3ème année

APPLICATION WEB

LICENCE 3 MIASHS parcours MIAGE

RAPPORT DE CONCEPTION - PROJET DE JEUX WEB EN JAVASCRIPT

ZAKARIA LAACHIRI

ACHEHBOUNE Mohammed Amine

2024/2025

MAI 2025

Module : APPLICATION WEB

Enseignant : Monsieur Michel BUFFA

MIAGE RÉSEAU DES MIAGE DE FRANCE Nice

PRÉSENTATION GÉNÉRALE

Contexte du projet

Nous avons réalisé un projet de création de jeux web en JavaScript dans le cadre du module Application Web. Notre objectif était de concevoir deux jeux différents avec des technologies front-end, ainsi qu'un site web permettant de les centraliser. Zakaria Laachiri a développé le site Ludiverse et un jeu basé sur le Canvas API, tandis que Mohammed Amine Achehboune a conçu un mini-jeu en utilisant le DOM JavaScript.

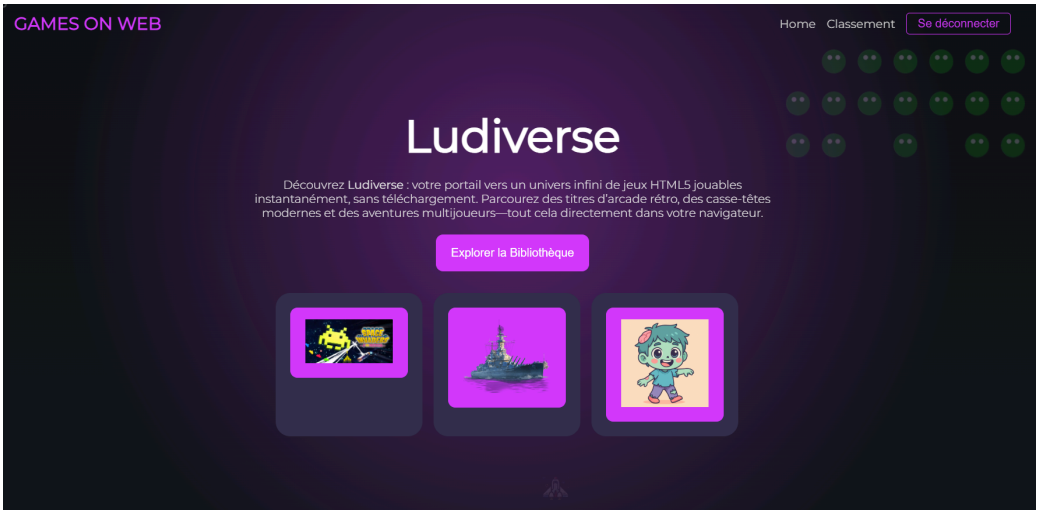
Objectifs

- Créer une plateforme web responsive (Ludiverse).
- Concevoir deux jeux originaux en JavaScript.
- Mettre en place un système de comptes utilisateurs avec connexion obligatoire.
- Permettre aux invités d'accéder à certains jeux simples.
- Restreindre l'accès au jeu Space Alien Invaders aux utilisateurs connectés (jeu plus stratégique).
- Gérer un classement des scores localement.
- Appliquer une structure modulaire pour le code de chaque jeu.

Répartition du travail

- Zakaria Laachiri : Conception du site Ludiverse, développement complet du jeu Canvas (Space Alien Invaders), gestion des comptes, scores et classement.
- Mohammed Amine Achehboune : Développement du jeu DOM (jeu de bateau), gestion des interactions DOM, logique de gameplay.

Site Web – Ludiverse



Description

Ludiverse est le nom de la plateforme web développée pour ce projet. Elle a été pensée comme une vitrine dynamique des jeux disponibles, avec une expérience utilisateur immersive.

Fonctionnalités principales

- Page d'accueil immersive avec une vidéo de fond.
- Présentation visuelle des jeux avec miniatures interactives et descriptions.
- Formulaire d'inscription et de connexion simulé via localStorage.
- Accès libre aux jeux simples pour les invités.
- Accès au jeu Space Invaders uniquement après connexion (jeu plus stratégique).
- Classement dynamique affichant les 10 meilleurs scores des utilisateurs.
- Navigation responsive avec menu burger pour les écrans mobiles.

Jeu 1 développé : Space Alien Invaders (Canvas)

Concept

Space Alien Invaders est un jeu de tir (shoot'em up) rétro inspiré de l'arcade. Le joueur pilote un vaisseau spatial et doit éliminer des vagues successives d'ennemis tout en esquivant leurs tirs. Le jeu progresse par niveaux avec une difficulté croissante (5 niveaux).

Mécaniques principales

- Déplacement du joueur avec les touches ← / → ou A / D.
- Tir avec la barre ESPACE.
- Pause avec la touche P ou bouton.

- Système de vies (perte de vie en cas de collision avec un tir ennemi).
- Vagues d'ennemis évolutives, selon le niveau atteint.
- Changement de fond, vitesse et complexité selon le niveau.
- Fin de partie via Game Over ou écran de Victoire selon les performances.
- Effets sonores dynamiques pour chaque phase (menu, gameplay, game over, muted).

Structure technique

Le jeu est entièrement développé avec JavaScript Vanilla et Canvas API, avec une architecture modulaire bien structurée.

Fichiers et modules

- Game.js : Moteur principal du jeu (initialisation, boucle de jeu, niveaux, gestion des états, sons, collisions, scoring).
- Player.js : Contrôle du joueur (déplacement, tir, collisions).
- Enemy.js : Gestion des ennemis (position, tir, déplacement).
- Bullet.js : Gère les projectiles (du joueur et des ennemis).
- enemyShoot.js : Module de tir automatique des ennemis.
- Score.js : Affichage et incrémentation du score du joueur.
- Lives.js : Système de points de vie, avec affichage.
- Son.js : Gestion des effets sonores.
- Levels/*.js : Cinq niveaux avec configuration des ennemis, vitesse, couleurs, complexité.
- collisions.js : Détection de collisions entre projectiles et sprites.
- ObjectGraphique.js : Classe de base pour les objets affichables.
- ecouteurs.js : Gestion des touches clavier.
- script.js : Initialisation et lancement du jeu depuis canvas.html.

Niveaux et progression

5 niveaux progressifs sont disponibles. Chaque niveau est défini dans un fichier distinct, avec un fond, une difficulté et un comportement différents.

Fin de partie

En cas de victoire ou de défaite, le score est enregistré dans localStorage et l'utilisateur est redirigé vers une page de fin.

Jeu 2 développé : Jeu bateau (DOM)

La logique du jeu se décompose en trois grandes modules :

1. **Phase de placement des bateaux**

- À l'aide de `placementScreen.js`, on affiche une grille vierge et une liste de bateaux à glisser–déposer.

- L'utilisateur peut changer l'axe de placement (horizontal/vertical), disposer manuellement chaque navire, ou tout placer aléatoirement via `playerBoard.randomize(…)`.

- Une fois tous les bateaux posés, le bouton « START GAME » devient actif et lance la partie.

2. **Modèles de données**

   - **`gameboard.js`** gère un plateau 10×10 :

     - `placeShip(coords, length, axis)` positionne un objet `{ beginningCoords, ship }` sur les cases autorisées,

     - `receiveAttack(coords)` marque un tir (touché ou manqué), met à jour l'état du `ship` (`hit()` / `isSunk()`), et renvoie `{ shipHit, adjacentCoords, beginningCoords }` pour gérer le contour d'un bateau coulé.

     - `randomize([longueurs])` réinitialise et répartit tous les bateaux aléatoirement.

   - **`ship.js`** implémente chaque navire :

     - propriété `length`, compteur `hits`, méthodes `hit()` et `isSunk()`.

   - **`player.js`** encapsule un joueur humain ou IA :

     - possède son `board` et un `attack(enemyBoard, [coords])` déléguant à `receiveAttack`. placementScreen

3. **Mécanique de jeu et IA**
   - Le **`gameController.js`** orchestre les tours :
     - `init()` crée deux joueurs (vous + IA), et place aléatoirement les bateaux de l'IA.
     - `playTurn(coords)` effectue un tir joueur, bascule le tour si le tir est manqué, et teste la victoire (`areAllShipsSunk()`).
     - `playComputerTurn()` fait appel à l'IA pour un tir sans coordonnées, via `computerPlayer.attack(...)`. gameboardship
   - L'IA dans **`computerPlayer.js`** combine :
     - **attaques aléatoires** tant qu'elle n'a pas touché un bateau,
     - **stratégie ciblée** une fois un contact obtenu : elle explore en spirale (haut, droite, bas, gauche), change de direction si elle bute ou rate, et réinitialise

quand le bateau est coulé. playergameController

4. **Affichage et score**

   - `gameScreen.js` se charge du rendu DOM :

     - crée deux grilles (`createBoardUI`) et affiche le statut de tour et le score (10 points par touche).
     - au clic sur la grille ennemie, appelle `game.playTurn`, puis colorie la case (`hit` ou `miss`) et, si un bateau est coulé, marque automatiquement toutes les cases adjacentes.
     - la même logique s'applique pour le tour de l'IA, avec un délai (`setTimeout`) pour simuler la réflexion.
     - lorsqu'un joueur gagne, on enregistre le score dans `localStorage` et on affiche une modal « You have CONQUERED ! ». computerPlayergameScreen

---

CONCLUSION

Points forts

- Projet réalisé intégralement à deux.
- Deux jeux originaux et fonctionnels avec deux technologies différentes.
- Site web fluide, responsive et centralisé.
- Bonne séparation du code par fichiers et modules.
- Respect des objectifs du module.

Difficultés rencontrées

La première difficulté qu'on a rencontrée était de choisir quel type de jeu développer. N'étant pas de grands amateurs de jeux vidéo, nous n'avions pas d'inspiration claire au départ. Cela nous a demandé du temps pour explorer différents concepts simples à mettre en œuvre en JavaScript. Après plusieurs essais et recherches, Zakaria a finalement choisi de créer un jeu de type Space Invaders, classique mais techniquement intéressant pour exploiter les capacités du Canvas API.

Améliorations futures

- Continuer le développement d'un jeu Babylon.js (zombie).
- Intégrer une base de données réelle pour gérer utilisateurs et scores (back-end au lieu de localStorage).
- Modulariser davantage les composants communs.