GameObjects and Components

The reason Unity has a GameObject and Component architecture is to decouple the different functionalities of objects in the game. Each object is likely to achieve its functionality in the game by touching a lot of different domains. For example, a ‘character’ from a FPS game needs to:

* Draw and animate itself (walking, crouching, etc) (Graphics)
* Move based on input (WASD, controller, etc) (Input)
* Play sounds (footsteps, gun shots, etc) (Audio)
* Collide with objects (Physics)
* AI

A single ‘Player’ class that contains all of this functionality would be burdensome and unwieldy: it’s doing too many things at once. An inheritance hierarchy is also unsuited to the task because not everything in the hierarchy is going to want all of the functionality. You don’t want to force child objects to sign up for functionality that they don’t need.

Components solve this issue by separating functionality into pick-n-mix packages that are much more digestible. In the above example, each point would be its own component, e.g. a Drawing / Renderering component, an AudioPlayer component, a Collider component etc. Each object in the game signs up for the components that it needs to achieve the functionality it wants.

In this way, a game object becomes just a shell that holds a collection of components. The shell presents ways to access its components but the core game functionality comes from its selection of components and the communication between them.