



# Editor Utility Widget Petit Deep Dive

動画アーカイブはこちら! (講演超絶へたくそですみません!)

[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の罨**
  - 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**

今日話すこと  
part2!!

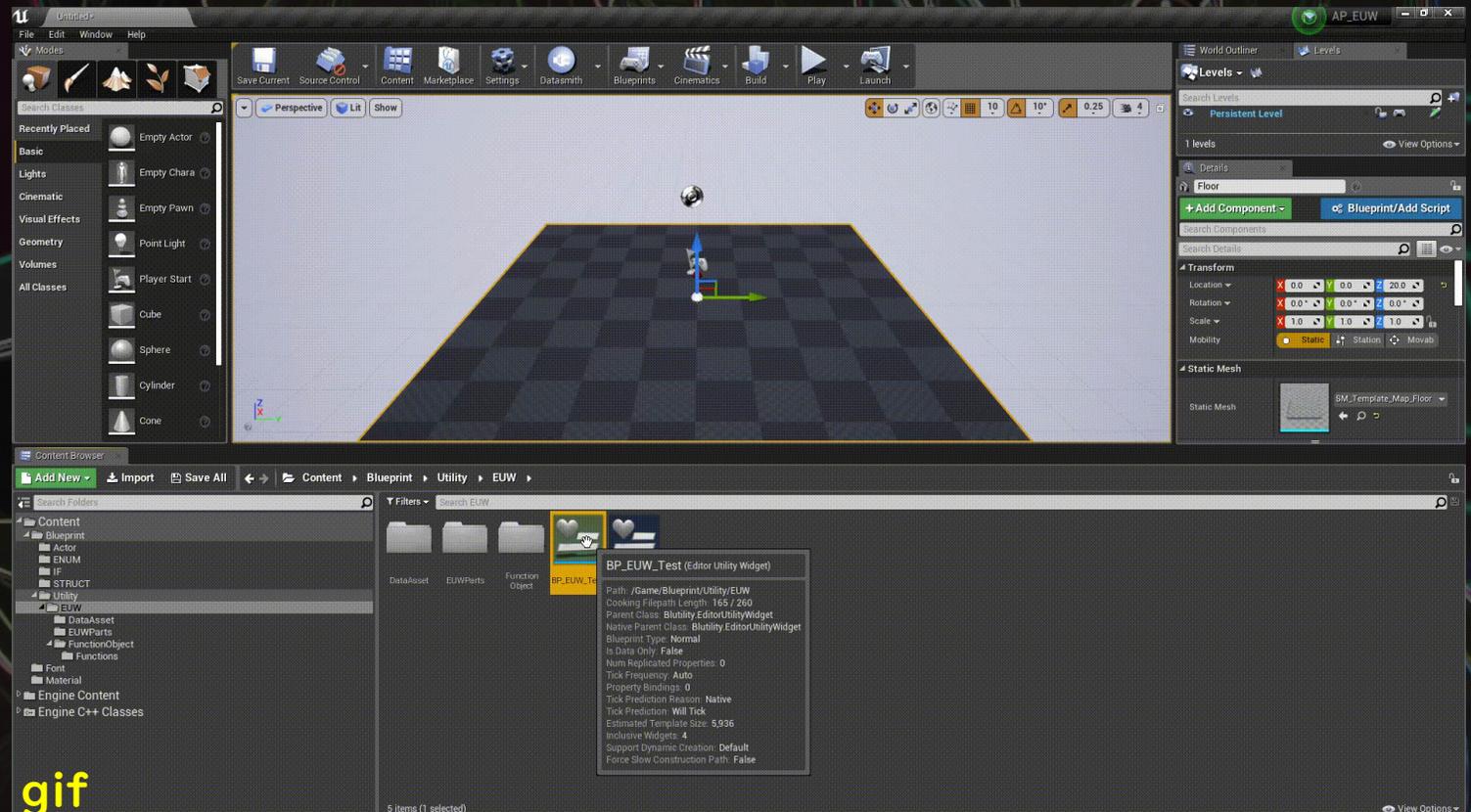
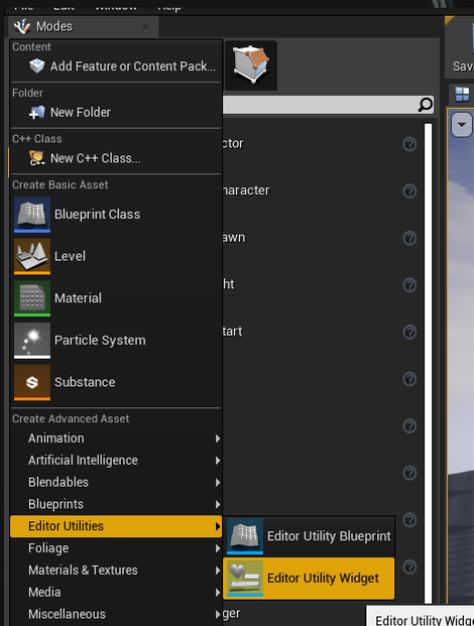


# Editor Utility Widget (EUW)って？

- 「**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。

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

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



# 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!



プチディーブ!



# Details View & Single Property View Widget

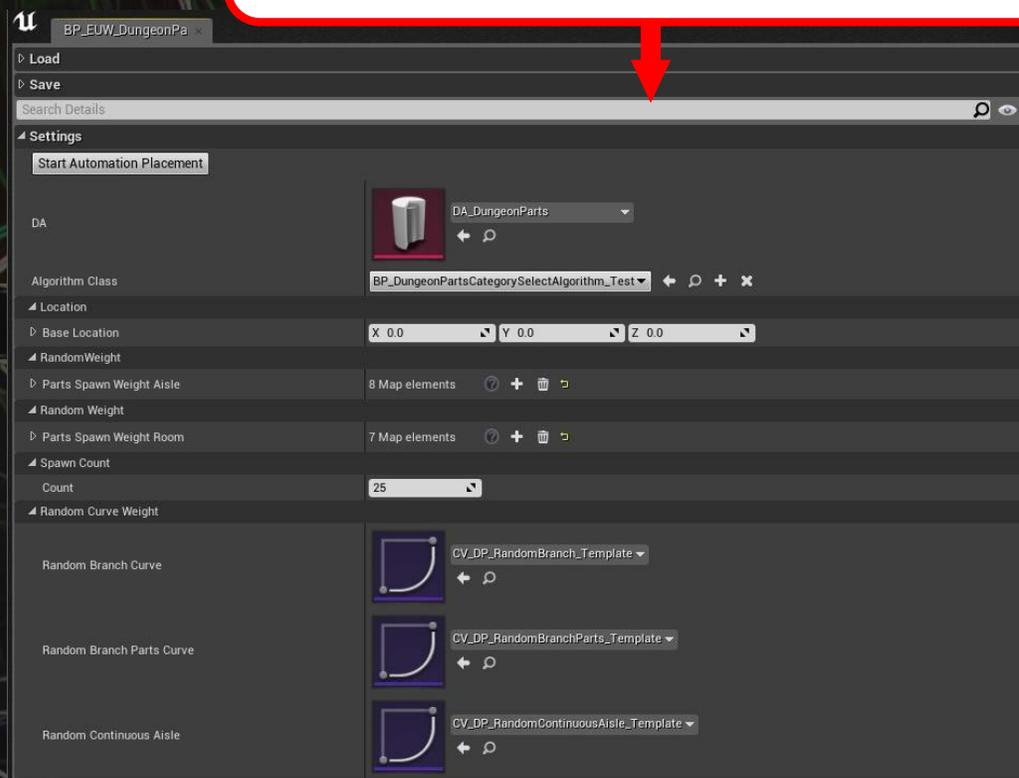
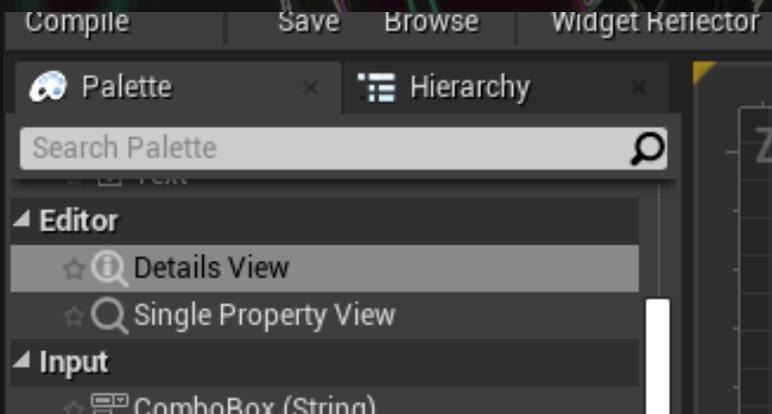


# Details View Widget

# 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。

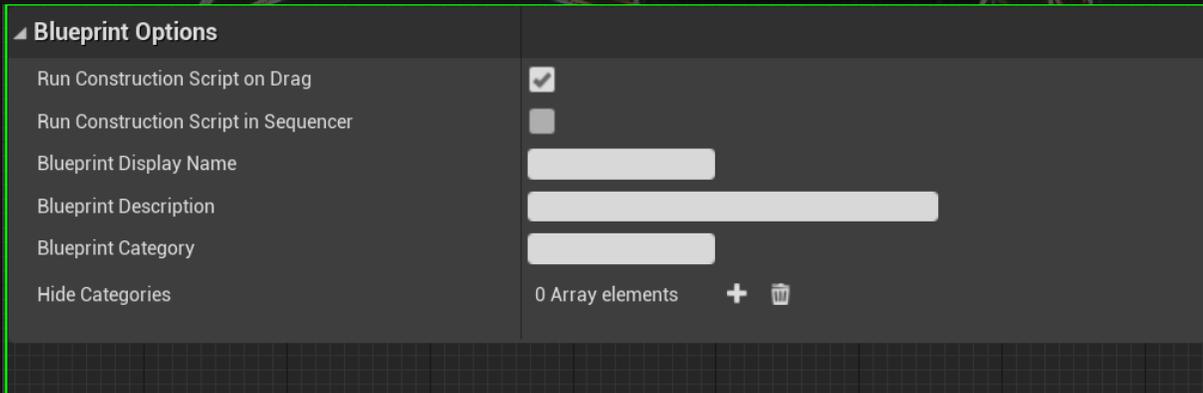
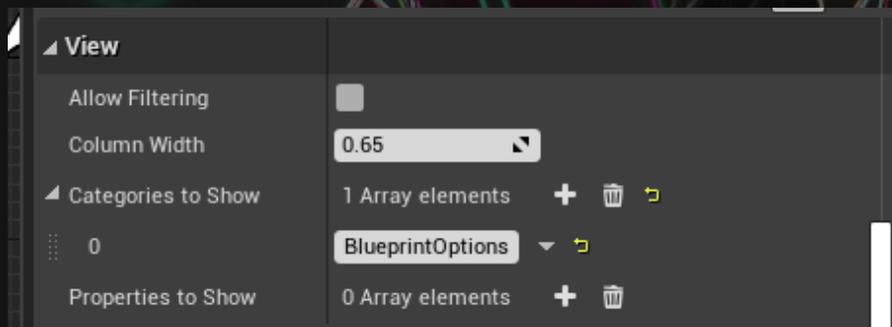
こんな風に、EUWの中にアセットの参照や、パラメーターなどを表示してくれる。  
**超便利!**



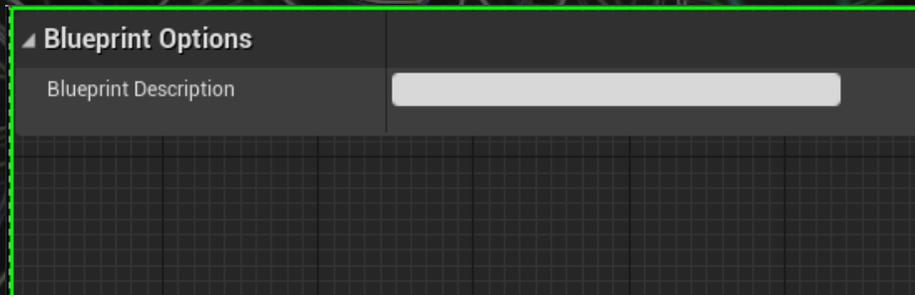
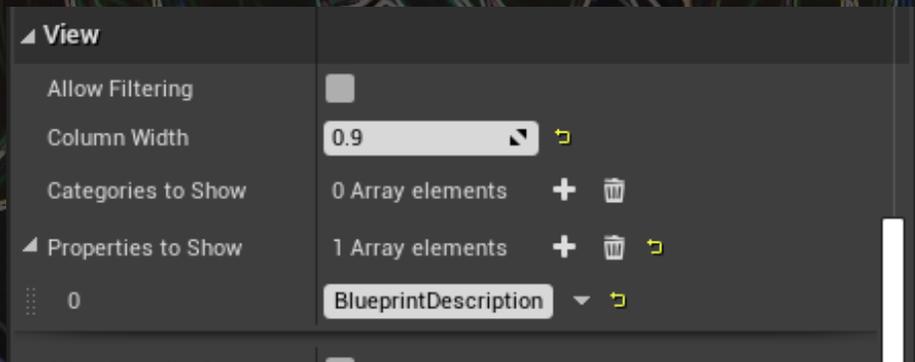
# Details View Widget

- 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

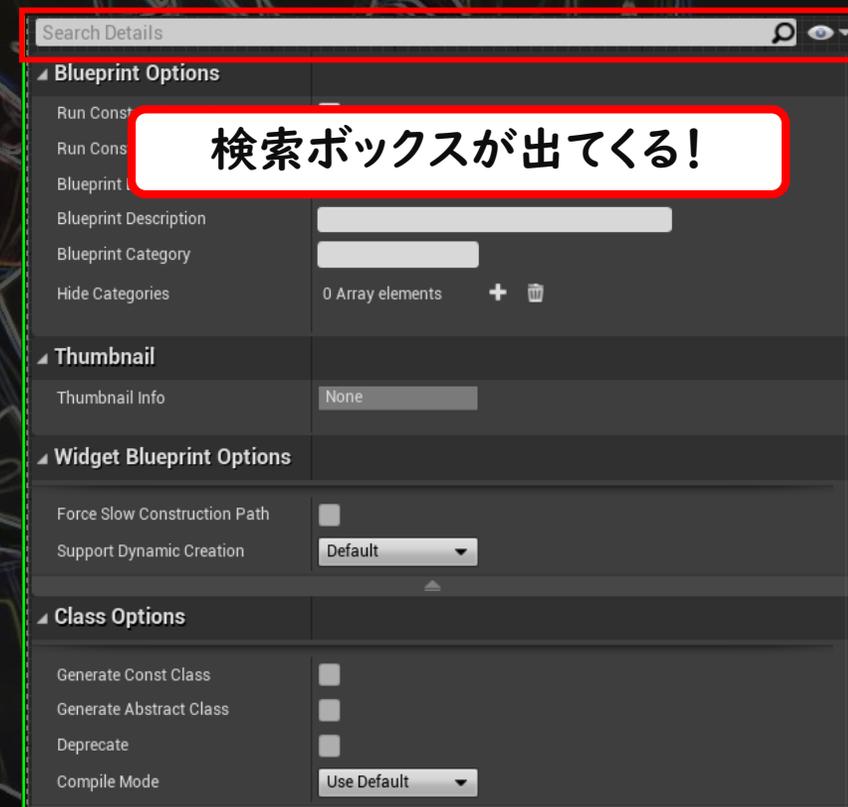
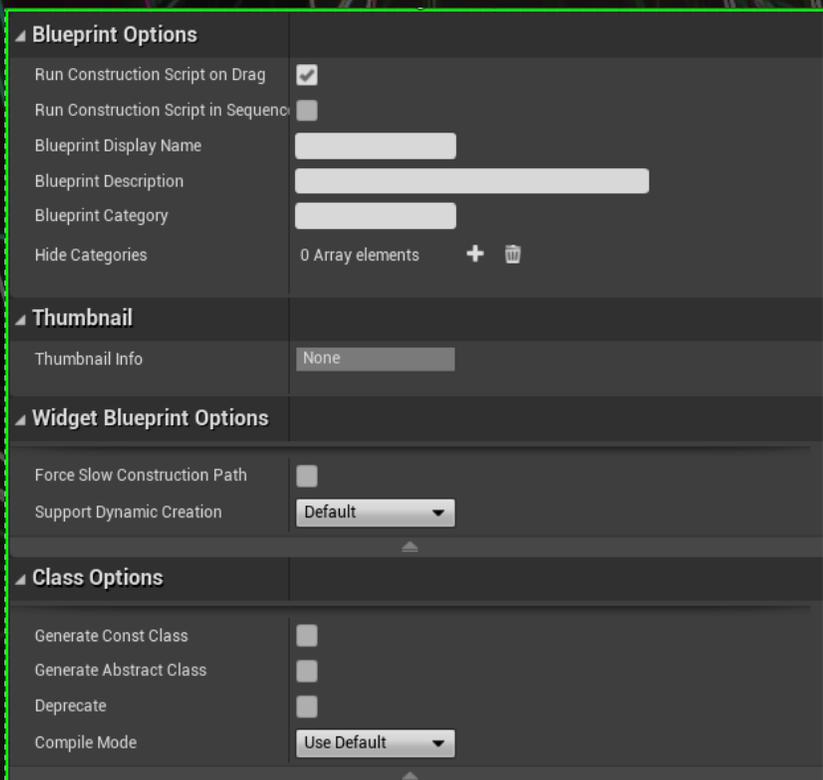
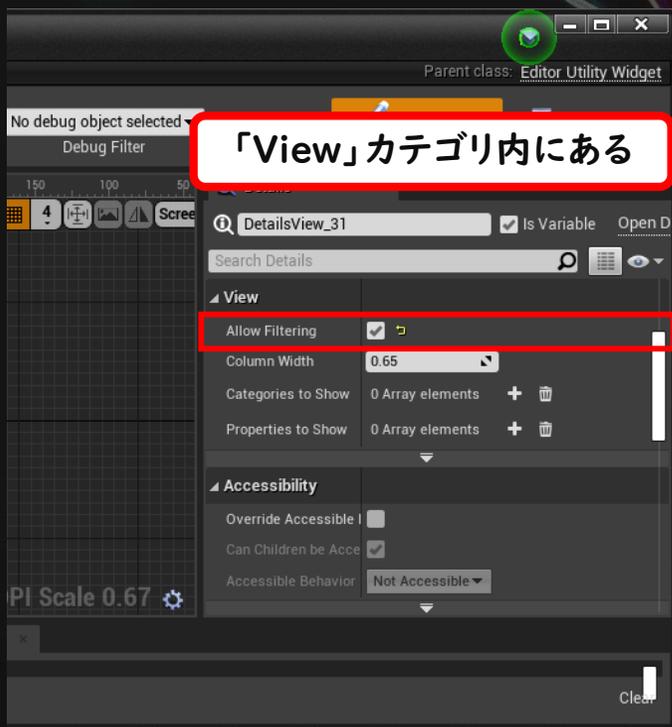


# Details View Widget

- 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



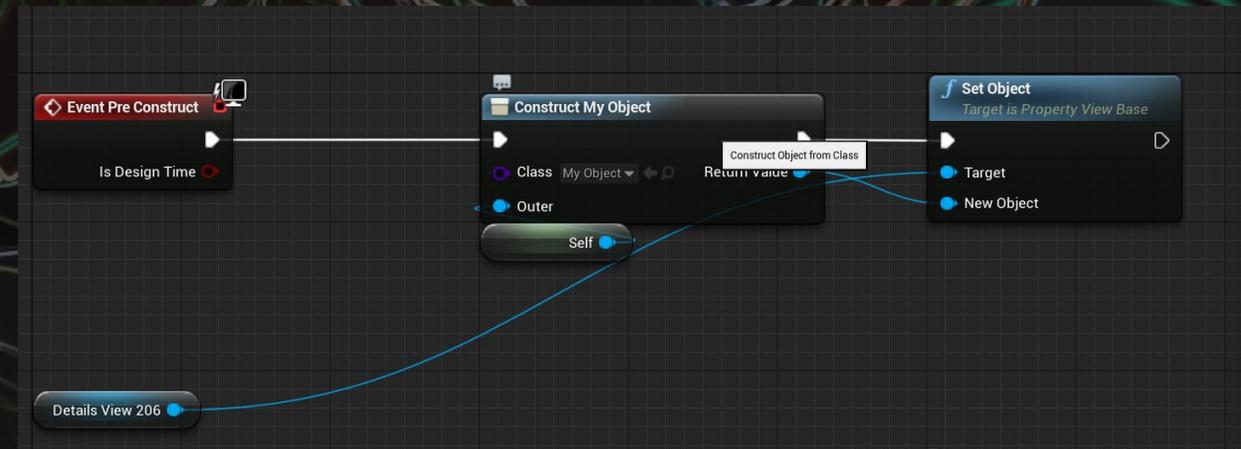
# Details View Widget

- 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     UPROPERTY(EditAnywhere)
19     bool A;
20
21     UPROPERTY(EditInstanceOnly)
22     bool B;
23
24     UPROPERTY(EditDefaultsOnly)
25     bool C;
26
27     UPROPERTY(VisibleAnywhere)
28     bool D;
29
30     UPROPERTY(VisibleInstanceOnly)
31     bool E;
32
33     UPROPERTY(VisibleDefaultsOnly)
34     bool F;
35
36     UPROPERTY(BlueprintReadOnly)
37     bool G;
38
39     UPROPERTY(BlueprintReadWrite)
40     bool H;
41
42     UPROPERTY(Transient)
43     bool J;
44
45     UPROPERTY()
46     bool K;
47
48 }
```

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



# Details View Widget

The screenshot shows the Details View Widget interface. On the left, a hierarchy panel shows the widget tree with 'DetailsView\_206' selected. The main area displays a list of properties A through F. On the right, the 'Force Hidden Property Visibility' checkbox is checked. A red arrow points from the 'Force Hidden Property Visibility' checkbox to a callout box.

**bForceHiddenPropertyVisibility = False**

このように表示される  
プロパティを制御できる

The screenshot shows the Details View Widget interface with the 'Force Hidden Property Visibility' checkbox checked. A red arrow points from the checkbox to a callout box. Below the main list, a second instance of the widget is shown with properties J and K, which are highlighted by a red box.

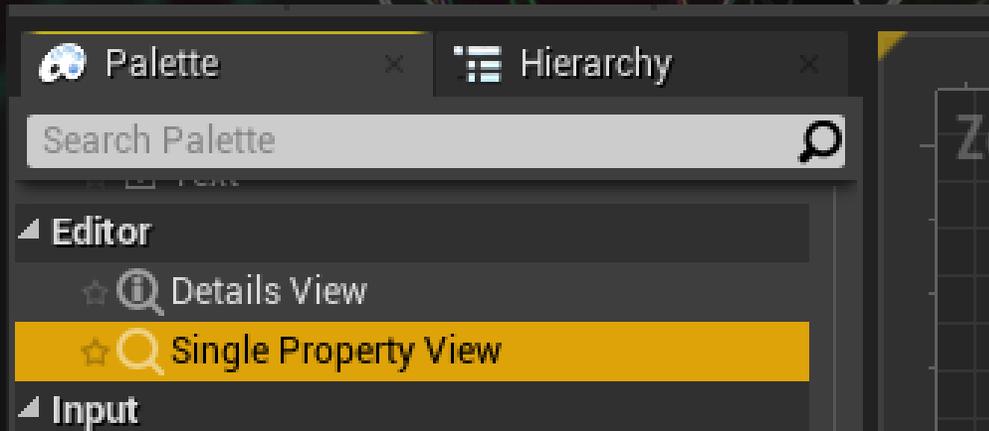
**bForceHiddenPropertyVisibility = True**

# Single Property View Widget

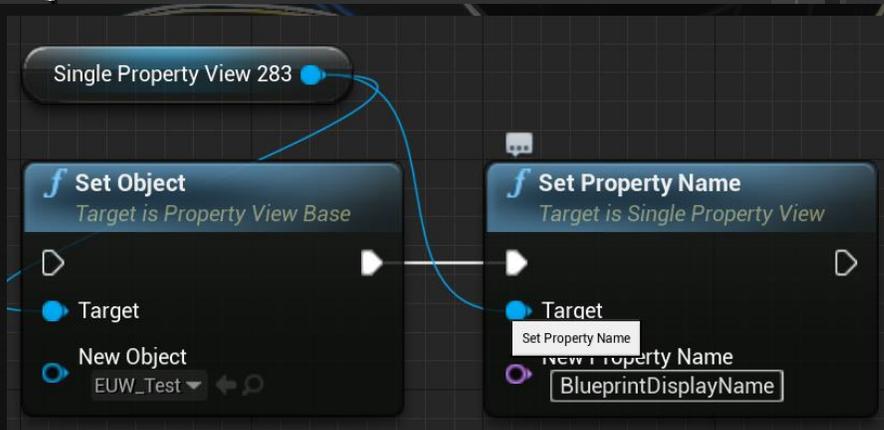
The background features a stylized, sketch-like illustration of a person with spiky hair, wearing a dark cap with a white 'Z' logo. The person is holding a spray can in their right hand. The entire scene is rendered in a dark, monochromatic style with white and light blue outlines and highlights. In the bottom right corner, there is a circular logo containing a stylized, abstract symbol.

# 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!

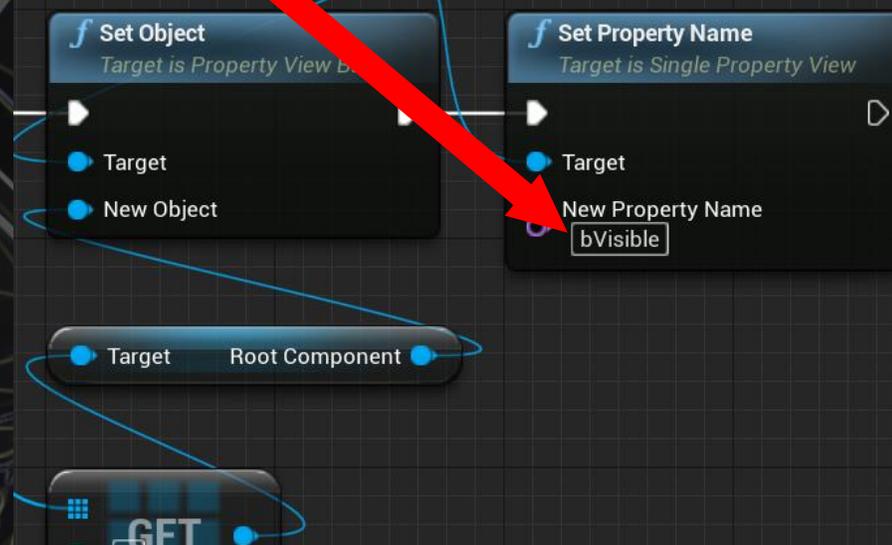
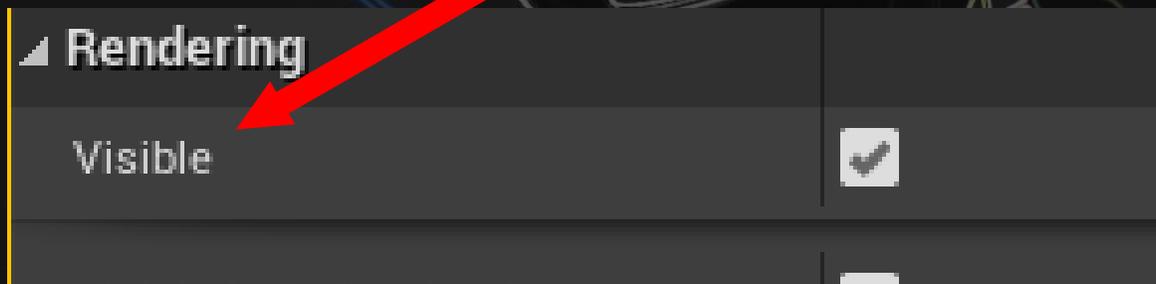


Blueprint Display Name

# 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。

Editorで表示される名前じゃなく  
実際の変数名で指定するのを気を付ける!



# Single Property View Widget

- 「NameOverride」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)

びじふる

Property Name: bVisible

Name Override: **びじふる**

Auto Load Asset:

Accessible Behavior: Not Accessible

Behavior: Tool Tip Text

EUW\_Web

Visible

Visible

表示名が変化!

EUW\_Web

びじふる

# 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

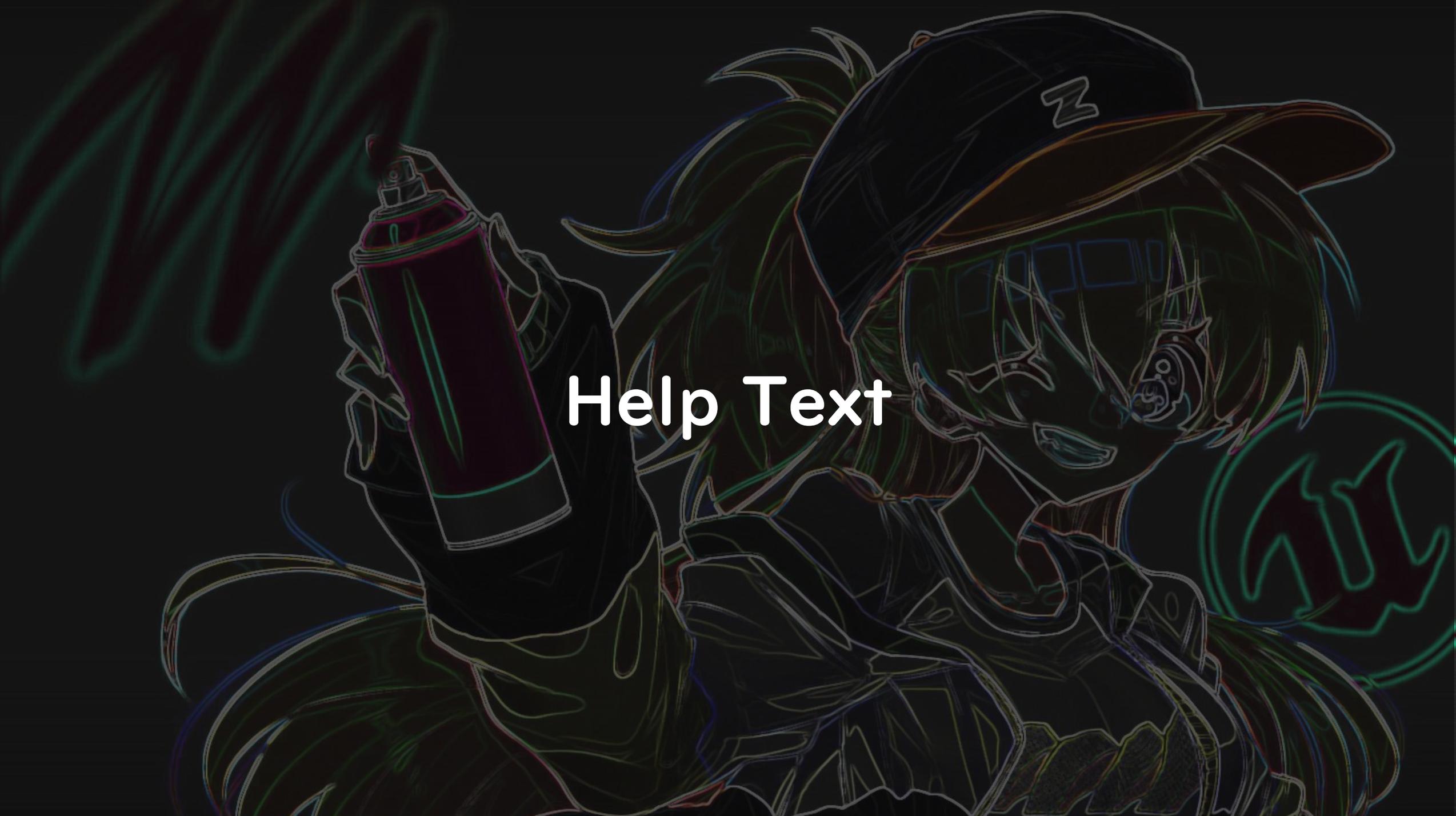
RootComponentが持っている変数  
「RelativeLocation」を設定

FVectorとかも構造体だから  
ダメなんだ…

```
55 else if (PropertyName == NAME_None)
56     MissingWidgetText = FPropertyViewHelper::UndefinedPropertyText;
57
58 else
59     UProperty* Property = ViewedObject->GetClass()->FindPropertyByName(PropertyName);
60     if (Property == nullptr)
61         MissingWidgetText = FPropertyViewHelper::UnknownPropertyText;
62     else if (!Property->HasAllPropertyFlags(CPF_Edit))
63         MissingWidgetText = FPropertyViewHelper::InvalidPropertyText;
64     else if (Cast<UStructProperty>(Property) || Cast<UArrayProperty>(Property)
65             || Cast<UMapProperty>(Property) || Cast<USetProperty>(Property))
66         MissingWidgetText = FPropertyViewHelper::UnsupportedPropertyText;
67     else
68         FPropertyEditorModule& PropertyEditorModule = FModuleManager::GetModuleChecked<FProperty
69         FSinglePropertyParams SinglePropertyParams;
70         SinglePropertyParams.NameOverride = NameOverride;
```



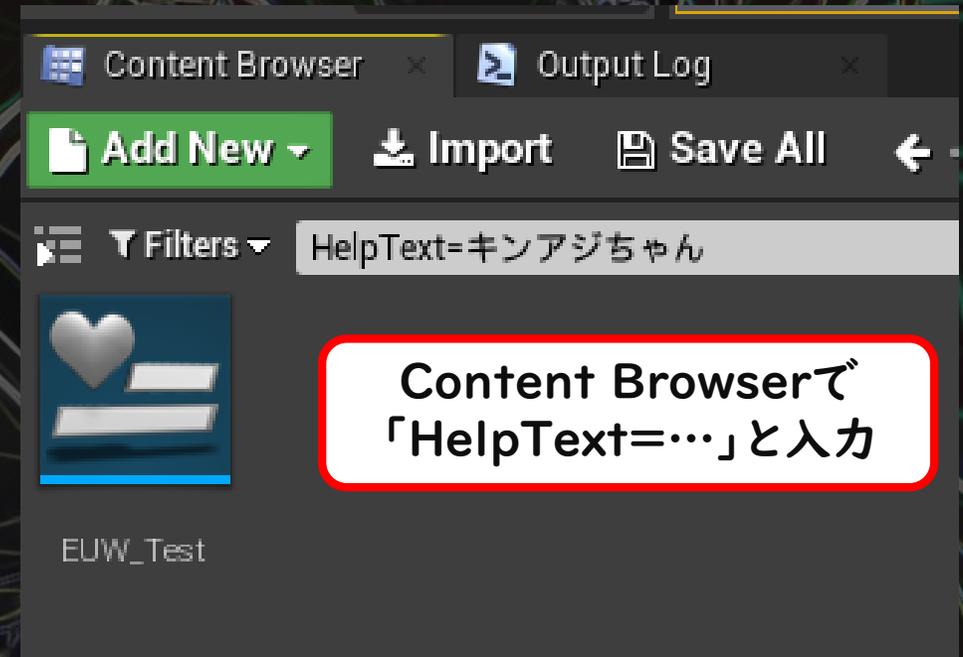
反映されない



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)

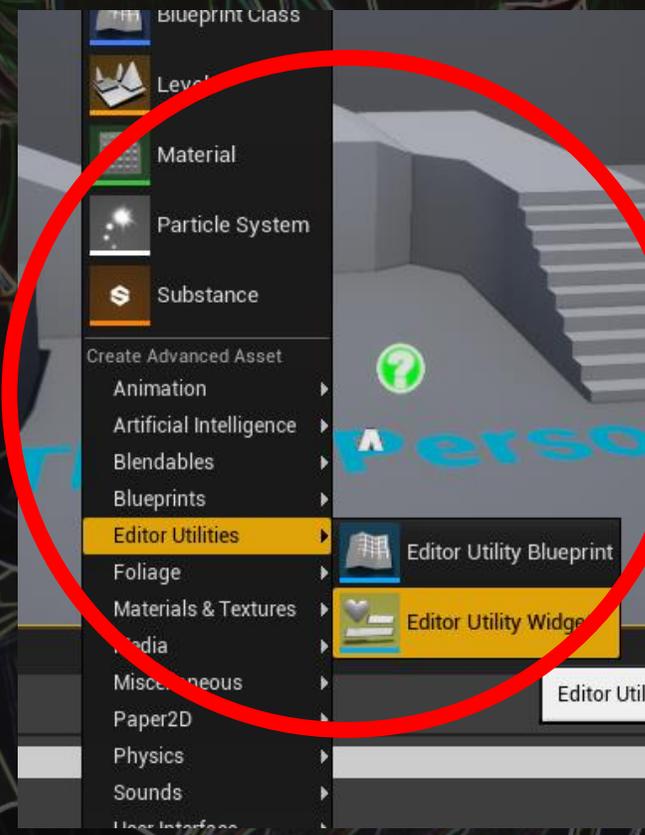
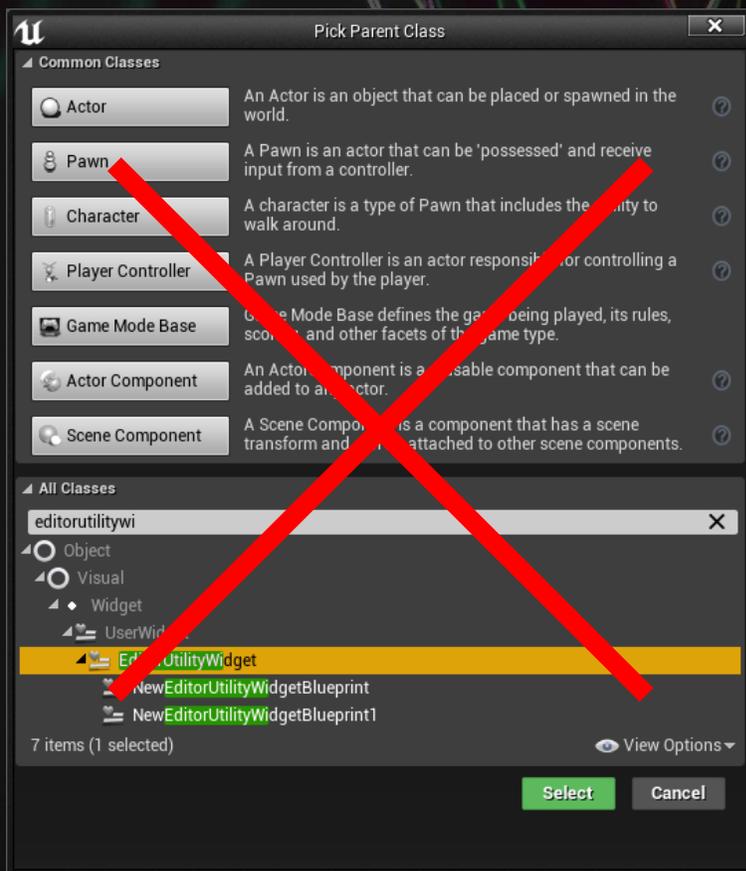




# Editor Utility Widgetの罨

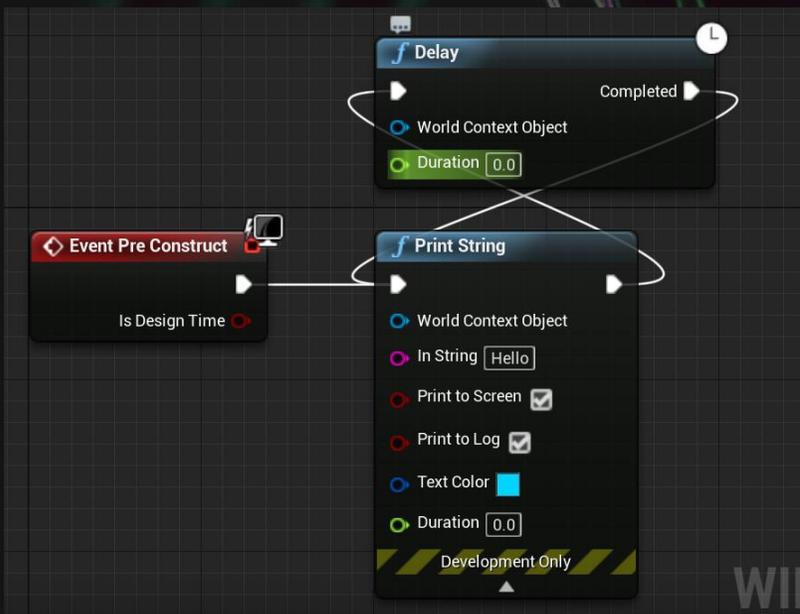
# その1

- 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。

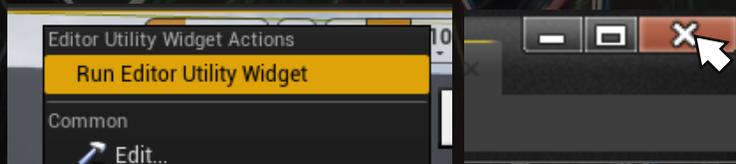


# その2

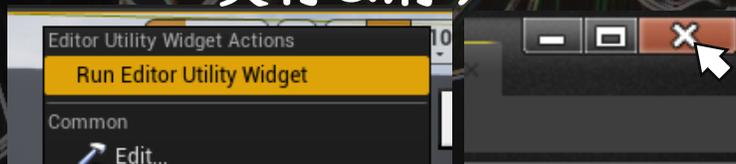
実はEUWを一度実行してEUWを消しても、EUWのインスタンスは残っています。  
→ **EUWのBPをコンパイルするとリセットされます!**  
(もしくは他のGCのタイミングで消える)



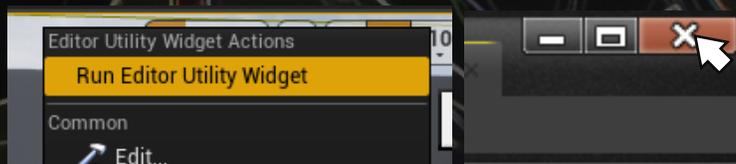
実行&消す



実行&消す



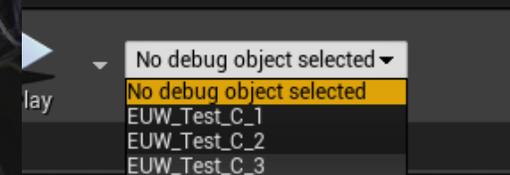
実行&消す



3つ...?

Hello  
Hello  
Hello

DebugFilterみると残っているのがわかる



ent Graph

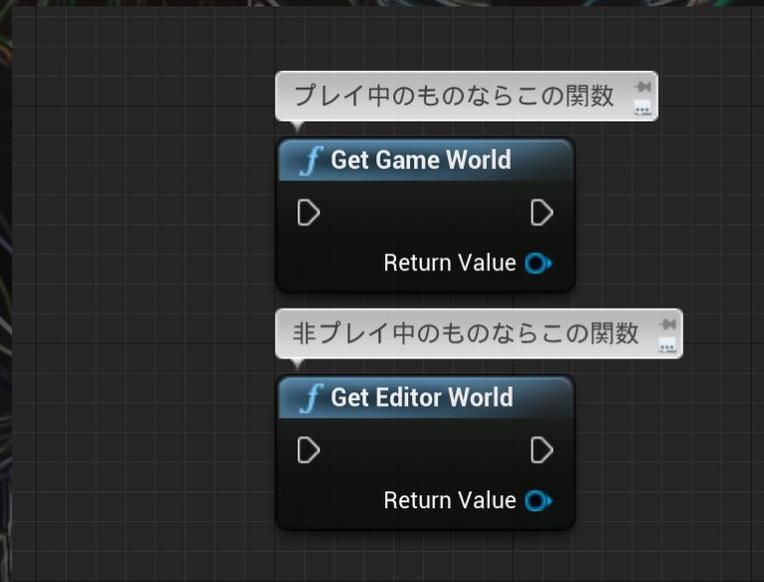
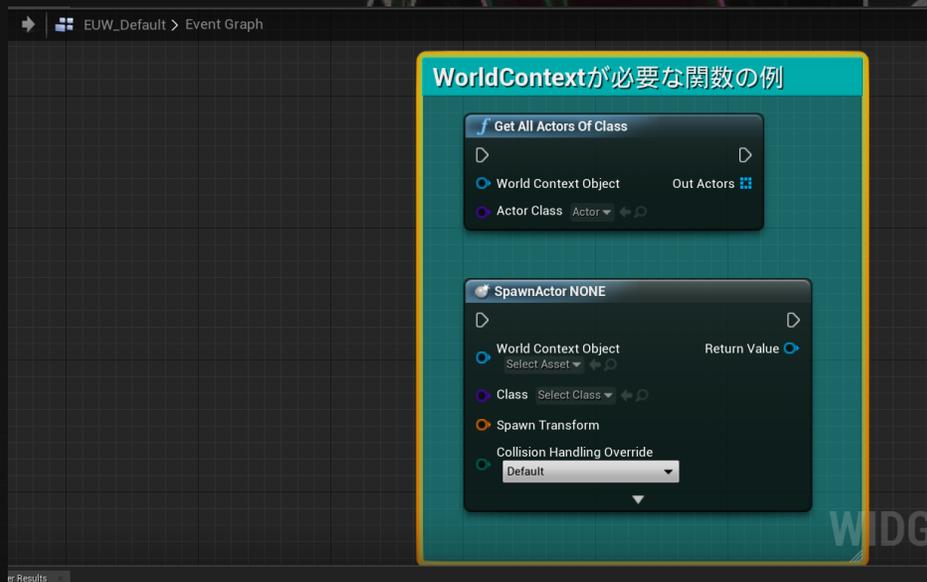
# その3

- 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

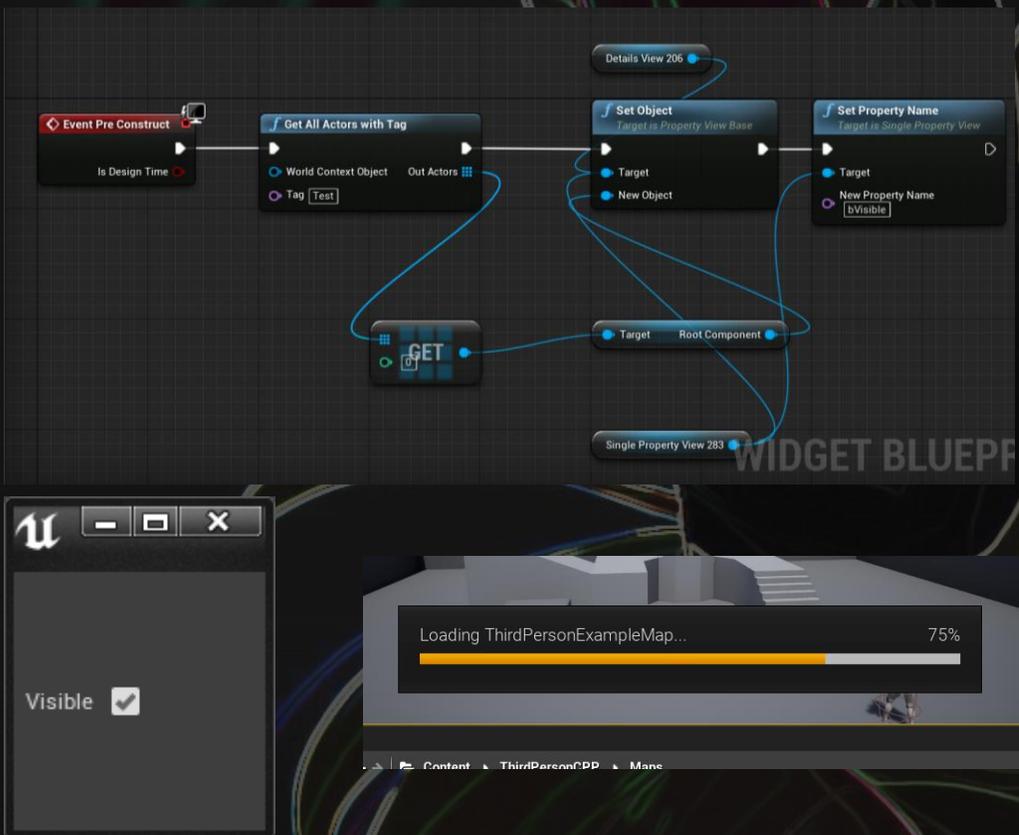
これらの関数の戻り値をつなげることで、機能するようになる



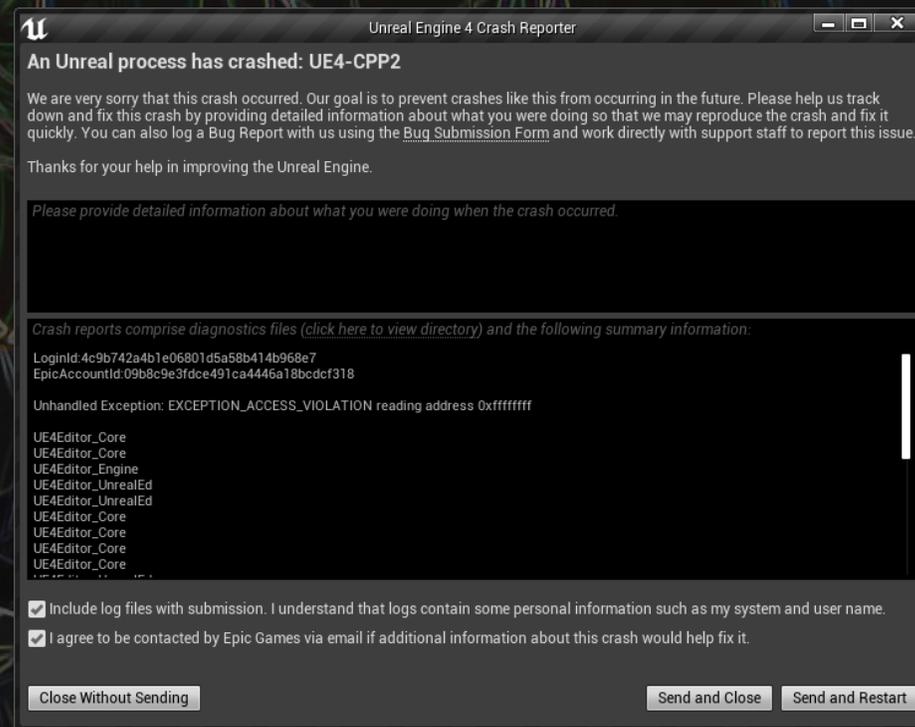
# その4

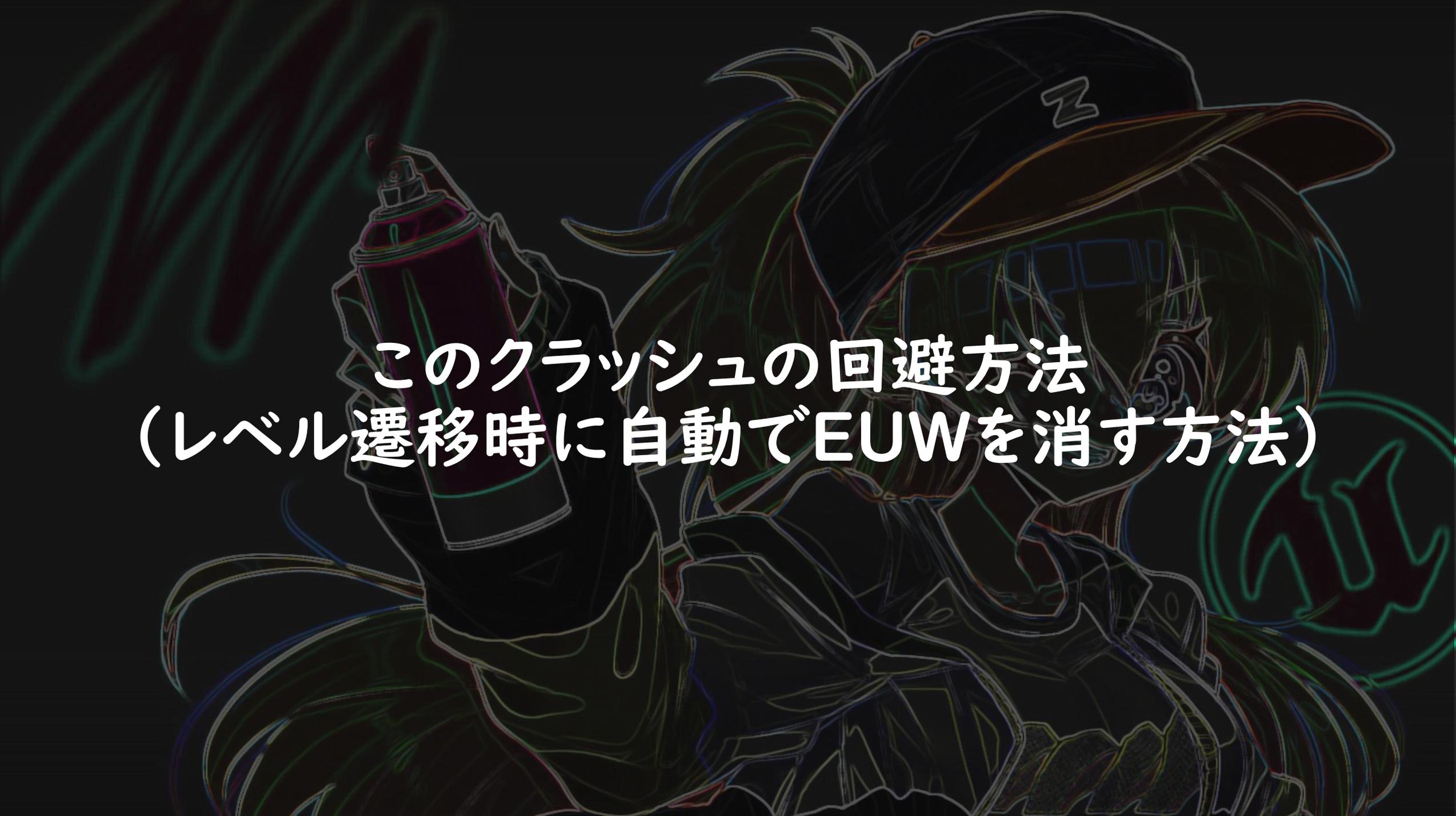
- Actorの参照をもったまま、別のレベルを開いたりしようとする  
クラッシュします。気を付けましょう。  
※変数に保持するだけなら問題なさそうでした。  
→Details View系のWidgetの参照が残っているとクラッシュする。

くらっしゅ!!

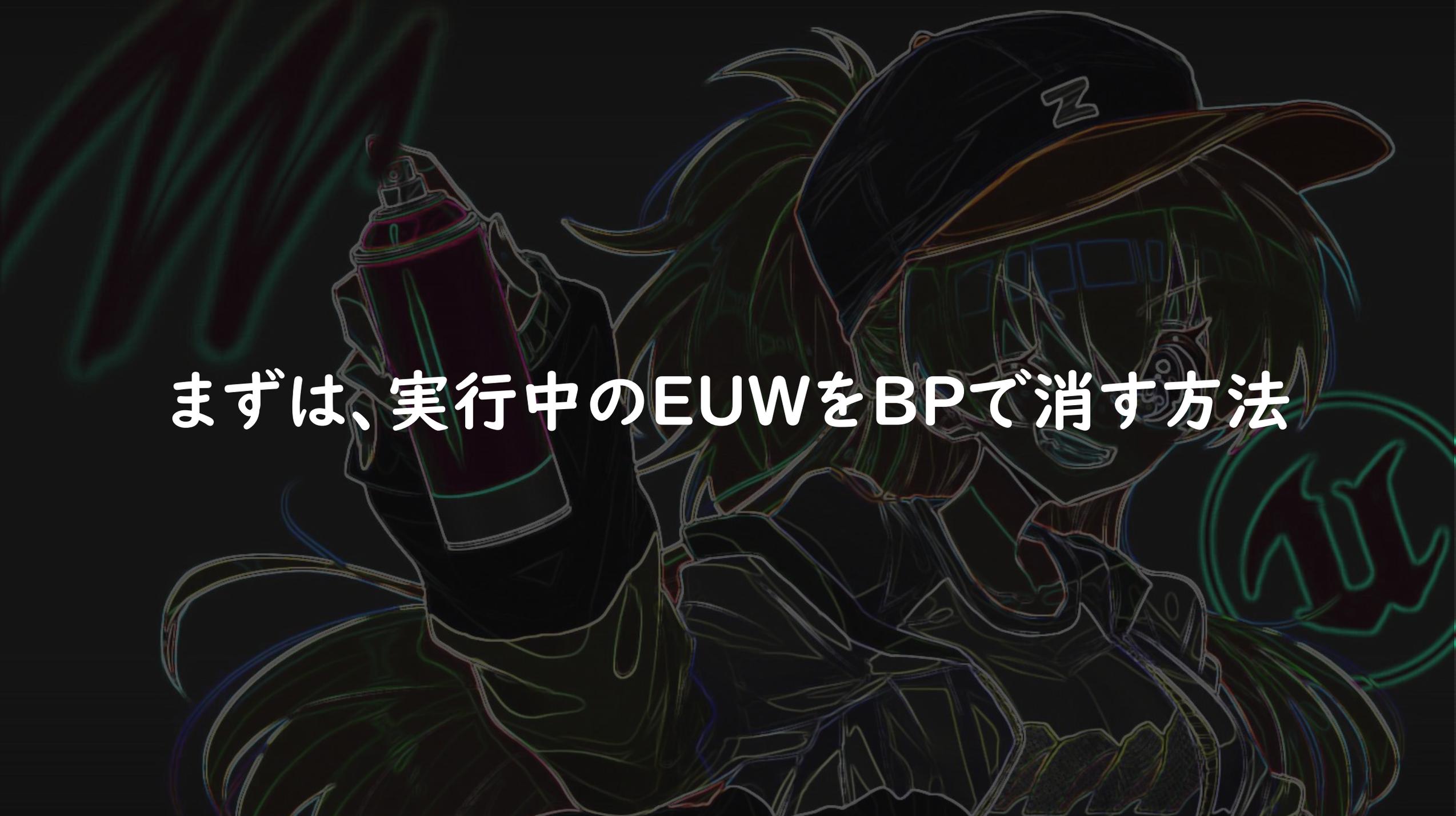


EUWを開いたまま  
別マップを開こうとする





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EUIW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。

```
ObjectInstances.Remove(Asset);

UEditorUtilityWidget* UEditorUtilitySubsystem::SpawnAndRegisterTab(UEditorUtilityWidgetBlueprint* InBlueprint)
{
    if (InBlueprint && !IsRunningCommandlet())
    {
        FName RegistrationName = FName(*(InBlueprint->GetPathName() + LOCTEXT("ActiveTabSuffix", "_ActiveTab").ToString()));
        FText DisplayName = FText::FromString(InBlueprint->GetName());
        FLevelEditorModule& LevelEditorModule = FModuleManager::GetModuleChecked<FLevelEditorModule>(TEXT("LevelEditor"));
        TSharedPtr<FTabManager> LevelEditorTabManager = LevelEditorModule.GetLevelEditorTabManager();
        if (!LevelEditorTabManager->HasTabSpawner(RegistrationName))
        {
            IBlutilityModule* BlutilityModule = FModuleManager::GetModulePtr<IBlutilityModule>("Blutility");
            LevelEditorTabManager->RegisterTab(RegistrationName, FText::FromString(DisplayName), FText::FromString(DisplayName), FText::FromString(DisplayName), FText::FromString(DisplayName), FText::FromString(DisplayName), FText::FromString(DisplayName), FText::FromString(DisplayName), FText::FromString(DisplayName), FText::FromString(DisplayName));
            LevelEditorTabManager->SetDisplayName(DisplayName);
            LevelEditorTabManager->SetGroup(BlutilityModule->GetGroup());
            InBlueprint->SetRegistrationName(RegistrationName);
            BlutilityModule->AddLoadedScriptUI(InBlueprint);
        }

        TSharedRef<SDockTab> NewDockTab = LevelEditorTabManager->InvokeTab(RegistrationName);
        return InBlueprint->GetCreatedWidget();
    }
}

return nullptr;
```

この辺りをコピー!

# 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& LevelEditorModule = GLevelEditorModule;
        TSharedPtr<FTabManager> LevelEditorTabManager = LevelEditorModule.GetTabManager();
        if (!LevelEditorTabManager->HasTab(RegistrationName))
        {
            return;
        }
        TSharedRef<SDockTab> NewDockTab = LevelEditorTabManager->InvokeTab(RegistrationName);
        NewDockTab->RequestCloseTab();
    }
}

return;
```

LevelEditor.hをinclude

いらなところは消す!

RequestCloseTab()を追加!

# 実行中のEUWをBPで消す方法

- Header

```
ueprimfunctionlibrary.cpp | myblueprintfunctionlibrary.h | myobject.cpp | myobject.h
CPP2 | 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",
        });
    }
}
```

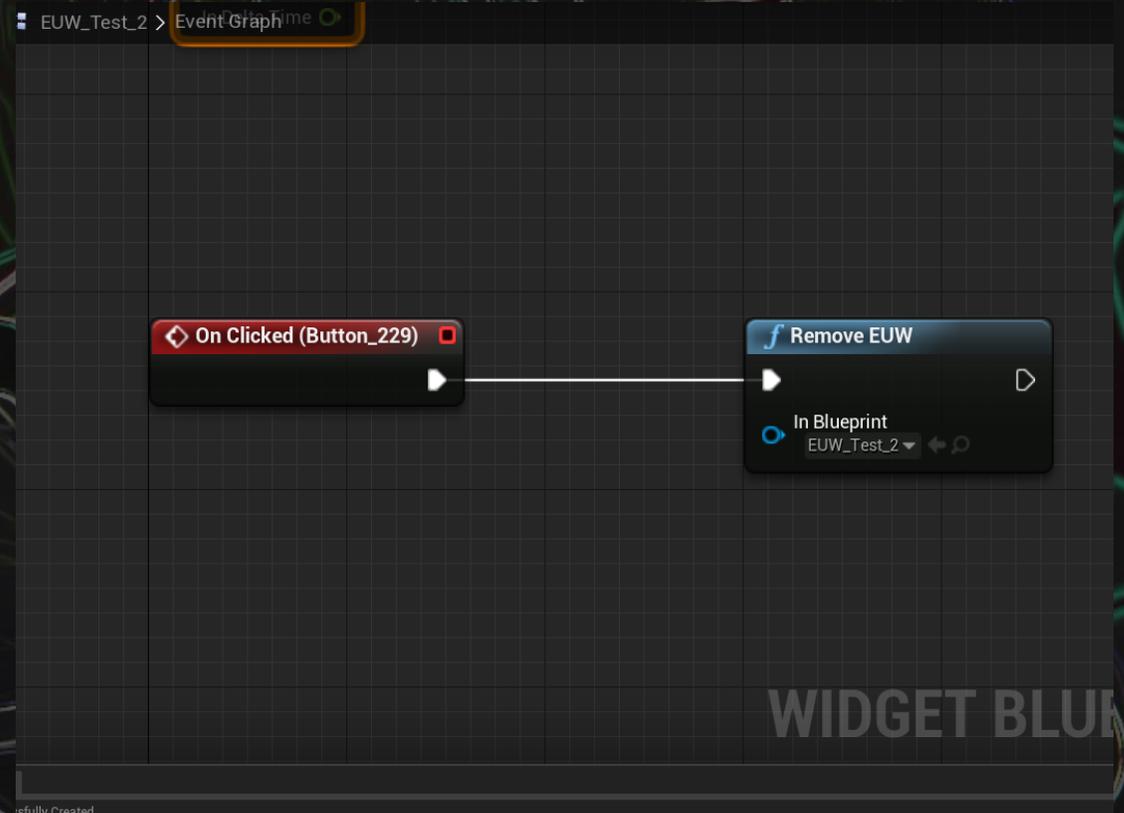
# 実行中のEUWをBPで消す方法

- ブループリント側

EUWにButtonを登録

RemoveSelfEUWWindow

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

```
5 #include "CoreMinimal.h"
6 #include "UObject/NoExportTypes.h"
7 #include "UObject/ScriptMacros.h"
8 #include "Blutility/Classes/EditorUtilityWidgetBlueprint.h"
9 #include "Blutility/Classes/EditorUtilityWidget.h"
10 #include "Toolkits/AssetEditorManager.h"
11 #include "Subsystems/AssetEditorSubsystem.h"
12 #include "MyEUW.generated.h"
13
14 UCLASS(Blueprintable, BlueprintType)
15 class PRESEN_API UMyEUW : public UEditorUtilityWidget
16 {
17     GENERATED_BODY()
18
19     //初期化
20     virtual bool Initialize() override;
21
22 public:
23     //レベルが開いたときなどのイベントをバインドする
24     UFUNCTION(BlueprintCallable)
25     void BindOnLevelChangeRequestedOpen();
26     //呼び出すイベント
27     void OnLevelChangeRequest(UObject* Asset);
28     //EUWを消す
29     UFUNCTION(BlueprintCallable)
30     void RemoveEUW(UEditorUtilityWidgetBlueprint* InBlueprint);
31     //EUWのBPで作成したアセットの参照(Editorで設定)
32     UPROPERTY(EditAnywhere, BlueprintReadWrite)
33     UEditorUtilityWidgetBlueprint* EUWBPRef;
34 }
```

# レベル遷移時に自動でEUWを消す方法

## • C++の中身追加部分

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。

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

```
3
4 #include "MyEUW.h"
5 #include "Modules/ModuleManager.h"
6 #include "LevelEditor.h"
7
8 #define LOCTEXT_NAMESPACE "MyObject"
9 bool UMyEUW::Initialize()
10 {
11     //初期化時にバインド
12     bool Condition = Super::Initialize();
13     BindOnLevelChangeRequestedOpen();
14     return Condition;
15 }
16
17 void UMyEUW::BindOnLevelChangeRequestedOpen()
18 {
19     //レベルが開いたときなどのイベントをバインドする
20     GEditor->GetEditorSubsystem<UAssetEditorSubsystem>()
21     ->OnAssetEditorRequestedOpen().AddUObject(this, &UMyEUW::OnLevelChangeRequest);
22 }
23
24 void UMyEUW::OnLevelChangeRequest(UObject* Asset)
25 {
26     //開いたアセットがマップかどうかを判定
27     UWorld* SelectedMap = Cast<UWorld>(Asset);
28     if (!SelectedMap)
29     {
30         return;
31     }
32     //自身のGeneratedBPClassを保持する変数があるかどうか
33     if (!EUWBPPref)
34     {
35         return;
36     }
37     //EUWを消す
38     RemoveEUW(EUWBPPref);
39 }
```

# 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

適当なアクターをDetailViewsにセット



WIDGET BLUEPRINT

Compiler Results  
• Fast Template Successfully Created.  
• [0214.06] Compile of EUW\_Default successful! [in 104 ms] (/Game/EUW\_Default/EUW\_Default)

作成したEUWクラス

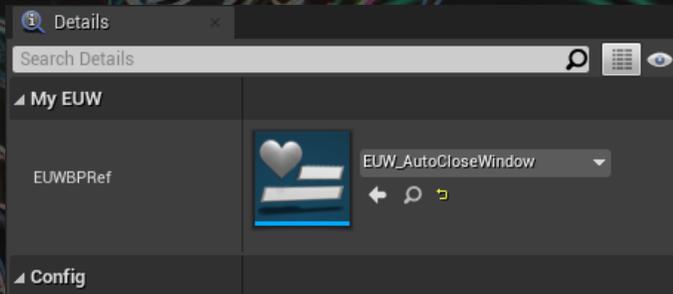
適当なアクターをDetailViewsにセット



WIDGET BLUEPRINT

Compiler Results

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

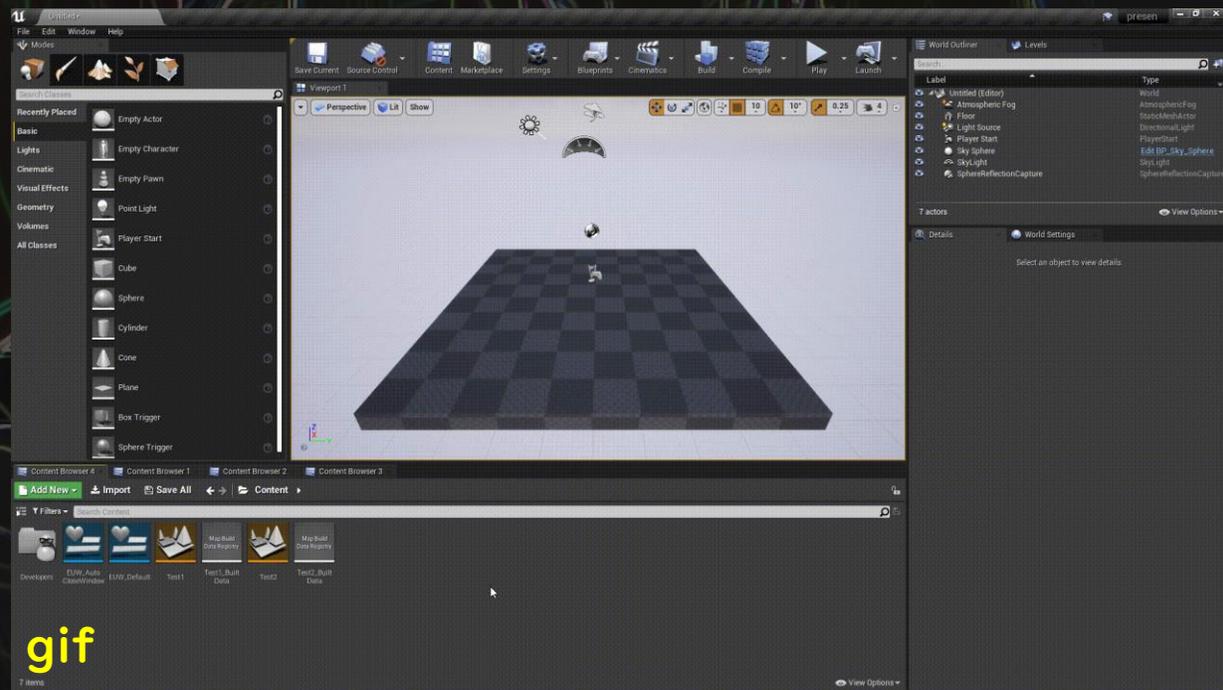
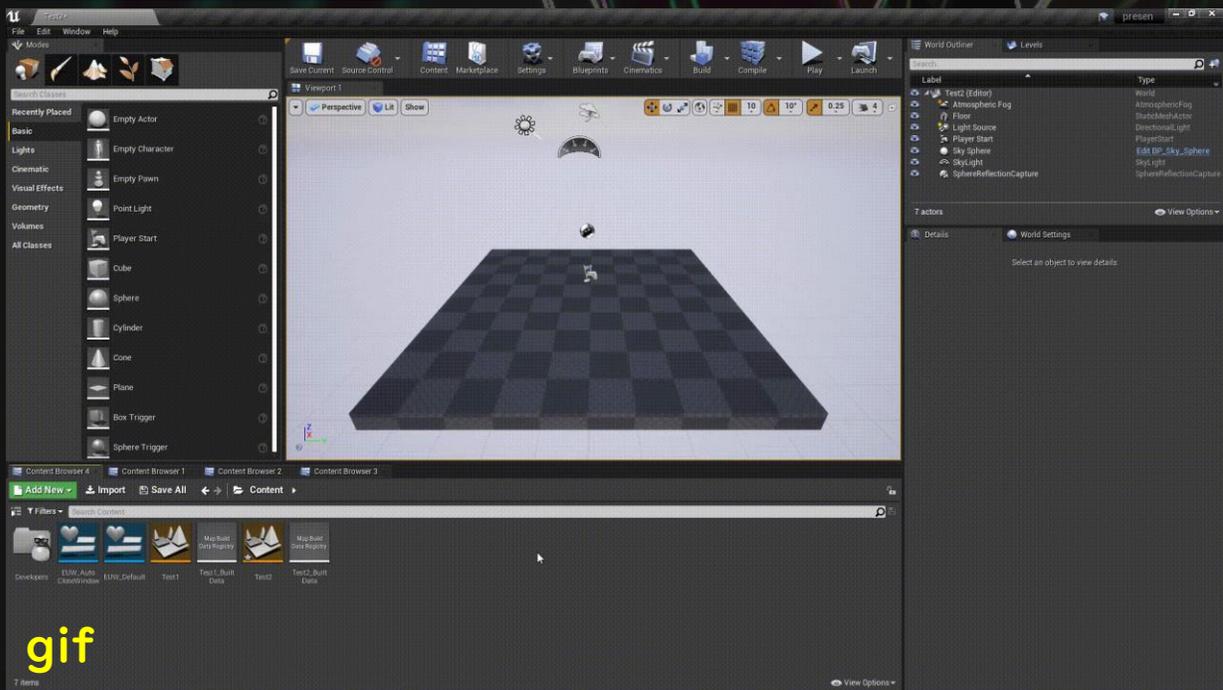
・ 結果

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!!

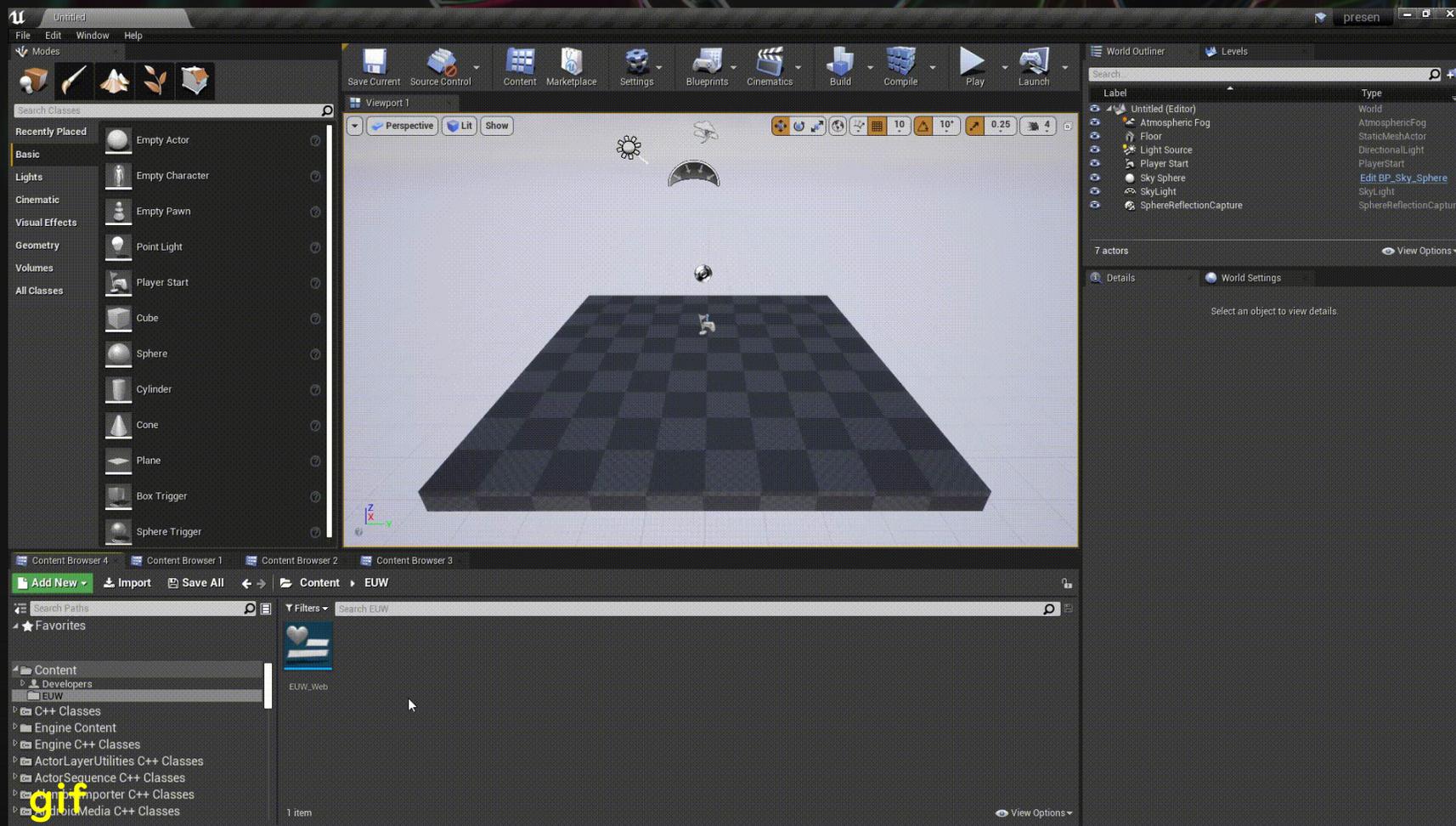




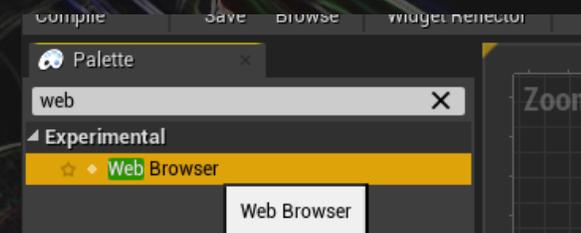
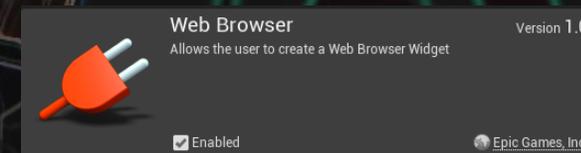
# Editor Utility Widgetの制作事例

# 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。

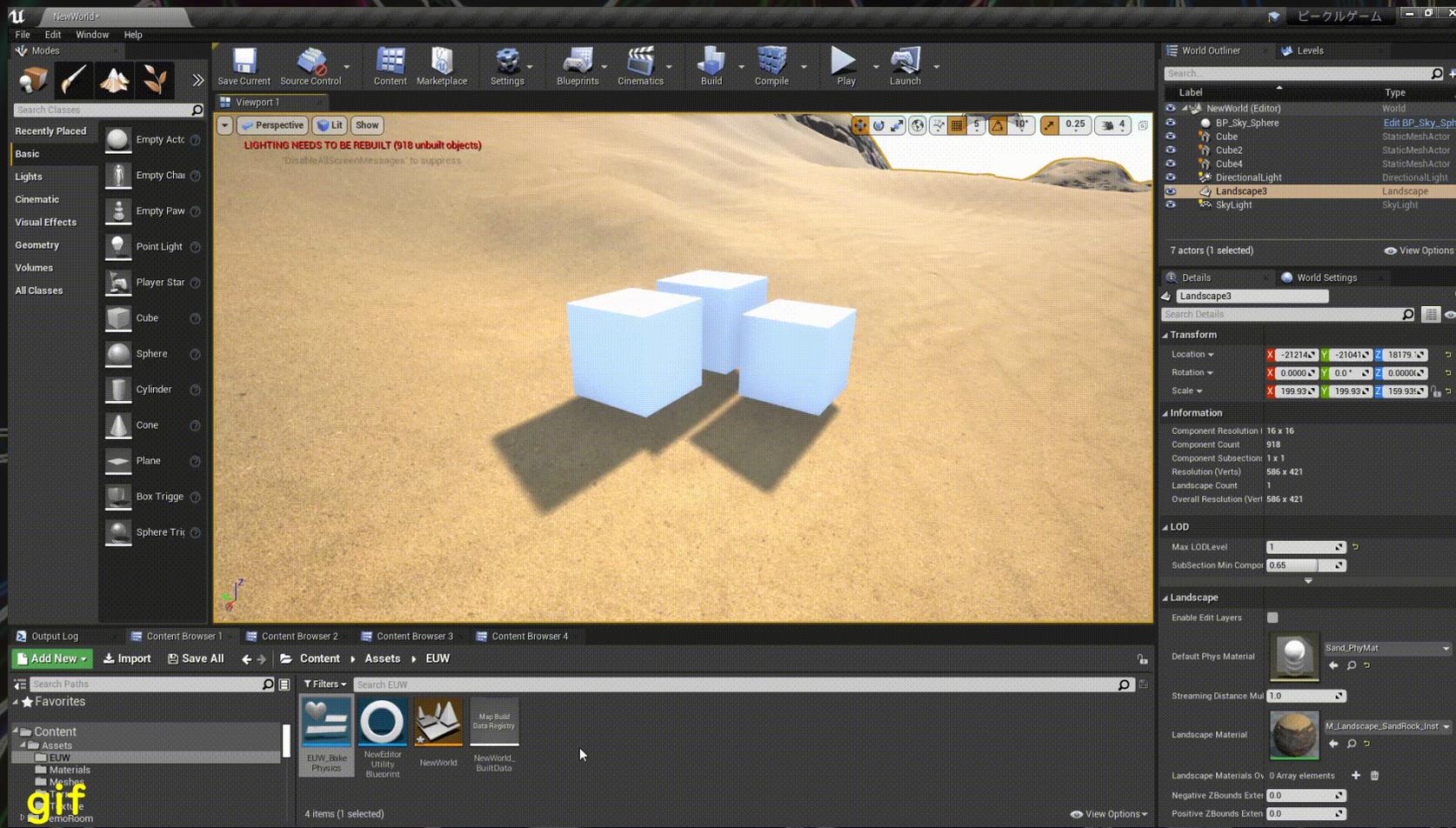


「Web Browser」プラグインを有効にすることで、「Web Browser」ウィジェットを使えるようになる。



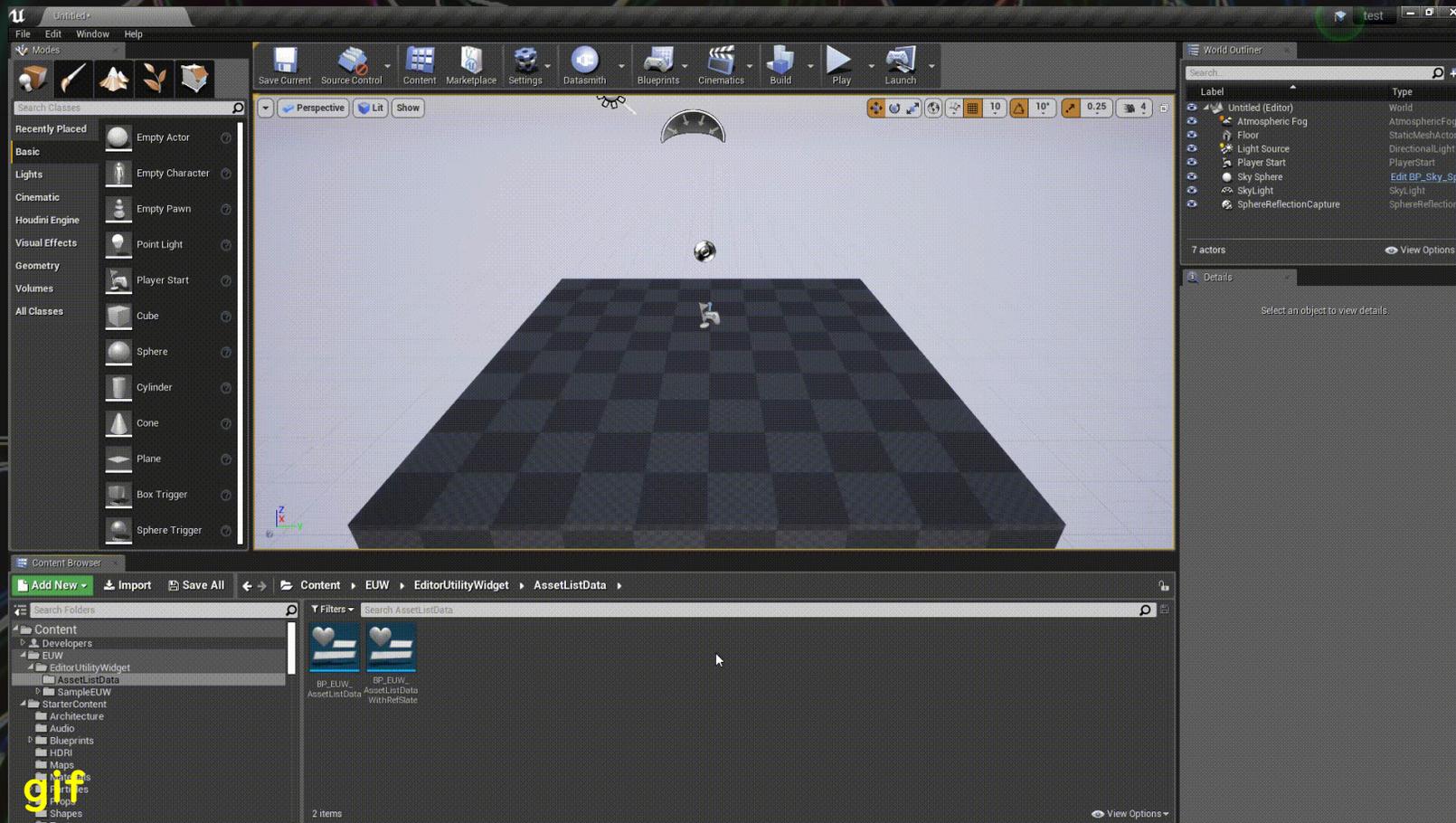
## ②配置物をPhysicsで設置させる

- 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)



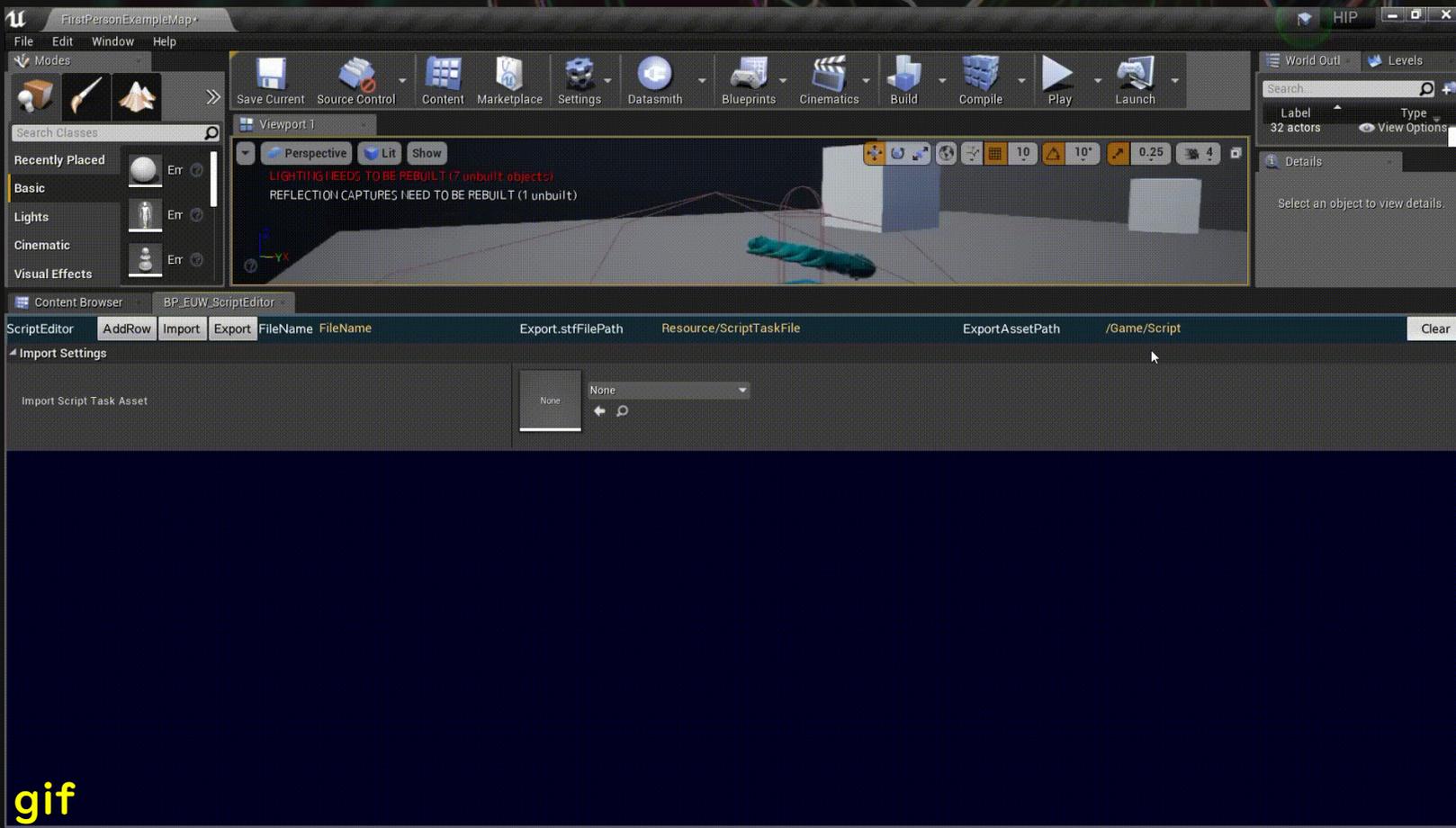
# ③AssetのFavorite機能

- 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)



このようなテキスト形式で管理!

```
1 Label, Test2<<
2 TextTest, テストスクリプト<<
3 TextTest, テスト用のテキストです。<<
4 こんな感じでテキストが出ます。<<
5 Label, Test1<<
6 TextTest, ペペペペ<<
7 TextTest, むむむむ<<
8 TextTest, Complete!<<
9 SetLocalArg, Buf1, 999<<
10 If, Buf1>1, Test1, Test2<<
11 Goto, Test1<<
[EOF]
```

# 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

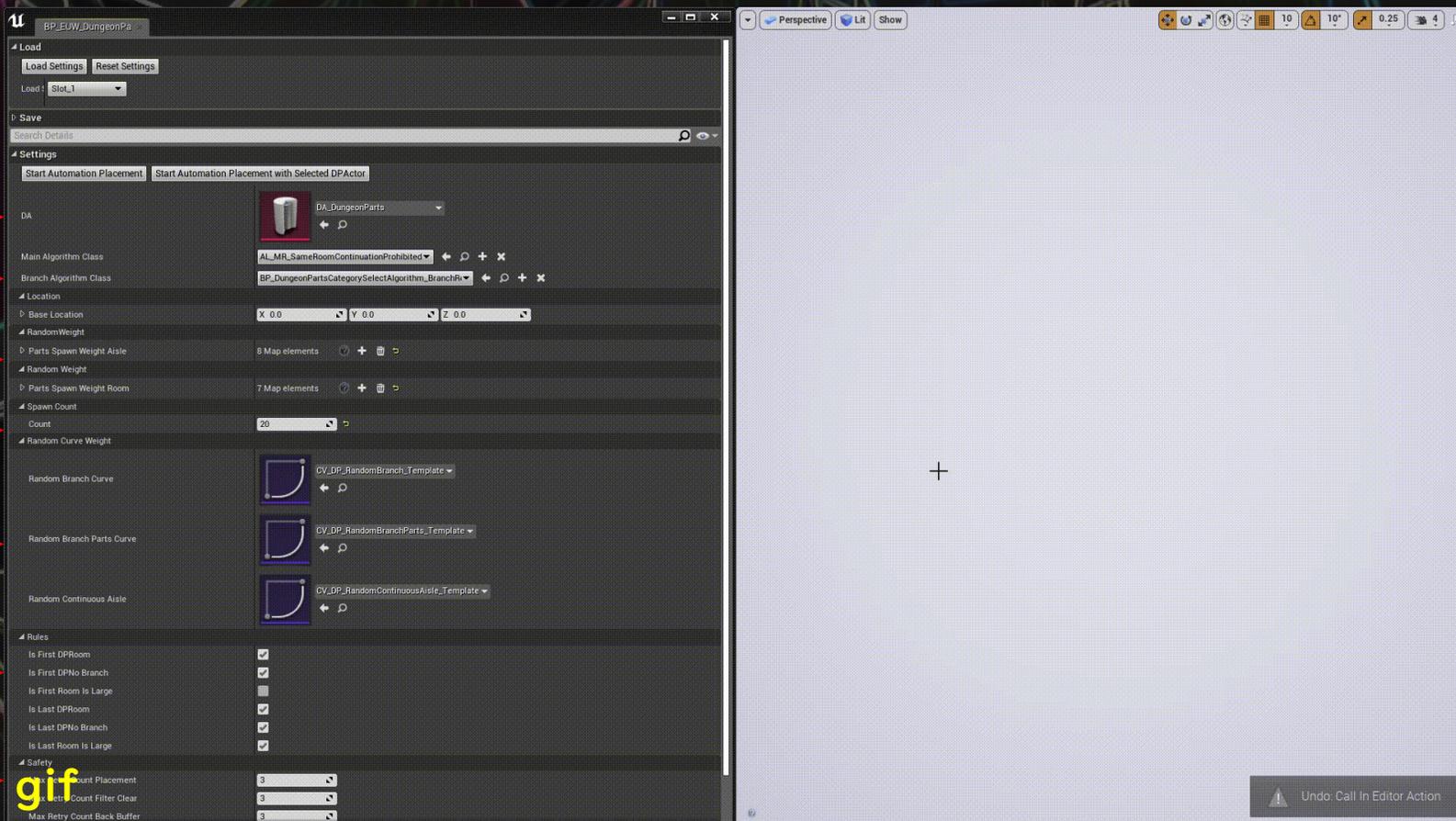
生成する部品クラスの出現ウェイト

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

お疲れ様です!

