M2206 - Intégration Web - S2

Conway - Le jeu de la vie

1 Contexte et description du sujet

L'objectif de ce mini-projet est de s'approprier un programme déjà 100% fonctionnel et de lui apporter des améliorations. Il s'agit ici du jeu de la vie de Conway[1].

Le jeu de la vie est un automate cellulaire avec des règles très simples. Le jeu est très bien décrit sur la page wikipédia et nous allons simplement rappeler ici les règles du jeu (extraites de la page wikipédia).

- Le jeu se déroule sur une grille à deux dimensions dont les cases (ou cellule) peuvent prendre deux états : vivant ou mort
- Une cellule possède huit voisins, qui sont les cellules adjacentes horizontalement, verticalement et diagonalement.
- À chaque étape, l'évolution d'une cellule est entièrement déterminée par l'état de ses huit voisines de la façon suivante :
  - une cellule morte possédant exactement trois voisines vivantes devient vivante (elle naît) ;
  - une cellule vivante possédant deux ou trois voisines vivantes le reste, sinon elle meurt.

Un premier code fonctionnel est mis à disposition sur Moodle. Ce code permet de :

- créer une page web contenant une zone de dessin représentant la grille de jeu : un canvas de taille 500x500
- d'initialiser au hasard l'état des cellules sur la grille
- de simuler le jeu de la vie sur cette grille

Vous allez devoir vous approprier le code existant pour le faire évoluer afin de pouvoir contrôler :

- la taille de la grille : largeur et hauteur
- la taille des cellule
- la vitesse du jeu
- le pourcentage de cellules vivantes au début du jeu
- les motifs de départ : aléatoire ou défini
- les modalités d'affichage
- etc.

[1] voir `https://fr.wikipedia.org/wiki/Jeu_de_la_vie`

2 Améliorations à apporter

Nous allons commencer par créer une interface permettant de contrôler quelques éléments du jeu.

Exercice I Création de l'interface

Question 1. *Créer, dans la page `html`, une division pour manipuler le menu*

Question 2. *Ajouter dans cette division un sélecteur de type `range` pour contrôler la largeur de la grille*

Question 3. *Puis ajouter un second sélecteur pour contrôler la hauteur de la grille*

Question 4. *Côté `JavaScript`, modifier le code existant pour récupérer ces informations : largeur et hauteur.*

Question 5. *Jusqu'à présent, nous n'avions manipulé le canvas que pour y récupérer sa largeur, sa hauteur et un contexte de dessin en deux dimensions. Nous allons maintenant le manipuler pour modifier sa largeur et sa hauteur.*

Vous pouvez simplement indiquer la largeur et la hauteur du canvas avec cette instruction :

```
canvas.width = width;
```

Avec `width` qui représente la largeur contrôlée par l'utilisateur. Mettez en œuvre le code nécessaire pour contrôler largeur et hauteur, et vérifier son effet sur le jeu. Si besoin, vous pouvez temporairement désactiver le lancement du jeu en commentant la dernière ligne du fichier `JavaScript`.

L'intervalle de valeurs raisonnable pour la largeur et la hauteur de la grille de jeu est $[100, 700]$.

Question 6. *Pour pouvoir prendre en considération les informations communiquées par l'utilisateur via l'interface, il faut pouvoir associer une action lorsqu'un contrôleur est modifié : `onchange=""`.*

Il faut donc que créer une fonction `initGame()` qui initialise le jeu et le démarre. C'est cette fonction qui sera associée aux contrôleurs.

Attention, cela impliquera certainement de transformer quelques variables déclarées avec `const` en `let`.

Question 7. *Ajouter maintenant un contrôleur pour la taille des cellules.*

L'intervalle de valeurs raisonnable pour la taille des cellules est $[5, 20]$.

Exercice II Refonte du code

Avant de continuer à apporter des améliorations au programme, nous allons encapsuler toutes les informations du jeu dans une structure de données plus simple à manipuler.

Pour l'instant, nous avons de nombreuses variables à manipuler : largeur et hauteur de la grille, taille des cellules, les cellules, etc. Nous allons rassembler tout cela dans une seule structure.

Question 1. *Créer la structure de données permettant de manipuler toutes les informations du jeu :*

```
const params = {};
```

Question 2. *Supprimer le code définissant la couleur des cellules vivantes et mortes (`ALIVE` et `DEAD`) dans le programme principal.*

Question 3. *Puis, ajouter ces deux lignes de code dans la fonction `initGame()` précédemment créée :*

```
params.ALIVE = "#ff0000";
params.DEAD = "#ffffff";
```

Question 4. *Le code n'est maintenant plus opérationnel. Vous allez devoir le corriger, fonction par fonction, pour le remettre en état de fonctionnement. Pour vous aidez, vous pouvez activer les outils d'aide au développeur de votre navigateur, notamment la* ***console****, et corriger une par une les erreurs indiquées.*

Vous pouvez aussi procéder de manière plus globale en remplaçant toutes les occurrences de `ALIVE` par `params.ALIVE`. Idem pour `DEAD`.

Attention cependant aux potentiels effets de bords du remplacement globale d'une chaîne de caractères par une autre.

Question 5. *Poursuivre ces modifications en rassemblant toutes les informations du jeu (largeur et hauteur de la grille, taille des cellules, les cellules, etc.) dans la variable `params`.*

En commençant par la largeur et la hauteur, vous devrez notamment mettre à jour la fonction `showGrid`. Dans cette fonction, le paramètre `cellSize` doit être remplacé par `params` et la variable `cellSize` doit être remplacée par `params.cellSize`. Il faudra effectuer la même opération pour `height` et `width`.

Question 6. *Réaliser la même mise à jour pour toutes les fonctions. Si les fonctions avaient en paramètre des informations du jeu, alors elles vont avoir `params` à la place. Il est bien évidemment possible d'ajouter d'autres paramètres aux fonctions si nécessaire.*

Exercice III Nouvelles fonctionnalités

Nous allons continuer à améliorer le code en ajoutant quelques nouvelles fonctionnalités.

Question 1. *Nous allons commencer par ajouter un contrôleur pour la vitesse du jeu, c-à-d. pour contrôler la fréquence d'exécution de l'animation.*

Pour cela, nous allons utiliser la fonction JavaScript `setTimeout()`[2] *qui permet d'exécuter du code après un délai exprimé en millisecondes.*

Ainsi, il suffit de créer une nouvelle entrée dans la variable `params` *:*

```
params.timeout = 100;
```

Puis de remplacer, dans la fonction `updateCells`*, l'appel* `window.requestAnimationFrame(updateCells);` *par :*

```
setTimeout(() => {
window.requestAnimationFrame(updateCells);
}, params.timeOut);
```

Question 2. *Ajouter maintenant un contrôleur pour le* `timeOut`*.*

L'intervalle de valeurs raisonnable pour le `timeOut` *est* $[10, 1000]$.

Question 3. *En procédant de la même manière que pour le contrôle de la largeur, hauteur et taille des cellules, la gestion du timeOut impose de redémarrer le jeu intégralement. Ce n'est pas forcément très intuitif.*

Il serait préférable de pouvoir modifier la vitesse d'exécution du jeu pour un jeu en cours, sans devoir le réinitialiser.

Pour cela, il suffit de créer une nouvelle fonction permettant de mettre à jour le timeOut lorsque l'utilisateur le modifie, sans réinitialiser le jeu.

Mettez en place cette nouvelle fonction côté `JavaScript` *et modifier l'appel associé au contrôleur côté* `html`*.*

Question 4. *Nous allons maintenant ajouter la possibilité de contrôler la couleur des cellules vivantes (rouge par défaut). Pour cela, vous pourrez utiliser un élément HTML* `input` *de type* `color`[3]*.*

À partir de la document disponible en ligne, mettez en œuvre cette nouvelle fonctionnalité.

Question 5. *Nous allons maintenant ajouter côté* `html` *(et donc* `JavaScript`*) un compteur indiquant le nombre de générations exécutées par le jeu.*

1. *Pour ce faire, vous pouvez ajouter un élément* `html` *vide mais identifié par un* `id`*.*
2. *Le récupérer côté* `JavaScript`
3. *Puis le mettre à jour avec la commande* `innerHTML` *appliquée à l'élément. Il faudra bien évidemment avoir ajouté un compteur de générations dans* `params`

Question 6. *Comme nous l'avons vu dans le projet précédant sur les fourmis, nous allons ajouter un bouton permettant de mettre le jeu en pause.*

1. *il faut commencer par stocker dans une variable les informations liées à l'animation*
2. *puis créer une fonction permettant de démarrer l'animation :* `startAnimation()`
3. *et une fonction permettant d'arrêter l'animation :* `stopAnimation()`
4. *enfin, côté* `html`*, ajouter un* `input` *de type* `radio` *permettant de contrôler l'animation*

[2] voir `https://www.w3schools.com/jsref/met_win_settimeout.asp`
[3] voir `https://developer.mozilla.org/fr/docs/Web/HTML/Element/Input/color`

5. *vous devrez ajouter à* `params` *une variable de type booléen permettant de savoir si l'animation est en cours ou non.*

Exercice IV Zone découverte et motifs particuliers

Pour terminer ce mini-projet, vous allez ajouter deux fonctionnalités nouvelles :

- L'affichage des zones visitées au moins une fois par une cellule vivante. Les cellules concernées devront apparaître d'une autre couleur (sélectionnable par l'utilisateur) et pourront toujours être parcourues par l'automate. Cela implique de revoir une partie du code existant et de manipuler une forme de "calque" sur la grille utilisée pour déterminer l'état des cellules.
- La prise en compte de motifs particuliers pour l'initialisation afin de simuler des comportements connus (voir la page wikipédia pour quelques motifs connus tels que "le planeur", "la grenouille", etc.)
- Enfin, n'hésitez pas à mettre quelques règles de `css` pour personnaliser l'interface et la rendre plus agréable à utiliser.