

## Projet PFA ECS - Pong

Antoine Lanco

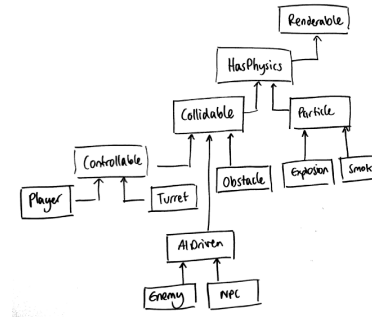
Universiter Paris Saclay

18 janvier 2021



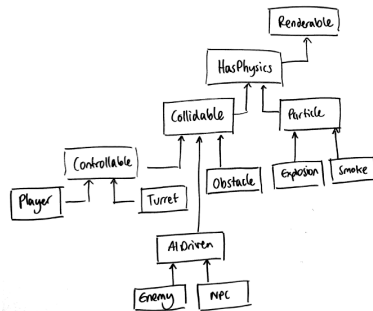
## Game Architecture

- L'architecture orientée objets est la plus répandue pour programmer des jeux, mais elle présente des inconvénients. Principalement les hiérarchies d'héritage peuvent devenir profondes et inflexibles.



## Game Architecture

- L'architecture orientée objets est la plus répandue pour programmer des jeux, mais elle présente des inconvénients. Principalement les hiérarchies d'héritage peuvent devenir profondes et inflexibles.

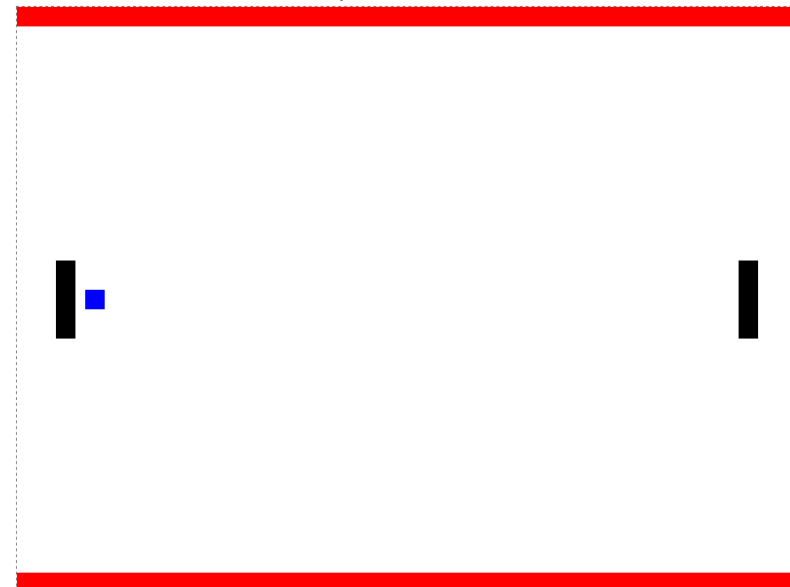


- L'architecture Entity Component System est une très bonne alternative à ce problème, car il n'y a plus de notion d'héritage. Néanmoins il faut quand même faire attention. Le manque de structure peut créer de nouveaux problèmes.



## Entity

Tous les éléments du jeu (visible, invisible, contrôlable ou non, ...)



## Component

- ▶ Toutes les propriétés que l'on peut attribuer à une entité.



## Component

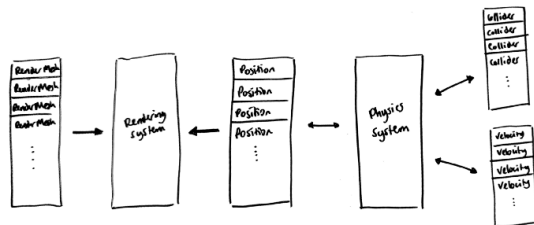
- ▶ Toutes les propriétés que l'on peut attribuer à une entité.



- ▶ Pour associer un composant et une entité, on doit pouvoir créer des tables Entité → Composant. En OCaml, on va vouloir donc une façon générique de créer des tables de hachage dont les clés sont des entités et les valeurs des composants (vecteurs, surface, ...).

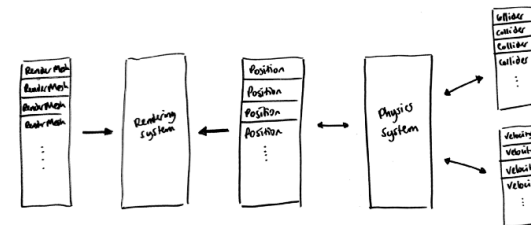
## System

- ▶ Les systèmes vont utiliser et/ou modifier les composants des entités. Par exemple le système "draw" va récupérer le composant "position" d'une entité pour pouvoir la dessiner.



## System

- ▶ Les systèmes vont utiliser et/ou modifier les composants des entités. Par exemple le système "draw" va récupérer le composant "position" d'une entité pour pouvoir la dessiner.



- ▶ Il faut faire attention, car si une entité appartient à un système qui a besoin de composant que l'entité n'a pas il va y avoir un problème.