**Rapport de projet POO: “The fountain of youth’s tale”**

**Documentation : Comment jouer à ce jeu ?**

Au début de votre aventure, une histoire vous sera proposée.

Avancez dans l’histoire en tapant dans la console les mots clé que vous souhaitez, ces mots clés seront proposé dans la phrase et seront sous forme de « mot-clé ».

Selon les mots clés que vous allez rentrer différents scénarios vont s’offrir à vous, certains vont vous mener à votre but tandis que d’autres vous feront perdre la partie. Lorsque vous perdez la console s’arrêtera simplement.

**Documentation développeur:**

Le diagramme de classe/UML: Diagram

Description automatically generated

Pour commencer, voici notre diagramme de classe. Elle a été notre première étape, elle est aussi un Schéma UML afin de pouvoir voir facilement les associations entre les différentes classes.

Notre classe que l’on pourrait appeler principale est : Location. C’est une classe abstraite car elle sera ensuite implémentée par les autres sous classes qui sont en l’occurrence nos planètes.

**Location** : Elle est caractérisée par un character, une MAP, le nom d’une planète en l’occurrence, nous avons choisi que la planète « Earth » serait la première c’est donc pour ça qu’elle est une constante.

Les sous classes sont les planètes que chacun d’entre nous sommes réparties. Elles héritent donc de cette classe Location.

**Items** : Le personnage principal aura une classe Item qui elle aussi sera abstraite. Elle est implémentée par les sous classes Drug,Food,Tale et Weapon. Items est un inventaire qui stocke les différentes données que le personnage possède dans le jeu.

**Player :** La classe Player quant à elle « crée » les attributs du joueurs protagoniste. Le Player devra savoir dans quelle planète il se situe, de qui est fait son équipage, et de quels objets il détient. C’est dans cette classe qu’il pourra aussi ajouter de nouvelles personnes dans son équipage et de nouveaux objets.

**NPCharacter** : Cette classe correspond à tous les personnages « secondaire » que le protagoniste pourra recruter dans son équipage. Ces character auront pour attribut :

* Une spécialité (Type énuméré dans enums)
  + ASTRONOMY
  + FOOD,
  + WEAPON,
  + MEDICINE,
  + STRATEGY,
  + ENGINEERING,
  + NONE
* Un nom

Cette classe aidera le « héro » à gagner la bataille finale.

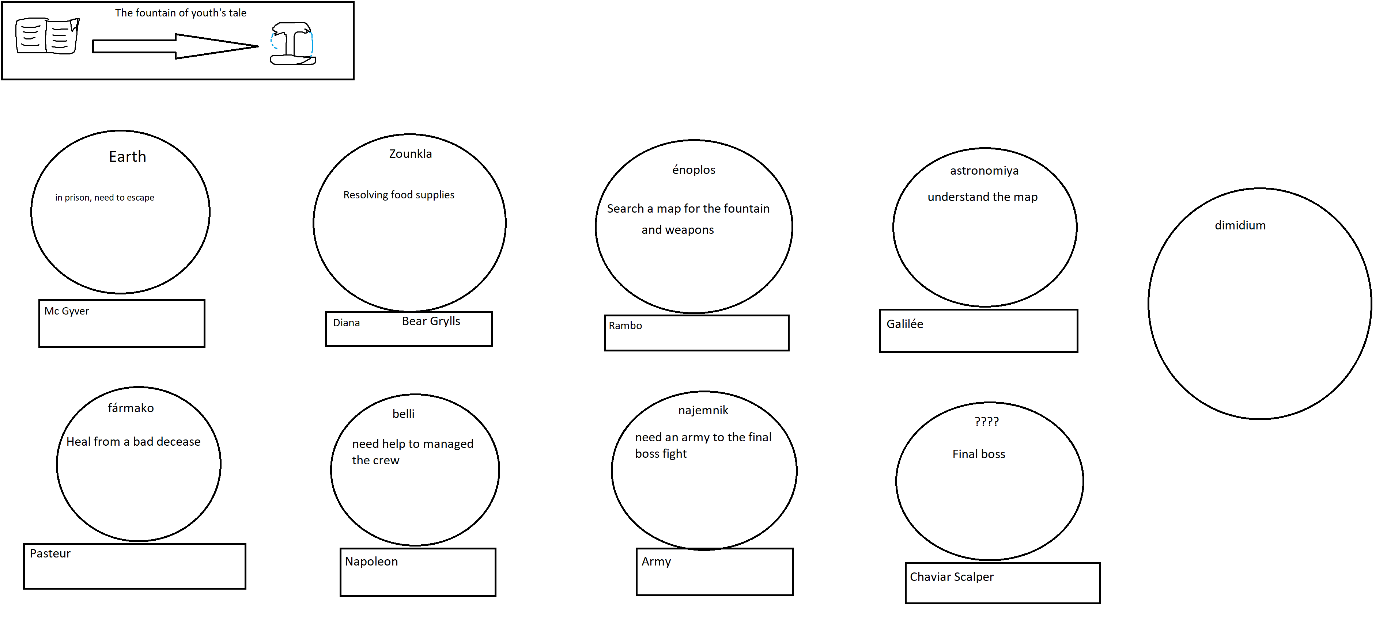
**World** : World est l’autre grosse classe de notre projet. C’est elle qui va regrouper les différentes planètes et qui permet le fonctionnement du jeu. World contient donc une liste de planètes. Cette liste étant actualisé à chaque fois que le joueur découvre une nouvelle planète.

Elle contient aussi la transition pour les autres planètes permettant une meilleure harmonie dans le jeu.

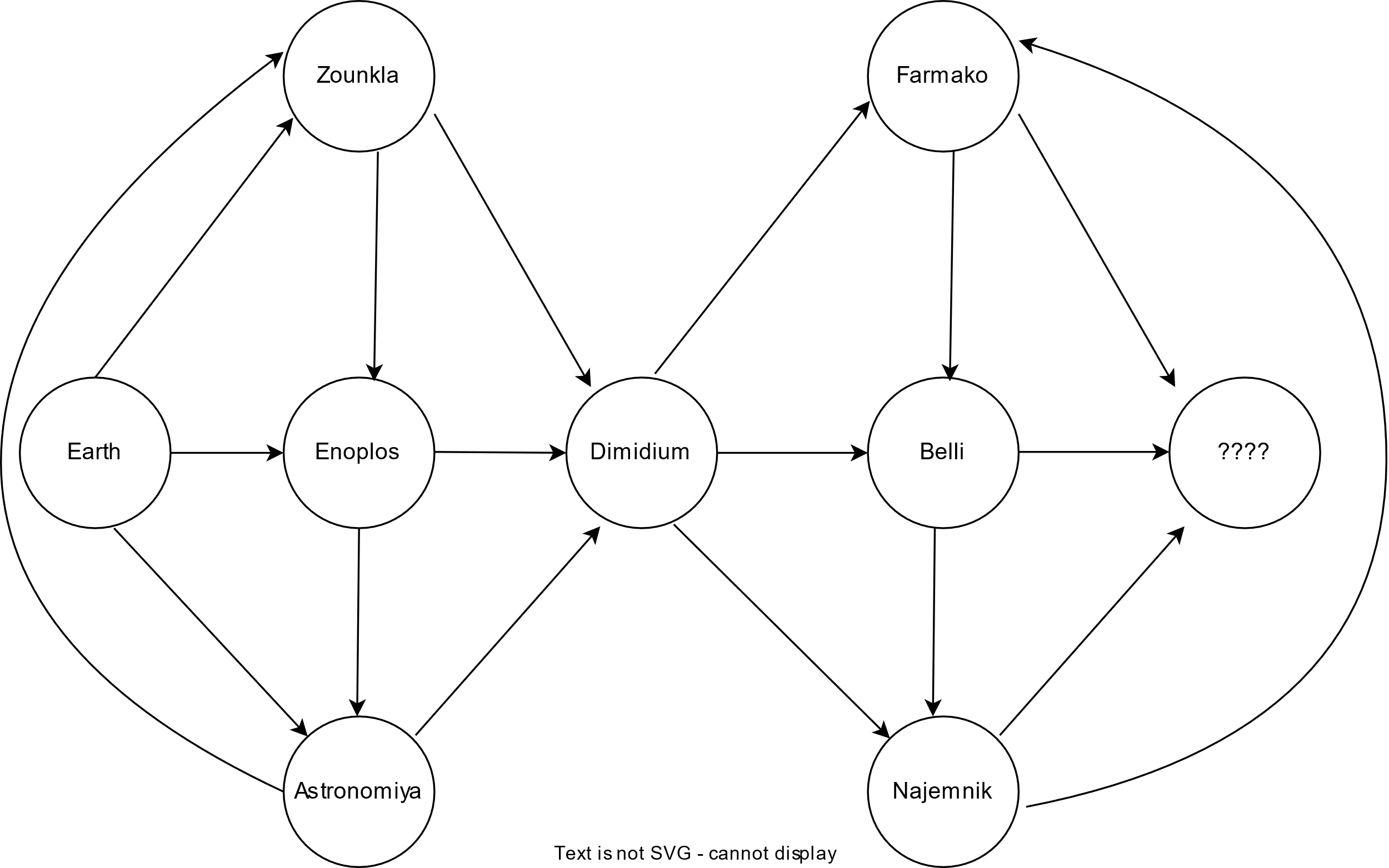
**Le diagramme de séquences :**

Lors des premières heures de réflexion, nous avons tout de suite compris qu’il fallait une base à notre projet ou une histoire. Ainsi, nous avons réalisé par l’intermédiaire de draw.io notre diagramme de séquence.

En premier lieu, nous avons réunis nos idées sur ce qu’il allait se passer sur chaque planète et les objectifs à atteindre. Ainsi, voilà comment se présente cette première partie.



Après cette première partie, nous avons ensuite fait le lien entre ces différentes planètes avons donc accompli notre diagramme de séquence :



L’histoire commence sur la planète Earth, ensuite, le protagoniste aura 3 choix qui s’offrent à lui :

* Aller sur la planète Zounkla
* Aller sur la planète Enoplos
* Aller sur la planète Astronomyia

Le 1er choix n’est pas de grande importance, cependant, nous voulons que le joueur passe obligatoirement par ces 3 planètes. S’il commencent par Zounkla, il devra suivre par Enoplos afin de pas perdre ensuite Astronomiya ect…

Nous lui laissons cependant le choix de directement partir sur Dimidium de n’importe quel de ces 3 planètes.

Si le joueur ne va pas sur ces 3 planètes, alors il lui manquera des items nécessaire sur Dimidium et perdra ainsi le jeu.

Dimidium est une planète intermédiaire qui permet ainsi de vérifier que le joueur est bien passée par ces 3 planètes.

Après cela, le même schéma est employé pour Najemnik, Belli et Farmako. Cependant, le joueur n’est pas obligé de passer par ces 3 planètes afin de gagner. Ces planètes aident juste à recevoir l’aide nécessaire pour accomplir l’objectif final.

La dernière planète se nomme « ???? » et c’est là ou le personnage principal affrontera le boss final.

**Le diagramme d’état :**

Chaque planète est différente et contient différents choix à faire et différente récompense.

Voici les différents choix et récompenses pour chaque planète :

**Planet : « ???? »**

Chart, diagram

Description automatically generated

**Planet Astronomiya:**

Graphical user interface, application

Description automatically generated

**Planet : Belli**

Graphical user interface, diagram

Description automatically generated

**Planet: Earth**

Timeline

Description automatically generated with medium confidence

**Planet Enoplos:** Graphical user interface, application

Description automatically generated**Planet: Farmako**A screenshot of a computer

Description automatically generated with medium confidence

**Planet: Najemnik**

Graphical user interface, diagram

Description automatically generated

**Planet Zounkla:**

Diagram

Description automatically generated

**Repartition des taches :**

Afin de réaliser ce projet très conséquent, nous nous sommes réparti les tâches.

**En commun :**

* Narrative de l’histoire
* Etat séquentiel de l’histoire

**Dylan :**

* UML
* Packages :
  + planets
  + world
  + common :
    - enums
    - files
    - input
    - output

**Yon :**

* Classes :
  + Earth
  + LastPlanet
  + Dimidium

**Jimmy :**

* Rapport
* Classes :
  + NAJEMNIK
  + BELLI
  + FARMAKO
  + PLAYER

C’est ainsi que nous nous sommes réparti la tâche pendant dans notre travail. Nous avons collaboré et regarder les différentes méthodes que chacun de nous à réaliser afin de pouvoir rendre un travail propre et soigné.