Yūrei

Document de Conception Technique (v1.0)

Auteur : Valérie Lecœur
01/10/2025

Table des matières

1. Introduction

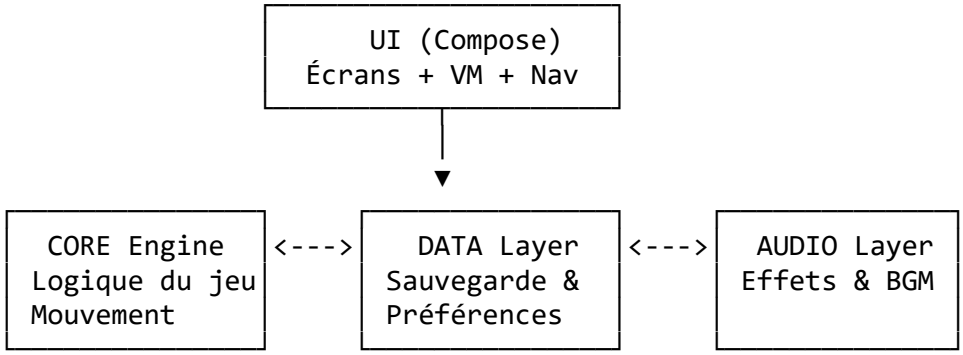
Le document présente la conception technique du jeu **Yūrei**, un projet Android inspiré du style *Meynapède*, basé sur **Kotlin** et **Jetpack Compose**. Il définit la structure du dépôt, les modules, les technologies et les choix architecturaux pour garantir modularité et maintenabilité.

2. Objectifs du projet

- Architecture modulaire et évolutive (multi-modules Kotlin/Android).
- Utilisation du modèle **MVVM** pour séparer la logique du rendu.
- Interface moderne avec **Jetpack Compose**.
- Compatibilité Android 10+ (API 29 à 36).
- Gameplay fluide et optimisé pour 60 FPS.

3. Architecture logicielle

3.1 Schéma global



3.2 Modules

Module	Description	Type
app	Entrée principale, injection et navigation	Application
core	Moteur du jeu (entités, collisions, logique)	Kotlin pur
ui	Interface utilisateur (Compose, écrans, thèmes)	Android Library
data	Sauvegardes et préférences	Android Library
audio	Musique et effets sonores	Android Library

4. Structure du dépôt

```
Yurei/
├── app/
├── core/
├── ui/
├── data/
├── audio/
└── assets/
```

Chaque module possède son propre `build.gradle.kts` et est inclus dans `settings.gradle.kts`.

5. Principes d'architecture

5.1 Modèle MVVM

- **Model** : logique du jeu, entités, persistance.
- **ViewModel** : coordination des états et du cycle de jeu.
- **View** : rendu visuel et interactions via Compose.

5.2 Responsabilités principales

Composant	Description
`GameLoop`	Contrôle du cycle de mise à jour (60 FPS)
`World`	Gestion du monde et collisions
`Dragon`	Joueur principal et comportement
`GameRepository`	Sauvegarde et chargement des données
`AudioManager`	Gestion audio (BGM/SFX)
`GameScreen`	Rendu principal via Compose.Canvas

6. Technologies et outils

Catégorie	Choix	Justification
Langage	Kotlin	Moderne, concis, multiplateforme
UI	Jetpack Compose	Interface réactive et fluide
Architecture	MVVM	Séparation claire et testable

Catégorie	Choix	Justification
Persistance	DataStore	Alternative moderne à SharedPreferences
Audio	ExoPlayer / SoundPool	Streaming et effets légers
Build	Gradle KTS	Configuration typée et modulaire
Tests	JUnit / Espresso	Validation unitaire et UI

7. Paramètres techniques

- **minSdk** : 29 (Android 10)
- **targetSdk / compileSdk** : 36
- **Langage JVM** : Kotlin 2.0.0 (JVM target 17)
- **Compose Compiler** : 1.5.14
- **Matériel visé** : Smartphones Android récents (RAM > 3 Go)

8. Intégration continue (prévisionnelle)

- **GitHub Actions** :
  - `build.yml` → build + lint + test sur `push/pull-request`
  - `release.yml` → publication sur Play Console (staging)
- Analyse statique : `ktlint`, `detekt`

9. Sécurité et performance

- Ressources optimisées (`.webp`, `.ogg`).
- Jeu **offline** (aucune dépendance réseau).
- Gestion mémoire soignée (entités recyclées, textures préchargées).

- Boucle stable à **60 FPS** via `GameLoop`.

10. Étapes suivantes

1. Création du squelette multi-modules Kotlin.
2. Configuration de `libs.versions.toml`.
3. Développement du `GameViewModel`.
4. Intégration du système audio et DataStore.
5. Rédaction du document de conception détaillée (diagrammes UML, séquence et données).