Marro Sébastien 21603380
Youssef Houssine 21601530

RAPPORT DE PROJET

Pour commencer, notre sujet de projet était le suivant: réaliser un jeu Puissance 4.
Dans ce rapport, nous allons, dans un premier temps, expliquer quels choix ont été pris afin de réaliser ce projet. Nous parlerons ainsi des choix de conception et des différentes fonctions que nous avons écrites.
Ces fonctions seront brièvement détaillées, afin de mieux comprendre dans quels objectifs celles-ci ont été faites et comment elles sont utilisées.
Dans un second temps, nous aborderons les difficultés que nous avons rencontrées, et quelles ont étés les solutions apportés pour résoudre ces obstacles.

I- Choix de conception

Dans un premier temps, nous avons créé un constructeur **Grille(i,j)**.
Ce constructeur contient une variable **this.player**, qui correspond au numéro du joueur, initialisée à 1 au début et prendra, en fonction de qui doit jouer, la valeur 1 ou 2.
Une autre variable **this.end**, qui permet de définir si la partie est finie ou non. La partie est finie quand la variable est égale à 0.

Nous avons ensuite défini **this.matriceDeJeu** comme étant un **Array** de **Array**, qui correspond à une matrice de i lignes et j colonnes. Chaque élément de notre Array est initialisé à 0.
Nous avons décidé de créer une telle matrice afin de faciliter l'accès et la modification de chaque élément de l'Array.

Nous avons ensuite créé 9 accesseurs au prototype du constructeur Grille que vous trouverez ci-dessous avec quelques explications.

Grille.prototype.pos :
Prend 2 paramètres i et j.
Retourne l'indice de l'élément de notre Array qui sera compris entre 0 et 41. Elle sera utile pour l'utilisation du clic de la souris et définir un id à chaque **<td>** de la **<table>** utilisée plus tard.

Grille.prototype.cell :
Prend 2 paramètres i et j.
Retourne l'élément de notre Array correspondant à la case de coordonnées (i, j).

Grille.prototype.modifcell :
Prend 2 paramètres i et j.
Regarde quel joueur doit jouer.
Change la valeur de *this.matriceDeJeu[i][j]* qui devient 3 si c'est au premier joueur de jouer et 5 sinon.
Ensuite, on vérifie l'alignement avec l'appel à *this.testligne(i,j)*.
Puis on appelle le constructeur modifColor(i,j).

Grille.prototype.modifColor :
Prend 2 paramètres i et j.
Récupère l'élément <td> par son id.
Modifie la couleur de fond de l'élément en fonction de la valeur de *this.matriceDeJeu[i][j],* en rouge si la valeur est 3 et jaune si la valeur est 5.

Grille.prototype.jouer :
Prend 2 paramètres i et j.
Regarde si la partie est en cours à l'aide de *this.end.*
Si elle est en cours, teste si la valeur de *this.matriceDeJeu[**5**][j]* est égale à "0". Si c'est le cas, on appelle l'accesseur *this.modifcell(5,j)*. Sinon teste si la valeur de *this.matriceDeJeu[**4**][j]* est égale à "0". Si c'est vérifié, on utilise *this.modifcell(4,j)*. Sinon on continue le même processus jusqu'à la vérification de *this.matriceDeJeu[**0**][j]*.

Il y a donc 6 vérifications au maximum. Celles-ci permettent de bien placer les pions le plus bas possible sur la bonne verticale.

Grille.prototype.testligne :
Prend 2 paramètres i et j.
Cet accesseur teste trois fonctions :
testHorizontal = function(x,y)
testVertical = function(x,y)
testDiagonal = function(x,y)

Ces fonctions utilisent plusieurs sous-fonctions regardant la case cliquée et 3 cases à côté, en diagonales ou 3 cases en dessous, en respectant certaines conditions pour éviter les erreurs. Et retourne une valeur.
Cette valeur peut être: 3, 5, 6, 8, 9, 10, 12, 13, 15, 18 et 20.
Ces valeurs correspondent à la somme des 4 cases regardées
Les trois fonctions retournent 1 si une des sous-fonctions retourne soit 12 soit 20.

On a ensuite un if statement qui regarde si une des 3 fonctions retourne 1 et si c'est le cas, this.end prend la valeur 1 (partie terminée). Puis affiche une alerte qui annonce le joueur gagnant.

Grille.prototype.afficher :
Déclaration de la fonction onClick, qui permet de pouvoir cliquer sur une des cases souhaitée pour jouer sur la verticale correspondante en appelant le constructeur jouer(i,j).
Ensuite, on récupère la table avec l'id "grille", on crée ensuite un 6 de tr et 7 td dans chacun des tr.
Chaque td prend ensuite un id avec l'appel à this.pos(i,j) puis on lui ajoute aussi la fonction onclick qui prend la valeur onClick(td,this,row,col) pour pouvoir cliquer sur chaque td.
On ajoute ensuite un style à la table et au tr et td.

Grille.prototype.recommencer :
Relance la partie, réinitialise this.end, this.player this.matriceDeJeu[i][j] à leur valeur de base.
Nettoie la sauvegarde locale avec localStorage.clear().

Grille.prototype.sauvegarder :
Sauvegarde la Grille en utilisant le localStorage et JSON.stringify.
Cette sauvegarde ne peut se faire que lorsque la partie n'est pas terminée.
Si la partie est terminée (par une victoire), nous avons décidé de ne pas pouvoir sauvegarder car cela ne sert pas puisqu'on ne peut pas rejouer après.

II- Problèmes rencontrés et solutions apportées

Tout d'abord, nous avons eu un problème au niveau de la fonction pour cliquer.
Lorsque nous cliquions, nous n'avions pas le résultats attendus (mauvais td pris en compte).
Nous avons résolus ce problème en ajoutant la méthode onclick à chaque td et lui donnions comme valeur la fonction onClick que nous avons créer.

Un second problème a été le test d'alignement, nous ne parvenions pas à trouver une méthode de test avec un Array de 42 éléments. Pour résoudre ce problème, nous avons décidé de redéfinir notre Array de 42 élements en Array d'Array.