Compte rendu pacman.io

Par Florian Vazelle et Geoffrey Glaive,
pour le cours de développement web avancé.
Source : https://github.com/florianvazelle/pacman.io

Choix des technos

Phaser 3

Phaser 3 est un moteur de jeu pour navigateur web, très complet et open source.

NodeJS

Le choix d'un serveur nodejs a été fait car nous voulions développer l'ensemble du projet en Javascript.

Fonctionnalités

Coté serveur

Au lancement du serveur :

1. Ce dernier génère un labyrinthe qui est exporté au format JSON.
2. Ce fichier correspond à la map entière.
3. Ensuite, ce fichier est séparer en "chunks"*.
4. Chaque "chunks" correspond à un fichier JSON.
5. Lorsque cette dernière opération est finie, le serveur se crée et commence à écouter les connections.
6. Les clients sont redirigés vers le dossier "public/".

7. Le jeu peut commencer.

La map change donc à chaque fois que le serveur est lancé.

Coté client

Lorsque qu'un client se connecte à l'adresse du serveur :

1. Le client a la possibilité de se choisir un pseudonyme.
2. Lorsqu'il clique sur le bouton "Start", son pacman s'initialise avec des coordonnées au hasard sur la map et un score de zéro.
3. Le client peut bouger son pacman vers le haut, le bas, la droite et la gauche, à l'aide des touches directionnelles.
4. Sur mobile, le client n'ayant pas de touche directionnelles, il doit cliquer sur l'écran pour voir apparaître un joystick. Le client peut déplacer son doigt où il veut, et une fois relâché, le joystick disparaît pour lui permettre de réappuyer où il veut.
5. Le but du jeu est d'avoir le meilleur score.
6. Le client augmente son score en déplaçant son pacman sur des boules, qui lui font gagner un point.
7. Les scores des 5 premiers et de votre pacman sont affichés dans le leaderboard qui est mis à jour à chaque frame.
8. Il est impossible pour le pacman de quitter le labyrinthe, car les murs sont infranchissables.
9. Les pacmans qui ont un plus gros score que celui du joueur sont de couleur rouge, ceux qui ont un plus petit score sont vert et ceux ayant le même sont jaunes.
10. Au contact d'un pacman de plus grand score, le pacman du client est "mangé". Il disparaît puis réapparaît ensuite à une location aléatoire et son score retombe à zéro.
11. Au contact d'un pacman de plus petit score, ce dernier est "mangé", et son score s'additionne au pacman du client.

*chunks : parcelle de la map

Architecture

```
C:\USERS\...\DOCUMENTS\GITHUB\PACMAN.IO
|  .gitignore
|  package.json
|  README.md
|  server.js                          Le serveur NodeJS avec Koa
|
+---lib                               Fonctions utilisées par le serveur pour générer la map,
|  |  toolbox.js                      divisée en deux sous-dossiers :
|  |
|  +---creator                        - Pour créer le labyrinthe
|  |     createmaze.js
|  |     maze.js
|  |     template-map.json
|  |
|  \---splitter                       - Pour séparer le labyrinthe en chunks (adaptée de Jerenaux/chunks_tutorial)
|        splitmap.js
|
\---public                            Ensemble des fichiers du jeu, c'est la partie client
   |  index.html
   |
   +---assets                         Ensemble des fichiers qui ne sont pas du code :
   |  +---fonts                       - Police de caractéres
   |  +---sprites                     - Images
   |  |  +---joystick
   |  |  |     base.png
   |  |  |     body.png
   |  |  |     cap.png
   |  |  |
   |  |  +---menu
   |  |  |     startBtn.png
   |  |  |
   |  |  \---pacman
   |  |        pacman_green_sprite.png
   |  |        pacman_red_sprite.png
   |  |        pacman_sprite.png
   |  |
   |  \---tilemaps                    - Fichier utile à la construction de la map (dans le jeu)
   |     +---maps                     La map et ses chunks en format JSON
   |     |  |  .gitignore
   |     |  |  fullmap.json
   |     |  |
   |     |  \---chunks                Les chunks sont stockées ici
   |     |        chunk0.json         Leur nombre dépend de la taille de la map
   |     |       .
   |     |       .
   |     |       .
   |     |        chunk69.json
   |     |        master.json
   |     |
   |     \---tiles                    Image de toute les textures de la map
   |           pacman-tiles-32x32.png
   |
   \---scripts                        Ensemble des fichiers javascript du jeu
     +---entities
     |     Pacman.js
     |
     +---scenes
     |  |  GamePlay.js                Le jeu
     |  |  GlobalVariables.js         Les variables globales
     |  |  MainMenu.js                Le menu
     |  |  Preload.js
     |  phaser.min.js                 La librairie Phaser 3
     |
     \---plugins                      Le dossier des plugins utilisés dans le jeu
           VJoy.js                    Virtual Joystick (adapté d'un code sous Phaser 2)
```

Problèmes rencontrés

Position et Vélocité

Le pacman était déplacé par ses positions x et y. Pour l'oeil humain, le sprite du pacman se déplaçait normalement, mais en réalité, pour le moteur du jeu, il se téléportait.

Vu que nous déplacions le pacman comme ceci :

```
if (upKey.isDown) {
    pacman.y -= 10;
} else if (downKey.isDown) {
    pacman.y += 10;
} else if (leftKey.isDown) {
    pacman.x -= 10;
} else if (rightKey.isDown) {
    pacman.x += 10;
}
```

De case en case, et non comme ceci :

```
pacman.body.velocity.x = 0;
pacman.body.velocity.y = 0;
if (upKey.isDown) {
    this.physics.moveTo(pacman, pacman.x, pacman.y - 10);
} else if (downKey.isDown) {
    this.physics.moveTo(pacman, pacman.x, pacman.y + 10);
} else if (leftKey.isDown) {
    this.physics.moveTo(pacman, pacman.x - 10, pacman.y);
} else if (rightKey.isDown) {
    this.physics.moveTo(pacman, pacman.x + 10, pacman.y);
}
```

Qui affecte la vélocité du sprite, c'est une fonction du moteur de jeu pour déplacer son sprite et ainsi le sprite est sensible aux autres objets physiques.

Map

Au commencement, la map était généré par des boucles pour créer des objets physique à chaque fois qu'il y avait un mur ou une boule. Cependant ce système marchait pour les

très petites maps et le calcul des collisions était lourd, ce qui pouvait parfois ralentir la machine du client.

Puis nous avons découvert qu'il était possible de créer une map au format CSV, où chaque nombre était associé à un sprite, ce qui permettait d'afficher des map beaucoup plus grosses.

De plus, quand bien même la collision était gérée, Il était impossible de faire disparaître le sprite d'une boule au survole de celle-ci. En effet, la map était statique et il fallait donc y inclure des layers*. Le format CSV ne permettait pas de stocker autant d'information, il fallait donc générer cette fois un fichier de map en JSON. Ainsi nous avons pu créer plusieurs layers, dynamique et statique.

De plus, pour que le jeu puisse traiter des maps de tailles différentes, il fallait ajouter une fonctionnalité supplémentaire : celle des chunks. Les chunks permettaient de ne pas afficher toute la map mais seulement les parcelles autour du joueur, plus précisément celle où se trouve le pacman du client et celles adjacentes.

*layer : couche