IUT Nantes

TD4 – Tech Web 2

Mini – Projet Démineur

Contenu

Yann Rolland

15/12/2013

15 décembre 2013

Rapport TD4 – Mini-projet Démineur

I. Introduction

Ce projet a pour but de réaliser un démineur en PHP/HTML, en utilisant les notions de Session, URL longues et formulaires. Le sujet (ci-joint, « demineur.pdf ») demandait une version simplifiée à l'extrême de ce jeu. Avec l'accord de notre professeur de TD, j'ai réalisé ici une version avancée, c'est-à-dire conforme aux règles de bases du démineur.

II. Les différents fichiers

L'archive finale contient les sources et ressources du projet dont voici la description :

- Dossier « images » : contient les images au format 32x32, prêtes à être utilisées par le programme.
- Fichier « demineur.php » : contient principalement le code HTML de la page générée ainsi que les appels des fonctions PHP.
- Fichier « fonctions.php » : contient les fonctions PHP utilisées dans ce programme.
- Fichier « style.css » : contient le code CSS utilisé pour la mise en page de l'HTML généré.

III. Règles du jeu

Le but du jeu de démineur est de repérer les mines disposées aléatoirement sur le plateau de jeu. En cliquant sur une case, on découvre la case. Si cette case contient une mine, alors on a perdu. Dans le cas contraire, la case indique le nombre de mines qu'elle touche. Dans le jeu original, on peut indiquer les mines avec un marqueur (un drapeau), ici, comme la seule action possible est un clic gauche, et pour simplifier le gameplay, on essaiera de ne découvrir que les cases sûres. La victoire se fait lorsque toutes les cases ne contenant pas de mines sont découvertes.

IV. Fichier CSS

Je n'expliquerai pas ici les techniques CSS utilisées, ce n'est pas le but de ce TD. Le gros du travail est le centrage des éléments.

V. Fichier demineur.php

Ce fichier contient la structure de la page HTML ainsi que l'appel des fonctions de jeu, on note l'appel de session_start() pour pouvoir accéder aux variables de session utiles par la suite. L'instruction requiere('fonctions.php') permet d'inclure le contenu de ce fichier à cet endroit et donc de les rendre visibles. La page est valide HTML5 (http://validator.w3.org/), est encodée en utf-8 et utilise les variables de session pour vérifier le contenu des champs HTML.

VI. Fonctionnalités du démineur

Dans ce jeu, de multiples fonctionnalités et sécurités ont été implémentées, voici une liste non-exhaustive :

- Modularité : 3 fichiers séparés, utilisation d'un maximum de fonctions
- Configuration : les valeurs par défaut du démineur sont accessibles et modifiables au tout début du fichier « fonctions.php », les images sont remplaçables.
- Une case découverte indique le nombre de mines qu'elle touche.
- Une case n'est cliquable que si elle peut faire une action utile.
- On peut commencer une nouvelle partie à tout moment avec le bouton « nouvelle partie ».
- La taille du plateau de jeu est changeable via les champs « lignes » et « colonnes ».
- Les valeurs entrées sont vérifiées :
    - Dans les champs du formulaire : gestion par le navigateur des valeurs non-valides.
    - Dans les valeurs passées par URL longue : vérification systématique du respect des bornes.
- Toutes les cases sont découvertes en cas de défaite.
- Le nombre de mine est calculé en fonction de la taille du tableau et d'un coefficient configurable.
- La propagation du « trou » se fait lors d'un clic sur une case non minée et ne touchant pas une mine.
- Un message indique la victoire ou la défaite.
- Une fois la partie terminée, plus aucune action sur le plateau n'est permise.

VII. Constantes et variables principales

Nous verrons ici les principales constantes et variables utilisées tout au long du programme :

COEFMINE	permet de régler le nombre de mines, sur COEFMINE case, il y a une mine.
DEFAULTLIG	nombre de lignes par défaut.
DEFAULTCOL	nombre de colonnes par défaut.
MAXLIG	nombre maximal de lignes.
MAXCOL	nombre maximal de colonnes.
MINLIG	nombre minimal de lignes (ne devrait pas être changé).
MINCOL	nombre minimal de colonnes (ne devrait pas être changé).
$dir	contient les décalages (offsets) des 8 cases entourant une case.
$_SESSION['nblignes']	Nombre de lignes actuel.
$_SESSION['nbcolonnes']	Nombre de colonnes actuel.
$_SESSION['nbmines']	Nombre de mines actuel.
$_SESSION['genererMap']	Permet de savoir si c'est le premier lancement d'une session (n'existe pas) ou juste une nouvelle partie (false).
$_SESSION['GAGNE']	Indique une victoire.
$_SESSION['PERDU']	Indique une défaite.
$_SESSION['terrain']	Tableau indiquant si une case est minée (true) ou non (false).
$_SESSION['jeu']	Tableau indiquant si une case est cachée (true) ou vue (false).
$_SESSION['mapVoisins']	Tableau indiquant le nombre de mines voisines d'une case.

VIII. Fonctions et algorithmes

function tab2D($lignes, $colonnes, $val)

Cette fonction utilise la fonction array_fill pour retourner un tableau de $lignes lignes et $colonnes colonnes rempli de la valeur $val.

function dansBornes($val, $min, $max)

Fonction basique vérifiant si $val est compris entre $min (inclus) et $max (inclus).

function dansPlateau($i, $j)

Cette fonction vérifie les coordonnées d'une case dans le plateau : $i entre 0 et $_SESSION['nblignes']-1, $j entre 0 et $_SESSION['nbcolonnes']-1.

function tailleValide($nblignes, $nbcolonnes)

Cette fonction vérifie si les dimensions sont conformes aux constantes définies pour la taille du plateau.

function affCase($i, $j)

Cette fonction affiche une case (caractérisée par son image) en générant le code HTML nécessaire. Si la case n'est pas vue, on utilise l'image « unknown.png », sinon on regarde si c'est une mine, auquel cas on utilise l'image « mine.png », sinon on utilise l'image correspondant au nombre de mines voisines (de 0.png à 8.png). On ne place le lien avec URL longue que si la case est utilisable (non vue). Le lien correspond aux coordonnées de la case passées par paramètres : $i et $j.

function affPlateau()

Cette fonction affiche le plateau de jeu dans son ensemble en utilisant une <table> html. Chaque case correspondant à un <tr> et un <td> est parcourue et envoyée à affCase avec ses coordonnées.

function genererJeu()

Cette fonction génère le « jeu » : met toutes les cases en cachées.

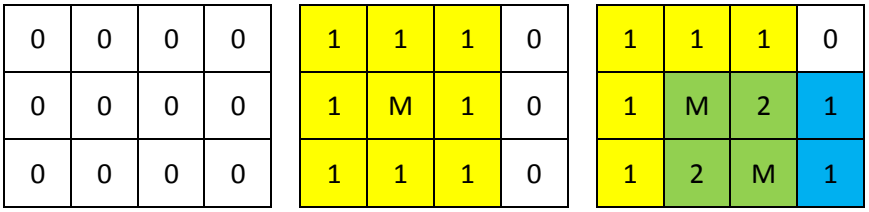
function genererTerrain()

Cette fonction génère le « terrain » : crée un tableau $_SESSION['terrain'] aux bonnes dimensions ne contenant pas de mines ainsi que le tableau $_SESSION['mapVoisins'] indiquant 0 pour toutes les cases. Elle tire ensuite aléatoirement une coordonnées jusqu'à avoir une case non minée, on place la mine dans $_SESSION['terrain'] puis on envoie ces coordonnées à ajouterVoisins() pour compléter la carte des mines voisines. On répète l'opération autant de fois que le nombre de mines demandé.

function ajouterVoisins($i, $j)

Cette fonction permet de compléter les informations des mines voisines : en plaçant une mine aux coordonnées ($i ;$j), les 8 cases voisines (dont le décalage est stocké dans $dir) doivent être incrémentées (si elles sont dans le plateau).

Exemple d'appels successifs (fusion de « terrain » (M pour Mine) et « mapVoisins ») :



Cet algorithme est plus performant que la version naïve qui consiste à placer toutes les mines puis d'itérer case par case toutes les cases en comptant les mines voisines car il ne modifie et ne parcourt que les cases ayant besoin d'êtres incrémentées.

function decouvrirJeu()

Cette fonction découvre toutes les cases non découvertes en itérant sur toutes les cases.

function initPartie($nblignes,$nbcolonnes)

Cette fonction détruit la session et la recrée pour plus de sécurité, initialise les dimensions du plateau suivant les paramètres, lance la génération du jeu, du terrain et initialise les variables de jeu, elle doit être appelée à chaque fois qu'une nouvelle partie commence.

function affInfo()

Affiche les informations de victoire ou de défaite selon les variables $_SESSION['GAGNE'] et $_SESSION['PERDU'].

function restePasMine()

Elle vérifie s'il reste des cases non minées et non vues en itérant sur toutes les cases, elle est utilisée pour vérifier la fin de partie.

function handleClick()

Cette fonction gère les actions du joueur qui sont passées par URL longue. Tout d'abord on vérifie si la partie est terminée ($_SESSION['GAGNE'] et $_SESSION['PERDU']) si ce n'est pas le cas, on teste la présence des valeurs 'i' et 'j' dans l'URL (méthode d'URL longue semblable au GET). Si elles sont présentes et valides (dans le plateau) et que la case n'était pas vue, on la place en vue. On vérifie ensuite le contenu de cette case. Si elle contient une mine, on indique la défaite et on découvre le jeu, sinon (case non minée), on propage le trou dans le cas où la case ne touche pas de mine. Enfin, on teste la fin de partie avec restePasMine().

function calculeNbMines($nblignes, $nbcolonnes)

Cette fonction permet de calculer le nombre de mines en fonction des dimensions du plateau et de COEFMINE. Le nombre retourné correspond au nombre total de cases divisé par COEFMINE, ce qui fait que sur COEFMINE cases, 1 case sera minée.

function handleReset()

Cette fonction gère la nouvelle partie et le premier lancement du jeu. Dans le cas du premier lancement, c'est-à-dire quand $_SESSION['genererMap'] n'existe pas, on utilise les valeurs par défaut pour initialiser la partie. Dans le cas où ce n'est pas le premier lancement, on vérifie si des dimensions sont passées en URL longue ($_GET['nblignes'] et $_GET['nbcolonnes']) et si elles sont valides (tailleValide($_GET['nblignes'],$_GET['nbcolonnes'])). On initialise ensuite la partie avec ces nouvelles valeurs.

function propagerTrou($i, $j)

Cette magnifique et spectaculaire fonction permet d'étendre un le trou suite à un clic sur une case non vue, non minée et ne touchant pas de mine. J'utilise un algorithme récursif pour parcourir les cases ayant besoin d'être modifiées, c'est-à-dire, qui doivent être découvertes. Pour cela, on part des coordonnées ($i ;$j), on met la case en vue, puis on regarde le nombre de mines qu'elle touche. Si elle n'en touche pas, on peut alors parcourir toutes les cases voisines non vues (qui, elles, pourront toucher une mine), si elles existent en effectuant la même opération. Cet algorithme est bien plus performant qu'un algorithme itératif qui demanderait d'examiner toutes les cases du plateau une par une et plusieurs fois en faisant toute une batterie de tests. Il correspond en fait à ce que ferait un joueur si la fonction n'existait pas : on sait qu'une case touchant une case non minée et égale à 0 n'est elle-même pas minée, et on poursuit autant que possible jusqu'à ne plus pouvoir trouver de cases voisines répondant à ces critères. Ainsi, le trou se propage.

IX. Conclusion

Le démineur final est entièrement jouable et respecte les règles du jeu original, il m'a permis de mieux cerner à la fois les mécanismes de sauvegarde des variables en PHP et les différentes protections à mettre en œuvre lors de la demande d'informations à l'utilisateur. On pourrait éventuellement proposer une manière simple de changer la difficulté du jeu (ajout d'un champ dans le formulaire qui agirait sur COEFMINE) et/ou ajouter une case à cliquer pour permettre de passer du mode « déminage » au mode « pose d'un drapeau ». Tout ceci serait facilement faisable mais la Gestion m'appelle.