UObject Instance Creation

Methods of creating new instances of Objects in gameplay code.

NewObject

`NewObject()` is the simplest UObject factory method. It takes in an optional outer object and class and creates a new instance with an automatically generated name.

```
template< class T >
T* NewObject
(
UObject* Outer=(UObject*)GetTransientPackage(),
UClass* Class=T::StaticClass()
)

```

Copy full snippet

Parameter	Description
`Outer`	(Optional) A `UObject` to set as the Outer for the Object being created.
`Class`	(Optional) A `UClass` specifying the class of the Object to be created.
Return Value	A pointer to the spawned instance of the specified class.

NewNamedObject

`NewNamedObject()` expands on `NewObject()` by allowing a name for the new instance as well as Object Flags and a template object to be specified as an argument.

```
template< class TClass >
TClass* NewNamedObject
(
UObject* Outer,
FName Name,
EObjectFlags Flags = RF_NoFlags,
UObject const* Template=NULL
)

```

Copy full snippet

Parameter	Description
`Outer`	A `UObject` to set as the Outer for the Object being created.
`Name`	An `FName` to set as the `Name` for the new Object.
`Flags`	(Optional) An `FObjectFlags` enum value describing the new Object.
`Template`	(Optional) A `UObject` to use as a template when creating the new Object.
Return Value	A pointer to the spawned instance of the specified class.

ConstructObject

For complete flexibility, new instances of `UObjects` can be created using the `ConstructObject()` function. This function calls `StaticConstructObject()`, which allocates the Object, executes the `ClassConstructor`, and performs any initialization such as loading config properties, loading localized properties, and instancing components.

```
template< class T >
T* ConstructObject
(
UClass* Class,
UObject* Outer = (UObject*)GetTransientPackage(),
FName Name=NAME_None,
EObjectFlags SetFlags=RF_NoFlags,
UObject const* Template=NULL,
bool bCopyTransientsFromClassDefaults=false,
struct FObjectInstancingGraph* InstanceGraph=NULL
)

```

Copy full snippet

Parameter	Description
`Class`	A `UClass` specifying the class of the Object to be created.
`Outer`	(Optional) A `UObject` to set as the Outer for the Object being created.
`Name`	(Optional) An `FName` to set as the `Name` for the new Object.
`SetFlags`	(Optional) An `EObjectFlags` enum value describing the new Object.
`Template`	(Optional) A `UObject` to use as a template when creating the new Object.
`bCopyTransientsFromClassDefaults`	(Optional) A `bool` that determines whether to copy transient properties from the class default object instead of the passed-in archetype pointer. If `true`, the class default object's transients are used.
`FObjectInstancingGraph`	(Optional) An `FObjectInstancingGraph` struct that contains the mappings of instanced objects and components to their templates. Used when for instancing components owned by the new Object.
Return Value	A pointer to the spawned instance of the specified class.

Object Flags

The `EObjectFlags` enumeration is used to quickly and succinctly describe an Object. There are various flags to describe the type of Object, how it is handled by garbage collection, the stage the Object is at in its lifetime, etc. There are also special all or none masks and predefined groups of flags.

Flag	Value	Description
Object Type		
RF_Public	`0x00000001`	The Object is visible outside of the package it is contained within.

Flag	Value	Description
RF_Standalone	`0x00000002`	The Object is kept around for editing even if is not referenced by anything.
RF_Native	`0x00000004`	The Objectis native. This is only used for `UClass` objects.
RF_Transactional	`0x00000008`	The Object is transactional.
RF_ClassDefaultObject	`0x00000010`	The Object is the default object for its class, i.e. the default template that new instances of that class use when being created.
RF_ArchetypeObject	`0x00000020`	The Object is a template for another object. It is treated like a class default object.
RF_Transient	`0x00000040`	The Object is not saved to disk.
Garbage Collection		
RF_RootSet	`0x00000080`	The Object is not garbage collected even if it is not referenced by anything.
RF_IsLazyReferenced	`0x00000100`	The Object is referenced by a lazy pointer and requires additional cleanup upon deletion.
RF_Unreachable	`0x00000200`	The Object is not reachable on the object graph.
RF_TagGarbageTemp	`0x00000400`	The Object is marked for use by various utilities that use garbage collection. This flag is not interpreted by the garbage collector itself.
Object Lifetime		
RF_NeedLoad	`0x00000800`	The Object needs loading.

Flag	Value	Description
RF_AsyncLoading	`0x00001000`	The Object is being loaded asynchronously.
RF_NeedPostLoad	`0x00002000`	The Object needs to be post-loaded.
RF_NeedPostLoadSubobjects	`0x00004000`	The Object still needs to instance sub-objects and fix up serialized component references
RF_PendingKill	`0x00008000`	The Object is pending destruction. Marks the Object as being invalid for gameplay purposes, but still a valid Object.
RF_BeginDestroyed	`0x00010000`	The Object has had `BeginDestroy()` called on it.
RF_FinishDestroyed	`0x00020000`	The Object has had `FinishDestroy()` called on it.
Special Masks		
RF_AllFlags	`0x0003ffff`	The Object has all flags. Used mainly for error checking.
RF_NoFlags	`0x00000000`	The Object has no flags. Used to avoid a cast.
Predefined Groups		
RF_Load	`RF_Public | RF_Standalone | RF_Native | RF_Transactional | RF_ClassDefaultObject | RF_ArchetypeObject`	Flags loaded from Unreal files.
RF_PropagateToSubobjects	`RF_Public | RF_ArchetypeObject | RF_Transactional`	Flags inherited by sub-objects from their super-objects.