POLYTECHNIQUE
MONTRÉAL

LE GÉNIE
EN PREMIÈRE CLASSE

LOG8715 - Architecture de jeux vidéo

Rapport TP1

Emmanuelle Roberge - 1849263

Gabriel Paquette - 1899088

16 février 2021

Structure des Composantes

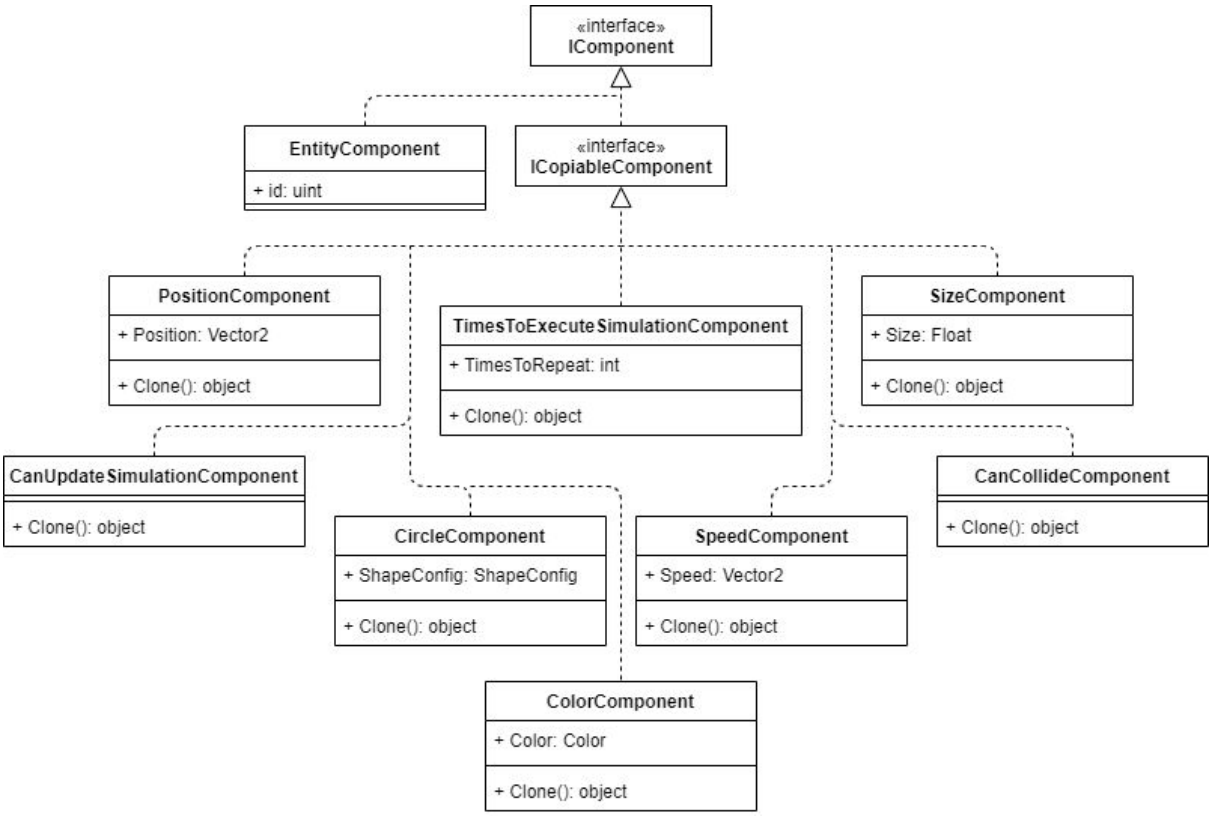

Figure 1: Diagramme des composantes

Une entité est définie par les composantes de la figure 1 (PositionComponent, CanUpdateSimulationComponent, TimesToExecuteSimulationComponent, CircleComponent, ColorComponent, SpeedComponent, SizeComponent et CanCollideComponent).

Nous utilisons le EntityComponent comme clé afin de classifier l'appartenance des composantes à chaque entité. Dans la classe World, nous avons un dictionnaire regroupant, selon le type de composantes, des collections de composantes de même types, dont l'accesseur qui est représenté par un EntityComponent.

Le nom des composantes est explicatif. Nous utilisons le pattern de "tag" pour les composantes commençant par "can", et utilisons l'existence de cette composante sur une entité pour déterminer comment l'entité est traitée.

La composante "TimesToExecuteSimulationComponent" est utilisée pour savoir combien de fois la simulation, c'est-à-dire la mise à jour de la position et des collisions, doit être faite par trame pour une entité.

Structure des Systèmes

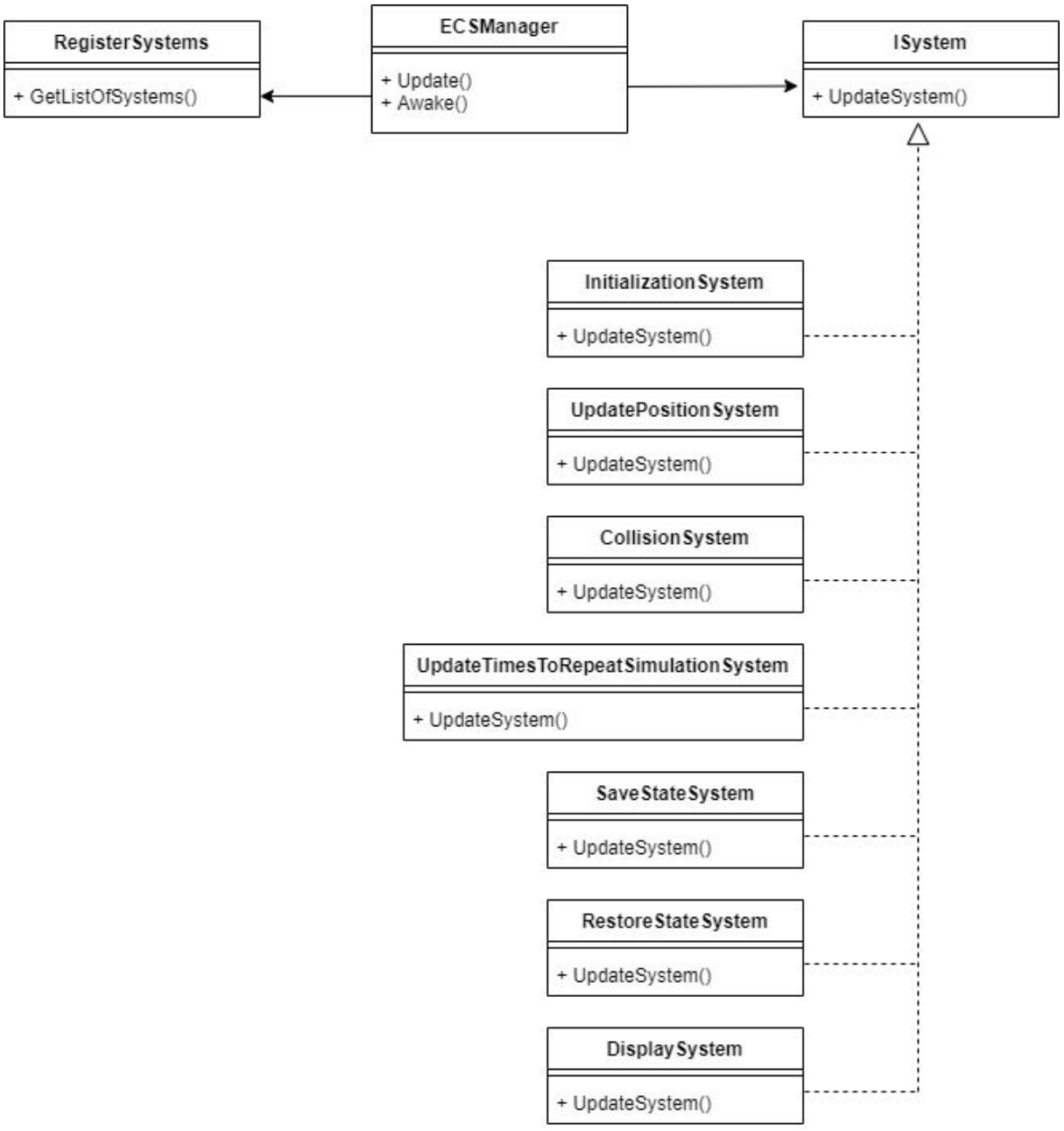

Figure 2: Diagramme des systèmes

Dans sa fonction Awake, le ECSManager appelle la fonction GetListOfSystems de la classe RegisterSystem qui sert à instancier tous les systèmes. UpdatePositionSystem, CollisionSystem et UpdateTimesToExecuteSystem seront instanciés en boucle (5 fois) pour permettre aux cercles de la moitié supérieure de l'écran d'être mis à jour à répétition.

À chaque trame, dans sa fonction Update, le ECSManager appelle la fonction UpdateSystem de tous les systèmes héritant de ISystem illustrés à la figure 2.

- **InitializationSystem** permet de créer les entités et leurs composantes de départ, grâce au fichier de Config, et ajoute un “CanUpdateSimulationComponent” aux entités.
- **UpdatePositionSystem** met à jour la position des cercles selon leur vitesse.
- **CollisionSystem** gère les collision des cercles avec les autres et avec les murs, tout en modifiant la couleur des cercles selon l’état résultant de la collision.
- **UpdateTimesToRepeatSimulationSystem** vérifie si le cercle est dans la moitié supérieure de l’écran, pour ensuite mettre à jour le nombre restant de simulations du déplacement et des collisions qu’un cercle doit faire.
- **SaveStateSystem** enregistre l’état de toutes les entités dans une liste en faisant une copie en profondeur de toutes les composantes présentes dans le dictionnaire de composantes. Ce système retire également tous les enregistrements plus vieux que deux secondes.
- **RestoreStateSystem** demande au World de rétablir l’état enregistré qui se rapproche le plus de deux secondes dans le passé par rapport au moment actuel.
- **DisplaySystem** appelle les différentes fonctions du ECSManager qui mettent à jour l’affichage des cercles.

Structure de la classe World

La classe World est un singleton servant à garder l’état de la simulation et des entités présentes et leurs composantes. Nous l’avons conçu pour pouvoir ajouter une composante de type quelconque à une entité sans avoir à modifier la classe World.

Cette classe conserve également des constantes, telles que le “timescale” qui représente le nombre de fois maximum qu’une entité peut être simulée par trame, ou encore les normales des murs composés par les bordures de l’écran. Il attribue le “id” unique à chaque entité lors de leur création et garde le compte des entités.