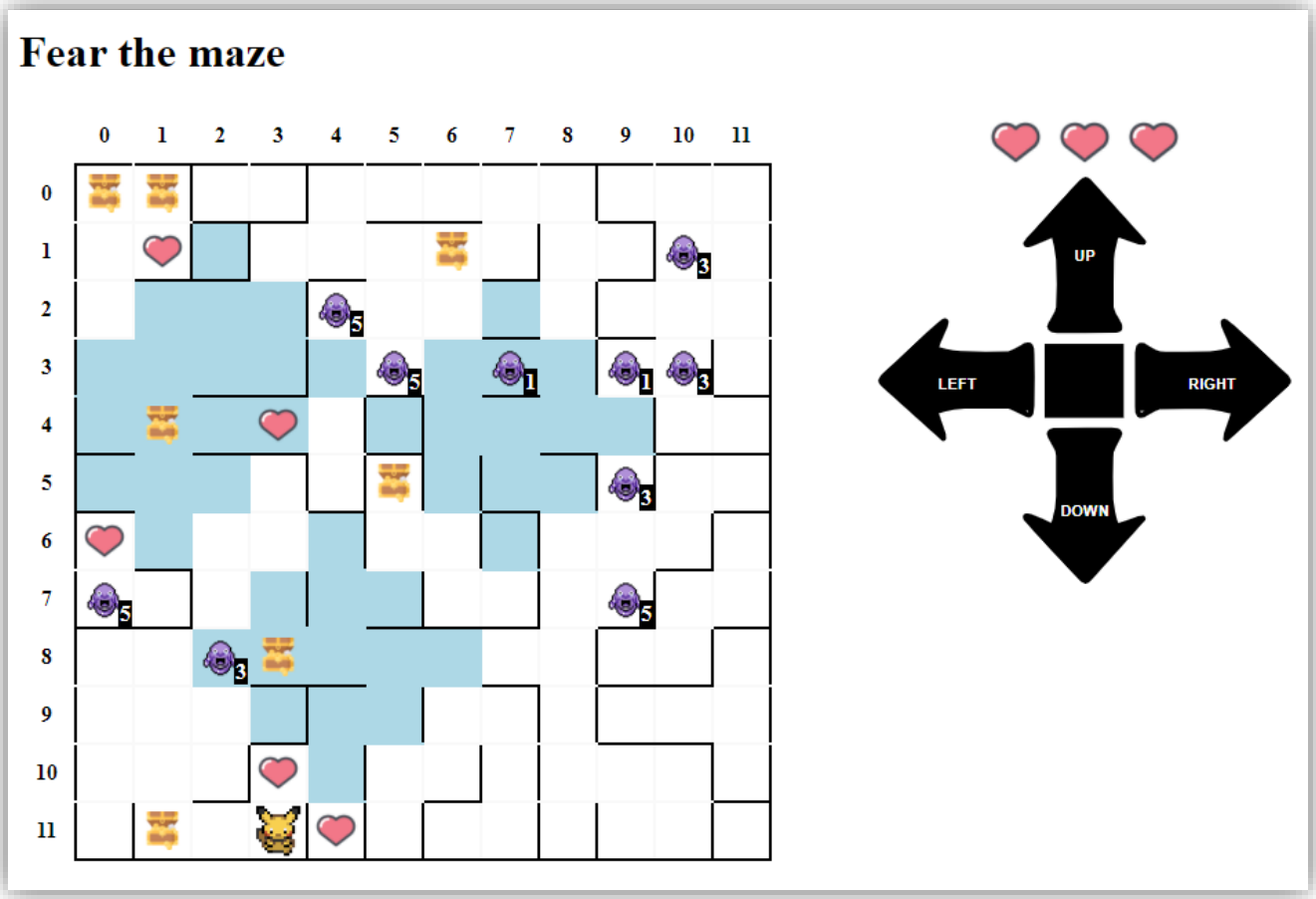
JavaScript

TP : Fear the maze

Nicolas LETHUILLIER
nlethuillier@gmail.com

Introduction

Fear the maze est un labyrinthe dans lequel le héros, une créature jaune aux joues rouges et aux oreilles noires garantie sans licence, doit récupérer des trésors, combattre des monstres, etc.



Le labyrinthe est un tableau avec **N** colonnes (abscisse) et **N** lignes (ordonnée).

Le héros est facilement identifiable, et apparaît à un emplacement aléatoire dans le labyrinthe.

Étape par étape, nous allons implémenter les différentes fonctionnalités du labyrinthe, et faire que le jeu soit jouable à l'aide des boutons à droite.

Afin de simplifier les opérations, et parce que le but est de vous familiariser avec la programmation en général, on met à votre disposition la bibliothèque ftm.js. Cette bibliothèque gère l'interfaçage entre vos fonctions et le fichier HTML. Elle vous offre les fonctions suivantes, que vous serez libres d'utiliser dans votre script :

createGrid(options) → ∅ *Modelizes the grid depending on the parameters included in options, and then displays the grid according to all the settings.* *Options can contains the following attributes:* - *size: int ; the width/height of the grid (**mandatory**)* - *walls: bool* - *hearts: bool* - *enemies: bool* - *loot: bool* - *ice: bool*
getCellContent(x: int, y: int) → EMPTY | HEART | LOOT | MONSTER *Returns the content of the cell, returning one of the defined constants*
getHearts() → int *Returns the number of hearts of the hero*
getHeroPosition() → { x: int, y: int } *Returns the current position of the hero in the grid.*
getMonsterPower(x: int, y: int) → int | null *Returns the power of the enemy in the given coordinates or null if there ain't any.*
getTreasures() → int *Returns the number of treasures on the grid*
getWalls(x: int, y: int) → { north: bool, east: bool, south: bool, west: bool } *Returns an object describing the walls on the given cell*
isFrozen(x: int, y: int) → bool *Returns true if the given cell is frozen*
loot(x: int, y: int) → ∅ *Picks a treasure up and removes it from the grid*
setHearts(n: int) → ∅ *Sets the number of hearts of the hero*
setHeroPosition(x: int, y: int) → ∅ *Moves the hero to the given position*

Une nouvelle partie est lancée à chaque fois que l'on rafraîchit la page.

Prérequis

- ✓ Un navigateur web (un vrai, pas Internet Explorer)
- ✓ Un IDE

Initialisation

- Extraire le projet à l'emplacement de son choix.
- Ouvrir le répertoire du projet avec un IDE.
- Créer le fichier **`script.js`** à la racine du projet.

La base du projet est construite. L'ensemble du jeu sera généré en JavaScript dans l'unique div du fichier HTML.

Affichage

- Éditer le fichier **`script.js`** et déclarer la constante **`options`**. C'est elle qui va contenir l'ensemble des paramètres du jeu (voir documentation de ftm.js).
- Lui ajouter ce qu'il faut d'attributs pour avoir une grille de taille 12 x 12.
- Appeler la fonction adéquate de ftm.js pour générer la grille à l'aide de ces paramètres.

Une grille s'affiche, avec votre héros positionné à une abscisse et une ordonnée aléatoires. Si vous actualisez, vous constatez que sa position change.

Déplacement

On souhaite pouvoir déplacer notre héros grâce aux commandes à droite de l'écran.

- Créer les fonctions **`goUp()`**, **`goDown()`**, **`goRight()`** et **`goLeft()`** qui déplacent le héros d'une case dans la direction choisie. Il est nécessaire de consulter la documentation de **`ftm.js`** pour savoir quelles fonctions de la bibliothèque appeler.
- Pour attacher les fonctions aux boutons, utiliser les lignes suivantes (nous étudierons plus tard leur fonctionnement) :

```
document.querySelector('#go-up').onclick = goUp;
document.querySelector('#go-down').onclick = goDown;
document.querySelector('#go-right').onclick = goRight;
document.querySelector('#go-up').onclick = goLeft;
```

Murs

- Adapter les options pour activer les murs.
- Adapter les fonctions **`goUp`**, **`goRight`**, **`goDown`** et **`goLeft`** pour empêcher le héros de traverser des murs. Dans le cas où il essayerait, rien ne doit se passer. Comme toujours, consulter la documentation de **`ftm.js`** pour savoir ce qu'on peut utiliser.

Cœurs

- Adapter les options pour activer les cœurs.

Au début de la partie, le héros doit avoir 3 cœurs.

- Appeler la fonction adéquate.

Lorsque le héros passe sur un cœur, il en gagne un supplémentaire.

- Adapter les fonctions de déplacement de façon à ajouter un cœur au héros s'il passe sur un cœur. Il y a 4 fonctions de déplacement, mais le comportement sera le même pour les quatre… peut-être peut-on factoriser ça dans une fonction !
- Faire en sorte qu'on ne puisse pas avoir plus de 5 cœurs.

Ennemis

- Adapter les options pour activer les ennemis.

On constate alors que sont apparus des monstres avec un nombre sous leur image. Cela correspond à leur puissance/vie, c'est-à-dire que c'est le nombre de cœurs qu'il faut avoir pour prendre le dessus et les vaincre, sans quoi la partie est perdue.

- Adapter votre code pour qu'une rencontre avec un ennemi retire des cœurs au héros et vainc l'ennemi, ou ouvre une boîte de dialogue pour indiquer que la partie est perdue (`alert`).

Trésors

- Adapter les options pour activer les trésors.

On constate alors que sont apparus des trésors. Le but du jeu va être de tous les récupérer.

- Adapter le code pour qu'un passage sur un trésor le récupère.
- Adapter le code pour qu'une boîte de dialogue apparaisse lorsqu'il ne reste plus aucun trésor à récupérer, pour indiquer à l'utilisateur qu'il a gagné.

Verglas

La dernière étape est d'implémenter du verglas. Tous les amateurs de monstres japonais sans licence se souviennent de ces grottes glacées où le moindre mouvement peut être fatal.

- Adapter les options pour activer le verglas.
- Adapter le code pour que, lorsque le héros marche sur une plaque de verglas, il avance d'une case supplémentaire dans la direction qu'il a prise. Il file ainsi dans la même direction jusqu'à un mur ou jusqu'à la prochaine case non verglacée.
  Essayer de répéter le moins de code possible.