Cahier des charges
Projet de fin d’etude
Flipper

Année 2025-2026

Kelly Gama Gonçalves - Arthur Guillemin - Ryan Annic - Emilie Caverne - Younes Benaggoun

Sommaire

1. Vision du projet

Présentation du projet

Ce projet consiste à concevoir et développer un flipper virtuel interactif, reproduisant fidèlement les mécaniques d'un flipper physique tout en exploitant les technologies web modernes.

L'objectif est de créer une expérience immersive mêlant : rendu graphique 3D, simulation physique réaliste, communication en temps réel, interaction via contrôleurs (clavier / IoT).

Description d'un flipper

Un flipper est une machine de jeu composée d'un plateau rempli d'obstacles, de cibles et de mécanismes. Le joueur doit empêcher la bille de tomber tout en maximisant son score grâce aux interactions avec les différents éléments du plateau.

Ce projet vise à transposer ces mécaniques dans un environnement numérique.

2. Objectifs et périmètre

Le projet vise à concevoir une table de flipper complète reposant sur trois applications distinctes, synchronisées en temps réel :

- **Playfield** : affichage du plateau de jeu en 3D
- **Backglass** : affichage du score et des informations de jeu
- **DMD** (Dot Matrix Display) : affichage des messages et feedback joueur

Fonctionnalités principales attendues

- Simulation physique réaliste de la bille (collisions, gravité, rebonds)
- Interaction avec les flippers
- Gestion du score et des événements de jeu
- Synchronisation temps réel entre les applications
- Intégration d'un système de contrôleurs (IoT)

Fonctionnalités bonus envisagées

- Mode deux joueurs
- Deuxième niveau
- Effets visuels avancés
- Effets sonores immersifs

- Éléments d'intelligence artificielle

Non-Objectifs :

- Pas de multijoueur en ligne distant
- Pas de version mobile

Personas :

- **Emilie :** étudiante en développement web, veut un projet technique ambitieux combinant 3D, temps réel et IoT.

- **Arthur:** étudiant en développement web, passionné de jeux d'arcade, recherche une expérience de flipper numérique fidèle aux sensations d'un vrai flipper physique.

- **Ryan :** étudiant en développement web, souhaite intégrer un ESP32 a un projet concret pour relier matériel physique et application web en temps réel.

3. Usecases :

Cas 1 :

Nom	**Lancer une Partie**
Acteur	Joueur
Objectif	démarrer une Nouvelle Partie
Préconditions	Machine Allumé, et déploiyé Machine libre (pas de joueur)
Déclencheur	Clique le bouton START
Scénario nominatif	Animation de BienVenue Score à Zéro La balle entre en la table (rampe de tirette) Nombre de vie défini
Postconditions	le jeu commence la balle bouge en table
Extensions/Exceptions	bouton ne marche pas erreur inconnu
Classes impliquées	Game, Physics Engine , Ball, Bar , ScoreManager

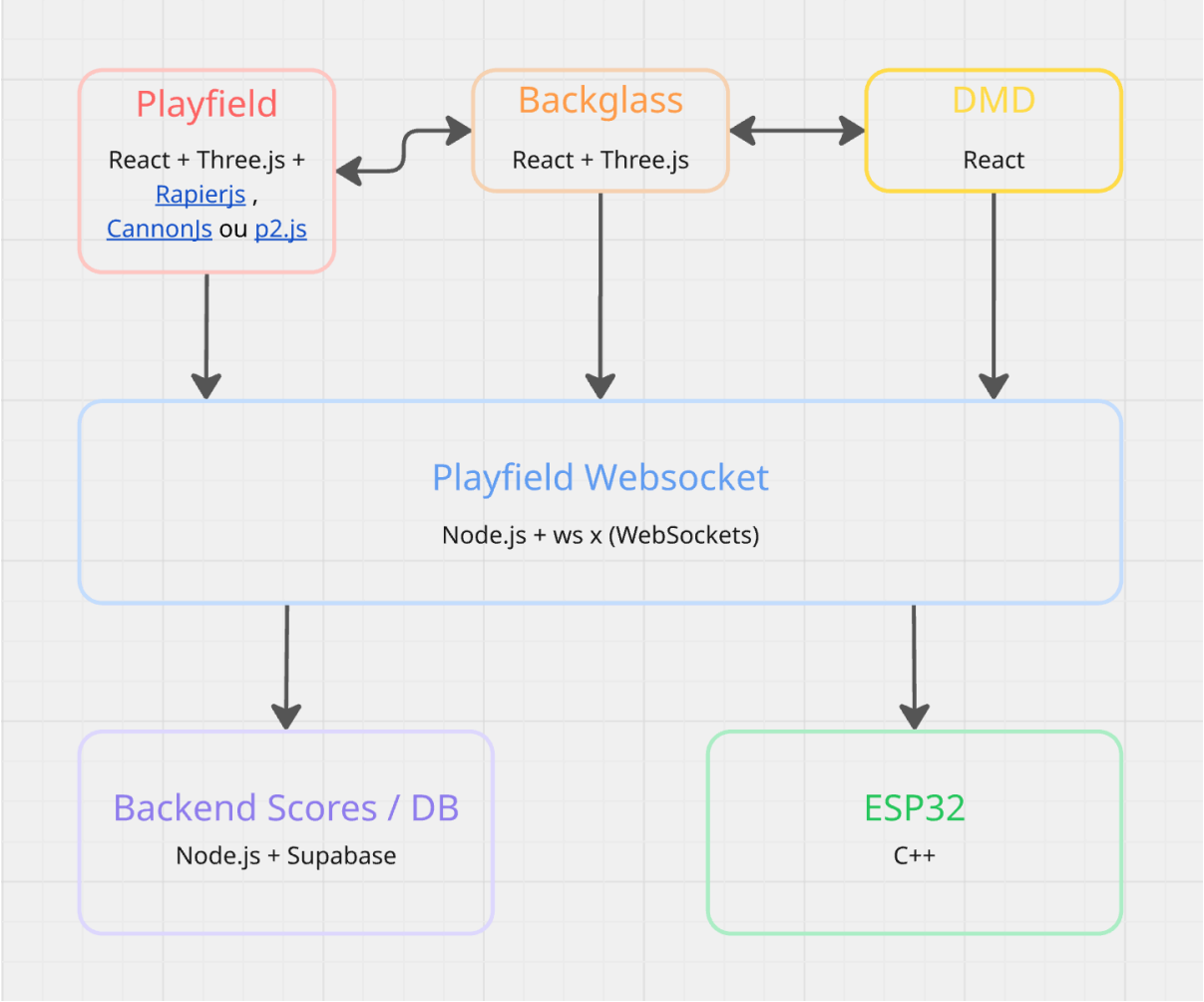
Cas 2 :

Nom	**Interaction balle / bumper**
Acteur principal	Joueur (via interaction physique)
Objectif	Gagner des points lors d'une collision
Préconditions	Une partie est en cours et la balle est active
Déclencheur	La balle entre en collision avec un bumper
Scénario nominatif	1. La balle se déplace sous l'effet de la gravité. 2. Le moteur physique détecte une collision. 3. Collision entre Ball et Bumper 4. Le système applique une force de rebond. 5. Une animation et un son sont déclenchés. 6. Des points sont ajoutés au score. 7. Le score est mis à jour à l'écran
Postconditions	Le score est augmenté et la balle continue son mouvement
Extensions/Exceptions	- Multiplicateur actif -> bonus de points - Combo actif -> score majoré
Classes impliquées	Physics Engine , Ball, Bumper, ScoreManager, Game

Cas 3 :

Nom	**High Score/ Le Joueur écrit son surnom sur le hall of Fame**
Acteur	Joueur
Objectif	Réalisé un score parmis les 10 premiers
Préconditions	Machine Allumé, jeux démarré ,
Déclencheur	Avoir un Score parmi les 10 premiers
Scénario nominatif	Jeux démarré le joueur réalise un haut score grâce au jeux ou grâce à plusieurs bonus Le jeux est terminé (Game Over) Le Score affiché Animation de félicitation
Postconditions	Score parmis les 10 premiers Animation de félicitation
Extensions/Exceptions	le joueur fini la parti en 11ème place au plus joueur refuse de mettre son surnon
Classes impliquées	Game, Physics Engine , Ball, Bar , ScoreManager, Animation

4. Architecture technique



5. Diagrames UML

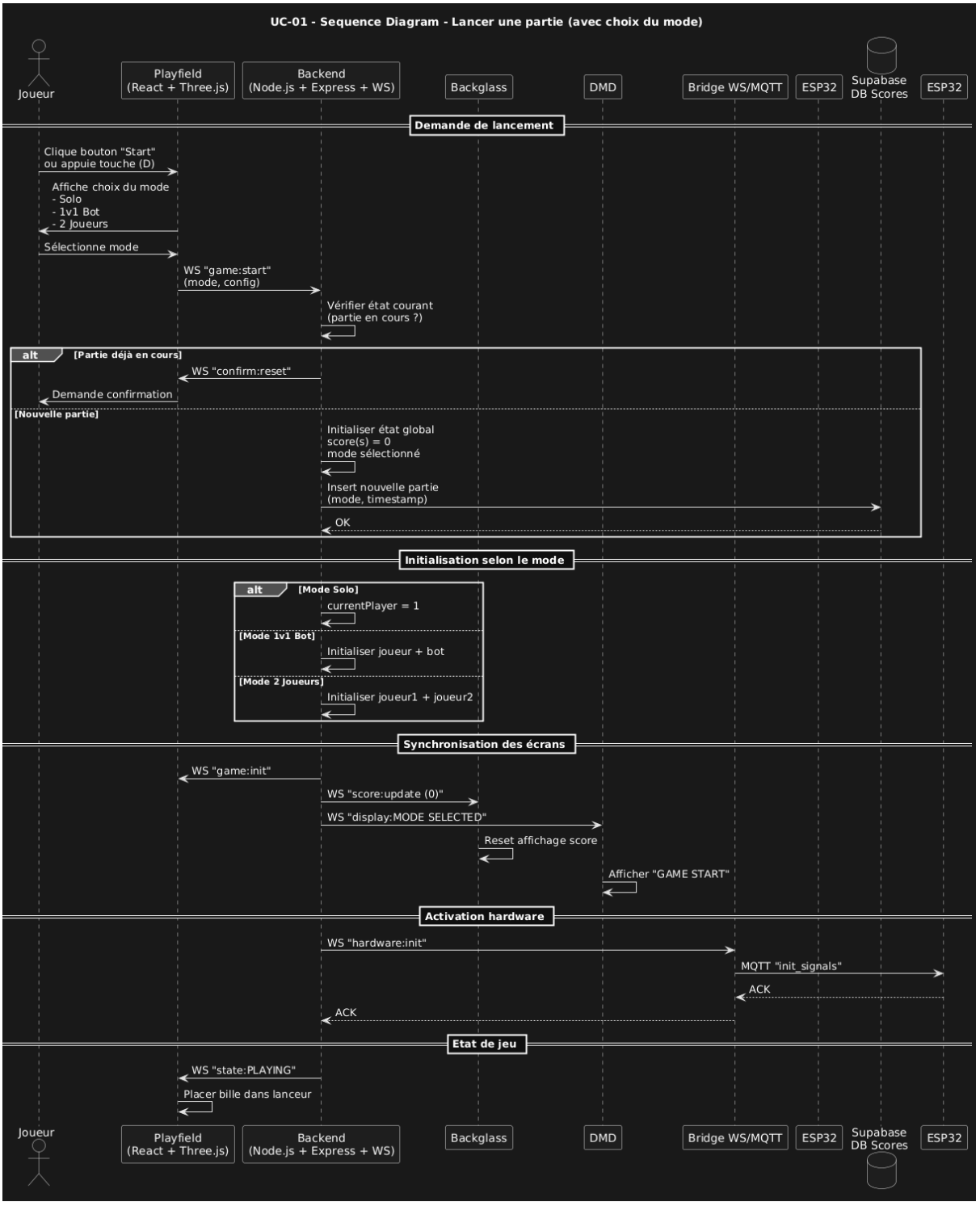
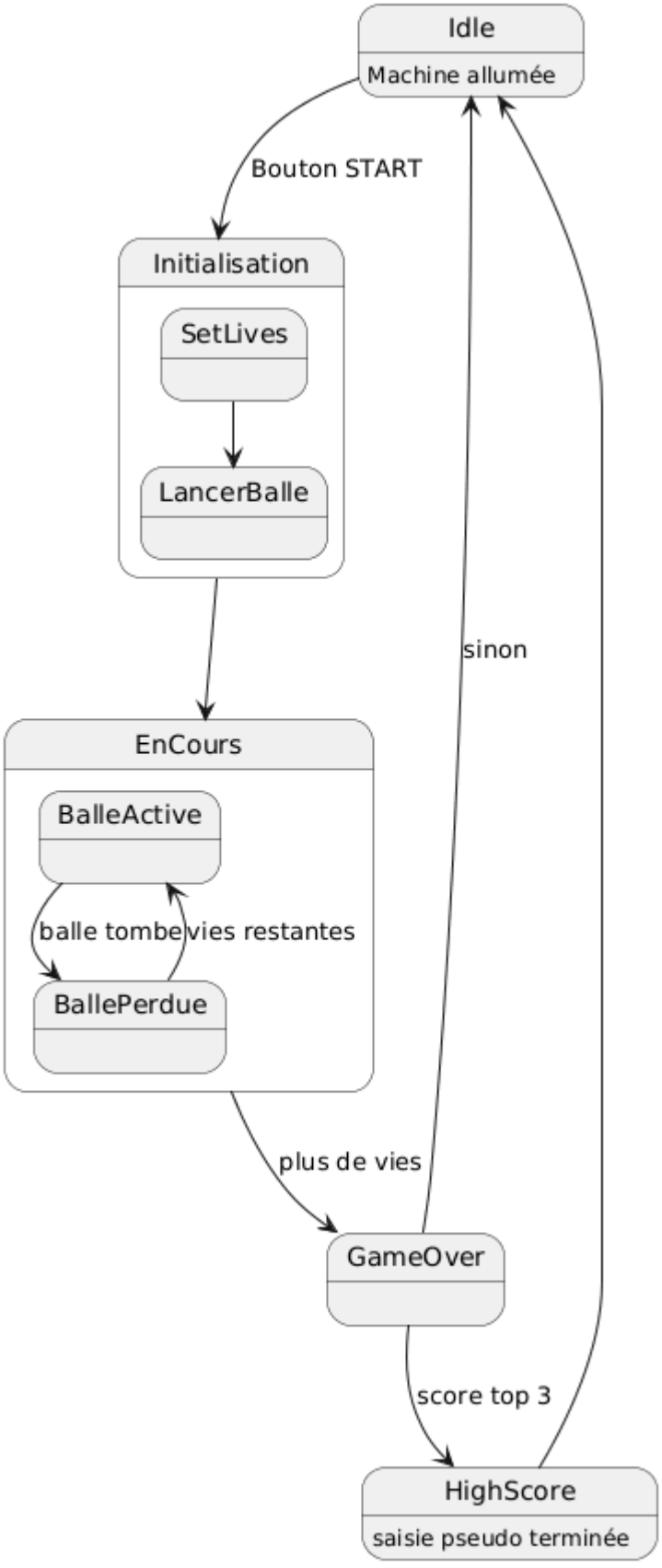


Diagramme d'état - Partie de Flipper



6. Stack technique

Composant	Techno	Justification
Playfield	React, threeJs, Rapierjs ou CannonJs ou p2.js, Blender	React assure une architecture modulaire. Three.js permet le rendu 3D web. Le moteur physique n'est pas encore définitivement choisi, on réfléchit à la solution la plus adaptée au projet. Blender est utilisé pour la création et l'exportation des modèles 3D.
Backglass	React, Three.js	Architecture cohérente avec le Playfield. Permet animations et éléments visuels dynamiques.
DMD	React	Adapté à l'affichage 2D temps réel (scores, états, animations). Léger et performant.
Bridge	Node js, websocket	Permet la communication temps réel entre les modules.
Backend Scores / DB	Node.js, Supabase	Supabase simplifie la gestion de la base de données et la persistance. Intégration rapide avec Node.js.
ESP32	C++	Permet le contrôle du matériel et la gestion des entrées physiques.

7. Risques et Contraintes techniques

Contraintes techniques

Le projet devra respecter les contraintes suivantes :

- Trois écrans physiques imposés
- Une communication en temps réel stable
- Une simulation physique crédible
- Une séparation claire des responsabilités
- Un code modulaire et maintenable

Risques identifiés

Risques	**Probabilité**	**Impact**	**Mitigation**
Physique instable	Moyenne/Forte	Très fort	Testers différents moteurs physique
Latence du WebSocket	Moyenne	fort	tests en local
Performance 3D	Moyenne	Moyen	Optimisation des collisions

Critères de réussite

Le projet sera considéré comme réussi si :

- Le gameplay est fonctionnel
- La physique est crédible
- Les applications sont synchronisées
- L’expérience utilisateur est fluide
- L’architecture est propre et modulaire

8. Conventions équipe

Stratégie Git

Le projet adopte une stratégie GitFlow :

- Branche **main** : branche stable contenant uniquement du code validé. Aucun push direct n'est autorisé.
- Branche **dev** : branche d'intégration des nouvelles fonctionnalités.
- Branches **feature** : branche de travaille
    - feature/<feature>
    - bugfix/<bugfix>

Règles de fusion

- Aucun merge direct vers main
- Minimum une review obligatoire par Pull Request
- Continuous Integration obligatoire avant validation du merge

Commits

Les commits suivent la norme Conventional Commits :

```
ifeat: add bumper collision
fix: correct score sync
docs: update CDC
```

Qualité du code

Afin d'assurer la cohérence et la qualité du code :

- Utilisation de ESLint (linting)
- Utilisation de Prettier (formatage)
- Vérifications automatiques via Husky (pre-commit)

Test

Les tests sont obligatoires sur le backend.

9. Roadmap et questions ouvertes

Questions ouvertes :

- Rapierjs , CannonJs ou p2.js ?

Roadmap

Semaines	A faire
S1	Cahier des charges, Setup Github, Websocket simple, Setup ThreeJS
S2	Bille gravité, Collisions, Backglass, Réception événements, Affichage scores, Gestion d'états, Communication client, Communication ESP32 et Websocket
S3	Flipper gauche/droit fonctionels, Collision Bumper et GAME OVER
S4	Gestion multi-joueurs et jeux contre l'ordinateur, Bugfix
S5	Mapping bouton ESP32, Gestion solénoïdes, Gestion de la latence
S6	Assets, Animations, Musiques, Sons
S7	Tests, Préparation soutenance

10. Arborescence et contenu du projet

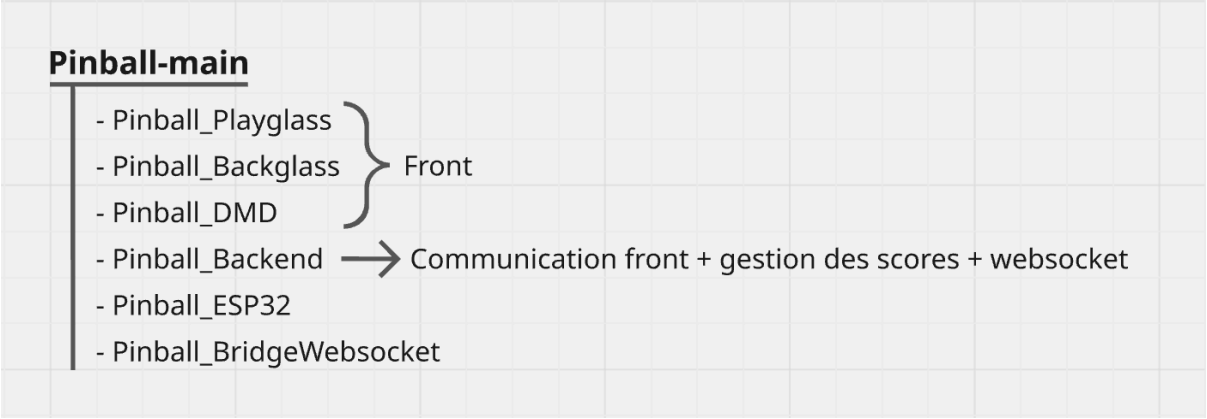
Arborescence

Le projet adopte une organisation en polyrepository (https://github.com/arthurGuillemin/pinball_main), permettant une séparation claire des responsabilités et une modularité accrue.

Repositories du projet

- **Frontend,** interface utilisateur et rendu graphique :
  - Playfield : https://github.com/arthurGuillemin/pinball_playglass
  - Backglass : https://github.com/arthurGuillemin/pinball_backglass
  - DMD : https://github.com/arthurGuillemin/pinball_dmd
- **Backend :** https://github.com/arthurGuillemin/pinball_backend
  Gestion des communications temps réel et logique métier
- **Bridge :** https://github.com/arthurGuillemin/pinball_bridge
  Interface entre le système IoT et l'application principale
- **ESP32 :** https://github.com/arthurGuillemin/pinball_esp32
  Gestion des périphériques physiques (contrôleurs / solénoïdes)

Schéma d'arboresence



11. Description graphique du projet

Le projet devra proposer :

- Une interface claire et lisible
- Un design cohérent (thème visuel)
- Des feedbacks visuels explicites
- Une expérience immersive

Lien du figma :

https://www.figma.com/design/N4ktQtBApoL2DtJ6cdA2Yz/Flipper?node-id=0-1&p=f&m=draw