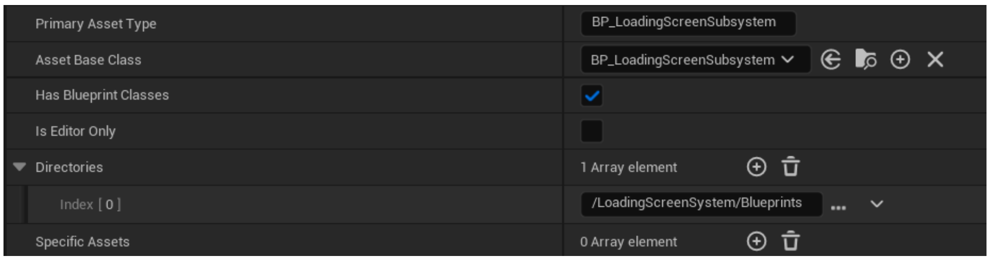
LoadingScreenSystem

Introduction

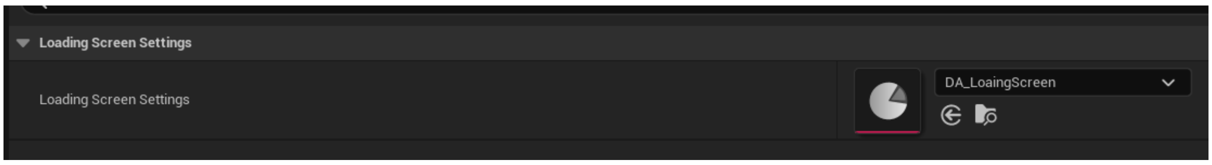
LoadingScreenSystem is a plug-in that handles the display and hiding of the loading interface when switching levels. It can be seamlessly added to the project, the loading interface at the start of the game can be customized, and the loading interface can be customized for different maps.

use

1. Turn on the LoadingScreenSystem plug-in
2. Add BP_LoadingScreenSubsystem to Asset Manager
   a. In Project Settings->Asset Manager, click Add Primary Asset Types to Scan
   b. Fill in the following information, as shown below:
      i. Primary Asset Type：BP_LoadingScreenSubsystem
      ii. Asset Base Type：BP_LoadingScreenSubsystem
      iii. Has Blueprint Classes：True
      iv. Directories：/LoadingScreenSystem/Blueprints
   c. 

Primary Asset Type	BP_LoadingScreenSubsystem
Asset Base Class	BP_LoadingScreenSubsystem
Has Blueprint Classes	✓
Is Editor Only	
Directories	1 Array element
Index [0]	/LoadingScreenSystem/Blueprints
Specific Assets	0 Array element

3. Create a Data Asset based on LoadingScreenSettings and put it in the LoadingScreenSettings of BP_LoadingScreenSubsystem, as shown in the figure:



Loading Screen Settings

Loading Screen Settings	DA_LoaingScreen

4. Set custom data in this Data Asset

extra information

1. LoadingScreenShown: Dynamic multicast delegate, broadcast when LoadingScreen is displayed
2. LoadingScreenHidden: dynamic multicast delegate, broadcast when LoadingScreen is hidden

Multiplayer screen synchronization is turned off (under testing)

1. Only valid for ServerTravel
2. Under testing, may be unstable
3. Steps for usage
    a. Add a reference to the module LoadingScreenSystem in the project's .builds.cs. PrivateDependencyModuleNames.AddRange(new string[] { "LoadingScreenSystem" });
    b. For PlayerController and GameState in the two converted Levels
    c. Add the following code to PlayerController

```
// UFUNCTION(Server, Reliable, Category = "LoadingScreen")
// void Server_LocalPlayerLoadingComplete();

// void Server_LocalPlayerLoadingComplete_Implementation()
// {
//  if(AGameStateBase* GameState = GetWorld()->GetGameState())
//  {
//     UFunction* Func = GameState->FindFunction(FName("Multi_OneDeviceLoadingComplete"));
//     if(!Func)
//     {
// UE_LOG(
//             LogTemp,
xi.    //           Warning,
xii.   //           TEXT("Function not found.")
xiii.  //           );
xiv.   //        return;
xv.    //     }
xvi.   //     struct FDynamicInterfaceParams{};
xvii.  //     FDynamicInterfaceParams Params = {};
xviii. //
xix.   //     GameState->ProcessEvent(Func, &Params);
xx.    //  }
xxi.   // }
```

d. Add the following code to Gamestate

```
i.     //UFUNCTION(BlueprintCallable, NetMulticast, Reliable,
       Category = "LoadingScreen")
ii.    // void Multi_EnableWaitHidingLoadingScreen();

       //void
       Multi_EnableWaitHidingLoadingScreen_Implementation()
iii.   // {
iv.    //  int32 CurrentPlayerNums =
       GetWorld()->GetGameState()->PlayerArray.Num();
v.     // if(GEngine)
vi.    //  {
vii.   //     if(auto LoadingScreenSubsystem =
       GetGameInstance()->GetSubsystem<ULoadingScreenSubsystem>(
       ))
viii.  //     {
ix.    //
       LoadingScreenSubsystem->EnableWaitHidingLoadingScreen(Cur
       rentPlayerNums);
x.     //     }
xi.    //  }
xii.   // }
xiii.  //
xiv.   // UFUNCTION(NetMulticast, Reliable, Category =
       "LoadingScreen")
xv.    // void Multi_OneDeviceLoadingComplete();

       // void Multi_OneDeviceLoadingComplete_Implementation()
```

xvi.
```
// {
```
xvii.
```
//  if(auto LoadingScreenSubsystem =
GetGameInstance()->GetSubsystem<ULoadingScreenSubsystem>(
))
```
xviii.
```
//  {
```
xix.
```
//
LoadingScreenSubsystem->OneDeviceLoadingComplete();
```
xx.
```
//  }
```
xxi.
```
// }
```

e. Call GameState's Multi_EnableWaitHidingLoadingScreen before executing ServerTravel