LEGER
Romain

LP CDED

Nouveaux modes de développement
Rapport n°5 du 14/10/2015

1. Gestion des collisions (1 heure)

La détection des collisions étant déjà effectué, il fallait dorénavant gérer les événements et les conséquences de celles-ci. Tout d'abord, j'ai créé une nouvelle variable liée à chaque snake, qui va gérer le nombre de vies du joueur. Lorsque la tête d'un snake entre en collision avec un autre serpent, alors il perd une vie. Dans le même sens, si les 2 têtes se touchent, alors les deux joueurs perdent une vie.

Lorsqu'un joueur touche un autre serpent, il perd alors une vie, et sa position est réinitialisée aléatoirement sur la carte. J'ai aussi géré l'affichage des vies pour chaque utilisateur dans une balise <p> dans la page principale.

2. Connexions et historique (1 heure)

Cet événement généré par les collisions amenait directement un problème : En effet, lors de la connexion d'un nouveau joueur, son snake apparaissait toujours au même endroit, en position (200,100). Donc si deux joueurs se connectaient à moins de 3 secondes d'intervalle, les snakes se touchaient directement, ce qui n'était vraiment pas optimal. Pour la coordonnée x, il était difficile de la modifier, car elle était utilisée pour initialiser les autres cercles, avec un certain décalage. Mais la position y pouvait être changée, car c'est la même pour tous les disques. Donc la position en y du serpent est générée aléatoirement, et son x reste fixe.

Un problème de performance était observé. En effet, dès que 3 joueurs ou plus se connectaient, le jeu devenait très lent. Je pensais que c'était lié au nombre de joueur. En guise de test, j'ai laissé pendant 10 minutes un seul snake sur la page, et le résultat était le même. C'était donc un autre problème, que j'ai identifié rapidement : L'historique. Celui-ci, servant pour le déplacement des disques derrière la tête, était incrémenté à chaque déplacement et pour chaque disque, puis envoyé au client. Chaque client recevait donc très rapidement un très grand nombre de données, et toutes non obligatoires. Donc j'ai forcé la taille de l'historique à l'écart entre 2 disques (10 dans le cas présent), et à chaque déplacement, je déplace d'une case vers la gauche chaque valeur, et ajoute à la fin la position précédente du disque. Dans ce cas, à chaque nouveau déplacement, plus d'opérations étaient effectuées, mais la taille des données de l'historique envoyée au client était de 10, et non plus d'une valeur devenant très lourde au fil du temps.

3. Fin de partie (30 minutes)

Maintenant que le principe des vies était mis en place, il fallait gérer la mort des snakes. Pour cela, nous initialisons le nombre de vies à 5, et lorsqu'un serpent n'a plus de vie, un message apparaît à l'écran de l'utilisateur qui contrôlait ce snake, pour signaler de la fin de partie pour lui. Il peut toujours rester sur la page, mais ne peut effectuer aucune action, ne contrôlant plus de snake. Dans ce cas, le serpent est supprimé de la vue de tous les clients, et le jeu continue pour les autres. L'utilisateur perdant peut recharger la page pour jouer à nouveau.

4. Musique, couleurs et aspect général (1 heure 30 minutes)

Le jeu étant de plus en plus jouable, il fallait améliorer l'aspect physique du jeu (contrairement à l'aspect pratique fait précédemment). J'ai donc ajouté un lecteur dans la page principale, pour jouer une musique lorsqu'un utilisateur est connecté. J'ai pour l'instant pris une musique que j'aimais bien, mais elle sera modifiée par la suite pour faire plus "jeu arcade".

Un autre aspect me dérangeait : La couleur des snakes. En effet, ils avaient tous la même couleur, (hormis la distinction tête et corps), et donc nous ne savons pas quel snake nous contrôlons. J'ai pensé à faire une couleur différente entre le snake de l'utilisateur courant, et même couleur pour les autres. Mais comme je n'ai pas très bon goût pour le choix des couleurs, j'ai décidé que chaque joueur aura un snake atypique. Un tableau de plus de 100 couleurs différentes est créé, et chaque joueur, à sa connexion, tire au hasard une première couleur pour la tête, et une autre pour le corps. Donc chaque joueur a un snake différent des autres (à moins d'une extrême malchance), et pourra facilement savoir (notamment grâce au clic) lequel est le sien.

Enfin, j'ai modifié l'emplacement des vies, n'étant plus tout en bas de la page, mais à droite du canvas, pour une vision plus facile du jeu et des vies de chaque joueur.

5. Déconnexion client et crash serveur (30 minutes)

Un problème intervenait souvent : Lorsqu'un joueur se déconnectait, le serveur se coupait. Ce problème apparaissait une fois sur 3 environ, donc j'ai eu du mal à trouver ce qui n'allait pas. Après quelques recherches, j'ai trouvé comment le résoudre. A chaque fois que je veux envoyer un message à un client, je regardais s'il existait toujours dans mon tableau, et maintenant il teste s'il est encore connecté, juste avant l'envoi, grâce à sa propriété "readyState".

6. Gestion du score (1 heure)

La mort des snakes a rendu le jeu un peu plus intéressant, mais il fallait faire plus, car le joueur qui arrivait à "tuer" les autres serpents n'était pas récompensé. Alors j'ai pensé à un système de score pour chaque joueur, ou les règles sont les suivantes : Si un snake 1 entre en collision avec le corps du snake 2, alors le snake 2 gagne un point, car je considère que le snake 2 a effectué des actions pour bloquer le snake 1. Par contre, si la tête du snake 1 touche la tête du snake 2, alors ils perdent chacun une vie, et personne ne gagne de point, sinon il serait facile de gagner des points. Les scores sont affichés dans un tableau, juste à côté des vies de chaque joueur.

7. Aspect général, partie 2 (30 minutes)

Enfin j'ai terminé cette séance par une modification de l'aspect global de la page. J'ai modifié le fichier css de l'index pour mettre plus de couleurs, et avoir un tableau de vies et score plus clair. La vue était maintenant plus claire, les objets plus rapprochés pour avoir l'ergonomie améliorée. De plus, le numéro du joueur est affiché dans la partie haute de la page, pour qu'il connaisse son score et son nombre de vies.

8. Prochaine étape

La prochaine séance sera dédiée aux fichiers de tests des fonctions. Cela devrait prendre une bonne partie de la séance. J'améliorerais encore le jeu si j'ai d'autres idées, et je commenterais plus mon code afin de le rendre plus facile à comprendre.

Pour finir, si j'ai le temps en fin de séance, je commencerais à réfléchir à la présentation du projet devant les autres membres du groupe, savoir ce que j'expliquerai, ce que je montrerai à cette occasion.