Editor Utility Widget
Petit Deep Dive

動画アーカイブはこちら！（講演超絶ヘタクソですみません！）

第4回UE4何でも勉強会 in 東京アーカイブ

ゲームデザイナー
キンアジ

注意

- 「Editor Utility Widget」のことを「EUW」と省略しています。
- スライド内の UE4Editor のキャプチャー画像は
  すべて4.24.2を使用しています。
- 以下のEUWの知識をつけてから読むことをお勧めします。(スライドは後日公開)
  また、一部の紹介した機能は、後日プロジェクトを公開するのでダウンロードして
  ご自身の環境でお試しください。

↓↓EUWのドキュメント

- Editor Utility Widget | Unreal Engine ドキュメント
- Editor Utility Widgetで色々便利にしてみた。by しょーご様
- 【UE4】Editor Utility Widgetについてのあれこれby 株式会社アンナプルナ(キンアジ)
- [UE4]エディタ上で動作するツール・エディタ拡張をUMGで簡単に作れる
  Editor Utility Widget についてby おかず様
- 【UE4】メモ:実はEditor Utility WidgetはBlutilityの完全上位互換だった話【★★】by キンアジ

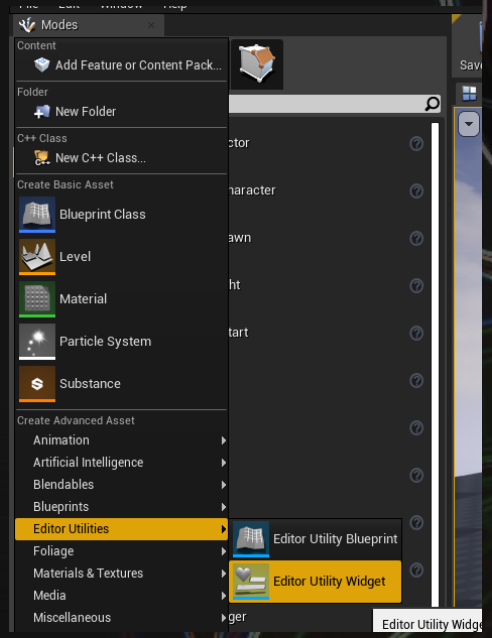
目次

- Details View & Single Property View Widget
  - Details View Widget
  - Single Property View Widget
- Help Text
- Editor Utility Widgetの罠
  - その1
  - その2
  - その3
  - その4
    - レベル遷移時に自動でEUWを消す方法
- Editor Utility Widgetの制作事例

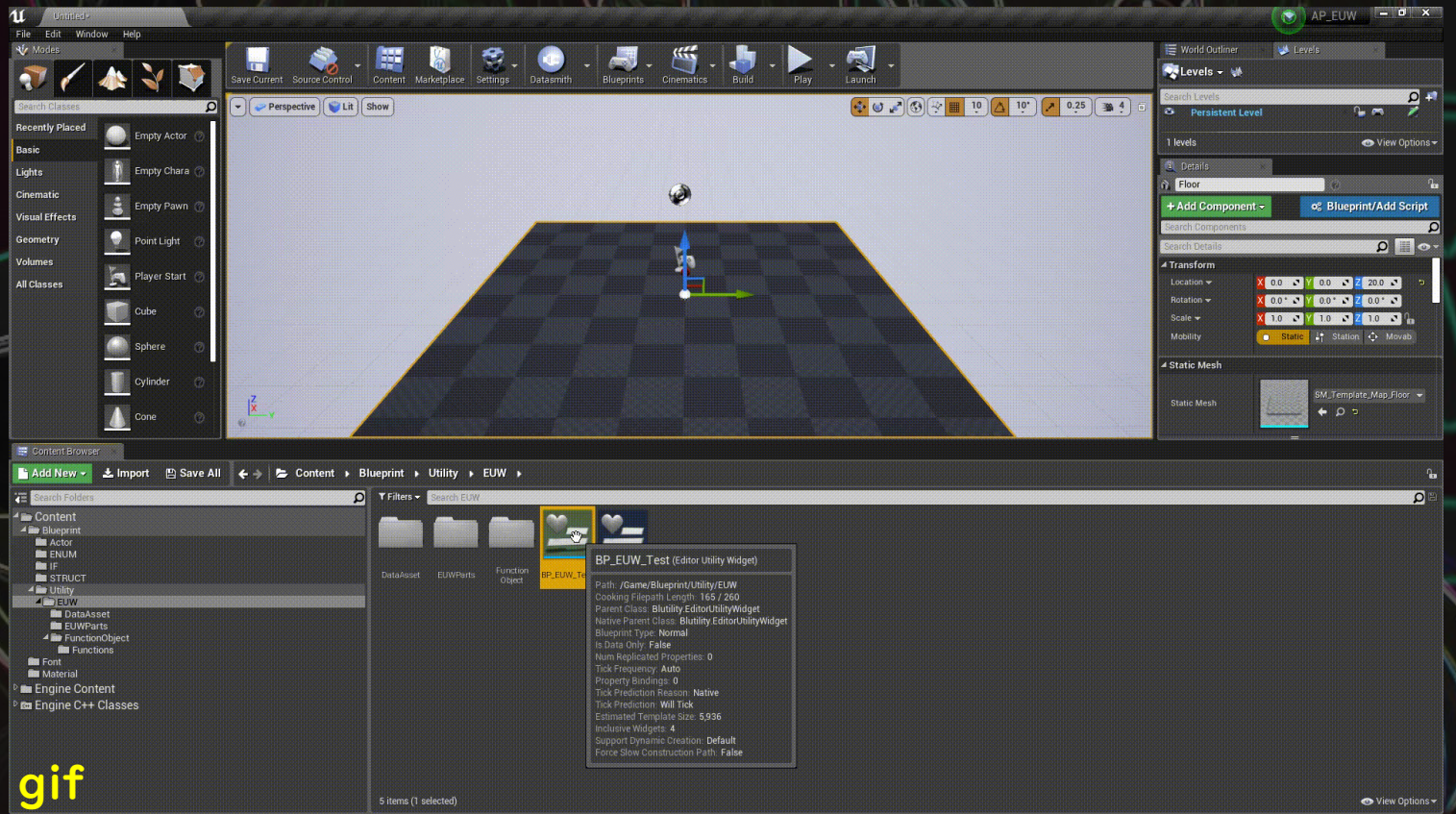
Editor Utility Widget（EUW）って？

- 「User Widget」を使って、エディター拡張ができるものです。このEUWは、その機能をエディター上のツールとして使用することができます。

「Add New」から「Editor Utilities」
→「Editor Utility Widget」で作成可能



EUWを右クリック→「Run Editor Utility Widget」選択で実行できる。



EUWの特徴

- **非プレイ中でも機能する**

→ゲームを実行していなくても、エディターツールなので機能を使うことができます。

- **Editor UtilityなBlueprintが使える**

→エディター上でのみ使えるBlueprint（アセットをセーブする「Save Asset」や、Data TableをReimportする「Fill Data Table from csv String」など）を使うことができます。

- **特殊なWidget「Details View」「Property View」が使える**

→Detail Windowなんかでよく使われているアセットの参照などのUIを使うことができます。

では早速Petit Deep Dive!

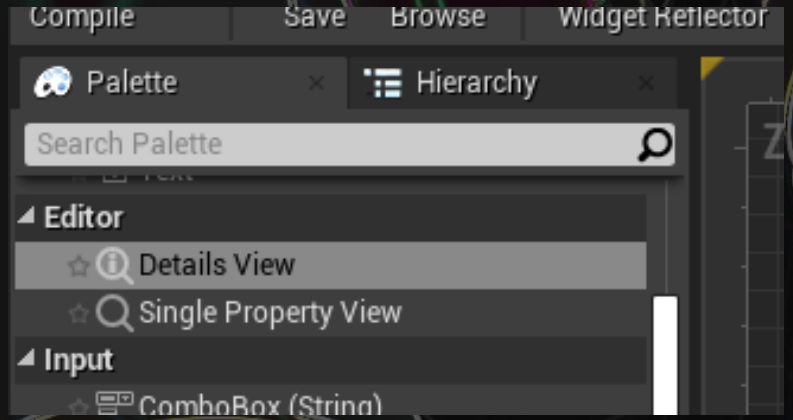
Details View & Single Property View Widget

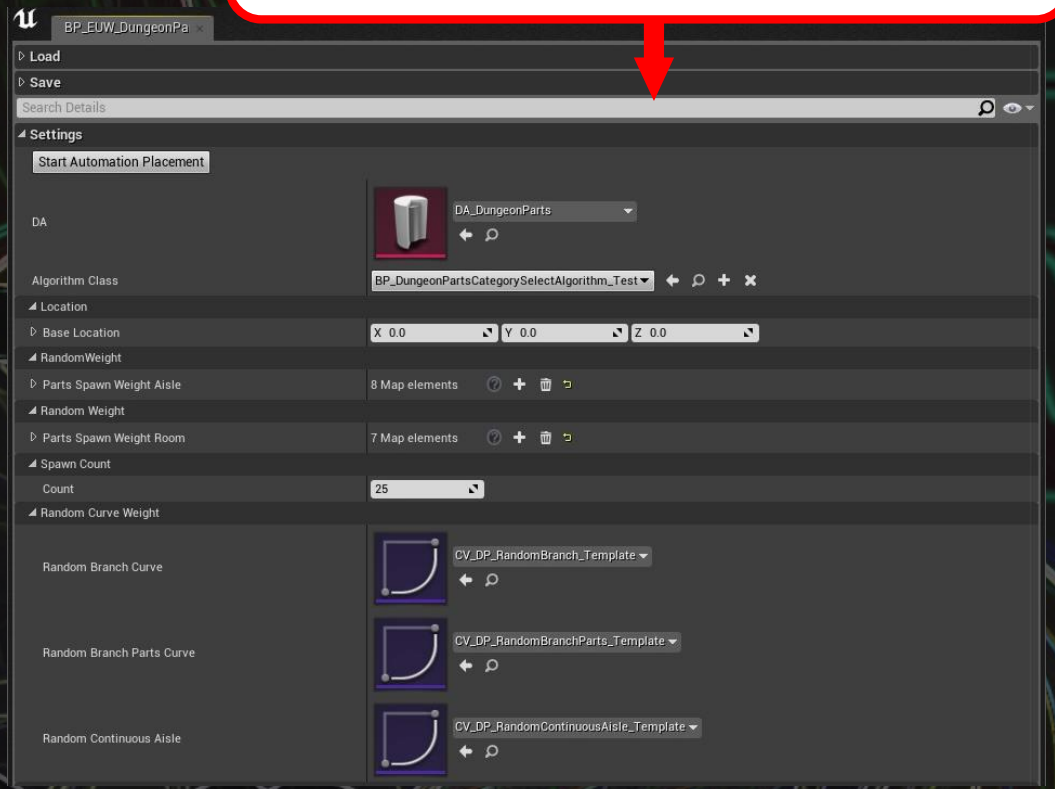
Details View Widget

Details View Widget

- UObject* を対象に、そのObjectが持っているUPROPERTYなプロパティと「Call In Editor」をつけたUFUNCTIONな関数を呼び出すボタンを表示してくれる。
  →「Set Object」関数でUObjectを指定。

こんな風に、EUWの中にアセットの参照や、パラメーターなどを表示してくれる。
超便利！

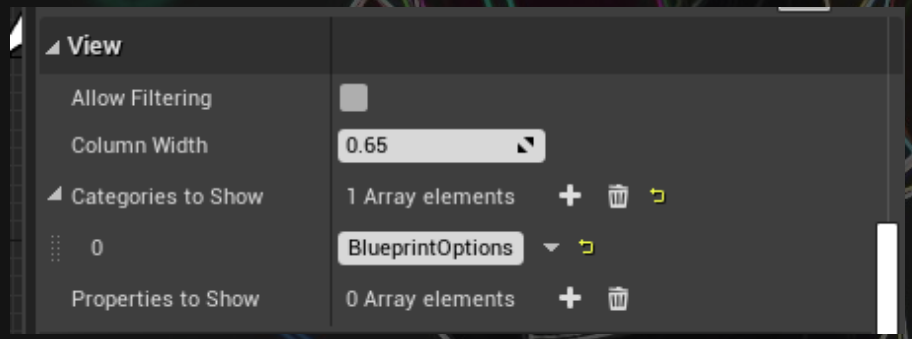


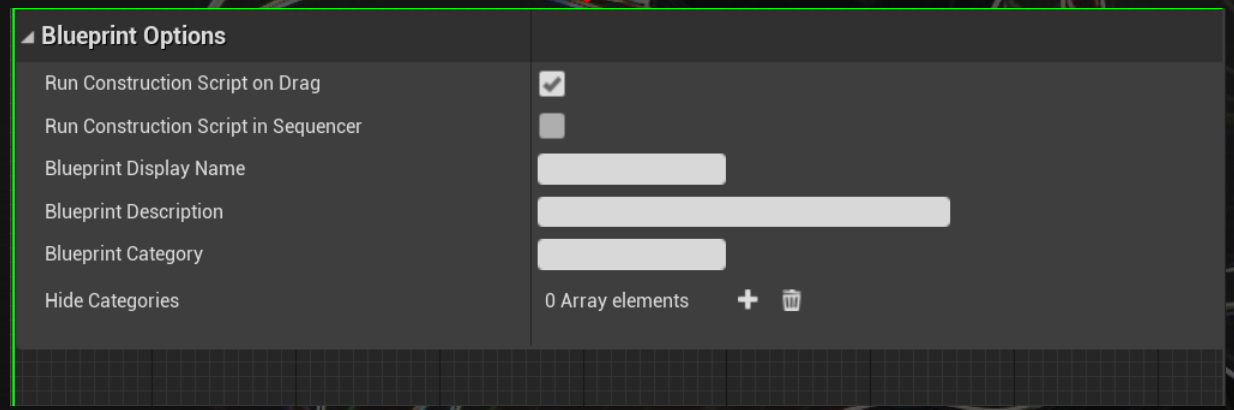


Details View Widget

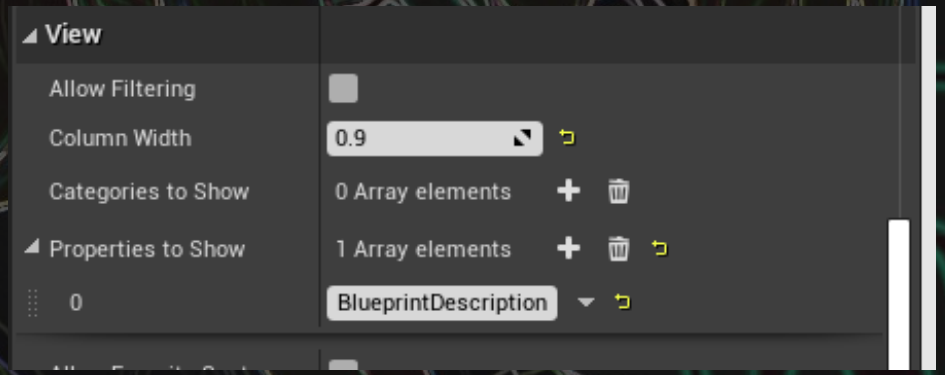
- UObject が持つプロパティが多い場合に、表示するプロパティを制限したい場合、「Categories to Show」と「Property to Show」という配列にそれぞれ、CategoryName,PropertyNameを指定することで表示するプロパティを制限することができます。

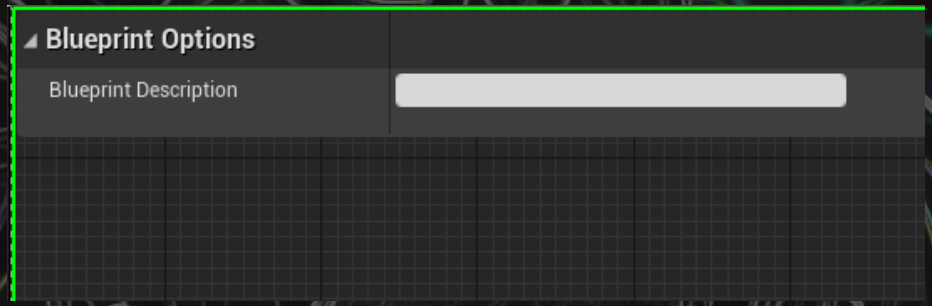
CategoryNameを指定して表示





PropertyNameNameを指定して表示



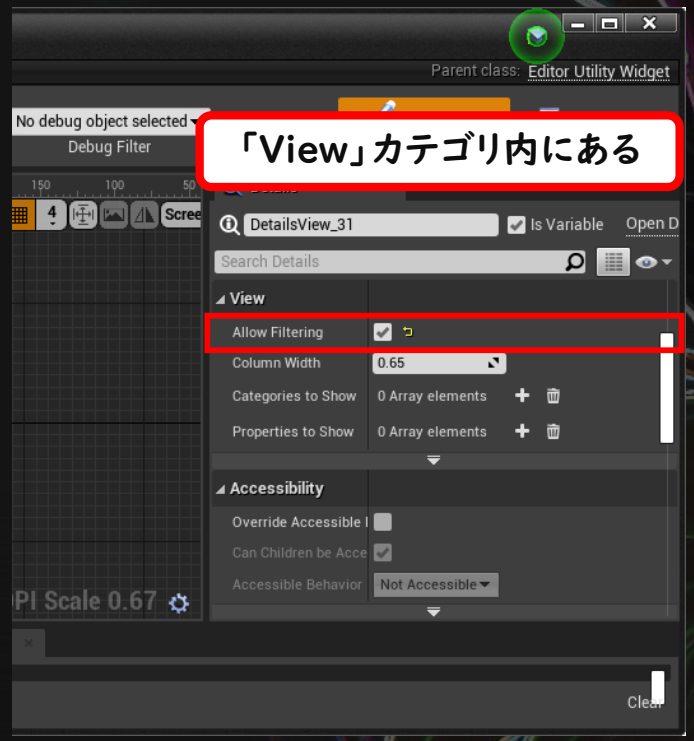


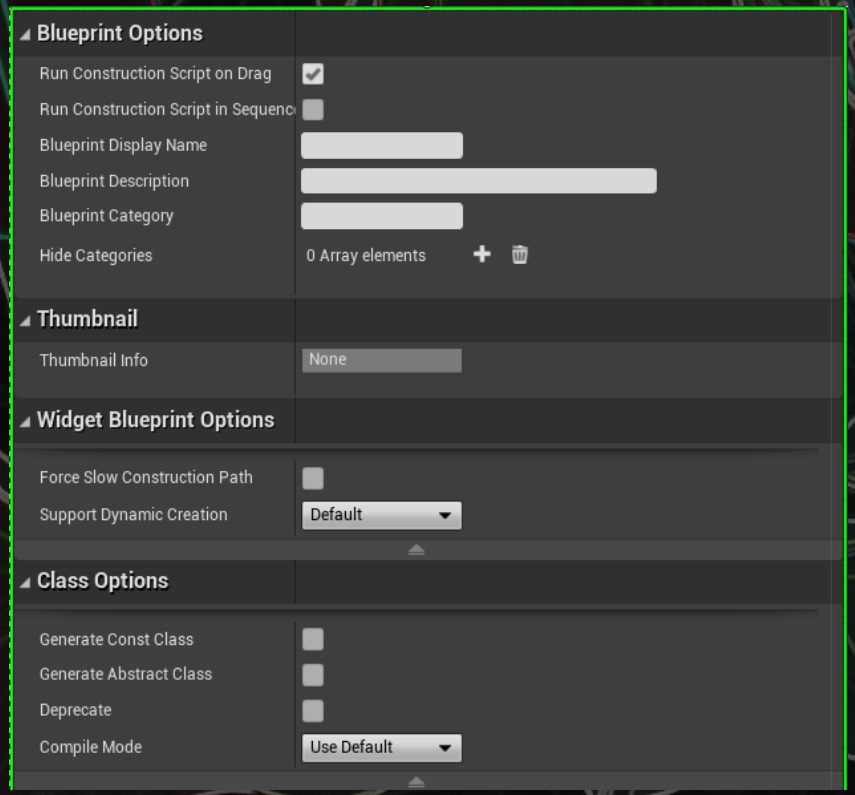
Details View Widget

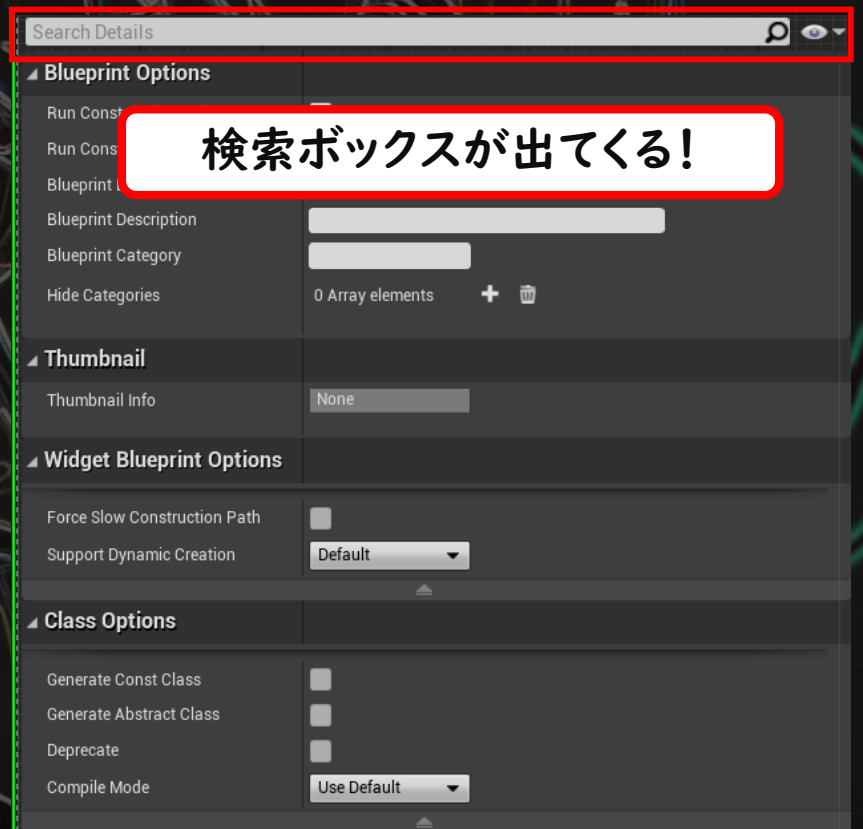
- Property Viewにフィルターを付けたい場合は、
「bAllowFiltering」をTrueにすることで、
Detail Viewの上に検索ボックスが現れる。(デフォルトでFalse)

bAllow Filtering=False

bAllow Filtering=True







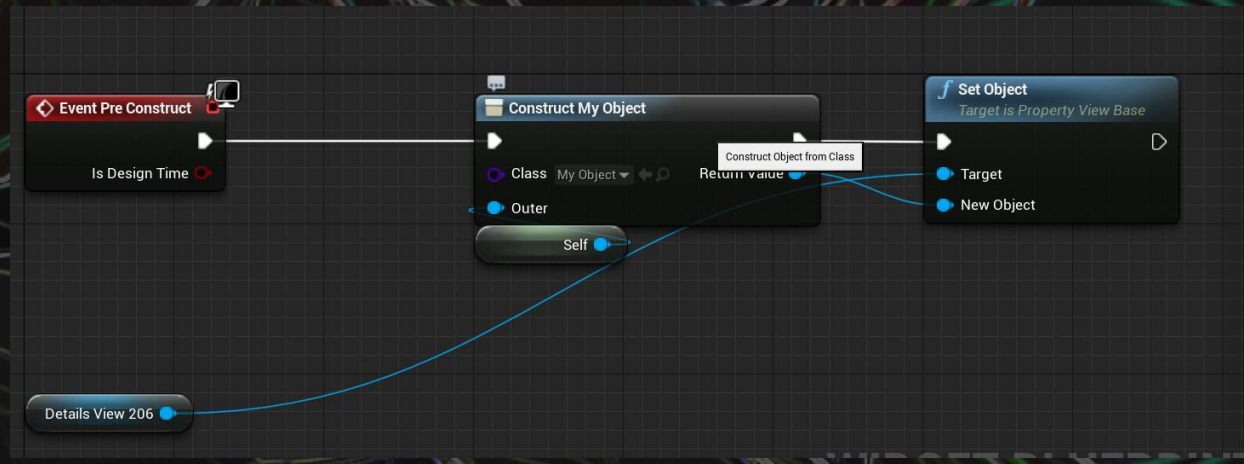
Details View Widget

- 表示されるプロパティは、デフォルトではUProperty*の「PropertyFlags」に「CPF_Edit」をもつUPROPERTY（EditAnywhere,VisibleDefaultOnlyとか）のみ。
  →しかし、「bForceHiddenPropertyVisibility」をTrueにすることで、すべてのUPROPERTYを表示するようになる。

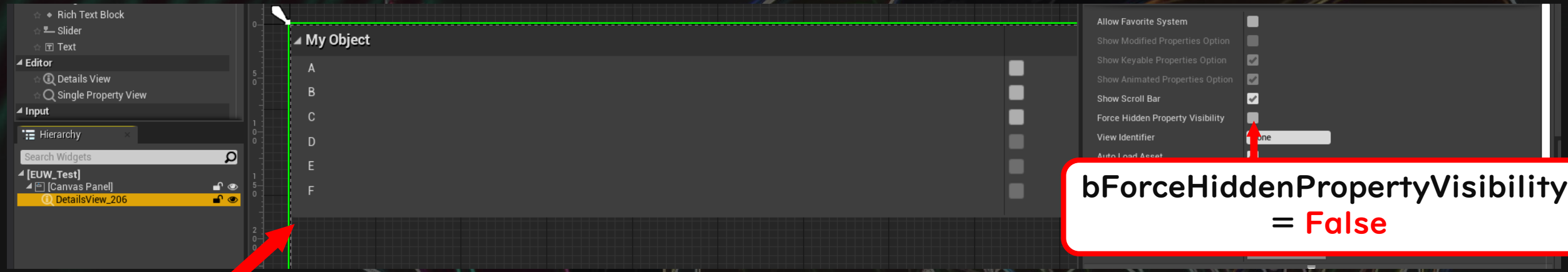
UObject継承のクラスにいろんなUPROPERTYをつける

```
12  UCLASS(Blueprintable)
13  class CPP2_API UMyObject : public UObject
14  {
15      GENERATED_BODY()
16
17  public:
18
19          UPROPERTY(EditAnywhere)
20              bool A;
21
22          UPROPERTY(EditInstanceOnly)
23              bool B;
24
25          UPROPERTY(EditDefaultsOnly)
26              bool C;
27
28          UPROPERTY(VisibleAnywhere)
29              bool D;
30
31          UPROPERTY(VisibleInstanceOnly)
32              bool E;
33
34          UPROPERTY(VisibleDefaultsOnly)
35              bool F;
36
37          UPROPERTY(BlueprintReadOnly)
38              bool G;
39
40          UPROPERTY(BlueprintReadWrite)
41              bool H;
42
43          UPROPERTY(Transient)
44              bool J;
45
46          UPROPERTY()
47              bool K;
```

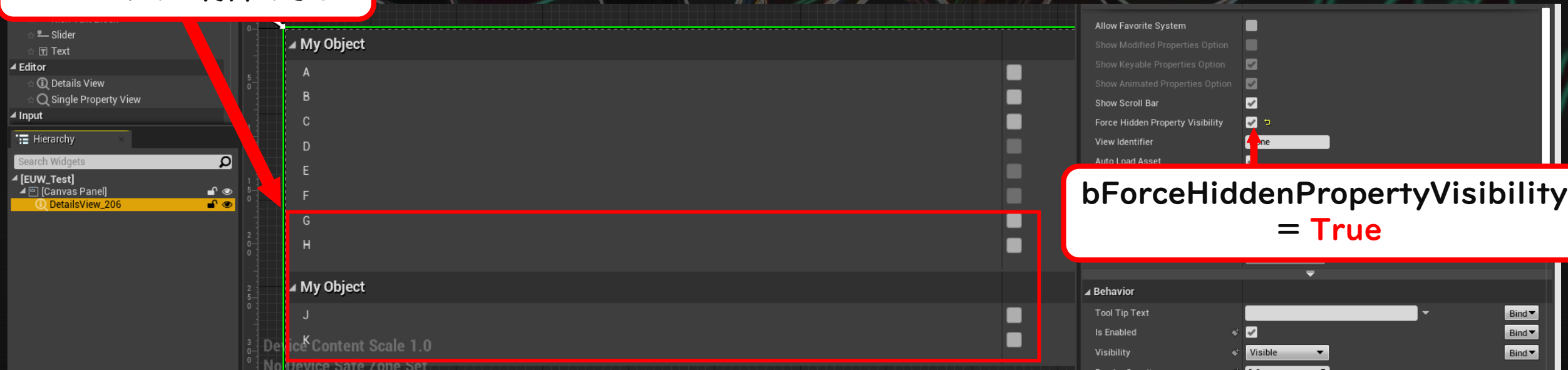
それをDetailViewで表示する
（Editor Utility WidgetのEventGraph）



Details View Widget



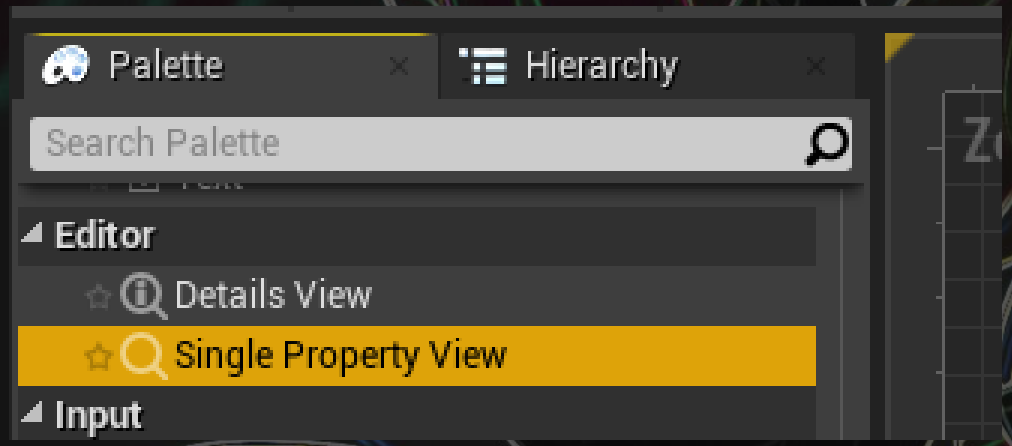
このように表示される
プロパティを制御できる

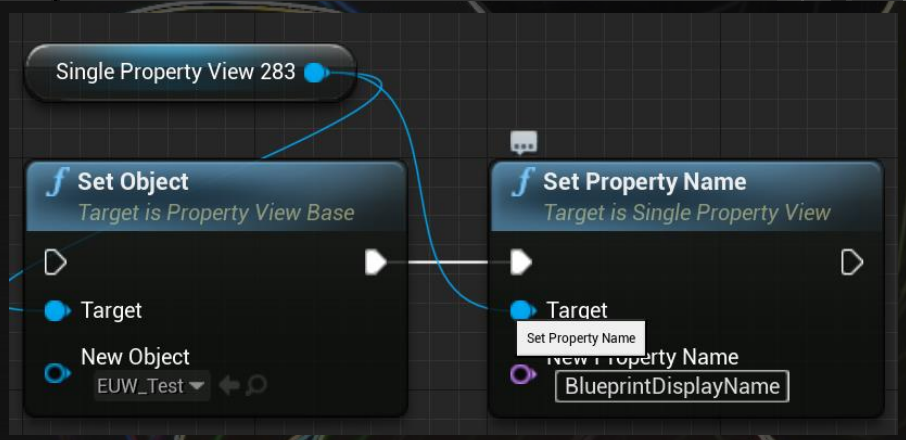


Single Property View Widget

Single Property View Widget

Details View同様プロパティを表示するものだが、
こちらは「PropertyName」を指定して使う。指定したPropertyNameの
プロパティのみ表示してくれる。





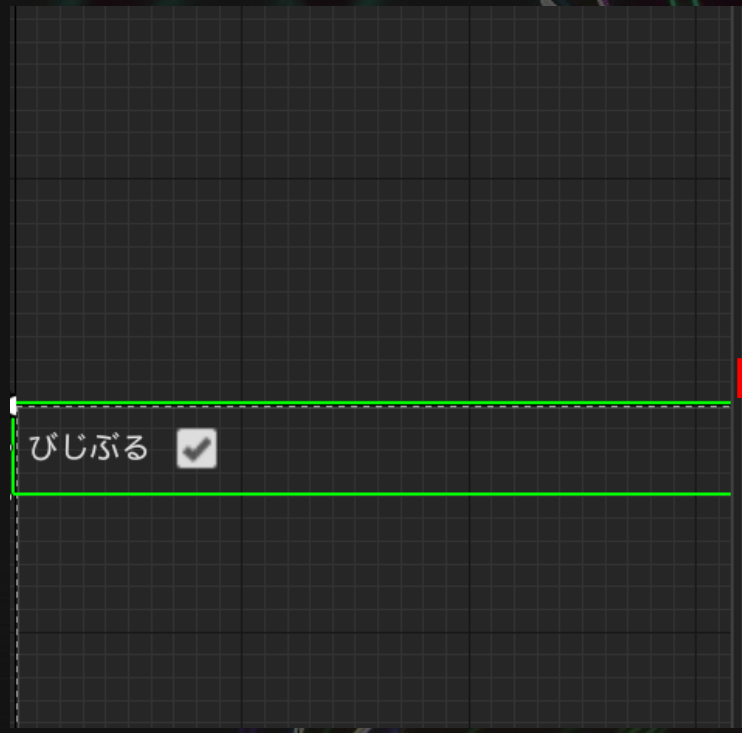
プロパティの一つを表示してくれる！
超便利！

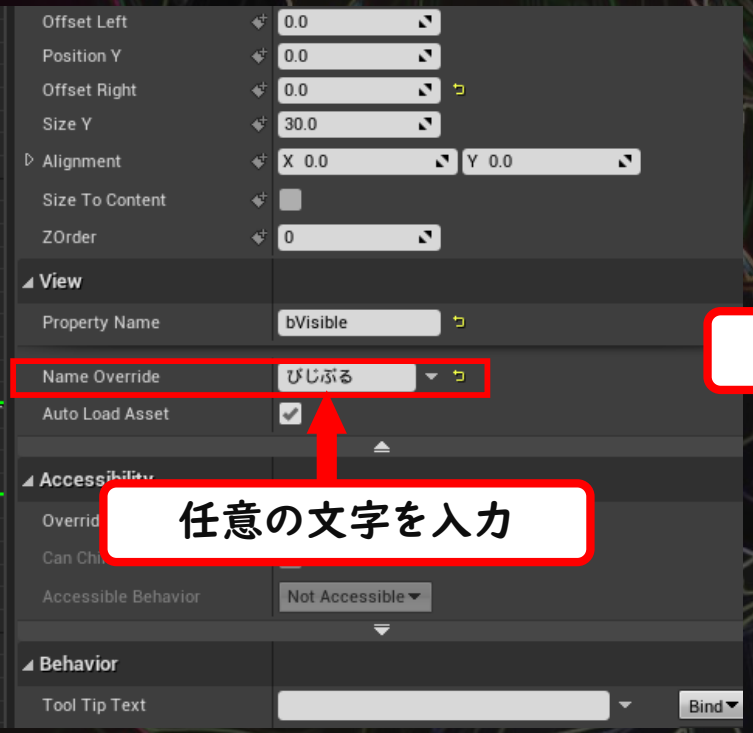
Single Property View Widget

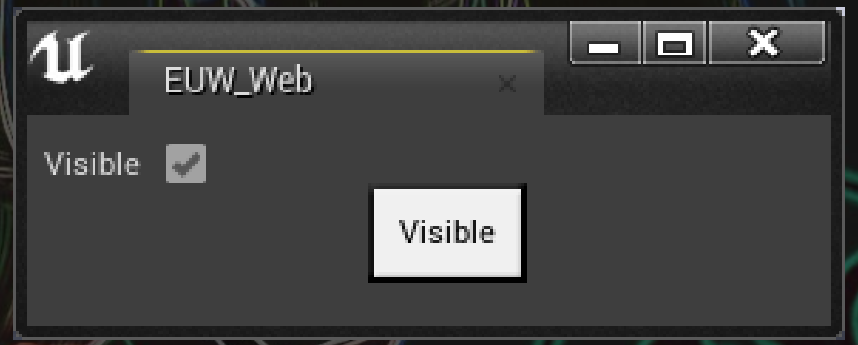
- PropertyNameは正しく指定すること。特にbool。
  （例えば、エディター上はSceneComponentの「bVisible」というプロパティは
  エディター上では「Visible」となっており勘違いしやすい。
  正しく「bVisible」と入力する。

Single Property View Widget

- 「NameOverreide」というプロパティに任意の文字を入れることで、表示される名前が入力した文字になる。
- 表示できるプロパティに制約が多い（次スライド）







表示名が変化！

Single Property View Widget

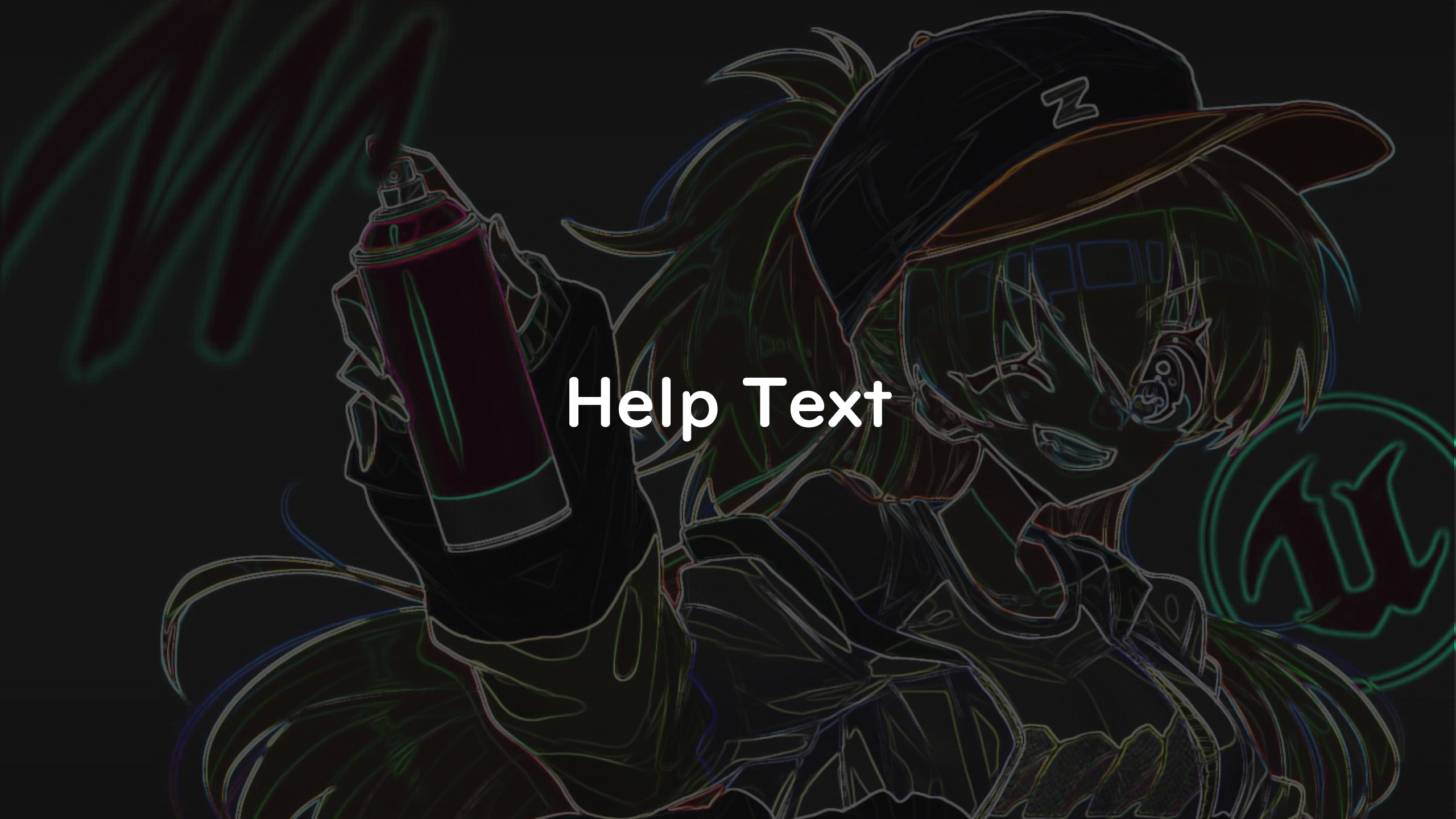
- Details Viewとは違いどんなプロパティでも表示できるわけではない。
  （ bForceHiddenPropertyVisibilityは設定できない）
  →「PropertyFlags」に「CPF_Edit」をもつUPROPERTYのみ。

・さらに、
UStructProperty,UArrayProperty,UMapProperty,USetProperty
なプロパティも設定できない。
（つまり構造体とか配列とかを設定することはできない）
※SinglePropertyView.cpp参照

```
55          else if (PropertyName == NAME_None)
56          {
57              MissingWidgetText = FPropertyViewHelper::UndefinedPropertyText;
58          }
59          else
60          {
61              UProperty* Property = ViewedObject->GetClass()->FindPropertyByName(PropertyName);
62              if (Property == nullptr)
63              {
64                  MissingWidgetText = FPropertyViewHelper::UnknownPropertyText;
65              }
66              else if (!Property->HasAllPropertyFlags(CPF_Edit))
67              {
68                  MissingWidgetText = FPropertyViewHelper::InvalidPropertyText;
69              }
70              else if (Cast<UStructProperty>(Property) || Cast<UArrayProperty>(Property)
71                  || Cast<UMapProperty>(Property) || Cast<USetProperty>(Property)
72                  )
73              {
74                  MissingWidgetText = FPropertyViewHelper::UnsupportedPropertyText;
75              }
76              else
77              {
78                  FPropertyEditorModule& PropertyEditorModule = FModuleManager::GetModuleChecked<FPropert
79                  FSinglePropertyParams SinglePropertyArgs;
80                  SinglePropertyArgs.NameOverride = NameOverride;
```

RootComponentが持っている変数
「RelativeLocation」を設定

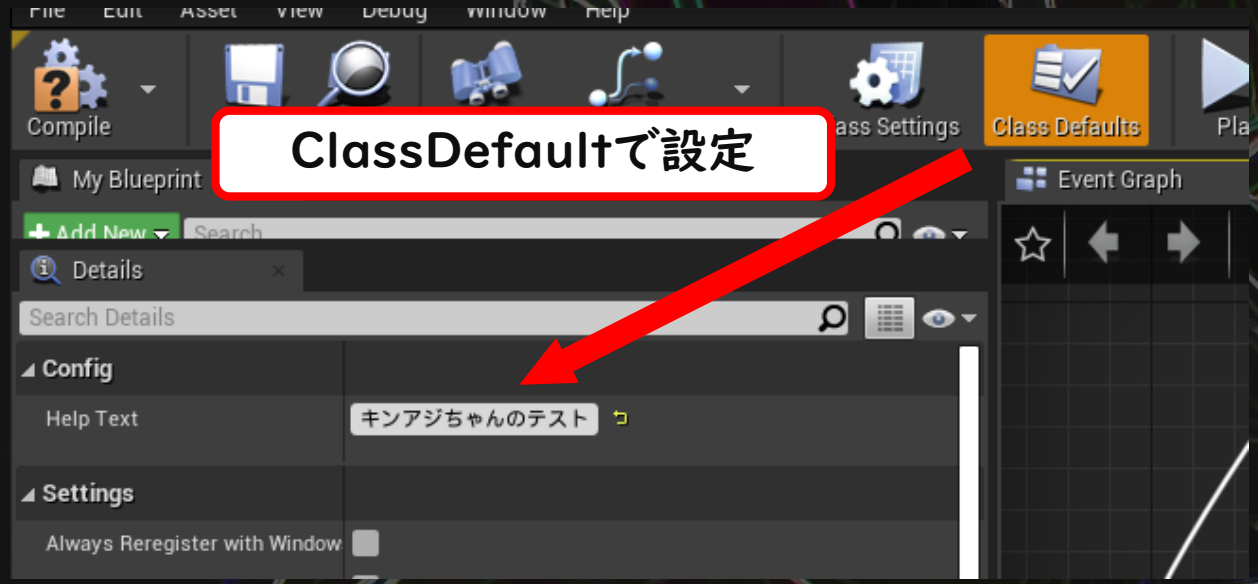
反映されない

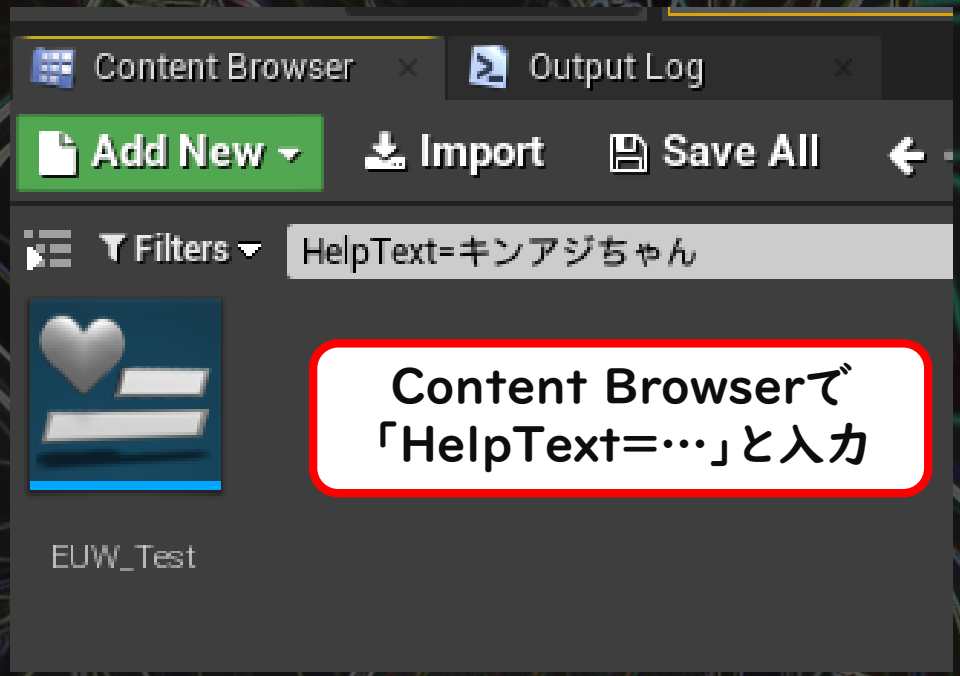

Help Text

Help Text

Content Browser上で補助的な検索ワードとなる文字を設定できる。

→どんな機能なのかをあらかじめ記載しておけば、そのテキストで検索が可能。
（UPROPERTY(AssetRegistrySearchable)なプロパティです）

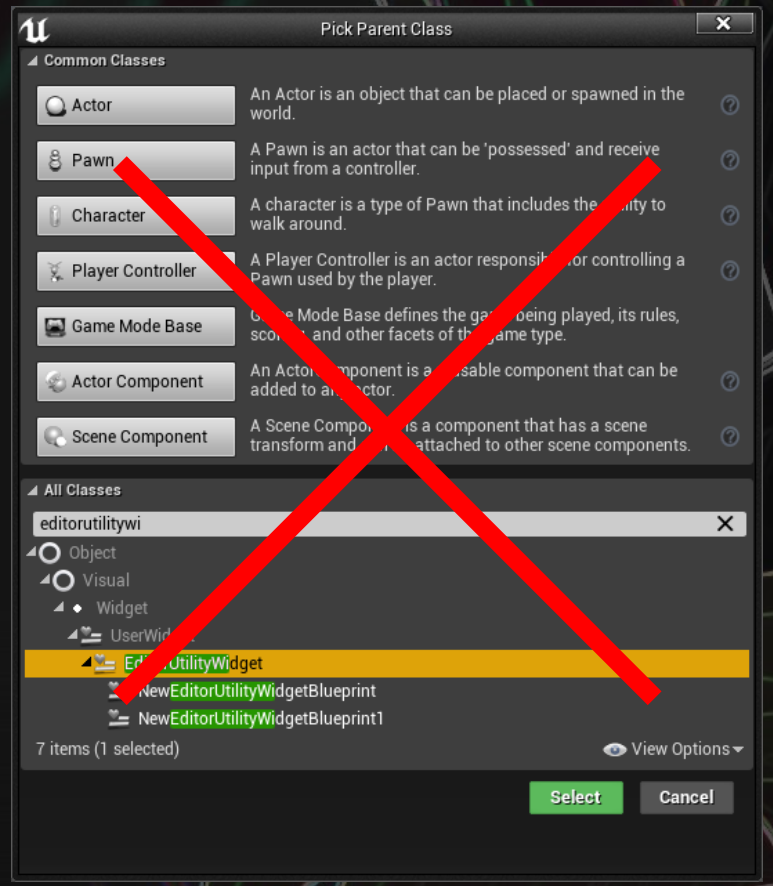


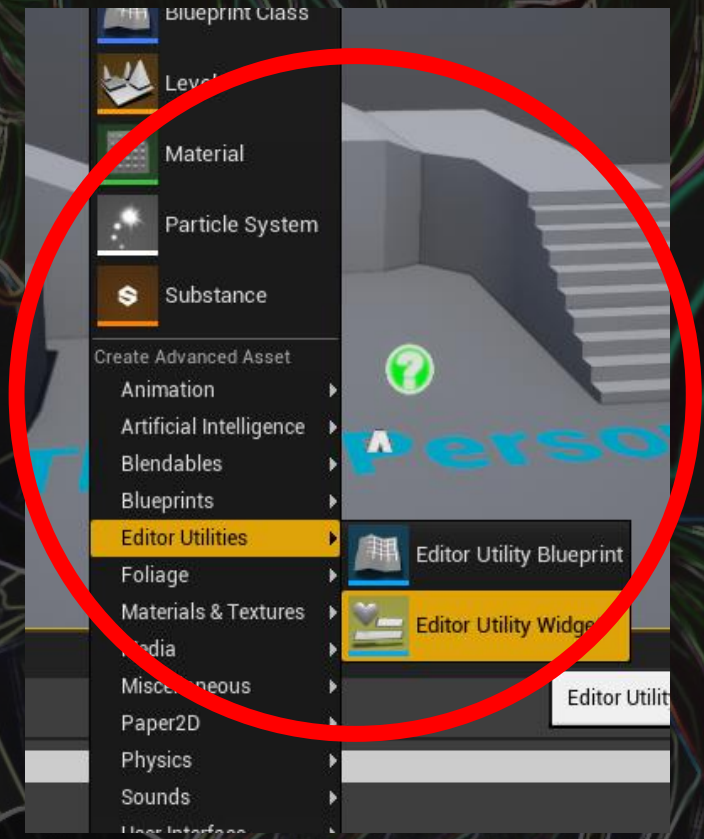


Editor Utility Widgetの罠

その1

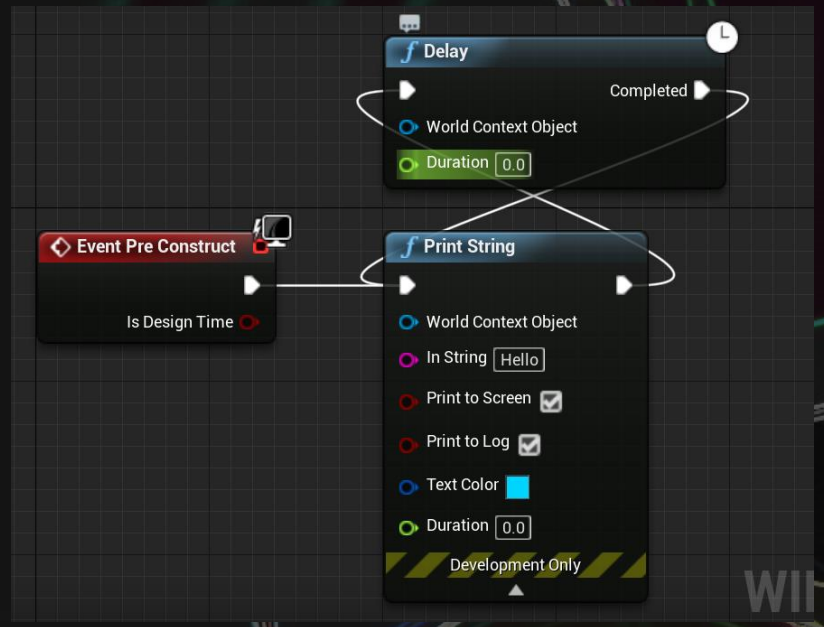
- 通常のBlueprintを作成する方法でEUWを作成した場合は、
  EUWアセット右クリック→「Run Editor Utility Widget」が使えません
  「Editor Utilities」から作成しましょう。
  ※Runが使えない以外は特に問題はなさそうです。



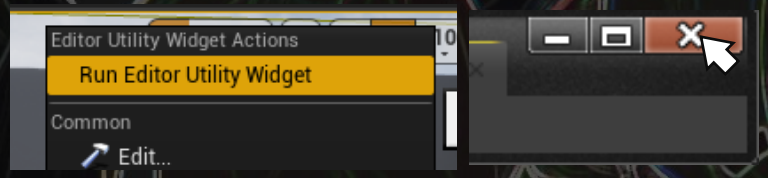


その2

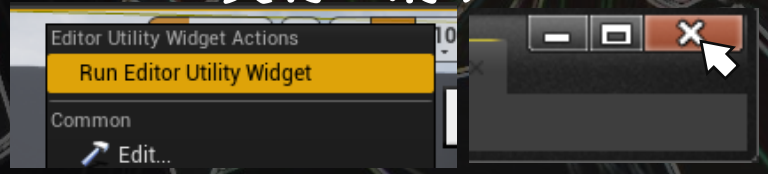
実はEUWを一度実行してEUWを消しても、EUWのインスタンスは残っています。
→**EUWのBPをコンパイルするとリセットされます！**
（もしくは他のGCのタイミングで消える）



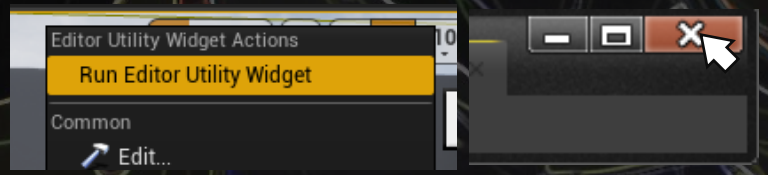
実行&消す



実行&消す

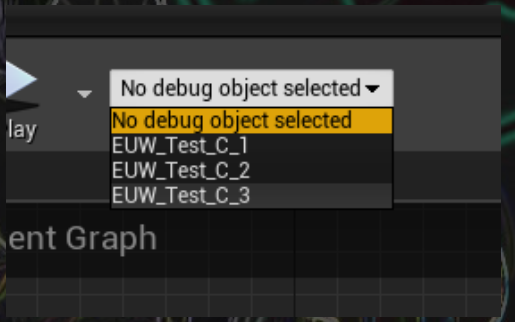


実行&消す



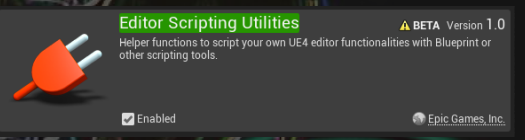
3つ…？

DebugFilterみると
残っているのがわかる

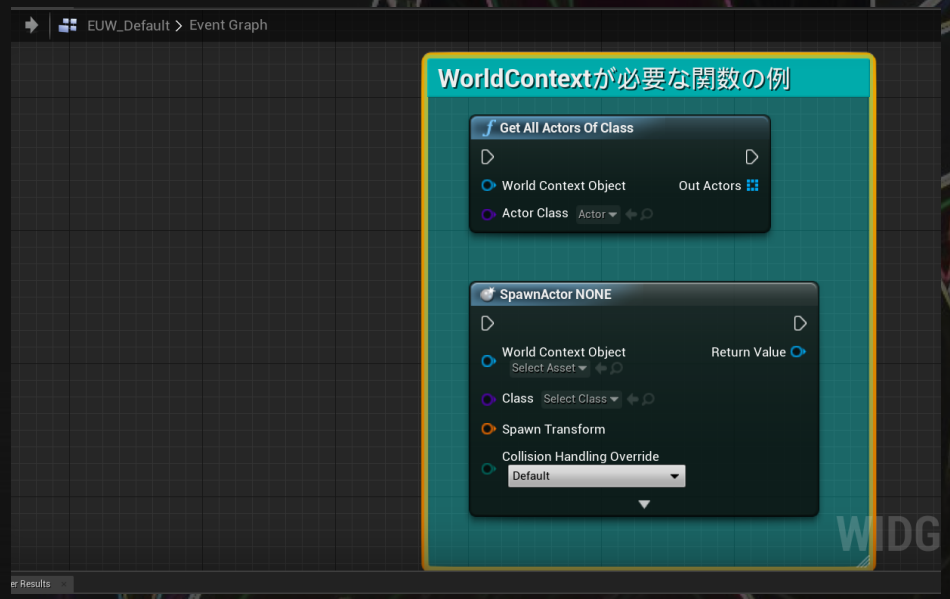


その3

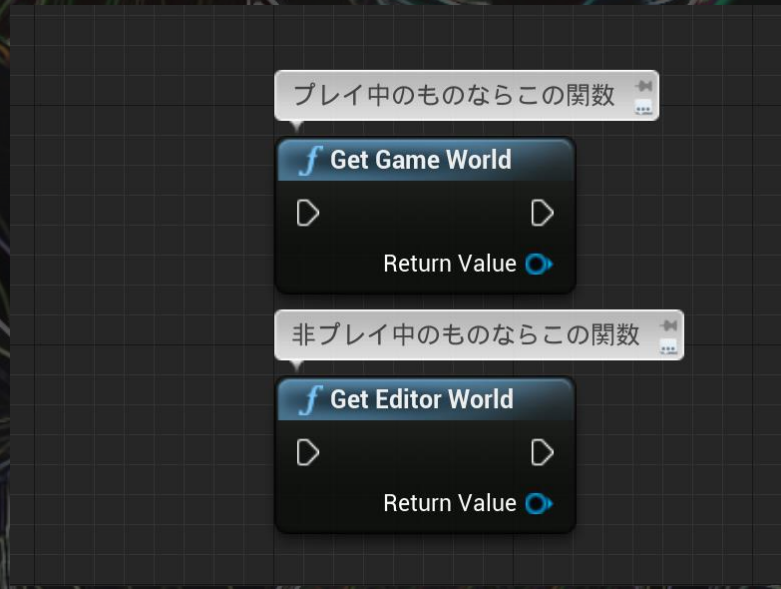
- EUWがActorではないため、WorldContextが必要な関数があります。
  そして、プレイ中か非プレイ中かで、指定するWorldContextが変わってきます。
  →FWorldContext
- プレイ中なら「Get Game World」非プレイ中なら「Get Editor World」
  でWorldContext用のオブジェクトを取得しよう！
  （Editor Scripting Utilitiesプラグインを有効にする必要あり）



WorldContextオブジェクトが必要な関数



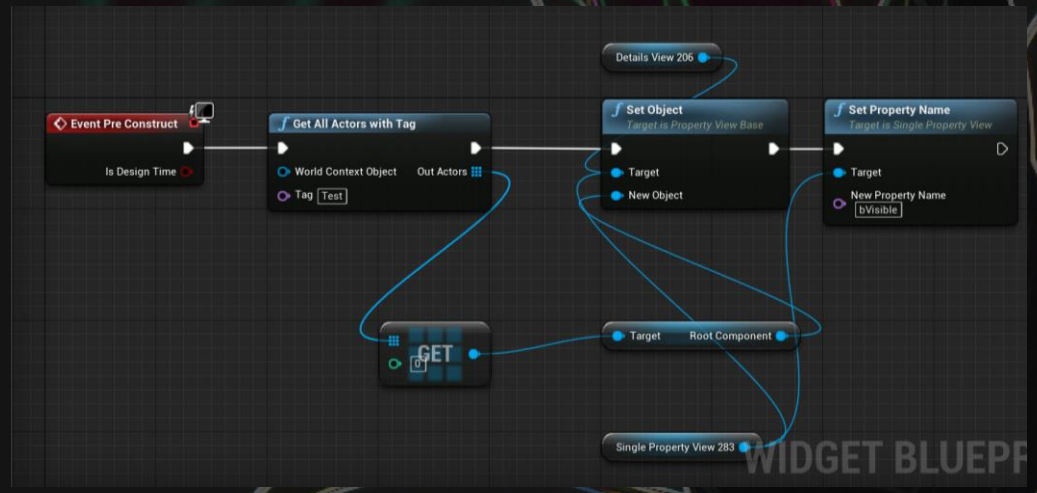
これらの関数の返り値をつなげることで、機能するようになる

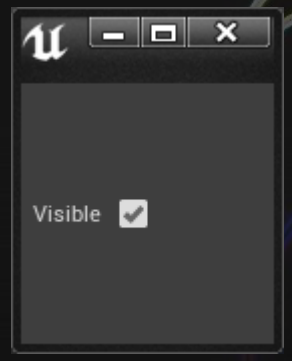


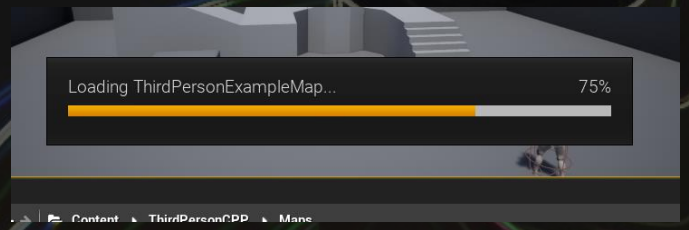
その4

- Actorの参照をもったまま、別のレベルを開いたりしようとすると
  クラッシュします。気を付けましょう。
  ※変数に保持するだけなら問題なさそうでした。
  →Details View系のWidgetの参照が残ってるとクラッシュする。

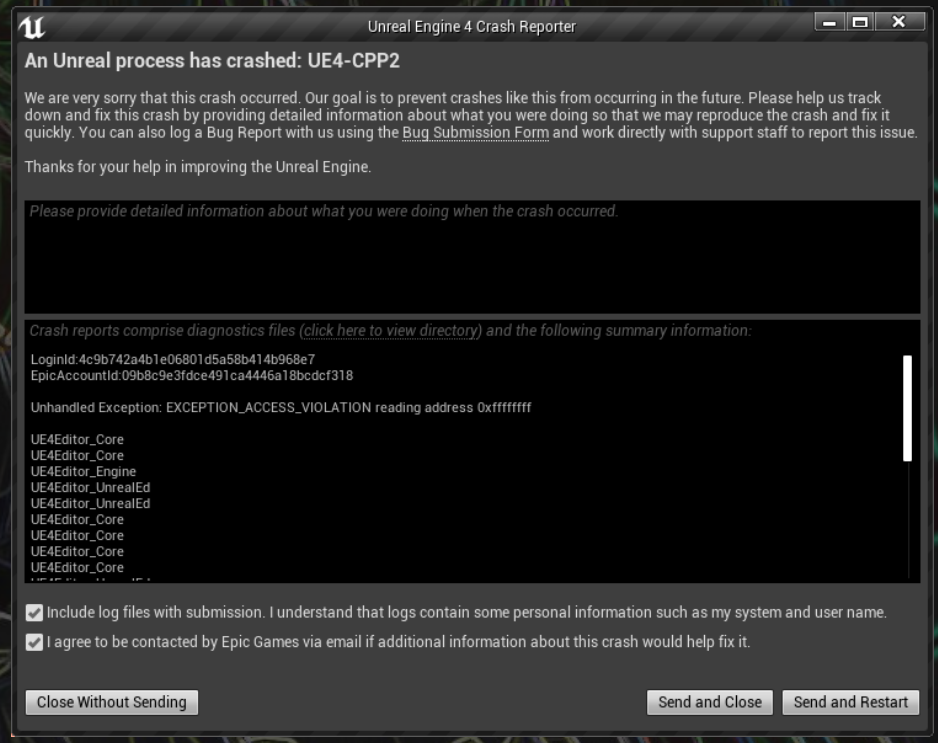
くらっしゅ！！

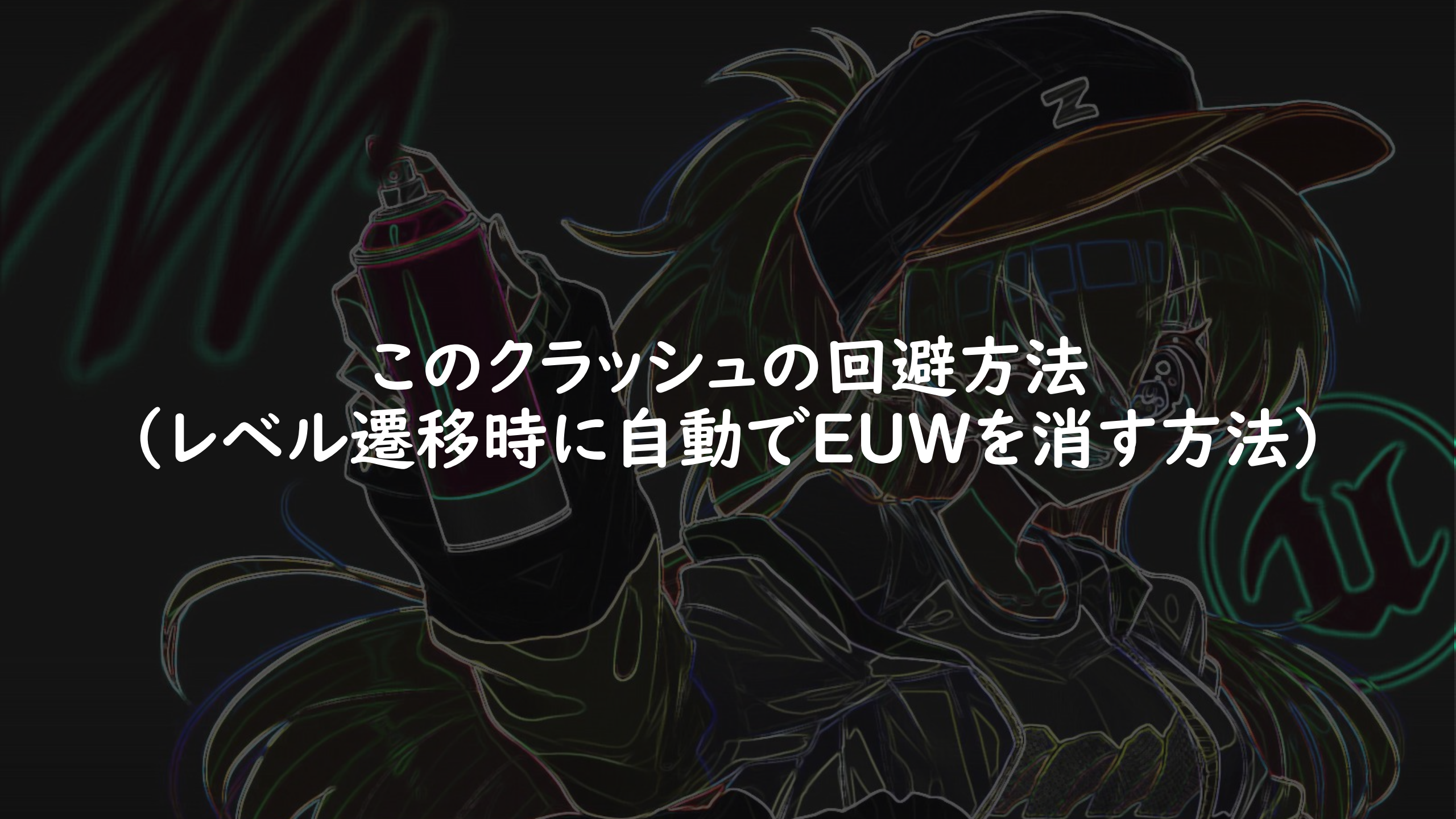




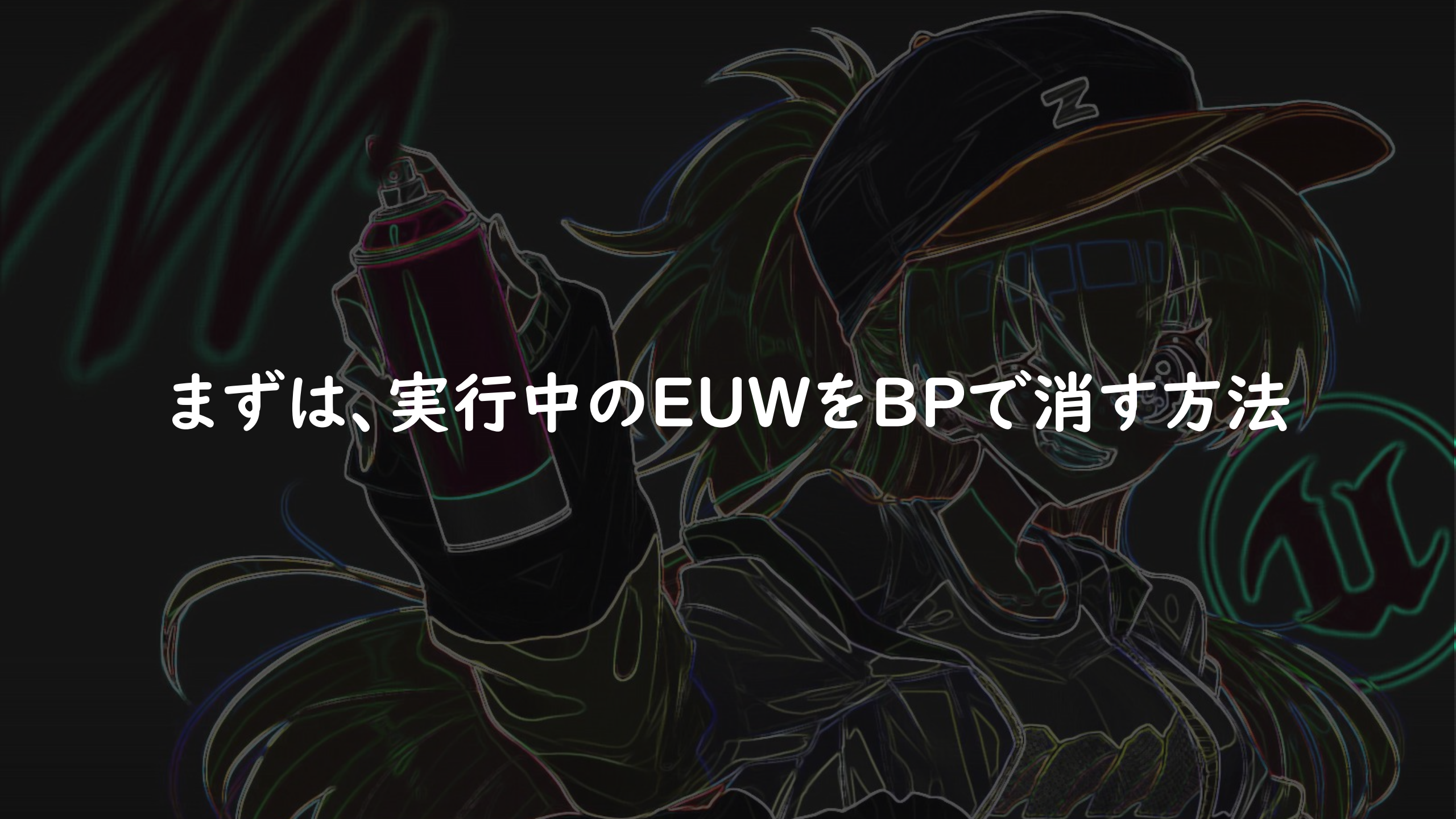


EUWを開いたまま
別マップを開こうとする





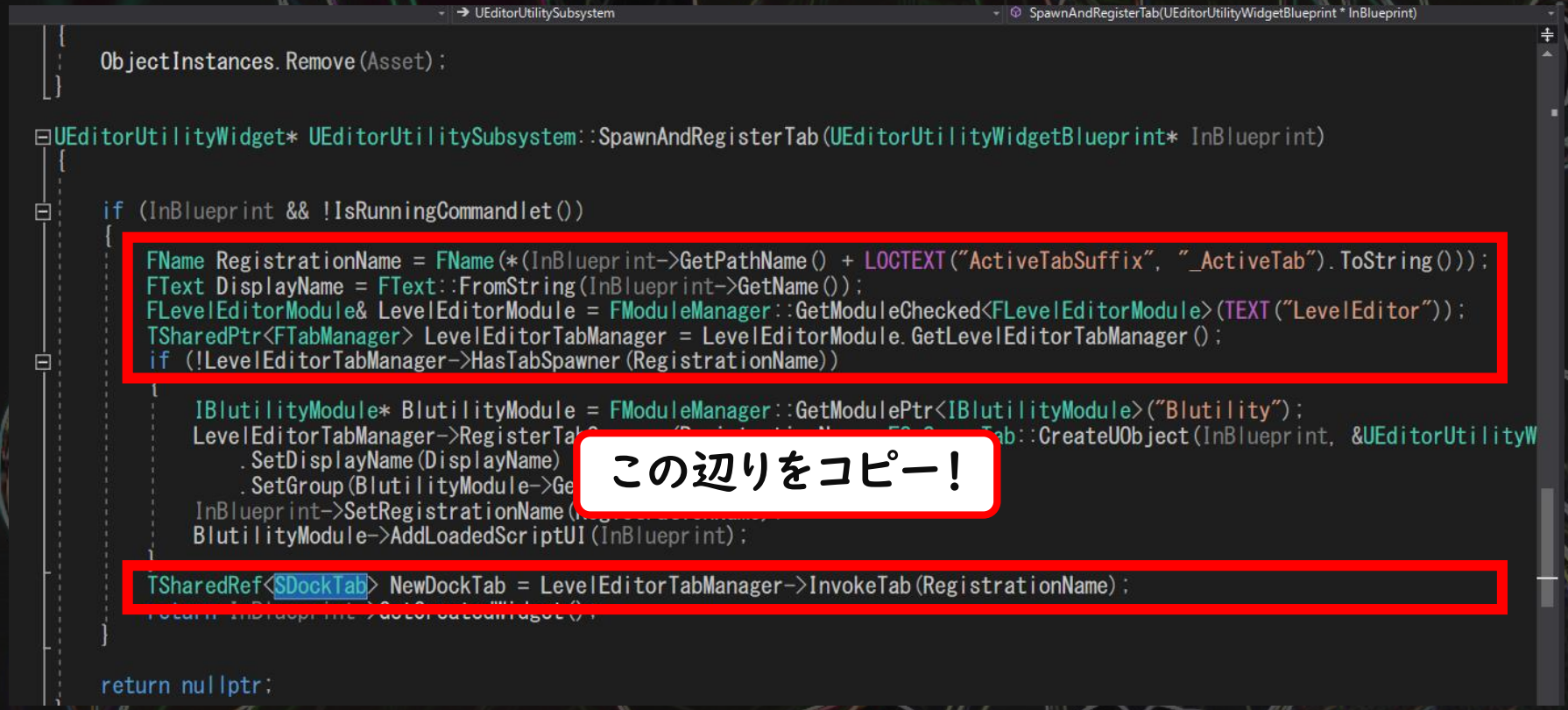
このクラッシュの回避方法
（レベル遷移時に自動でEUWを消す方法）



まずは、実行中のEUWをBPで消す方法

実行中のEUWをBPで消す方法

- EditorUtilitySubsystem.cppの「SpawnAndRegisterTab()」内のタブを取得する部分をコピーして使用する。



実行中のEUWをBPで消す方法

- 自作のFunctionLibraryクラスにペーストする。
  →SDockTabの関数「RequestCloseTab()」を使ってウィンドウを消す。

```
// Fill out your copyright notice in the Description page of Project Settings.

#include "MyBlueprintFunctionLibrary.h"
#include "LevelEditor.h"

#define LOCTEXT_NAMESPACE "MyBlueprintFunctionLibrary"
void UMyBlueprintFunctionLibrary::RemoveEUW(UEditorUtilityWidgetBlueprint* InBlueprint)
{

    if (InBlueprint && !IsRunningCommandlet())
    {
        FName RegistrationName = FName(*(InBlueprint->GetPathName() + LOCTEXT("ActiveTabSuffix", "_ActiveTab").ToString()));
        FText DisplayName = FText::FromString(InBlueprint->GetName());
        FLevelEditorModule& LevelEditorMod
        TSharedPtr<FTabManager> LevelEdito
        if (!LevelEditorTabManager->HasTab
        {
            return;
        }
        TSharedRef<SDockTab> NewDockTab = LevelEditorTabManager->InvokeTab(RegistrationName);
        NewDockTab->RequestCloseTab();
    }

    return;
}
```

LevelEditor.hをinclude

いらないところは消す！

RequestCloseTab()を追加！

実行中のEUWをBPで消す方法

- Header

```
UMyBlueprintFunctionLibrary
 1  // Fill out your copyright notice in the Description page of Project Settings.
 2
 3  #pragma once
 4
 5  #include "CoreMinimal.h"
 6  #include "Kismet/BlueprintFunctionLibrary.h"
 7  #include "Blutility/Classes/EditorUtilityWidgetBlueprint.h"
 8  #include "MyBlueprintFunctionLibrary.generated.h"
 9
10  /**
11   *
12   */
13  UCLASS()
14  class CPP2_API UMyBlueprintFunctionLibrary : public UBlueprintFunctionLibrary
15  {
16      GENERATED_BODY()
17  public:
18          UFUNCTION(BlueprintCallable)
19          static void RemoveEUW(UEditorUtilityWidgetBlueprint* InBlueprint);
20
21  };
22
```

実行中のEUWをBPで消す方法

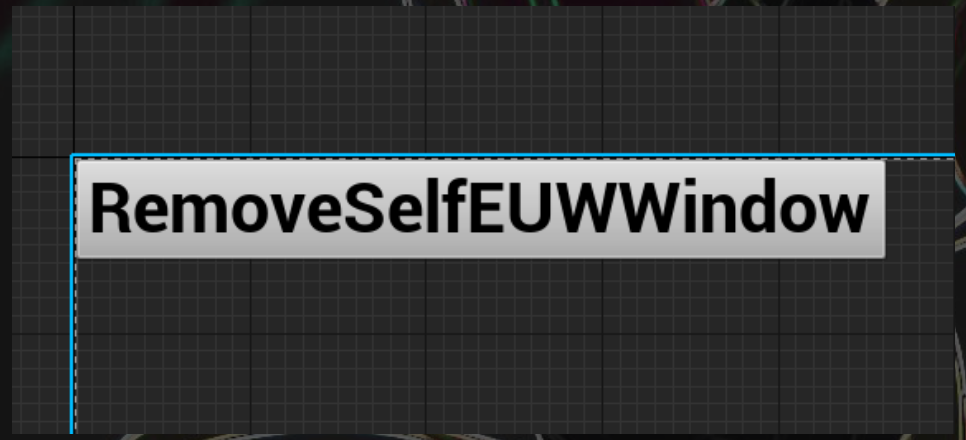
- Moduleは「Blutility,UMGEditor,Slate」の3つを追加

```
public class CPP2 : ModuleRules
{
    public CPP2(ReadOnlyTargetRules Target) : base(Target)
    {
        //Type = TargetType.Editor;
        PCHUsage = PCHUsageMode.UseExplicitOrSharedPCHs;

        //PrivateIncludePaths.Add("Editor/Blutility/Private");

        PrivateIncludePathModuleNames.Add("AssetTools");

        PublicDependencyModuleNames.AddRange(new string[]
            {
                "Core",
                "CoreUObject",
                "Engine",
                "InputCore",
                "HeadMountedDisplay",
                "Blutility",
                "UMGEditor",
                "Slate",
            });
    }
}
```

実行中のEUWをBPで消す方法

- ブループリント側

EUWにButtonを登録



それが押されたら、
自身のEUWアセットを指定し、そのEUWを消す

実行中のEUWをBPで消す方法

・結果

消せた！！

次は、レベルを開く際のイベントをバインド

レベル遷移時に自動でEUWを消す方法

- EUWを継承したクラスを作成

```
#include "CoreMinimal.h"
#include "UObject/NoExportTypes.h"
#include "UObject/ScriptMacros.h"
#include "Blutility/Classes/EditorUtilityWidgetBlueprint.h"
#include "Blutility/Classes/EditorUtilityWidget.h"
#include "Toolkits/AssetEditorManager.h"
#include "Subsystems/AssetEditorSubsystem.h"
#include "MyEUW.generated.h"

UCLASS(Blueprintable,BlueprintType)
class PRESEN_API UMyEUW : public UEditorUtilityWidget
{
    GENERATED_BODY()

        //初期化
        virtual bool Initialize() override;

public:
        //レベルが開いたときなどのイベントをバインドする
        UFUNCTION(BlueprintCallable)
        void BindOnLevelChangeRequestedOpen();
        //呼び出すイベント
        void OnLevelChangeRequest(UObject* Asset);
        //EUWを消す
        UFUNCTION(BlueprintCallable)
        void RemoveEUW(UEditorUtilityWidgetBlueprint* InBlueprint);
        //EUWのBPで作成したアセットの参照（Editorで設定）
        UPROPERTY(EditAnywhere,BlueprintReadWrite)
            UEditorUtilityWidgetBlueprint* EUWBPRef;
```

レベル遷移時に自動でEUWを消す方法

- C++の中身追加部分

レベルを開く際、
UAssetEditorSubsystem::OnAssetEditorRequestedOpen()
がデリゲートされるので、そのイベントを受け取って、EUWを消す

UUserWidget::Initialize()の
オーバーライド
（アセットを開いた時のデリゲートを
バインド）

バインド

デリゲートされる処理の中身
開いたアセットが「レベル（UWorld*）」
であれば、EUWを消す。

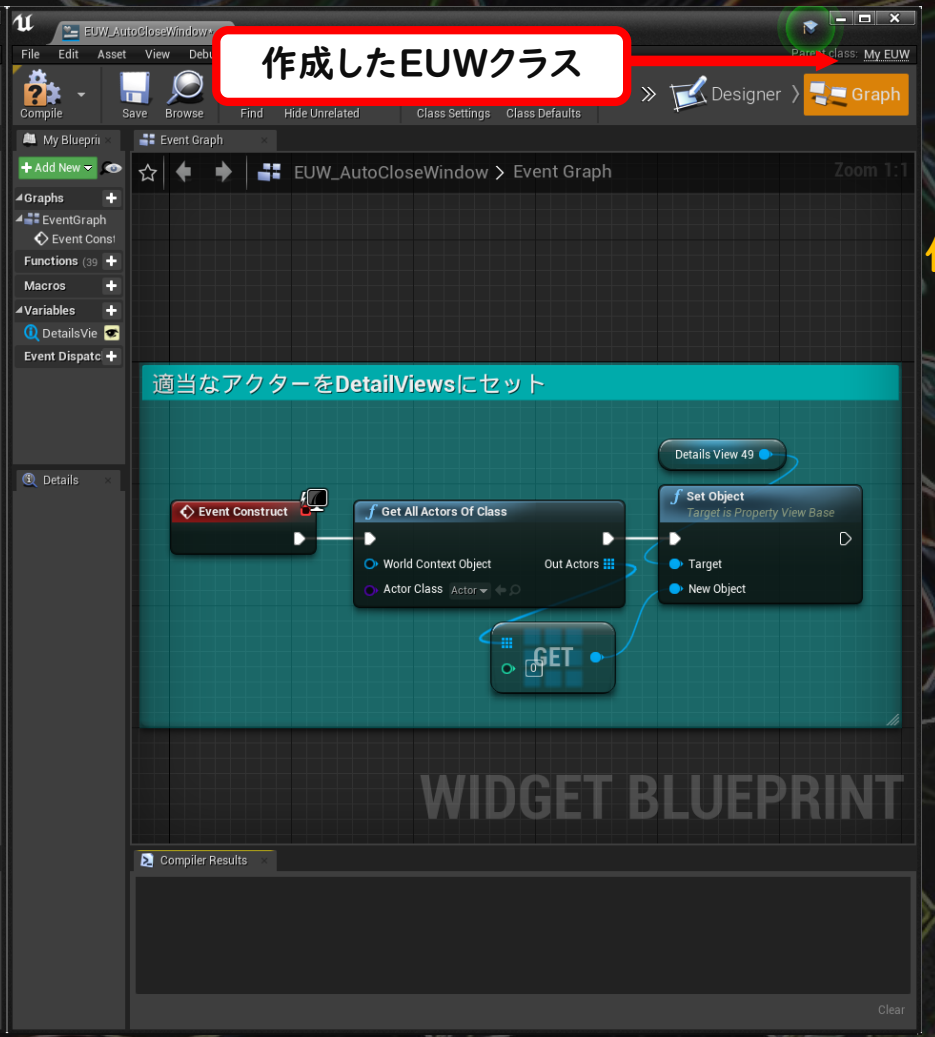
```cpp
#include "MyEUW.h"
#include "Modules/ModuleManager.h"
#include "LevelEditor.h"

#define LOCTEXT_NAMESPACE "MyObject"
bool UMyEUW::Initialize()
{
    //初期化時にバインド
    bool Condition = Super::Initialize();
    BindOnLevelChangeRequestedOpen();
    return Condition;
}

void UMyEUW::BindOnLevelChangeRequestedOpen()
{
    //レベルが開いたときなどのイベントをバインドする
    GEditor->GetEditorSubsystem<UAssetEditorSubsystem>()
    ->OnAssetEditorRequestedOpen().AddUObject(this, &UMyEUW::OnLevelChangeRequest);
}

void UMyEUW::OnLevelChangeRequest(UObject* Asset)
{
    //開いたアセットがマップかどうかを判定
    UWorld* SelectedMap = Cast<UWorld>(Asset);
    if (!SelectedMap)
    {
        return;
    }
    //自身のGeneratedBPClassを保持する変数が有効かどうか
    if (!EUWBPRef)
    {
        return;
    }
    //EUWを消す
    RemoveEUW(EUWBPRef);
}
```

レベル遷移時に自動でEUWを消す方法

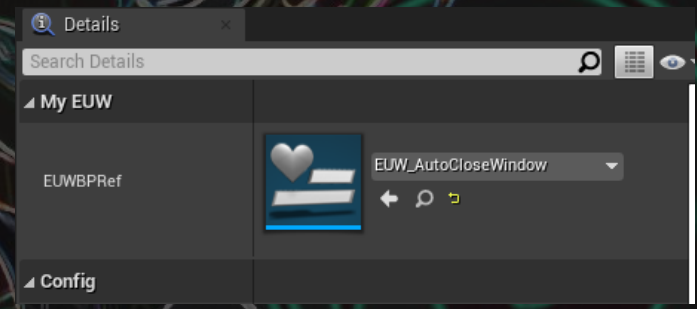
- ブループリント側（例）

通常のEUWクラス

作成したEUWクラス



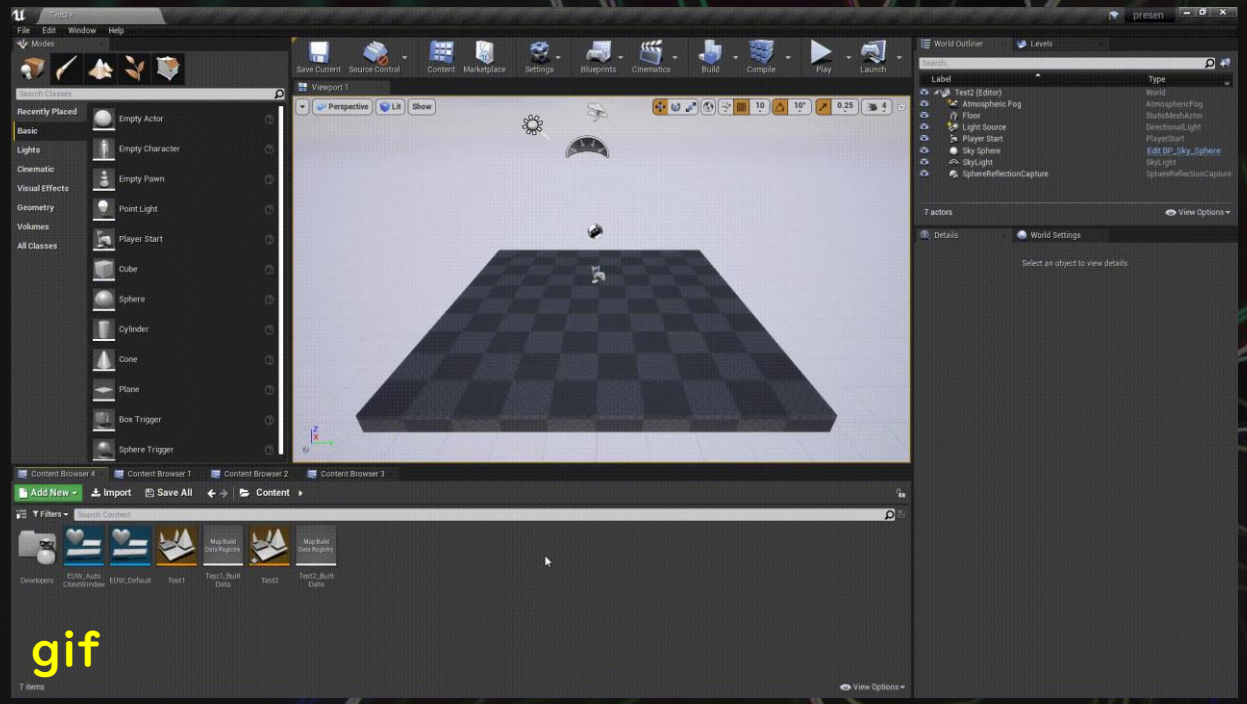
作成したEUWクラスの
ClassDefaultで、
「EUWBPRef」に、
作成した自身のBPアセットを設定する。



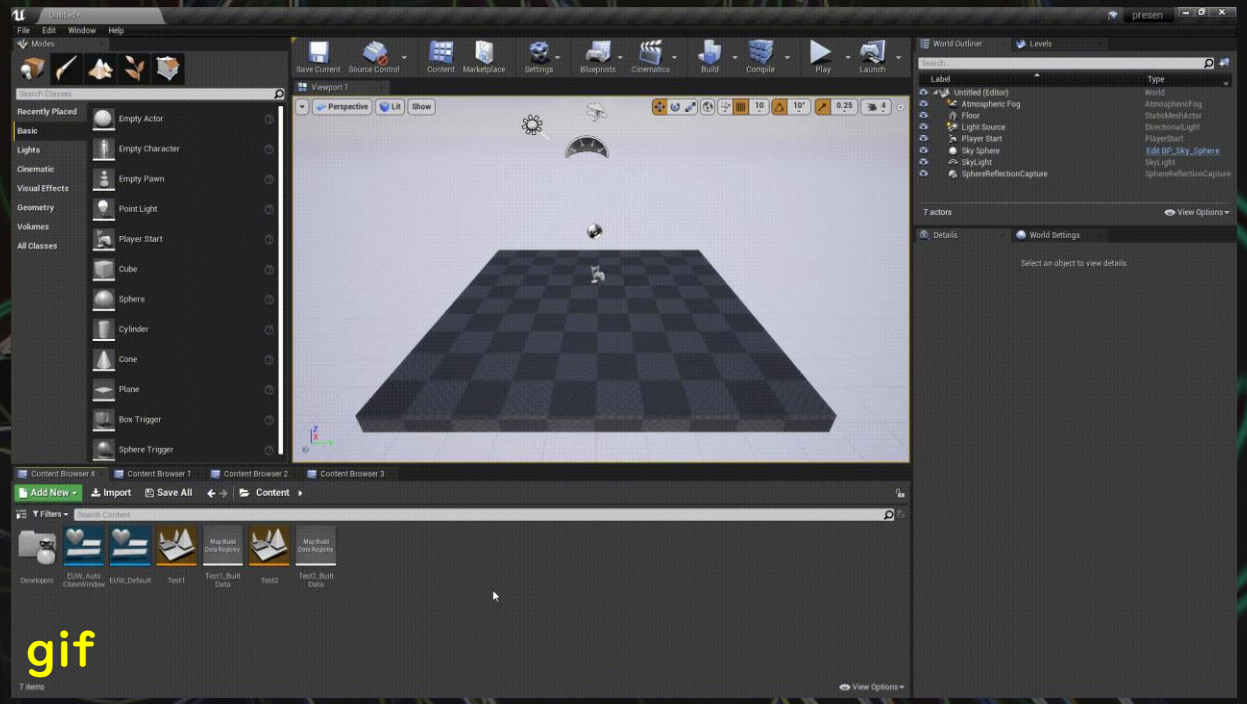
レベル遷移時に自動でEUWを消す方法

・結果

通常のEUWクラス　ばいばいエディター!!



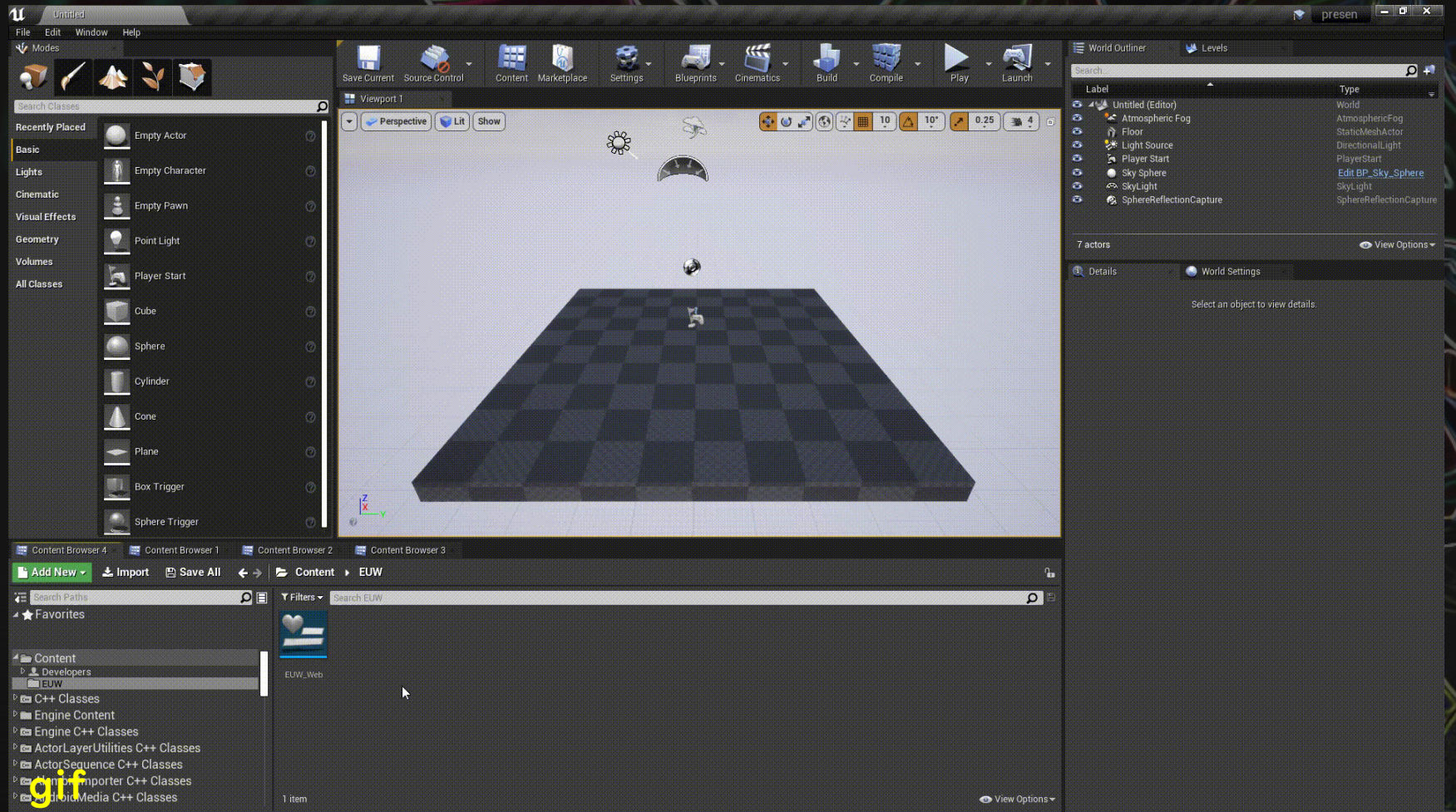
作成したEUWクラス　くらっしゅ回避!!



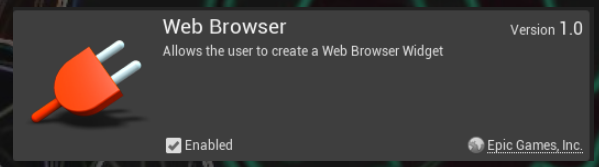
Editor Utility Widgetの制作事例

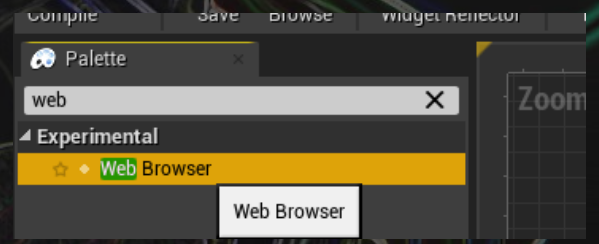
①エディター上でWebを表示

- 「Web Browser」プラグインをつかうことで、EUW上にWebサイトを表示できる。ドキュメントなどのリンクを入れることで、いちいちアクティブウィンドウの切り替えをしなくてもよくなり、すぐにドキュメントに飛んだりできる。



「Web Browser」プラグインを有効にすることで、「Web Browser」ウィジェットを使えるようになる。

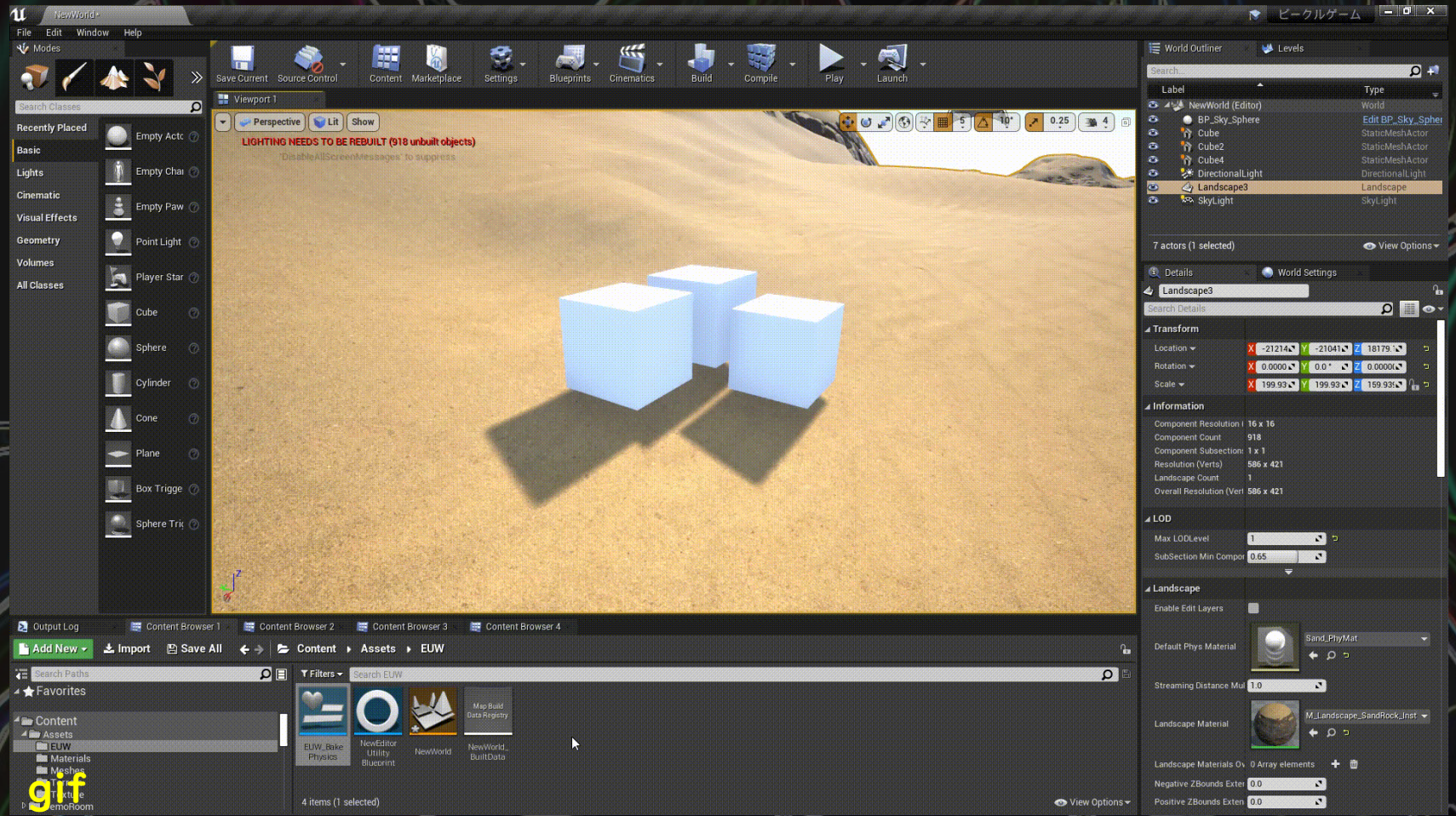




gif

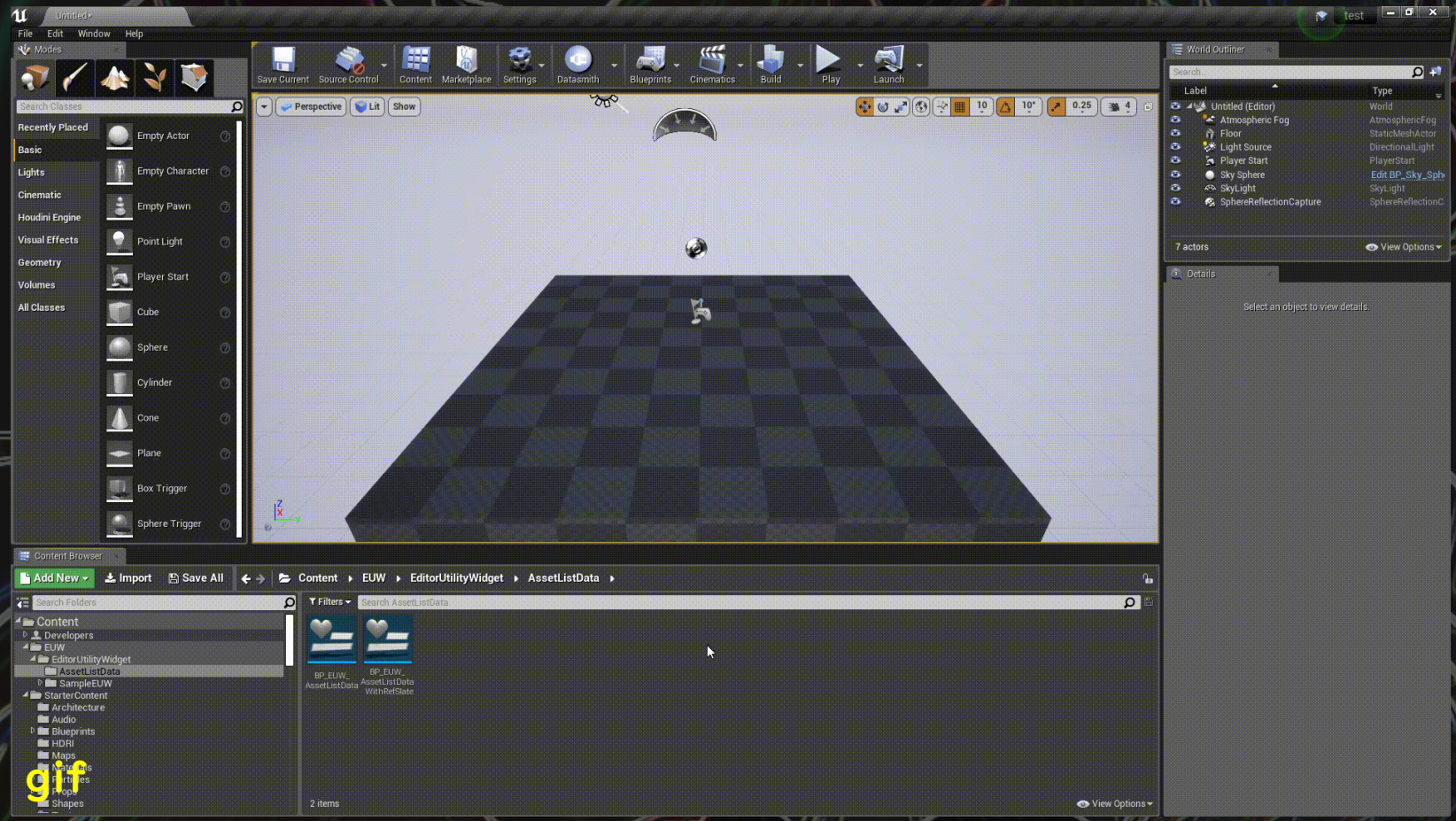
②配置物をPhysicsで設置させる

- LandScapeなどに箱などを設置させるときに、いちいちRotationを細かく設定するのは面倒です。なので、SimulateしたPhysicsの結果をそのままTransformに上書きするEUWを作成。（ブループリントオンリー）



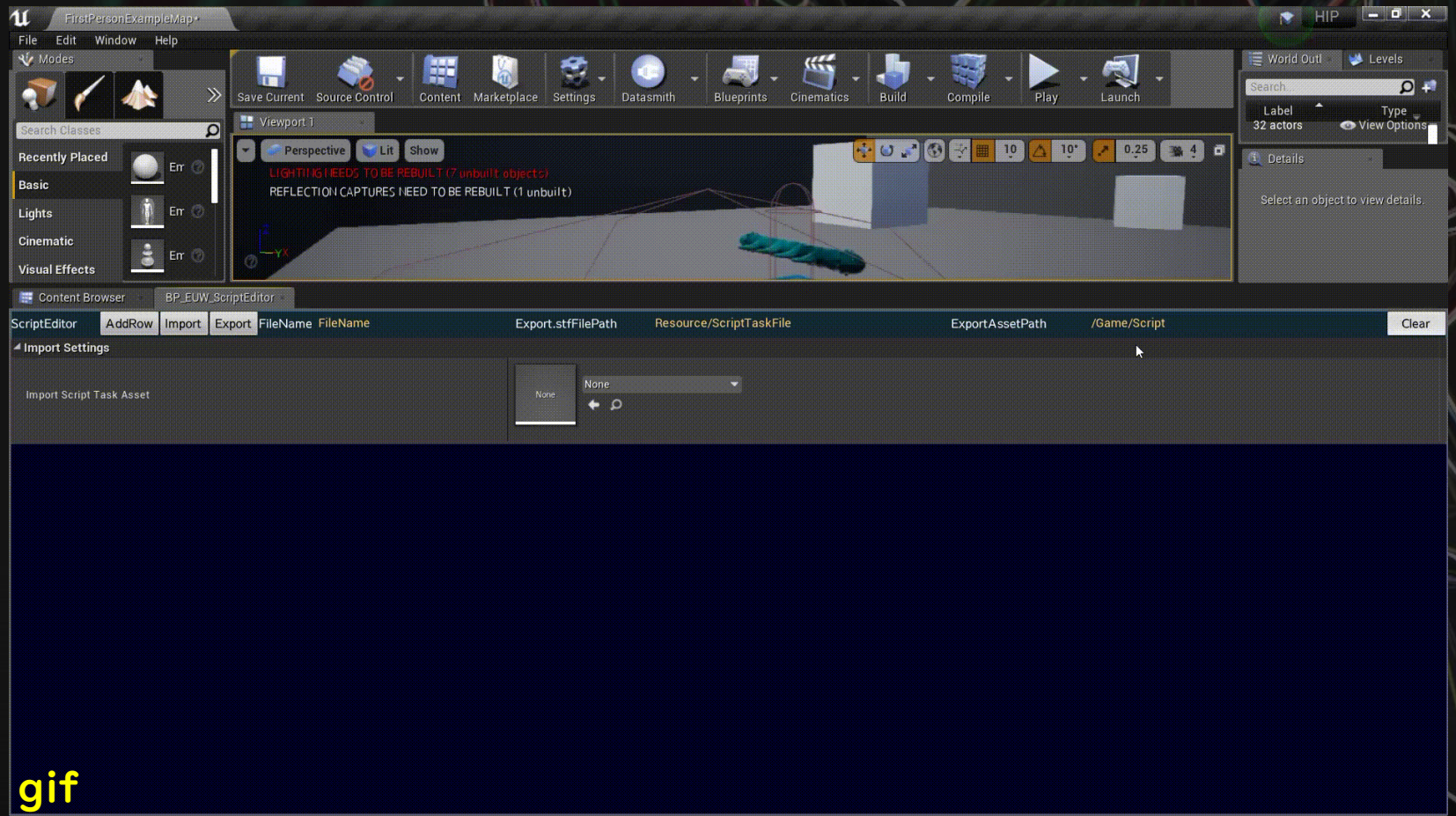
③AssetのFavorite機能

- 選択したアセットをお気に入りに登録できるツール。
  Content Browserにはフォルダのお気に入りしかないので、
  アセットのお気に入りを自作。



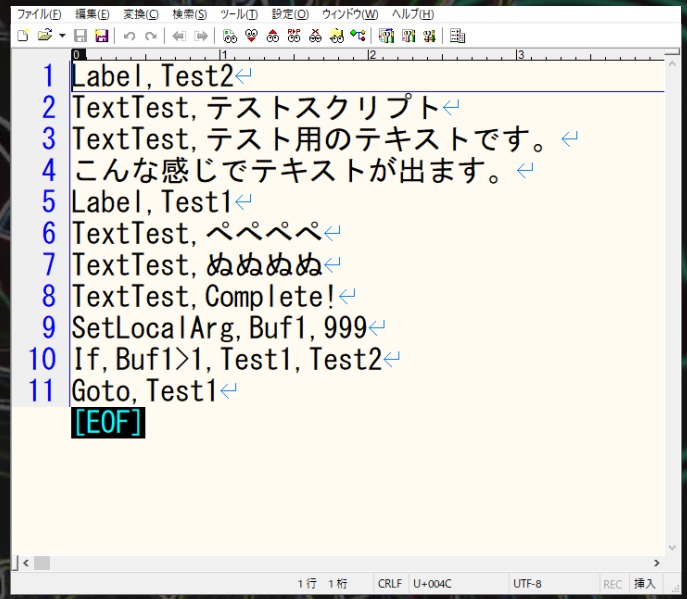
④ScriptEditor

- 自社用に作っているScript用のプラグインのScriptEditor。
  関数名をプルダウンで選んだり、引数のデフォルト値を勝手に入力してくれる。（まだ試作）



gif

このようなテキスト形式で管理！



⑤自動ダンジョン生成ツール

- あらかじめいくつかのマップの部品を作成しておくことで、
  自動でその部品を指定した数・アルゴリズムでつなげてくれるツール。
  レギュレーションをアルゴリズムに取り込んでおり、一瞬でいろんなパターン
  を作成できる。

通路の部品クラスをもったデータアセット

生成するアルゴリズムのクラス

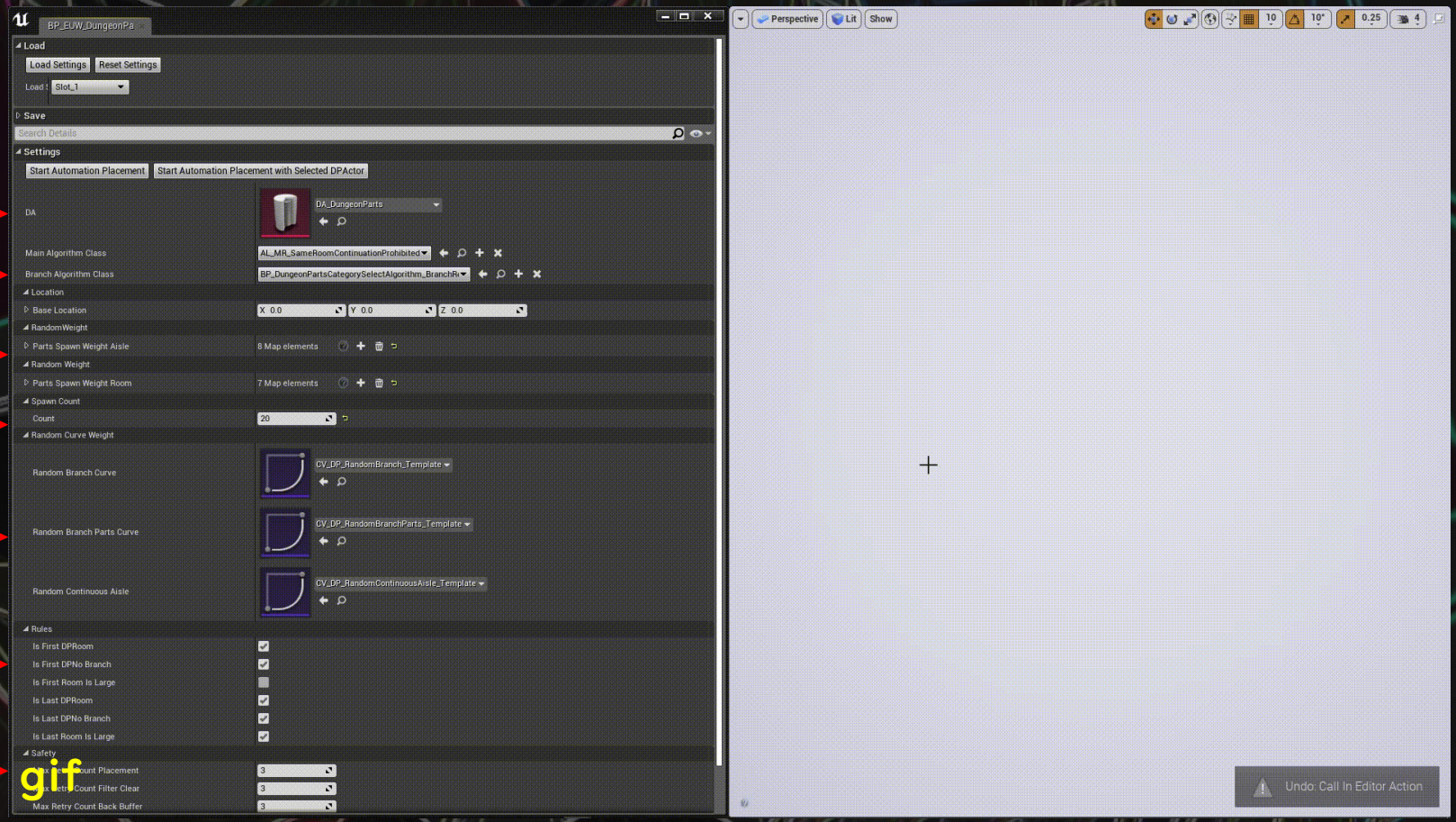
生成する部品クラスの出現ウェイト

生成する部品の数

分岐の数や連続する部品の種類の
ランダムな範囲のカーブアセット

最初や最後の部品を固定するフラグ

生成失敗したときに再試行する回数



以上

ご清聴ありがとうございました