ProgRes - Projet NodeJS

Programmer un Jeu Multijoueur en Temps Réel avec NodeJs

Le but du projet consiste à utiliser NodeJs pour créer jeu multijoueur client/serveur en temps réel. Pour rester simple, seul le serveur connaitra le fonctionnement du jeu. Les clients se connecteront depuis leur navigateur internet, pour récupérer l'état du jeu et envoyer des commandes au serveur, qui modifiera l'état de leur joueur, et enverra l'état du jeu à intervalle régulier à tous les joueurs.

Définition du jeu

Le jeu se joue dans un monde fermé, en deux dimensions. Chaque joueur contrôle une **entité joueur** (par exemple un tank, un personnage, un chat, etc...). Le joueur peut déplacer son **entité** dans le monde à l'aide des touches haut, bas, droite, et gauche du clavier. Le joueur peut envoyer des **balles** (peut-être de simples balles, ou d'autres objets, par exemple, des rockets, des boomerang, des suchis, etc...). Lorsque l'utilisateur clique sur le canvas, une balle doit partir de l'entité du joueur en direction de l'endroit ou l'utilisateur a cliqué, et continuer sa route jusqu'a rencontrer un autre joueur ou un élément de décors. La balle doit se déplacer à vitesse constante (plutôt rapidement), ainsi, si un joueur est sur la trajectoire d'une balle, il doit avoir le temps de se déplacer pour éviter la balle. (exemple de jeu qui ne contient pas tout ce qui est demandé : http://world-of-cats.bramas.fr)

Le monde dans lequel évolue les joueurs possède des éléments de décors statiques (des murs par exemple). Les entités des joueurs et les balles, ne peuvent pas traverser ces éléments de décors. En particulier le monde devra être entouré de murs pour éviter que les joueurs sortent du canvas.

Chaque joueur possède un score qui correspond au nombre de fois où une de ses balles a touché un autre joueur moins le nombre de fois où une balle l'a touché (sans être négatif).

L'entité des joueurs, les balles, et les éléments de décors doivent être des images désiné dans un canvas de taille fixe (le fond du canvas pourra aussi être recouvert d'images, comme de l'herbe par exemple).

Au dessus, de chaque entité de joueur, le nom du joueur doit être écrit (on pourra commencer par supposer que le nom du joueur est son identifiant), et le score doit être afficher dans la page (dans le canvas ou autre part dans la page).

La vitesse de déplacement des joueurs ne doit pas dépendre du nombre de fois ou l'utilisateur appuie sur les touches, elle doit être constante pendant tout le temps ou une touche de mouvement est appuyée. Ensuite, les joueurs ne pourront pas tirer un nombre infini de balles : par exemple, on pourra supposer qu'il possède un chargeur de 5 balles et que de nouvelles balles sont ajoutées à son chargeur toutes les secondes (jusqu'a atteindre au maximum 5).

D'autre comportement pourrons être implémentés mais ne sont pas obligatoire pour avoir une très bonne note (voir paragraphe *Extensions possibles*).

Mise en Place du Jeu

Le jeu se composera de plusieurs fichiers javascript pour le client et le serveur. Le projet utilisera les dépendances suivantes:

- *browserify* pour assembler les fichiers javascript pour le client.
- *express* pour gérer le server http.
- *socket.io* et *socket.io-client* pour la communication en temps réel entre le client et le serveur.

Le client et le serveur communique par envoie d'événements.

Le Client

Le client doit afficher le monde, les entités joueur, et les balles. De base, la page html se compose uniquement d'un élement `canvas` :

```
<canvas id="canvas" width="1000" height="600">
    Mettez à jour votre navigateur!
</canvas>
```

Dans ce canvas, on peut utiliser l'API HTML5 pour dessiner des objets simples (principalement des images). Lorsque le client se connect au serveur, il doit recevoir du serveur un identifiant unique. Ensuite, doit stocker tous les objets qui compose l'environement, et ces objets vont être mis à jour régulièrement lorsque le serveur enverra des mise à jour à intervalle régulier.

Lorsqu'une touche est enfoncée ou relachée (événement `keydown` ou `keyup` du document), un événement doit être envoyé au serveur.

Lorsque l'utilisateur clique avec sa souris dans le canvas (événement `click` sur le canvas), un événement doit aussi être envoyé au server.

Une fois que tout fonctionnera, le client pourra afficher le score (soit à côté du nom des jouers, soit en haut à droite du canvas, soit en dehors du canvas dans la page html).

Le Serveur

Le serveur se compose du fichier `serveur.js` et du fichier `game.js`. Le fichier `serveur.js` initialise la connexion avec les clients. Lorsqu'un nouveau client se connecte, il crée un identifiant unique et l'envoie à ce client. Le fichier `game.js` contient les objects du jeu ainsi que toutes les méthodes permettant de modifier l'état du jeu avec le temps et en fonction des inputs envoyés par les joueurs.

Les objects (entités joueur, balles et décours) sont stockés dans trois structures (`players` , `bullets` et `map`) pouvant être des tableaux ou des objects.

Lorsque le serveur reçoit des inputs (concernant les mouvement d'un joueur), le serveur doit en déduire la direction vers laquelle le joueur veut se déplacer (si BAS et GAUCHE sont appuyé alors le joueurs veut se déplacer dans la direction `(1, -1)`).

Le jeu est mis à jour dans une fonction qui (1) fait avancer les entités joueurs en fonction de leur direction, (2) fait avancer les balles en fonction de leur direction, (3) résoud les collisions entres objets (joueur/décors, balles/décors, balles/joueur, et joueur/joueur).

L'état du jeu doit être envoyé régulièrement à tous les clients.

Implémentation des collisions

Il est conseillé de donné à chaque objets (entité des joueurs, balles, élément de décor) les propriétés suivante : position dans le canvas (x, et y), largeur et hauteur (correspondant souvent au dimensions de l'image utilisé pour représenté l'objet).

Ainsi pour vérifier les collisions entre deux objets, il suffit alors de vérifier si les rectangles qui les définissent s'intersectent :

```
function collide(obj1, obj2) {
    return obj1.x + obj1.width > obj2.x &&
           obj2.x + obj2.width > obj1.x &&
           obj1.y + obj1.height > obj2.y &&
           obj2.y + obj2.height > obj1.y;
}
```

Extensions Possibles

Le projet doit être fait de manière progressive. Commencer par afficher uniquement une image pour chaque joueur et gérer le déplacement des entités joueur. Faites en sorte que les positions de chaque joueur soient synchronisées parmi tous les

clients.

Ensuite ajouter des éléments de décors (ce n'est pas la peine d'envoyer les éléments de décors à chaque fois, une fois au début c'est suffisant) et les afficher du côté du client. Ensuite gérer les collisions entre les éléments décors et les joueurs.

Une fois cela effectué, vous pourrez ajouter les balles, leur apparition lors des cliques de souris et leur déplacement en fonction du temps. Puis la gestion du score. Les collisions joueurs/joueurs pourront être gérées à la fin.

Pour la notation, même si tout ce qui est décrit plus haut n'est pas terminé, il est possible d'avoir la moyenne.

Une fois la base du jeu terminée, les extensions suivantes pourront être abordées (ordre arbitraire):

- Possibilité de choisir un nom (visible au dessus de son personnage).
- Inclure d'autres objets dans l'environement (tel que des bonus permettant de se déplacer plus vite, ou permettant d'augmenter la capacité de sont chargeur, ou des piege faisant diminuer le score)
- Ajouter d'autre type d'armes. Et la possibilité de changer d'arme en cliquant sur une touche en particulier. Par exemple des armes dont les balles vont plus vite, ou font des dégat de zone, des mines, etc...
- Ajouter des missions, par exemple ajouter une zone dans le monde, et le but et de rester sur cette zone pour faire augmenter son score.