**1. Classe Jeu**

C'est la classe principale qui orchestre le jeu. Elle peut gérer l'initialisation du jeu, la boucle principale, les entrées utilisateurs et le rendu des éléments.

**Responsabilités** :

* Initialisation de SFML et des autres composants.
* Boucle de jeu (mise à jour, rendu).
* Gestion des événements (entrées clavier, souris).
* Interaction avec les autres systèmes (système de quêtes, combat, etc.).

**Attributs** :

* sf::RenderWindow window : Fenêtre du jeu.
* Monde monde : Le monde du jeu.
* Personnage joueur : Le joueur (instance de la classe personnage).
* std::vector<Quete> quetes : Liste des quêtes disponibles.

**Méthodes** :

* void initialiser()
* void bouclePrincipale()
* void gererEvenements()

**2. Classe Monde**

La classe Monde contient toutes les informations sur le monde du jeu, y compris les cartes, les zones et la génération procédurale.

**Responsabilités** :

* Chargement et gestion du monde du jeu.
* Génération procédurale de la carte.
* Chargement des villes, villages, donjons, etc.
* Gestion du terrain et des environnements.

**Attributs** :

* std::vector<Zone> zones : Une liste de zones ou de chunks du monde.
* std::vector<Entite> entites : Liste des entités (PNJ, monstres).

**Méthodes** :

* void genererTerrain()
* void chargerZone()
* void ajouterEntite()

**3. Classe Zone**

Une **Zone** peut représenter une portion de la carte ou un environnement spécifique (comme une ville, un donjon ou une zone sauvage).

**Responsabilités** :

* Stocker les informations spécifiques à une zone.
* Charger et afficher des entités spécifiques à la zone.
* Gérer la logique propre à la zone (quêtes, ennemis, etc.).

**Attributs** :

* std::string nom : Le nom de la zone.
* std::vector<Entite> entites : Liste des entités dans cette zone.
* sf::Texture textureTerrain : Texture pour le terrain de la zone.

**Méthodes** :

* void afficher()
* void gererEntites()

**4. Classe Entite (ou Personnage, PNJ, Monstre)**

Les entités sont des objets dans le monde du jeu, comme des personnages jouables, des PNJ ou des ennemis.

**Responsabilités** :

* Gérer les caractéristiques de base des entités.
* Gérer les mouvements, la santé, les interactions, etc.

**Attributs** :

* sf::Sprite sprite : Le sprite de l'entité.
* int pointsDeVie : La vie de l'entité.
* int pointsDeMagie : La magie de l'entité (pour les magiciens).
* sf::Vector2f position : La position dans le monde.

**Méthodes** :

* void deplacer()
* void attaquer()
* void recevoirDegats()

**5. Classe Personnage (hérite de Entite)**

La classe Personnage représente le joueur. Elle hérite de Entite, mais a des fonctionnalités spécifiques au joueur, comme l'inventaire, les compétences et l'équipement.

**Responsabilités** :

* Gérer l'inventaire du joueur.
* Gérer les compétences et les niveaux.
* Gérer les quêtes du joueur.

**Attributs** :

* std::vector<Item> inventaire : Inventaire du joueur.
* std::vector<Competence> competences : Liste des compétences.
* std::vector<Quete> quetesEnCours : Quêtes en cours.
* sf::Keyboard::Key toucheDeplacement : Les touches associées au déplacement.

**Méthodes** :

* void utiliserObjet()
* void attaquer()
* void activerSort()

**6. Classe Quete**

Une **quête** représente un objectif à accomplir par le joueur. Les quêtes peuvent avoir différents états (non commencée, en cours, terminée).

**Responsabilités** :

* Gérer les objectifs de la quête.
* Suivre l'état de la quête.

**Attributs** :

* std::string titre : Titre de la quête.
* std::string description : Description de la quête.
* bool estComplete : Statut de la quête.
* std::vector<Objectif> objectifs : Objectifs à accomplir.

**Méthodes** :

* void marquerCommeComplete()
* bool verifierObjectifs()

**7. Classe Objectif**

Chaque quête contient des objectifs spécifiques, comme trouver un objet, tuer un ennemi, ou parler à un PNJ.

**Responsabilités** :

* Définir un objectif à accomplir pour la quête.

**Attributs** :

* std::string type : Type de l'objectif (parler, tuer, ramasser, etc.).
* bool estAccompli : L'état de l'objectif.

**Méthodes** :

* void accomplir()
* bool verifierAccomplissement()

**8. Classe Combat**

Le système de combat gère les interactions entre le joueur et les ennemis.

**Responsabilités** :

* Gérer le combat en temps réel.
* Calculer les dégâts.
* Appliquer les effets du combat (statistiques, conditions, etc.).

**Attributs** :

* Entite\* attaquant : Le personnage qui attaque.
* Entite\* defendant : L'entité qui défend.

**Méthodes** :

* void lancerAttaque()
* void calculerDegats()
* void appliquerEffet()

**9. Classe Item**

Les objets dans le jeu, comme les armes, armures, potions, etc.

**Responsabilités** :

* Gérer les objets ramassés par le joueur.

**Attributs** :

* std::string nom : Nom de l'objet.
* std::string type : Type d'objet (arme, armure, potion).
* int valeur : Valeur de l'objet (prix ou effet).

**Méthodes** :

* void utiliser()
* void equiper()

**10. Classe UI (Interface utilisateur)**

Cette classe s'occupe de l'affichage de l'interface utilisateur : les barres de vie, les menus, les fenêtres de dialogue, etc.

**Responsabilités** :

* Afficher les éléments à l'écran comme les barres de vie, les quêtes, et les inventaires.

**Attributs** :

* sf::Text texte : Texte affiché à l'écran.
* sf::RectangleShape barreDeVie : Représente la barre de vie.

**Méthodes** :

* void afficherInventaire()
* void afficherBarreDeVie()

**Résumé du diagramme de classe :**

Votre diagramme de classes pourrait contenir les éléments suivants :

* **1- Jeu** : Composant principal qui orchestre tout.
* **2- Monde** : Contient les différentes zones du jeu.
* **3- Zone** : Représente des parties spécifiques du monde.
* **4- Entité** : Classe de base pour les personnages et ennemis.
* **5- Personnage** : Hérite d’Entité et représente le joueur.
* **6- Quête** et **Objectif** : Gèrent les quêtes et leurs objectifs.
* **7- Combat** : Gestion du système de combat.
* **8- Item** : Représente des objets que le joueur peut ramasser et utiliser.
* **9- UI** : Interface utilisateur pour afficher les informations à l'écran

Une image contenant cercle, diagramme, dessin, croquis

Description générée automatiquement