Définir l'environnement de simulation

Besoins non-fonctionnels

Performances:

La simulation doit être assez légère pour pouvoir entrainer IA
Pour l'entraînement on doit accélérer le temps (pas 10 min en dur pour un match)
Doit se lancer sur un ordinateur du crémi.
Il faut pouvoir lancer 1000 matchs en une journée
Paramétrer la simulation avec différentes règles (taille du terrain, nombre de joueurs, catégorie de robots, caractéristique de la balle, hauteur de l'herbe sur le terrain, fautes)

Besoins fonctionnels

Simuler des parties de foot : 2 équipes s'affrontent, elles doivent marquer des buts dans les cages ennemies (règles foot classique pour les but et l'engagement). On doit avoir au moins un gardien dans chaque équipe. Le match est découpé en 2 mi-temps de 5 min. Un but vaut 1 point au score et chaque joueur présent sur le terrain gagne un point si leur équipe marque(récompense pour l'apprentissage). **->**Simulation avec Python & la bibliothèques Gym
Héberger sur git
Exécutable depuis le terminal
Jouable au clavier jusqu'à deux joueurs
Obtenir des statistiques (agressivité, possession balle, score) pour évaluer les stratégies apprises par les IA.

Sous Besoins fonctionnels - Implémentation

Gérer les paramètres:

L'utilisateur peut choisir les paramètres de la partie.

```
Class Rules()
```

attributs: tableau_de_dimensions terrainSize, int NbPlayerEquipe, tableau_de_dimensions balleSize, bool Fautes,bool grass, ball_weight, Robot robotType
méthodes: des getter

Simuler un match:

Cette classe organise les match et ...

```
Class Partie(Rules myRules)
```

attributs: terrain, équipe droite, équipe gauche, score,chrono,balle
méthodes: réinitialiser, initialiser, étape,afficher, dernier_joueur_touché_balle

***Simuler le Terrain*:**

Implémentation d'un terrain de foot aux normes de la Robocup.
reproduire le terrain suivant les règles du jeu avec des variables facilement modifiable
la classe terrain serait un agrégat, les éléments qui forment le terrain (surface de réparation, cage, zone engagement …) seraient des attributs privés de cette classe (VO?).
+hauteur herbe

```
Class Terrain(RuleSetTerrain)
```

simule le terrain de jeu

attributs: tableau_de_dimensions, liste_obstacles
méthodes: afficher

Equipe de joueurs
Permettre à la classe match de donner ke score à une équipe plutot qu'a un joueur.
`Class Equipe( list<Agent>)`
attributs: liste<Agent()>

gérer les interactions entre les joueurs du match (déplacements, tirs, rôles
Joueurs :
position , direction , vitesse, numéro, rôle (gardien, sur le terrain), type de robot (small,kid,adult), hériter des méthodes de déplacement selon le type de robot.
entity
`Class Joueur(Robot Kid)`
simule un joueur
attributs: position, vitesse ,taille, couleur, orientation_initiale, orientation, vitesse_désiré, score,role,numéro,robot
méthodes: afficher, déplacer, tirer

avoir différents types de robots (différents caractéristiques pour chaque types ex : taille, forces, vitesse ...etc)
classe robot : + classes filles small,kid,adult
taille, poids, bornes sup vitesse déplacement, borne sup/inf force de frappe, *champ de vision(à voir)*
`Class Robot()`
attributs: int taille, int poids, int vitesse déplacement, borne_int force frappe,
méthodes: différent type de tir en fonction ?

simuler la balle du match de foot:
description...
`Class Ballon()`
simule un ballon 2D
attributs: position, vitesse ,taille, poids,couleur,rebonds,friction, accélération,direction
méthodes: afficher, déplacer , accélérer, collision, rebondir.

actions :
tirer la balle, se tourner, avancer reculer

tirer_la_balle:
direction de tir, force de frappe, destination de la balle .