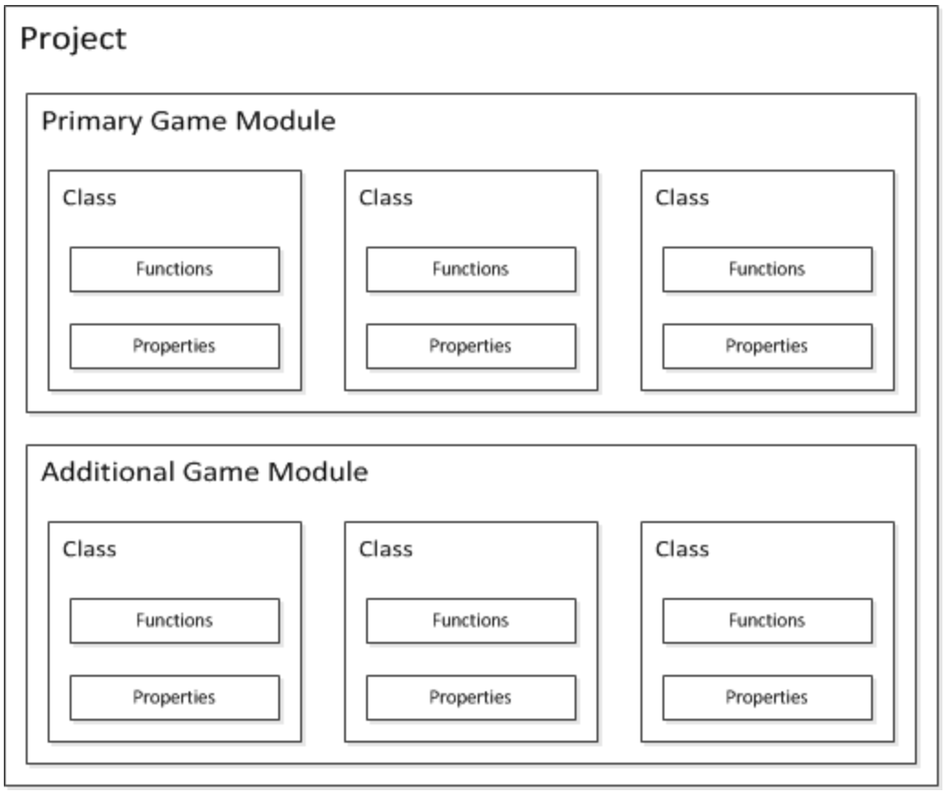
Gameplay Architecture

Reference for creating and implementing gameplay classes.

When programming gameplay elements using C++ code, each module can contain many C++ classes.



Each class defines a template for a new Actor or Object. Within the class header file, the class and any class functions and properties are declared. Classes can also contain structs, data structures that help with organization and manipulation of related properties. Structures can also be defined on their own. Interfaces allow additional gameplay behavior to be implemented by different classes.

When programming with Unreal Engine, it is possible to have standard C++ classes, functions, and variables. These can be defined using standard C++ syntax. However, `UCLASS()`, `UFUNCTION()`, and `UPROPERTY()` macros can be used to make Unreal Engine aware of the new classes, functions, and variables. For instance, a variable with a declaration prefaced by a `UPROPERTY()` macro can be garbage collected by the engine, and can be

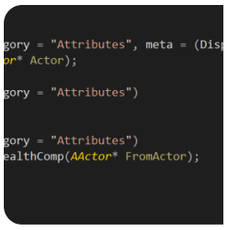
displayed and edited within Unreal Editor. There are also `UINTERFACE()` and `USTRUCT()` macros, and keywords for each macro that can be used to specify the behavior of the class, function, property, interface, or struct within Unreal Engine and Unreal Editor.

In addition to the above macros, there is a UPARAM() macro that is primarily used when exposing C++ code to Blueprints. To see examples of UPARAM() being used, see the Exposing Gameplay Elements to Blueprints documentation.

Gameplay Programming Reference Directory

Gameplay Classes

Reference for creating and implementing gameplay classes.

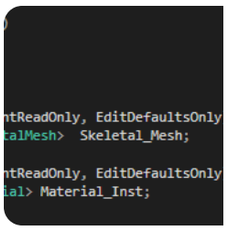


UFunctions

Overview for creating and implementing functions for gameplay Classes

Properties

Reference for creating and implementing properties for gameplay classes.



Structs

Reference to creating and implementing structs for gameplay classes.

Unreal Interfaces

Create and implement Unreal Interfaces in C++ and Blueprints.