GYGameplayEffectDataTable

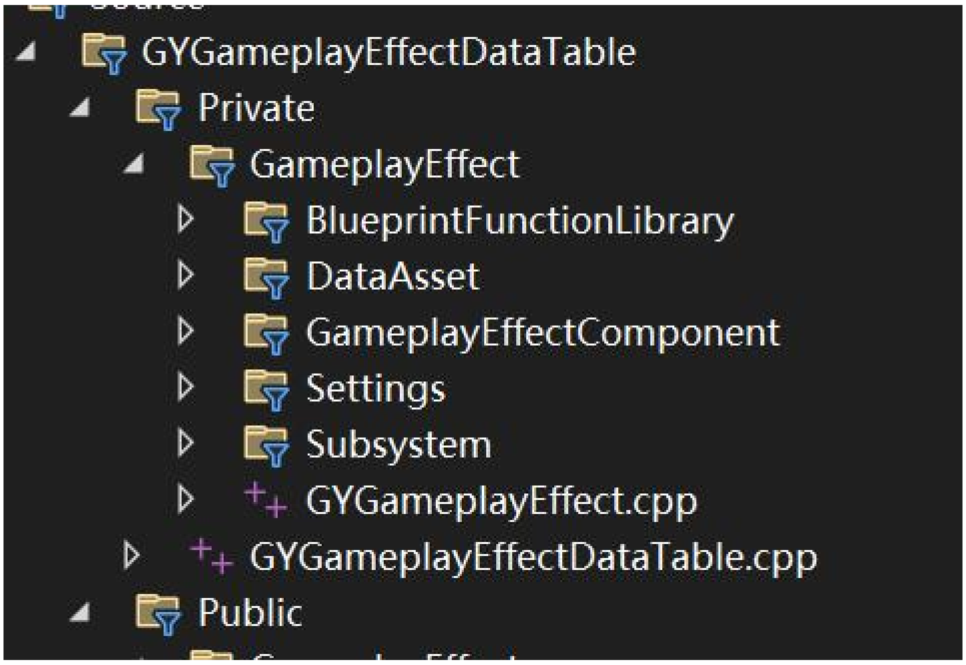
Contents

Introduction：

The GYGameplayEffectDataTable plugin allows us to use GE through a DataTable. It also supports custom UGameplayEffect types, GE components, and DataTable types. Through the interface, we can easily trigger the effect of GE.

This is beneficial for GE’s unified management, while also reducing the tedious process of creating GE blueprints repeatedly, and is more conducive to project maintenance.

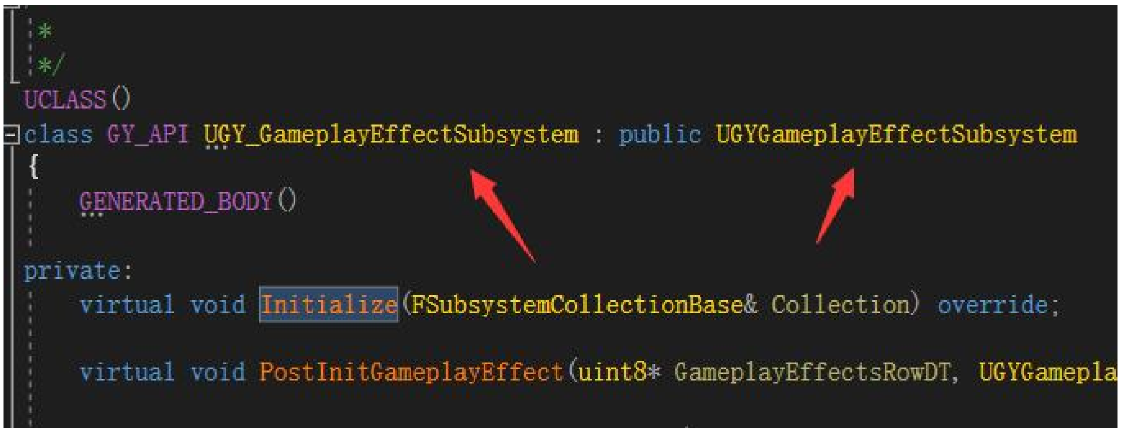
Plugin Directory：



```
GYGameplayEffectDataTable
  Private
    GameplayEffect
      BlueprintFunctionLibrary
      DataAsset
      GameplayEffectComponent
      Settings
      Subsystem
      GYGameplayEffect.cpp
    GYGameplayEffectDataTable.cpp
  Public
```

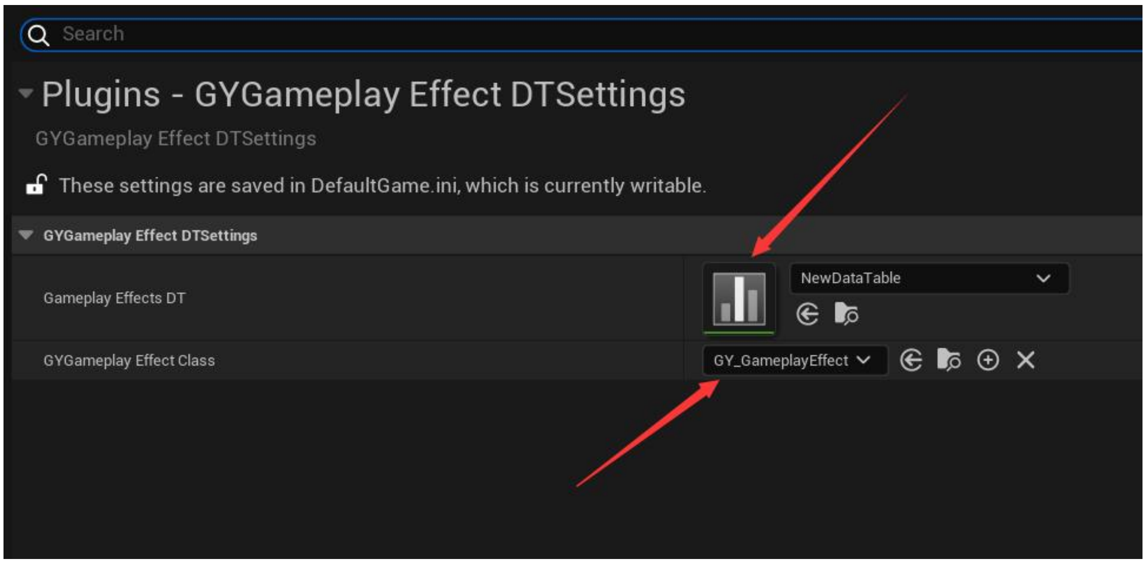
From top to bottom, BlueprintFunctionalLibrary contains our GE trigger interface and other libraries. The DataAsset contains the structure of our GE's DataTable. GameplayEffectComponent stores the data of GE components. The Settings section is our project settings class. Subsystem is the manager class of our plugin.

Use Flow:



```
 *
 */
UCLASS()
class GY_API UGY_GameplayEffectSubsystem : public UGYGameplayEffectSubsystem
{
    GENERATED_BODY()

private:
    virtual void Initialize(FSubsystemCollectionBase& Collection) override;

    virtual void PostInitGameplayEffect(uint8* GameplayEffectsRowDT, UGYGamepla
```

Firstly, we inherit UGYGameplayEffectSubsystem in our own project and generate a GE subsystem.Then open the editor to create a DataTable for FGY_GameplayEffectsDT.



Search

Plugins - GYGameplay Effect DTSettings

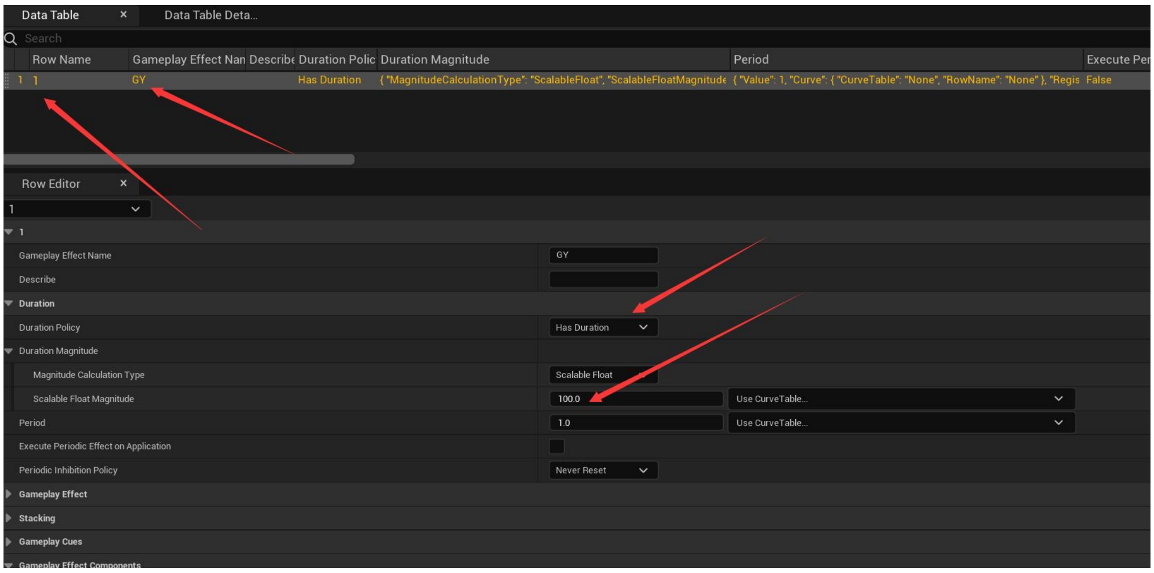
GYGameplay Effect DTSettings

These settings are saved in DefaultGame.ini, which is currently writable.

GYGameplay Effect DTSettings

Gameplay Effects DT	NewDataTable
GYGameplay Effect Class	GY_GameplayEffect

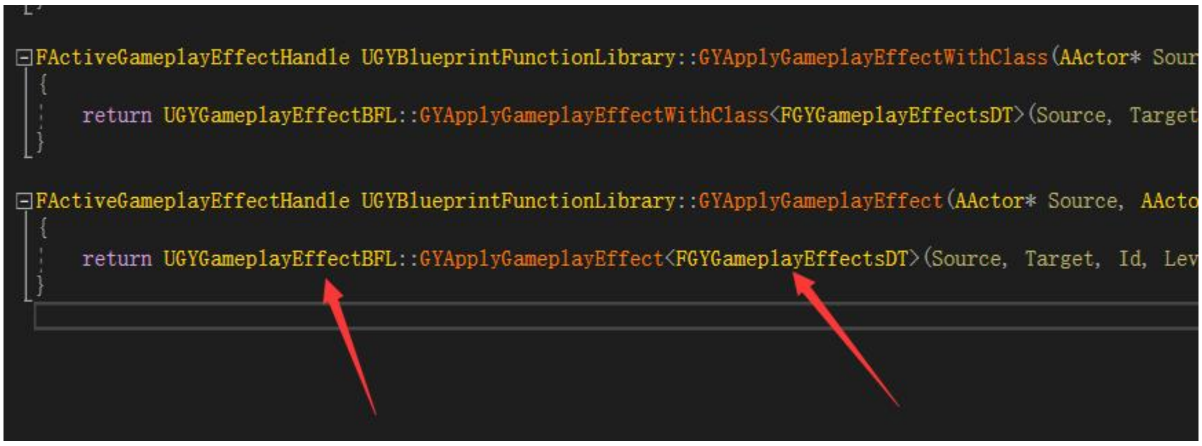
Configure the DataTable into the project settings of the plugin.



	Row Name	Gameplay Effect Nan	Describe	Duration Polic	Duration Magnitude	Period	Execute Per
1	1	GY		Has Duration	{ "MagnitudeCalculationType": "ScalableFloat", "ScalableFloatMagnitude	{ "Value": 1, "Curve": { "CurveTable": "None", "RowName": "None" }, "Regis	False

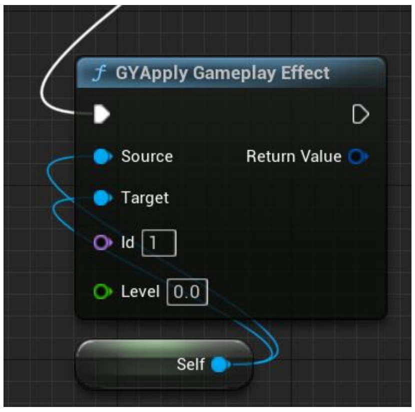
1		
Gameplay Effect Name	GY	
Describe		
Duration		
Duration Policy	Has Duration	
Duration Magnitude		
Magnitude Calculation Type	Scalable Float	
Scalable Float Magnitude	100.0	Use CurveTable...
Period	1.0	Use CurveTable...
Execute Periodic Effect on Application	☐	
Periodic Inhibition Policy	Never Reset	
Gameplay Effect		
Stacking		
Gameplay Cues		
Gameplay Effect Components		

Open the DataTable and configure them according to the arrows in the diagram, with RowName set to 1. RowName is the identifier that triggers this GE.



```
FActiveGameplayEffectHandle UGYBlueprintFunctionLibrary::GYApplyGameplayEffectWithClass(AActor* Sour
{
    return UGYGameplayEffectBFL::GYApplyGameplayEffectWithClass<FGYGameplayEffectsDT>(Source, Target
}

FActiveGameplayEffectHandle UGYBlueprintFunctionLibrary::GYApplyGameplayEffect(AActor* Source, AActo
{
    return UGYGameplayEffectBFL::GYApplyGameplayEffect<FGYGameplayEffectsDT>(Source, Target, Id, Lev
}
```

Encapsulate the UGYGameplayAffectBFL:: GYAppleGameplayEffect static template function in your own project library. Please note that FGYGameplayAffectsDT is of the DataTable type that we are using.

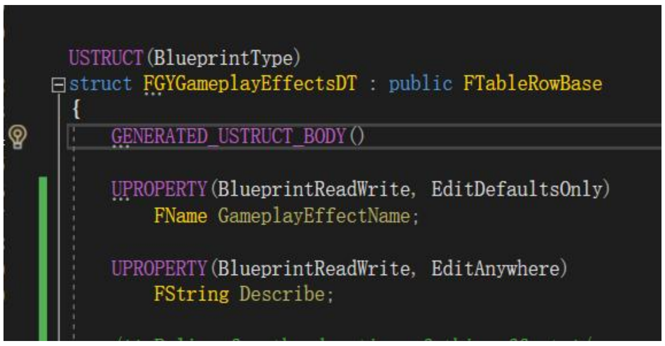


Then we can easily trigger our GE through the ID.

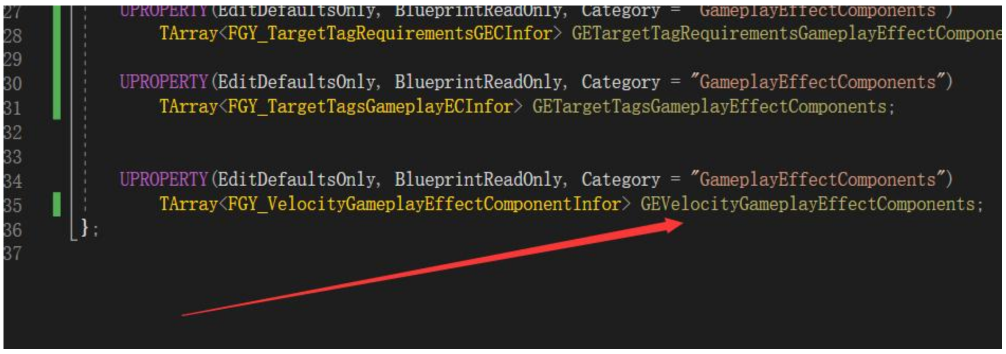
Custom DataTable:

Copy and paste all member variables from FGY_GameplayEffectsDT directly into a new DataTable class. The reason why inheritance is not necessary here is to ensure that the order of variables in the DataTable is consistent, and to ensure that the order of variables exported to Excel in the future is consistent. Of course, if you don't consider the aesthetics of variable properties, you can also directly inherit FGY_GameplayEffectsDT.

The reason why it is not encapsulated as a # define macro here is that Unreal Engine's UHT will be earlier than C++preprocessing, which can cause Unreal Engine macros such as UPROPERTY to fail.

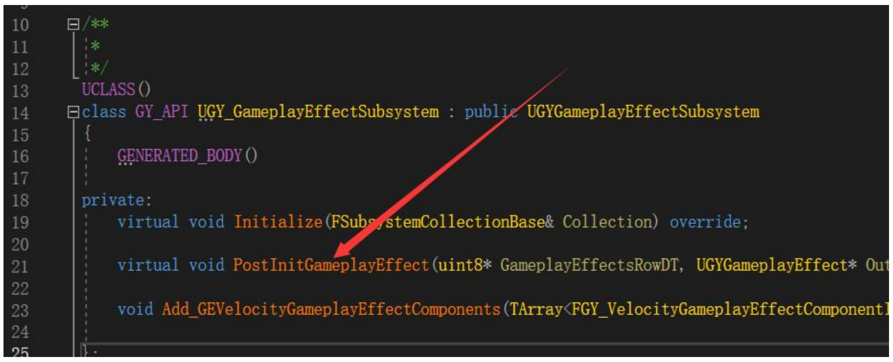


```
USTRUCT(BlueprintType)
struct FGYGameplayEffectsDT : public FTableRowBase
{
    GENERATED_USTRUCT_BODY()

    UPROPERTY(BlueprintReadWrite, EditDefaultsOnly)
        FName GameplayEffectName;

    UPROPERTY(BlueprintReadWrite, EditAnywhere)
        FString Describe;
```



```
    UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category = "GameplayEffectComponents")
        TArray<FGY_TargetTagRequirementsGECInfor> GETargetTagRequirementsGameplayEffectCompone

    UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category = "GameplayEffectComponents")
        TArray<FGY_TargetTagsGameplayECInfor> GETargetTagsGameplayEffectComponents;


    UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category = "GameplayEffectComponents")
        TArray<FGY_VelocityGameplayEffectComponentInfor> GEVelocityGameplayEffectComponents;
};
```

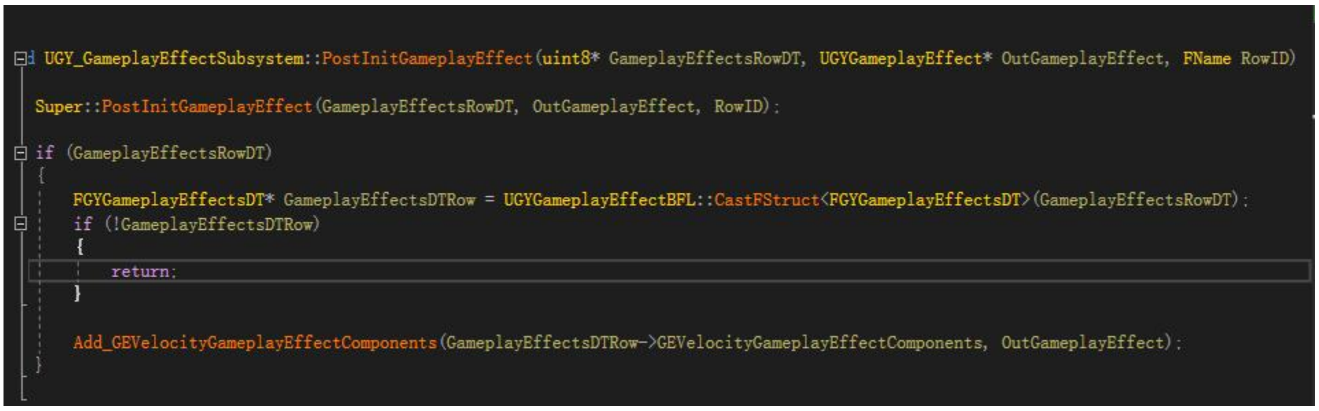
FGYGameplayEffectsDT is a custom DataTable created by me by copying and pasting FGY_GameplayEffectsDT, and adding my own custom GE component GEVelocityGameplayEffectComponents to it.



```
/**
 *
 */
UCLASS()
class GY_API UGY_GameplayEffectSubsystem : public UGYGameplayEffectSubsystem
{
    GENERATED_BODY()

private:
    virtual void Initialize(FSubsystemCollectionBase& Collection) override;

    virtual void PostInitGameplayEffect(uint8* GameplayEffectsRowDT, UGYGameplayEffect* Out

    void Add_GEVelocityGameplayEffectComponents(TArray<FGY_VelocityGameplayEffectComponentI

};
```

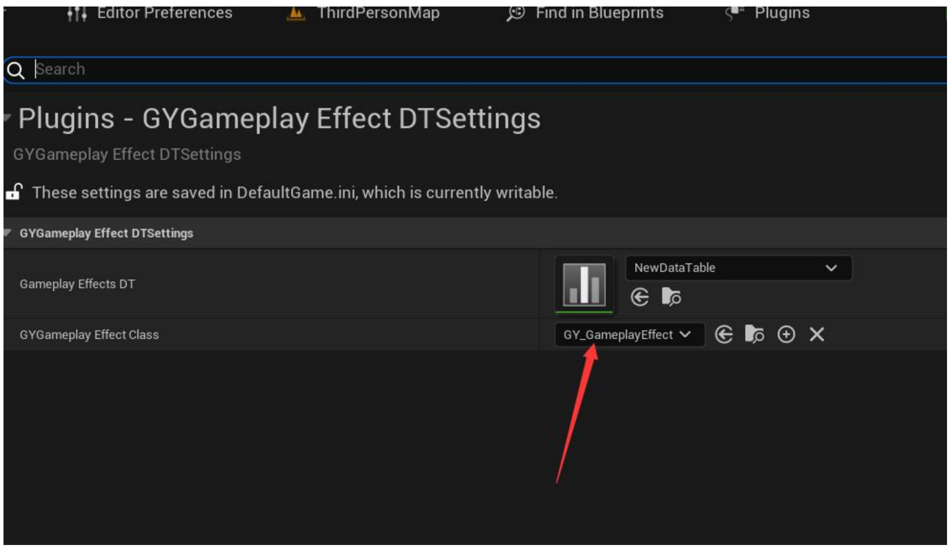
通过重写 UGYGameplayEffectSubsystem 中的 PostInitGameplayEffect 虚函数，我们可以处理我们自己的 DataTable 中自定义的数据。

By rewriting the PostInitGameplayEffect virtual function in UGYGameplayEffectSubsystem, we can handle custom data in our own DataTable.



```
void UGY_GameplayEffectSubsystem::PostInitGameplayEffect(uint8* GameplayEffectsRowDT, UGYGameplayEffect* OutGameplayEffect, FName RowID)
{
	Super::PostInitGameplayEffect(GameplayEffectsRowDT, OutGameplayEffect, RowID);

	if (GameplayEffectsRowDT)
	{
		FGYGameplayEffectsDT* GameplayEffectsDTRow = UGYGameplayEffectBFL::CastFStruct<FGYGameplayEffectsDT>(GameplayEffectsRowDT);
		if (!GameplayEffectsDTRow)
		{
			return;
		}

		Add_GEVelocityGameplayEffectComponents(GameplayEffectsDTRow->GEVelocityGameplayEffectComponents, OutGameplayEffect);
	}
}
```

You can refer to my writing style and use GameplayEffectsRowDT to obtain our custom DataTable data.



Plugins - GYGameplay Effect DTSettings

GYGameplay Effect DTSettings

These settings are saved in DefaultGame.ini, which is currently writable.

GYGameplay Effect DTSettings

Gameplay Effects DT	NewDataTable
GYGameplay Effect Class	GY_GameplayEffect

By inheriting UGY_GameplayEffect, we can customize our UGYGameplayEffect type and configure it to the above image. The OutGameplayEffectt parameter passed into the PostInitGameplayEffect virtual function in UGYGameplayEffectSubsystem is our custom UGYGameplayEffect. Simple Cast can achieve custom logic processing.

Summary：

The plugin is completely free, and everyone is welcome to use and provide feedback.