Basic Components

Describes the most common types of Components in Unreal Engine and where you can learn more about them.

A **Component** is a piece of functionality that can be added to an Actor.

When you add a Component to an Actor, the Actor can use the functionality that the Component provides. For example:

- A Spot Light Component will make your Actor emit light like a spot light.
- A Rotating Movement Component will make your Actor spin around.
- An Audio Component will give your Actor the ability to play sounds.

Unlike Actors, Components can't exist by themselves. They must be attached to an Actor.

When adding Components to an Actor, you are putting together the bits and pieces that make up that Actor as a whole. For example, a car (Actor) consists of its wheels, steering wheel, body, lights, etc. (Components). Similarly, an Actor that represents a player character might have individual Components for the Skeletal Mesh (the character's visual representation), the Camera that follows the character around, and the Controller that receives input from the player.

After you put together the different Components that make up the Actor, you can place the Actor in your Level even without providing any **Blueprint** script (or C++ code) that dictates how the Actor should function. In keeping with the example above, a car (Actor) can exist as an object in the world (Level) without a driver (Blueprint or C++ code) telling it what to do. Blueprint or C++ code can then be used to access each of the car's Components individually (for example, if the gas pedal Component has been pressed, Blueprint logic could make the car speed up).

Component Instancing

Contrary to the default behavior of sub-objects in general, Components created as sub-objects within an Actor are instanced, meaning each Actor instance of a particular class gets its own unique instances of the Components.

A simple way to visualize this is to imagine the car from the example above above. A Car class might use Components to represent the wheels of the car. Four Wheel Components would be created as sub-objects in the default properties of the class and assigned to a "Wheels" array. When a new Car instance is created, new instances of the Wheel Components are created specifically for that particular Car. If this were not the case, when one Car in the world moved, the wheels of all Cars would turn; which is clearly not the desired behavior. Instancing of Components by default simplifies the process of quickly adding unique sub-objects to Actors.

Without Component instancing, all Component variables would need to be declared using the `Instanced` property specifier.