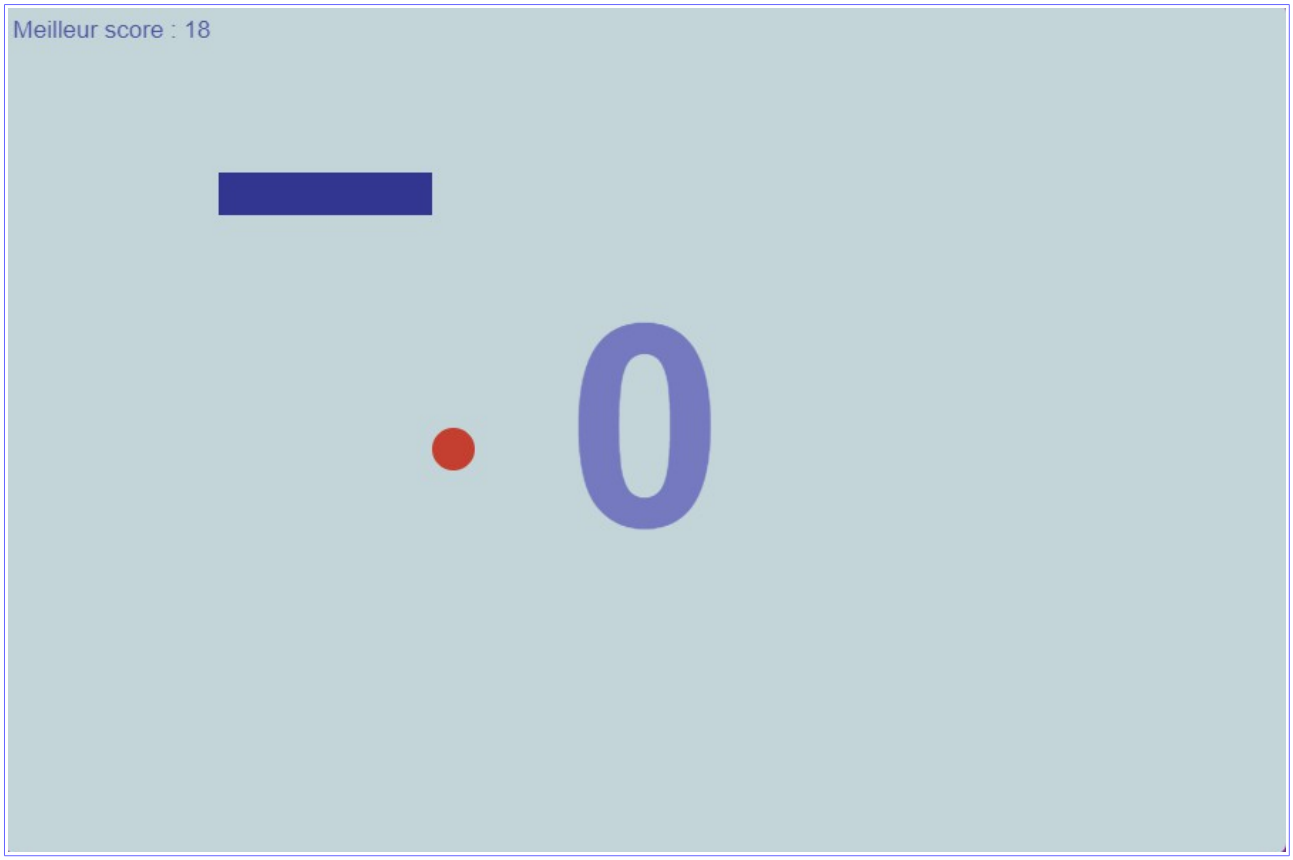
Projet ICN – Dossier réalisé par LABBE Alexandrine

Mini jeu - Snake

Réalisation :
LABBE Alexandrine
ETIENNE Lohan



Sommaire

Pourquoi ce projet ?

Le jeu Snake possède un concept simple l'ayant amené à être porté sur un très grand nombre de plates-formes et à être reproduit par grand nombre de personnes de différentes façons. Ainsi, il est enrichissant de comprendre l'une des façons de créer ce jeu classique.

Bien que les jeu sont de plus en plus développés, sophistiqués et compliqués, il était intéressant de revenir aux bases d'un jeu culte et simpliste afin de comprendre l'origine de ces nouveaux jeux bien plus approfondis et de faire un jeu à notre niveau.

Point culture :
Snake, signifiant « serpent », est un jeu vidéo sorti en 1976. Le joueur contrôle une ligne (le « serpent ») et doit manger des pommes afin de grandir. Si le serpent touche les bordures ou son propre corps, le jeu est perdu.
Le jeu a regagner en popularité à partir de 1998 lorsque Nokia l'as intégré dans ses produits. Il est alors devenu un classique du jeu sur appareil mobile.

Cela a été très instructif de découvrir les étapes de développement d'un mini-jeu, les défis rencontrés, le partages des tâches entre le jeu et sa page Web l'introduisant.

Quel est l'objectif ?

Notre objectif est de créer notre mini-jeu, de le mettre en ligne et par conséquent créer sa page de présentation de la même façon qu'un jeu normal. Il y a donc une page d'accueil présentant le projet, un cahier de bord pour son avancement, des explications ainsi qu'une lien permettant de jouer.
Ensuite, nous devons créer le jeu avec le peut de temps et de connaissances que nous avons.

Comment allons-nous réaliser notre projet ?

Point culture :
L'Hyper TextMarkup Language, ou HTML, est un langage de balisage conçu pour la création de pages web. Il permet d'inclure des textes, des liens, des photos et même des sons et des vidéos. Il a été développé par World Wide Web Consortium (W3C) et WHATWG dans sa version initiale en 1997.

Point culture :
Le Cascading Style Sheets, ou CSS, est une façon de changer l'apparence des pages HTML. Il a lui aussi été conçu par W3C et il est supporter par la plupart des navigateurs modernes. Un des avantages à l'utilisation du CSS est que la page web peut toujours être affichée et comprise même si le CSS ne fonctionne pas ou est retiré

Nous allons tout d'abord créer une page d'accueil présentant en quelques lignes le projet qui sera visible dès que quelqu'un visite le site. Elle sera codée en HTML et mise en forme en CSS.

Cette page conduira à quelques autres pages tel qu'un carnet de bord et un mini-tutoriel. Enfin, sur la page d'accueil, un bouton conduira à la page sur laquelle se trouve notre jeu Snake, en 2D, codé grâce à la balise canvas introduite avec HTML5 mais majoritairement grâce à du JavaScript.

Point culture :
Le JavaScript est un langage de programmation interprété. Il est souvent utiliser dans les pages web afin de créer du contenu dynamique tel que des messages popup, une horloge ou comme ici un jeu.
Le JavaScript à été développé par Brendan Eich en 1995 et est à différencier du Java qui est un tout autre langage de programmation.

Comment avons-nous réparti nos tâches ?

Lohan : coder la page web qui sers de présentation pour le projet.

Alexandrine : coder le Jeu du Snake en JavaScript

Ce que nous avons fait ensemble : Nous avons choisi vers quelle direction se tournerait le projet ainsi que l'aspect du site.

Ma partie

La première difficulté que j'ai rencontré dès le début de l'année était que je n'avais aucune connaissances en programmation.

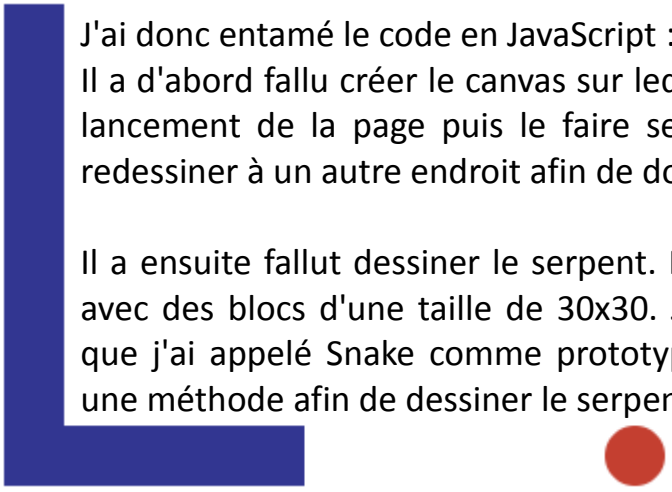
J'ai donc, à l'aide des exercices donnés par la professeur et d'internet, commencé à faire des recherches et des petits projets en utilisant du HTML puis en rajoutant petit à petit du CSS.
J'ai appris à faire des pages assez simple et certaines plus élaboré. Par la suite, dans l'optique de faire un jeu, j'ai fais des recherches sur les différents langages pour savoir lequel serait le plus adapté.

Le JavaScript me paraissait le plus adapté pour le projet. Le langage n'est pas trop compliqué. Bien qu'étant limité, il me permet quand même de faire des petits projets intéressants. On peut l'intégrer à une page web sans avoir besoin de logiciel annexe pour faire fonctionner le jeu et beaucoup de ressources sont disponibles sur internet afin de comprendre son fonctionnement.

Dans un second temps, à l'aide de cours et ressources trouvés sur internet, j'ai commencé la création de mon « Snake ».
La première étape était la création d'une page HTML auquel j'ai du relier mon script JavaScript.

J'ai donc entamé le code en JavaScript :
Il a d'abord fallu créer le canvas sur lequel le jeu se dessine, le faire s'afficher dès le lancement de la page puis le faire se rafraîchir en effaçant son contenu pour le redessiner à un autre endroit afin de donner l'illusion que les formes se déplaçaient.

Il a ensuite fallut dessiner le serpent. Pour cela, j'ai « découpé » le canvas en grille avec des blocs d'une taille de 30x30. J'ai ensuite utilisé une fonction constructeur que j'ai appelé Snake comme prototype pour le serpent dans laquelle j'ai rajouté une méthode afin de dessiner le serpent dans le canvas.



Le canvas étant découpé en blocs, je construis le serpent à l'aide de plusieurs blocs. Pour faire cela j'utilise des *Arrays*.

Une fois le serpent créé, il me manque la capacité de le diriger. Pour cela il y a deux principales possibilités :
- afficher des boutons sur l'écran
- utiliser les flèches du clavier.

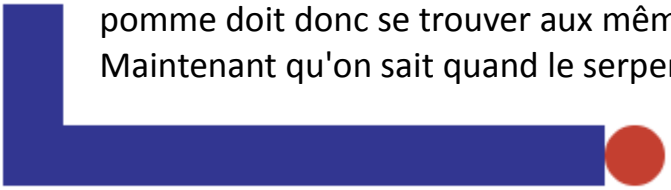
Trouvant les boutons trop compliquer pour jouer, je décide d'utiliser les flèches du clavier.
Indiquer uniquement la direction du serpent n'est pas suffisant car le serpent ne peut pas prendre n'importe quelle direction. Celui-ci ne peut pas aller dans la direction opposée à celle dans laquelle il va déjà. Par exemple, si le serpent va vers la droite, il ne peut pas aller directement vers la gauche.

Il faut maintenant créer la pomme que le serpent va manger. Bien que la technique pour la fabriquer soit principalement la même que pour le serpent, il y a une différence, la pomme est ronde mais cela ne change pas beaucoup de choses en dehors de la façon de dessiner (il faut la position du centre de la pomme puis faire un rond en utilisant les radius).

Avant de m'occuper des dernières choses concernant la pomme, je code les événement pour savoir quand le jeu est perdu, 2 choix possible :

- Le serpent passe sur son propre corps, je regarde si la position de sa tête (le premier bloc) correspond à la même position en x et y qu'une autre partie de son corps. Si le corps et la tête se retrouvent sur une même position, la valeur « true » est retournée et le jeu est perdu.
- Le serpent mange un mur, la tête étant encore une fois le bloc qui dirige, il suffit de vérifier que la tête est bien comprise entre les valeurs minimums et maximums en x et y du canvas, en d'autres termes qu'elle est toujours dedans. Si elle ne l'est pas, la valeur « true » est retournée et le jeu est perdu.

Le serpent sachant maintenant détecter les collisions avec les murs et son propre corps, il est temps qu'il sache détecter quand il mange une « pomme ». La tête de la pomme doit donc se trouver aux mêmes coordonnées x et y que la pomme.
Maintenant qu'on sait quand le serpent mange la pomme il y a deux choses à faire :



🍎 Il faut créer une nouvelles pommes en lui donnant une nouvelle position. Cette position doit être aléatoire, on utilise donc la fonction *random* et *round* (pour avoir un chiffre entier car le nombre aléatoire est compris entre 0 et 1) de l'objet *Math.* On fait cela pour les coordonnées x et y de la pomme afin de lui donner ses nouvelles coordonnées.

🍎 La deuxième chose à faire est de faire grandir le serpent à chaque fois que celui-ci mange une pomme. Pour cela il suffit de comprendre la façon dont celui-ci se déplace. À chaque fois qu'un certain délais passe, le dernier bloc du serpent est enlevé et un nouveau est mis à la place de la tête ce qui donne l'illusion de déplacement. Afin de faire grandir le serpent, il suffit simplement de détecter quand le serpent mange une pomme et de ne pas enlever le dernier bloc. Si le dernier n'est pas enlever mais qu'un nouveau est ajouté, le serpent va donc grandir.

L'une des dernières choses qu'il reste est d'annoncer quand le jeu est perdu ainsi que d'afficher le score.

Pour afficher quand le jeu est perdu, il me suffit de créer une fonction écrivant le message voulu ainsi que l'endroit où il doit se dessiner sur le canvas. Bien sûr, quand le jeu est perdu, il faut un moyen de le relancer. Je la touche *Espace* qui est relié à une fonction *restart* afin de recommencer le jeu et je réinitialise le jeu.
Pour afficher le score, je créée une variable que j'initialise au début du jeu et qui augmente à chaque fois que le serpent mange une pomme. Cependant on ne voit pas la variable, il faut donc l'afficher à l'aide encore une fois d'une fonction.

Après avoir testé plusieurs fois le jeu, certains petits bugs sont apparu. Si j'appuie plusieurs fois sur la touche *Espace* pour redémarrer, le serpent fini par aller de plus en plus vite. Il suffit pour régler cela de mettre une référence pour le délais de déplacement dans la fonction *restart* et de réinitialiser le délais à chaque fois.
Deuxième problème, la pomme pouvant apparaître à n'importe quel endroit, il n'est pas impossible qu'elle apparaisse sur le corps du serpent. Pour éviter cela, je vérifie avant de dessiner la pomme, si les nouvelles coordonnées se trouvent sur le corps du serpent en vérifiant bloc par bloc. Si elle n'y est pas, je la dessine, sinon de nouvelles coordonnées sont calculées.

Pour finir, il ne reste plus qu'un peu de mise en page sur le canvas, j'utilise donc du CSS intégré à mon JavaScript.

Après quelques réflexions, je décide d'ajouter 2 options au jeu :

- Je rajoute l'affichage du meilleur score, pour cela, je copie la fonction *drawScore* pour l'affichage et je met le meilleur score dans la variable *bestScore*. Je compare ensuite le score actuel avec le meilleur score, si le score actuel est supérieur, *bestScore* prend cette valeur.

- La dernière option ajoutée est l'accélération du serpent. Lorsque le serpent mange une pomme, le délais de rafraîchissement est réinitialisé, la valeur de *delay* est réduite et un nouveau délais de rafraîchissement est appliqué. Pour relancer le jeu et que le serpent aille à la bonne vitesse, *delay* reprend sa valeur initiale dans la fonction restart et le délais de rafraîchissement est encore une fois modifié pour retrouver sa valeur d'origine.

Bilans et perspectives :

Ce que je pourrai améliorer sur mon projet : rajouter des options pour changer les couleurs ou des nouvelles possibilités pour personnaliser le jeu tel que des pommes qui feraient ralentir le serpent.

Ce que j'ai appris en travaillant sur ce projet ?

J'ai beaucoup appris avec ce projet, n'ayant à l'origine que peu de connaissances et d'expériences en programmation, j'ai appris certaines bases ainsi que le fonctionnement en général du JavaScript. J'ai aussi appris que le travail en collaboration et le partage des tâches pouvait s'avérer plus compliqué à gérer que prévu mais cela m'a donné une vision à petite échelle du travail et de la gestion de projet dans le domaine de l'informatique. Bien que compliqué à certains moments, la collaboration et l'entraide à tout de même pu se faire au sein du groupe.

Dictionnaires des variables :

canvasWidth : donne la largeur du canvas en pixel.

canvasHeight : donne la longueur du canvas en pixel.

blockSize : sert à définir la taille des blocs (le canvas est découpé en blocs).

ctx : sert à enregistrer le contexte du canvas.

delay : sert à indiquer la valeur du délais de rafraîchissement du canvas.

snakee : va prendre en charge la création du serpent.

applee : va prendre en charge la création de la pomme.

widthInBlocks : permet de déterminer la largeur du canvas en bloc.

heightInBlocks : permet de déterminer la longueur du canvas en bloc.

score : Prend la valeur du score, elle est initialisée au début.

bestScore : Prend la valeur du meilleur score.

timeout : enregistre le délais de rafraîchissement du canvas.

Dictionnaire des fonctions :

init : sert à initialiser le jeu.
- Création dans la variable du même nom
- Il prend la taille définie par *canvasWidth* et *canvasHeight*.
- Application de CSS (bordure, centrer, couleur d'arrière plan).
- Récupération du body de la page HTML afin d'y accrocher le canvas (en utilisant *document.body.appendChild(canvas)*).
- Application d'un *contexte* au canvas, puis appel des fonctions Snake et Apple afin de dessiner le serpent et la pomme.
- Appel de la fonction *refreshCanvas*.

RefreshCanvas : actualise le canvas pour déplacer le serpent.
- Appel de la fontion *advance* sur le serpent (*snakee*) pour le faire avancer.
- Si le serpent mange une pomme :
  - Augmentation du score
  - Nouvelles coordonnées appliquées à la pomme (*applee*) grâce à *applee.setNewPosition*.
  - Remise à 0 du délais de rafraîchissement.
  - Réduction du délais selon le score.
  - Vérification de la nouvelle position de la pomme, si elle est sur le serpent, donner une nouvelle position.
- Remise à 0 du canvas.
- Affichage du score
- Affichage du meilleur score
- Affichage du serpent aux nouvelles coordonnées.
- Affichage de la pomme aux nouvelles coordonnées.
- Nouveau délais de rafraîchissement