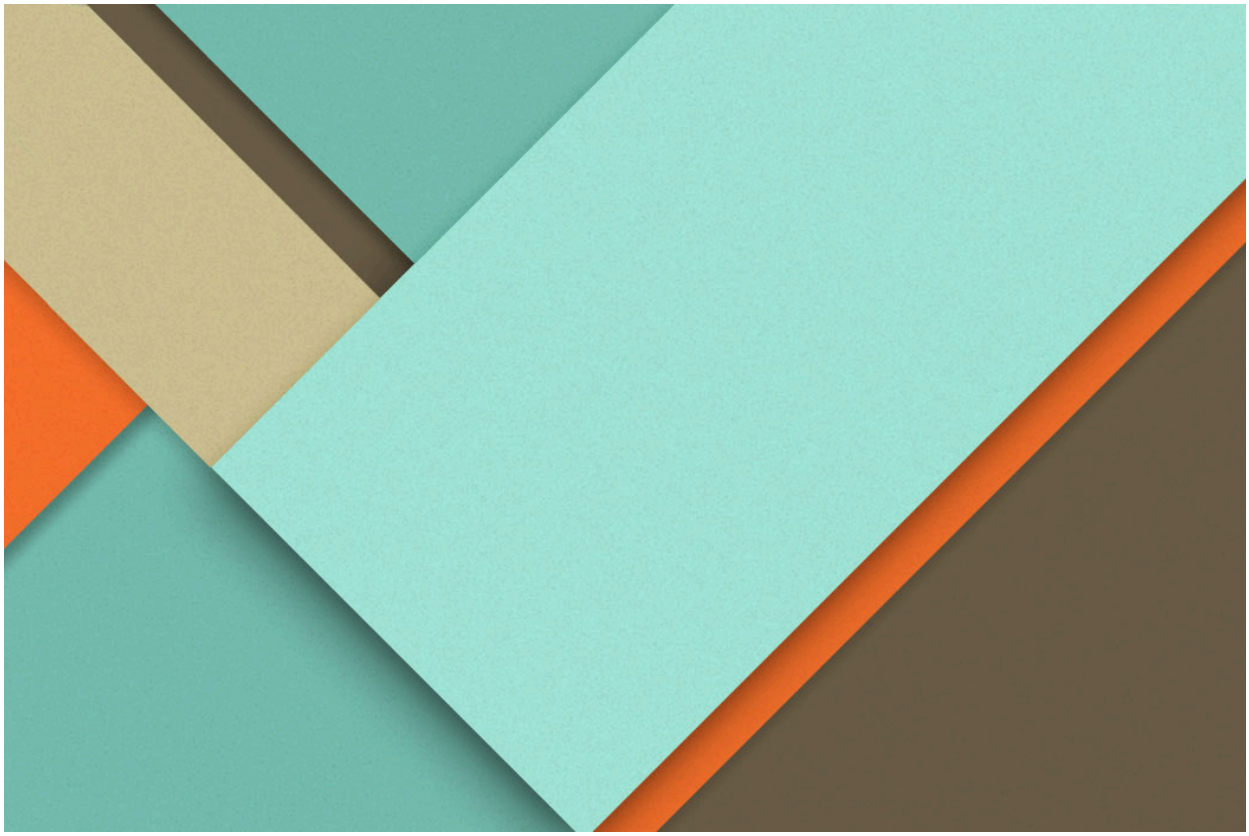

Monsage

16.04.2025-03.06.2025

—

Moreau Baptiste, Bertrand Arthur, Mouniapin Théo

Groupe 3il

Document Technique

1. Introduction

Le présent document détaille l'architecture technique et le fonctionnement interne du jeu d'aventure textuel "MONSAGE". Ce jeu de type rogue-like procédural met le joueur aux commandes d'une équipe de personnages progressant à travers des étages générés dynamiquement, affrontant des monstres et gérant des événements aléatoires. L'objectif est de fournir une compréhension claire des mécanismes de jeu, de l'interaction entre les classes et des conventions de développement adoptées.

2. Architecture Générale

Le programme est structuré en plusieurs packages principaux, chacun encapsulant une fonctionnalité clé du jeu :

- Jeu : Contient la classe principale Main (point d'entrée et boucle de jeu) et la classe Équipe (gestion du groupe de personnages et de leur inventaire commun).
- Personnage : Définit la classe abstraite Personnage et ses spécialisations concrètes Tank, Mage, Support, Dps. Ces classes gèrent les statistiques et la progression des héros. Ces quatre classes sont elles-mêmes parentes des huit autres (ex : Assassin, Chevalier, Nécromancien, Prêtre) qui définissent les stats spéciales et compétences uniques. Contient aussi la classe EffetTemporaire pour gérer les buffs, debuffs et dégâts sur la durée (DoT) appliqués aux entités.
- Monstre : Comprend la classe Monstre (et ses sous-classes potentielles comme MercenairePNJ) définissant les antagonistes, leurs attributs et comportements d'attaque.
- Inventaire : Contient les classes Inventaire (gestion des objets et de l'or de l'équipe) et Objet (définition des items consommables et de leurs effets).
- Event : Définit la classe abstraite Evenement et ses implémentations (TerrainRuine, Ecclesiastique, Mercenaire, Fantôme, Shop) qui introduisent des scénarios interactifs et des modificateurs de jeu.
- Db : Gère la persistance des données via la classe Database, utilisant SQLite pour la sauvegarde et le chargement des parties.

3. Flux Principal du Jeu (Classe Main)

La classe Main orchestre le déroulement du jeu. Son flux principal peut être résumé comme suit :

A. Initialisation **:** Connexion à la base de données (Database), initialisation du Scanner pour les entrées utilisateur.

2. Gestion de Session : Une boucle permet de démarrer une nouvelle session de jeu (après une défaite ou en quittant une partie précédente).
3. Démarrage de Partie :
    - Saisie du pseudo du joueur.
    - Vérification des sauvegardes existantes pour ce joueur.
    - Proposition de charger une partie existante ou de commencer une "Nouvelle Partie".
        - Chargement : Les données de l'équipe, de l'inventaire et de l'étage sont restaurées depuis la base de données. Si une sauvegarde non-la-plus-récente est chargée, les sauvegardes ultérieures pour ce joueur sont supprimées.
        - Nouvelle Partie : Une nouvelle instance d'Equipe est créée, et le joueur choisit son personnage initial.
4. Boucle d'Étage : Tant que l'équipe est victorieuse et que le joueur souhaite continuer :
    - Affichage de l'étage actuel.
    - Déclenchement d'Événement (si applicable) : À certains paliers d'étages (étages 5, 15, 25...), un événement aléatoire (TerrainRuine, Ecclesiastique, etc.) est sélectionné et déclenché. Chaque événement possède sa propre logique d'interaction et de conséquences.
    - Phase de Préparation : Le joueur peut accéder au Shop, gérer son InventaireCommun, ou consulter les Stats Équipe.
    - Phase de Combat :
        - Génération des Monstres pour l'étage (nombre et type variant avec le niveau de l'étage, incluant des boss).
        - Boucle de combat par rounds :
            - Tour des Joueurs : L'équipe dispose de 4 Points d'Action (PA). Le joueur sélectionne un personnage et une action (Attaque de base, Ultime, Utiliser Objet, Changer de Place). Un personnage ne peut effectuer qu'une action de combat par tour d'équipe, mais utiliser un objet consomme un PA sans compter comme action de combat. Les effets temporaires des personnages sont mis à jour.
            - Tour des Monstres : Chaque monstre vivant effectue une attaque sur le personnage en tête de l'équipe (le "leader"). Les effets temporaires des monstres sont mis à jour.
            - Vérification des conditions de fin de combat (équipe anéantie ou tous les monstres vaincus).

- Fin de Combat :
  - Victoire : L'équipe gagne de l'XP (répartie entre les membres vivants) et de l'or. L'étage est incrémenté. Le joueur peut se voir proposer d'ajouter un nouveau membre à son équipe à certains paliers.
  - Défaite : La session de jeu se termine, proposant de recommencer.

5. Menu Pause **:** Accessible à la plupart des points d'entrée utilisateur en tapant "M". Permet de Reprendre, Sauvegarder et Quitter (crée une nouvelle sauvegarde, gère la limite de 5 par joueur), ou Quitter sans Sauvegarder. La variable globale continuerJeuGlobal et des valeurs de retour spéciales des méthodes de lecture gèrent la sortie des boucles.
6. Fin de Session **:** Déconnexion de la base de données et fermeture du Scanner.

4. Interaction des Classes Clés

- Equipe et Personnage : Equipe maintient une List<Personnage>. Les actions des personnages (attaques, ultis) sont définies dans leurs classes respectives et affectent les instances de Monstre. La progression (XP, niveaux) est gérée au niveau de Personnage.
- Personnage et Monstre avec EffetTemporaire: Les deux peuvent être affectés par des EffetTemporaire (buffs, debuffs, DoTs). Les méthodes appliquerEffet, mettreAJourEffets, et annulerLogiqueEffetStats dans Personnage et Monstre gèrent l'application, la durée, et la suppression des effets, ainsi que leur impact mécanique (modification des stats de base via des modificateur...Effets, application de dégâts directs pour les DoTs).
- Equipe et Inventaire : L'instance d'Inventaire est un attribut d'Equipe, représentant l'inventaire partagé. Les Objets sont stockés dans un tableau objetsSpecifiques à slots fixes.
- Objet et Cibles (Personnage/Monstre) : La méthode Objet.utiliser(Personnage utilisateur, Object cibleGenerique) détermine le type de la cibleGenerique (via instanceof) et applique son effet (soin, application d'un EffetTemporaire).
- Evenement et Equipe : Les sous-classes d'Evenement interagissent avec l'Equipe pour modifier son or, la santé de ses membres, ou déclencher des combats spéciaux (e.g., Mercenaire).

- Main et Database : Main utilise Database pour initialiser le jeu (charger/nouvelle partie) et pour sauvegarder la progression via le menu pause. Database encapsule toutes les requêtes SQL SQLite.

5. Règles Techniques et Conventions

- Gestion des Entrées Utilisateur : Les méthodes statiques Main.lireStringAvecMenuPause et Main.lireIntAvecMenuPause sont utilisées pour toutes les lectures du Scanner. Elles permettent d'intercepter une commande "M" pour ouvrir le menu pause.
- Persistance : SQLite est utilisé pour la sauvegarde. La classe Database gère la création des tables (Sauvegarde, PersonnageSauvegarde, InventaireEquipe) et les opérations CRUD. Les sauvegardes sont liées à un nomJoueur et une limite de 5 sauvegardes par joueur est appliquée (la plus ancienne est supprimée).
- Gestion des Effets : Un système d'objets EffetTemporaire est utilisé. Ces objets sont créés via des méthodes factory statiques pour la cohérence. Ils sont stockés dans une liste par Personnage et Monstre. Leur impact mécanique est géré par des modificateur...Effets sur les stats de base et par des logiques spécifiques dans mettreAJourEffets pour les DoTs.
- Scaling des Monstres : Les statistiques des monstres (PV, force, défense) et le nombre de monstres par étage augmentent dynamiquement en fonction du niveau de l'étage, suivant des règles prédéfinies pour assurer une courbe de difficulté progressive.
- Affichage Console : Des codes couleurs ANSI et emojis sont utilisés pour améliorer la lisibilité. L'affichage des statistiques d'équipe et de l'état du combat est formaté pour une présentation en colonnes.
- Modularité : Le découpage en packages vise à maintenir une séparation des préoccupations et à faciliter la maintenance et l'évolution du code.

6. Conclusion

Cette architecture fournit une base solide pour le jeu "Donjon Infini". Les interactions entre les classes sont conçues pour être logiques et extensibles. Les défis futurs incluent l'équilibrage fin des statistiques et des compétences, l'ajout de plus de contenu (personnages, monstres, objets, événements), et potentiellement l'amélioration de l'interface utilisateur.