UNIVERSITE COTE D'AZUR/DS4H

MIAGE – LICENCE 3 MIASHS parcours MIAGE

1645 Rte des Lucioles, 06410 Biot

Rapport de conception

PROJET INFORMATIQUE - JEUX JAVASCRIPT

Réalisé par : Nicolas Toupin, Jérémy Moncada

Encadrant :

Monsieur Michel Buffa

MAI 2024

1. Introduction

Objectif du Projet

L'objectif principal de ce projet est de développer une plateforme web
qui héberge trois jeux distincts : The Awakening, Shadow Travelers et Dream Travelers.
Cette plateforme est conçue pour offrir une navigation fluide grâce
à une architecture de type Single Page Application (SPA),
garantissant ainsi une expérience utilisateur cohérente et sans interruption.

Présentation des Jeux

- The Awakening : Un jeu en 2D qui plonge le joueur dans un univers mystérieux où il doit résoudre des énigmes et surmonter des obstacles pour progresser.
- Shadow Travelers : Un jeu de plateforme en 2D qui
  combine des éléments de stratégie et d'action,
  où le joueur doit naviguer à travers des niveaux remplis de défis et d'ennemis.
- Dream Travelers : Un jeu de réflexion en 3D isométrique qui met au défi les
  joueurs avec des puzzles complexes et des mécaniques de jeu innovantes, utilisant la puissance de Babylon.js pour des graphismes immersifs.

Répartition du Travail

Le développement de ces jeux a été réparti entre deux
membres de l'équipe. Nicolas Toupin a pris en charge le développement de The Awakening,
tandis que Jérémy Moncada a développé Shadow Travelers et Dream Travelers.

2. Architecture Générale et Technologies

Architecture Globale

Structure du Projet

Le projet est organisé en plusieurs dossiers distincts,
chacun dédié à un jeu ou à une fonctionnalité spécifique :

- TheAwakening/ : Contient les fichiers relatifs au jeu The Awakening,
  incluant le code JavaScript et les assets nécessaires.
- ShadowTravelers/ : Regroupe les fichiers pour le jeu Shadow Travelers.
- DreamTravelers/ : Héberge le code et les ressources pour le jeu Dream Travelers,
  avec une structure similaire.
- sections/ : Ce dossier est crucial pour l'approche Single Page Application (SPA).
  Il contient des sous-dossiers comme home/, progress/, et connexion/,
  qui représentent différentes sections de l'application.

Navigation SPA

L'application utilise une architecture SPA pour offrir une navigation fluide entre les différentes sections, telles que l'accueil (sections/home/), la progression (sections/progress/), et la connexion (sections/connexion/). Cette approche minimise les rechargements de page, rendant l'expérience utilisateur plus rapide et plus réactive.
Le chargement des sections est géré via JavaScript, qui injecte le contenu HTML et lance les scripts spécifiques à chaque section.

Point d'Entrée

Le fichier principal de l'application est index.html, situé à la racine du projet. Il sert de point d'entrée en chargeant la structure de base de l'application ainsi que les scripts initiaux nécessaires au fonctionnement de la SPA.

Technologies Utilisées

Frontend

Le frontend de l'application est construit avec les technologies web standards : HTML5, CSS3, et JavaScript (ES6+). Ces technologies assurent une compatibilité large et une performance optimale sur les navigateurs modernes.

Bibliothèques/Frameworks Clés

- Babylon.js : Utilisé spécifiquement pour le jeu Dream Travelers, Babylon.js est une bibliothèque puissante pour la création de graphismes 3D.
  Elle offre des capacités avancées de gestion de scène, de rendu 3D, et d'interactions, ce qui en fait un choix idéal pour développer des jeux immersifs en 3D.
- Nous avons utilisé **GSAP (GreenSock Animation Platform)** pour gérer les animations dans Dream Travelers, notamment pour les fragments. GSAP est une bibliothèque puissante et flexible qui permet de créer des animations fluides et performantes.

Outils

Pour la gestion de version et la collaboration, Git et GitHub sont utilisés. Ces outils permettent un suivi efficace des modifications du code, facilitent la collaboration entre les membres de l'équipe, et assurent une gestion structurée du développement du projet.

3. Conception Mutualisée et Composants Partagés

Stratégie de Mutualisation

La mutualisation du code est une stratégie clé adoptée pour améliorer la maintenabilité et la cohérence du projet. En factorisant le code commun, nous avons pu réduire la duplication et faciliter les mises à jour. Tout en reconnaissant les différences inhérentes à chaque jeu, nous avons cherché à partager autant de logique que possible.

Composants Partagés Globaux

Gestion de la Progression

Un système centralisé de gestion de la progression a été conçu pour suivre le niveau atteint dans chaque jeu.
Elle repose sur l'utilisation de localStorage pour stocker les identifiants de jeu (GAME_IDS) et les niveaux atteints. Ce système permet de centraliser la progression des joueurs et de l'afficher dans la section progress.

Système de Navigation

La logique de chargement des sections est un composant partagé essentiel à l'architecture SPA. Elle permet de charger dynamiquement le contenu des sections sans recharger la page entière, assurant ainsi une expérience utilisateur fluide et cohérente.

Mutualisation Spécifique aux Jeux (Design Patterns Communs)

Architecture de Niveaux (BaseLevel)

Dans les jeux Dream Travelers et Shadow Travelers, chaque niveau (ex: Level1, Level2) hérite d'une classe BaseLevel. Ce pattern permet de centraliser des fonctionnalités communes telles que l'initialisation de la scène ou du canvas, la boucle de jeu (update/draw), et la gestion des inputs basiques.

- Dream Travelers utilise Babylon.js pour gérer une scène 3D, tandis que Shadow Travelers utilise l'API Canvas pour un rendu 2D. Bien que le concept de BaseLevel soit similaire, les implémentations sont distinctes et adaptées aux besoins spécifiques de chaque jeu.

Gestionnaire de Niveaux (LevelManager)

Le LevelManager est présent dans plusieurs jeux, notamment dans TheAwakening/js/levelManager.js et ShadowTravelers/js/levelManager.js. Il orchestre le chargement et la transition entre les niveaux, assurant une gestion fluide et cohérente de la progression du joueur à travers les différents défis proposés.

4. Conception Détaillée par Jeu

4.1 The Awakening

Structure Principale

- **Classes Clés :**
- **Game** : Gère la logique principale du jeu, y compris le cycle de vie et l'état global.
- **Controls** : Gère les entrées utilisateur et les interactions.
- **Physics** : Responsable de la gestion des collisions et des mouvements physiques.
- **UI** : Gère l'affichage des éléments d'interface utilisateur.
- **LevelManager** : Orchestration du chargement et de la transition entre les niveaux.

Gameplay

The Awakening est un jeu de plateforme en 2D où le joueur doit naviguer à travers des niveaux remplis d'obstacles et de puzzles. Les mécaniques principales incluent le saut, la course et le dash qui permet de traverser des objets.

Graphismes

Le jeu utilise des sprites pour représenter les personnages et les environnements.

Défis de Conception Spécifiques

La gestion des collisions 2D et l'animation fluide des sprites ont été des défis majeurs, nécessitant une attention particulière aux détails pour assurer une expérience de jeu fluide.

4.2 Shadow Travelers

Moteur 2D

Shadow Travelers utilise l'API Canvas pour le rendu 2D, avec context et canvas pour dessiner et animer les éléments du jeu.

Structure des Niveaux

Les niveaux héritent de BaseLevel (version 2D), centralisant la logique de base et facilitant la gestion des niveaux.

Gameplay

C'est un jeu de plateforme 2D avec un scrolling horizontal géré dans BaseLevel.js. Le joueur doit naviguer à travers des niveaux remplis d'obstacles et d'ennemis.

Entité Joueur (Player.js)

Gère les mouvements 2D, y compris les sauts et les déplacements,
ainsi que les collisions avec l'environnement.

Éléments Interactifs / Ennemis

- **Ghost.js** : PNJ avec dialogues, ajoutant une dimension narrative.
- **JumpingObstacle.js** et **DirectionalObstacle.js** :
  Créent des défis dynamiques pour le joueur.
- **ShadowZone.js** et **Guardian.js** : Introduisent des mécaniques de jeu.
- **Exit.js** : Gère les sorties de niveau.

Graphismes

Les sprites sont créés, ajoutant une esthétique unique au jeu.

Défis de Conception Spécifiques

La gestion du scrolling de la caméra, les interactions spécifiques des obstacles, et la logique des fantômes/gardiens ont été des défis majeurs.

4.3 Dream Travelers

Moteur 3D

Dream Travelers utilise Babylon.js pour créer un environnement 3D immersif. La scène est gérée par une caméra isométrique ArcRotateCamera, et des lumières HemisphericLight et DirectionalLight sont utilisées pour créer des effets visuels réalistes.

Structure des Niveaux

Chaque niveau hérite de BaseLevel, centralisant la logique commune.
La classe Grid (DreamTravelers/js/elements/Grid.js) est utilisée pour gérer la position des éléments dans le monde 3D.

Entité Joueur (Player.js)

- **Chargement de Modèle 3D** :
  Le joueur est représenté par un modèle 3D (samourai_floating.glb), chargé dynamiquement dans la scène.
- **Pathfinding** : Le système de pathfinding utilise un A* simplifié. findDescentPath gère la descente, tandis que la montée est réalisée en inversant ce chemin.
  Des cas spécifiques, comme les escaliers et les plateformes rotatives, sont gérés pour assurer un mouvement fluide.

Éléments Interactifs Complexes

- **RotatingPlatform.js** :
  Gère les plateformes rotatives qui ajoutent une dimension de complexité au gameplay.
- **Clouds.js** : Crée des nuages rotatifs.
- **Slider.js** : Gère les mécaniques de glissements permettant d'attacher un joueur dessus et de le déplacer pour atteindre des emplacements spécifiques.
- **Stairs.js** : Permet au joueur de monter et descendre entre les niveaux.
- **Fragment.js** et **Exit.js** : Gèrent les objectifs de collecte et les sorties de niveau.

Défis de Conception Spécifiques

Le pathfinding complexe, la gestion des interactions 3D (clics, rotations), et la création d'énigmes stimulantes (Niveau 3) ont été des défis clés.

5. Défis Rencontrés et Solutions Techniques

Pathfinding dans Dream Travelers

Le pathfinding a été l'un des défis majeurs du projet, en particulier pour gérer les déplacements verticaux. La difficulté résidait dans la montée, qui nécessitait une approche innovante.
La solution a été d'utiliser un algorithme A* simplifié pour la descente, puis d'inverser ce chemin pour gérer la montée.
Cette stratégie a permis de réutiliser efficacement la logique existante et d'assurer un mouvement fluide du joueur.

Complexité des Niveaux

Concevoir des énigmes intéressantes, notamment pour le Niveau 3 de Dream Travelers, a été un défi significatif. Chaque jeu devait offrir des parcours stimulants et variés, ce qui a nécessité une réflexion approfondie sur le design des niveaux et les mécaniques de jeu.

Gestion des Assets

La création manuelle de sprites pour The Awakening et Shadow Travelers a ajouté une couche de complexité, tout comme l'intégration de modèles 3D dans Dream Travelers. Ces tâches ont nécessité une attention particulière pour garantir une cohérence visuelle et tech nique à travers les jeux.

Cohérence de l'Expérience Utilisateur

Assurer une navigation fluide et une présentation homogène malgré les différences techniques entre les jeux a été essentiel. L'architecture SPA a joué un rôle clé en permettant une transition sans heurts entre les différentes sections et jeux.

6. Conclusion et Pistes d'Amélioration

Bilan

Le projet a abouti à la création de trois jeux fonctionnels intégrés dans une plateforme SPA, avec une mutualisation partielle du code. Les points forts incluent la solution efficace du pathfinding et une navigation fluide, offrant une expérience utilisateur cohérente.

Apprentissages

Ce projet a permis de maîtriser des outils et concepts clés tels que Babylon.js pour le rendu 3D, la conception modulaire pour une meilleure maintenabilité, et l'algorithme A* pour le pathfinding. La gestion de projet et la collaboration ont également été renforcées.

Pistes d'Amélioration / Travaux Futurs

- **Factorisation Plus Poussée** :
  Envisager la création d'une classe Entity de base pour centraliser davantage de logique commune.
- **Amélioration des Performances** :
  Optimiser le code et les ressources pour une meilleure réactivité.
- **Ajout de Plus de Niveaux / Nouveaux Jeux** :
  Étendre le contenu pour enrichir l'expérience utilisateur.
- **Tests Unitaires / Fonctionnels** : Mettre en place des tests pour garantir la robustesse et la fiabilité du code.