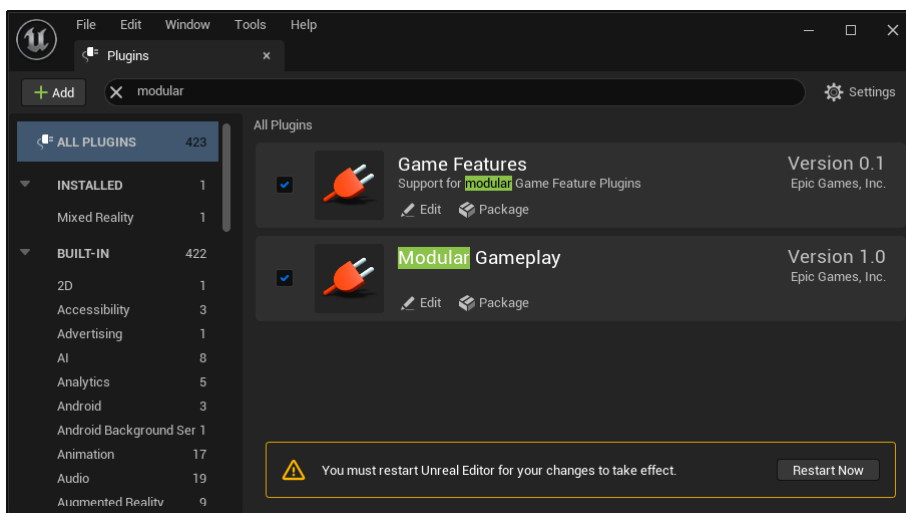
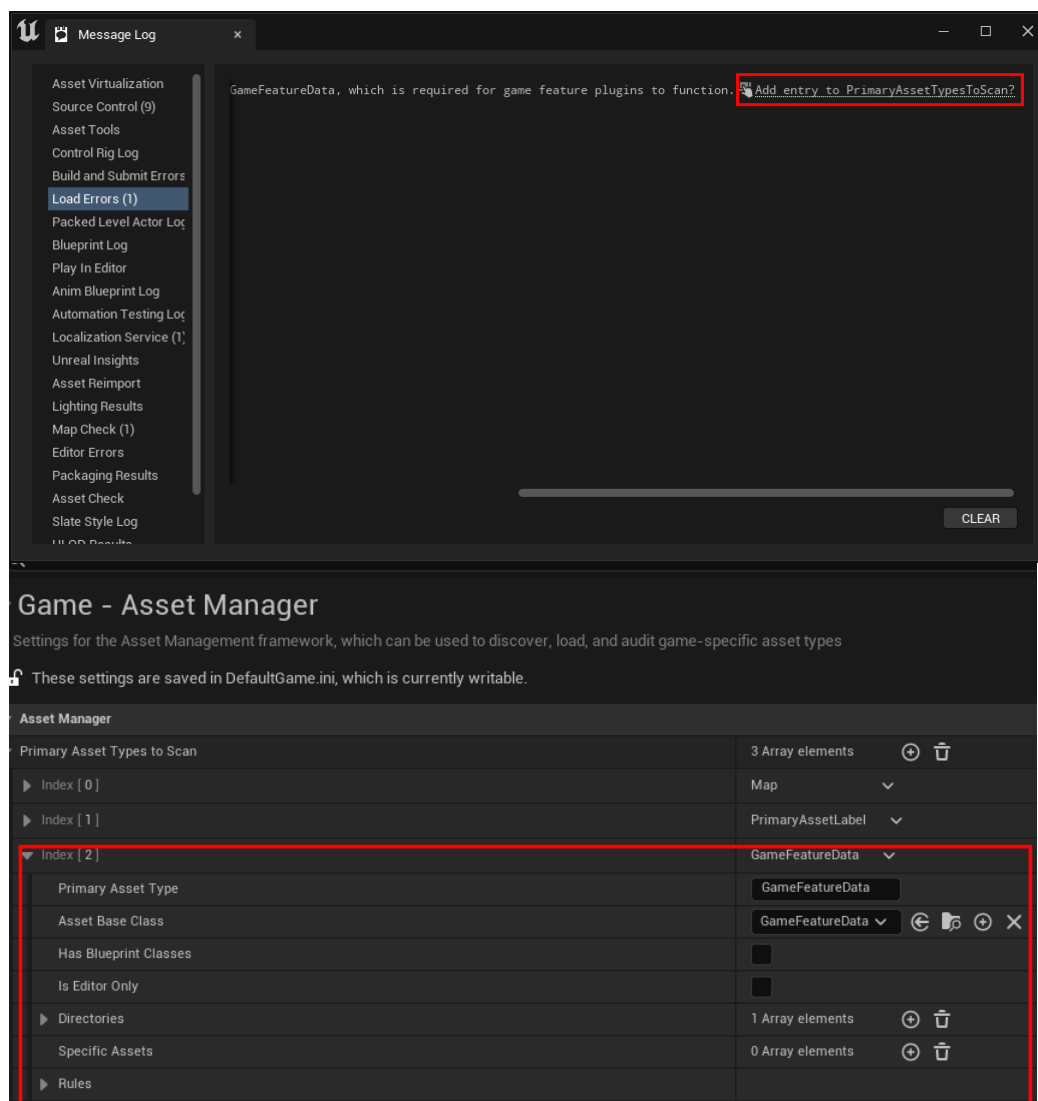


启用插件

首先启用两个插件：Game Features和Modular Gameplay

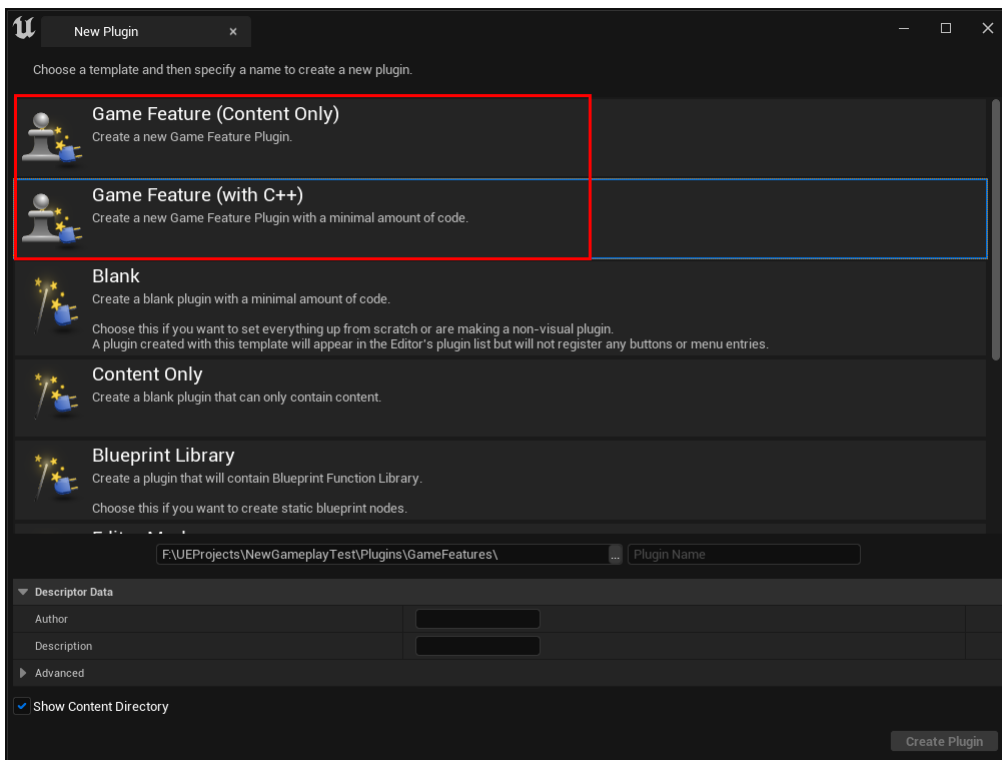


重启编辑器后，会出现弹窗，提示需要将GameFeatureData添加为PrimaryAssetDataScan。直接点击弹窗消息最后的链接，直接添加。



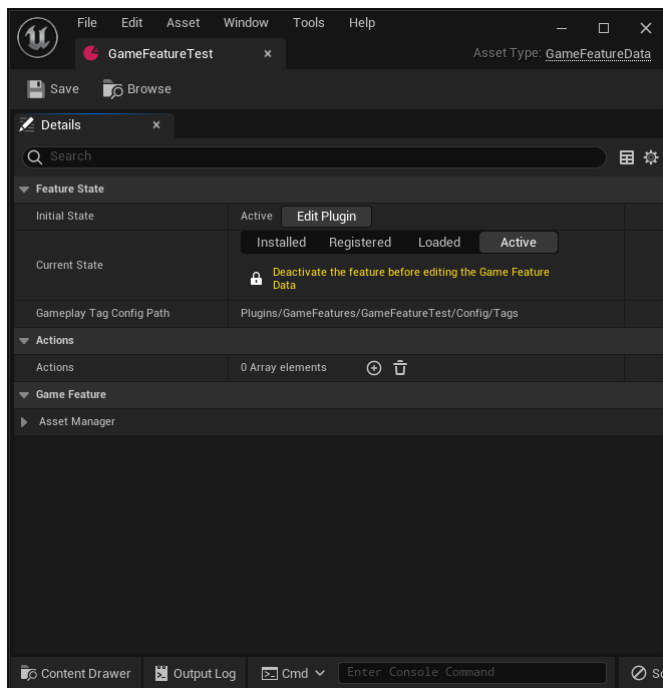
创建GameFeature插件和模块

可以通过创建插件的方式创建一个GameFeature，有两个可选插件模板，一个是仅包含资产的插件，另一个则是也包含C++的GameFeature插件。



GameFeature注册表

插件创建完成后，在Content目录中会有一个DataAsset资产文件用于注册和管理该GameFeature的功能。



Initial State

可以编辑插件信息，也可以设置该GameFeature在编辑器和游戏启动时的初始化状态。

Current State

当前的Game Feature状态，包含四个状态：

Installed: 当前模块仅存在于存储硬件，不加载

Registered: 插件中的资产可以被发现，但不加载

Loaded: 加载插件中的资产到内存，但是不激活

Active: 激活插件功能

Actions

功能使用

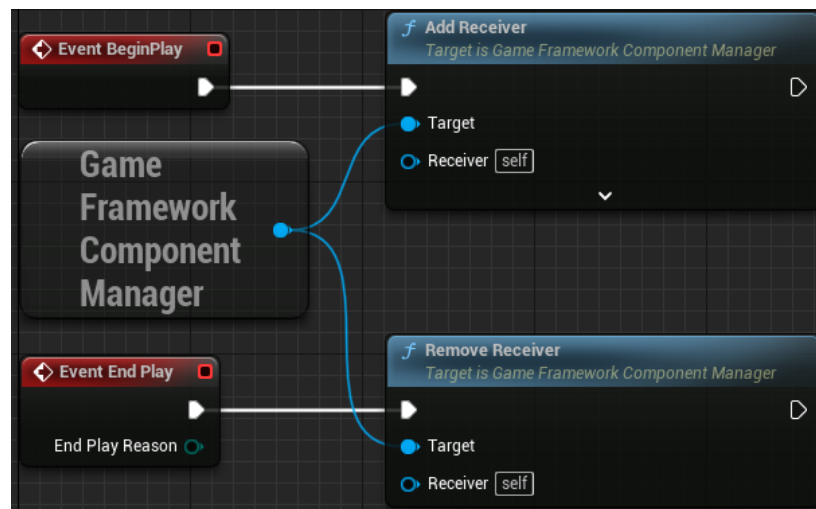
仅在注册表中为指定类型添加组件或功能并不能直接起作用，还需要对被处理的类注册Receiver，下例为给ACustPlayerController添加一个组件的设置。

在ACustPlayerController中注册Receiver：

```
void ACustPlayerController::PreInitializeComponents()
{
    Super::PreInitializeComponents();
    UGameFrameworkComponentManager::AddGameFrameworkComponentsReceiver(this);
}

void ACustPlayerController::EndPlay(const EEndPlayReason::Type EndPlayReason)
{
    UGameFrameworkComponentManager::RemoveGameFrameworkComponentReceiver(this);
    Super::EndPlay(EndPlayReason);
}
```

蓝图：



FEATURE STATE

Initial State

Active

Edit Plugin

Current State

Installed

Registered

Loaded

Active

🔒

Deactivate the feature before editing the Game Feature Data

MyFeature

Activated

Load

Activate

Deactivate

Unload

ListGameFeaturePlugins [-activeonly] [-alphasort] [-csv]
 LoadGameFeaturePlugin XXX
 DeactivateGameFeaturePlugin URL
 UnloadGameFeaturePlugin URL

编辑器触发

代码触发

命令行触发

```

/** The manager subsystem for game features */
UCLASS()
class GAMEFEATURES_API UGameFeaturesSubsystem : public UEngineSubsystem
{
    GENERATED_BODY()
    //....
public:
    /** Returns all the active plugins GameFeatureDatas */
    void GetGameFeatureDataForActivePlugins(TArray<const UGameFeatureData*> & OutActivePluginFeatureDatas);

    /** Returns the game feature data for the plugin specified by PluginURL */
    const UGameFeatureData* GetGameFeatureDataForActivePluginByURL(const FString& PluginURL);

    /** Loads a single game feature plugin. */
    void LoadGameFeaturePlugin(const FString& PluginURL, const FGameFeaturePluginLoadComplete& CompleteDelegate);

    /** Loads a single game feature plugin and activates it. */
    void LoadAndActivateGameFeaturePlugin(const FString& PluginURL, const FGameFeaturePluginLoadComplete& CompleteDelegate);

    /** Gets the Install_Percent for single game feature plugin if it is active. */
    bool GetGameFeaturePluginInstallPercent(const FString& PluginURL, float& Install_Percent);

    /** Deactivates the specified plugin */
    void DeactivateGameFeaturePlugin(const FString& PluginURL);
    void DeactivateGameFeaturePlugin(const FString& PluginURL, const FGameFeaturePluginDeactivateComplete& CompleteDelegate);

    /** Unloads the specified game feature plugin. */
    void UnloadGameFeaturePlugin(const FString& PluginURL, bool bKeepRegistered = false);
    void UnloadGameFeaturePlugin(const FString& PluginURL, const FGameFeaturePluginUnloadComplete& CompleteDelegate, bool bKeepRegistered = false);

    /** Uninstall the specified game feature plugin. Will remove the game feature plugin from the device if it was downloaded */
    void UninstallGameFeaturePlugin(const FString& PluginURL);
    void UninstallGameFeaturePlugin(const FString& PluginURL, const FGameFeaturePluginUninstallComplete& CompleteDelegate);

    /** If the specified plugin is a built-in plugin, return the URL used to identify it. Returns true if the plugin exists, false if it was not found */
    bool GetPluginURLForBuiltInPluginByName(const FString& PluginName, FString& OutPluginURL);

    /** Get the plugin path from the plugin URL */
    FString GetPluginPathFromPluginURL(const FString& PluginURL);
  
```

知乎 @大钊

Modular Gameplay

UGameFrameworkComponentManager(GameInstanceSubsystem)

AGameFrameworkComponent(ActorComponent)

Game Features

UGameFeaturesSubsystem(EngineSubsystem)

UGameFeatureData(PrimaryDataAsset)

UGameFeatureAction(Object)

UGameFeaturePluginStateMachine(Object)

