Projet Snake

SENET Alexandre - GIRARDIN Florian / INFRES12

I. Choix techniques

Nous avons décidé de réaliser le Snake grâce à un tableau de tableau. Chaque partie du serpent comprend donc deux informations : x et y qui rendent très simple la récupération de la position de la tête et du reste du corps.

Tout ce qui permet l'affichage de la partie graphique est basé sur un canvas. Ce dernier est remplie grâce à une matrice à laquelle on attribue les valeurs comme indiqué ci-après :

- Case "vide" représentée par un 0

- Fruit représenté par un 1

- Tête représentée par un 2

- Corps représenté par un 3

- Mur représenté par un 4

Chaque chiffre permet de choisir l'image à afficher. Cela implique néanmoins que toute la matrice est chargée à nouveau à chaque déplacement du serpent.

L'ensemble du projet est réalisé en suivant le modèle MVC fournit.

II. Le modèle MVC

1) Modèle

Il contient l'initialisation de certaines variables ainsi que toutes les fonctions de retour au contrôleur.

2) Vue

Elle comprend principalement des fonctionnalités d'initialisation pour charger notamment les images et les skins. Un tableau est rempli d'images chargées une fois lors de l'appel au constructeur. On retrouve donc le sol, le mur, le fruit, le corps du serpent et sa tête. Pour gérer la rotation de cette dernière lors du déplacement du serpent, nous avons d'abord tenté d'utiliser les fonctions de rotations incluses mais sans grand succès. Dans un souci de simplicité, nous avons donc décidé de charger 4 images différentes en fonction de la direction et n'avons gardé qu'une image pour représenter chaque partie du corps .

On trouve également une fonction de mise à jour. Elle est appelée à chaque déplacement du serpent et parcours la matrice pour afficher l'image correspondante.

Enfin, une fonction qui rend plus clair une défaite pour l'utilisateur. Nous avons choisi de remplir le canvas d'un rectangle blanc légèrement transparent en plus d'afficher un message sur le côté.

3) Contrôleur :

Le contrôleur possède la plupart des fonctionnalités puisqu'il permet le lien entre le modèle et la vue.

La fonction initListener permet de définir quelles touches seront utilisées pour entre autres contrôler le serpent. Elle permet également de récupérer les éléments html pour utiliser les boutons de choix de niveau et de reset dont nous parlerons ensuite.

La fonction initGrid permet d'afficher le sol, les murs, le fruit initial et la tête du serpent. Nous remplissons la matrice de 0 ou de 4 pour dessiner les sols et les murs puis nous ajoutons le serpent aléatoirement dans le canvas en s'assurant qu'il ne puisse être dans un mur. Enfin, nous ajoutons le fruit en faisant également la vérification. Nous utilisons cette fois une fonction placeFruit qui servira à replacer un fruit chaque fois que celui-ci est mangé, en plus de l'initialisation.

La fonction moveSnake permet le replacement du serpent au bon endroit dans la matrice. Elle est lancée grâce à un timer et évalue la direction pour modifier la position du snake.

Algorithme :

- Récupérer la tête.

- La tête devient corps.

- Récupérer la queue.

- Incrémenter la position de la tête récupérée en fonction de la direction et ajouter cette nouvelle position au serpent.

- Tester si un mur, un fruit ou un morceau du corps est à la nouvelle position.

- Si c'est un mur ou le corps, on lance la fonction de fin de partie killSnake. Si c'est un fruit, on place un nouveau fruit et augmente le score. Le score est ainsi affiché à l'utilisateur à chaque fruit mangé. A noter que le score est calculé en fonction du niveau de difficulté sélectionné (moyen par défaut).

La fonction killSnake met à jour le tableau des meilleurs scores et la direction à null. En ce qui concerne le tableau des meilleurs scores, nous avons décidé de créer un top5 persistant. Pour cela nous utilisons la fonctionnalité localStorage de window. Un bug non identifié pour le moment fait que les meilleurs scores sont dupliqués. Cela est probablement dû au côté asynchrone de la fonction setInterval, mais aucune correction n’a été trouvée pour le moment.

La fonction de reset permet de relancer une partie à chaque instant. Elle est utilisée dans d'autres fonctionnalités mais peut également être appelée par l'utilisateur grâce au bouton retry ou à la touche de barre d'espace.

La fonction du choix de la difficulté permet à l'utilisateur, via 3 boutons, de varier le niveau. En pratique, ce choix influe sur la vitesse de rafraîchissement du timer et donc, pour l'utilisateur, sur la vitesse de déplacement du serpent. De plus, davantage de points sont attribués à chaque fruit mangés si le niveau est plus difficile. Changer le niveau de difficulté reset la partie automatiquement.

La fonction du choix du skin du serpent permet de modifier le skin du serpent. Nous avons créé de petit canvas cliquable qui représente les skins utilisables actuellement dans le jeu. Le choix d'un skin reset la partie automatiquement.

III. Notice d'utilisation

A l'arrivée sur la page, plusieurs possibilités s'offrent à l'utilisateur.

Il peut décider de jouer en utilisant les touches fléchées de son clavier, ou les touches Z,Q,S,D, pour déplacer le serpent.

Il peut consulter le top 5 des joueurs.

Il peut sélectionner un niveau de difficulté (par défaut moyen).

Il peut choisir parmi les différents skins (par défaut, le skin le plus à gauche).

Il peut utiliser le bouton reset ou la barre d'espace pour réinitialiser la partie si la disposition de départ ne lui convient pas.

Cas d'erreur :

Nous avons pu noter chez l'un de nos testeurs utilisant Google Chrome un cas que nous n'avons su résoudre. Au lancement du fichier .html depuis l'explorateur Windows, les canvas ne se chargent pas si Google Chrome n'a pas déjà au moins un onglet d'ouvert et il faut alors reset une fois pour pouvoir jouer. Si un onglet est déjà ouvert, aucun problème. Ce cas ne s'est pas présenté chez chez l'autre testeur mais nous n'avons pas pu tester à grande échelle pour voir si le problème est récurrent. L'utilisation de window.onload peut potentiellement ne pas être suffisante.