Splash & Loading

1. Introduction

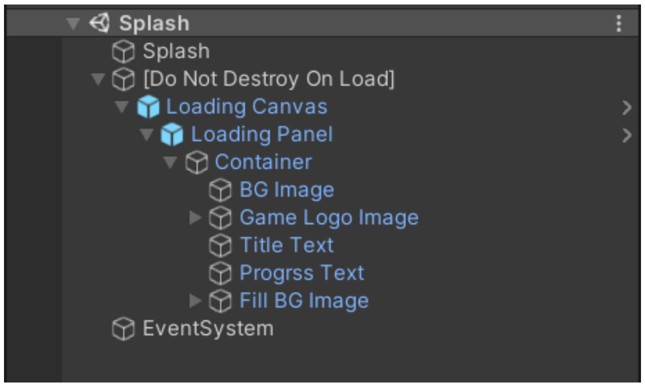
Splash and Loading screen is essential for all the games. To keep the user engaged while we're loading the levels, controllers and other stuff of game, it is very important to provide some sort of feedback to user. Loading screen is one way to do so.

This module is intended to make loading easy, generic and reusable.

2. Splash Scene

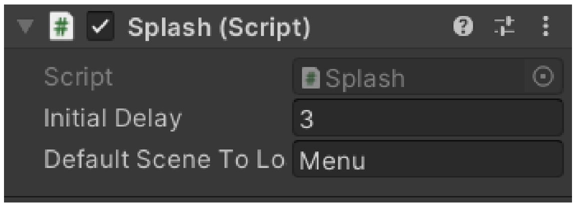
Import "Splash and Loading V1.0.0.unitypackage" to your project.

Open the scene named Splash present in "_CGS/Scenes/Splash". This scene contains the following hierarchy.



```
▼ Splash
    Splash
  ▼ [Do Not Destroy On Load]
    ▼ Loading Canvas
      ▼ Loading Panel
        ▼ Container
            BG Image
          ▶ Game Logo Image
            Title Text
            Progrss Text
          ▶ Fill BG Image
    EventSystem
```

There is a gameObject named "Splash" and script named "Splash.cs" attached to it. Select the gameObject and you'll see the following inspector window.



```
Splash (Script)
Script                Splash
Initial Delay         3
Default Scene To Lo   Menu
```

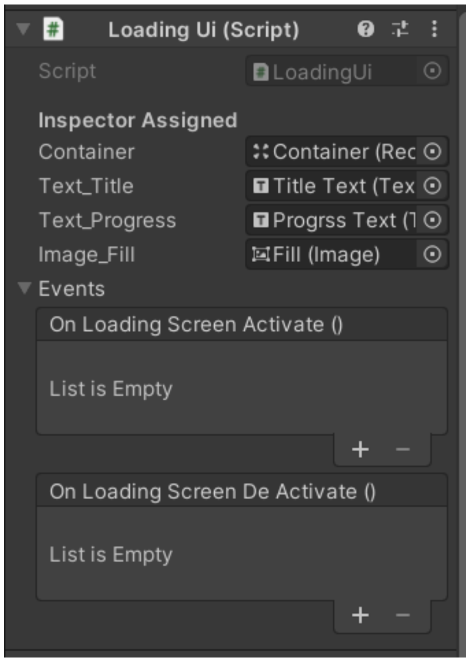
This script exposes two fields in inspector the "Initial Delay" and the "Default Scene To Load". "Initial Delay" contains the time in seconds to hold the application on this screen. "Default Scene To Load" will contain the scene name to load right after the splash screen. So in above example the application will stay on splash scene for three seconds and then start loading the scene named "Menu".

Next is "[Do Not Destroy On Load]" gameObject. This object has "DoNotDestoryOnLoad.cs" script attached to it which make sures that this object will not destroy between scene loading and so its child gameObjecs.

In its child there's a Loading Canvas and that canvas has a Loading Panel as its child. So, this Loading Canvas will persist between scenes and will be available throughout the life cycle of the game. Therefore, you can call the loading screen at any moment in any scene via singleton instance of LoadingUi.cs

3. LoadingUi.cs

This script has exposed following interface in inspector.



Loading Ui (Script)

Script	LoadingUi
Inspector Assigned	
Container	Container (Rec
Text_Title	Title Text (Tex
Text_Progress	Progrss Text (T
Image_Fill	Fill (Image)

Events

On Loading Screen Activate ()

List is Empty

On Loading Screen De Activate ()

List is Empty

First there are few references for loading title text, progress text and fill bar etc. You can add more fields as per your requirements. This script also have public events i.e. "OnLoadingScreenActivate" and "OnLoadingScreenDeActivate" so that you want this script to do certain stuff on loading begins and on loading ends just register a function to these events. We'll see an example in a while.

Singleton

public static LoadingUi instance;

Public Methods

public void LoadScene(string sceneToLoad, string title = "Loading…", bool disableLoadingWhenDone = true);

This method takes name of scene to load. A title of loading screen which is defaulted to "Loading…" and a Boolean parameter to specify that when the scene loads whether or not it should disable the loading screen, default value of this parameter is true.

Example:

```
LoadingUi.instance.LoadScene("GameOver");

LoadingUi.instance.LoadScene("GamePlay", "Loading Map…", true);
```

public void LoadScene(int sceneToLoad, string title = "Loading…", bool disableLoadingWhenDone = true);

This is just an overloaded variant of the previous methods. Here instead of name, it takes index of scene to load instead.

Example:

```
LoadingUi.instance.LoadScene("GameOver");
```

public void ActivateLoadingScreen();

This method will activate the loading screen so that you can do stuff behind the scene like reposition the player etc. But keep in mind that you have to assign the loading progress and title manual via "**UpdateProgress(float progress, string title)**" method while performing your background tasks.

public void DeActivateLoadingScreen();

If you have loaded a scene via LoadScene(…) function as described above and have set it's disableWhenLoaded parameter to false. Then use this method to disable the loading screen.

This method will de activate the loading screen. This method is useful if you're handling the loading manually.