est elle qui stocke toutes les chaines de caractères affichées à l'écran : celles de la grille de jeu et des pions disponibles. La taille de sa première dimension (36) s'explique par le fait qu'il faut pouvoir stocker les 9 chaines de caractères des 4 pions en hauteur. La taille de sa deuxième dimension (14) s'explique par le fait qu'il faut pouvoir stocker la chaine de caractères des 4 pions du plateau et des 4 pions des pions disponibles, mais il faut aussi pouvoir insérer les délimitations verticales de la grille (grâce à des « | »).

- Exemple **:**

| | PLATEAU | | | | | | | PIONS DISPONIBLES | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | … | 13 |
| 0 à 8 | *affichagePion[0]* … *affichagePion[8]* | \| … \| | *affichagePion[0]* … *affichagePion[8]* | \| … \| | *affichagePion[0]* … *affichagePion[8]* | \| … \| | *affichagePion[0]* … *affichagePion[8]* | *affichagePion[0]* … *affichagePion[8]* | \| … \| | *affichagePion[0]* … *affichagePion[8]* | \| … \| | | *affichagePion[0]* … *affichagePion[8]* |
| 9 à 17 | *affichagePion[9]* … *affichagePion[17]* | \| … \| | *affichagePion[9]* … *affichagePion[17]]* | \| … \| | *affichagePion[9]* … *affichagePion[17]* | \| … \| | *affichagePion[9]* … *affichagePion[17]* | *affichagePion[9]* … *affichagePion[17]* | \| … \| | *affichagePion[9]* … *affichagePion[17]* | \| … \| | | *affichagePion[9]* … *affichagePion[17]* |
| 18 à 26 | *affichagePion[18]* … *affichagePion[26]* | \| … \| | *affichagePion[18]* … *affichagePion[26]* | \| … \| | *affichagePion[18]* … *affichagePion[26]* | \| … \| | *affichagePion[18]* … *affichagePion[26]* | *affichagePion[18]* … *affichagePion[26]* | \| … \| | *affichagePion[18]* … *affichagePion[26]* | \| … \| | | *affichagePion[18]* … *affichagePion[26]* |
| 27 à 35 | *affichagePion[27]* … *affichagePion[35]* | \| … \| | *affichagePion[27]* … *affichagePion[35]* | \| … \| | *affichagePion[27]* … *affichagePion[35]* | \| … \| | *affichagePion[27]* … *affichagePion[35]* | *affichagePion[27]* … *affichagePion[35]* | \| … \| | *affichagePion[27]* … *affichagePion[35]* | \| … \| | | *affichagePion[27]* … *affichagePion[35]* |

## B. Gestion du jeu

Concernant la gestion du jeu, c'est-à-dire la gestion de l'emplacement de chaque pion dans le tableau ainsi que leurs caractéristiques, nous avons aussi choisi d'utiliser des tableaux booléens :

- ***bool infos[,]***

Ce tableau d'une taille de [4, 20] permet de gérer l'emplacement de chaque pion sur le plateau. Nous avons choisi d'identifier les pions de la manière suivante : chaque pion est caractérisé par 5 booléens dont les 4 premiers sont pour définir ses caractéristiques et le dernier représentant sa présence ou non sur le plateau.

- Exemple **:**

Le pion de forme carrée, plein, clair et grand est représenté de la manière suivante :

| | | | |
|---|---|---|---|
| true | false | true | true |

De plus, le nous avons choisi d'ajouter un cinquième booléen permettant d'afficher ou non le pion. Voici l'exemple du même pion affiché :

| | | | | |
|---|---|---|---|---|
| true | false | true | true | **true** |

Et cette fois-ci non affiché :

| | | | | |
|---|---|---|---|---|
| true | false | true | true | **false** |

Ainsi, voici la représentation du tableau initial *infos* :

| | Colonne plateau 1 | | | | | Colonne plateau 2 | | | | | Colonne plateau 3 | | | | | Colonne plateau 4 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| 0 | false | false | false | false | false | false | false | false | false | false | false | false | false | false | false | false | false | false | false | false |
| 1 | false | false | false | false | false | false | false | false | false | false | false | false | false | false | false | false | false | false | false | false |
| 2 | false | false | false | false | false | false | false | false | false | false | false | false | false | false | false | false | false | false | false | false |
| 3 | false | false | false | false | false | false | false | false | false | false | false | false | false | false | false | false | false | false | false | false |

Maintenant, voici la représentation du tableau *infos* permettant d'afficher le pion de forme carrée, plein, clair et grand dans la case en haut à gauche du plateau :

| | Colonne plateau 1 | | | | | Colonne plateau 2 | | | | | Colonne plateau 3 | | | | | Colonne plateau 4 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| 0 | **true** | **false** | **true** | **true** | **true** | false | false | false | false | false | false | false | false | false | false | false | false | false | false | false |
| 1 | false | false | false | false | false | false | false | false | false | false | false | false | false | false | false | false | false | false | false | false |
| 2 | false | false | false | false | false | false | false | false | false | false | false | false | false | false | false | false | false | false | false | false |
| 3 | false | false | false | false | false | false | false | false | false | false | false | false | false | false | false | false | false | false | false | false |

- ***bool pions[,]***

Ce tableau de dimension [4,20] a le même fonctionnement que *infos* et permet d'afficher tous les pions présents dans la partie « Pions disponibles » de la grille. Il est initialisé de base afin d'afficher tous les pions possibles.

- ***bool pionJoue[]***

Ce tableau de dimension [5] permet de faire l'intermédiaire entre *infos* et *pions*. En effet, lorsqu'un pion doit être placé sur le plateau, ses caractéristiques sont copiées depuis *pions* dans *pionJoue* puis recopiée dans *infos* à l'emplacement désiré. Chaque indice de *pionJoue* correspond donc à chaque caractéristique du pion joué.

- ***int placementX, placementY***

Ces deux entiers représentent les positions du dernier pion joué.

- **int choixPionsX, choixPionsX**

Ces deux entiers représentent les positions du dernier pion choisi.

# III. Présentations des principales méthodes.

Présentations des différentes méthodes que nous avons jugé utile de justifier :

- `string[] figure(bool taille, bool interieur, bool forme, bool couleur)`

Cette fonction prend en argument chaque caractéristique d'un pion, et renvoie le tableau *affichagePion* correspondant au pion choisi.

- `void ActuGrille(string[,] grille, bool[,] infos, bool[,] pions)`

Cette fonction prend en argument le tableau *grille*, afin d'actualiser la grille en fonction des tableaux *infos*, et *pions*.

- `void actuAffichage(string[,] grille, int choixPionsX, int choixPionsY, bool color)`

Cette fonction va de pair avec *ActuGrille*. En effet, lorsque la grille est actualisée, *actuAffichage* permet d'afficher la grille de jeu (c'est-à-dire le tableau *grille*) dans la console. Elle prend en argument les entiers *choixPionsX* et *choixPionsY* ainsi que le booléen *color*. Ces trois arguments permettent lors de l'affichage, de surligner le pion choisi par l'IA dans la console.

- `bool[] choixPions(int posx, int posy, bool[,] pions)`

Cette fonction permet autant à l'IA qu'au joueur de choisir un pion. Elle prend donc en argument deux entiers *posx* et *posy* qui indiquent la position du pion sur la grille des pions disponibles. Elle renvoie ensuite le tableau *pionJoue* correspond au pion choisi et passe la 5ème caractéristique du pion choisi sur *false* (pour le faire disparaitre du tableau pions).

- `bool verifPions(int posx, int posy, bool[,] infos, out  bool[] arg)`

Cette fonction permet de vérifier si 4 pions sont alignés sur le plateau. En effet, elle prend en argument deux entier *posx* et *posy* qui indiquent la position du pion venant d'être placé. Ainsi, lorsque le pion placé forme une ligne, une colonne ou une diagonale sur le plateau, la fonction renvoie *true* ainsi que le tableau de booléen *arg* de taille [16] en out contenant toutes les caractéristiques des 4 pions alignés.

- `bool verifVictoire(int posx, int posy, bool[,] infos)`

Cette fonction prend en argument deux entier *posx* et *posy* qui indiquent la position du pion venant d'être placé. Elle lance la fonction *verifPions*, si cette fonction renvoie *true*, il va y avoir une vérification du tableau en out *arg* afin de déterminer s'il existe une caractéristique commune aux 4 pions alignés. Si c'est le cas, la fonction renvoie *true*, c'est la victoire.

- `void joueur(bool[,] pions, bool[,] infos, bool[] pionJoue, string[,] grille, out int placementX, out int placementY)`

Cette fonction permet au joueur de choisir à quel emplacement placer le pion choisi par l'IA. Elle prend donc en argument le tableau *pionJoue* et en out deux entiers *placementX* et *placementY* correspondant aux coordonnées du placement du pion par le joueur qui seront utilisés par la suite.

- `void adversaireNiv1(bool[,] pions, bool[,] infos, string[,] grille, out int placementX, out int placementY, bool[] pionJoue)`

Cette fonction correspond au niveau 1 de l'IA. Elle choisi un emplacement au hasard afin de placer le pion donné par le joueur. Elle prend donc en argument le tableau *pionJoue* et en out deux entiers *placementX* et *placementY* correspondant aux coordonnées du placement du pion par l'IA qui seront utilisés par la suite.

- `adversaireNiv2(bool[,] pions, bool[,] infos, string[,] grille, out int placementX, out int placementY, bool[] pionJoue)`

Cette fonction correspond au niveau 2 de l'IA. Elle choisi possède trois niveaux de jeu :

- si **un seul pion** est sur le plateau : l'IA va placer son pion de manière à commencer à former une ligne, colonne ou diagonale (choisi de manière aléatoire).
- si **deux pions sont déjà alignés** : l'IA va placer son pion de manière à continuer à former une ligne, colonne ou diagonale si son pion possède une caractéristique commune avec les 2 pions déjà placés, sinon, elle le placera au hasard.
- Si **trois pions sont déjà alignés** : l'IA va placer son pion de manière à terminer la ligne, colonne ou diagonale, et ainsi gagner la partie. Si son pion ne possède pas une caractéristique commune avec les 3 autres, elle le placera au hasard.

La fonction prend donc en argument le tableau *pionJoue* et en out deux entiers *placementX* et *placementY* correspondant aux coordonnées du placement du pion par l'IA qui seront utilisés par la suite.

La fonction dépend de 4 autres sous fonctions :

- `adversaireNiv2_verifDiagG(bool[,] pions, bool[,] infos, string[,] grille, out int placementX, out int placementY, bool[] pionJoue, int nbPions)`

- `adversaireNiv2_verifDiagD(bool[,] pions, bool[,] infos, string[,] grille, out int placementX, out int placementY, bool[] pionJoue, int nbPions)`

- `adversaireNiv2_verifColonne(bool[,] pions, bool[,] infos, string[,] grille, out int placementX, out int placementY, bool[] pionJoue, int nbPions)`

- `adversaireNiv2_verifLigne(bool[,] pions, bool[,] infos, string[,] grille, out int placementX, out int placementY, bool[] pionJoue, int nbPions)`

Chacune de ces sous fonctions permet de vérifier l'alignement de 1, 2, ou 3 pions (argument *int nbPions*) sur la diagonale droite, la diagonale gauche, les lignes, et les colonnes.

- `void joueurDonnePion(bool[,] pions, ref bool[] pionJoue)`

Cette fonction permet au joueur de donner un pion à l'IA. Pour cela, lorsque le joueur à choisi le pion joué, la fonction va modifier le tableau *pionJoue* en utilisant la fonction *choixPions*.

- `void adversaireDonnePionNiv1(bool[,] pions, ref bool[] pionJoue, string[,] grille)`

Cette fonction permet à l'IA de donner un pion au joueur. Pour cela, une fois que l'IA aura choisi un pion de manière aléatoire, la fonction va modifier le tableau *pionJoue* en utilisant la fonction *choixPions*.

- `bool adversaireDonnePionNiv2(ref bool[] pionJoue, ref bool[,] infos, bool[,] pions, string[,] grille)`

Cette fonction permet à l'IA de donner un pion au joueur qui ne permettra jamais de le faire gagner lorsque 3 pions sont déjà alignés.

Pour cela, la fonction utilise la sous fonction :

- `adversaireDonnePionNiv2_verif(ref bool[] pionJoue, bool[,] infos, bool[,] pions, out int indice)`

Cette fonction retourne, dans le cas où 3 pions sont alignés, un tableau booléen *posGagnante* de taille [10, 2] contenant les positions des emplacements dans lesquels un pion est manquant pour gagner la partie.

Ensuite, la fonction va appliquer avec chaque pion restant la fonction *verifVictoire* à toutes les positions potentiellement gagnantes du tableau *posGagnante* pour identifier le pion qui ne fera pas gagner l'adversaire. Si aucun pion n'est identifié, elle jouera au hasard. Enfin, elle modifiera le tableau *pionJoue* grâce à la fonction *choixPions*.

- `int partie(bool[,] pions, bool[,] infos, string[,] grille, int difficulty)`

Cette fonction permet de gérer le déroulement de chaque partie en :

1) Choisissant au hasard qui commence.
2) Déroulant la partie tant qu'il n'y a pas de victoire ni égalité.
3) Lançant tour à tour les fonctions permettant au joueur de jouer : *joueurDonnePion*, puis *adversaireNiv1/2* (selon la difficulté choisie).
4) Lançant tour à tour les fonctions permettant à l'IA de jouer : *adversaireDonnePionNiv1/2* (selon la difficulté), puis *joueur*
5) Affichant le gagnant de la partie.
6) Retournant *-1* pour défaite, *0* pour égalité, ou *1* pour victoire.

- `void Main(string[] args)`

Le Main du programme permet d'effectuer l'initialisation des principales variables et de gérer la répétition des parties, en comptant notamment le nombre de défaites, d'égalités, et de victoires.