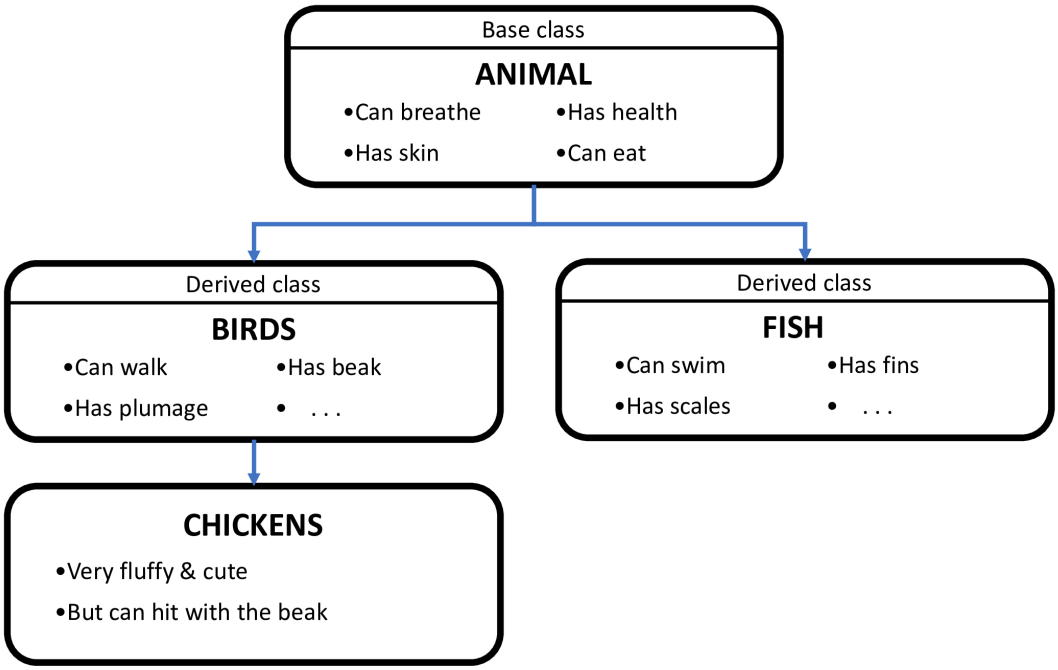
Introduction to ECS+

ECS+ is a new programming paradigm based on ECS and implemented in Apparatus.

Talking about OOP, we consider our practical task as multiplicity of special abstract things. In terms of UE4 these abstractions are named UObjects and over them we can apply such a principle like an inheritance. We create some main abstraction called „Base class“, define properties (i.e. variables and functions) and by deriving from that class we can create other abstractions which will get all the parent properties and define their own additional or distinctive ones. The most popular example in that way is the tree of animals inheritance, where the base class represent any animal exist, but the others (derived from the base one) represent separate animal sorts:



By using base class properties we can change current state of objects of derived class. But the problem is the game logic can become too scattered across the classes.

In game development there is such an approach named ECS. Apparatus provides all of the basic ECS idioms and even more. To be unambiguous the framework uses a different naming scheme as compared to classic ECS. Here is the list of analogous terms:

ECS Term	Apparatus Term
Entity	Subject
Component	Detail
System	Mechanic
A group of Systems	Mechanism
Archetype	Fingerprint
Chunk	Belt

Built-in Subjects

At a lower level, `Subjective` is an interface and any class that implements it may be called a `Subject`. There are several subjects already implemented in the framework:

- `SubjectiveActorComponent` (based on `ActorComponent`),
- `SubjectiveUserWidget` (based on `UserWidget`),
- `SubjectiveActor` (based on `Actor`).

Those should be sufficient for the most cases, but you can easily implement your own additional subject classes in C++.

Built-in Mechanisms

The `Mechanical` class is also an interface, with the most useful mechanisms already implemented:

- `MechanicalActor` (inherited from `Actor`),
- `MechanicalGameModeBase` (`GameModeBase`),
- `MechanicalGameMode` (`GameMode`).

You would hardly ever need to implement your own mechanisms. Should you wish to, you can also do it in C++.

From:
http://turbanov.ru/wiki/ - **Turbopedia**

Permanent link:
http://turbanov.ru/wiki/en/toolworks/docs/apparatus/ecs?rev=1617442875

Last update: **2021/04/03 09:41**