Using Slate In-Game

Using Slate UI widgets for in-game user interfaces.

You can use **Slate widgets** to create Heads Up Displays (HUDs) or other User Interface (UI) elements, such as menus. These UI consist of one or more container widgets, each of which may be hold several other types of widgets that are responsible for a particular aspect of the UI.

For example, you may have a single overall widget for your game's HUD, as well as widgets for the main menu, options menu, pause menu, scoreboard, and so on. Each of these widgets would then be made up of a combination of other custom widgets, labels, text boxes, images, and other types of elements.

Each of these container widgets can then be added or removed to the user's **Viewport** depending on the circumstances in the game:

- When the game starts, it adds a main menu widget to the viewport.
- When the user chooses one of the options in the menu, the main menu widget would then be removed from the viewport.
    - If the option opens another menu, the new menu is added to the viewport.
- If the player pauses the game at any point, the pause menu widget would be added to the viewport.
- When the game is resumed, the pause menu widget is removed from the viewport.

- When the HUD for the player is initialized, the HUD widget would be added to the viewport.

Project Setup

In order to use the Slate User Interface (UI) Framework, your project must be set up properly so that it is aware of the framework. This allows you to include the Slate.h header and reference the various elements of the framework necessary for building a UI with slate.

Module Dependencies

The Slate framework is stored in a few modules. In order to make your project aware of these, some dependencies must be set up in the *.build.cs file for your project.

The modules your project needs access to are:

Module	Dependency Type
InputCore	Public
Slate	Private
SlateCore	Private

To set up the Slate module dependencies:

1. Open the [ProjectName].build.cs file for your project. It is located in the [ProjectDir]/[ProjectName]/Source/[ProjectName] directory.
2. Add the InputCore public dependency by adding `"InputCore"` to the `PublicDependencyModuleNames`.

```
1 PublicDependencyModuleNames.AddRange(new string[] { "Core",
  "CoreUObject", "Engine", "InputCore" });
2
```

Copy full snippet

The InputCore module is set as a public dependency by default when code projects are created.

3. Add Slate and SlateCore private dependencies. A line exists in the *.build.cs file for adding private dependencies:

```
PrivateDependencyModuleNames.AddRange(new string[] { });

```

Copy full snippet

Add the SlateCore and Slate modules to that line:

```
PrivateDependencyModuleNames.AddRange(new string[] { "Slate", "SlateCore" });

```

Copy full snippet

Depending on when you created your project, and with what version of the engine, it may already have the Slate dependencies set up in the *.build.cs files but commented out. You can uncomment the appropriate lines to set up the dependencies.

```
// Uncomment if you are using Slate UI
// PrivateDependencyModuleNames.AddRange(new string[] { "Slate", "SlateCore" });
```

Copy full snippet

Displaying Widgets

To display a Slate widget in your game, You must add it to the game's viewport. Overlaid widgets are ordered according to the Z-order specified when adding them, with larger Z-

order values appearing on top of smaller Z-orders.

Accessing the Game Viewport

The game's viewport is an instance of the `GameViewportClient` class. You can access a reference to the current game viewport using the `GameViewport` member of `UEngine`, which can be accessed using the `GEngine` global pointer to the current `UEngine` instance for the game.

For example:

```
GEngine->GameViewport

```

Copy full snippet

> `GEngine` and `GameViewport` can be `NULL` so you should always check their values before trying to access them or any of their members.

Adding the Widget to the Viewport

Slate widgets are added to the viewport by passing a reference (a `TSharedref<SWidget>`) to the widget to `GameViewportClient::AddViewportWidgetContent()`. This function takes in both a widget and a Z-order, which determines the sorting order for the new widget as mentioned previously. The Z-order is optional, however, and defaults to a value of `0`.

The reference to the widget you want to add to the viewport can be stored as a member of some class, such as your HUD, or it can be created and passed in at the time of calling the function.

Passing a reference to a widget stored in a member variable (as a `TSharedPtr`):

```
GEngine->GameViewport->AddViewportWidgetContent(
SNew(MyWidgetPtr.ToSharedRef())
);
```

```
4
```

Copy full snippet

Creating the widget using `SNew()` when passing it to `GameViewportClient::AddViewportWidgetContent()`:

```
1 GEngine->GameViewport->AddViewportWidgetContent(
2 SNew(SWeakWidget)
3 .PossiblyNullContent(MyWidgetClass)
4 );
5
```

Copy full snippet

Or using `SAssignNew()` to create it, assign it to a `TSharedPtr` member, and pass it in:

```
1 GEngine->GameViewport->AddViewportWidgetContent(
2 SAssignNew(MyWidgetPtr, SWeakWidget)
3 .PossiblyNullContent(MyWidgetClass)
4 );
5
```

Copy full snippet

Removing Widgets from the Viewport

You can remove Slate widgets from the viewport individually by passing a reference to a previously added widget to `GameViewportClient::RemoveViewportWidgetContent()`.

For example:

```
1 GEngine->GameViewport->RemoveViewportWidgetContent(
2 SNew(MyWidgetPtr.ToSharedRef())
3 );
4
```

Copy full snippet

In addition, you can remove all widgets from the viewport at once by calling `GameViewportClient::RemoveAllViewportWidgets()`

For example:

```
1 GEngine->GameViewport->RemoveAllViewportWidgets();
2
```

Copy full snippet

`GEngine` and `GameViewport` can be `NULL` so you should always check their values before trying to access them or any of their members.