IFT1015 ~ TP1 ~ Conway's Game of Life

Introduction *z

Présentation du TP

Le TP2 a pour but de vous faire pratiquer les concepts suivants : les boucles, les tableaux, les fonctions, la décomposition fonctionnelle, le traitement d'événements, et la programmation web. Le code que vous devez écrire (fichier "tp2.js") implante un jeu qui exécute dans l'environnement du fureteur. Bien qu'il soit possible et même judiscieux d'utiliser codeBoot ou bien node.js pour développer certaines parties du jeu ultimement il faudra faire des tests et du déboguage avec les outils de développement du fureteur. Vous pouvez utiliser le code qui a été montré dans le cours mais vous ne devez pas utiliser du code provenant d'ailleurs (du web).

Présentation du jeu -> page wikipédia

Déroulement du jeu

Le *Game of Life* (Jeu de la Vie) est un jeu dit à "0 joueurs", au sens où le déroulement du jeu dépend seulement de sa configuration initiale.

Les cases de la grille du jeu représentent des cellules mortes ou vivantes. Une cellule est dite vivante lorsqu'elle est colorée. À chaque "tour", chaque cellule change d'état en fonction de ses voisins.

- Une cellule qui a moins de deux voisins vivants à une étape sera morte à la prochaine étape (*sous-population*)
- Une cellule qui a deux ou trois voisins vivants continura de vivre à la prochaine étape
- Une cellule qui a plus de trois voisins vivants sera morte à la prochaine étape (*sur-population*)
- Une cellule morte qui a exactement trois voisins vivants "naîtra" à l'étape suivante et sera donc vivante (*reproduction*)

La partie se déroule donc ainsi :

1. Décider (ou laisser au hasard) quelles cellules commencent vivantes et quelles commencent mortes
2. Démarrer le jeu et laisser la grille changer d'état toute seule
3. Apprécier les jolis motifs de l'automate qui évoluent sur l'écran.

Interface

La partie HTML qui décrit la grille et les boutons de l'interface est gérée pour vous, il vous reste à implanter le code qui sera appelé lorsqu'on clique sur chaque bouton.

- *Play* : Anime la grille. Ce bouton fait l'équivalent d'appuyer rapidement sur le bouton *Step*
- *Step* : Avance d'un tour dans le jeu. Chaque cellule de la grille va alors passer à son prochain état selon les règles mentionnées plus haut
- *Reset : Remet la grille à zéro (toutes les cellules tombent à l'état* mort*)
- *Random* : remplit la grille avec une densité aléatoire d'un certain pourcentage. On peut choisir le pourcentage approximatif de cellules vivantes en entrant un chiffre entre 1 et 99 dans la zone de texte à côté.
- Les deux zones de texte sous la ligne de boutons permettent de changer les dimensions de la grille **sans perdre l'état des cellules qui ne sont pas supprimées**. Voir l'animation ici

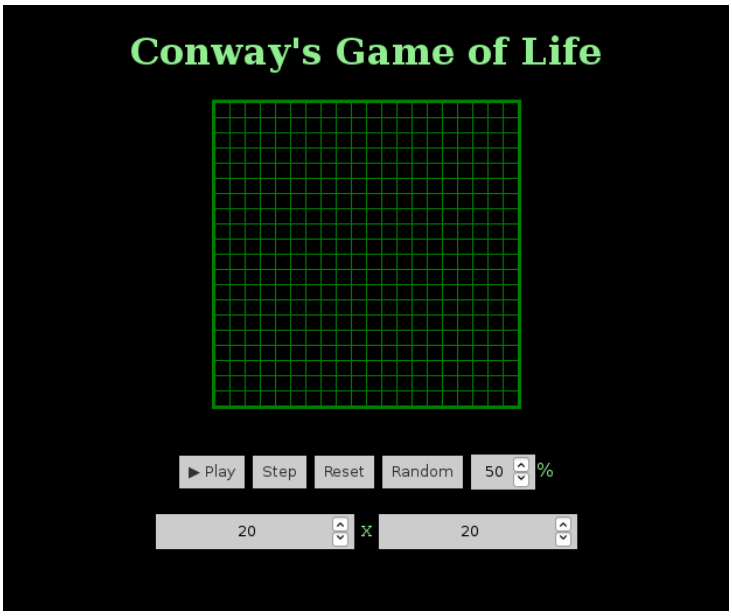
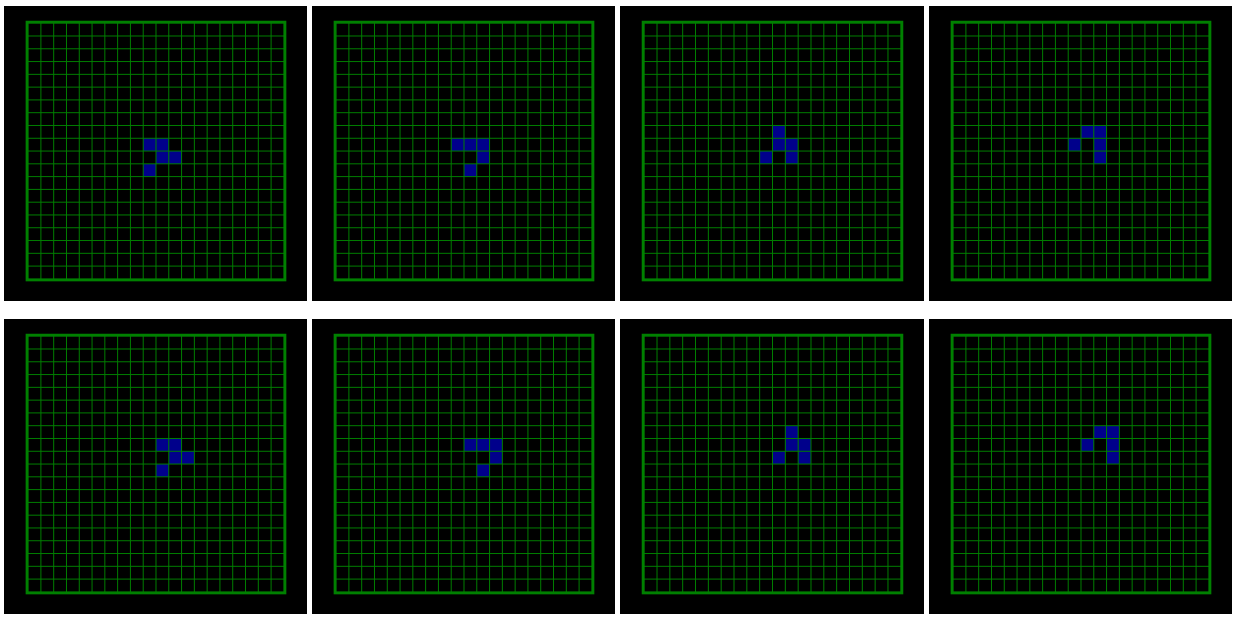


Figure 1: Grille de jeu

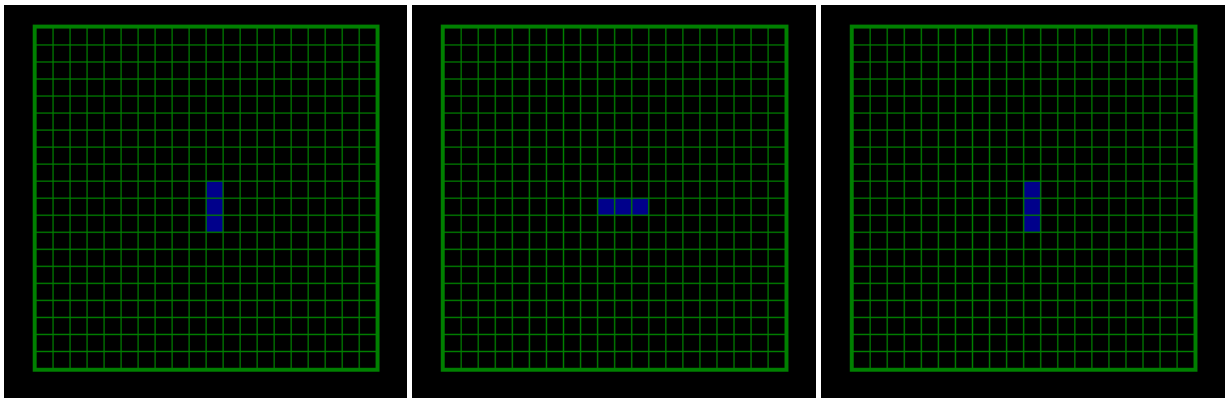
Exemples

Voici quelques exemples de figures que vous devriez retrouver si votre code fonctionne bien :

Le *glider* (planeur)



L'oscillateur simple



Le *still life* (la structure stable)

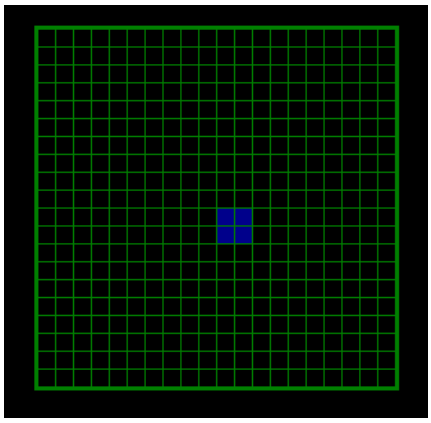


Figure 2: Structure qui ne varie pas dans le temps

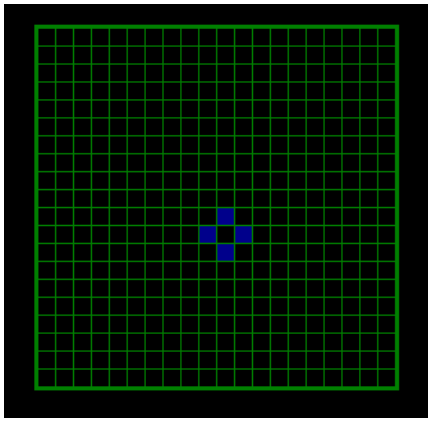


Figure 3: Autre structure qui ne varie pas dans le temps

Détails techniques

De codeBoot à pas codeBoot

Puisque le travail pratique demandera d'écrire du code qui sera lié à une page HTML donnée, vous ne pourrez plus écrire et exécuter votre code dans codeBoot.

Vous devrez écrire votre code dans un fichier (il vous faudra donc un éditeur de texte - voir les choix plus bas) et exécuter votre code dans un navigateur.

Pour lier votre code à la page HTML :

1. téléchargez le fichier `game-of-life.zip` sur StudiUM et décompressez-le quelque part.
2. Ouvrez le fichier `index.html` dans Firefox ou Chromium
3. Vous pourrez ensuite modifier le fichier `tp1.js` qui contient le squelette de l'application. Les fonctions à coder seront appelées automatiquement lorsqu'on clique sur les différents boutons.

Notez : vous devrez remettre *uniquement le fichier* `tp1.js`. Vous ne pouvez pas modifier les autres fichiers pour programmer le jeu.

Choix de l'éditeur de texte :

Sur les ordinateurs du lab (et sur GNU/Linux en général) :

Ouvrez `Kate` ou `gedit` depuis le menu en bas à gauche. Vous devriez comprendre assez intuitivement comment modifier un fichier Javascript, leurs interfaces sont très intuitives.

Chez vous (assumant Windows ou Mac) :

- Windows : `Notepad++`
- Mac : `TextMate`

Utilisateurs avancés :

Si vous souhaitez aller plus loin, vous voudrez possiblement tester l'un des deux éditeurs de texte mythiques pour programmeurs :

- `Emacs`~~, sans l'ombre d'un doute le seul vrai éditeur de texte pour programmeurs,~~[1] est définitivement à essayer. Emacs est à la fois un éditeur de texte et un environnement programmable en `Elisp`, ce qui permet de modifier et d'étendre ses fonctionnalités à l'infini. C'est bien souvent le choix des programmeurs avancés qui désirent être aussi efficaces que possible.
- `vim` : l'éditeur de texte en console installé par défaut sur la majorité des ordinateurs et serveurs sous GNU/Linux, également très utilisé par les programmeurs avancés. Plusieurs plugins avancés sont disponibles pour cet et diteur et on peut même (avec assez de motivation) écrire ses propres plugins en *VimScript*.

Un certain apprentissage est requis pour s'habituer aux deux éditeurs de texte, mais l'effort investi au début est *très payant*. Vous pouvez consulter des tutoriels : Tour guidé d'Emacs, Tutoriel interactif pour vim.

Ces deux projets étant des projets libres, certaines variantes (plus jolies et avec de la configuration par défaut plus intuitive) existent :

- Spacemacs
- NeoVim

[1] Voir Editor war sur Wikipédia pour plus de détails

Implantation *z

Expliquer l'API

Liste de fonctions à coder

- Random : remplit la grille avec une densité aléatoire d'un certain pourcentage. Pour cela, la fonction `random` est appellée avec en paramètre le pourcentage (0 à 99) entré dans la zone de texte juste à côté.

Vous pouvez (et c'est même encouragé) vous créer vos propres fonctions pour vous aider.

Comportement de la grille

De façon générale, en informatique, on considère que le pixel situé en haut à gauche est le pixel (0, 0)

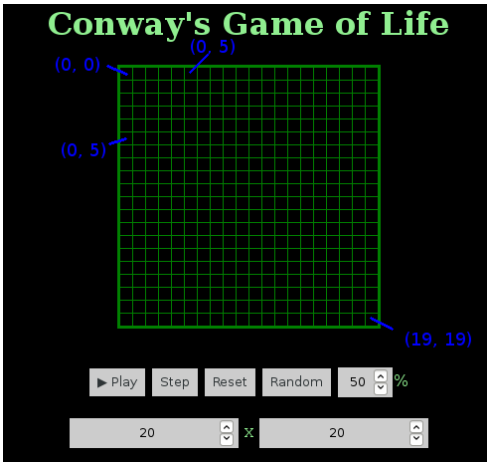
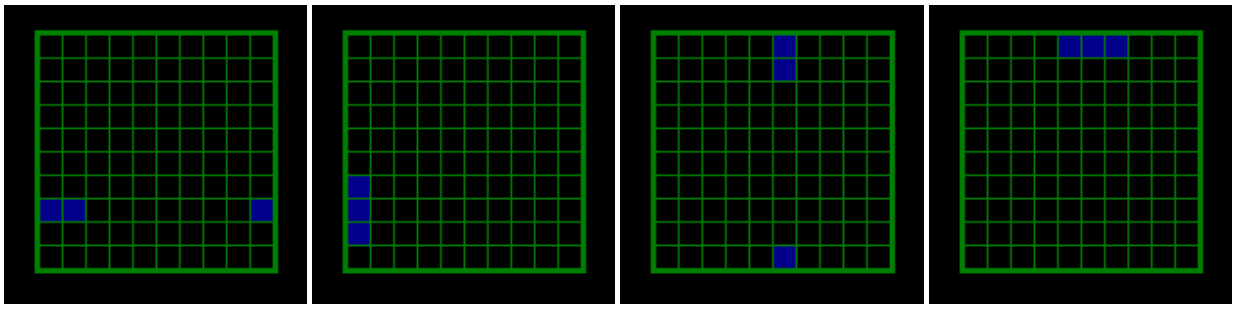


Figure 4: Coordonnées sur une grille 20x20

Un peu comme les *wrap zones* de Pac-Man, la grille est considérée comme périodique. Autrement dit, sur une grille de 10x10, le pixel à la position (0, 5) est "voisin" de (9, 5) et le pixel (5, 0) est voisin de (5, 9) :



Bonus : Version trois couleurs

Si vous terminez le travail plus tôt que les autres et que vous souhaitez posser le jeu plus loin, vous êtes invités à remettre un *deuxième fichier*, `bonus.js` pour des points bonus.

Vous devez adapter le plus gracieusement possible votre code original pour supporter *trois grilles de Game of Life* parallèles, une représentée par le rouge, une représentée par le vert et une représentée par le bleu.

La couleur affichée à l'écran sera une combinaison de ces trois couleurs. Vous aurez donc $2^3 = 8$ états possibles :

- Noir (mort)
- Rouge (vivant dans la grille 1 seulement)
- Vert (vivant dans la grille 2 seulement)
- Jaune (vivant dans les grilles 1 et 2, mort dans la grille 3)
- Bleu (vivant dans la grille 3)
- Fuchsia (vivant dans les grilles 1 et 3)
- Turquoise (vivant dans les grilles 2 et 3)
- Blanc (vivant dans toutes les grilles)

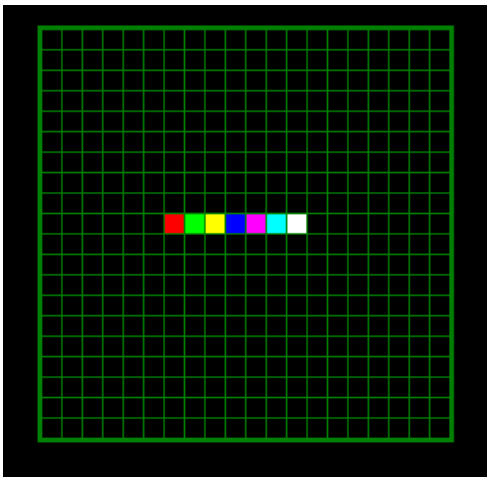


Figure 5: Couleurs possibles

De plus :

- lorsqu'on clique sur une cellule, sa couleur doit passer à la prochaine couleur dans la liste (dans cet ordre).
- La fonction de randomisation doit attribuer une couleur aléatoire parmi les huit à chaque cellule

Voir l'animation ici

Évaluation

- Ce travail compte pour 15% dans la note finale du cours. Vous **devez** faire le travail par groupes de 2 personnes. Indiquez vos noms clairement dans les commentaires au début de votre code. Un travail fait seul engendrera une lourde pénalité (qui ne peut **pas** être compensée par la question bonus). Les équipes de plus de deux seront refusées.
- Vous devez seulement remettre votre fichier `tp1.js`. Si vous avez fait la question bonus, remettez également le fichier `bonus.js`. La remise doit se sur le site Studium du cours avant la date indiquée sur la page prévue à cet effet.
- Voici les critères d'évaluation du travail :
  - l'exactitude (respect de la spécification)
  - l'élégance et la lisibilité du code
  - la présence de commentaires explicatifs lorsque nécessaire
  - le choix des identificateurs
  - la décomposition fonctionnelle et le choix de tests unitaires pertinents.
- Indications :

- La performance de votre code doit être raisonnable
- Chaque fonction devrait avoir un bref commentaire pour indiquer ce qu'elle fait
- Il devrait y avoir des lignes blanches pour que le code ne soit pas trop dense (utilisez votre bon sens pour arriver à un code facile à lire)
- Les identificateurs doivent être bien choisis pour être compréhensibles (évitez les noms à une lettre, à l'exception de `i`, `j`, ... pour les variables d'itérations des boucles `for`)
- Vous devez respecter le standard de code pour ce projet (soit, les noms de variables en camelCase).