W2 - Spécialisation

W-SPE-502

My_Battleship

Coulez votre adversaire !

1.0

{EPITECH.}

My_Battleship

repository name: my_battleship
repository rights: ramassage-tek
language: JavaScript

- Your repository must contain the totality of your source files, but no useless files (binary, temp files, obj files,...).

Détails Administratifs

- Le projet est à réaliser seul.
- Les sources doivent être rendues avec BLIH.

Introduction

Vous reprenez le projet inachevé (archive disponible avec ce sujet) de quelqu'un d'autre. Vous allez devoir le débugger et terminer le projet.
Nous vous recommandons de bien prendre connaissance de l'ensemble du code (il y a plusieurs fichiers inclus dans la page HTML). Il peut également être de bon ton de se renseigner sur les librairies tierces utilisées.
Le précédent développeur même s'il vous a laissé un projet inachevé, s'est appliqué à mettre en place de bonnes pratiques d'architecture, et/ou des astuces de code, inspirez-vous ! Par ailleurs il vous a laissé un peu de documentation et des commentaires dans le code, exploitez les !
Nous vous recommandons également de faire des "commits atomiques" (si ce terme ne vous dit rien, renseignez-vous ;)) cela vous permettra plus facilement de revenir en arrière si besoin.

RESTRICTIONS

Vous n'avez pas le droit d'utiliser des librairies autres que celles déjà incluses dans le projet (excepté pour l'exercice 17, pour ce dernier, seul socket.io est autorisée)
L'intégralité du code que vous rendez (même si ce n'est pas le vôtre à la base) doit être valide JSLint (http://www.jslint.com et sans changer les directives en en-tête des fichiers (si vous avez besoin de le faire, vous devrez être capable de le justifier en soutenance). Par ailleurs vous devrez être capable, en soutenance, d'expliquer n'importe quelle partie du code (même si ce n'est pas le vôtre à la base)

le code qui vous ait fourni est déjà (dans la mesure du possible) valide JSLint

PROJET

Les exercices n'ont pas toujours de rapport entre eux, il est tout de même très conseillé de les aborder dans l'ordre dans lequel ils sont donnés.

+ EXERCICE 0

Ouvrez la console (outils développeur) de votre navigateur, corrigez l'erreur JS qui y est affichée ("TypeError : player.setGame…").

Pour cet exercice, vous ne devez pas modifier le contenu du fichier game.js !

+ EXERCICE 1

Faire en sorte que les bateaux une fois placés s'effacent correctement de la grande map.

Pour cet exercice, vous ne devez pas modifier le contenu du fichier game.js !

+ EXERCICE 2

Implémentez la fonction renderMiniMap (déclarée dans le fichier game.js ligne 219, et appelée dans le même fichier à la ligne 160) qui a pour effet de colorer les cases de la minimap (sur la gauche) pour correspondre à l'emplacement des bateaux choisi par le joueur (les positions des bateaux sont enregistrées au moment où le joueur clique sur la grande grille).
Les différentes cases doivent être colorées de la couleur du bateau qui occupe la case.

+ EXERCICE 3

Faire en sorte d'empêcher que l'utilisateur puisse placer 2 bateaux sur une même case, ou qu'un de ces bateaux sorte du terrain.
Si ces conditions ne sont pas respectées, il ne doit pas être du tout possible de placer le bateau (on ne doit pas passer au bateau suivant).

+ EXERCICE 4

Faire en sorte de permettre à l'utilisateur de placer ses bateaux verticalement : pendant la phase de placement, le fait de faire un "clique droit" doit changer l'orientation du bateau.

+ EXERCICE 5

Faire en sorte que l'ordinateur place ses bateaux de façon aléatoire (en respectant les mêmes règles de placement que le joueur : pas de chevauchement, pas de sortie de la map).

Pour cet exercice, vous n'avez pas obligatoirement à créer de nouvelles fonctions.

+ EXERCICE 6

Faire en sorte que les cases où l'utilisateur a déjà tiré soient affichées sur la carte (rouge si c'est touché, gris si c'est manqué).

+ EXERCICE 7

Faire en sorte que si le joueur choisit une case où il a déjà tiré, le message soit différent du message classique, et l'information (touché ou manqué) ne change pas (par rapport au premier tir qui a eu lieu).

+ Exercice 8

Modifier le code pour que l'ordinateur joue de façon plus intelligente (sans pour autant "tricher", l'objet computer ne doit pas toucher à la propriété grid de l'autre joueur), actuellement il tire toujours dans la première case en haut à gauche…

+ Exercice 9

Une fois que l'ordinateur a joué (a fait une tentative de tir), si le joueur est touché (et uniquement dans ce cas-là) la "Mini Map" doit être mise à jour pour refléter la perte subie par le joueur. Si un bateau est complètement coulé, la classe css "sunk" doit être attribuée à l'icône du bateau correspondant (au-dessus de la "mini map").

+ Exercice 10

Une fois que l'ordinateur a joué (a fait une tentative de tir), la main devrait être redonnée au joueur, or actuellement, ce n'est pas le cas. Trouvez, ce qui provoque ce bug, corrigez-le.

+ Exercice 11

Implémentez la détection de fin de partie (fonction gameIsOver, dans le fichier game.js ligne 105 et appelée à la ligne 80).

+ Exercice 12

Implémentez la possibilité de choisir qui commence (joueur humain, ordinateur, ou aléatoire).

+ Exercice 13

Ajoutez du son pour chaque tir : un son indépendamment de la réussite du tir puis un son en cas de tir réussi, et un autre en cas de tri raté.

+ Exercice 14

Ajoutez des animations dans les cases de la grille consécutivement à un tir du joueur : une animation d'explosion pour un tir réussi, et une autre en cas de tri raté.

+ EXERCICE 15

Implémentez plusieurs niveaux de jeu pour l'IA.

niveau facile : tirs aléatoires
niveau difficile : tirs réfléchis

+ EXERCICE 16

Implémentez la possibilité d'avoir une aide de jeu : si le joueur le désir, l'ordinateur lui suggère une case à jouer.

+ EXERCICE 17

Implémentez une version multi-joueurs (en réseaux).

Si vous en êtes là, un assistant sera sûrement de bon conseil pour la suite des opérations.