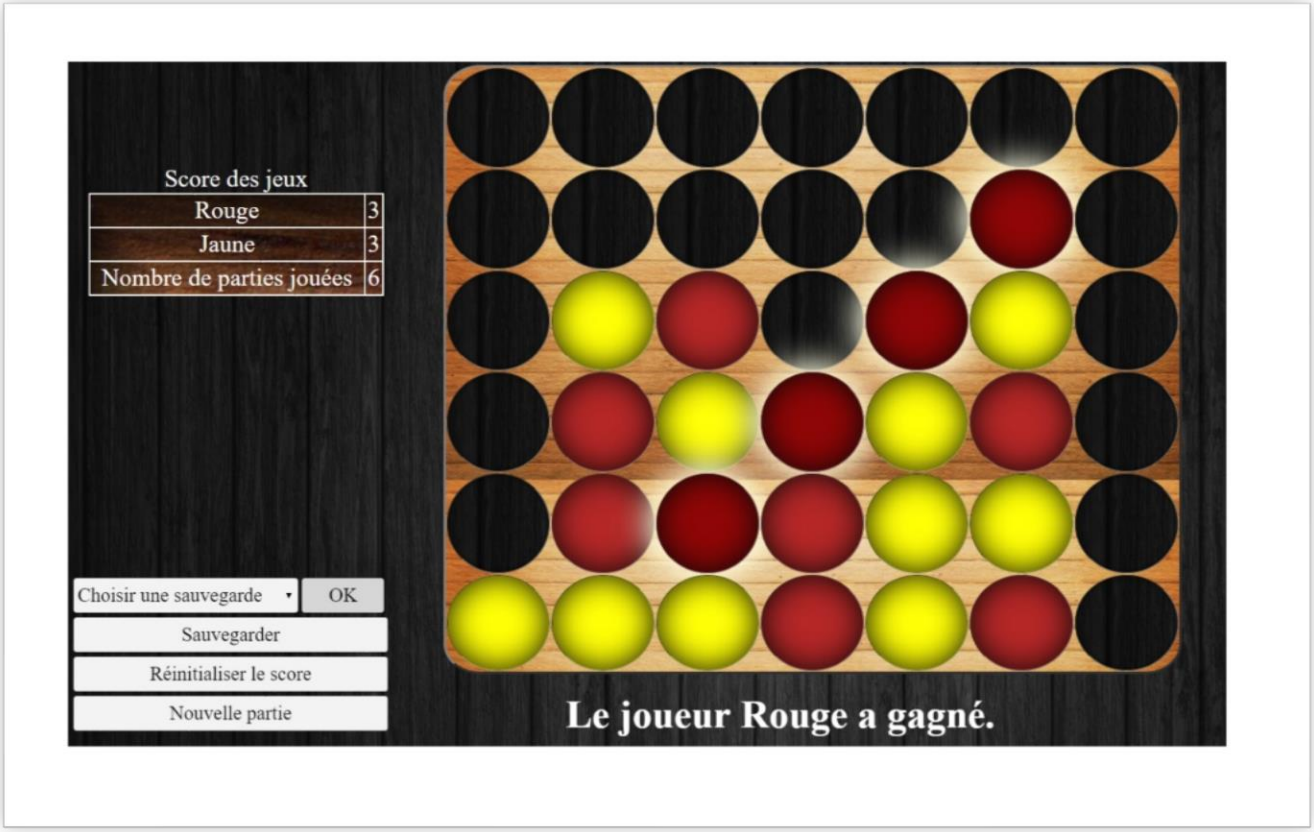

Rapport de Projet Programmation Web avancée côté client

LE JEU DE PUISSANCE 4

Yohann Tognetti
Yasmine Ben Fredj

2018-2019

Table des matières :

❖ Introduction :

Le Puissance 4 est un jeu de **stratégie**, commercialisé pour la première fois en 1974 par la Compagnie Milton Bradley, connue sous le nom de MB et détenue depuis 1984 par la société Hasbro. Il se joue par deux personnes et le but est d'aligner une suite de 4 pions de même couleur sur une **grille** comptant **6** rangées et **7 colonnes**. Chaque joueur dispose de 21 pions d'une couleur (par convention, en général **jaune ou rouge**). Tour à tour, les joueurs placent un pion dans la colonne de leur choix, le pion coulisse alors jusqu'à la position la plus basse possible dans la colonne ensuite c'est à l'adversaire de jouer.

Le vainqueur est le joueur qui réalise un alignement (horizontal, vertical diagonal), consécutif d'au moins quatre pions de sa couleur. Et si toutes les cases de la grille de jeu sont remplies et qu'aucun des deux joueurs n'a réalisé un tel alignement, la partie est déclarée nulle.

❖ Rapport :

Notre projet Java-script est constituer d'un fichier HTML, un fichier JavaScript et un fichier CSS.

I. JavaScript :

Pour commencer, dans notre fichier « Puissance.js », on a défini notre fonction principale c'est un constructeur qu'on a nommé « grille ». Il va prendre en paramètre deux entiers « i » et « j » qui représentent le nombre des colonnes et nombre des lignes de la grille. Il va donc construire les propriétés suivantes :

- Un tableau « **array** » de (j x i) cases.
- Une variable « **ligne** » égale à i.
- Une variable « **colone** » égale à j.
- Une variable « **numJoueur** » initialiser à un (1).
- Une variable « **end** » initialiser à zero(0) .
- Une variable « **namesave** » qui représente le nombre ou le numéro des parties sauvegarder par les joueurs, initialiser à moins un (-1) car il n'y a aucune partie sauvegarder puis elle sera incrémentée à chaque sauvegarde.

- Une fonction « **cell(i,j)** » qui prend en paramètre le numéro de colonne « i » et « j » le nombre de ligne et qui renvoie dans « array » la cellule de cordonnés (i,j).

- Une fonction « **posToId(i,j)** » qui prend en paramètre « i » et « j » et qui identifie la cellule de cordonné (i,j).

- La fonction «**modifcell (i,j,value)** » qui prend en paramètre « i », « j » et « value », elle change la valeur dans la cellule (i,j) par « value ».

- La fonction «**testjoueur (i,j)** » qui prend en paramètre « i »et « j » qui va seulement tester le placement libre du joueur. Si c'est le premier joueur on ajoutera au tableau array la valeur « X », sinon elle prend la valeur « O » avec la fonction « modifcell ». Elle modifie au passage la valeur dans la variable « numJoueur ».

- Fonction « **jouer (j)** » qui prend en paramètre un joueur « j », Lorsque le joueur « j » clique sur une colonne, elle joue son pion du joueur.

  Si le jeu n'est pas terminé (this.end==0) :

  - Le pion va coulisser tant que les cellules sont vides et le nombre de ligne est supérieur ou égale à 0.
  - Si c'est le 1er joueur on change la valeur de la cellule par « X » sinon par « O ».

  Cette fonction définie avec CSS la forme des pions (cell.style=):

  - X : Cercle Rouge
  - O : Cercle Jaune

- Une fonction « **verif(i,j)** » qui prend en paramètre « i » et « j » qui va vérifier l'alignement des pions et déclare le gagnant. Cette fonction contient un constructeur « compteurAligner » qui prend lui-même en paramètre :
  - Deux nombres « istart » « jstart » qui représente la ligne et la colonne du début de l'alignement :
    - Pour un décalage vertical : (istart,jstart)=(i ,0)
    - Pour un décalage horizontal : (istart,jstart)=(i ,j)
    - Pour un décalage Diagonal- : « istart » va augmenter et « jstart » va diminuer tant qu'ils restent dans la grille.
    - Pour un décalage Diagonal- : « istart » va diminuer et « jstart » va augmenter tant qu'ils restent dans la grille.
  - Deux nombres « pasi » et « pasj » qui représente le décalage de l'alignement qui va être effectuer :
    - (pasi,pasj)= (0,1) : décalage Vertical
    - (pasi,pasj)= (1,0) : décalage Horizontal
    - (pasi,pasj)= (-1,1) : décalage Diagonal-
    - (pasi,pasj)= (1,1) : décalage Diagonal+
  - Un objet « obj » qui est la grille du jeu.

  Cette fonction vérifie à chaque tour s'il y a 4 cellules consécutifs de même couleur verticalement, horizontalement ou dans la diagonale. Si c'est bien le cas, elle changer la couleur de l'arrière-plan (CSS) des cellules sélectionner pour avoir un effet fusionner. Elle déclare bien évidement la fin de la partie et le joueur gagnant.

  Si la partie jouer (donc fini) est sauvegarder, elle va la retirer de la liste des sauvegardes.

- Une fonction « **affiche()** » qui va non seulement permettre d'afficher et mettre en page la grille et les cellules (Rouge / Jaune / Vide), mais aussi de prendre en compte l'action du clic sur les cellules. Elle contient une fonction « ClickHandler » qui va détecter l'action d'un click.

- Une fonction « **Save()** » qui va permettre à l'aide de la méthode «localStorage.setItem()» d'enregistrer une nouvelle sauvegarde dans la liste des parties .

- Une fonction « **setButton()** » qui va créer les boutons :
  - « **Sauvegarder** » : pour sauvegarder une partie.
  - « **Nouvelles partie** » : pour commencer une nouvelle partie.
  - « **Réinitialiser le score** » : pour réinitialiser le score à zéro.
  - « **OK** » : pour valider une sélection de parties sauvegarder.

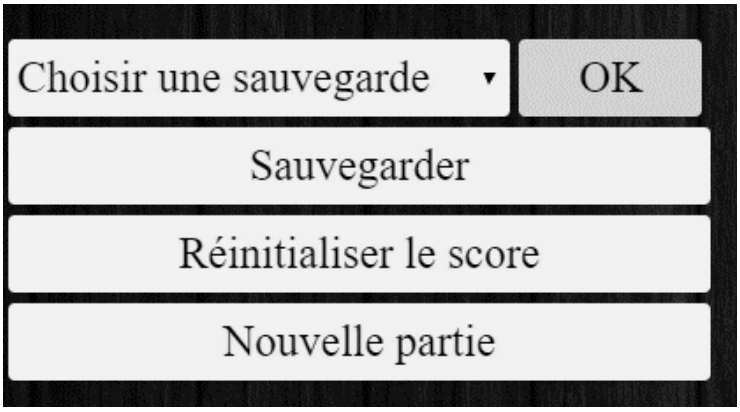

Figure 1

Et le select « **Choisir une sauvegarde** » dans lequel on trouve les parties sauvegarder.

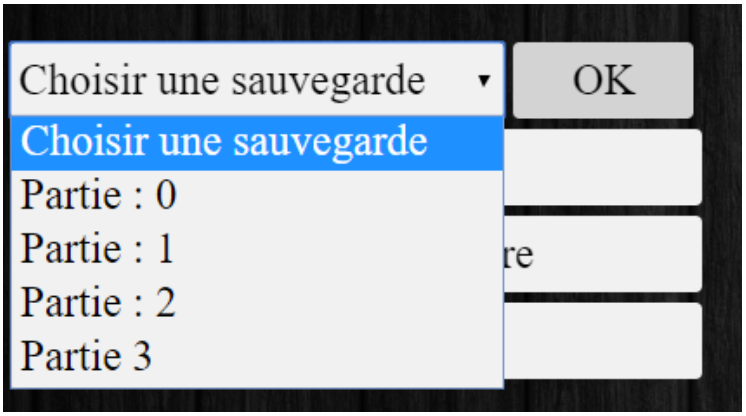

Figure 2: Example

On définit aussi les fonctions suivantes :

- Une fonction « **updatescore** » qui prend en paramètre le joueur gagnant dans une partie donnée et qui va incrémenter le nombre des parties gagner par ce joueur.

- Une fonction « **Setupscoretable()** » qui crée un tableau de score appelé « Score des jeux » constituer de 3 lignes et 2 colonnes. Ce tableau nous informe du score des joueur (parties gagner par chaque joueur) et le nombre de parties jouer.

Figure 3

- Une fonction « **Affichescore()**» qui va afficher le score des joueur et le nombre de parties joués.

- Une fonction « **afficheinformation(numjoueur)** » qui va afficher sur l'écran le tour du joueur pris en paramètre.

- Une fonction « **newGame (obj)** » qui prend en paramètre un objet qui va être « grille », une partie du jeu. Cette fonction supprime l'objet et crée un nouveau puis l'affiche.

- Une fonction « **Setup (parti)** » qui va commencer une nouvelle partie si les joueurs ne choisissent aucune partie sauvegarder, sinon elle va reprendre la partie choisie par les joueurs.

- Une fonction « **charger (obj)** » qui va charger la partie sauvegarder choisi par les joueurs ou bien la nouvelle partie.

II. CSS :

Le fichier « Puissance.css » décrit le style de présentation de notre jeu de **puissance4**.

III. HTML :

Notre fichier « Puissance.html »ne contient qu'une construction basique HTML, ils appellent les fichier « Puissance4.js », « Puissance4.css » et exécute la fonction « setup(-1) » pour commencer une parties de jeu.

❖ Conclusion :

Notre jeu permet à deux joueurs de jouer en s'alternant devant une même page web. On a ajouté un tableau « score des jeux » qui affiches le nombre des parties jouer ainsi que le score de chacun des joueur 'rouge' ou 'jaune'.

Le bouton « sauvegarder » permet aux joueurs de sauvegarder un ou plusieurs parties commencer et de les récupérer avec le sélecteur « choisir une sauvegarde »et le bouton « Ok » pour valider. Chaque partie sauvegarder sera supprimer seulement si elle est reprise et terminer.

Le bouton « Réinitialiser le score » supprime les données du tableau et les initialise à zéros. Puis le bouton « Nouvelle partie » qui recommence la partie à chaque fois que le jouer clique dessus sans affecter la sauvegarde.

On aurait pu rajouter d'autre fonctionnalité tel que :

- Le jeu contre l'ordinateur (machine-learning).
- Des élément graphique et animations…

Ce projet a été très intéressant, il nous a permis de mieux connaitre le fonctionnement du langage de programmation Java-script.

❖ Référence :

- https://fr.wikipedia.org/wiki/Puissance_4
- https://developer.mozilla.org/fr/docs/Web/JavaScript
- https://openclassrooms.com/fr/courses/2984401-apprenez-a-coder-avec-javascript
- https://www.rapidtables.com/web/css/css-color.html#red
- https://developer.mozilla.org/fr/docs/Web/CSS/