QUARKS STRUGGLE

<========= Documentation Technique =========>

Table des matières

I – Ressources utilisées

La plupart des ressources utilisées dans le projet proviennent d'internet excepté pour les éléments (sprites) qui composent l'UI. La plupart des références peuvent être retrouvés en jeu dans les crédits.

La ressource la plus importante est Volumetric Lines. Téléchargeable gratuitement sur l'Asset Store, ce sont des lignes ressemblant aux lasers que l'ont peut observer dans quasiment tous les films et jeux de science-fiction.

II – Fonctionnement global du jeu

Je vais essayer ci-dessous d'expliquer le plus clairement possible comment fonctionnent certains points-clés du jeu.

A – Vaisseaux

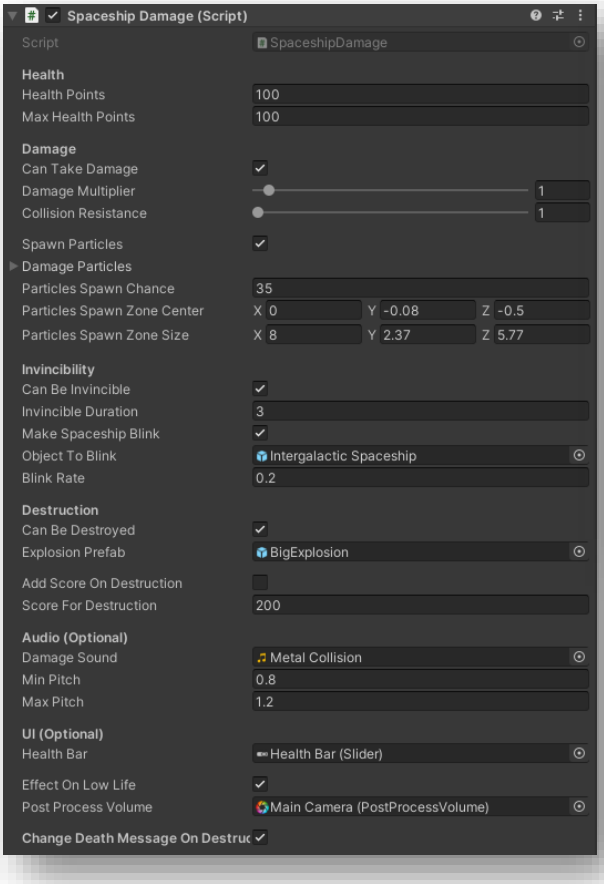
Aussi bien alliés qu'ennemis, tous les vaisseaux fonctionne sur le même principe. Tout commence par leur hiérarchie que voilà :

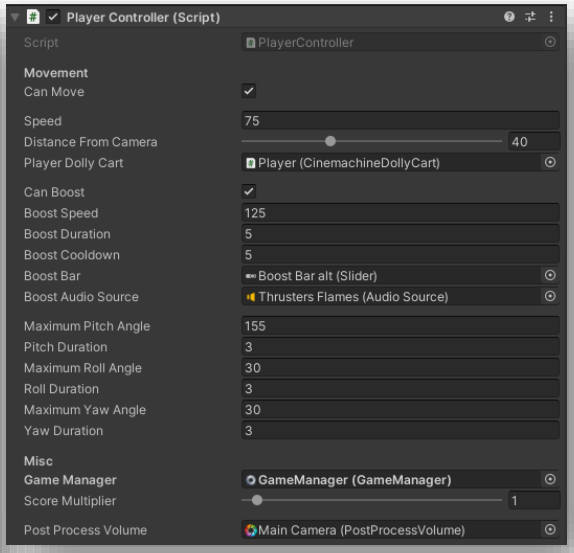
- NomVaisseauDollyCart[1]
  - NomVaisseau
    - Modèle
    - Weapons
    - Thrusters Flames
    - Stats Canvas[2]
  - Camera[2]

[1] : Seulement pour les vaisseaux qui suivent le rail.
[2] : Seulement pour le joueur.

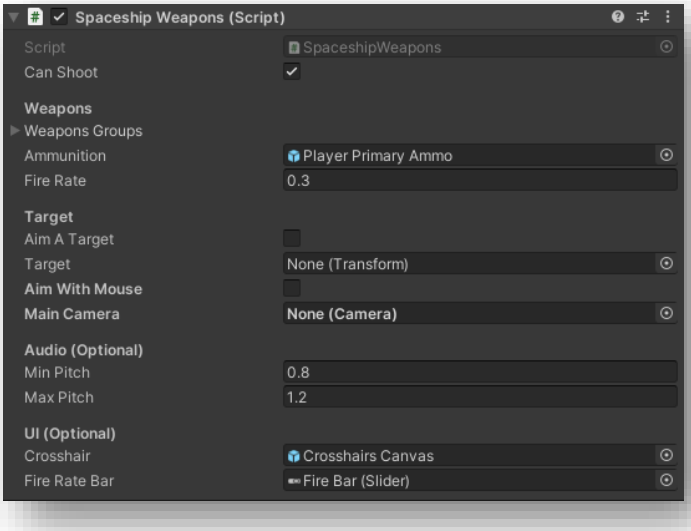
Pour les vaisseaux qui suivent le rail, nous retrouvons en premier lieu un GameObject avec un composant CinemachineDollyCart. En effet, le rail sur la carte est un CinemachineSmoothPath plus souvent utilisé pour animer des caméras.

Il y a ensuite un GameObject portant le nom du vaisseau (ici NomVaisseau). Celui-ci est principalement composé de Colliders, d'un Rigidbody, d'une AudioSource et de deux scripts. Le premier script s'occupe de la gestion des points de vie et des dégâts reçus. Il peut notamment faire apparaître des particules pour simuler des dégâts visibles sur le vaisseau, rendre le vaisseau invincible temporairement après un choc et détruire le vaisseau. Le second script est celui qui contrôle le vaisseau. Si c'est un ennemi, nous y verrons EnemiesIA. Si c'est le joueur alors ce sera PlayerController.





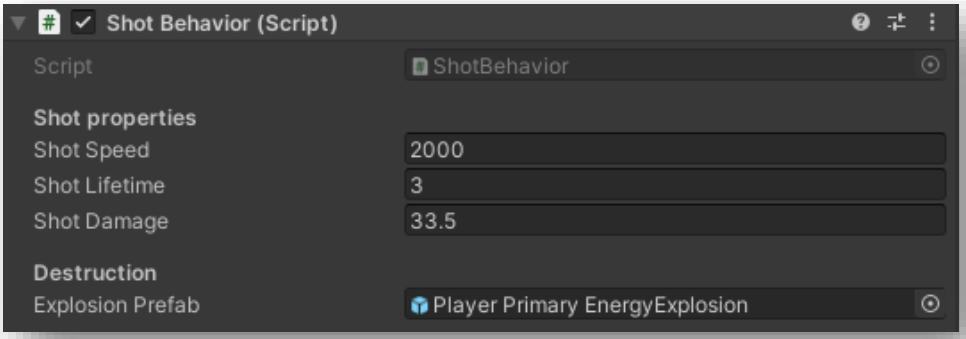
Ensuite, dans le dernier niveau de la hiérarchie, nous y trouvons trois GameObject. Le premier est simplement le modèle du vaisseau. Il est suivi de Weapons avec le script SpaceshipWeapons. Comme son nom l'indique, ce dernier gère les armes du vaisseau. Cependant, ce n'est pas ce script qui donne l'ordre de faire feu. Cette tâche revient au contrôleur du vaisseau.



Enfin, le dernier GameObject se nomme Thruster Flames. Ce n'est rien d'autre que l'effet de particule qui simule la flamme du propulseur.

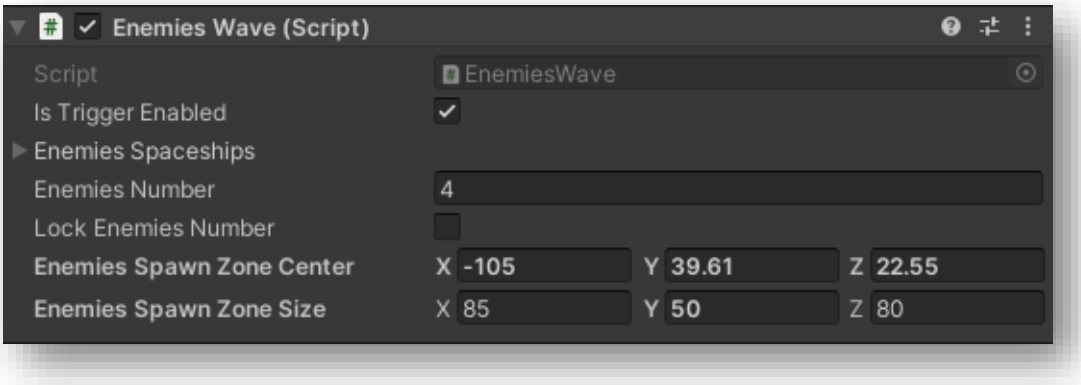
B – Lasers

Tous les lasers tirés par les vaisseaux proviennent de la même base, Volumetric Lines. Le prefab fourni a été dupliqué et modifié pour correspondre à chaque utilisation. Des lasers bleus pour le joueur, des rouges pour les ennemis. Pour correspondre à ce que l'on attend du projet, j'ai modifié le script qui déplace le laser. Il n'utilise plus transform.position pour se déplacer mais AddForce() pour éviter qu'il passe au travers des GameObject qu'il touche. J'y ai aussi fait en sorte qu'il se détruise à la collision ou après un certain temps. En plus de rendre le tout plus réaliste, cela permet de nettoyer la scène des lasers qui ne sont plus utiles.



C – Vagues d'ennemis

Tout au long de son parcours, le joueur sera confronté à des vagues d'ennemis. Ces vagues apparaissent en passant dans des triggers placés sur le chemin. C'est alors qu'un nombre défini d'ennemis est instancié. Actuellement, les ennemis apparaissent juste devant le joueur. Il est possible d'imaginer ajouter une animation qui se joue à l'apparition qui fait descendre progressivement l'ennemi jusqu'à sa position.



D – Niveaux de difficultés

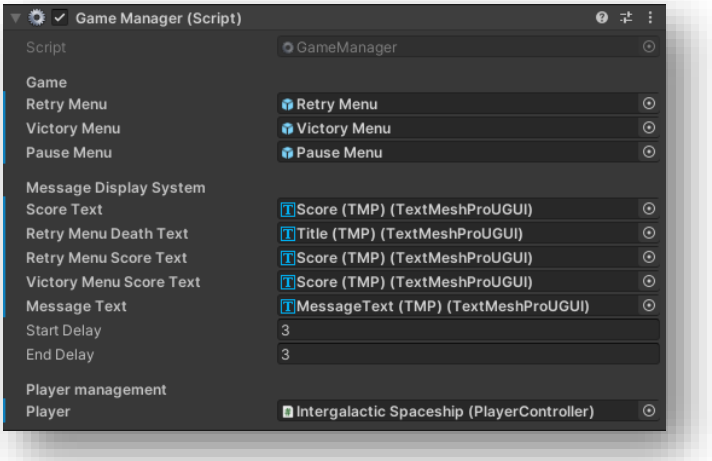
Depuis le menu principal, le joueur a la possibilité de sélection une difficulté (Facile, Moyen, Difficile). Le niveau de difficultés influe les caractéristiques suivantes :

- Point de vie du joueur :
  - Facile : vie doublée
  - Difficile : vie réduite de moitié
- Point de vie des ennemis (Boss inclus) :
  - Facile : divisé par deux
  - Difficile : doublé
- Nombre d'ennemis :
  - Facile : divisé par deux
  - Difficile : doublé
- Score final :
  - Facile : aucun changement
  - Normal : doublé
  - Difficile : triplé.

Il est aussi possible d'altérer la précision des ennemis en utilisant Random.insideUnitSphere. Nous pourrions donc avoir une zone plus grande en mode facile et plus petite en difficile.

E – Déroulement de la partie

Tout le déroulement de la partie (démarrage, boucle, fin) est géré par le GameManager. À l'aide de coroutines, le GameManager affiche ou cache des éléments à l'écran, déclenche la fin de la partie si le joueur perd, etc.



F – Musique

Enfin, la musique est gérée par un GameObject dédié et le script MusicPlayer. Il y est possible d'y avoir plusieurs listes de lectures et de la changer à tout moment. Ce script se charge automatiquement de relancer une nouvelle musique à la fin de la précédente.