Bataille Navale

1.Le projet

Contexte

Le projet consiste en la reprise et l'amélioration d'un jeu en ligne. Le jeu est actuellement hébergé sur un site web et permet aux utilisateurs d'affronter l'ordinateur dans des parties de bataille navale.

La partie graphique du jeu a été confiée à une autre personne, ce qui permet de vous concentrer sur les aspects fonctionnels et techniques de l'application. En tant que développeur, votre mission sera de prendre en charge la gestion de l'application.

Le jeu a été développé en utilisant PHP objet. Vous devrez donc avoir une bonne maîtrise de ce langage et de ses concepts de programmation orientée objet. Une connaissance des bases de données sera certainement nécessaire pour gérer les données des utilisateurs et des parties.

Plan de Travail proposé

1. Analyse du code existant : commencer par une revue approfondie du code existant pour comprendre son architecture et identifier les points à améliorer (se servir notamment de la partie 3-Organisation générale de l'application en page 4 pour gagner du temps).

2. Définition des besoins : étudier la partie 4-Modifications demandées en page 9 afin d'identifier clairement les besoins en termes de nouvelles fonctionnalités et de statistiques.

3. Développement et tests : Implémenter les nouvelles fonctionnalités et effectuer des tests rigoureux pour s'assurer de leur bon fonctionnement.

4. Déploiement : mettre en production les améliorations et surveiller les performances pour effectuer des ajustements si nécessaire.

Ce projet offre une opportunité de travailler sur un jeu en ligne, en mettant à profit vos compétences en développement PHP et en gestion de projet. Il vous permettra de mettre à profit les compétences de développement acquises lors du premier et du second semestre.

2.Règles du jeu de bataille navale

Cette partie rappelle les règles générales de la bataille navale. Il s'agit d'un jeu de stratégie où deux joueurs placent des navires sur une grille et tentent de couler les navires de l'adversaire en devinant leurs positions. Dans l'application, le second joueur est simulé par l'ordinateur.

Objectif

Le but du jeu est de couler tous les navires de l'adversaire avant qu'il ne coule les vôtres.

Matériel

- Deux grilles de jeu par joueur : une pour placer ses propres navires et une autre pour marquer les tirs sur les navires de l'adversaire.
- Une liste de navires à placer.

Préparation

1. **Grille de jeu :** Chaque joueur dispose de deux grilles dont la taille peut être variable. Dans la maquette de l'application proposée, les grilles sont de 20x20 cases.
2. **Placement des navires :** Chaque joueur place ses navires sur sa grille. Les types de navires et leur taille peuvent varier. Dans la maquette, la configuration standard est proposée :
    - 1 Porte-avions (5 cases)
    - 1 Croiseur (4 cases)
    - 1 Contre-torpilleur (3 cases)
    - 1 Sous-marin (3 cases)
    - 1 Torpilleur (2 cases)

Une case occupée par un navire du joueur est symbolisée par  .

Une case occupée par un navire du joueur est symbolisée par  .

Règles de placement

- Les navires peuvent être placés horizontalement ou verticalement, mais pas en diagonale.
- Les navires ne peuvent pas se chevaucher.
- Il est permis de placer les navires bord à bord.

Déroulement du jeu

1. **Tour de jeu :** Les joueurs jouent à tour de rôle. À chaque tour, un joueur définit une coordonnée (par exemple, B-3) pour tirer sur la grille de l'adversaire. Dans la maquette, le joueur sélectionne à l'aide de la souris une case de la grille de droite
2. **Résultat du tir :**
   - **Touché** : Si la coordonnée correspond à une partie d'un navire, le joueur annonce "Touché". Dans la maquette, la case devient  .
   - **Coulé :** Si toutes les parties d'un navire ont été touchées, le joueur annonce "Coulé". Dans la maquette, toutes les cases du navire deviennent  .
   - **À l'eau :** Si la coordonnée ne correspond à aucun navire, le joueur annonce "À l'eau". Dans la maquette, la case devient  .
3. **Marquage des tirs :** Chaque joueur marque ses tirs sur sa grille de tir et les impacts reçus sur sa grille de placement. Dans la maquette, les tirs sont affichés automatiquement.

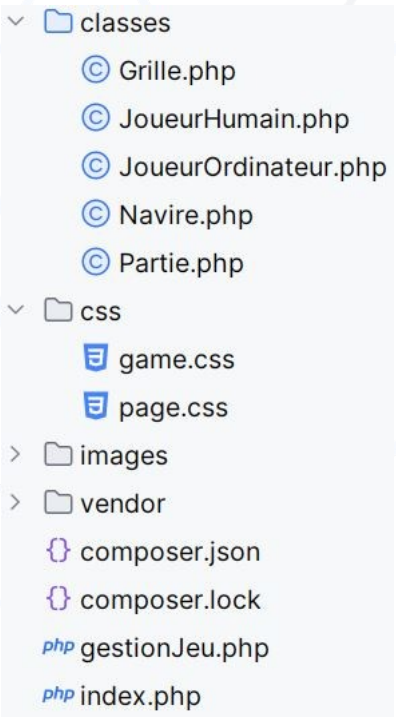
Fin de la partie

La partie se termine lorsqu'un joueur a coulé tous les navires de l'adversaire. Le joueur ayant coulé tous les navires adverses est déclaré vainqueur.

Dans le cas où le joueur humain capitule, la partie est déclarée comme perdue pour ce joueur.

3.Organisation générale de l'application

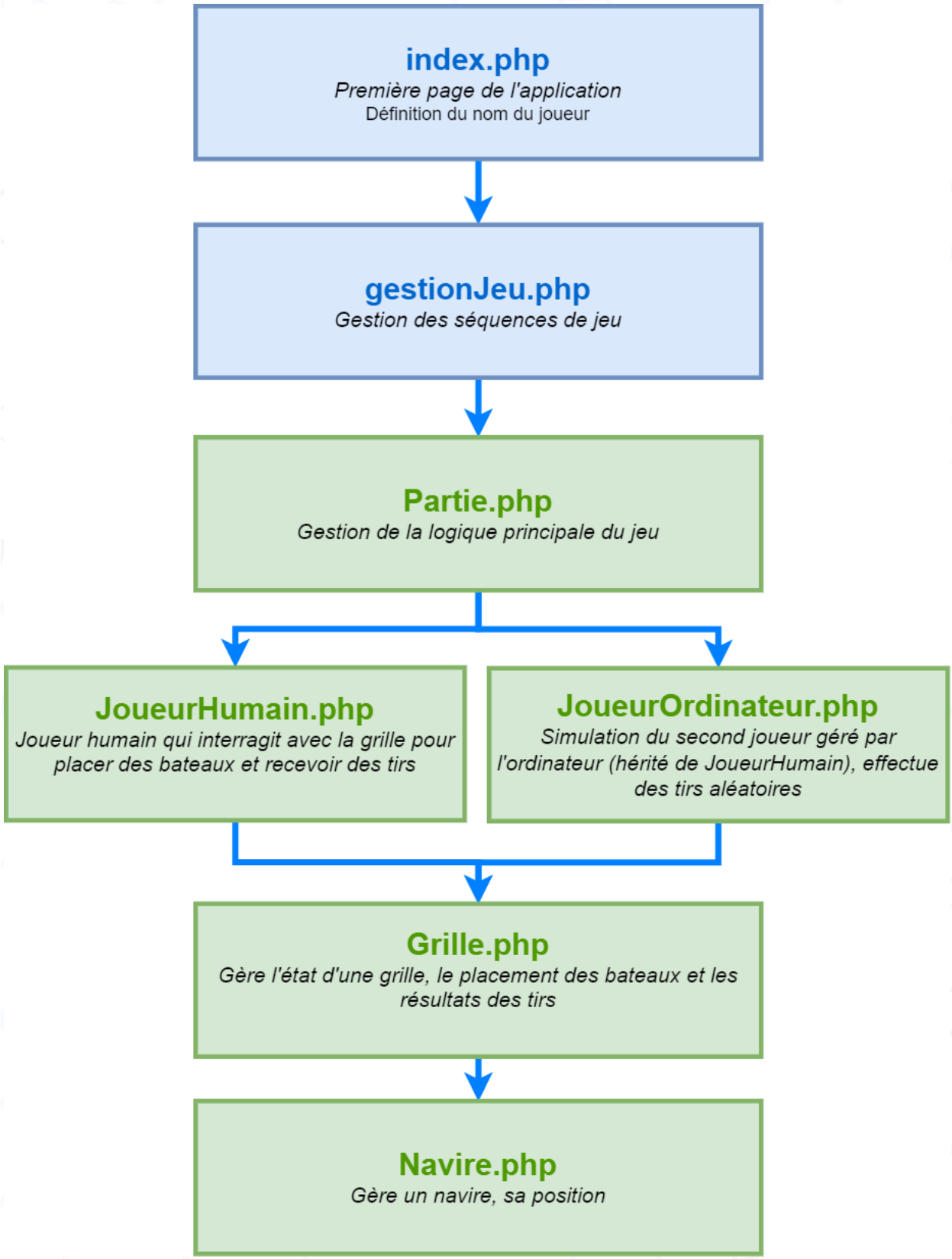
Arborescence du projet



Répertoires :

- classes : contient le code objet du projet.
- css : contient les feuilles de style du projet
  - page.css : est utilisé pour la page de connexions.
  - game.css : est utilisé pour la mise en page du jeu (interface, grilles, navires, informations affichées…).
- images : contient les images utilisées pour la mise en forme des pages et du jeu lui-même
- vendor : contient les éléments d'autoloading (non activé pour l'instant) et les dépendances du projet (aucune pour l'instant). Associé au fichier composer.json qui contient un exemple de déclaration du projet.
- A la racine :
  - index.php : page de connexion au jeu.
  - gestionJeu.php : interface principale dans laquelle se déroule le jeu.

Organisation fonctionnelle du jeu



Description des classes et fichiers :

index.php

Il s'agit de la première page de l'application, elle constitue le point d'entrée du joueur dans le jeu.

Elle contient un formulaire permettant au joueur de saisir son nom, et de commencer la partie.

gestionJeu.php

C'est la page contenant l'interface de jeu. Elle permet de gérer les séquences suivantes :

- **Gestion de la session utilisateur**. Une session nommée '
- ' permet de stocker un objet de type Partie (voir Partie.php page 7). Lors du premier tour de jeu, la session est créée. Lors des tours suivants, l'objet est récupéré. Dans tous les cas, l'objet Partie est accessible via la variable `$partie`.
- **Gestion des grilles de jeu**. Deux grilles sont créées pour le joueur humain (`$playerGrid`) et le joueur ordinateur (`$computerGrid`) au sein de l'objet (`$partie`)
- **Affichage des grilles de jeu.** Les grilles sont affichées dans l'interface graphique.
  - La grille de gauche correspond à la grille Sidu joueur, ses bateaux sont visibles. Les cases ne sont pas sélectionnables sur cette grille.
  - La grille de droite correspond à la grille de l'ordinateur, ses bateaux sont invisibles lors du jeu normal (afin de faciliter les phases de tests et de mise au point de l'application, les bateaux sont actuellement visibles, ce comportement peut être modifié dans la feuille de style `game.css`, via le style `navireOrdinateur`).
  - Affichage de la grille est réalisé par un parcours du tableau à deux dimensions représentant la grille de jeu, chaque case du tableau représente une case de la grille de jeu. Chaque case de la grille de jeu est un bouton sélectionnable ou non par l'utilisateur suivant la phase de jeu (tour du joueur ou de l'ordinateur). En fonction du contenu de la case du tableau, la feuille de style correspondante à la case et affichée.
- **Sélection des coordonnées de tir.** Si c'est au tour du joueur humain de jouer, les cases de la grille de droite sont rendues disponibles. Au contraire, si c'est autour de l'ordinateur de jouer, les cases de la grille de droite sont désactivées. Lors d'un clic sur une case, les coordonnées de la case sélectionnée sont envoyées comme variable `$_POST['coord']` contenant les coordonnées sous la forme `x,y`.
- **Traitement des coordonnées de tir.** Si des coordonnées ont été définies lors du tour précédent, elles sont récupérées via la variable `$_POST['coord']`, sont séparés en deux coordonnées distinctes `$x` et `$y`, puis elles sont traitées par l'objet `$partie` via la méthode `joueurJoue`.
- **Gestion de la fin de partie.** La détection de la fin de partie a lieu lors de l'utilisation de la méthode `joueurJoue`. De plus, un bouton nommé `'end_game'` est utilisable facultativement par le joueur pour mettre fin à la partie.

Partie.php

La classe contient deux conteneurs, un pour le joueur humain et l'autre pour le joueur ordinateur. Le constructeur initialise ces deux conteneurs et fait toujours commencer le joueur humain en premier.

La méthode `joueurJoue` est toujours appelée, elle permet de traiter le tir d'un joueur humain ou ordinateur. La méthode `tirerSurLaCase` est appelée et une vérification est effectuée pour savoir si tous les navires de l'autre joueur ont été coulés. Éventuellement, la fin de la partie est traitée via la méthode `finDePartie`.

Si c'est au tour du joueur humain de jouer, seule la méthode `joueurJoue` est exécutée. Par contre, si c'est au tour de l'ordinateur de jouer, la méthode `ordinateurJoue` est appelée, avec les mêmes vérifications.

JoueurHumain.php

La classe contient le nom du joueur ainsi que sa grille de jeu.

Le constructeur initialise la grille et place les navires de manière aléatoire.

La méthode la plus importante est `tirerSurLaCase` qui permet de mettre à jour l'état de la case de la grille du joueur dont les coordonnées sont passées en paramètres.

Une autre méthode utile est `getGrille` qui renvoie l'objet grille associé au joueur.

JoueurOrdinateur.php

La classe est héritée de la classe joueurHumain, elle possède donc les attributs et les méthodes de la classe supérieure.

Le constructeur est cependant surchargé, le nom du joueur ordinateur n'est pas utilisé, la grille est initialisée avec un paramètre permettant d'utiliser une feuille de style différente pour les navires (notamment de les cacher pendant la phase de jeu). Les navires sont placés aléatoirement sur la grille.

Grille.php

Le constructeur initialise une grille pour un joueur humain ou un joueur ordinateur (dans le cas d'un joueur ordinateur le paramètre `$estGrilleEnnemi` est défini à true lors de l'appel au constructeur).

La méthode `placeNaviresAleatoirement` permet de placer les navires sur la grille, en tenant compte des obstacles déjà placés (autres navires). Les navires les plus grands sont placés en premier, les navires ne peuvent être placés que verticalement ou horizontalement).

La méthode `peutPlacerNavire` permet de savoir si un navire peut-être placé à des coordonnées x et y en en tenant compte de l'orientation du navire.

La méthode `majGrilleAvecNavire` est utilisée pour mettre à jour le style CSS des cases de la grille contenant les navires.

La méthode `majEtatCaseApresTir` utilisée pour mettre à jour le style CSS d'une case dont les coordonnées sont passées en paramètres, après un tir. L'état de la case est mis à jour dans l'un des états **coulés**, **touché**, **raté**.

La méthode `toutNaviresCoules` est utilisée pour détecter si l'ensemble des navires de la grille ont été coulés, ce qui signifie que le joueur auquel la grille est attachée a perdu.

La méthode `getGrille` est un accesseur qui retourne la grille du jeu.

NAVIRE.PHP

Un navire comporte une taille, une position ainsi que le nombre de fois où le navire a été touché.

Le constructeur permet d'initialiser la taille du navire, sans position définie et sans tir touché au démarrage.

La propriété `$position` est un tableau contenant la liste des coordonnées des cases dans lequel le navire se trouve.

La méthode `placeNavire` permet de placer le navire à certaines coordonnées et orientations. Aucune vérification de placement n'est effectuée par cette méthode, par contre elle remplit la liste des coordonnées du tableau `$position`.

La méthode `getPosition` renvoie la liste des positions du navire sous la forme d'un tableau.

La méthode `estTouche` détermine si des coordonnées passées en paramètres permettent de toucher le navire.

La méthode `estCoule` détermine si le navire écoulé ou pas. Il compte pour cela le nombre de tirs touchés et tient compte de sa taille.

4.Modifications demandées

- Les fonctionnalités principales sont notées ***[P000]***
  - Ces fonctionnalités doivent être **obligatoirement** réalisées.
- Les fonctionnalités secondaires sont notées ***[S000]***
  - Ces fonctionnalités peuvent être réalisées de manière **facultative**, en fonction de l'avancée de votre travail.

Les modifications sont à effectuer dans l'ordre de votre choix. Par contre, les fonctionnalités principales doivent être réalisées AVANT les fonctionnalités facultatives.

Jeu

- ***[P010]*** Améliorer la fin de la partie
  - ***[P011]*** En proposant un résumé de partie contenant les statistiques de la partie (pourcentage de tirs réussis, temps passé)
  - ***[P012]*** En proposant une fin moins minimaliste qu'un simple message et un retour à la page principale du jeu (méthode `finDePartie` de la classe Partie)
- ***[P020]*** Améliorer l'interface utilisateur
  - ***[P021]*** Afficher des statistiques en temps réel pendant la partie, comme le nombre de bateaux restants, le nombre de tirs réussis…
  - ***[P022]*** Afficher le nom des navires et afficher les informations d'état dans l'interface (points de vie des navires et santé des joueurs…)
  - ***[S023]*** Générer des obstacles sur la carte. Attention, cette fonctionnalité est moins simple qu'elle n'y paraît (il faut tenir compte de ces obstacles lors du placement des navires)
  - ***[P024]*** Le joueur humain peut cliquer sur des cases déjà jouées, interdire ce comportement (sauf si la fonctionnalité facultative de déplacement des navires est implémentée ***[S080]***)
- ***[S030]*** Envoyer des notifications ou des alertes lorsque des jalons statistiques sont atteints (par exemple meilleur score personnel).
- ***[S040]*** Gérer des points de récompense
  - ***[S041]*** Attribuer des points de récompense basés sur les performances statistiques, comme un certain nombre de victoires, un taux de réussite à atteindre ou une série de victoires.
  - ***[S042]*** Permettre d'échanger ces points de récompenses contre des améliorations cosmétiques ou des avantage en jeu .
- ***[S050]*** Améliorer le rendu du jeu
  - ***[S051]*** Rendre aléatoire le design des cases (afficher par exemple des épaves, trésors, monstres marins…)
  - ***[S052]*** Adapter l'environnement de jeu comme la météo, la localisation de la carte (mer tropicale, plein océan, environnement arctique).
  - ***[S053]*** Introduire des événements aléatoires basés sur les statistiques, ajoutant une couche d'imprévisibilité.
  - ***[S054]*** Créer un mode histoire, des narrations évolutives basées sur les performances et les choix des joueurs, rendant chaque partie unique.
  - ***[S055]*** Fournir des retours visuels clairs pour les actions du joueur (tirs, coulés, etc.).

Joueur humain ou ordinateur

- [P060] Afficher le rang du joueur (matelot, capitaine, amiral…)
- [P070] Permettre à un joueur de proposer une partie nulle, dans ce cas, aucun des joueurs ne perd de points.
- [S080] Permettre le déplacement des navires
- [P090] Permettre l'utilisation d'un missile de croisière une fois par partie (9 cases)

Joueur humain

- [P100] Création d'un profil utilisateur personnel
  - [P101] Identifier le joueur de manière unique dans le jeu
  - [P102] Stocker les statistiques de progression et de carrière du joueur dans le profil
- [P110] Implémenter un système de niveaux où les joueurs gagnent de l'expérience en jouant des parties. Les statistiques peuvent montrer la progression vers le prochain niveau.
- [S120] Permettre au joueur de personnaliser son expérience (choix des couleurs, avatars...).
- [S130] Attribuer des badges pour des réalisations spécifiques, comme un certain nombre de victoires consécutives ou un taux de réussite élevé. Afficher une liste des réalisations débloquées et celles restantes à atteindre.
- [S140] Permettre aux joueurs de débloquer ou d'améliorer des équipements pour leurs bateaux en fonction de leurs statistiques ou implémenter un système d'arbres de compétences où les joueurs peuvent débloquer des capacités basées sur leurs statistiques de jeu.

Joueur Ordinateur

- [P150] Générer et afficher un nom aléatoire pour le joueur ordinateur
- [P160] Améliorer la manière de joueur de l'ordinateur
  - [P161] Effectuer des tirs proches d'impact lorsque un navire a été touché
  - [P162] Le joueur ordinateur peut tirer sur des cases déjà jouées, interdire ce comportement (sauf si la fonctionnalité facultative de déplacement des navires est implémentée [S080])
- [S170] Utiliser plusieurs modèles de jeu pour le joueur (nul, facile, moyen, difficile…)
- [S180] Fournir des statistiques sur les performances du joueur ordinateur pour aider à élaborer des stratégies.
- [S190] Utiliser les statistiques pour prédire les mouvements ou stratégies probables de l'adversaire.

Statistiques générales

- [P200] Établir un classement des joueurs basés sur diverses statistiques (nombre de victoires/défaites/nul, plus grand nombre de parties jouées, plus grand nombre de parties gagnées à la suite, pourcentage de précision des tirs…)
- [P210] Afficher des statistiques globales pour l'ensemble de la communauté, comme le nombre total de parties jouées ou le taux moyen de réussite.
- [S220]Permettre aux joueurs de partager leurs statistiques sur les réseaux sociaux (utilisation d'une API nécessaire)
- [S230]Fournir aux joueurs un tableau de bord avec des statistiques détaillées sur leurs performances et leurs progrès.
  - [S231]Utiliser des graphiques pour visualiser les performances au fil du temps, comme les victoires/défaites, les séries de victoires...
  - [S232]Permettre aux joueurs de comparer leurs statistiques avec la moyenne globale des joueurs.