LEGER
Romain

LP CDED

Nouveaux modes de développement
Rapport n°2 du 23/09/2015

1. Reprise et réorganisation (30 minutes)

Pour commencer, j'ai fait du nettoyage dans le projet, et réorganisé les dossiers. Tout d'abord, avec la création d'un dossier "rapport", qui contiendra tous les rapports de chaque semaine. Ensuite, j'ai ignoré les fichiers *crt* et *key* qui étaient présents sur le serveur. J'ai aussi rajouté un fichier *"Documentation.txt"*, qui explique aux utilisateurs et/ou administrateurs comment jouer au jeu, ou comment le lancer.

2. Dessiner un cercle (30 minutes)

Tout d'abord, il fallait reprendre ce que j'avais déjà fait, relire le code pour savoir où j'en étais exactement. *Paper.js* entant déjà installé, il ne restait plus qu'à trouver comment faire pour dessiner une forme. Après quelques courtes recherches, j'ai trouvé comment dessiner le cercle. Puis, j'ai modifié le canevas, pour obtenir plus d'espace de "jeu" sur la page.

3. Déplacement du cercle (2 heures)

Maintenant le cercle dessiné, il fallait trouver comment le déplacer sur la page. J'ai trouvé assez vite comment le déplacer directement d'un point à un autre grâce au clic. Pour cela, je supprimais le cercle existant, puis recréais un à l'endroit voulu. Puis, j'ai amélioré ce programme pour faire déplacer automatiquement ce cercle, sans avoir à le supprimer et le recréer. Mais ce n'était toujours pas le résultat attendu.

J'ai donc repris l'idée du *"snake"* traditionnel : un déplacement automatique dès le début. Donc à chaque frame, je le faisais se déplacer d'un pixel horizontalement. Nous avions une première ébauche du snake, mais maintenant, il fallait trouver comment faire l'intervention de la souris. L'idée a toujours été de faire une opération entre le point cliqué et le centre du cercle créé. Grâce à la documentation, j'ai trouvé un outil utile : le vecteur, qui permet d'avoir des informations entre deux points dans un canevas. Mais il fallait ensuite connaître ses propriétés afin de l'utiliser efficacement.

La première propriété choisie était l'angle. En effet, grâce à la valeur en degré de cet angle, nous pouvions déterminer, grâce au cosinus et au sinus, le déplacement en x et en y à prévoir pour le prochain déplacement. Après plusieurs difficultés (la fonction cosinus prenait la valeur en gradient), j'ai réussi à faire avancer du "bon côté" le cercle, afin que sa direction soit le point cliqué par l'utilisateur. Mais une autre technique plus rapide pouvait être utilisée, je l'ai donc codée après cette première solution. Il fallait en effet normaliser le vecteur, et les valeurs de déplacement en x et en y étaient directement données. Le cercle pouvait donc se déplacer selon le choix de l'utilisateur.

4. Construire le serpent (1 heure 30 minutes)

Le cercle dorénavant créé et déplaçable, la création d'un serpent était donc possible. Mon choix était de créer plusieurs cercles, chaque cercle correspondant à un bout du corps du serpent (hormis le premier, d'une couleur différente, dédiée à la tête). J'ai donc fait une fonction pour créer facilement les cercles, ce qui nous aidera par la suite. Puis, je leur donnais une valeur fixe d'écart en x et en y par rapport à la tête. Ce choix n'était évidemment pas le plus réussi, il fallait passer à une autre méthode.

J'ai eu l'idée après ce premier passage, d'effectuer un historique des précédentes positions de tous les cercles. Les cercles devaient avoir un décalage de "20" entre eux, donc pendant 20 frames (ce qui est très rapide en temps réel), le déplacement n'est qu'horizontal, en stockant toutes les positions pour chacun des cercles. Puis, au bout de 20 frames, chacun des cercles (hormis la tête), prent la valeur du cercle suivant, avec un retard de 20 frames. Les cercles suivent donc toujours le même chemin.

Enfin, j'ai géré la sortie du parcours de serpent, lorsqu'il sort du terrain, il réapparaît à l'opposé du canevas. (Dépassement en x=1000, remise en *x*=0, sans changement du y, et inversement). Le déplacement du snake était donc opérationnel, il reste encore quelques étapes avant qu'il soit totalement jouable.

5. Animation avec WebSocket

Pour pouvoir utiliser l'outil WebSocket, il faut tout d'abord l'installer, avec *npm install --save ws*. Puis, il faut modifier le fichier *server.js* afin d'intégrer cette technologie au serveur. J'ai donc complété ce fichier, grâce à la documentation accessible sur Internet, pour permettre de l'utiliser. Cela n'a pas été simple, car il fallait ajouter une surcouche WWS à HTTPS pour l'utiliser correctement, puis ajouter des fonctions de réception de message et de connexion. Puis, j'ai modifié le fichier client *canvasPerso.js*, pour émettre un message lors de la connexion au serveur. Après quelques problèmes lié au fichier *server.js*, les premiers messages sont arrivés sur le serveur. J'ai ensuite testé différentes fonctionnalités, pour être plus à l'aise avec cet outil.

6. Prochaine étape

La prochaine tâche consiste à envoyer directement l'information du serveur au client, car actuellement, grâce au JavaScript, tout est géré niveau client. Dans le but de faire un site multijoueurs, il est indispensable que ce soit le serveur qui s'occupe de l'environnement. Je ferais donc cette partie durant la matinée du 30 septembre, puis lors de l'après-midi, commencer la partie multijoueur du site, en gérant les connexions (et création des serpents par joueur), les déconnexions et enfin les collisions.