

# Ultimate Starter Kit

---

Documentation

*Henry Jooste*

*None*

## Table of contents

---

|                          |    |
|--------------------------|----|
| 1. Getting Started       | 4  |
| 1.1 Requirements         | 4  |
| 1.2 Installation         | 4  |
| 1.3 Plugin Content       | 4  |
| 1.4 Features             | 4  |
| 2. Core Functionality    | 5  |
| 2.1 Input Devices        | 5  |
| 2.2 Game Instance        | 6  |
| 3. Logger                | 9  |
| 3.1 Introduction         | 9  |
| 3.2 Log Levels           | 9  |
| 3.3 Logging Methods      | 9  |
| 3.4 API Reference        | 10 |
| 3.5 Blueprint Usage      | 11 |
| 3.6 C++ Usage            | 11 |
| 4. Trackable Data        | 13 |
| 4.1 Overview             | 13 |
| 4.2 Data                 | 14 |
| 4.3 Component            | 15 |
| 5. Audio                 | 17 |
| 5.1 Audio Overview       | 17 |
| 5.2 Audio Utils          | 18 |
| 5.3 Music Player         | 20 |
| 6. 3D Platformer         | 21 |
| 6.1 Overview             | 21 |
| 6.2 Character            | 22 |
| 6.3 Animation Instance   | 24 |
| 6.4 Shadow Decal         | 25 |
| 7. UI & Widgets          | 26 |
| 7.1 Input Indicator      | 26 |
| 7.2 Input Indicator Icon | 28 |
| 7.3 Menu                 | 29 |
| 7.4 Menu Item            | 31 |
| 8. Utils                 | 35 |
| 8.1 Config Utils         | 35 |

|                   |    |
|-------------------|----|
| 8.2 Platform      | 37 |
| 8.3 Project Utils | 43 |

# 1. Getting Started

---

## 1.1 Requirements

---

The Ultimate Starter Kit plugin is only available for Unreal Engine 5.0.3 and newer. The plugin also depends on the following plugins:

1. Niagara
2. Enhanced Input

## 1.2 Installation

---

1. Download the latest release from [GitHub](#)
2. Navigate to `C:\Program Files\Epic Games\UE_{VERSION}\Engine\Plugins`
3. Create a `Marketplace` folder if needed
4. Extract the release and copy to the `Marketplace` folder
5. Open Unreal Engine
6. Click on `Edit > Plugins`
7. Enable the plugin under the `Built-in > Other` category
8. Restart Unreal Engine

## 1.3 Plugin Content

---

The Ultimate Starter Kit plugin includes content that can be used in your Blueprints. You might need to enable this first:

1. Open the `Content Browser`
2. Click on the settings button
3. Enable the `Show Plugin Content` setting

## 1.4 Features

---

The Ultimate Starter Kit plugin comes with the following features:

- **Logger**: A system used to easily log info to file and via on-screen messages
- **Save data management**: A system used to easily save/load game data with support for multiple save slots
- **Input Indicators**: A system used to easily detect different input devices and update the UI to display the correct indicators
- **Currency**: A system that is used to easily manage different types of currency
- **Audio**: A system used to manage the basic properties of audio files
- **Stats**: A system used to easily manage character stats with an optional regenerate ability
- **3D platformer character**: Basic 3D platformer character and animation template
- **Menu System**: A customizable menu system with support for controllers and complex menu layouts
- **Platform Utils**: Easily detect different platform types and architectures

## 2. Core Functionality

---

### 2.1 Input Devices

---

#### 2.1.1 Introduction

---

Supported input devices. This is used to update the input indicators when using different input devices

#### 2.1.2 Values

---

| Value                 | Description  |
|-----------------------|--|
| KeyboardMouse         | Using a keyboard and mouse                                     |
| GenericController     | Using a controller on a desktop build                          |
| XboxController        | Using an Xbox controller                                       |
| PlaystationController | Using a Playstation controller                                 |
| SwitchController      | Using a Nintendo Switch controller                             |
| Unknown               | Unknown device (used before initializing the input indicators) |

## 2.2 Game Instance

---

### 2.2.1 Introduction

A base game instance with support for saving and loading game data using multiple save slots

### 2.2.2 Dependencies

The `USKGameInstance` relies on other components of this plugin to work:

- **Logger:** Used to log useful information to help you debug any issues you might experience

### 2.2.3 Using the Game Instance

You need to create a blueprint using the `USKGameInstance_Implementation` as a parent before using the game instance. The input indicators feature is already configured if you use this base class. If you prefer to set this up manually, you can use `USKGameInstance` instead. After creating your own game instance blueprint, set this as the default game instance:

1. Open the Project Settings
2. Go to Project > Maps & Modes
3. Change the `Game Instance Class` value to your own blueprint

### 2.2.4 Save Data

You need to create a `USK Save Game` object before you can save/load data. This object contains all the data that you want to save. Just add the data you want to save as variables to the blueprint. The `Game Instance` will handle the rest. You also need to set the following properties before you can save/load data:

- **Save Game Class:** A reference to the `USK Save Game` class that contains the data you want to save

**NB:** You are required to set the save slot before you can save/load data. If not, you will get a `nullptr` and might cause your game to crash

### 2.2.5 Input Indicators

The Game Instance will automatically detect input events and update the current input device if needed. If the input device is changed, other classes will be notified through the `OnInputDeviceUpdated` event

## 2.2.6 API Reference

### Properties

| Property                           | Description   | Type                      | Default Value        |
|------------------------------------|---|---------------------------|----------------------|
| SaveGameClass                      | The class that holds the data that should be saved/loaded                           | TSubclassOf<UUSKSaveGame> |                      |
| InputMappingContext                | The input mapping context used to extract the keys based on specific input actions  | UInputMappingContext*     | <code>nullptr</code> |
| KeyboardMouseInputMappings         | A map of all keyboard/mouse keys and the texture displayed in the indicator         | TMap<FKey, UTexture2D*>   |                      |
| GenericControllerInputMappings     | A map of all generic controller keys and the texture displayed in the indicator     | TMap<FKey, UTexture2D*>   |                      |
| XboxControllerInputMappings        | A map of all Xbox controller keys and the texture displayed in the indicator        | TMap<FKey, UTexture2D*>   |                      |
| PlaystationControllerInputMappings | A map of all Playstation controller keys and the texture displayed in the indicator | TMap<FKey, UTexture2D*>   |                      |
| SwitchControllerInputMappings      | A map of all Switch controller keys and the texture displayed in the indicator      | TMap<FKey, UTexture2D*>   |                      |

### Events

| Name                 | Description   | Params |
|----------------------|---|--------|
| OnDataLoadedEvent    | Event used to notify other classes when the save data is loaded             |        |
| OnInputDeviceUpdated | Event used to notify other classes when the current input device is updated |        |

### Functions

| Name                  | Description  | Params   | Return   |
|-----------------------|--|--|--|
| GetSaveData           | Get the save data that is currently loaded         |  | <b>UUSKSaveGame*</b><br>A reference to the current save data                                   |
| SaveData              | Save the modified data currently in memory         |  |  |
| SetCurrentSaveSlot    | Set the current save slot                          | <b>Index (int)</b><br>The index of the save slot   |  |
| IsSaveSlotUsed        | Check if a save slot is used                       | <b>Index (int)</b><br>The index of the save slot to check  | <b>bool</b><br>A boolean value indicating if the save slot is used                             |
| GetInputIndicatorIcon | Get the input indicator icon for a specific action | <b>InputAction (UInputAction*)</b><br>The input action<br><br><b>Amount (int)</b><br>The amount of icons to retrieve | <b>TArray&lt;UTexture2D*&gt;</b><br>An array of input indicator icons for the specified action |

## 2.2.7 Blueprint Usage

---

You can use the `USKGameInstance` using Blueprints by adding one of the following nodes:

- Ultimate Starter Kit > Save Data > Get Save Data
- Ultimate Starter Kit > Save Data > Save Data
- Ultimate Starter Kit > Save Data > Set Current Save Slot
- Ultimate Starter Kit > Save Data > Is Save Slot Used
- Ultimate Starter Kit > Input > Get Input Indicator Icon

## 2.2.8 C++ Usage

---

Before you can use the plugin, you first need to enable the plugin in your `Build.cs` file:

```
PublicDependencyModuleNames.Add("USK");
```

The `USKGameInstance` can now be used in any of your C++ files:

```
#include "USK/Core/USKGameInstance.h"

void ATestActor::Test()
{
    // USKGameInstance is a pointer to the UUSKGameInstance
    UUSKSaveGame* SaveData = USKGameInstance->GetSaveData();
    USKGameInstance->SaveData();
    USKGameInstance->SetCurrentSaveSlot(Index);
    bool IsSaveSlotUsedValue = USKGameInstance->IsSaveSlotUsed(Index);
    TArray<UTexture2D*> InputIndicatorIcon = USKGameInstance->GetInputIndicatorIcon(InputAction, Amount);
}
```



## 3. Logger

---

### 3.1 Introduction

---

A system used to easily log info to file and via on-screen messages

### 3.2 Log Levels

---

This plugin supports the following log types:

1. **Trace:** Logs that contain the most detailed messages. These messages may contain sensitive application data. These messages are disabled by default and should never be enabled in a production environment
2. **Debug:** Logs that are used for interactive investigation during development. These logs should primarily contain information useful for debugging and have no long-term value
3. **Information:** Logs that track the general flow of the application. These logs should have long-term value
4. **Warning:** Logs that highlight an abnormal or unexpected event in the application flow, but do not otherwise cause the application execution to stop
5. **Error:** Logs that highlight when the current flow of execution is stopped due to a failure. These should indicate a failure in the current activity, not an application-wide failure

The log levels corresponds to the following verbosity level in Unreal Engine:

| Log Level   | Log Verbosity |
|-------------|---------------|
| Trace       | VeryVerbose   |
| Debug       | Verbose       |
| Information | Display       |
| Warning     | Warning       |
| Error       | Error         |

The plugin will automatically ignore certain log levels based on the type of build:

| Log Level   | Development | Shipping |
|-------------|-------------|----------|
| Trace       | Enabled     | Disabled |
| Debug       | Enabled     | Disabled |
| Information | Enabled     | Enabled  |
| Warning     | Enabled     | Enabled  |
| Error       | Enabled     | Enabled  |

### 3.3 Logging Methods

---

There are 2 different logging methods. Both of these are used each time you log something:

- **On-screen messages:** These messages will appear on-screen for 5 seconds (only used when running the game through the editor)
- **Log File:** Everything you log is also written to a file using the Unreal Engine logging feature

## 3.4 API Reference

---

### 3.4.1 Macros

| Name            | Description  | Params  | Return |
|-----------------|--|---|--------|
| USK_LOG_TRACE   | Log trace information using the current function name as the tag | <b>Text (FString)</b><br>The text that should be logged out |        |
| USK_LOG_DEBUG   | Log debug information using the current function name as the tag | <b>Text (FString)</b><br>The text that should be logged out |        |
| USK_LOG_INFO    | Log information using the current function name as the tag       | <b>Text (FString)</b><br>The text that should be logged out |        |
| USK_LOG_WARNING | Log a warning using the current function name as the tag         | <b>Text (FString)</b><br>The text that should be logged out |        |
| USK_LOG_ERROR   | Log an error using the current function name as the tag          | <b>Text (FString)</b><br>The text that should be logged out |        |

### 3.4.2 Functions

| Name    | Description           | Params   | Return |
|---------|-----------------------|--|--------|
| Trace   | Log trace information | <b>Tag (FString)</b><br>The category of the log entry. This is usually the function or class name. It allows you to find out exactly where this is being logged<br><br><b>Text (FString)</b><br>The text that should be logged out |        |
| Debug   | Log debug information | <b>Tag (FString)</b><br>The category of the log entry. This is usually the function or class name. It allows you to find out exactly where this is being logged<br><br><b>Text (FString)</b><br>The text that should be logged out |        |
| Info    | Log information       | <b>Tag (FString)</b><br>The category of the log entry. This is usually the function or class name. It allows you to find out exactly where this is being logged<br><br><b>Text (FString)</b><br>The text that should be logged out |        |
| Warning | Log a warning         | <b>Tag (FString)</b><br>The category of the log entry. This is usually the function or class name. It allows you to find out exactly where this is being logged<br><br><b>Text (FString)</b><br>The text that should be logged out |        |
| Error   | Log an error          | <b>Tag (FString)</b><br>The category of the log entry. This is usually the function or class name. It allows you to find out exactly where this is being logged<br><br><b>Text (FString)</b><br>The text that should be logged out |        |

## 3.5 Blueprint Usage

You can easily log info using Blueprints by adding one of the following nodes:

- Ultimate Starter Kit > Logger > Log Trace
- Ultimate Starter Kit > Logger > Log Debug
- Ultimate Starter Kit > Logger > Log Info
- Ultimate Starter Kit > Logger > Log Warning
- Ultimate Starter Kit > Logger > Log Error

## 3.6 C++ Usage

The logging is handled through a static class/functions. You first need to enable the plugin in your `Build.cs` file:

```
PublicDependencyModuleNames.Add("USK");
```

The logger can now be used in any of your C++ files:

```
#include "USK/Logger/Log.h"

void ATestActor::Test()
{
    USK_LOG_TRACE("Testing trace logging");
    USK_LOG_DEBUG("Testing debug logging");
    USK_LOG_INFO("Testing info logging");
    USK_LOG_WARNING("Testing warning logging");
    USK_LOG_ERROR("Testing error logging");

    ULog::Trace("Custom Tag", "Testing trace logging");
    ULog::Debug("Custom Tag", "Testing debug logging");
    ULog::Info("Custom Tag", "Testing info logging");
    ULog::Warning("Custom Tag", "Testing warning logging");
    ULog::Error("Custom Tag", "Testing error logging");
}
```

## 4. Trackable Data

---

### 4.1 Overview

---

#### 4.1.1 Introduction

---

A system that is used to easily manage different types of actor data

#### 4.1.2 Trackable Data Component

---

Before you can manage the data, you need to create a [Trackable Data Component](#) and add it to the actor/character containing the data

#### 4.1.3 Built-in data

---

The following data can automatically be managed without creating custom components:

1. Currency (using the `Currency Component` )
2. Stats (using the `Stats Component` )

## 4.2 Data

---

### 4.2.1 Introduction

---

All trackable data use the `FTrackableData` struct to specify the default values and behaviours

### 4.2.2 Properties

---

| Property        | Description  | Type  | Default Value |
|-----------------|--|-------|---------------|
| InitialValue    | The initial value of the data  | float |               |
| EnforceMaxValue | Should we enforce a maximum value?                                       | bool  |               |
| MaxValue        | The maximum value of the data  | float | 100.0f        |
| AutoSave        | Should all value updates automatically be saved using the game instance? | bool  |               |
| AutoGenerate    | Should we automatically generate value every second?                     | bool  |               |
| GenerateAmount  | The amount of value to generate every second                             | float |               |
| GenerateDelay   | The delay before the value starts generating after losing value          | float |               |

## 4.3 Component

### 4.3.1 Introduction

A component that is used to easily manage/track different types of actor data

### 4.3.2 Dependencies

The `TrackableDataComponent` relies on other components of this plugin to work:

- [Logger](#): Used to log useful information to help you debug any issues you might experience
- [Game Instance](#): Used to monitor for input device changes and handle saving/loading game data

### 4.3.3 Data

The data to track is configured by adding items to the `Data` map. The component should be added to the actor/character containing the data

### 4.3.4 API Reference

#### Properties

| Property | Description              | Type                        | Default Value |
|----------|--------------------------|-----------------------------|---------------|
| Data     | The map of data to track | TMap<FName, FTrackableData> |               |

#### Events

| Name           | Description   | Params   |
|----------------|---|--|
| OnValueZero    | Event used to notify other classes every time the data value reaches 0  | <b>Name (FName)</b><br>The name of the data item   |
| OnValueUpdated | Event used to notify other classes every time the data value is updated | <b>Name (FName)</b><br>The name of the data item<br><br><b>Value (FName)</b><br>The current value of the data item<br><br><b>ValuePercentage (FName)</b><br>The percentage of the current value compared to the max value of the data item |

## Functions

| Name               | Description   | Params  | Return  |
|--------------------|---|---|---|
| GetValue           | Get the amount of the data                                    | <b>Name (FName)</b><br>The name of the data item  | <b>float</b><br>The current amount of the data item                       |
| GetValuePercentage | Get the value of the data as a percentage of to the max value | <b>Name (FName)</b><br>The name of the data item  | <b>float</b><br>The value of the data as a percentage of to the max value |
| Add                | Add an amount to the data                                     | <b>Name (FName)</b><br>The name of the data item<br><br><b>Amount (float)</b><br>The amount to add    | <b>float</b><br>The new amount of the data item                           |
| Remove             | Remove an amount from the data                                | <b>Name (FName)</b><br>The name of the data item<br><br><b>Amount (float)</b><br>The amount to remove | <b>float</b><br>The new amount of the data item                           |

### 4.3.5 Blueprint Usage

You can use the `TrackableDataComponent` using Blueprints by adding one of the following nodes:

- Ultimate Starter Kit > Trackable Data > Get Value
- Ultimate Starter Kit > Trackable Data > Get Value Percentage
- Ultimate Starter Kit > Trackable Data > Add
- Ultimate Starter Kit > Trackable Data > Remove

### 4.3.6 C++ Usage

Before you can use the plugin, you first need to enable the plugin in your `Build.cs` file:

```
PublicDependencyModuleNames.Add("USK");
```

The `TrackableDataComponent` can now be used in any of your C++ files:

```
#include "USK/Data/TrackableDataComponent.h"

void ATestActor::Test()
{
    // TrackableDataComponent is a pointer to the UTrackableDataComponent
    float Value = TrackableDataComponent->GetValue(Name);
    float ValuePercentage = TrackableDataComponent->GetValuePercentage(Name);
    float AddValue = TrackableDataComponent->Add(Name, Amount);
    float RemoveValue = TrackableDataComponent->Remove(Name, Amount);
}
```



## 5. Audio

---

### 5.1 Audio Overview

---

#### 5.1.1 Introduction

---

A system used to manage the basic properties of audio files. It includes different sound classes, a sound mix and sound attenuation settings

#### 5.1.2 Sound Classes

---

The audio system includes a few basic preconfigured sound classes:

| Class name            | Group   | Volume |
|-----------------------|---------|--------|
| USK_EffectsSoundClass | Effects | 1.0    |
| USK_MusicSoundClass   | Music   | 0.5    |
| USK_UISoundClass      | UI      | 1.0    |
| USK_VoiceSoundClass   | Voice   | 3.0    |

## 5.2 Audio Utils

### 5.2.1 Introduction

The audio utils class is used to easily play sound effects

### 5.2.2 Dependencies

The `AudioUtils` relies on other components of this plugin to work:

- [Logger](#): Used to log useful information to help you debug any issues you might experience

### 5.2.3 API Reference

#### Functions

| Name              | Description   | Params  | Return |
|-------------------|---|---|--------|
| PlaySound2D       | Play a 2D sound                                       | <b>WorldContext (UObject*)</b><br>The top level object representing a map<br><br><b>SoundFX (USoundBase*)</b><br>The USoundBase to play                             |        |
| PlayRandomSound2D | Play a random 2D sound                                | <b>WorldContext (UObject*)</b><br>The top level object representing a map<br><br><b>SoundFX (TArray)</b><br>The array of USoundBase to select the random sound from |        |
| PlaySound         | Play a sound at the specified actor's location        | <b>Actor (AActor*)</b><br>The actor where the sound will be played<br><br><b>SoundFX (USoundBase*)</b><br>The USoundBase to play                                    |        |
| PlayRandomSound   | Play a random sound at the specified actor's location | <b>Actor (AActor*)</b><br>The actor where the sound will be played<br><br><b>SoundFX (TArray)</b><br>The array of USoundBase to select the random sound from        |        |

### 5.2.4 Blueprint Usage

You can use the `AudioUtils` using Blueprints by adding one of the following nodes:

- Ultimate Starter Kit > Audio > Play Sound2D
- Ultimate Starter Kit > Audio > Play Random Sound2D
- Ultimate Starter Kit > Audio > Play Sound
- Ultimate Starter Kit > Audio > Play Random Sound

## 5.2.5 C++ Usage

---

Before you can use the plugin, you first need to enable the plugin in your `Build.cs` file:

```
PublicDependencyModuleNames.Add("USK");
```

The `AudioUtils` can now be used in any of your C++ files:

```
#include "USK/Audio/AudioUtils.h"

void ATestActor::Test()
{
    UAudioUtils::PlaySound2D(WorldContext, SoundFX);
    UAudioUtils::PlayRandomSound2D(WorldContext, SoundFX);
    UAudioUtils::PlaySound(Actor, SoundFX);
    UAudioUtils::PlayRandomSound(Actor, SoundFX);
}
```

## 5.3 Music Player

### 5.3.1 Introduction

Actor responsible for playing, pausing and stopping music. It also allows you to adjust music volume

### 5.3.2 Dependencies

The `inal` relies on other components of this plugin to work:

- [Logger](#): Used to log useful information to help you debug any issues you might experience

### 5.3.3 Components

The `inal` uses the following components:

| Name        | Description  | Type             |
|-------------|--|------------------|
| AudioPlayer | Actor responsible for playing, pausing and stopping music. It also allows you to adjust music volume | UAudioComponent* |

#### ## API Reference ### Properties

| Property    | Description  | Type | Default Value |
|-------------|--|------|---------------|
| PlayOnStart | Should the music automatically play when the actor is spawned? | bool | true          |

#### ### Functions

| Name      | Description                             | Params   | Return |
|-----------|---|--|--------|
| SetVolume | Adjust the playback volume of the music | <b>Volume (float)</b><br>The new volume of the music |        |
| Play      | Play the music                          |  |        |
| Pause     | Pause the music                         |  |        |
| Stop      | Stop the music                          |  |        |

## Blueprint Usage You can use the `inal` using Blueprints by adding one of the following nodes:

- Ultimate Starter Kit > Audio > Set Volume
- Ultimate Starter Kit > Audio > Play
- Ultimate Starter Kit > Audio > Pause
- Ultimate Starter Kit > Audio > Stop

## C++ Usage Before you can use the plugin, you first need to enable the plugin in your `Build.cs` file:

```
PublicDependencyModuleNames.Add("USK");
```

The `inal` can now be used in any of your C++ files:

```
#include "USK/Audio/MusicPlayer.h"

void ATestActor::Test()
{
    // inal is a pointer to the final
    inal->SetVolume(Volume);
    inal->Play();
    inal->Pause();
    inal->Stop();
}
```

## 6. 3D Platformer

---

### 6.1 Overview

---

#### 6.1.1 Introduction

---

The plugin includes a basic 3D platformer character and animation template. This can easily be extended to add unique functionality

## 6.2 Character

---

### 6.2.1 Introduction

---

Base character that can be used for 3D platformer games

### 6.2.2 Dependencies

---

The `PlatformerCharacter` relies on other components of this plugin to work:

- [Logger](#): Used to log useful information to help you debug any issues you might experience

### 6.2.3 Features

---

The following features are included in the 3D platformer character class:

1. **Double Jumping**: Additional jump with a different jump animation
2. **Variable Jump Height**: Adjust jump height based on how long the jump button is pressed
3. **Coyote Time**: Allow the character to jump for a short time after falling off the platform
4. **Shadow Decal**: A decal used as a shadow to indicate where the character will land
5. **Adjustable Camera Distance**: The camera automatically zooms in on the character while idle and zooms out as soon as the character starts moving
6. **Jump & Land effects**: Sound effects and particles when jumping and landing

All these features can be configured to meet your needs and can also be disabled

### 6.2.4 Components

---

The `PlatformerCharacter` uses the following components:

| Name               | Description   | Type                 |
|--------------------|---|----------------------|
| SpringArmComponent | Base character that can be used for 3D platformer games | USpringArmComponent* |
| CameraComponent    | The camera used by the character                        | UCameraComponent*    |

## ## API Reference ### Properties

| Property                      | Description  | Type                      | Default Value |
|-------------------------------|--|---------------------------|---------------|
| InputMappingContext           | The input mapping context used by the player                                       | UInputMappingContext*     | nullptr       |
| MoveAction                    | The move input action  | UInputAction*             | nullptr       |
| LookAroundAction              | The camera rotation input action   | UInputAction*             | nullptr       |
| JumpAction                    | The jump input action  | UInputAction*             | nullptr       |
| ShadowDecalClass              | The shadow decal class used to draw a shadow below the character while in the air  | TSubclassOf<AShadowDecal> |               |
| JumpSoundEffects              | An array of sound effects played when jumping                                      | TArray<USoundBase*>       |               |
| JumpParticleFx                | The particle effects spawned when jumping  | UNiagaraSystem*           | nullptr       |
| JumpParticleFxSpawnOffset     | The offset applied to the location of the jump particles when spawning             | FVector                   |               |
| LandedSoundEffects            | An array of sound effects played when landing                                      | TArray<USoundBase*>       |               |
| LandParticleFx                | The particle effects spawned when landing  | UNiagaraSystem*           | nullptr       |
| LandParticleFxSpawnOffset     | The offset applied to the location of the land particles when spawning             | FVector                   |               |
| VariableJumpHeight            | Does the character support variable jump height?                                   | bool                      | true          |
| VariableJumpHeightMaxHoldTime | The amount of time to hold the jump button to reach the max height                 | float                     | 0.3f          |
| JumpVelocity                  | The velocity applied to the character when jumping                                 | float                     | 500.0f        |
| AirControl                    | The amount of lateral movement control available to the character while in the air | float                     | 1000.0f       |
| FallingFriction               | The amount of friction to apply to lateral air movement when falling               | float                     | 3.5f          |
| Gravity                       | The amount of gravity applied to the character                                     | float                     | 2.0f          |
| CanDoubleJump                 | Can the character perform a double jump?   | bool                      | true          |
| CanCoyoteJump                 | Does the character support coyote time when trying to jump?                        | bool                      | true          |
| CoyoteJumpTime                | The amount of coyote time for the character  | float                     | 0.375f        |
| CoyoteJumpVelocity            | The velocity applied to the character when performing a coyote jump                | float                     | 700.0f        |
| BrakingFriction               | Friction coefficient applied when braking  | float                     | 10.0f         |
| MaxAcceleration               | The rate of change of velocity   | float                     | 2500.0f       |
| TargetArmLength               | Length of the spring arm component   | float                     | 350.0f        |
| ArmLengthMultiplier           | The multiplier applied to the spring arm component when the character is moving    | float                     | 0.4f          |
| CameraAdjustmentSpeed         | The speed used when adjusting the camera distance                                  | float                     | 3.0f          |

## Blueprint Usage There is no additional functions exposed to Blueprints. Just create the character and add it to your level

## 6.3 Animation Instance

### 6.3.1 Introduction

Base animation instance for a 3D platformer character

### 6.3.2 Dependencies

The `PlatformerAnimationInstance` relies on other components of this plugin to work:

- [Logger](#): Used to log useful information to help you debug any issues you might experience

### 6.3.3 API Reference

#### Properties

| Property            | Description  | Type           | Default Value        |
|---------------------|--|----------------|----------------------|
| IdleAnimation       | The animation used when the character is in the idle state | UAnimSequence* | <code>nullptr</code> |
| WalkAnimation       | The animation used when the character is walking           | UAnimSequence* | <code>nullptr</code> |
| RunAnimation        | The animation used when the character is running           | UAnimSequence* | <code>nullptr</code> |
| JumpAnimation       | The animation used when the character is jumping           | UAnimSequence* | <code>nullptr</code> |
| DoubleJumpAnimation | The animation used when the character is double jumping    | UAnimSequence* | <code>nullptr</code> |
| FallAnimation       | The animation used when the character is falling           | UAnimSequence* | <code>nullptr</code> |
| LandAnimation       | The animation used when the character is landing           | UAnimSequence* | <code>nullptr</code> |
| MovementSpeed       | The movement speed fo the character                        | float          |                      |
| IsInAir             | Is the character currently in the air?                     | bool           |                      |
| IsDoubleJumping     | Is the character double jumping?                           | bool           |                      |

### 6.3.4 Blueprint Usage

You can use this template by creating your own animation blueprint and selecting `UPlatformerAnimationInstance` as the parent class. Set your animations and use this for your 3D platformer character



## 6.4 Shadow Decal

### 6.4.1 Introduction

Decal used to draw a shadow beneath a character when the character is in the air

### 6.4.2 Dependencies

The `ShadowDecal` relies on other components of this plugin to work:

- [Logger](#): Used to log useful information to help you debug any issues you might experience

### 6.4.3 API Reference

#### Functions

| Name       | Description                 | Params  | Return |
|------------|-----------------------------|---|--------|
| Initialize | Initialize the shadow decal | <b>OwnerCharacter (ACharacter*)</b><br>The character owning this shadow decal |        |

### 6.4.4 Blueprint Usage

You can use the `ShadowDecal` using Blueprints by adding one of the following nodes:

- Ultimate Starter Kit > Shadow Decal > Initialize

### 6.4.5 C++ Usage

Before you can use the plugin, you first need to enable the plugin in your `Build.cs` file:

```
PublicDependencyModuleNames.Add("USK");
```

The `ShadowDecal` can now be used in any of your C++ files:

```
#include "USK/Character/ShadowDecal.h"

void ATestActor::Test()
{
    // ShadowDecal is a pointer to the AShadowDecal
    ShadowDecal->Initialize(OwnerCharacter);
}
```

## 7. UI & Widgets

### 7.1 Input Indicator

#### 7.1.1 Introduction

A widget used to display input indicators based on the current input device and input action

#### 7.1.2 Dependencies

The `InputIndicator` relies on other components of this plugin to work:

- **Logger:** Used to log useful information to help you debug any issues you might experience
- **Game Instance:** Used to monitor for input device changes and handle saving/loading game data

#### 7.1.3 Required Widgets

There is already a `InputIndicator_Implementation` that you can use in your projects. But if you create your own instance of this widget, you need to add the following before you can compile:

| Name      | Description                                   | Type            |
|-----------|---|-----------------|
| Container | The container used to display multiple images | UHorizontalBox* |

## API Reference ### Properties

| Property                | Description  | Type                             | Default Value        |
|-------------------------|--|----------------------------------|----------------------|
| InputIndicatorIconClass | The input indicator icon class                       | TSubclassOf<UInputIndicatorIcon> |                      |
| Action                  | The input action displayed by widget                 | UInputAction*                    | <code>nullptr</code> |
| Size                    | The size of the image                                | float                            | 50.0f                |
| Amount                  | The amount of images to display for the input action | int                              | 1                    |

### Functions

| Name         | Description                                     | Params  | Return |
|--------------|---|---|--------|
| UpdateAction | Update the input action displayed by the widget | <b>NewAction (UInputAction*)</b><br>The new action<br><br><b>NewAmount (int)</b><br>The new amount of images to display |        |

## Blueprint Usage You can use the `InputIndicator` using Blueprints by adding one of the following nodes:

- Ultimate Starter Kit > UI > Update Action

## C++ Usage Before you can use the plugin, you first need to enable the plugin in your `Build.cs` file:

```
PublicDependencyModuleNames.Add("USK");
```

The `InputIndicator` can now be used in any of your C++ files:

```
#include "USK/Widgets/InputIndicator.h"

void ATestActor::Test()
{
    // InputIndicator is a pointer to the UInputIndicator
```

```
InputIndicator->UpdateAction(NewAction, NewAmount);  
}
```

## 7.2 Input Indicator Icon

### 7.2.1 Introduction

A widget used to display a single input indicator image

### 7.2.2 Dependencies

The `InputIndicatorIcon` relies on other components of this plugin to work:

- [Logger](#): Used to log useful information to help you debug any issues you might experience

### 7.2.3 Required Widgets

There is already a `InputIndicatorIcon_Implementation` that you can use in your projects. But if you create your own instance of this widget, you need to add the following before you can compile:

| Name      | Description                                     | Type      |
|-----------|---|-----------|
| Container | The size box container used to resize the image | USizeBox* |
| Image     | The image used to display the input indicator   | UIImage*  |

## API Reference ### Properties

| Property | Description | Type | Default Value |
|----------|-------------|------|---------------|
|----------|-------------|------|---------------|

### Functions

| Name       | Description     | Params  | Return |
|------------|-----------------|---|--------|
| UpdateIcon | Update the icon | <b>Size (float)</b><br>The size of the image<br><br><b>Icon (UTexture2D*)</b><br>The new icon |        |

## Blueprint Usage You can use the `InputIndicatorIcon` using Blueprints by adding one of the following nodes:

- Ultimate Starter Kit > UI > Update Icon

## C++ Usage Before you can use the plugin, you first need to enable the plugin in your `Build.cs` file:

```
PublicDependencyModuleNames.Add("USK");
```

The `InputIndicatorIcon` can now be used in any of your C++ files:

```
#include "USK/Widgets/InputIndicatorIcon.h"

void ATestActor::Test()
{
    // InputIndicatorIcon is a pointer to the UInputIndicatorIcon
    InputIndicatorIcon->UpdateIcon(Size, Icon);
}
```

## 7.3 Menu

### 7.3.1 Introduction

A widget used to display menu items and handle navigation between the items

### 7.3.2 Dependencies

The `Menu` relies on other components of this plugin to work:

- [Logger](#): Used to log useful information to help you debug any issues you might experience

### 7.3.3 Optional Widgets

You can add the following widgets to enable extra functionality:

| Name            | Description   | Type        |
|-----------------|---|-------------|
| ScrollContainer | Scroll container used for large menus with many items | UScrollBox* |

## API Reference ### Properties

| Property              | Description   | Type                  | Default Value        |
|-----------------------|---|-----------------------|----------------------|
| AddInputBindingOnLoad | Should the input binding automatically be added as soon as the widget is loaded?                              | bool                  |                      |
| PauseGameWhileVisible | Should the game automatically be paused/resumed based on the visibility of the menu?                          | bool                  |                      |
| SelectedSFX           | The sound effect played when a menu item is selected  | USoundBase*           | <code>nullptr</code> |
| BackSFX               | The sound effect played when trying to go back to a previous menu or closing the menu through the back button | USoundBase*           | <code>nullptr</code> |
| InputMappingContext   | The input mapping context used to navigate the menu   | UInputMappingContext* | <code>nullptr</code> |
| MenuUpInputAction     | The input action used to navigate up  | UInputAction*         | <code>nullptr</code> |
| MenuDownInputAction   | The input action used to navigate down  | UInputAction*         | <code>nullptr</code> |
| MenuLeftInputAction   | The input action used to navigate left  | UInputAction*         | <code>nullptr</code> |
| MenuRightInputAction  | The input action used to navigate right   | UInputAction*         | <code>nullptr</code> |
| MenuSelectInputAction | The input action used to select a menu item   | UInputAction*         | <code>nullptr</code> |
| MenuBackInputAction   | The input action used to go back to a previous menu or close the menu   | UInputAction*         | <code>nullptr</code> |

### Events

| Name        | Description  | Params |
|-------------|--|--------|
| OnBackEvent | Event used to handle the back/close action of the menu |        |

### ### Functions

| Name           | Description                                  | Params | Return |
|----------------|--|--------|--------|
| OnMenuUp       | Navigate up                                  |        |        |
| OnMenuDown     | Navigate down                                |        |        |
| OnMenuLeft     | Navigate left                                |        |        |
| OnMenuRight    | Navigate right                               |        |        |
| OnMenuSelected | Select the current menu item                 |        |        |
| OnMenuBack     | Go back to a previous menu or close the menu |        |        |

## Blueprint Usage You can use the `Menu` using Blueprints by adding one of the following nodes:

- Ultimate Starter Kit > UI > On Menu Up
- Ultimate Starter Kit > UI > On Menu Down
- Ultimate Starter Kit > UI > On Menu Left
- Ultimate Starter Kit > UI > On Menu Right
- Ultimate Starter Kit > UI > On Menu Selected
- Ultimate Starter Kit > UI > On Menu Back

## C++ Usage Before you can use the plugin, you first need to enable the plugin in your `Build.cs` file:

```
PublicDependencyModuleNames.Add("USK");
```

The `Menu` can now be used in any of your C++ files:

```
#include "USK/Widgets/Menu.h"

void ATestActor::Test()
{
    // Menu is a pointer to the UMenu
    Menu->OnMenuUp();
    Menu->OnMenuDown();
    Menu->OnMenuLeft();
    Menu->OnMenuRight();
    Menu->OnMenuSelected();
    Menu->OnMenuBack();
}
```

## 7.4 Menu Item

---

### 7.4.1 Menu Navigation

---

#### Introduction

All the supported menu navigation types

#### Values

| Value                 | Description                     |
|-----------------------|---------------------------------|
| Disabled              | No navigation allowed           |
| HighlightItem         | Highlight a different menu item |
| IncreaseDecreaseValue | Increase or decrease the value  |

## 7.4.2 Menu Item Widget

### Introduction

A widget used to display a title, text and value in the form of a menu item

### Dependencies

The `MenuItem` relies on other components of this plugin to work:

- [Logger](#): Used to log useful information to help you debug any issues you might experience

### Required Widgets

You need to add the following before you can compile the `MenuItem` widget:

| Name       | Description   | Type        |
|------------|---|-------------|
| NormalText | The TextBlock used to display the text of the menu item while not highlighted | UTextBlock* |

## Optional Widgets You can add the following widgets to enable extra functionality:

| Name             | Description   | Type        |
|------------------|---|-------------|
| Title            | The TextBlock used to display the title of the menu item                  | UTextBlock* |
| HighlightedText  | The TextBlock used to display the text of the menu item while highlighted | UTextBlock* |
| BorderLeft       | The border displayed on the left of the menu item                         | UIImage*    |
| BorderRight      | The border displayed on the right of the menu item                        | UIImage*    |
| BorderBackground | The background border display in the menu item                            | UIImage*    |
| ButtonLeft       | The button displayed on the left of the menu item                         | UIImage*    |
| ButtonRight      | The button displayed on the right of the menu item                        | UIImage*    |
| ButtonBackground | The background button display in the menu item                            | UIImage*    |

## Optional Animations You can add the following widgets to enable extra functionality:

| Name                 | Description  |
|----------------------|--|
| HighlightedAnimation | The animation played when the menu item is highlighted |



## ## API Reference ### Properties

| Property                        | Description  | Type              | Default Value        |
|---------------------------------|--|-------------------|----------------------|
| HighlightedAnimation            | The animation played when the menu item is highlighted                           | UWidgetAnimation* | <code>nullptr</code> |
| FocusByDefault                  | Should the menu item be focused by default?                                      | bool              |                      |
| HideOnConsoles                  | Should the menu item be hidden on consoles?                                      | bool              |                      |
| TitleText                       | The title text displayed in the menu item  | FText             |                      |
| MenuItemText                    | The text displayed in the menu item  | FText             |                      |
| HighlightedSFX                  | The sound effect played when the menu item is highlighted                        | USoundBase*       | <code>nullptr</code> |
| BorderNormalColor               | The color of the border when not highlighted                                     | FLinearColor      |                      |
| BorderHighlightedColor          | The color of the border when highlighted   | FLinearColor      |                      |
| BorderNormalImage               | The image of the border when not highlighted                                     | UTexture2D*       | <code>nullptr</code> |
| BorderHighlightedImage          | The image of the border when highlighted   | UTexture2D*       | <code>nullptr</code> |
| BorderLeftNormalImage           | The image of the left border when not highlighted                                | UTexture2D*       | <code>nullptr</code> |
| BorderLeftHighlightedImage      | The image of the left border when highlighted                                    | UTexture2D*       | <code>nullptr</code> |
| BorderRightNormalImage          | The image of the right border when not highlighted                               | UTexture2D*       | <code>nullptr</code> |
| BorderRightHighlightedImage     | The image of the right border when highlighted                                   | UTexture2D*       | <code>nullptr</code> |
| BackgroundNormalColor           | The color of the button when not highlighted                                     | FLinearColor      |                      |
| BackgroundHighlightedColor      | The color of the button when highlighted   | FLinearColor      |                      |
| BackgroundNormalImage           | The image of the button when not highlighted                                     | UTexture2D*       | <code>nullptr</code> |
| BackgroundHighlightedImage      | The image of the button when highlighted   | UTexture2D*       | <code>nullptr</code> |
| BackgroundLeftNormalImage       | The image of the left button when not highlighted                                | UTexture2D*       | <code>nullptr</code> |
| BackgroundLeftHighlightedImage  | The image of the left button when highlighted                                    | UTexture2D*       | <code>nullptr</code> |
| BackgroundRightNormalImage      | The image of the right button when not highlighted                               | UTexture2D*       | <code>nullptr</code> |
| BackgroundRightHighlightedImage | The image of the right button when highlighted                                   | UTexture2D*       | <code>nullptr</code> |
| DefaultValue                    | The default value of the menu item   | int               | 100                  |
| MinValue                        | The minimum value of the menu item   | int               | 0                    |
| MaxValue                        | The maximum value of the menu item   | int               | 100                  |
| VerticalNavigation              | The type of navigation used by the menu item when pressing the up or down key    | EMenuNavigation   |                      |
| MenuItemUp                      | The menu item highlighted when the up key is pressed                             | UMenuItem*        | <code>nullptr</code> |
| MenuItemDown                    | The menu item highlighted when the down key is pressed                           | UMenuItem*        | <code>nullptr</code> |
| HorizontalNavigation            | The type of navigation used by the menu item when pressing the left or right key | EMenuNavigation   |                      |
| MenuItemLeft                    | The menu item highlighted when the left key is pressed                           | UMenuItem*        | <code>nullptr</code> |
| MenuItemRight                   | The menu item highlighted when the right key is pressed                          | UMenuItem*        | <code>nullptr</code> |

## ### Events

| Name            | Description   | Params   |
|-----------------|---|--|
| OnSelectedEvent | Event used to notify other classes that the menu item was selected        |  |
| OnValueChanged  | Event used to notify other classes that the menu item's value was updated | <b>Value (int)</b><br>The new value of the menu item |

## ### Functions

| Name                | Description                                | Params   | Return   |
|---------------------|--|--|--|
| SetText             | Set the text display in the menu item      | <b>Text (FText)</b><br>The new text displayed in the menu item   |  |
| SetHighlightedState | Set the highlighted state of the menu item | <b>IsHighlighted (bool)</b><br>Is the menu item highlighted?<br><br><b>PlayHighlightedSound (bool)</b><br>Should the highlighted sound effect be played? |  |
| GetValue            | Get the current value of the menu item     |  | <b>int</b><br>The current value of the menu item |
| UpdateValue         | Update the value of the menu item          | <b>IncreaseValue (bool)</b><br>Should the value be increased?  |  |

## Blueprint Usage You can use the `MenuItem` using Blueprints by adding one of the following nodes:

- Ultimate Starter Kit > UI > Set Text
- Ultimate Starter Kit > UI > Set Highlighted State
- Ultimate Starter Kit > UI > Get Value
- Ultimate Starter Kit > UI > Update Value

## C++ Usage Before you can use the plugin, you first need to enable the plugin in your `Build.cs` file:

```
PublicDependencyModuleNames.Add("USK");
```

The `MenuItem` can now be used in any of your C++ files:

```
#include "USK/Widgets/MenuItem.h"

void ATestActor::Test()
{
    // MenuItem is a pointer to the UMenuItem
    MenuItem->SetText(Text);
    MenuItem->SetHighlightedState(IsHighlighted, PlayHighlightedSound);
    int Value = MenuItem->GetValue();
    MenuItem->UpdateValue(IncreaseValue);
}
```

## 8. Utils

### 8.1 Config Utils

#### 8.1.1 Introduction

A Blueprint Function Library class used to extract config values

#### 8.1.2 Dependencies

The `ConfigUtils` relies on other components of this plugin to work:

- [Logger](#): Used to log useful information to help you debug any issues you might experience

#### 8.1.3 API Reference

##### Functions

| Name               | Description  | Params  | Return   |
|--------------------|--|---|--|
| GetConfigValue     | Extract a config value from a given config file          | <b>Filename (FString)</b><br>The name of the config file<br><br><b>Section (FString)</b><br>The section in the config file<br><br><b>Key (FString)</b><br>The key in the config file<br><br><b>DefaultValue (FString)</b><br>The default value to return if the config file can't be read | <b>FString</b><br>The value extracted from the config file |
| GetGameConfigValue | Extract a config value from the default game config file | <b>Section (FString)</b><br>The section in the config file<br><br><b>Key (FString)</b><br>The key in the config file<br><br><b>DefaultValue (FString)</b><br>The default value to return if the config file can't be read   | <b>FString</b><br>The value extracted from the config file |

#### 8.1.4 Blueprint Usage

You can use the `ConfigUtils` using Blueprints by adding one of the following nodes:

- Ultimate Starter Kit > Utils > Config > Get Config Value
- Ultimate Starter Kit > Utils > Config > Get Game Config Value

#### 8.1.5 C++ Usage

Before you can use the plugin, you first need to enable the plugin in your `Build.cs` file:

```
PublicDependencyModuleNames.Add("USK");
```

The `ConfigUtils` can now be used in any of your C++ files:

```
#include "USK/Utils/ConfigUtils.h"

void ATestActor::Test()
{
    FString ConfigValue = UConfigUtils::GetConfigValue(Filename, Section, Key, DefaultValue);
    FString GameConfigValue = UConfigUtils::GetGameConfigValue(Section, Key, DefaultValue);
}
```

## 8.2 Platform

---

### 8.2.1 Platform Type

---

#### Introduction

The types of supported platform types

#### Values

| Value       | Description                        |
|-------------|------------------------------------|
| Unknown     | An unknown or unsupported platform |
| Windows     | Windows (any architecture)         |
| MacOS       | MacOS (any architecture)           |
| Linux       | Linux (any architecture)           |
| Xbox        | Xbox console                       |
| Playstation | Playstation console                |
| Switch      | Nintendo Switch console            |
| Android     | Android (any architecture)         |
| IOS         | iOS                                |

## 8.2.2 Platform Utils

---

### Introduction

A Blueprint Function Library class used for platform detection

## API Reference

### FUNCTIONS

| Name          | Description                                 | Params | Return  |
|---------------|---|--------|---|
| GetPlatform   | Get the current platform                    |        | <b>EPlatform</b><br>The current platform  |
| IsInEditor    | Is the build running inside the editor?     |        | <b>bool</b><br>A boolean value indicating if the build is running inside the editor     |
| IsDesktop     | Is the build running on a desktop platform? |        | <b>bool</b><br>A boolean value indicating if the build is running on a desktop platform |
| IsWindows     | Is the build running on Windows?            |        | <b>bool</b><br>A boolean value indicating if the build is running on Windows            |
| IsMacOS       | Is the build running on MacOS?              |        | <b>bool</b><br>A boolean value indicating if the build is running on MacOS              |
| IsMacOSx86    | Is the build running on MacOS (x86)?        |        | <b>bool</b><br>A boolean value indicating if the build is running on MacOS (x86)        |
| IsMacOSArm    | Is the build running on MacOS (ARM)?        |        | <b>bool</b><br>A boolean value indicating if the build is running on MacOS (ARM)        |
| IsLinux       | Is the build running on Linux?              |        | <b>bool</b><br>A boolean value indicating if the build is running on Linux              |
| IsConsole     | Is the build running on a console platform? |        | <b>bool</b><br>A boolean value indicating if the build is running on a console platform |
| IsXbox        | Is the build running on Xbox?               |        | <b>bool</b><br>A boolean value indicating if the build is running on Xbox               |
| IsPlaystation | Is the build running on Playstation?        |        | <b>bool</b><br>A boolean value indicating if the build is running on Playstation        |
| IsSwitch      | Is the build running on Switch?             |        | <b>bool</b><br>A boolean value indicating if the build is running on Switch             |
| IsMobile      | Is the build running on a mobile platform?  |        | <b>bool</b><br>A boolean value indicating if the build is running on a mobile platform  |
| IsAndroid     | Is the build running on Android?            |        | <b>bool</b><br>A boolean value indicating if the build is running on Android            |
| IsAndroidx86  | Is the build running on Android (x86)?      |        | <b>bool</b><br>A boolean value indicating if the build is running on Android (x86)      |
| IsAndroidx64  | Is the build running on Android (x64)?      |        | <b>bool</b><br>A boolean value indicating if the build is running on Android (x64)      |
| IsAndroidArm  | Is the build running on Android (ARM)?      |        |   |



|                |  |  |  |
|----------------|--|--|--|
|                |  |  | <b>bool</b><br>A boolean value indicating if the build is running on Android (ARM)   |
| IsAndroidArm64 | Is the build running on Android (ARM64)? |  | <b>bool</b><br>A boolean value indicating if the build is running on Android (ARM64) |
| IsIOS          | Is the build running on iOS?             |  | <b>bool</b><br>A boolean value indicating if the build is running on iOS             |

## Blueprint Usage

You can use the `PlatformUtils` using Blueprints by adding one of the following nodes:

- Ultimate Starter Kit > Utils > Platform > Get Platform
- Ultimate Starter Kit > Utils > Platform > Is In Editor
- Ultimate Starter Kit > Utils > Platform > Is Desktop
- Ultimate Starter Kit > Utils > Platform > Is Windows
- Ultimate Starter Kit > Utils > Platform > Is MacOS
- Ultimate Starter Kit > Utils > Platform > Is MacOS (x86)
- Ultimate Starter Kit > Utils > Platform > Is MacOS (ARM)
- Ultimate Starter Kit > Utils > Platform > Is Linux
- Ultimate Starter Kit > Utils > Platform > Is Console
- Ultimate Starter Kit > Utils > Platform > Is Xbox
- Ultimate Starter Kit > Utils > Platform > Is Playstation
- Ultimate Starter Kit > Utils > Platform > Is Switch
- Ultimate Starter Kit > Utils > Platform > Is Mobile
- Ultimate Starter Kit > Utils > Platform > Is Android
- Ultimate Starter Kit > Utils > Platform > Is Android (x86)
- Ultimate Starter Kit > Utils > Platform > Is Android (x64)
- Ultimate Starter Kit > Utils > Platform > Is Android (ARM)
- Ultimate Starter Kit > Utils > Platform > Is Android (ARM64)
- Ultimate Starter Kit > Utils > Platform > Is iOS

## C++ Usage

Before you can use the plugin, you first need to enable the plugin in your `Build.cs` file:

```
PublicDependencyModuleNames.Add("USK");
```

The `PlatformUtils` can now be used in any of your C++ files:

```
#include "USK/Utils/PlatformUtils.h"

void ATestActor::Test()
{
    EPlatform Platform = UPlatformUtils::GetPlatform();
    bool IsInEditorValue = UPlatformUtils::IsInEditor();
    bool IsDesktopValue = UPlatformUtils::IsDesktop();
    bool IsWindowsValue = UPlatformUtils::IsWindows();
    bool IsMacOSValue = UPlatformUtils::IsMacOS();
    bool IsMacOSx86Value = UPlatformUtils::IsMacOSx86();
    bool IsMacOSArmValue = UPlatformUtils::IsMacOSArm();
    bool IsLinuxValue = UPlatformUtils::IsLinux();
    bool IsConsoleValue = UPlatformUtils::IsConsole();
    bool IsXboxValue = UPlatformUtils::IsXbox();
    bool IsPlaystationValue = UPlatformUtils::IsPlaystation();
    bool IsSwitchValue = UPlatformUtils::IsSwitch();
}
```

```
bool IsMobileValue = UPlatformUtils::IsMobile();
bool IsAndroidValue = UPlatformUtils::IsAndroid();
bool IsAndroidx86Value = UPlatformUtils::IsAndroidx86();
bool IsAndroidx64Value = UPlatformUtils::IsAndroidx64();
bool IsAndroidArmValue = UPlatformUtils::IsAndroidArm();
bool IsAndroidArm64Value = UPlatformUtils::IsAndroidArm64();
bool IsIOSValue = UPlatformUtils::IsIOS();
}
```

## 8.3 Project Utils

### 8.3.1 Introduction

A Blueprint Function Library class used to extract project values

### 8.3.2 API Reference

#### Functions

| Name                      | Description  | Params | Return   |
|---------------------------|--|--------|--|
| GetProjectId              | Get the project ID from the game config file               |        | <b>FString</b><br>The project ID               |
| GetProjectName            | Get the project name from the game config file             |        | <b>FString</b><br>The project name             |
| GetProjectDescription     | Get the project description from the game config file      |        | <b>FString</b><br>The project description      |
| GetProjectVersion         | Get the project version from the game config file          |        | <b>FString</b><br>The project version          |
| GetProjectCompanyName     | Get the project company name from the game config file     |        | <b>FString</b><br>The project company name     |
| GetProjectCopyrightNotice | Get the project copyright notice from the game config file |        | <b>FString</b><br>The project copyright notice |
| GetProjectLicensingTerms  | Get the project licensing terms from the game config file  |        | <b>FString</b><br>The project licensing terms  |
| GetProjectHomepage        | Get the project homepage from the game config file         |        | <b>FString</b><br>The project homepage         |

### 8.3.3 Blueprint Usage

You can use the `ProjectUtils` using Blueprints by adding one of the following nodes:

- Ultimate Starter Kit > Utils > Project > Get Project Id
- Ultimate Starter Kit > Utils > Project > Get Project Name
- Ultimate Starter Kit > Utils > Project > Get Project Description
- Ultimate Starter Kit > Utils > Project > Get Project Version
- Ultimate Starter Kit > Utils > Project > Get Project Company Name
- Ultimate Starter Kit > Utils > Project > Get Project Copyright Notice
- Ultimate Starter Kit > Utils > Project > Get Project Licensing Terms
- Ultimate Starter Kit > Utils > Project > Get Project Homepage

### 8.3.4 C++ Usage

Before you can use the plugin, you first need to enable the plugin in your `Build.cs` file:

```
PublicDependencyModuleNames.Add("USK");
```

The `ProjectUtils` can now be used in any of your C++ files:

```
#include "USK/Utils/ProjectUtils.h"

void ATestActor::Test()
{
    FString ProjectId = UProjectUtils::GetProjectId();
    FString ProjectName = UProjectUtils::GetProjectName();
    FString ProjectDescription = UProjectUtils::GetProjectDescription();
    FString ProjectVersion = UProjectUtils::GetProjectVersion();
    FString ProjectCompanyName = UProjectUtils::GetProjectCompanyName();
    FString ProjectCopyrightNotice = UProjectUtils::GetProjectCopyrightNotice();
    FString ProjectLicensingTerms = UProjectUtils::GetProjectLicensingTerms();
    FString ProjectHomepage = UProjectUtils::GetProjectHomepage();
}
```