BELBACHIR Yanis

ARAR Karim

Aventure sur Namek

Figure 1 By Yanis and Karim

1. ***Introduction***

Introduction au projet :

Le projet est un jeu d'aventure textuel où le joueur interagit avec des salles, des objets, et effectue des actions pour progresser dans le jeu. Les éléments principaux incluent un joueur, des salles, des commandes, des objets, et un moteur de jeu.

Contexte du jeu :

Le jeu *Aventure sur Namek* se déroule sur la planète fictive de Namek, un lieu emblématique de l'univers *Dragon Ball Z*. Le joueur y incarne un aventurier qui doit

parcourir diverses salles pour collecter les sept boules de cristal dispersées à travers différentes zones. Toutefois, il devra éviter d'entrer dans le vaisseau de Freezer, car cela entraînera une défaite immédiate.

Objectif du jeu :

Le but du jeu est de récupérer les 7 boules de cristal situées dans différentes salles de la planète Namek et de les ramener au village. . Le joueur doit naviguer d'une salle à l'autre, collecter les boules et éviter le vaisseau de Freezer. Dès que toutes les boules sont collectées, le joueur gagne. Si le joueur entre dans le vaisseau de Freezer, il perd.

2. Guide utilisateur

Commandes du jeu :

 help : Affiche un message d'aide avec les commandes disponibles dans le jeu.

- ***Exemple : help***

 take <objet> : Le joueur ramasse un objet de la salle et l'ajoute à son inventaire.

- ***Exemple : take Boule de cristal***

 go <direction> : Le joueur se déplace vers une salle voisine dans la direction spécifiée (N, S, E, O, U, D).

- ***Exemple : go N***

 look : Affiche tous les objets présents dans la salle actuelle**.**

- ***Exemple : look***

 check : Vérifie l'inventaire du joueur.

- ***Exemple : check***

 drop <objet> : Permet au joueur de poser un objet de son inventaire dans la salle actuelle.

- ***Exemple : drop Boule de cristal***

 back : Permet au joueur de revenir à la salle précédente**.**

- ***Exemple : back***

 quit : Quitte le jeu.

- ***Exemple : quit***

Conditions de victoire :

- Le joueur gagne dès qu'il a récupéré toutes les boules de cristal et les a ramenées au village.

Conditions de défaite :

- Le joueur perd si, lors de ses déplacements, il entre dans la salle de Freezer (le vaisseau de Freezer). C'est une défaite immédiate.

3.Guide développeur

Description des classes :

Classe Game

- **Rôle** : Gère le flux principal du jeu, y compris la configuration, la boucle de jeu, et le traitement des commandes.
- **Attributs** :
  - finished : Indique si le jeu est terminé.
  - rooms : Liste des salles disponibles.
  - commands : Dictionnaire des commandes disponibles.
  - player : Instance de la classe Player.
  - directions : Ensemble des directions valides.
  - items : Liste des objets du jeu.
- **Méthodes principales** :
  - setup() : Initialise les salles, commandes, et objets.
  - play() : Lance la boucle principale du jeu.
  - process_command(command_string) : Traite les commandes saisies par le joueur.

Classe Player

- **Rôle** : Représente le joueur dans le jeu.
- **Attributs :**
  - name : Nom du joueur.

    - history : Liste des salles visitées.
    - current_room : Salle actuelle du joueur.
    - inventory : Inventaire du joueur (dictionnaire d'objets).
    - max_weight : Poids maximal que le joueur peut porter.
- **Méthodes principales** :
    - move(direction) : Déplace le joueur dans une direction donnée.
    - back() : Retourne à la salle précédente.
    - get_inventory() : Affiche le contenu de l'inventaire.

Classe Room

- **Rôle** : Représente une salle dans le jeu.
- **Attributs :**
    - name : Nom de la salle.
    - description : Description de la salle.
    - exits : Dictionnaire des sorties possibles.
    - inventory : Ensemble des objets présents dans la salle.
    - Characters : Dictionnaire des personnages présents
- **Méthodes principales** :
    - get_exit(direction) : Retourne la salle adjacente dans une direction donnée.
    - get_exit_string() : Retourne une description des sorties disponibles.
    - get_inventory() : Retourne les objets présents dans la salle.
    - add_to_inventory(item) : Ajoute un objet à la salle.
    - remove_from_inventory(item) : Retire un objet de la salle.

Classe Command

- **Rôle** : Représente une commande que le joueur peut utiliser.
- **Attributs :**

    - command_word : Mot clé de la commande.
    - help_string : Description de l'aide pour la commande.
    - action : Fonction exécutée par la commande.
    - number_of_parameters : Nombre de paramètres requis.
- **Méthodes principales** :
    - __str__() : Retourne une représentation textuelle de la commande.

Classe Item

- **Rôle** : Représente un objet que le joueur peut ramasser ou utiliser.
- **Attributs :**
    - name : Nom de l'objet.
    - description : Description de l'objet.
    - weight : Poids de l'objet.
- **Méthodes principales** :
    - __str__() : Retourne une représentation textuelle de l'objet.

Diagramme de classes :

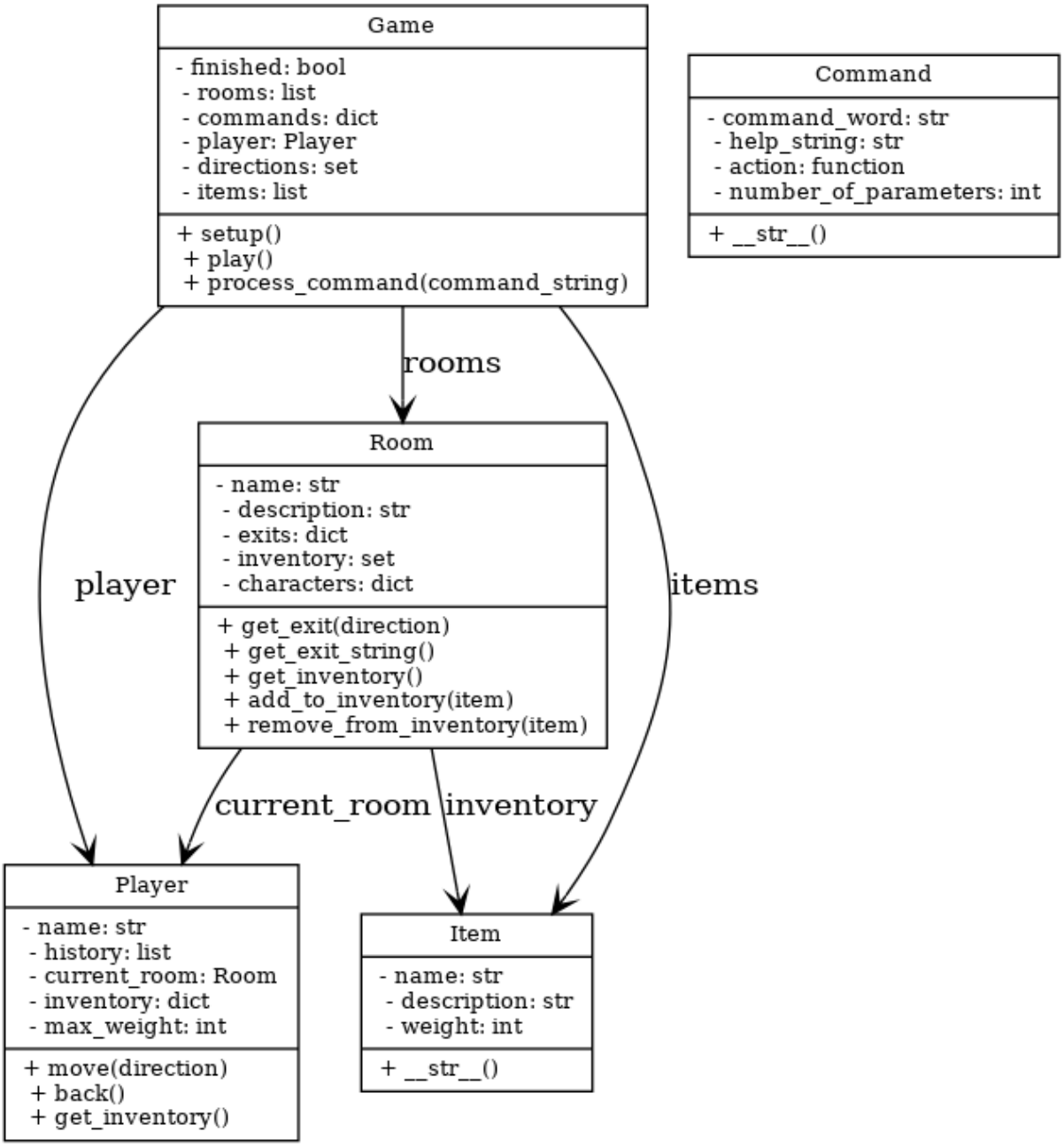

Figure 2: Diagramme de classe

Explications des relations entre les classes :

 Game : La classe principale qui orchestre toutes les autres classes. Elle gère le joueur, les salles, les objets, et les commandes.

 Player : Représente le joueur et interagit avec les salles (Room) et les objets (Item).

 Room : Contient des objets (Item) et des sorties vers d'autres salles.

 Item : Des objets manipulables présents dans les salles ou l'inventaire du joueur.

 Command : Définit les commandes utilisables par le joueur.

Comment contribuer et étendre le projet :

 Ajouter de nouvelles commandes :

- Créez une fonction dans le module actions.
- Ajoutez une nouvelle instance de Command dans la méthode Game.setup().

 Ajouter de nouveaux objets :

- Instanciez un nouvel objet de la classe Item.
- Ajoutez-le à une salle en utilisant Room.add_to_inventory().

 Étendre les salles :

- Créez une nouvelle salle avec Room.
- Connectez-la à d'autres salles via le dictionnaire exits.

 Ajouter des tests :

- Utilisez des assertions ou un framework comme unittest pour valider le comportement des classes.

3. Perspectives de développement

Améliorations envisagées :

 Ajout de nouveaux personnages et ennemis : Par exemple, un système de combat où le joueur doit affronter Freezer avant de récupérer les boules.

 Système de sauvegarde : Permettre au joueur de sauvegarder sa progression et de reprendre plus tard.

 Ajout d'énigmes : Chaque salle pourrait comporter une énigme ou un puzzle que le joueur doit résoudre pour récupérer certaines boules de cristal.

4. CONCLUSION

Le jeu *Aventure sur Namek* présente une expérience ludique simple mais captivante où le joueur doit naviguer à travers diverses salles, collecter les boules de cristal et éviter de tomber dans le piège du vaisseau de Freezer. Bien que la version actuelle soit textuelle, elle constitue une base solide pour des extensions futures, notamment avec une interface graphique, un système de combat et des mécaniques de gameplay plus avancées.

5. ANNEXES

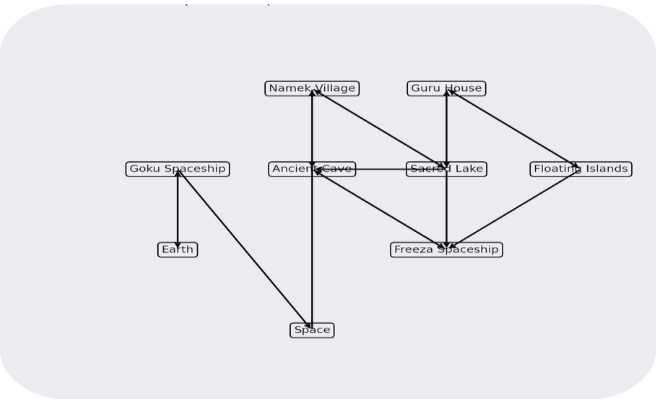

Figure 3: Organisation des différents lieux