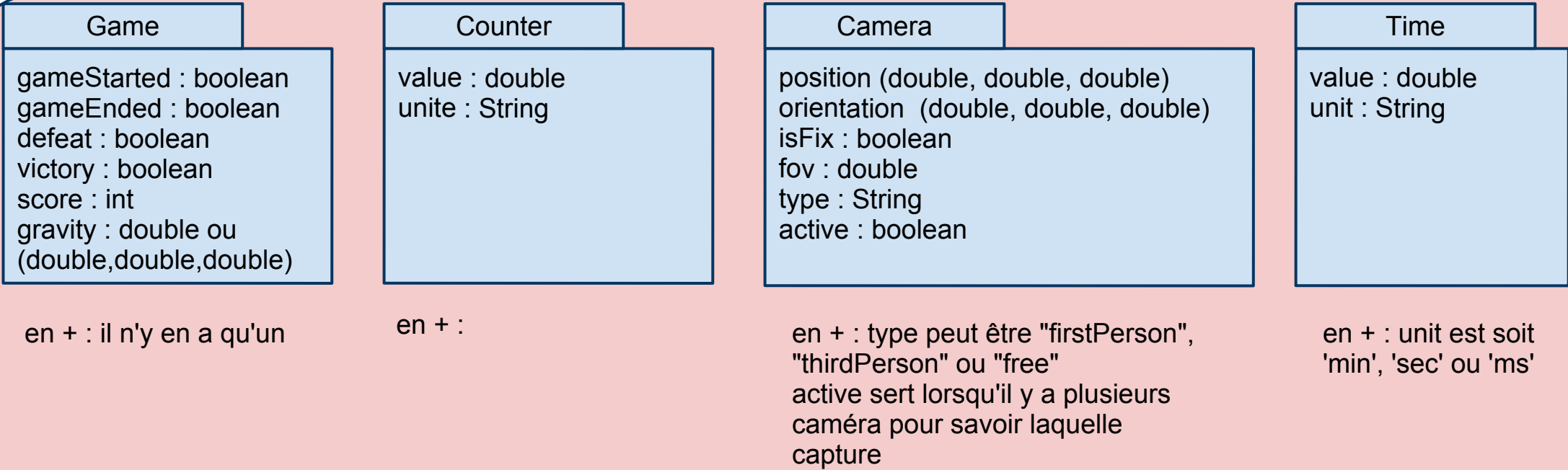


note : l'utilisateur pourra créer de nouveaux types qui seront soit hérités de Object, soit d'une classe héritée de Object en Javascript, les types ne sont pas définis mais je les ai mis pour plus de clarté

La grammaire haut niveau permet l'héritage multiple. Il reste à savoir si ce sera implémentable. A priori oui puisque la notion d'héritage n'apparaît pas dans la grammaire bas niveau (les attributs seront des ressources et des paramètres d'entités, il suffira de les créer les uns à la suite des autres en veillant bien à ne pas mettre des doublons).  
ex :  
maison is construction and breakable

type = classe

Autres mots-clé représentant des classes :



- list (un groupe d'élément ordonné, sert à l'utilisateur pour ne pas avoir à rappeler chaque objets, ex : armee is list of 10 character)  
- player | ally | enemy | neutral (correspond à ne rien préciser)  
- multiple (à la différence d'une liste où on ne pourra pas enlever ou ajouter d'éléments hors déclaration, signifie qu'on peut générer plusieurs fois cet élément pendant le jeu, par contre on ne peut pas accéder aux éléments en particulier, sauf dans les règles du jeu, ex : jangoFett is character enemy multiple), les types sont multiple par définition, ainsi que les enemy, ally et neutral pendant le jeu : generate 10 jangoFett in zoneGenerationClones  
generate 5 obstacle on circuit1  
si on ne définit pas multiple, il ne sera pas possible de générer plusieurs objets.

generate in zone / on objet / at coordonnées, pour générer avec une position aléatoire avec une restriction  
- in : à l'intérieur d'une zone en veillant à ce qu'il n'y ait aucune collision lors de la génération  
- on : juste au dessus de l'objet en veillant à ce qu'il n'y ait aucune collision lors de la génération  
- at : génère le premier objet aux coordonnées indiquées et les suivants autour de lui en veillant à ce qu'il n'y ait aucune collision lors de la génération

