Système multi-agents

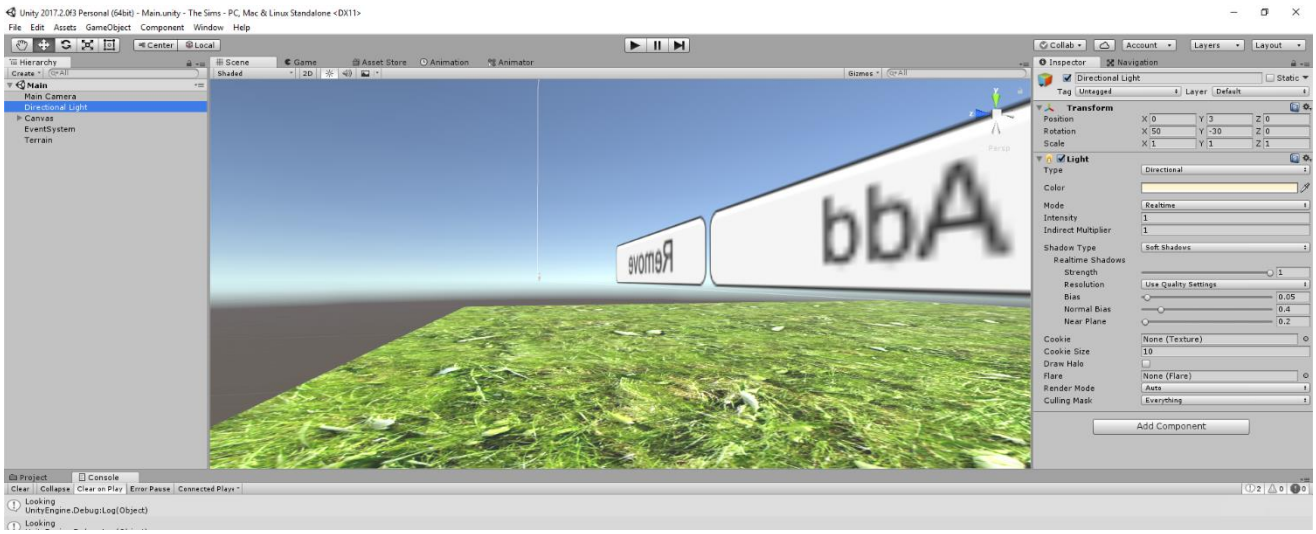
Architecture logicielle et qualité

NEBOIT Jean-Charles – COURNUT Jules

Introduction

Pour ce projet de réalisation d'un système multi-agent nous avons choisi de représenter un environnement 3D dans lequel évoluerait des agents humains dont le but serait de communiquer entre eux. Chacun de ces agents possède un caractère propre qui entrainera un comportement différent. En effet, il existe dans notre système 3 types d'agent : le charismatique, l'amical et le timide. Chaque agent possède une jauge sociale qui se vide au cours du temps et communiquer entre eux permet de faire remonter cette jauge. Leur différence de comportement s'exprime par une nécessité de communiquer qui arrive plus ou moins tôt selon leur caractère. Ainsi, le charismatique aura « envie » de parler plus souvent que le timide qui lui laissera sa jauge sociale se vider. La simulation consistera donc à regarder l'évolution de plusieurs agents qui auront des caractères choisis au hasard. Au sein de chaque caractère, il y a aussi un système d'aléatoire qui fait que même les agents de même caractère peuvent avoir un comportement différent.

Pour réaliser cette simulation nous avons utilisé le moteur de jeu Unity car il nous permet de réaliser un environnement 3D relativement facilement et qu'il intègre aussi un système de navigation avec le Navigation Mesh qui nous a donc servis de système de coordonnées en 3 dimensions. Pour la gestion de version nous avons utilisé Github.



Conception

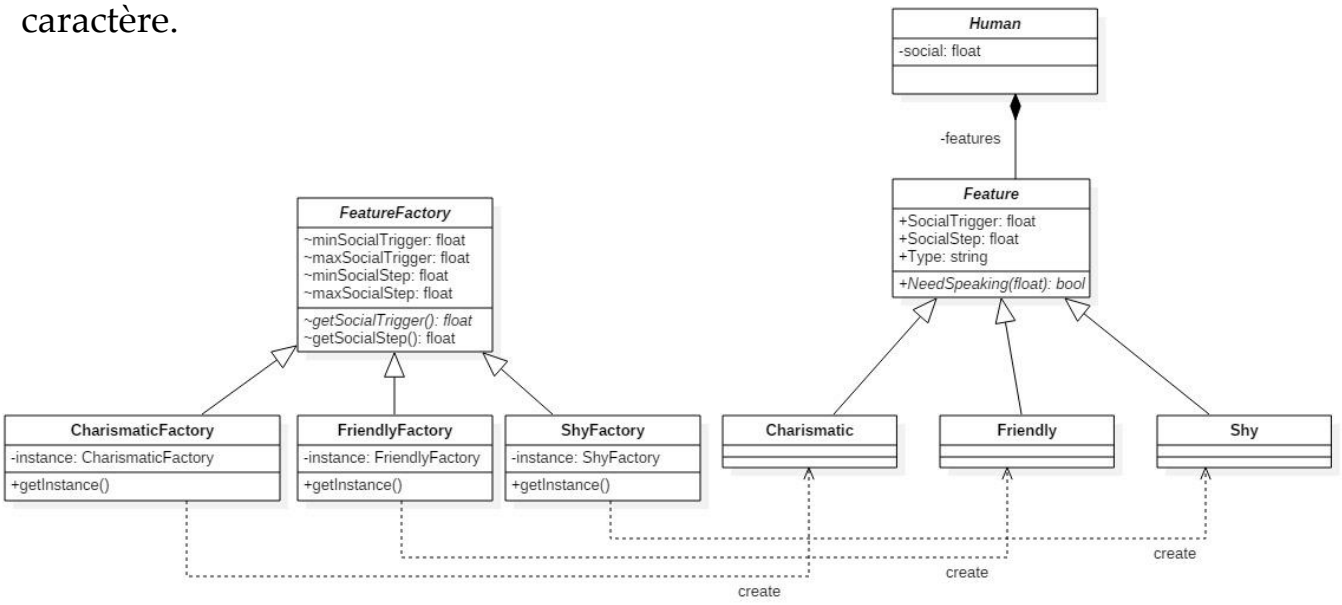
Lors de la conception, nous avons dû mettre en place 3 patrons de conception. En effet, plusieurs problèmes se sont présentés :

- Nous devions pouvoir créer un agent sans renseigner son caractère.

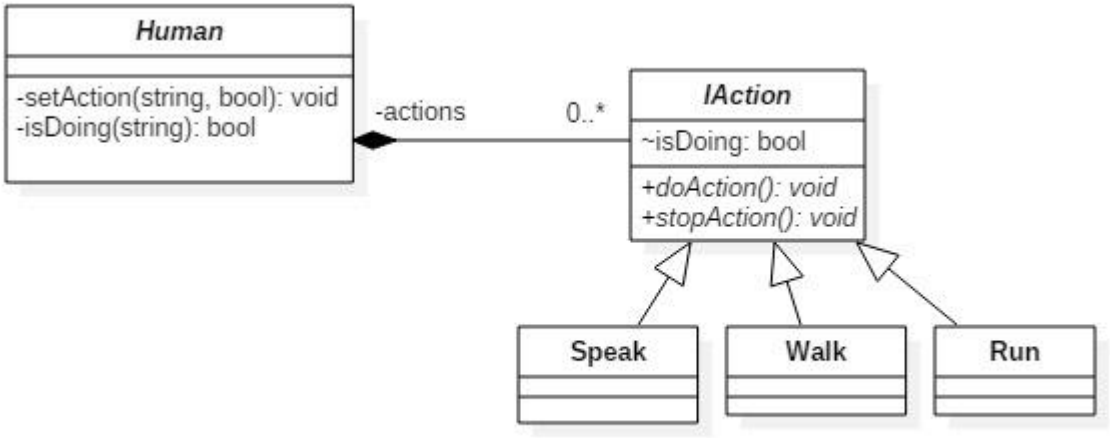
- Chacune des actions faisables par chaque agent (parler, courir, marcher, …) devait pouvoir être interchangeable dans leurs utilisations et devait changer dynamiquement.

- L'outil de visualisation devait pouvoir suivre l'évolution de la simulation en étant averti à chaque changement.

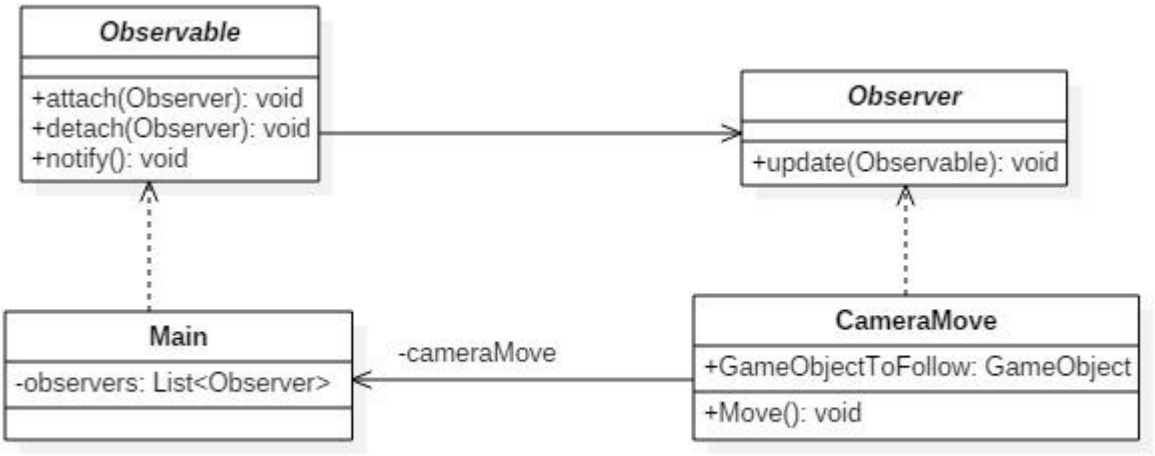
Pour le premier problème qui était un problème de création nous avons mis en place une Fabrique simple qui aura pour responsabilité le choix du caractère de l'agent créer ainsi que le système d'aléatoire. En effet au sein d'un caractère, chaque agent a un Social Trigger qui représente le seuil à partir duquel l'agent a envie de parler. Ce seuil est choisi aléatoirement lors de la création à l'intérieur d'une fourchette propre à chaque caractère. On choisit aussi un Social Step qui est la manière dont la jauge diminue au cours du temps, plus ou moins rapidement. Ici aussi il est choisi aléatoirement selon le caractère.



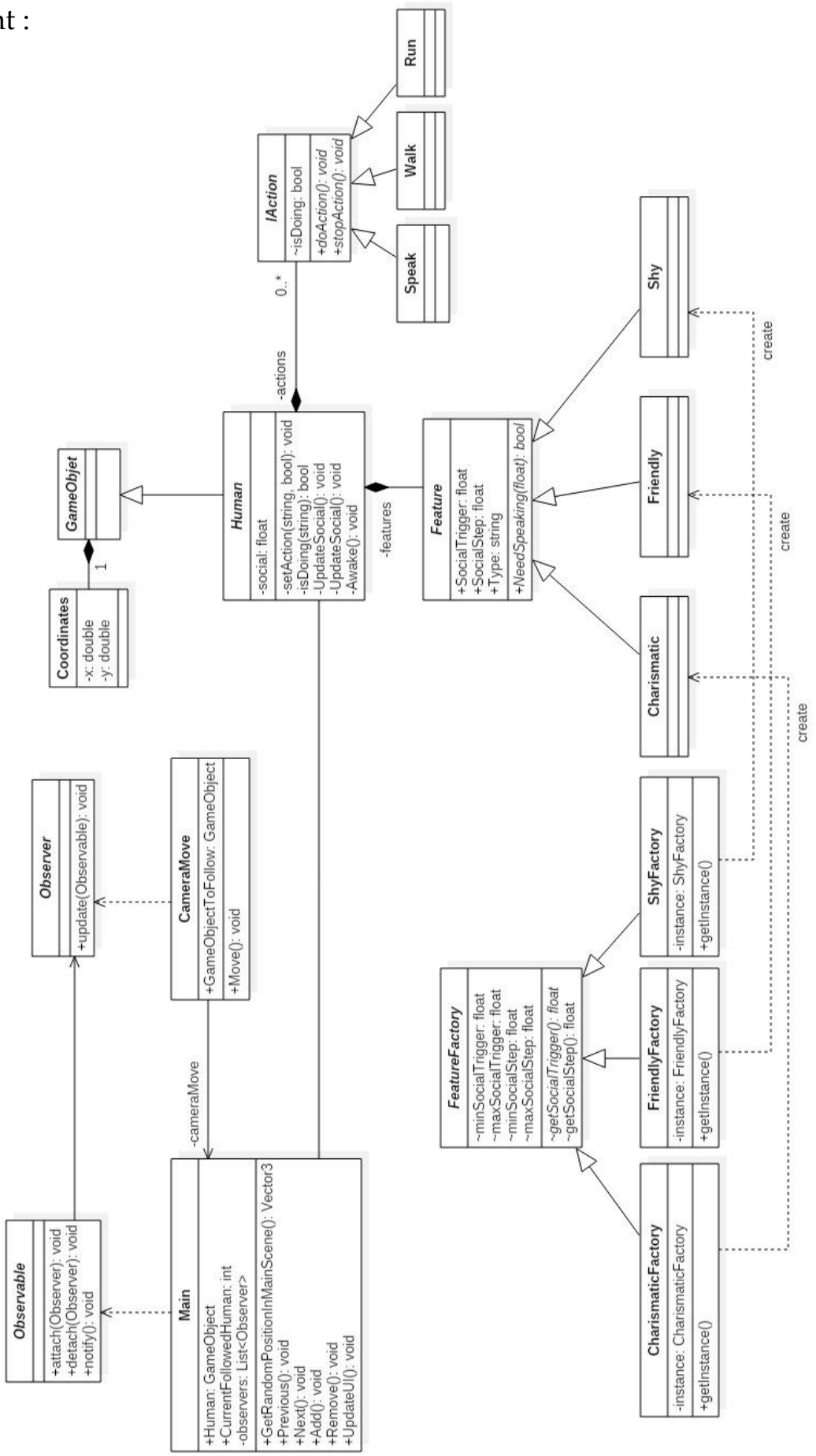
Pour représenter les actions et résoudre le second problème nous avons utilisé le patron État car chaque action dépend de l'état de la valeur de la jauge sociale.



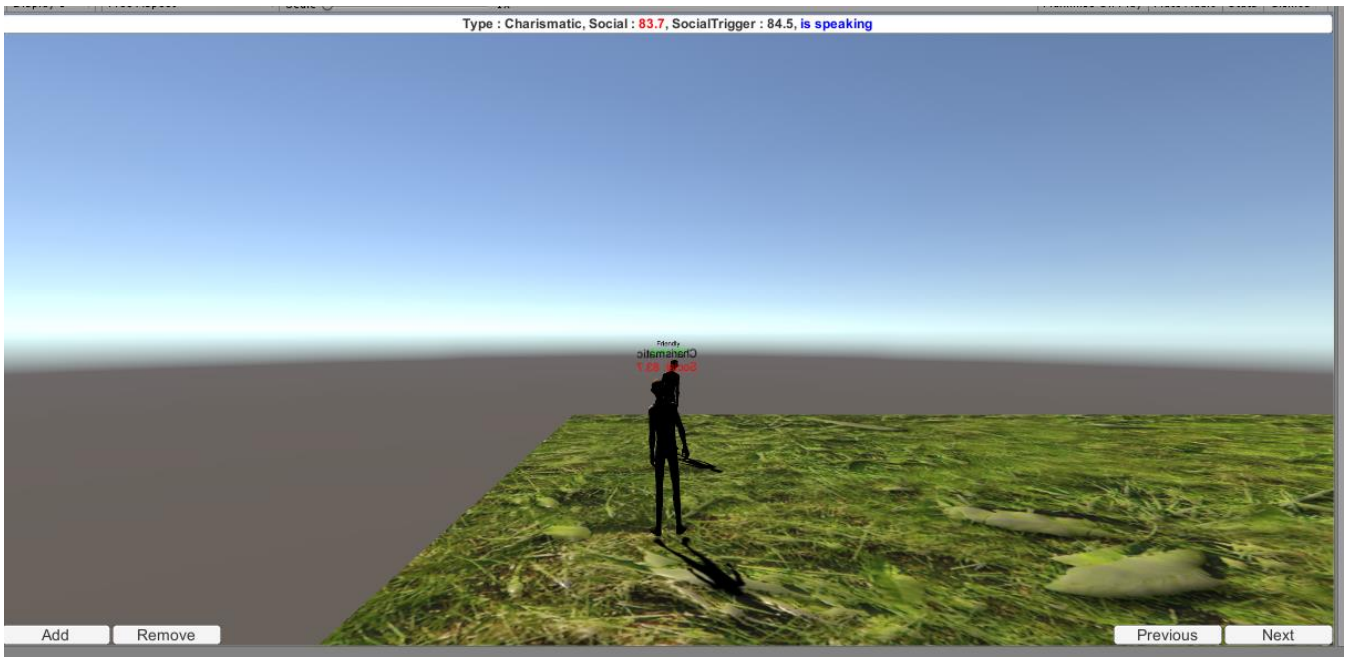
Pour le dernier problème nous avons dû mettre en œuvre un patron Observateur. Nous avons appliqué une interface observable au programme principal de la simulation et nous avons créé une classe CameraMove qui représente les mouvements de la camera qui sera donc notifié à chaque changement au sein de la simulation.



Le diagramme représentant donc le programme en entier est donc le suivant :



Résultats



Notre interface se présente donc ainsi avec quatre boutons :

- Add qui permet d'ajouter un agent d'un caractère aléatoire

- Remove qui permet de supprimer l'agent courant c'est-à-dire celui que l'on regarde

- Previous et Next qui permet de changer d'agent courant

En haut de l'interface on a un bandereau de texte qui indique le type de l'agent courant ainsi que son niveau Social, son Social Trigger et son Social Step.

Quand un agent doit parler il rejoint l'agent le plus proche et parle jusqu'à atteindre les 100 points de social. Un état « is Speaking » apparait dans le bandereau lors de l'échange. Cependant, un léger disfonctionnement fait que lorsque un des deux agent a « fini » de parler c'est-à-dire que son social est à 100, il abandonne l'autre agent le laissant « parler tout seul » en quelque sorte.