Répartition des Tâches - Projet Wargame

Vue d'ensemble

Ce document détaille la répartition des tâches et l'organisation du projet Wargame entre les 5 membres de l'équipe.

Répartition des Rôles

1. Développeur Core Game Logic

Responsabilités : Logique fondamentale du jeu

Tâches :

- Implémentation des règles du jeu
- Gestion des tours de jeu
- Système de combat
- Calcul des déplacements
- Classes de base (Unit, Terrain, Player)
- Tests unitaires pour la logique de base

2. Développeur Interface Graphique (GUI)

Responsabilités : Interface utilisateur complète

Tâches :

- Création de la fenêtre principale
- Affichage du plateau hexagonal
- Rendu des unités et terrains
- Gestion des interactions utilisateur
- Menu du jeu
- Interface de configuration

3. Développeur Gestion des Données

Responsabilités : Persistance et gestion des données

Tâches :

- Système de sauvegarde/chargement
- Gestion des scénarios
- Configuration du jeu
- Gestion des ressources (images, sons)
- Sérialisation/désérialisation
- Base de données des unités et terrains

4. Développeur IA et Événements

Responsabilités : Intelligence artificielle et événements spéciaux

Tâches :

- Implémentation de l'IA des adversaires
- Système d'événements externes
- Actions d'opportunité
- Pathfinding
- Stratégies de l'IA
- Tests des comportements de l'IA

5. Développeur Intégration et Extensions

Responsabilités : Intégration et fonctionnalités additionnelles

Tâches :

- Coordination de l'intégration des composants
- Éditeur de scénarios
- Mode campagne
- Système de ligne de tir
- Gestion du brouillard de guerre
- Tests d'intégration

Structure du Projet

Architecture Maven

```
src/
├── main/
│   ├── java/
│   │   ├── core/          # Junior
│   │   ├── gui/           # Rayane
│   │   ├── data/          # Claudia
│   │   ├── ai/            # Thed
│   │   └── extensions/    # Marwan
│   └── resources/
└── test/
    └── java/
```

Dépendances entre Modules

```
core/ <-- Utilisé par tous les autres modules
data/ <-- Utilisé par gui/, ai/, extensions/
gui/  <-- Utilisé par extensions/
ai/   <-- Utilisé par extensions/
```

Organisation et Communication

Points de Communication

1. **Interfaces bien définies**
    - Documentation claire des API
    - Contrats d'interface explicites
    - Points d'intégration définis
2. **Réunions**
    - Réunion hebdomadaire de synchronisation
    - Points quotidiens rapides si nécessaire
    - Revues de code régulières
3. **Outils de Collaboration**
    - Git pour le contrôle de version
    - Issues pour le suivi des tâches
    - Wiki pour la documentation

Bonnes Pratiques

1. **Développement**
    - Utilisation de design patterns
    - Code documenté avec Javadoc
    - Tests unitaires systématiques
    - Respect des conventions de codage Java
2. **Intégration**
    - Branches par fonctionnalité
    - Pull requests obligatoires
    - Tests d'intégration automatisés
    - Revue de code avant merge

Planning

Phase 1 : Initialisation

- Mise en place de l'architecture
- Définition des interfaces
- Configuration de l'environnement

Phase 2 : Développement Core

- Implémentation des composants de base
- Développement parallèle des modules
- Tests unitaires

Phase 3 : Intégration

- Intégration progressive des composants
- Développement des fonctionnalités avancées
- Tests d'intégration

Phase 4 : Finalisation

- Tests complets
- Débogage
- Optimisation
- Documentation

Phase 5 : Préparation Soutenance

- Préparation des livrables
- Rédaction du rapport

- Préparation de la présentation

Livrables

Documentation

- Javadoc pour chaque module
- Documentation technique
- Manuel utilisateur
- Rapport final

Code

- Code source commenté
- Tests unitaires et d'intégration
- Fichier JAR exécutable

Présentation

- Support de présentation
- Démonstration fonctionnelle
- Documentation des choix techniques