3ème année

APPLICATION WEB

LICENCE 3 MIASHS parcours MIAGE

RAPPORT DE CONCEPTION - PROJET DE JEUX WEB EN JAVASCRIPT

Yann **MONTENOT**

Adame **LOTFI**

Téo **VIGLIETTI**

2024/2025

MAI 2025

Module : APPLICATION WEB

Enseignant : Monsieur Michel BUFFA

MIAGE RÉSEAU DES MIAGE DE FRANCE Nice

PRÉSENTATION GÉNÉRALE

Contexte du projet

Dans le cadre de notre cours de conception de jeux web, nous avons réalisé un projet de groupe consistant à créer trois jeux vidéo en JavaScript, chacun utilisant une technologie différente : Canvas, DOM et Babylon.js.

Objectifs

- ✔ Concevoir trois jeux originaux et fonctionnels.
- ✔ Créer un site vitrine centralisé pour les lancer et afficher les scores.
- ✔ Favoriser à la fois l'autonomie et la cohésion d'équipe.

Répartition des tâches

- **Yann MONTENOT** : Jeu *Ratscooter* (Canvas) + site vitrine complet.
- **Téo VIGLIETTI** : Tower Defense (DOM).
- **Adame LOTFI** : Jeu Babylon.js (nom à préciser).

Technologies utilisées

- JavaScript (Canvas, DOM, Babylon.js)
- HTML5 / CSS3
- Stockage local (localStorage ou JSON)

Ratscooter (Canvas)

Objectif du jeu

Le joueur incarne un rat sur un scooter parcourant les échafaudages de la ville de Paris. L'objectif est d'aller le plus loin possible sans tomber.

Principales mécaniques

- Défilement horizontal automatique avec génération dynamique de plateformes
- Sauts et chutes réalistes selon la position sur la plateforme
- Système de boost (barre de boost, animation de flamme, son associé)
- Bonus aléatoires (croissants / baguettes) rapportant des points
- Flottement des scores lors de la collecte de bonus
- Jauge de boost dynamique (gestion de vitesse + feedback visuel)

- Gestion des transitions de chute (inclinaison progressive, game over)
- Menu de pause et d'accueil intégré

Architecture du code

- Game.js : logique du jeu principale (animation, collisions, scoring...)
- Player.js : gestion du joueur, inclut un véhicule animé et un corps de rat dessiné manuellement (Canvas 2D)
- Plateforme.js : gestion des plateformes (dessin, mouvement, bonus associés)
- ObjetSouris.js : effet visuel suivant la souris
- ecouteurs.js : centralisation des événements (clavier, souris)
- scores.js : persistance des meilleurs scores via localStorage

Difficultés

- Gérer les transitions de chute de manière fluide (rotation + mouvement latéral)

Points notables

- Dessins du rat faits manuellement via Canvas 2D
- Utilisation d'une animation .gif pour la flamme du boost – bibliothèque Gifler
- Système de score global intégré au site vitrine

Tower Defense - DOM

Objectif du jeu

Dans Tower Defense, le joueur incarne un stratège chargé de défendre sa base contre des vagues d'ennemis. L'objectif est de placer des tours défensives de manière stratégique pour empêcher les ennemis d'atteindre la fin du parcours.

Principales mécaniques

Le joueur peut acheter trois types de tours différents, chacunes ayant des caractéristiques propres telles que la portée, la cadence de tir et les dégâts. Les tours tirent automatiquement sur les ennemis à portée. L'or est gagné en éliminant les ennemis, ce qui permet d'acheter de nouvelles tours. Il est également possible de récupérer la moitié de l'or investi en vendant une tour.

Il existe trois types d'ennemis : un classique, un rapide et un tank. Ces ennemis arrivent par vagues successives (au total dix vagues) que le joueur est invité à déclencher. Les vagues deviennent

progressivement plus difficiles, introduisant des adversaires plus rapides, plus résistants et plus nombreux.

Un chronomètre intégré permet de mesurer le temps mis par le joueur pour éliminer les dix vagues. Il est possible de lancer les vagues sans attendre, ce qui permet de battre des records de temps, mais augmente également la difficulté du jeu.

Le meilleur temps est alors sauvegardé et peut être retrouvé sur la page principale sous forme de classement.

Architecture du code

Le projet se compose de plusieurs fichiers, chacun ayant un rôle spécifique : index.html est le point d'entrée du jeu, styles.css définit son apparence, et game.js gère la logique principale. Les fichiers map.js, tower.js, towerTypes.js, enemy.js et enemyTypes.js assurent la gestion de la carte, des tours et des ennemis. Les vagues sont configurées via waves.js et wavesInformation.js, tandis que bullet.js gère les projectiles. L'ambiance sonore est contrôlée par audio.js, l'interface utilisateur par ath.js, les menus par menu.js, et le chronomètre par chronometre.js.

Difficultés

Toute la difficulté du jeu réside dans le choix de lancer ou non les vagues rapidement : plus on les enchaîne vite, meilleur peut être le temps final, mais cela rend la victoire bien plus difficile. Il convient également de bien placer ses tours et de choisir les types adaptés à chaque situation.

Points notables

Le jeu propose un style graphique original, avec un visuel en crayon à main levée qui lui donne un aspect unique et fait main. Ce style, combiné à une ambiance sonore immersive avec musique de fond et effets sonores soignés, rend l'expérience de jeu plus vivante et agréable.

Jeu 3 - BabylonJS

Objectif du jeu

Dans *CrazyBowling*, le joueur incarne un lanceur de bowling dans une version revisitée et excentrique du bowling classique. L'objectif est de faire tomber le maximum de quilles possibles en deux lancers par manche, et ce, sur un total de dix manches. Le joueur peut ainsi accumuler des points et tenter de battre son meilleur score personnel.

Principales mécaniques

Le gameplay repose sur la physique réaliste fournie par BabylonJS et le plugin Havok Physics. Le joueur contrôle la boule de bowling via les touches du clavier (Q pour aller à gauche, D pour aller à droite, Z pour lancer la boule). Chaque manche permet deux tirs. Après chaque lancer, le nombre de quilles renversées est comptabilisé, et le score est mis à jour. Si toutes les quilles tombent dès le premier lancer, le joueur passe automatiquement à la manche suivante.

Le score total de la session est sauvegardé en local, et s'il dépasse le meilleur score enregistré pour l'utilisateur, ce nouveau score est mis à jour dans le classement personnel.

Architecture du code

- **index (Crazybowling.html)** : point d'entrée du jeu avec intégration des librairies BabylonJS et de la scène.
- **app.js** : initialise le jeu après authentification.
- **auth.js** : gère l'authentification et la navigation entre les pages.
- **scene.js** : crée la scène 3D, la caméra, les lumières, le sol, et lance l'ensemble du jeu.
- **ball.js** : gère la création, le déplacement et la physique de la boule.
- **pins.js** : gère les quilles, leur placement, leur chute, et la détection de leur état.
- **controls.js** : configure les contrôles clavier.
- **score.js** : assure la gestion du score par manche et global, avec sauvegarde locale.
- **ui.js** : affiche les boutons de relance ou nouvelle partie, ainsi que les informations à l'écran (score, manche).

Difficultés

La principale difficulté réside dans la gestion de la physique avec BabylonJS et notamment avec le plugin Havok, il faut bien positionner la caméra de sorte a que le jeu soit bien visible pour l'utilisateur?

Points notables

Le jeu tire sa force de sa réalisation en 3D temps réel avec BabylonJS et de l'intégration d'une physique réaliste pour les objets (boule de bowling, et les quilles). L'interaction fluide avec le clavier et les animations donnent une impression de fluidité et d'immersion. Le jeu est également pensé pour une utilisation connectée, avec authentification et sauvegarde des scores personnalisés, permettant à chaque joueur de suivre ses performances au fil du temps.

GamesHub (Site central)

Rôle

- ✔ Interface commune pour présenter et lancer les trois jeux
- ✔ Affichage dynamique des scores

- ✔ Responsive design (PC et mobile)

Fonctions clés

- ✔ Score géré en localStorage avec structure JSON
- ✔ Intégration redirection
- ✔ Images adaptées aux jeux

Capture d'écran à insérer (page d'accueil + page de jeu)

CONCLUSION

Bilan

- Projet riche en apprentissage individuel et collectif
- Diversité des technologies testées
- Site vitrine fonctionnel et fluide

Points forts

- Trois jeux originaux et réalisés de manière autonome
- Intégration réussie sur une plateforme commune + déploiement sur Vercel
- Respect des contraintes techniques

Améliorations possibles

- Modularisation des composants communs (scoreboard, styles CSS, …)

GitHub

Tous les fichiers (code source, rapport) sont disponibles sur notre dépôt GitHub : https://github.com/Yanou83/ProjetJeuxJS

Jouer en ligne :

https://projetjeuxjs.onrender.com/