FACULTÉ DES SCIENCES
Unité de formation et de recherche
université angers

L3 Informatique - 2024/2025
UE Développement Web
David Lesaint

CC PHP : Session 1 - 1h30 - Sur machine

Téléchargez et décompressez l'archive déposée sur Moodle pour cet examen. Le dossier résultant contient différents fichiers à réutiliser ou à compléter. A l'issue du temps imparti, archivez votre dossier et déposez l'archive sur Moodle.

Les exercices portent sur le jeu Rush Hour. En début de partie, des voitures de différentes couleurs, dont une voiture rouge, sont positionnées sur une grille $n \times n$ (voir Figure 1). Chaque voiture est orientée horizontalement ou verticalement et recouvre une ou plusieurs cases de la grille. Le joueur peut déplacer n'importe quelle voiture en ligne ou en colonne, selon son orientation, tant qu'aucune autre voiture ne la bloque et qu'elle reste dans la grille. L'objectif est de déplacer la voiture rouge jusqu'au bord droit de la grille. La taille de la grille, le nombre de voitures, leurs caractéristiques et leurs positions de départ sont configurables et constituent un "défi", le jeu en proposant plusieurs de difficulté variable.

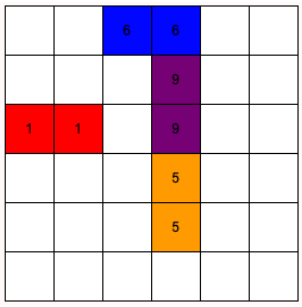

FIGURE 1 – Défi

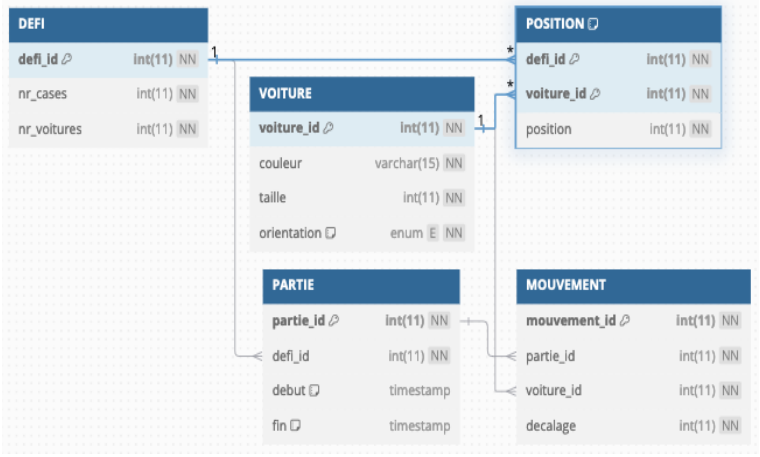

FIGURE 2 – Schéma relationnel

Une modélisation relationnelle du jeu est donnée en Figure 2.

- La table `VOITURE` modélise une voiture par sa couleur, sa taille (nombre de cases couvertes) et son orientation (lettre `H` pour "horizontale" ou `V` pour "verticale").
- La table `DEFI` modélise un défi par une taille de ligne n et un nombre de voitures.
- La table `POSITION` identifie les voitures d'un défi et modélise leurs positions de départ. Par convention, on assimile la position d'une voiture à la case la plus en haut à gauche qu'elle occupe sur la grille. Cette **case de référence** "bouge" avec la voiture et sa position initiale est donnée en colonne `position` par la formule $n \times (x-1) + y$ où $x \geq 1$ et $y \geq 1$ sont ses numéros de ligne et de colonne. Par exemple, la case de référence de la voiture violette d'id 9 a $(2, 4)$ comme coordonnées initiales, soit $10 = 6 * (2-1) + 4$.
- La table `PARTIE` modélise une partie par le défi que le joueur relève, une date de démarrage et éventuellement une date de fin.
- La table `MOUVEMENT` modélise, pour une partie donnée, tout déplacement de voiture par un décalage "unitaire" de sa case de référence (colonne `decalage`) : un décalage de -1 signifie que la voiture est déplacée d'une case vers le haut ou la gauche (selon l'orientation de la voiture), un décalage de 1 signifie qu'elle est déplacée d'une case vers le bas ou la droite. On suppose ici que l'ordre chronologique des décalages effectués lors d'une partie correspond à l'ordre sur les identifiants des enregistrements de `MOUVEMENT` (colonne `mouvement_id`).

Les exercices sont indépendants et peuvent être traités dans n'importe quel ordre.

Exercice 1. [Modèle objet]

En vous inspirant du modèle relationnel, implémentez dans le fichier **objet.php** un modèle objet pour représenter voitures et défis. En guise de test, le script doit construire le défi illustré en Figure 1 et l'afficher de manière sommaire sous forme de chaînes de caractères (pas de génération `HTML`). Vous ne modéliserez pas les concepts de partie ou de mouvement.

Exercice 2. [CSV]

Le fichier **partie.csv** contient l'historique des mouvements de voitures effectués lors d'une partie. Chaque ligne correspond à un mouvement : le premier champ est l'identifiant de la voiture et le second le décalage effectué. Les champs sont séparés par des virgules et les lignes sont triées dans l'ordre chronologique des mouvements. Implémentez la fonction documentée dans le fichier **partie.php** pour convertir le contenu du fichier `CSV` en un tableau associatif selon le format prescrit. Vous veillerez à verrouiller l'accès au fichier en cours de lecture.

Exercice 3. [Sessions, traitement, BDD]

On propose ici de développer un mini-site web pour le jeu. Le joueur calibre d'abord son défi en choisissant une taille de ligne et un nombre de voitures. Le site extrait alors un défi de sa base de données s'il en existe et le cas échéant affiche la grille de départ. Le joueur peut ensuite déplacer chaque voiture en cliquant sur une des cases qu'elle occupe : un simple clic décale la voiture de $+1$, un clic avec appui sur `SHIFT` la décale de -1. Tout déplacement invalide est rejeté et un avertissement s'affiche en console. Si le joueur a gagné, une alerte s'affiche auquel cas il peut relever un nouveau défi.

Le site est implémenté en `HTML`, `CSS`, `JS`, et `PHP` et s'appuie sur la base de données présentée en Figure 2. Le fichier `JS` gère l'interaction avec le joueur et communique avec le serveur sans que la page soit jamais regénérée. Au chargement du fichier `HTML`, il invoque **voitures.php** pour récupérer toutes les voitures de la base qui seront gardées en mémoire. Lorsque l'utilisateur clique sur le bouton "jouer une partie", il communique à **defi.php** la taille de ligne et le nombre de voitures renseignées dans le formulaire. **defi.php** essaie d'extraire un défi qui correspond, et en cas de succès, crée une partie et renvoie la grille de départ. Complétez-le en répondant aux questions qui suivent et en vous aidant des instructions dans le code source. Au préalable, importez via `phpMyAdmin` le fichier `SQL` qui créera la base de données et ajustez vos identifiants dans **connexpdo.inc.php**.

1. Récupérer les champs du formulaire soumis en `HTTP GET`.

L'extraction des défis qui correspondent aux champs est déjà implémentée.

2. Si aucun défi n'a pu être extrait, renvoyer le tableau `["statut"=>"KO", "defi"=>[], "positions"=>[], "debug"=>"..."]` formatté en objet `JSON`. Attention : tous les éléments sont requis avec les bonnes clés. Vous pouvez placer dans l'élément de clé `debug` les données que vous voulez (variables, etc.) : elles seront affichées en console pour vous aider à déboguer.

Si plusieurs défis correspondent, l'un d'eux est choisi aléatoirement, les cases de référence des voitures sont extraites, et une nouvelle partie est insérée en base.

3. Renvoyer comme réponse un tableau `PHP` formatté en objet `JSON` et contenant le statut `"OK"`, le défi, les coordonnées `(x,y)` des cases de référence des voitures, et le champ de débogage.

4. Sauvegarder dans les données de session l'id de la partie, l'id du défi, la taille de ligne, le nombre de voitures et les coordonnées de leurs cases de référence.

Lorsque l'utilisateur tente de déplacer une voiture, le fichier `JS` soumet à **mouvement.php** l'id de la voiture, le décalage demandé (`1` ou `-1`), et le détail des voitures du défi. **mouvement.php** doit vérifier si le mouvement est valide et, le cas échéant, actualiser la position de la voiture et indiquer si l'objectif est atteint. Complétez-le en répondant aux questions qui suivent et en vous aidant des instructions du code source.

5. Récupérer les données soumises en `HTTP POST`.

6. Extraire des données de session l'id de la partie, l'id du défi, la taille de ligne et le nombre de voitures.

7. A partir des coordonnées des cases de référence stockées dans la session, créer un tableau contenant pour chaque voiture les coordonnées de **toutes** les cases qu'elle occupe en fonction de son orientation et sa taille.

8. Tester alors la validité du mouvement et, le cas échéant, actualiser la case de référence de la voiture dans les données de session.

9. Sinon, renvoyer le tableau `["statut"=>"KO","debug"=>"..."]` formatté en objet `JSON`.

Si le mouvement est valide, il est inséré en base de données.

10. Si le mouvement est valide, renvoyer le même type d'objet `JSON` qu'en question 3. S'il conclut la partie, fixer la propriété `statut` de la réponse à la valeur `"VICTOIRE"` et mettre à jour la date de fin de la partie en base de données. S'il ne conclut pas la partie, fixer simplement la propriété à la valeur `"OK"`.

Exercice 4. [XML]

Le fichier **l3_cc_24_php_rush_hour.xml** stocke des données de voitures, défis, positions, parties et mouvements. Sa structure, ses balises et attributs s'inspirent du modèle relationel présenté en Figure 2. L'objectif est de parser ce fichier pour en extraire différentes informations. Complétez le fichier **exo_xml.php** pour répondre aux questions suivantes.

1. Implémentez la fonction `lire_xml_table` qui lit la "table" demandée et renvoie son contenu sous forme de tableau associatif.

2. Implémentez la fonction `s1_temps_moyen_parties` qui calcule le temps moyen passé par partie.

3. Implémentez la fonction `s2_mouvements_voiture` qui calcule les nombres minimum et maximum de déplacements de la voiture rouge effectués par partie.

4. Implémentez la fonction `stats_jeu` qui récapitule les informations extraites dans les questions précédentes en appelant les fonctions correspondantes.