

Introduction

Enoncé

ANALYSE ET DEVELOPPEMENT DU JEU DEMINEUR

Principe du jeu

Le jeu comporte une grille avec des cases.

Certaines cases comportent aléatoirement des mines, les autres indiquent le nombre de mines adjacentes.

L'objectif est de découvrir toutes les cases de la grille non minées, avec pour seule indication le nombre de mines dans les zones adjacentes aux cases découvertes.

Conditions

Le jeu ne compte que 2 conditions.

Quand le joueur clique sur une case il peut soit découvrir une mine, soit un chiffre (indiquant le nombre de mines adjacentes).

Dans le premier cas, le joueur perd. Sinon le jeu continu jusqu'à ce que le joueur ait découvert toutes les cases non minées.

S'il parvient à découvrir toutes les cases non minées, la partie est gagnée.

Cahier des charges

Dans le cadre de ce travail, voici toutes les options qui devaient être implémentées.

L'initialisation des mines devait être aléatoire et le premier clic ne devait jamais tomber sur une mine.

Il devait être possible de choisir un mode de difficulté (débutant, intermédiaire ou avancé).

Ensuite, 2 modes de jeu devaient être implémentés :

- Un mode où c'est le joueur qui découvre des cases
- Un mode dans lequel c'est une IA découvre des cases

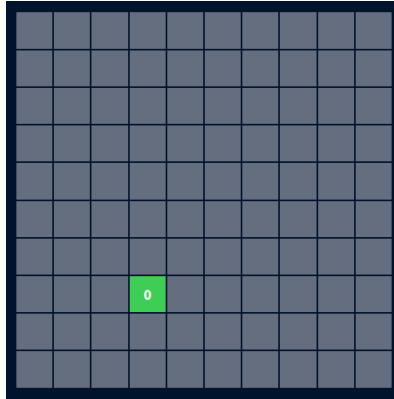
Il devait y avoir une option pour mettre des drapeaux.

Le programme devait également afficher des statistiques sur la partie en cours.

Algorithmie de base

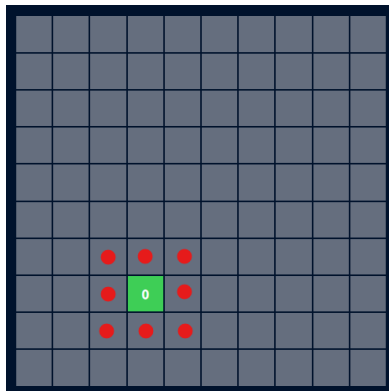
Dans cet algorithme, nous allons considérer une grille de largeur l et de hauteur h et comportant x mines.

Au tout début, on clique sur une case au hasard.

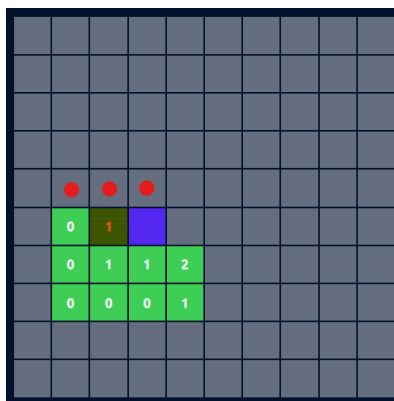


On poursuit en découvrant les cases les plus évidentes. C'est à dire les cases où on peut être sûr qu'il n'y a pas de mine.

Par exemple, tout autour d'une case indiquant le chiffre 0. C'est d'ailleurs le fonctionnement que nous utiliserons dans notre IA.

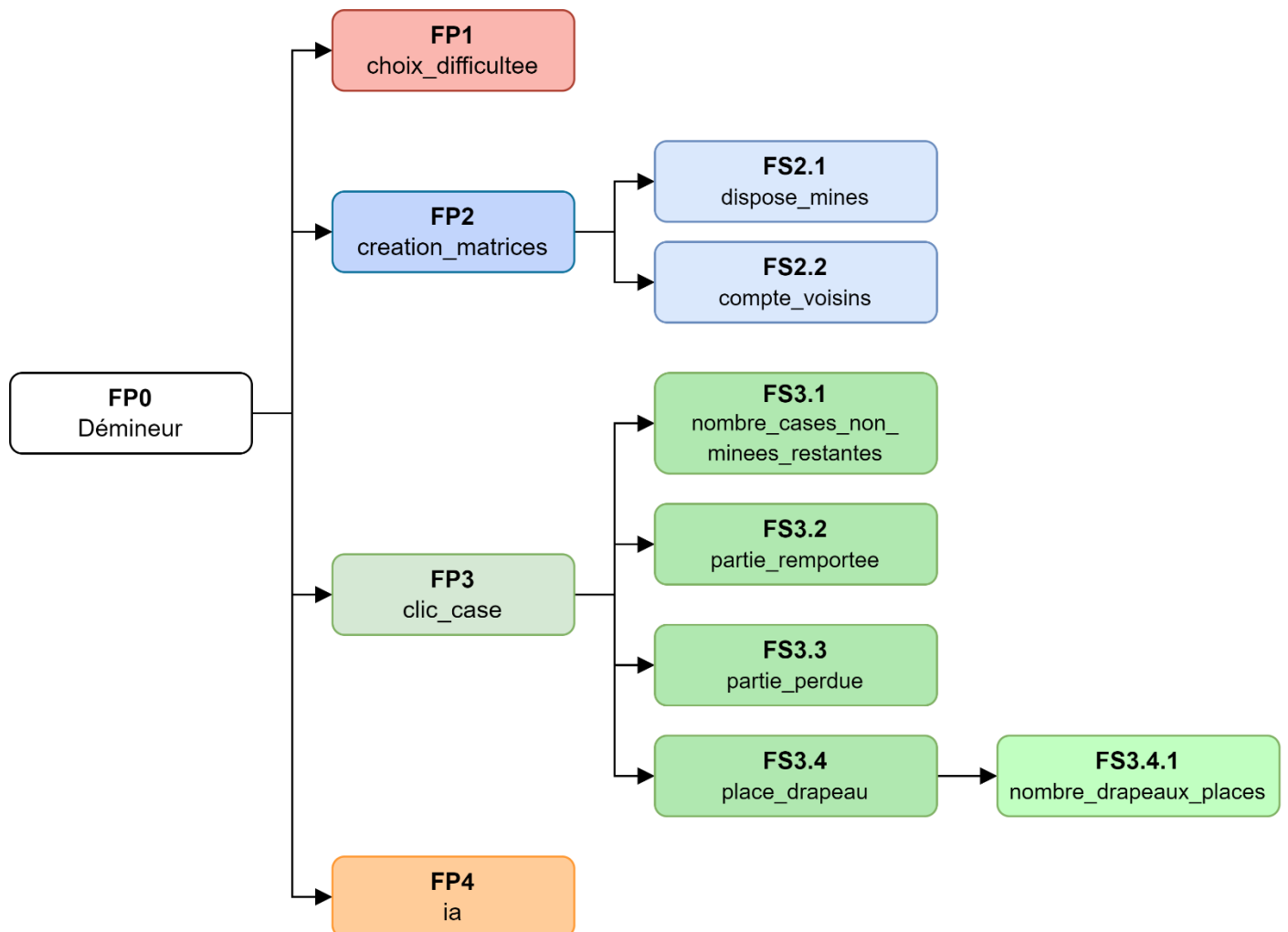


Un autre exemple, tout autour d'une case indiquant le chiffre 1 et dont on connaît l'emplacement de la mine. On marque par ailleurs l'emplacement de la mine par un drapeau.



Ce schéma se répète jusqu'à la fin de la partie. Lorsqu'il n'y a plus de cases à découvrir, la partie est gagnée.

Arbre Hiérarchique Fonctionnel



Analyse descendante

Niveau 0

FP1 – choix_difficulte	01/01/2023 TB
Description Cette fonction permet au joueur de choisir la difficulté de la partie	
E/S <div><div>(Paramètre) String - difficulté</div><div>choix_difficultee</div><div>(Global) Int – nb_mines (Global) Int – largeur (Global) Int – hauteur</div></div>	
Logigramme <div>FP1 choix_difficultee</div>	
Algorithme Si difficulté = « expert » : nb_mines ← 20 largeur ← 15 hauteur ← 15 Sinon si difficulté = « intermediaire » : nb_mines ← 10 largeur ← 10 hauteur ← 10 Sinon : nb_mines ← 5 largeur ← 10 hauteur ← 10 Fin Si	

FP2 – creation_matrices	08/01/2023 PR
Description Cette fonction crée les différentes matrices de jeu	
E/S <div style="display: flex; justify-content: space-between; align-items: center; padding: 10px;"> <div style="border: 1px solid black; background-color: #4a7ebb; color: white; padding: 10px; width: 35%;"> (Paramètre) Bool – premiere_generation (Global) Int – largeur (Global) Int – hauteur (Global) Int[][] - matrice_mines (Global) Int[][] - matrice_nombre_voisins (Global) Int[][] - matrice_cases_cliques (Global) Int[][] - matrice_drapeaux </div> <div style="border: 1px solid black; background-color: #4a7ebb; color: white; padding: 10px; width: 20%; text-align: center;"> creation_matrices </div> <div style="border: 1px solid black; background-color: #4a7ebb; color: white; padding: 10px; width: 35%;"> (Global) Int[][] - matrice_mines (Global) Int[][] - matrice_nombre_voisins (Global) Int[][] - matrice_cases_cliques (Global) Int[][] - matrice_drapeaux </div> </div>	
Logigramme <div style="text-align: center; margin-top: 20px;"> <pre> graph LR FP2[FP2 creation_matrices] --> FS2_1[FS2.1 dispose_mines] FP2 --> FS2_2[FS2.2 compte_voisins] </pre> </div>	
Algorithme <p> $\text{matrice_mines} \leftarrow$ matrice d'entiers à 2 dimensions de taille $x = \text{largeur}$ et $y = \text{hauteur}$ $\text{matrice_nombre_voisins} \leftarrow$ matrice d'entiers à 2 dimensions de taille $x = \text{largeur}$ et $y = \text{hauteur}$ $\text{matrice_cases_cliques} \leftarrow$ matrice d'entiers à 2 dimensions de taille $x = \text{largeur}$ et $y = \text{hauteur}$ $\text{matrice_drapeaux} \leftarrow$ matrice d'entiers à 2 dimensions de taille $x = \text{largeur}$ et $y = \text{hauteur}$ </p> <p> Pour x de 1 à largeur : Pour y de 1 à hauteur : $\text{matrice_mines}[x, y] \leftarrow 0$ $\text{matrice_nombre_voisins} \leftarrow 0$ Si $\text{premiere_generation} = \text{vrai}$: $\text{matrice_cases_cliques}[x, y] \leftarrow 0$ $\text{matrice_drapeaux}[x, y] \leftarrow 0$ Fin si Fin Pour Fin pour </p> <p> $\text{dispose_mines}()$ $\text{compte_voisins}()$ </p>	

FP3 – clic_case

09/01/2023
TB

Description

Cette fonction gère l'évènement de clic sur le plateau de jeu

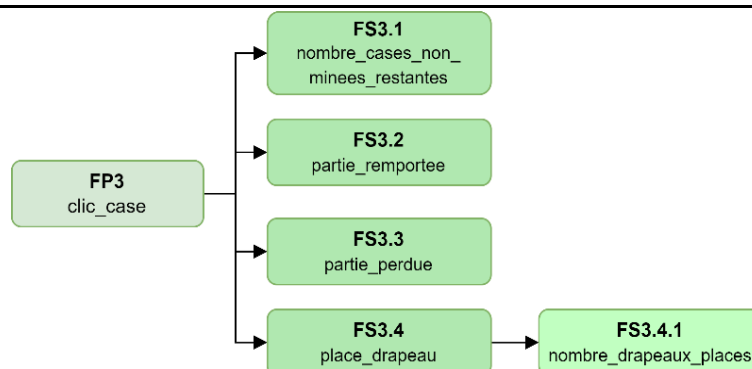
E/S

(Paramètre) Int – x
(Paramètre) Int – y
(Paramètre) String – type_clic
(Global) Int – nombre_cliques
(Global) Int[][] - matrice_mines
(Global) Int[][] - matrice_cases_cliques
(Global) Int[][] – matrice_drapeaux

clic_case

(Global) Int – nombre_cliques
(Global) Int[][] - matrice_cases_cliques

Logigramme



Algorithme

```

Si type_clic = « gauche »
  Si matrice_drapeaux[x, y] = 1 :
    Afficher « Vous ne pouvez pas cliquer sur une case avec un drapeau ! »
  Sinon :
    nombre_cliques ← nombre_cliques + 1

    Si nombre_cliques = 1 :
      Tant que matrice_mines[x, y] = 1 :
        creation_matrices()
      Fin tant que
    Fin Si

    Si matrice_cases_cliques[x, y] = 0 :

      matrice_cases_cliques[x, y] ← 1


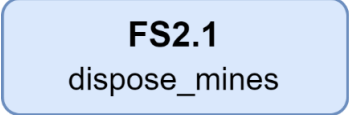
      Si matrice_mines[x, y] = 1 :
        partie_perdue()
      Sinon si nombre_cases_non_minee_restantes() = 0 :
        partie_reportee()
      Fin si

    Fin Si

  Fin si
Sinon :
  place_drapeau(x, y)
Fin si
  
```

FP4 – ia	12/01/2023 DD
Description Cette fonction découvre une case à proximité d'une case découverte dont le nombre de mines voisines est égal à 0	
E/S <div> <div> (Global) Int – largeur (Global) Int - hauteur (Global) Int[][] - matrice_cases_cliques (Global) Int[][] - matrice_nombre_voisins </div> <div>ia</div> </div>	
Logigramme <div>FP4 ia</div>	
Algorithme <pre> case_decouverte ← faux Pour x de 1 à largeur : Pour y de 1 à hauteur : Si matrice_cases_cliques[x, y] = 0 : Pour i de -1 à 1 : Pour j de -1 à 1 : Si case_decouverte = faux Et (i != 0 et j != 0) Et (x + i >= 1 et x + i <= largeur et y + j >= 0 et y + j <= hauteur) : Si matrice_cases_cliques[x + i, y + j] = 1 Et matrice_nombre_voisins[x + i, y + j] = 0 : case_decouverte ← vrai clic_case(x, y, "gauche") Fin si Fin si Fin si Fin pour Fin pour Fin si Fin pour </pre>	

Niveau 1

FS2.1 – dispose_mines	20/01/2023 PR
Description Cette fonction place aléatoirement les mines	
E/S 	
Logigramme 	
Algorithme Pour i de 1 à nb_mines : x ← aléatoire de 1 à largeur y ← aléatoire de 1 à hauteur matrice_mines[x, y] ← 1 Fin pour	

FS2.2 – compte_voisins	20/01/2023 PR
Description Cette fonction permet de compter le nombre de mines autour de chaque case et de le sauvegarder dans la matrice <code>matrice_nombre_voisins</code>	
E/S <div><div>(Global) Int – largeur (Global) Int – hauteur (Global) Int[][] - <code>matrice_nombre_voisins</code> (Global) Int[][] - <code>matrice_mines</code></div><div>creation_matrices</div><div>(Global) Int[][] - <code>matrice_nombre_voisins</code></div></div>	
Logigramme <div>FS2.2 compte_voisins</div>	
Algorithme <pre>Pour x de 1 à largeur : Pour y de 1 à hauteur : Pour i de -1 à 1 : Pour j de -1 à 1 : Si x + i >= 1 et x + i <= largeur et y + j >= 1 et y + j <= hauteur : matrice_nombre_voisins[x, y] ← matrice_mines[x + i, y + j] Fin si Fin Pour Fin Pour Fin Pour Fin pour</pre>	

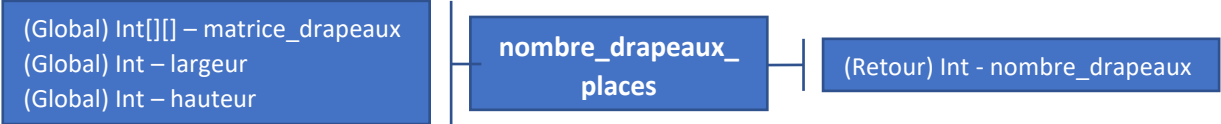
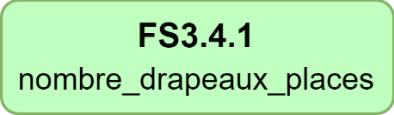
FS3.1 – nombre_cases_non_minees_restantes	20/01/2023 TB
Description Cette fonction permet de compter le nombre de cases non minées restantes	
E/S <div> <div> (Global) Int[][] – matrice_cases_cliques (Global) Int[][] – matrice_mines (Global) Int – largeur (Global) Int – hauteur </div> <div> nombre_cases_non_minees_restantes </div> <div> (Retour) Int - nombre_cases_non_minees </div> </div>	
Logigramme <div> FS3.1 nombre_cases_non_minees_restantes </div>	
Algorithme <pre> nombre_cases_non_minees ← 0 Pour x de 1 à largeur : Pour y de 1 à hauteur : Si matrice_cases_cliques[x,y] = 0 et matrice_mines[x,y] = 0 : nombre_cases_non_minees ← nombre_cases_non_minees + 1 Fin Si Fin Pour Fin Pour Retourner nombre_cases_non_minees </pre>	

FS3.2 – partie_remportee	20/01/2023 TB - DD
Description Cette fonction est appelée lorsque le joueur gagne la partie	
E/S N/A	
Logigramme <div>FS3.2 partie_remportee</div> N/A	
Algorithme Afficher la page de victoire	

FS3.3 – partie_perdue	20/01/2023 TB - DD
Description Cette fonction est appelée lorsque le joueur perd la partie	
E/S N/A	
Logigramme <div>FS3.3 partie_perdue</div>	
Algorithme Afficher la page de défaite	

FS3.4 – place_drapeau	20/01/2023 TB - DD
Description Cette fonction permet de placer un drapeau	
E/S <div style="display: flex; align-items: center; justify-content: space-between; margin-top: 10px;"> <div style="background-color: #4a7ebb; color: white; padding: 10px; border-radius: 5px; width: 30%;"> (Paramètre) Int – x (Paramètre) Int – y (Global) Int[][] – matrice_drapeaux (Global) Int - nombre_mines (Global) Int[][] - matrice_cases_cliquees </div> <div style="border-left: 1px solid black; border-right: 1px solid black; width: 10%;"></div> <div style="display: flex; align-items: center; justify-content: center; width: 40%;"> <div style="background-color: #4a7ebb; color: white; padding: 10px; border-radius: 5px; text-align: center; width: 40%;"> place_drapeau </div> <div style="width: 10%; border-left: 1px solid black; border-right: 1px solid black; height: 20px;"></div> <div style="background-color: #4a7ebb; color: white; padding: 10px; border-radius: 5px; text-align: center; width: 40%;"> (Global) Int[][] – matrice_drapeaux </div> </div> </div>	
Logigramme <div style="text-align: center; margin-top: 20px;"> <div style="background-color: #c6efce; padding: 10px; border-radius: 10px; display: inline-block; text-align: center;"> FS3.4 place_drapeau </div> <div style="font-size: 24px; margin: 0 10px;">→</div> <div style="background-color: #c6efce; padding: 10px; border-radius: 10px; display: inline-block; text-align: center;"> FS3.4.1 nombre_drapeaux_places </div> </div>	
Algorithme <pre> Si matrice_drapeaux[x, y] = 1 : matrice_drapeaux[x, y] ← 0 Sinon si nombre_drapeaux_places() < nombre_mines : Si matrice_cases_cliquees[x, y] = 0 : matrice_drapeaux[x, y] ← 1 Sinon : Afficher « Vous ne pouvez pas placer de drapeau sur une case déjà cliquée ! » Fin si Sinon : Afficher « Nombre de drapeaux maximum atteint ! » Fin si </pre>	

Niveau 2

FS3.4.1 – nombre_drapeaux_places	20/01/2023 TB - DD
Description Cette fonction permet de compter le nombre de drapeaux placés	
E/S 	
Logigramme 	
Algorithme nombre_drapeaux \leftarrow 0 Pour i de 1 à largeur : Pour j de 1 à hauteur : Si matrice_drapeaux[i, j] = 1 : nombre_drapeaux \leftarrow nombre_drapeaux + 1 Fin Si Fin Pour Fin Pour Retourner nombre_drapeaux	

Conclusion

Nous avons choisi d'utiliser le langage JavaScript pour l'algorithmie de ce projet pour deux raisons.

La première est évidemment la facilité avec laquelle ce langage peut être exécuté chez les clients. En effet, il suffit à l'utilisateur d'avoir un navigateur pour pouvoir l'exécuter.

La seconde est son type. En effet, le langage est un langage orienté évènement et il se prête parfaitement à ce projet puisque l'algorithmie doit majoritairement s'exécuter au clic du client.

Comme je le sous-entendais donc, nous avons développé une interface graphique sous forme d'un site web, disponible à cette URL : <https://theo-bnts.github.io/minesweeper>.

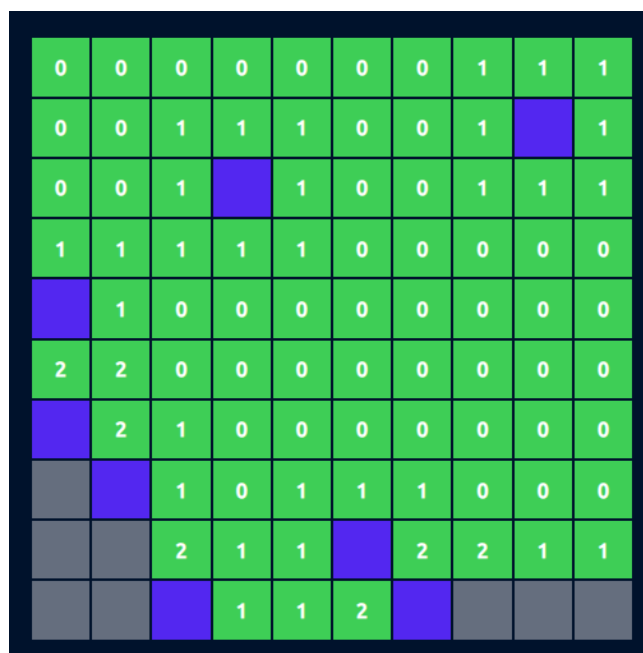
L'utilisateur peut donc interagir avec la grille avec ces 3 actions :

- Clic gauche : révèle la case.
- Clic droit : pose un drapeau.
- Bouton IA : découvre une case (voir introduction)

Nous avons bien pris en charge l'initialisation aléatoire des mines et fais en sorte que cette génération se fasse au premier clic. Ainsi, le premier clic ne peut pas tomber sur une mine. Le développement de différentes difficultés a été pris en compte, des statistiques affichent en bas de page et une IA a été développée.

Le cahier des charges est donc bien complètement respecté et le jeu est complètement fonctionnel.

Pour finir, voici une capture d'écran prise du jeu en fonctionnement.



0	0	0	0	0	0	0	1	1	1
0	0	1	1	1	0	0	1		1
0	0	1		1	0	0	1	1	1
1	1	1	1	1	0	0	0	0	0
	1	0	0	0	0	0	0	0	0
2	2	0	0	0	0	0	0	0	0
	2	1	0	0	0	0	0	0	0
		1	0	1	1	1	0	0	0
		2	1	1		2	2	1	1
			1	1	2				

Code

```
let nb_mines;
let largeur;
let hauteur;
let nombre_cliques;
let matrice_mines;
let matrice_nombre_voisins;
let matrice_cases_cliques;
let matrice_drapeaux;

/// FP1 ///

function choix_difficulte()
{
    let difficulte = new URLSearchParams(window.location.search).get("difficulte");

    if (difficulte === "expert")
    {
        nb_mines = 20;
        largeur = 15;
        hauteur = 15;
    }
    else if (difficulte === "intermediaire")
    {
        nb_mines = 10;
        largeur = 10;
        hauteur = 10;
    }
    else
    {
        nb_mines = 5;
        largeur = 10;
        hauteur = 10;
    }
}

/// FP2 ///

function creation_matrices(premiere_generation)
{
    matrice_mines = [];
    matrice_nombre_voisins = [];
    matrice_cases_cliques = [];
    if (premiere_generation)
    {
        matrice_drapeaux = [];
    }

    for (let x = 0; x < largeur; x++)
```



```
{
    matrice_mines[x] = [];
    matrice_nombre_voisins[x] = [];
    matrice_cases_cliques[x] = [];
    if (premiere_generation)
    {
        matrice_drapeaux[x] = [];
    }

    for (let y = 0; y < hauteur; y++)
    {
        matrice_mines[x][y] = 0;
        matrice_nombre_voisins[x][y] = 0;
        matrice_cases_cliques[x][y] = 0;
        if (premiere_generation)
        {
            matrice_drapeaux[x][y] = 0;
        }
    }
}

dispose_mines();

compte_voisins();
}

function dispose_mines()
{
    for (let i = 0; i < nb_mines; i++)
    {
        let x = Math.floor(Math.random() * largeur);
        let y = Math.floor(Math.random() * hauteur);

        matrice_mines[x][y] = 1;
    }
}

function compte_voisins()
{
    for (let x = 0; x < largeur; x++)
    {
        for (let y = 0; y < hauteur; y++)
        {
            for (let i = -1; i <= 1; i++)
            {
                for (let j = -1; j <= 1; j++)
                {
                    if (x + i >= 0 && x + i < largeur && y + j >= 0 && y + j < hauteur)
                    {
                        if (
                            x + i >= 0
```

```

        && x + i < largeur
        && y + j >= 0
        && y + j < hauteur
    )
}
matrice_nombre_voisins[x][y] += matrice_mines[x + i][y + j];
}
}
}
}
}
}
}

/// FP3 ///

function clic_case(x, y, type_clique)
{
    if (type_clique === "gauche")
    {
        if (matrice_drapeaux[x][y] === 1)
        {
            alert("Vous ne pouvez pas cliquer sur une case avec un drapeau !");
        }
        else
        {
            nombre_clicques++;

            if (nombre_clicques === 1)
            {
                do
                {
                    creation_matrices(false);
                }
                while (matrice_mines[x][y] === 1);
            }

            if (matrice_cases_clicques[x][y] === 0)
            {
                matrice_cases_clicques[x][y] = 1;

                if (matrice_mines[x][y] === 1)
                {
                    partie_perdue();
                }
                else if (nombre_cases_non_minees_restantes() === 0)
                {
                    partie_remportee();
                }
            }
        }
    }
}
```

```
}
else
{
    place_drapeau(x, y);
}

affiche_matrices();
}

function nombre_cases_non_minees_restantes()
{
    let nombre_cases_non_minees = 0;

    for (let x = 0; x < largeur; x++)
    {
        for (let y = 0; y < hauteur; y++)
        {
            if (matrice_cases_cliques[x][y] === 0 && matrice_mines[x][y] === 0)
            {
                nombre_cases_non_minees++;
            }
        }
    }

    return nombre_cases_non_minees;
}

function partie_remontee()
{
    window.location.href = "victoire.html";
}

function partie_perdue()
{
    window.location.href = "defaite.html";
}

function place_drapeau(x, y)
{
    if (matrice_drapeaux[x][y] === 1)
    {
        matrice_drapeaux[x][y] = 0;
    }
    else if (nombre_drapeaux_places() < nb_mines)
    {
        if (matrice_cases_cliques[x][y] === 0)
        {
            matrice_drapeaux[x][y] = 1;
        }
    }
    else

```

```
    {
        alert("Vous ne pouvez pas placer de drapeau sur une case déjà cliquée !");
    }
}
else
{
    alert("Nombre de drapeaux maximum atteint !");
}
}

function nombre_drapeaux_places()
{
    let nombre_drapeaux = 0;

    for (let x = 0; x < largeur; x++)
    {
        for (let y = 0; y < hauteur; y++)
        {
            if (matrice_drapeaux[x][y] === 1)
            {
                nombre_drapeaux++;
            }
        }
    }

    return nombre_drapeaux;
}

/// FP4 ///

function ia()
{
    let case_decouverte = false;

    for (let x = 0; x < largeur; x++)
    {
        for (let y = 0; y < hauteur; y++)
        {
            if (matrice_cases_cliques[x][y] === 0)
            {
                for (let i = -1; i <= 1; i++)
                {
                    for (let j = -1; j <= 1; j++)
                    {
                        if (
                            case_decouverte === false
                            && (i !== 0 || j !== 0)
                            && x + i >= 0 && x + i < largeur && y + j >= 0 && y + j <
hauteur
                        )
                        {
```

```
        if (matrice_cases_cliques[x + i][y + j] === 1 &&
matrice_nombre_voisins[x + i][y + j] === 0)
        {
            case_decouverte = true;

            clic_case(x, y, "gauche");
        }
    }
}
}
}
}

if (case_decouverte === false)
{
    alert("Aucune case à proximité d'une case découverte dont le nombre de mines
voisines est égal à 0 !");
}
}

/// AFFICHAGE ///

function affiche_matrices()
{
    let grille = document.getElementById("grille");

    while (grille.firstChild)
    {
        grille.removeChild(grille.firstChild);
    }

    for (let y = 0; y < largeur; y++)
    {
        let ligne = document.createElement("tr");

        for (let x = 0; x < hauteur; x++)
        {
            let case_ = document.createElement("td");

            case_.setAttribute("id", "case_" + x + "_" + y);
            case_.setAttribute("onclick", "clic_case(" + x + ", " + y + ", 'gauche')");
            case_.setAttribute("oncontextmenu", "clic_case(" + x + ", " + y + ",
'droite')");

            if (matrice_cases_cliques[x][y] === 1)
            {
                case_.setAttribute("class", "case case_cliquee");

                case_.innerHTML = matrice_nombre_voisins[x][y];
            }
        }
    }
}
```

```
        else if (matrice_drapeaux[x][y] === 1)
        {
            case_.setAttribute("class", "case case_drapeau");
        }
        else
        {
            case_.setAttribute("class", "case");
        }

        ligne.appendChild(case_);
    }

    grille.appendChild(ligne);
}

let nombre_mines = document.getElementById("nombre_mines");
nombre_mines.innerHTML = nb_mines;

let nombre_drapeaux = document.getElementById("nombre_drapeaux");
nombre_drapeaux.innerHTML = nombre_drapeaux_places();

let nombre_cases_non_minees_restantes_ =
document.getElementById("nombre_cases_non_minees_restantes");
nombre_cases_non_minees_restantes_.innerHTML = nombre_cases_non_minees_restantes();
}

/// MAIN ///

function main()
{
    nombre_cliques = 0;

    choix_difficulte();

    creation_matrices(true);

    affiche_matrices();
}
```