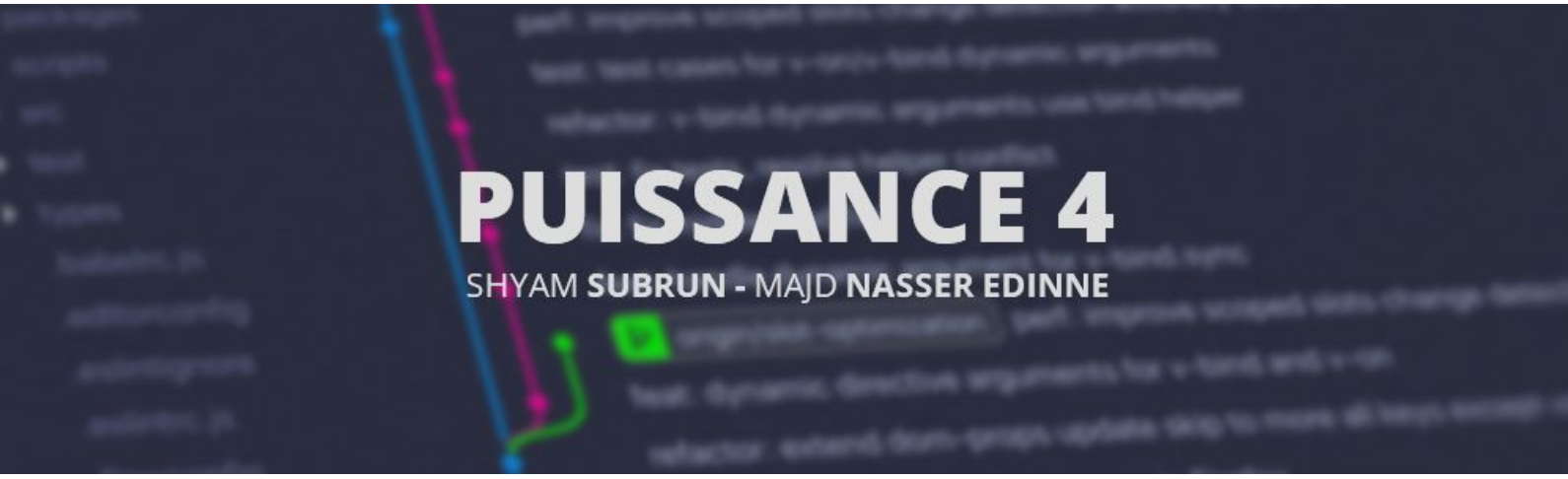
UPEC
Connaissance - Action

UNIVERSITÉ
PARIS-EST CRÉTEIL
VAL DE MARNE



RAPPORT PROJET
PUISSANCE 4

Licence 2 Informatique - **Groupe 2A**

Majd NASSER EDDINE
Shyam SUBRUN

2020 / 2021

responsable de module
Lamia BENAMARA

Sommaire

Présentation

Le Puissance 4

Le jeu du Puissance 4 se joue à 2 où les joueurs se confrontent sur une grille de 6 lignes par 7 colonnes. Chacun possède des jetons de couleur différente et ils jouent tour à tour. Le but est de réussir à aligner 4 pions de la même couleur pour gagner, que ce soit horizontalement, verticalement ou diagonalement. La particularité de ce jeu est toute simple : la gravité. On ne peut placer un pion que sur l'une des 7 colonnes et ce pion doit prendre la ligne la plus basse possible. Beaucoup de stratégie et un peu de concentration sont nécessaires pour être le premier à aligner ses pions !

Introduction

Au cours de l'année 2020, en programmation orientée objet, qui se déroule dans le module Java organisé par M. BENAMARA, nous avons vu les bases du langage Java, c'est-à-dire la création de classes et d'instances, la notion de mutabilité, les paquetages, les exceptions, les API, la javadoc ainsi que les méthodes. Nous avons décidé de nous mettre ensemble car tout au long de notre première année de licence ainsi que la deuxième, nous avons travaillé ensemble ce qui nous a permis de se comprendre assez vite quand l'un de nous souhaitait exprimer une idée vague initialement et nous sommes à l'aise quand il s'agit de travailler ensemble. Nous nous complétons aussi pas mal.

Notre projet

Il nous a été demandé de créer notre propre Puissance 4 en langage Java avec deux modes : joueur contre joueur et joueur contre ordinateur (intelligence artificielle). Le délai du travail était de 90 jours. À l'aide de nos connaissances acquises au cours de l'année ainsi que notre curiosité vis-à-vis du codage nous avons réussi à développer ce qui nous a été demandé dans les délais impartis.

Choix de programmation (structure de données)

Nous avons fait plusieurs choix concernant la programmation

- Il n'a pas été jugé nécessaire l'utilisation d'interfaces
- 3 classes (en + du main) sont utilisées : Grille, Joueur, Position
- Tout l'aspect visuel & organisation se trouve dans le main
- L'ensemble des méthodes techniques se trouvent dans la grille

Les différentes classes

GRILLE

Cette classe comprenant 5 attributs, 1 constructeur et 13 méthodes est la plus importante de notre programme. Elle contient les méthodes qui font la majorité des calculs, notamment pour l'intelligence artificielle. C'est elle qui régit la façon dont la grille évolue et les actions à la fois du joueur et de l'intelligence artificielle, elle s'occupe aussi d'effectuer toutes les vérifications nécessaires (de victoire par exemple).

JOUEUR

Cette classe gère les joueurs, à la fois les humains mais aussi l'intelligence artificielle. Elle associe chaque joueur à un nom et à un jeton. Elle est très souvent utilisée, notamment dans le main, car c'est elle qui nous permet d'obtenir les informations sur notre joueur et de gérer les tours.

POSITION

Cette classe nous permet de gérer les positions d'une manière plus efficace et propre. Elle prend deux arguments simples : x & y. À l'aide d'un getter, on peut, sur nos autres classes, facilement gérer les positions que l'on appel généralement p (respectivement on a p.X() et p.Y() pour les positions).

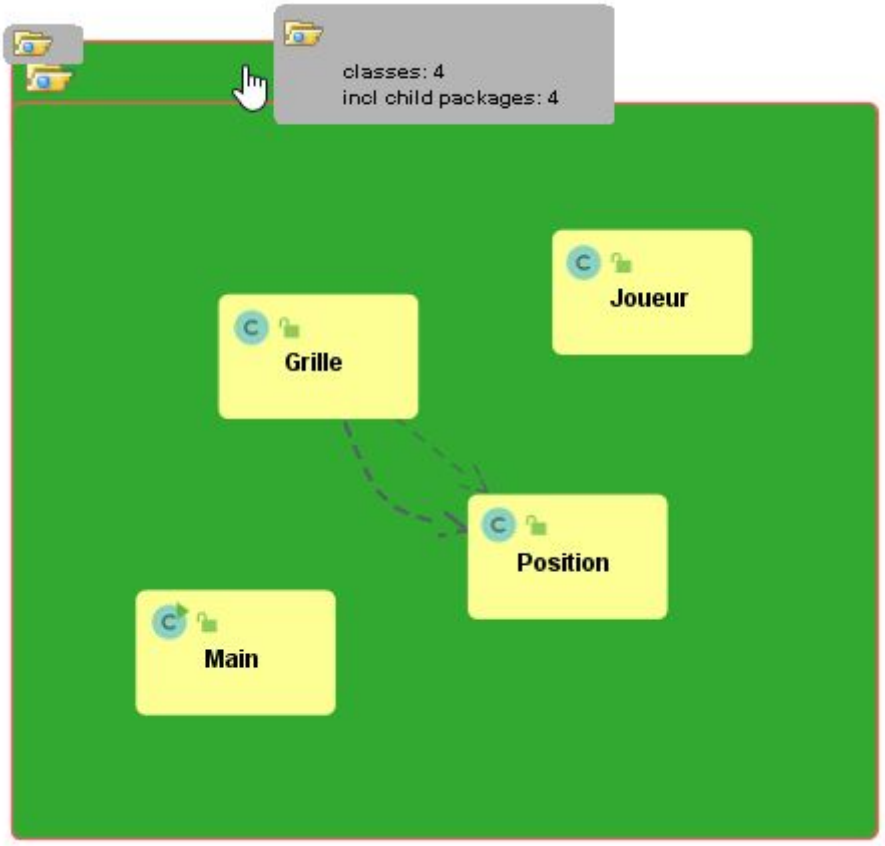
MAIN

Le main quant à lui va se dissocier en deux cas dès le début : humain vs humain et humain vs ordinateur. Les deux parties restent relativement similaire, utilisant notamment la même structure ci-dessous :

- On demande le ou les noms des joueurs
- On annonce le début de la partie
- On initialise le premier tour au premier joueur
- Si c'est au tour d'un humain de jouer
    - On lui affiche la grille actuelle
    - On lui demande de choisir une colonne
    - On vérifie qu'elle existe (sinon on redemande)
    - On vérifie qu'elle n'est pas pleine (sinon on redemande)
    - On pose le jeton
- Si c'est au tour de l'IA de joueur
    - On suppose qu'il joue parfaitement sans faire d'erreur, on le fait jouer directement et on continue
- On vérifie que le joueur qui vient de jouer ne vient pas de compléter une ligne, une colonne ou une diagonale de 4 jetons
- On incrémente le nombre de tour et on vérifie que le nombre de tour max a été atteint ou pas (42)
- On change de joueur et on recommence le processus

On a fait le choix ici d'utiliser beaucoup de boucles et de conditions. Les boucles vont nous permettre de réitérer les demandes au joueur tant qu'il n'a pas rempli certaines conditions dont on a besoin, elles vont nous permettre aussi de réitérer les mêmes instructions pour chaque tour. Au besoin de les quitter (une fois que le joueur nous a bien rempli les conditions ou une fois que la partie est finie), on utilise un break, si on a besoin de recommencer une boucle on utilise un continue.

Diagramme de classe générale du programme



On peut voir qu'une interaction existe entre notre grille et notre classe position. On l'a réalisé de manière à simplifier certaines méthodes dans notre grille. Il n'y a que très peu d'interaction d'une manière générale car nous ne voulions pas créer des classes inutiles qui risquent de surcharger notre projet. Il n'a pas été jugé nécessaire par nous de créer certaines classes car cela aurait pu compliquer les choses. Cependant, cela permet d'avoir un diagramme clair et concis de notre réalisation.

Pour améliorer les choses, on aurait pu par exemple créer une classe spécialement dédiée à l'intelligence artificielle, une autre pour les vérifications liées aux positions et encore une autre pour l'aspect visuel de notre programme (au lieu de placer nos instructions dans le main).

Fonctionnalités les plus pertinentes

```java
// Méthode permettant d'afficher notre grille de jeu
// - pas de paramètre d'entrée
// - pas de valeur de retour
public void Afficher() {
    System.out.println();
    for(int i = 0; i < Hauteur; i++) {
        System.out.print("|");
        for(int j = 0; j < Largeur; j++) System.out.print(Contenu[i][j] + "|");
        System.out.println();
    }
    System.out.print(" ");
    for(int k = 1; k <= Largeur; k++) System.out.print(k + " ");
    System.out.println();
}
```

Méthode **Afficher()**

Cette méthode nous permet d'afficher notre grille de jeu dès que nous en avons besoin. Elle ne prend pas de paramètre et n'a pas de valeur de retour, elle affiche directement la grille à l'aide de System.out.println().

Elle fonctionne en deux parties :

- La première partie est une double boucle qui va parcourir tous les éléments de notre tableau Contenu et les afficher un-à-un, en veillant à séparer les colonnes à l'aide du caractère « | ».

- La seconde partie est une boucle simple qui affiche les chiffres de 1 à 7, permettant au joueur de correctement visualiser les colonnes dans lesquelles il s'apprête à placer son jeton.

```java
// Méthode permettant de vérifier l'alignement de 4 pions à partir d'une position
// - paramètre d'entrée : une position de départ de type POSITION
// - valeur de retour : est-ce que un alignement de 4 pions existent à partir de cette p
public boolean Verifier(Position p) {
    int x = p.X(); int y = p.Y();
    char jeton = Contenu[x][y];
    int ver = 0; int hor = 0; int diag1 = 0; int diag2 = 0;

    // Vérification verticale
    if(x-1 >= 0) if(Contenu[x-1][y] == jeton) if(x-2 >= 0) if(Contenu[x-2][y] == jeton)
    if(x+1 < Hauteur) if(Contenu[x+1][y] == jeton) if(x+2 < Hauteur) if(Contenu[x+2][y]
    if(ver >= 4) return true;
```

Méthode **Verifier()**

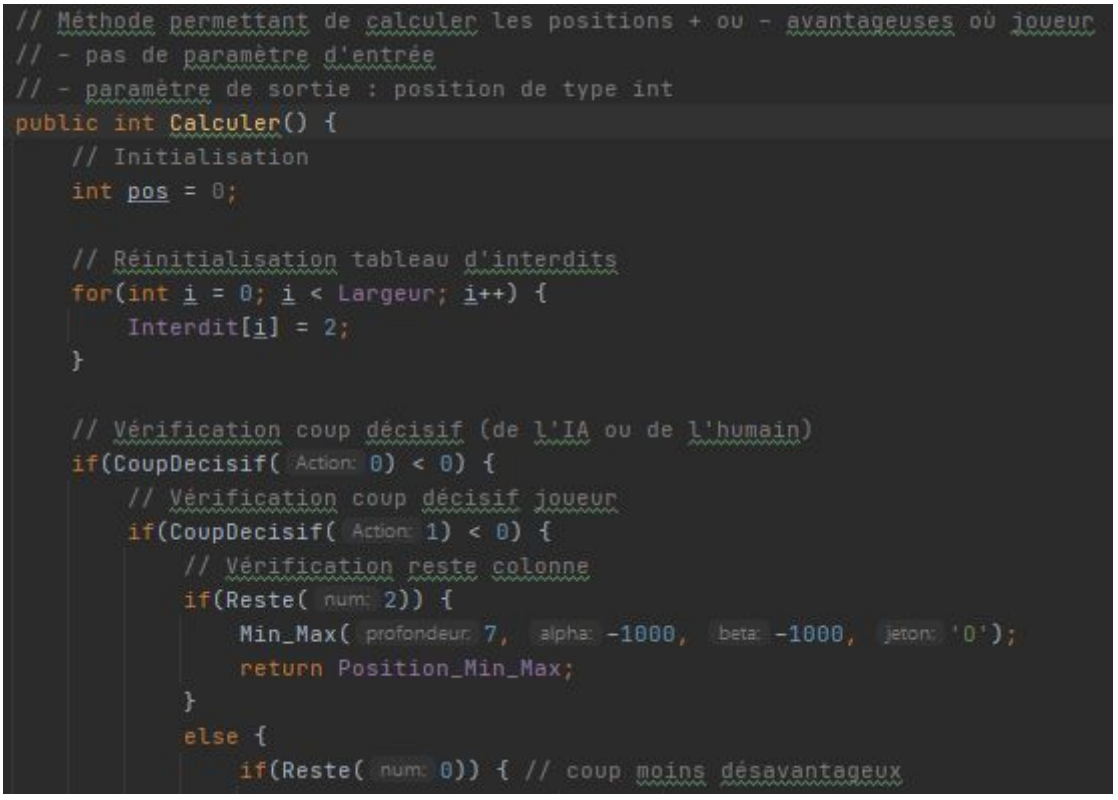
Cette méthode va nous permettre de vérifier si une suite de 4 jetons existe à partir d'une certaine position x y. Elle agit de la manière suivante : elle récupère le jeton placé aux coordonnées sur lesquelles on effectue la vérification, elle va vérifier les 3 cases au dessus de ce jeton pour voir s'ils correspondent à ce même jeton, puis elle va faire de même avec les 3 cases au dessous et va faire la somme des jetons consécutifs ayant respectés la condition. S'il y en a plus de 3, alors avec celui du milieu on a une suite de 4 jetons consécutifs. La méthode fait la même chose de manière horizontale et sur les deux diagonales. Elle retourne vraie si elle trouve une suite.

```java
// Méthode permettant de placer un jeton
// - paramètres d'entrée : colonne de type int & joueur de type JOUEUR
// - paramètre de sortie : ligne où le jeton a été placé, de type int
public int PlacerJeton(int x, Joueur joueur) {
    int y = Haut(x);
    if(y >= 0) Contenu[y][x] = joueur.getJeton();
    return y;
}
```

Méthode **PlacerJeton()**

Sûrement l'une des plus importantes, cette méthode permet de tout simplement placer un jeton pour un joueur à l'aide d'une colonne en entrée. Elle vérifie d'abord que la colonne n'est pas pleine puis elle place le

jeton en haut de cette colonne. Elle retourne la position verticale du jeton qui vient d'être placé, sinon elle retourne -1 si elle n'a pu rien placer.



```
// Méthode permettant de calculer les positions + ou - avantageuses où joueur
// - pas de paramètre d'entrée
// - paramètre de sortie : position de type int
public int Calculer() {
    // Initialisation
    int pos = 0;

    // Réinitialisation tableau d'interdits
    for(int i = 0; i < Largeur; i++) {
        Interdit[i] = 2;
    }

    // Vérification coup décisif (de l'IA ou de l'humain)
    if(CoupDecisif( Action: 0) < 0) {
        // Vérification coup décisif joueur
        if(CoupDecisif( Action: 1) < 0) {
            // Vérification reste colonne
            if(Reste( num: 2)) {
                Min_Max( profondeur: 7,  alpha: -1000,  beta: -1000,  jeton: 'O');
                return Position_Min_Max;
            }
            else {
                if(Reste( num: 0)) { // coup moins désavantageux
```

Méthode **Calculer()**

La méthode ci-dessus est la méthode maîtresse de l'ensemble du système de l'intelligence artificielle, c'est à partir d'elle que les méthodes AttribuerPoints(), Points_2Pions(), Points_3Pions(), CoupDecisif() et Min_Max() vont être appelées. Elle permet de calculer le coup de l'intelligence artificielle, de la manière la plus optimisée que nous avons pu faire. Elle retourne la colonne finale que l'intelligence artificielle doit jouer. C'est elle qui va aussi initialiser le tableau d'interdit, on peut parler de profondeur 0.

Répartition des tâches au sein de l'équipe

Shyam	Ensemble	Majd
Apparence visuel Coups décisifs Calculs meilleures positions Tests	Algorithme minmax Intégration alphabeta Gestion des positions Javadoc Rapport	Gestion des classes Vérification victoire Attribution des points Fonctions de base [inversement, haut, ...]

Nous avons eu des soucis à certain moment avec Gitlab (l'utilisation de l'interface git cmd étant assez complexe pour nous pour une première fois) et c'est pourquoi quelques fois nous avons été obligé de commit notre partie sur l'ordinateur de l'autre ou que certaines parties soient commits avec d'autres.

Globalement, étant donné que nous avons travaillé en méthode agile, une bonne partie des tâches les plus importantes ont été réalisées à deux, car elle était d'une complexité mathématique importante et il fallait que nous deux puissions comprendre la réalisation pour avancer sur les méthodes suivantes.

Difficultés rencontrées

Au tout début, nous devions trouver un EDI où nous pourrions exécuter le programme que nous développerons. Nous avons dans un premier temps essayé BlueJ mais cela n'a pas été suffisant pour nos besoins car nous devions lier l'EDI à GITLAB et avions besoin de plus de fonctionnalités. Après quelques recherches & essais, nous avons fait le choix d'utiliser IntelliJ Idea.

Connecter Gitlab à BlueJ n'a pas été une tâche facile, mais avec de la persévérance et beaucoup de recherche de documentation, nous avons réussi à les connecter pour que l'historique de notre avancement soit bien répertorié sur gitlab. Il nous a fallu par la suite prendre l'habitude de réaliser des commit & pushs régulier pour bien suivre l'avancement et ne pas être perdu entre nous.

Nous voulions nous voir pour travailler ensemble mais nous habitions à 2h de trajet l'un de l'autre et avec le couvre feu cela ne nous facilite pas la tâche. Donc nous avons privilégié les visioconférences avec ZOOM et Meet Jitsi où nous avons pu profiter de partages d'écran, transmissions de fichiers, liens et autres sources relativement facilement.

Pour l'étape du MinMax nous avons pris beaucoup de temps car cela a été fait en plusieurs fois, la difficulté principale était de reprendre l'avancement de l'autre après plusieurs heures voire plusieurs jours, que ce soit sur le plan du développement ou de l'interprétation des formules mathématiques. C'est pourquoi, il y a certains moments où nous étions dans l'obligation de s'accorder des réunions pour se replacer ou alors de travailler de manière agile en simultané, c'est-à-dire au même moment, de manière à avoir une meilleure compréhension du travail et un avancement plus rapide. Nous avons par ailleurs beaucoup utilisé un logiciel de prise de note où nous notions ce à quoi correspondait chacune des valeurs et/ou constantes que l'on utilisait de manière à ce que notre binôme puisse comprendre sans trop de difficulté ce que l'on développait.

Conclusion

Concernant la réalisation du jeu, nous avons respecté tout ce qui a été demandé, c'est-à-dire : les deux modes de jeu (joueur contre joueur et joueur contre ordinateur), l'utilisation de MinMax et AlphaBeta, (...).

Lors de notre réflexion durant ce travail, nos compétences en développement (particulièrement en langage Java) ont été accrue. Dorénavant, nous sommes capables de lire une documentation complète (en français ou en anglais), ce qui va nous permettre d'apprendre à interpréter une multitude d'autres langages de programmation à l'avenir.

Grâce à ce projet, nous avons aussi appris à utiliser Gitlab de manière intermédiaire, en assimilant par exemple les principes de push ou de commit. On comprend alors pourquoi cela est utilisé dans les grandes entreprises, lors de travaux d'équipe pour permettre à l'ensemble de l'organisation et de la hiérarchie de suivre l'avancement d'un projet, de ne pas être perdu, de pouvoir revert vers une version plus ancienne en cas de bug, de pouvoir mettre aisément à jour un programme, etc... .

Finalement, nous sommes plutôt satisfait du projet car nous avons pu respecter les délais impartis. La communication dans l'équipe a été parfaite, il n'y pas eu de confusion importante et l'interaction est restée globalement fluide.

Sources

Puissance 4

- https://fr.wikipedia.org/wiki/Puissance_4
- https://openclassrooms.com/forum/sujet/aide-tp-projet-puissance-4-78469
- https://stackoverflow.com/questions/35301814/puissance-4-game-in-javascript

Java doc

- https://openclassrooms.com/fr/courses/1115306-presentation-de-la-javadoc

IDE

- https://www.easypartner.fr/blog/les-meilleurs-ide-pour-developpeur-java/
- https://www.jetbrains.com/fr-fr/idea/download/#section=windows
- https://openclassrooms.com/fr/courses/4975451-demarrez-votre-projet-avec-java/4976266-demarrez-avec-un-ide
- https://riptutorial.com/fr/intellij-idea

Gitlab

- https://git-etudiants.lacl.fr/
- https://github.com/SocialGouv/tutoriel-gitlab
- https://romainlebreton.github.io/ProgWeb-CoteServeur/assets/initiation%20GIT_IUT_TP.pdf
- https://git-scm.com/book/fr/v2/Git-sur-le-serveur-GitLab

Ressources UPEC

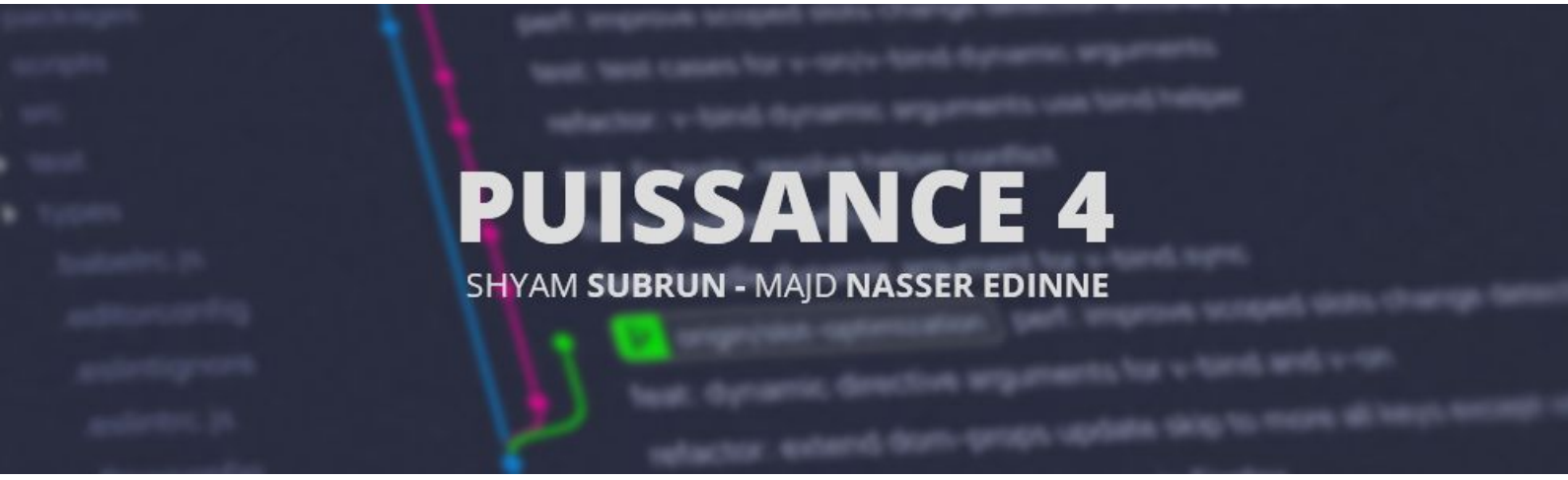
- http://lacl-upec.github.io/l2epo/
- http://lacl-upec.github.io/l2epo/2016/02/05/Objet.html

Aides MinMAX

- https://eprel-v2.u-pec.fr/pluginfile.php/73678/mod_resource/content/1/Doc_MinMax_AlphaB%C3%A9ta.pdf
- https://www.christian-schmidt.fr/puissance4
- https://openclassrooms.com/forum/sujet/minmax-puissance-4-65404
- https://www.di.ens.fr/~granboul/enseignement/mmfai/algo2003-2004/tp5/

UPEC
Connaissance - Action

UNIVERSITÉ
PARIS-EST CRÉTEIL
VAL DE MARNE



RAPPORT PROJET
PUISSANCE 4

Licence 2 Informatique - **Groupe 2A**

Majd NASSER EDDINE
Shyam SUBRUN

2020 / 2021

responsable de module
Lamia BENAMARA