SaveLoadSystem

Introduction

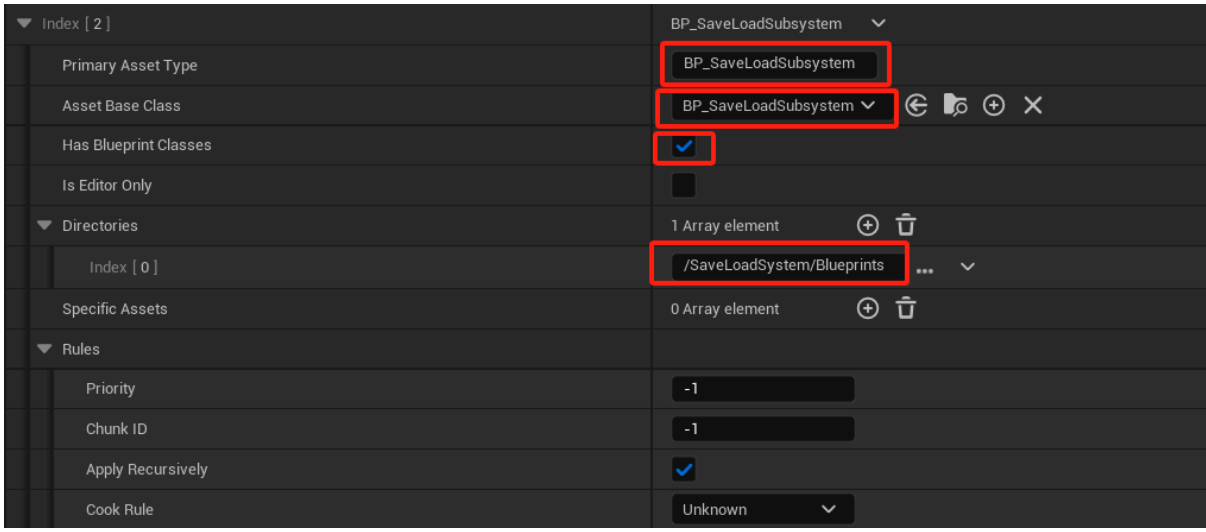
SaveLoadSystem Plugin provides flexible functions for saving and loading game state. The plugin supports both Blueprint and C++ projects. Saving and loading can be realized according to the implementation of the interface, and it supports Actor implementation interface and Widget implementation interface.Highly scalable and available everywhere. Provides SaveGame queries at edit time and run time.

Use method

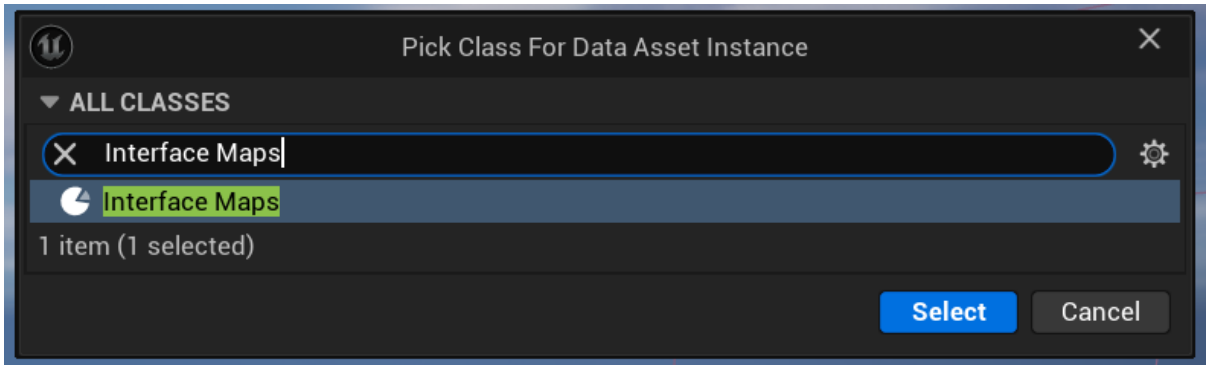
Preconditions

1. Create a Blueprint or C++ project
2. Install and open the SaveLoadSystem plugin
3. Add BP_SaveLoadSubsystem to Asset Manager
    a. Go to Project Settings -> Asset Manager page
    b. AddedPrimary Asset Types to Scan

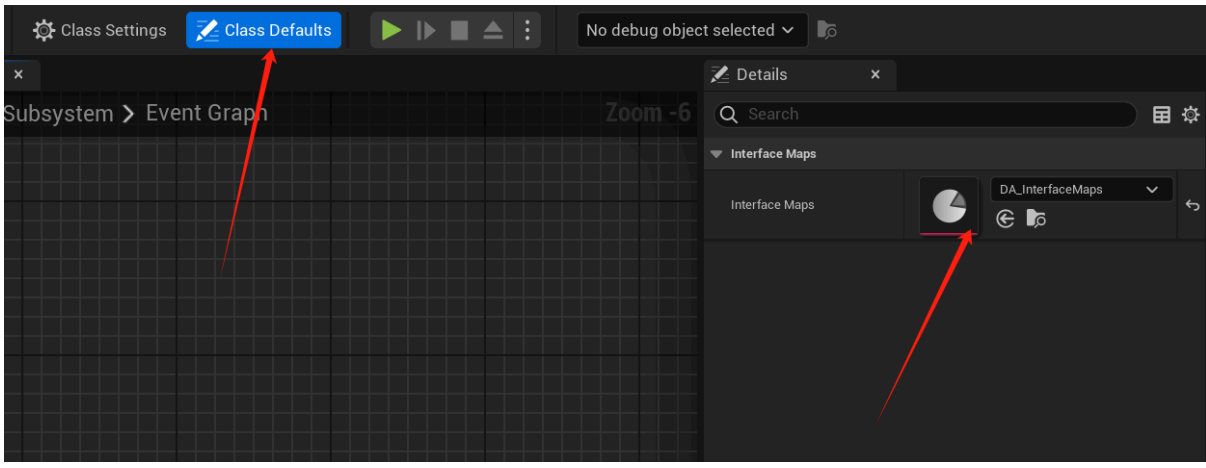
c. Fill in the information according to the picture below



4. Create a Data Asset based on Interface Maps at your preferred location

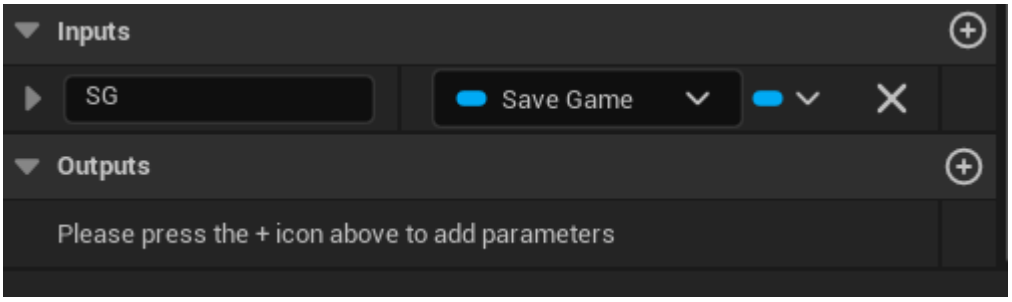
a. 

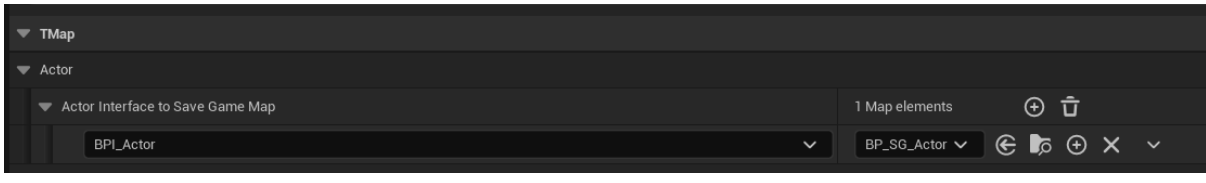
5. Open BP_SaveLoadSubsystem and assign this Data Asset to Interface Maps

a. 

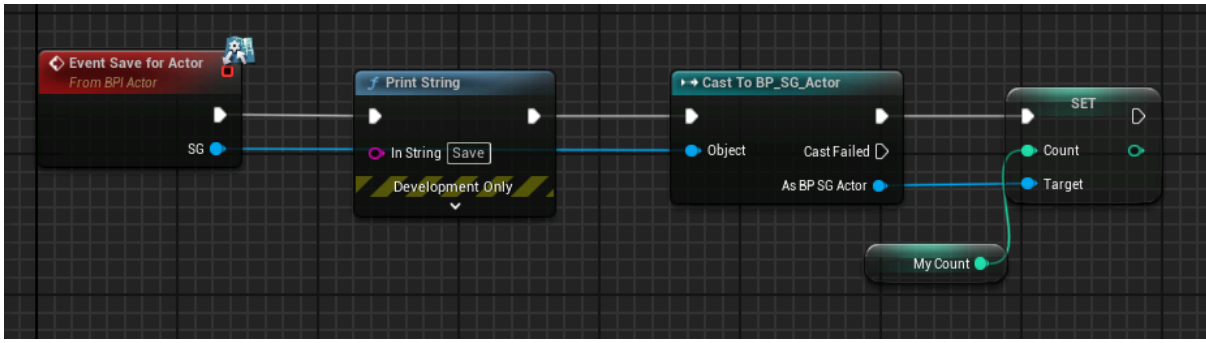
Actor implements interface

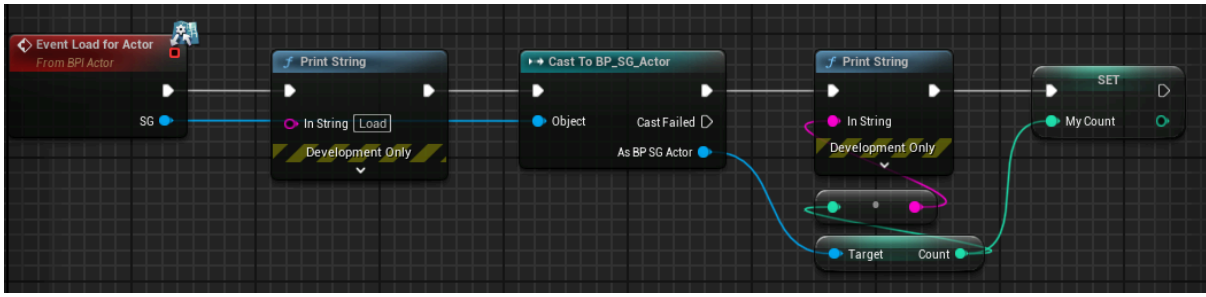
1. Create a Save Game object to store the data you want to save and load.
2. Create a Blueprint Interface and declare two functions for Sava and Load. The requirements are as follows:
    a. Save function
        i. Name changed to SaveForActor
        ii. There is a formal parameter: Save Game Object Reference
    b. Load function:
        i. Name changed to LoadForActor
        ii. There is a formal parameter: Save Game Object Reference
    c. As shown in the picture:
        i. 

        ii. 

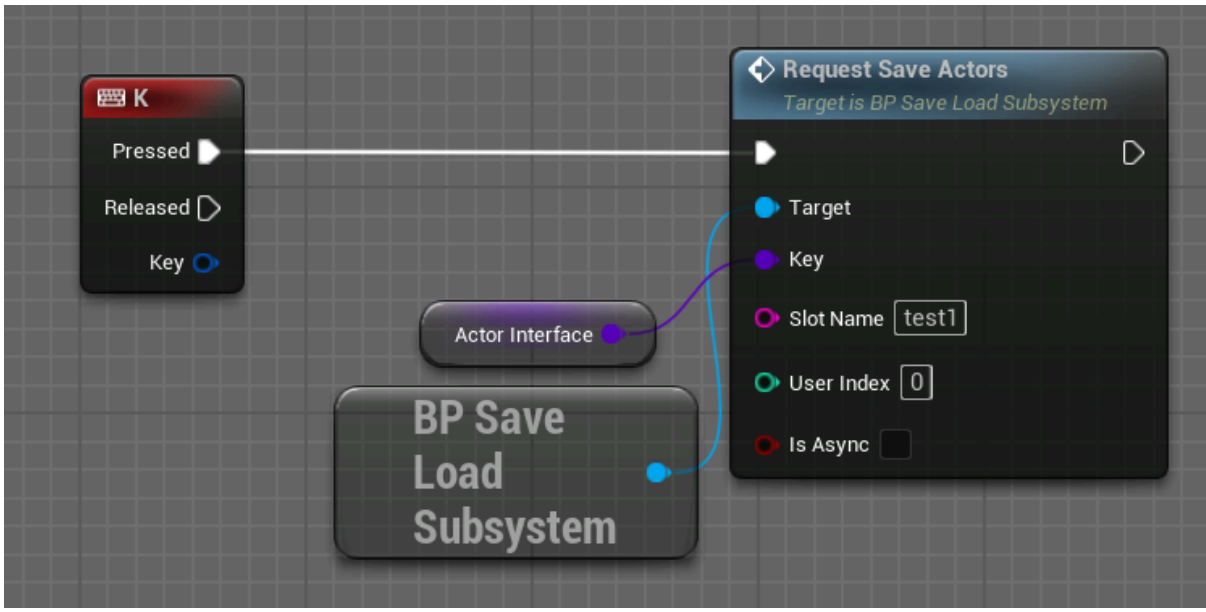
3. Open the Data Asset created previously, add an ActorInterfaceToSaveGameMap element, put Blueprint Interface into Key, and Save Game into Value. As shown below:
    a. 

4. Create an Actor to implement Blueprint Interface, as shown below:

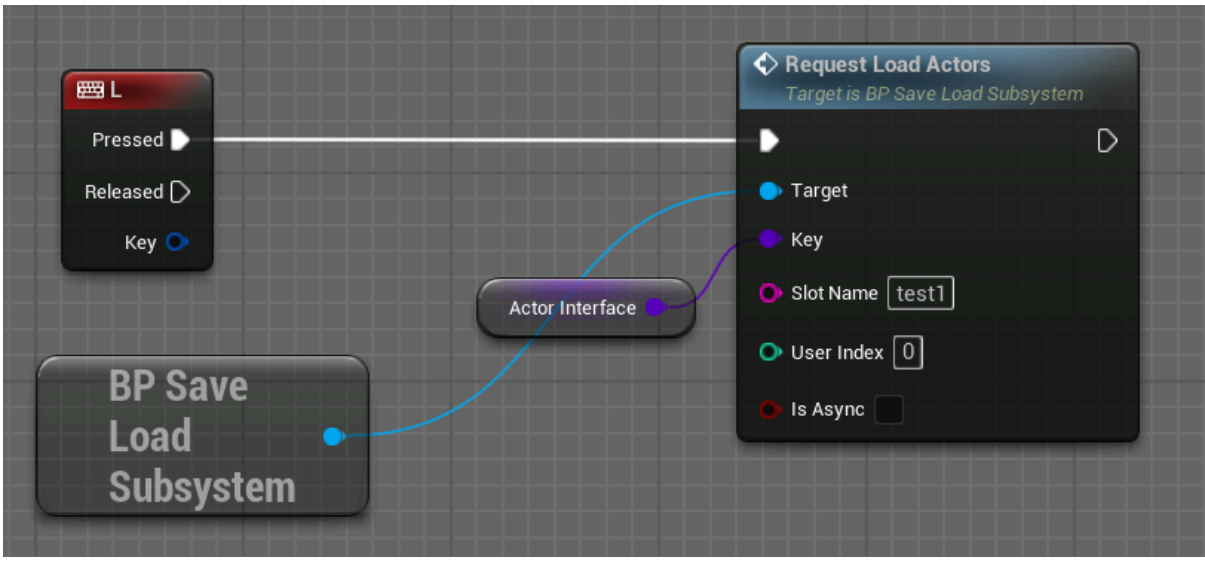
a. 

b. 

5. Finally, at the right time, request the Save and Load functions.
   a. Request Save function:
   BP_SaveLoadSubsystem->RequestSaveActors



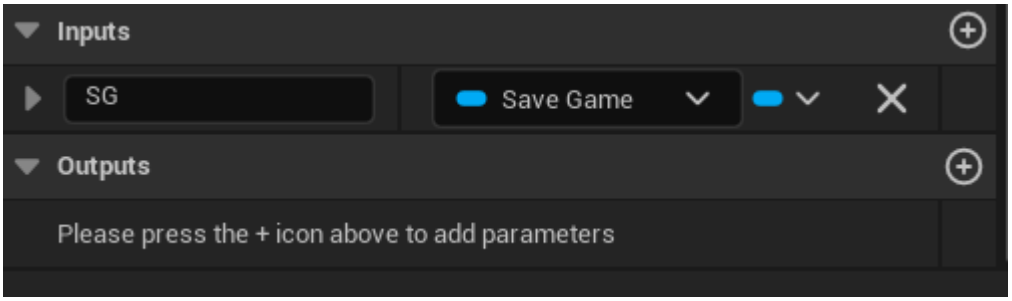
b. Request Load function:
BP_SaveLoadSubsystem->RequestLoadActors



Widget implements interface

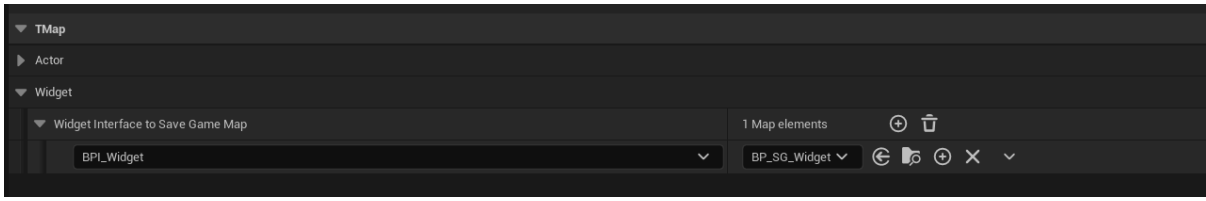
1. Create a Save Game object to store the data you want to save and load.
2. Create a Blueprint Interface and declare two functions for Sava and Load. The requirements are as follows:
   a. Save function
      i. Name changed to SaveForWidget
      ii. There is a formal parameter: Save Game Object Reference
   b. Load function:
      i. Name changed to LoadForWidget
      ii. There is a formal parameter: Save Game Object Reference
   c. As shown in the picture:
      i.

ii.



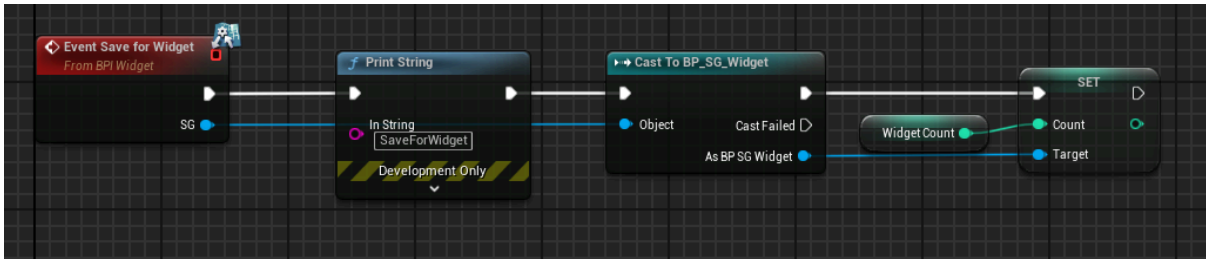
3. Open the Data Asset created previously, add a WidgetInterfaceToSaveGameMap element, put Blueprint Interface into Key, and Save Game into Value. As shown below:

a.

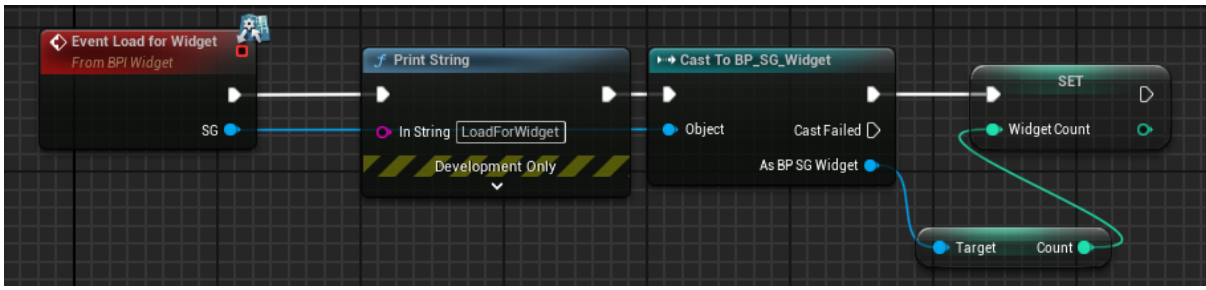


4. Create a Widget and implement Blueprint Interface, as shown below:

a.

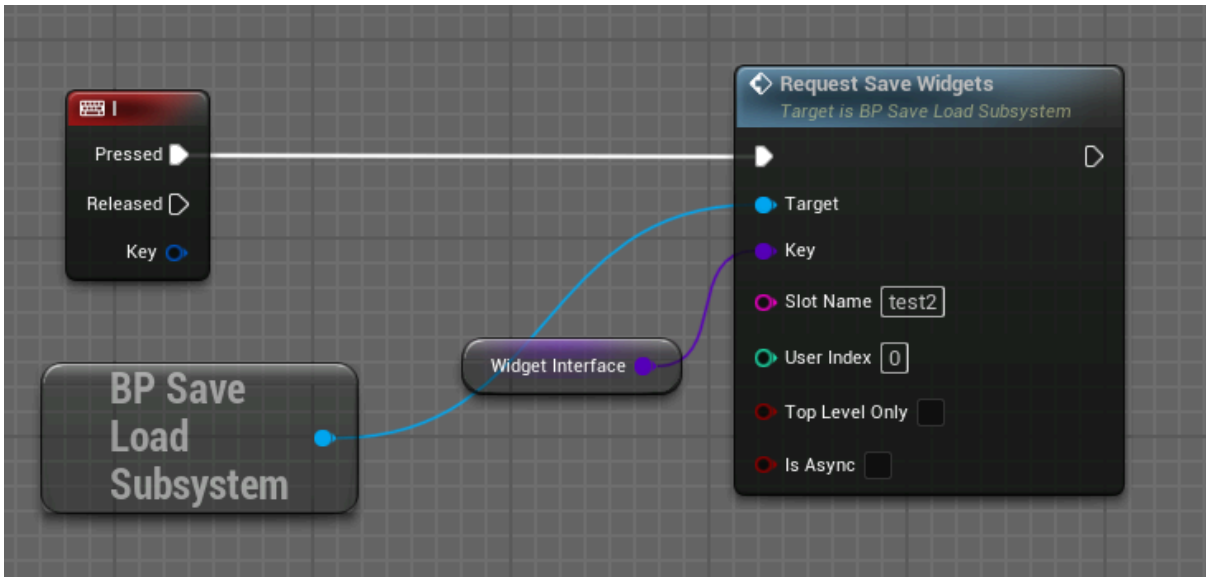


b.

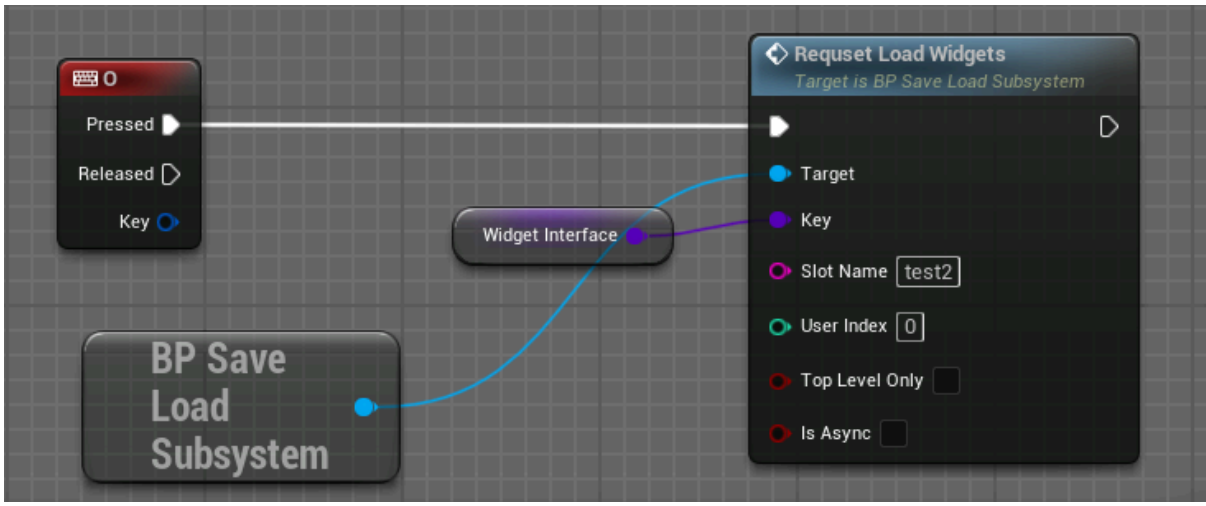


5. Finally, at the right time, request the Save and Load functions

a. Request Save function:
   BP_SaveLoadSubsystem->RequestSaveWidgets



b. Request Load function:
   BP_SaveLoadSubsystem->RequestSaveWidgets



Query Save Game at Runtime

1. Find all Save Games:USaveLoadSubsystem::GetAllSaveGames
2. Find Save Game containing Name: USaveLoadSubsystem::GetSaveGamesIncludeName
3. Find Save Game that does not contain Name: USaveLoadSubsystem::GetSaveGamesExcludeName

Other Function

1. Delete Save Game function：
   BP_SaveLoadSubsystem->RequestDeleteGame
2. Initialization function:
   BP_SaveLoadSubsystem->OnSubsystemInitialized. You can customize the logic inside
3. Deinitialization function:
   BP_SaveLoadSubsystem->OnSubsystemDeinitialized. You can customize the logic inside

Question

1. Can interfaces be defined in C++?
    a. Can't. Interfaces can only be defined using blueprints and must meet conditions before they will take effect.