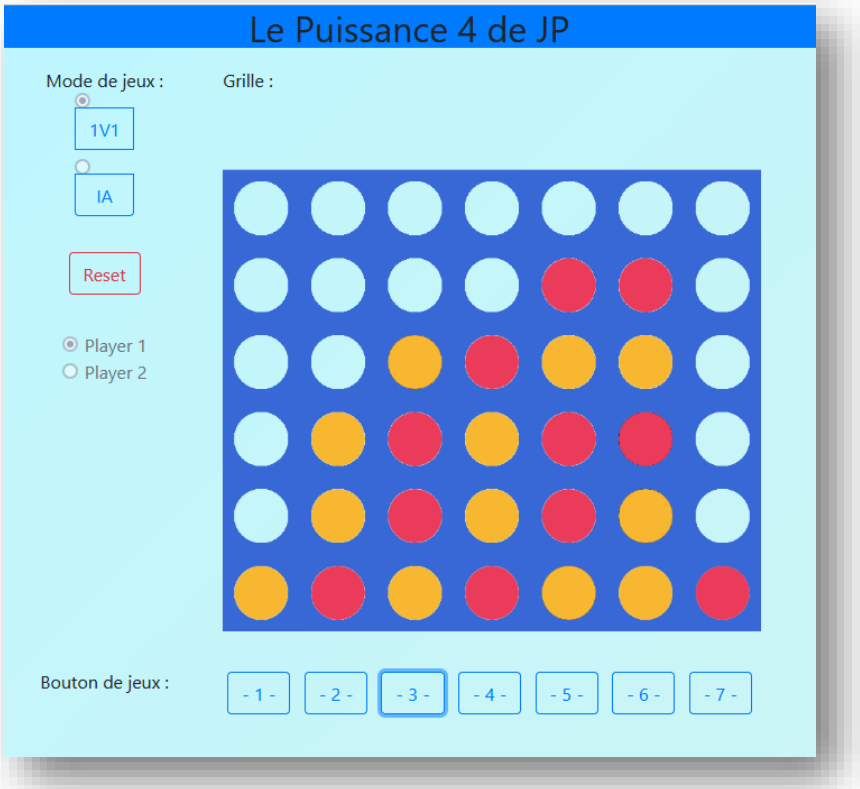
CAZALS Paul

LASERRE Julie



PUISSANCE 4

Développement Web – Janvier 2022

HTML / CSS / Javascript

Rapport de Projet

- Manuel d'utilisation
- Explication du code

Table des matières

1. Introduction

Dans le cadre du module « Développement web » nous avons développé le célèbre jeu du puissance 4. Au travers de ce rapport, nous allons dans un premier temps montrer comment fonctionne notre code, notamment les fonctions les plus importantes et comment utiliser le jeu (manuel d'utilisateur).

2. Cahier des charges

L'objectif de ce projet est de créer un jeu : le Puissance 4.
Le cahier des charges de notre projet est composé de plusieurs parties distinctes :

- Possibilité de jouer contre un adversaire humain.
- Créer une animation pour voir le jeton tomber et rebondir
- Possibilité de jouer contre une IA
- Implémenter l'IA en utilisant l'algorithme du minimax
- Possibilité de laisser l'IA commencer une partie
- Créer un écran (modal) pour afficher la victoire du joueur
- Avoir un panneau de contrôle pour gérer l'activation ou non de l'IA, laisser un joueur donné débuter la partie

3. Fonctionnement du code

3.1 – Mise en place du MVC

L'architecture de notre code doit respecter le modèle MVC, ce modèle permet d'organiser notre code en 3 grandes parties :

- Le modèle (Model) qui gère les données et le contenu algorithmique de notre jeu
- La vue (View) qui est la représentation visuelle du modèle.
- Le contrôleur (Controller) qui réalisé le lien entre l'utilisateur (la vue) et le système (le modèle)

3.2 – Model

Notre classe modèle possède plusieurs fonctions dont la fonction *checkWinner.* Cette fonction permet de vérifier chaque ligne, colonne, diagonale et antidiagonale afin de vérifier si un des 2 joueurs remporte la partie. Cette fonction est utilisée seulement dans le cas d'une partie où 2 joueurs s'affrontent.

En plus de cette fonction, nous avons créé une fonction *isgameover.* Cette fonction reprend la même fonctionnalité que la fonction *checkWinner.* Elle vérifie en faisant un contrôle sur les lignes, colonnes, diagonales et antidiagonales autour du jeton si le joueur ou l'IA remporte la partie. Cette fonction est donc uniquement utilisée dans le cas d'une partie contre l'IA car pour réaliser la vérification d'une victoire, cette fonction s'appuie sur les matrices copiées.

Nous avons créé une fonction *getPositionfromMove* permettant de tester le coup que l'IA souhaite réaliser et renvoie un potentiel tableau qui représente la grille de jeu avec les jetons déjà en place dans la grille et le jeton que l'IA souhaite jouer. Ces tableaux ainsi générés seront utilisés dans la fonction *getBestmove* afin de vérifier que le coup que l'IA souhaite jouer est le meilleur ou non.

Le model contient aussi une fonction *getBestmove,* cette fonction permet de choisir le meilleur coup que peut jouer l'IA en fonction du score en s'appuyant sur l'algorithme minimax. L'IA réalise des scores test en positionnant facultativement un jeton dans la grille, le score avec ce jeton afin de voir si c'est le meilleur coup à réaliser. Ce score test est ensuite comparé au score maximum, si celui-ci lui est supérieur c'est qu'il s'agit du meilleur mouvement à réaliser et l'IA va donc pouvoir jouer son coup.
Ce modèle contient aussi une des fonctions les plus importantes (si ce n'est la plus importante) celle du minimax. L'algorithme minimax est un algorithme qui consiste à minimiser la perte maximum. C'est-à-dire, dans le cas de notre puissance 4, l'algorithme va chercher à réaliser le meilleur coup possible. Pour cela l'algorithme va réaliser un coup fictif puis calculer si ce coup est le meilleur et aussi calculer si les coups suivants lui permettront de gagner ou de s'approcher de la victoire.

Enfin une des dernières fonctions importantes de notre model est la fonction *evaluation*, cette fonction renvoie un score pour un joueur en fonction du plateau joué. L'évaluation de ces scores se fait sur une rangée de 2 jetons ou une rangée de 3 jetons. Plusieurs boucles for

permettent de calculer le score avec le jeton fictif ajouté par l'IA. Détailler toutes les boucles va être très long, il vaut mieux aller voir directement le code de cette fonction.

3.3 – View

Cette partie est donc la représentation visuelle de notre puissance 4. C'est donc dans cette partie que nous avons créé notre bouton reset qui permet de mettre à jour la grille et de faire disparaître tous les jetons de la grille afin de recommencer une nouvelle partie.
La View contient aussi l'initialisation de notre jeton avec la déclaration de sa taille et de sa vitesse ainsi que de sa couleur qui change en fonction du tour et donc du joueur qui doit jouer.

La View regroupant la partie visuelle c'est donc ici que nous avons créé notre fonction *drawTableau.* Cette fonction dessine notre tableau avec la largeur et la hauteur qui lui ai imposé et la couleur (#3867d6). Cette fonction permet de créer un carré de couleur bleu. Pour créer une grille il faut donc créer les trous qui servent d'emplacement pour les jetons. Pour les créer nous avons écrit une fonction *clearCircle* afin de créer des cercles de la même couleur que la couleur de fond de notre page.

Dans notre view il y a une autre fonction importante, la fonction *affichage.* Cette fonction permet de gérer l'affichage de notre grille ainsi que l'animation de la chute et du rebond du jeton. C'est dans cette fonction que nous avons défini l'animation (sa chute ainsi que son rebond) et le temps de durée de l'animation. Afin de rendre l'animation plus réaliste nous faisons chuter le jeton avec une vitesse initiale, nous le faisons rebondir et perdre de la vitesse et enfin nous mettons le jeton à sa place dans la grille.

Le cahier des charges indiquait qu'il fallait pouvoir choisir quel joueur commence la partie. C'est pour répondre à ce besoin que nous avons notre fonction *changePlayer.* Dans cette fonction nous récupérons l'état de nos boutons players et de notre bouton IA afin de déterminer dans quelle configuration le joueur veut jouer. Il peut choisi quel joueur commence ou alors décider d'affronter l'IA.

Lorsque la partie a démarrée, on ne peut plus sélectionner le joueur qui doit commencer (cela semble logique). C'est donc pour cela que nous avons créé notre fonction *disablechangeplayer.* Cette fonction permet de griser les boutons de choix du joueur dès lors que la partie a débuté.

3.4 – Controller

Le controller réalise le lien entre l'utilisateur (view) et le système (model). Le controller contient un *constructor* qui permet de tracer notre grille (le tableau bleu que nous créons plus les cercles de la couleur du fond). Pour tracer cela, notre *constructor* fait appel aux fonctions *drawTableau* et *ClearCircle* présentes dans notre *view*.

Dans le controller nous avons aussi une fonction *bindFunction* qui gère toute la partie jeu. C'est-à-dire que c'est ici que nous gérons le choix du joueur qui commence la partie, du

changement de tour (avec le changement des couleurs des jetons) et aussi que nous gérons le tour de jeu de l'IA.

En effet nous avons notre dernière fonction : *afterMove* qui permet de gérer le tour de l'IA. Si le bouton IA est coché et que le joueur veut l'affronter, cette fonction entre en jeu et permet de gérer le changement de tour avec le changement de la couleur des jetons et gère l'animation de la chute du jeton dans le cas où c'est l'IA qui joue.

3.5 – HTMS & CSS

Notre fichier JS qui contient le MVC est le fichier qui permet de faire fonctionner notre jeu. Mais en plus de cela nous avons 2 autres fichiers permettant de gérer le côté graphique de notre puissance 4.

Nous avons un fichier HTML qui s'appuie sur le boostrap pour créer notre page et nos boutons permettant de choisir le style de jeu et surtout les boutons de jeu qui sont présents en dessous de la grille. A l'aide du boostrap nous avons aussi pu ajouter le titre de notre jeu avec son bandeau de couleur et aussi notre fond animé avec son dégradé de couleur bleu qui défile pendant toute la durée du jeu.

Afin de gérer le côté graphique nous avons aussi un fichier css : *style.css.* Ce fichier permet de gérer la police d'écriture de notre titre ainsi que la taille de nos différentes parties de la page de jeu. En effet nous avons un cadre jeu qui contient notre grille qui est elle-même définie dans un cadre tableau et qui nous permet ainsi de définir la taille de notre grille de jeu et son alignement sur la page.

4. Manuel d'utilisation

L'utilisation de notre puissance 4 s'avère simple et intuitive. La page est responsive ce qui permet de jouer facilement sur smartphone. Elle a été pensée pour que chacun puisse l'utiliser de manière agréable. Toute fois et comme dans tout projet, il est nécessaire de faire un manuel d'utilisation.

4.1 – Accès à l'application Z

Pour accéder à notre jeu rien de plus simple, il suffit de se rendre sur le lien suivant :

paulcazals.fr/P4JP

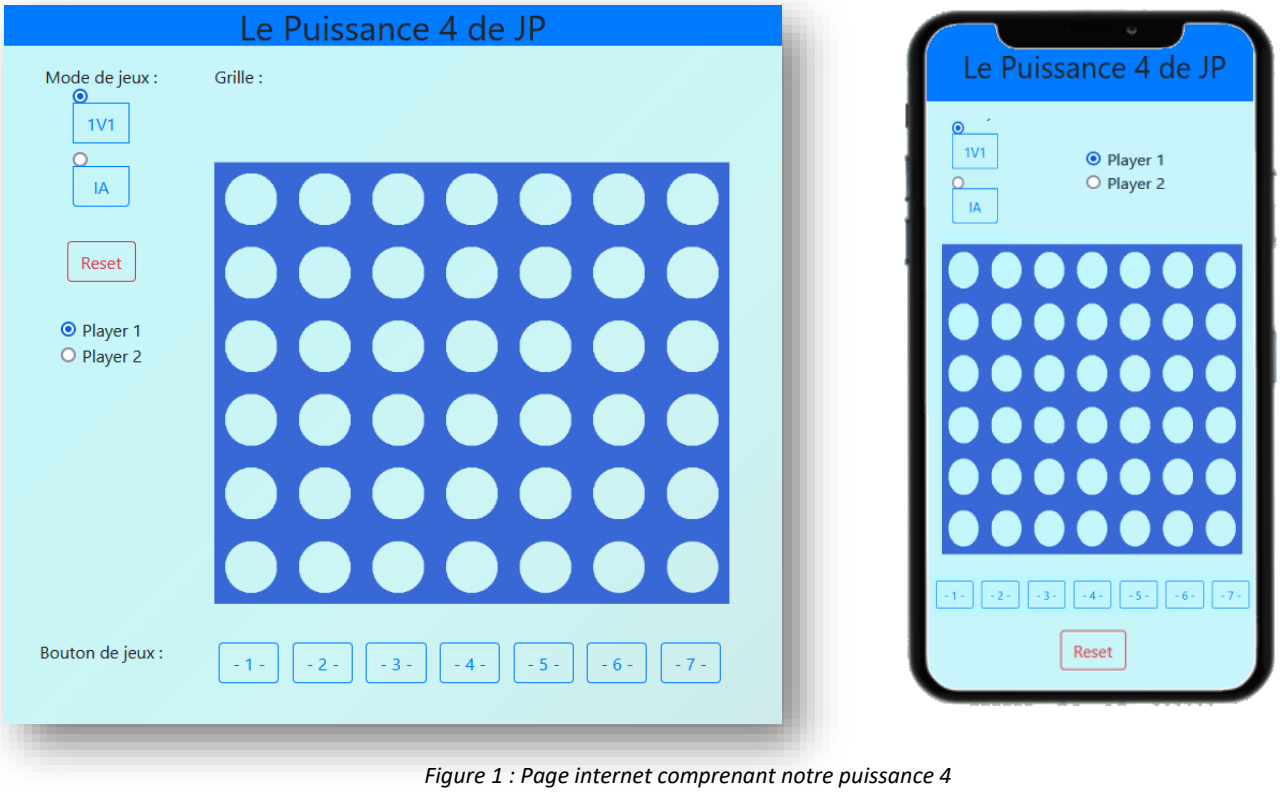


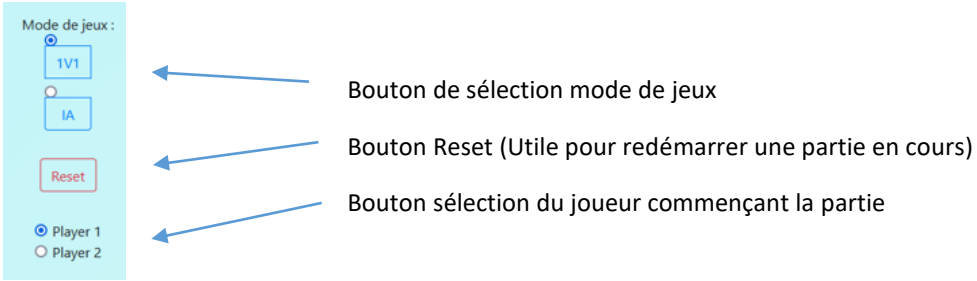
Figure 1 : Page internet comprenant notre puissance 4

Nous arrivons donc sur une page internet digne des plus grandes qui nous permet de jouer à ce jeu merveilleux.

4.2 – Système de jeu

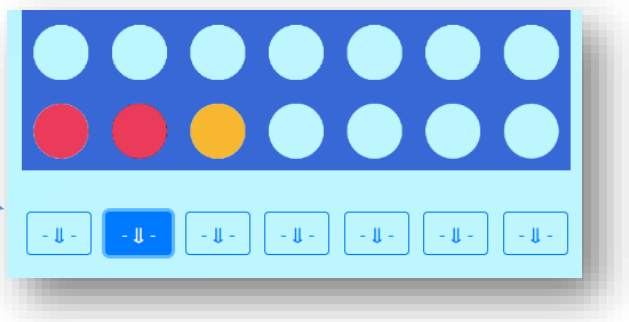
Pour jouer, il suffit de sélectionner le mode de jeux (Bouton radio sur le côté). Nous avons le choix entre le mode **1V1** et le mode avec **IA**.

- Pour le mode **1V1**, sélectionner le bouton radio correspondant, sélectionner quel joueur démarre (Par défaut le Player 1 / couleur rouge commence) si le joueur 2 souhaite commencer il faut sélectionner « Player 2 » (couleur jaune commence).



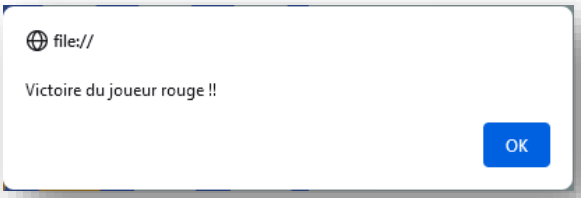
Pour positionner les jetons dans la grille, il suffit de cliquer sur les boutons de jeux sous la grille. Pour mettre un jeton dans la première colonne, appuyé sur le premier bouton. Pour la deuxième colonne appuyer sur le 2ième bouton, etc.

Rampe de bouton permettant de stocker les jetons dans la grille



Les boutons de sélections de joueurs indiquent à qui est le tour.

Une fois qu'un des deux joueurs aligne 4 jetons (verticalement / horizontalement / diagonalement) un message affiche qui a gagné la partie, après avoir cliqué sur « OK » la partie se remet à 0.



- Pour le mode **IA,** il suffit de sélectionner le mode IA et de jouer le premier jeton. L'IA jouera automatique son jeton et les suivants.

- Le bouton **RESET** peut être utilisé pour interrompre et recommencer une partie à tout moment de celle-ci.

5. Axes d'améliorations

Finalement notre jeu est totalement opérationnel et répond à une bonne partie du cahier des charges.
Nous avons la possibilité de faire jouer 2 joueurs qui peuvent s'affronter en choisissant lequel d'entre eux va débuter la partie. Il y a aussi la possibilité pour un joueur d'affronter une IA qui s'appuie sur l'algorithme minimax. Notre code est aussi en format MVC comme il nous a été demandé de la faire et nous avons également l'animation du jeton qui tombe et qui rebondit lorsque celui-ci est placé sur la grille.

Cependant il reste quelques axes d'améliorations que nous pouvons développer.
Premièrement on pourrait avoir la possibilité de choisir qui de l'IA ou du joueur débute la partie. Ce n'est aujourd'hui pas le cas et lorsque nous avons essayé de le faire notre IA ne réagissait plus correctement.
Deuxièmement nous pourrions améliorer la façon de jouer. En effet pour placer les jetons sur notre grille il faut appuyer sur le bouton correspondant à la colonne dans laquelle nous voulons ajouter notre jeton. Or nous pourrions ajouter un jeton directement en cliquant sur la colonne de la grille où nous voulons l'ajouter. Et en plus de cela nous pourrions créer une animation sur les jetons au survol de la souris sur la grille. Lorsque la souris survol la grille nous pourrions placer le jeton qui doit être joué au-dessus de la grille et le déplacer selon la colonne que la souris survol afin de bien visualiser sur quelle colonne nous allons jouer.
Enfin le dernier axe d'amélioration qui n'est pas présent dans le cahier des charges mais qui peut être intéressant à développer est la difficulté de l'IA. En effet on pourrait choisir entre une difficulté facile, moyenne et difficile. Pour cela il faudrait changer le paramètre de profondeur dans l'algorithme minimax et mettre en place un système de choix.
Tout ces améliorations n'ont pas été effectué par manque de temps. Ayant plusieurs projets en parallèle nous nous somme concentrées sur le cahier des charges voulu initial.