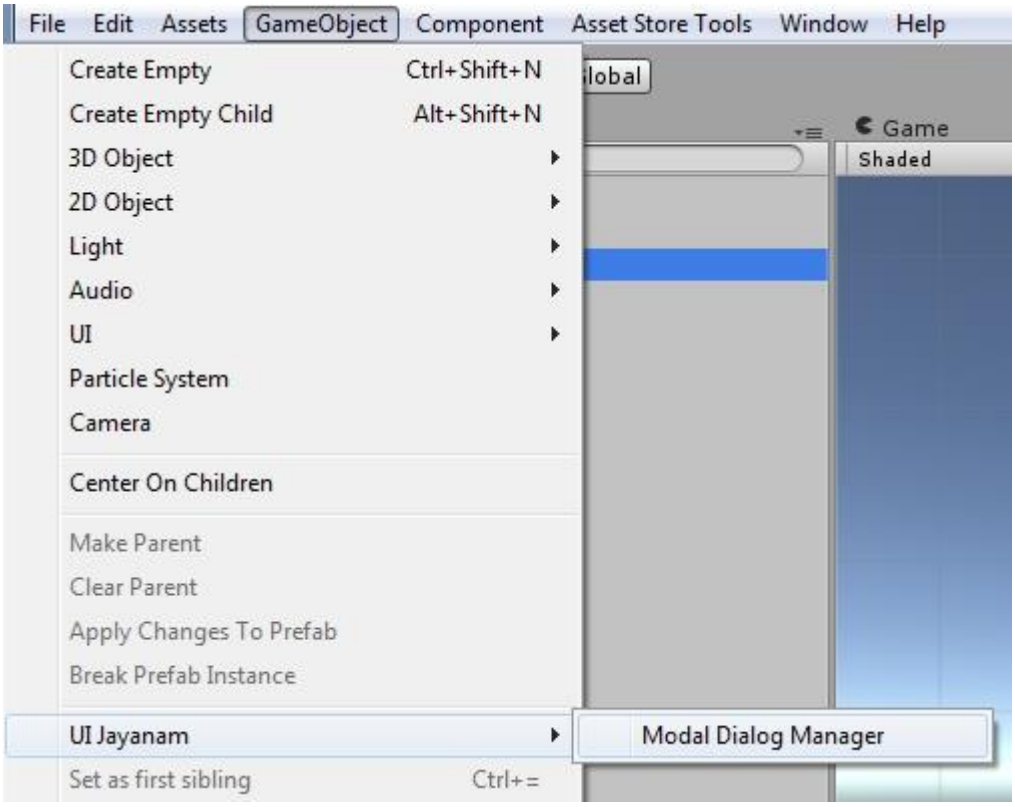
UI Modal Dialog for Unity

1 GETTING STARTED

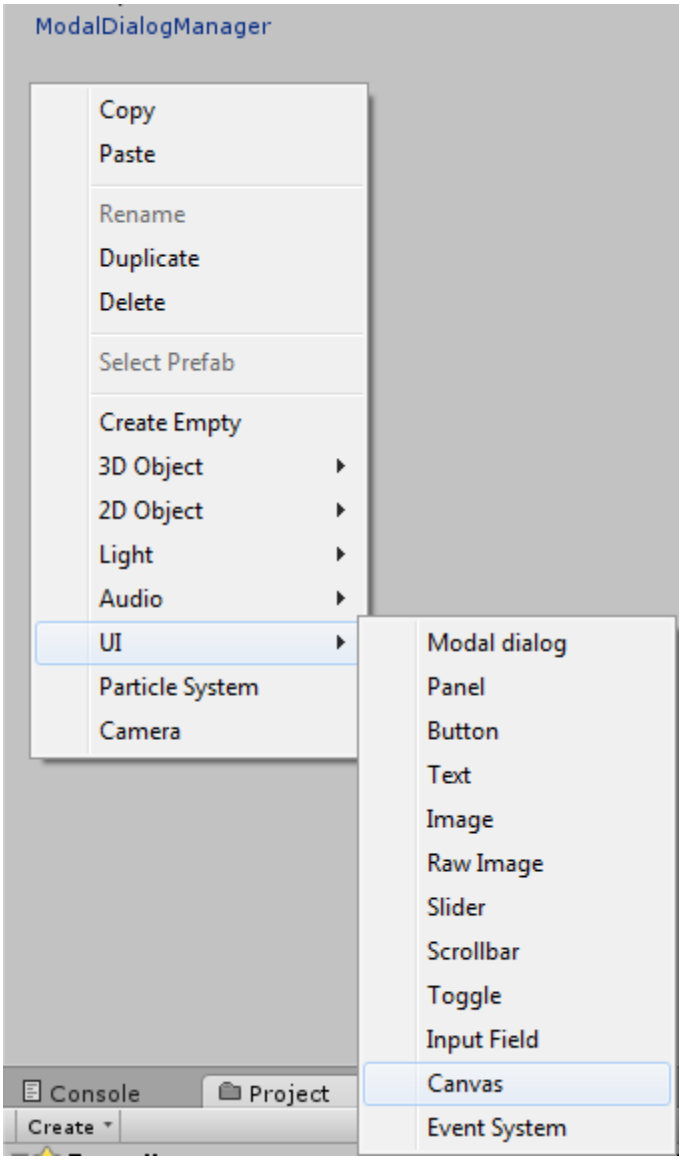
This chapter is about getting you started using the asset **UI Modal Dialog** to create modal dialogs within Unity. After you imported the package, the first thing you need to do is to add a *ModalDialogManager* gameobject to your scene.

1.1 ADD MODALDIALOGMANAGER TO THE SCENE

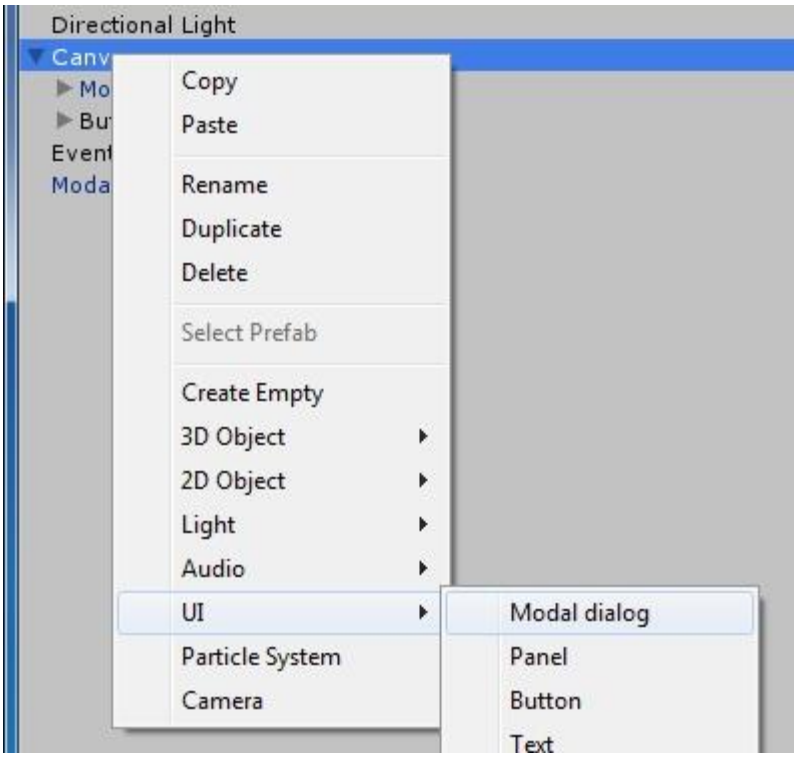


1.2 ADD A DEFAULT MODAL DIALOG

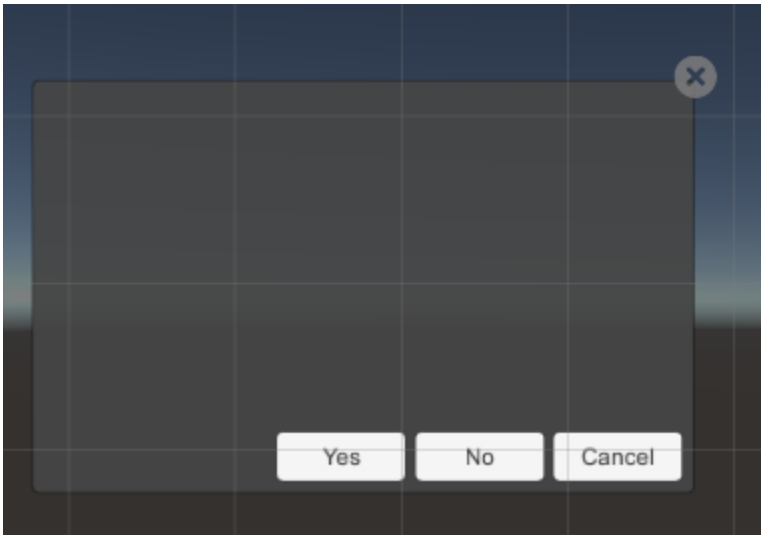
After that you need to add a Unity UI Canvas component. Right click in the Unity Hierarchy Window and select the Canvas item from the Context Menu:



Select the Canvas and then add the modal dialog from the UI Context Menu:



After these steps a default modal dialog with the button *Yes, No* and *Cancel* is added to your Canvas:



1.3 OPEN A MODAL DIALOG

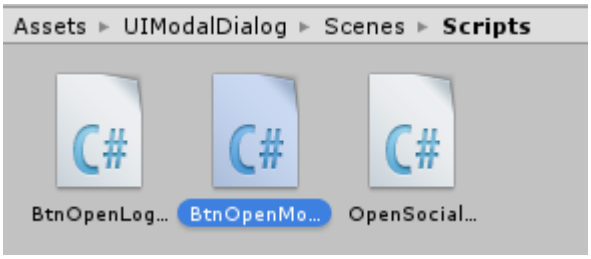
A modal dialog can be opened by this line of Code:

```
ModalDialogManager.Instance.ShowDialog();
```

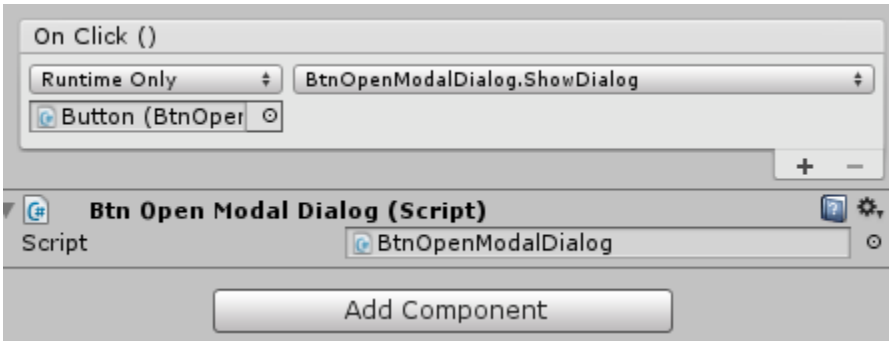
You can call it for example when clicking a button like we did it in our demo Scene *UIModalDemo*.

Therefore add a button to the Canvas:

Attach a C#-script with a method in which you call the code. In the package you can find a script in the folder UIModalDialog->Scenes->Scripts (BtnOpenModalDialog.cs):



Select the button in the scene and connect the method *ShowDialog* of the script with the *OnClick* event of the button:



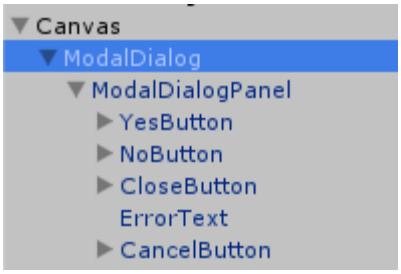
This is the method that is called then:

```
public void ShowDialog()
{
    ModalDialogManager.Instance.ShowDialog();
}
```

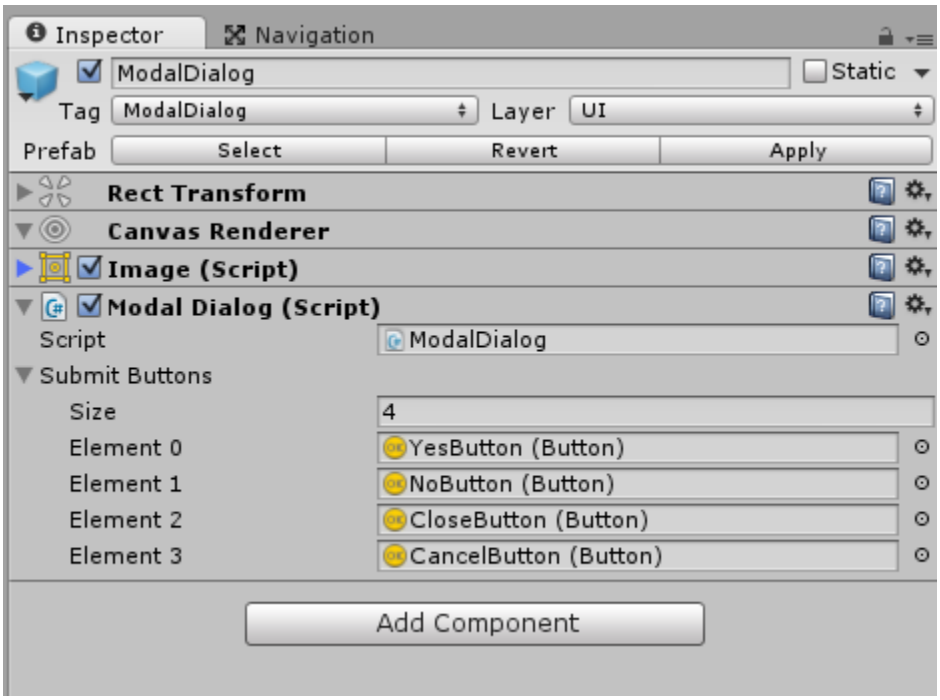
Of course you can add any other UI assets to the panel of the modal dialog like Images, Text or whatever you like to display.

1.4 WHAT CLOSES THE DIALOG?

The modal dialog is closed when one of the buttons contained in the modal dialog is pressed. But you can drag any UI button to the dialog and use it to close the dialog. You just have to add it to the collection of submit buttons (an array) of the *ModalDialog* gameobject. Select the object in the hierarchy:



And in the inspector you can see the assigned submit buttons which will close the dialog:

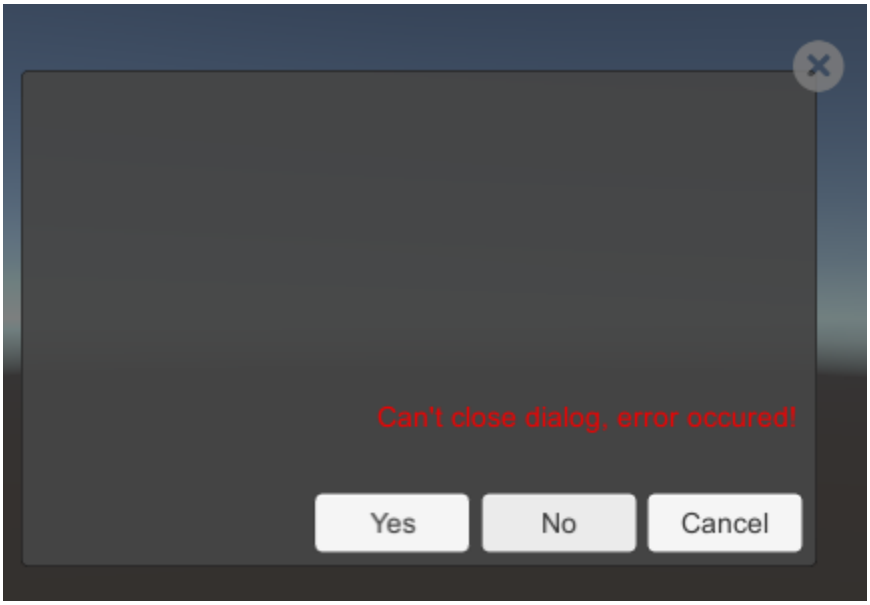


1.5 PREVENT CLOSING

There will be situations for which you want to prevent the dialog from being closed after a submit button is pressed. You can do this in a C#-script by binding the event *DialogClosing* of the *ModalDialogManager*. Here is an example used in the *BtnOpenModalDialog.cs* script:

```
using UnityEngine;
using System.Collections;
using UnityEngine.UI;

public class BtnOpenModalDialog : MonoBehaviour
{

    // Use this for initialization
    void Awake()
    {
        ModalDialogManager.Instance.DialogClosing += Instance_DialogClosing;
    }


    void Instance_DialogClosing(object sender,
Assets.UIModalDialog.Scripts.DialogClosingEventArgs e)
    {
        if (e.CloseButton.name == "NoButton")
        {
            // Access components of the dialog before closing
            Text errorText = e.DialogPanel.FindChild("ErrorText").GetComponent<Text>();
            errorText.text = "Can't close dialog, error occured!";
            e.Cancel = true;
        }
    }

    public void ShowDialog()
    {
        ModalDialogManager.Instance.ShowDialog();
    }
}
```

In this example the modal dialog contains a *UI Text* component to display an error when the *No*-button is pressed:



This is just a simple example but you can do any kind of validation in the *DialogClosing* eventhandler.

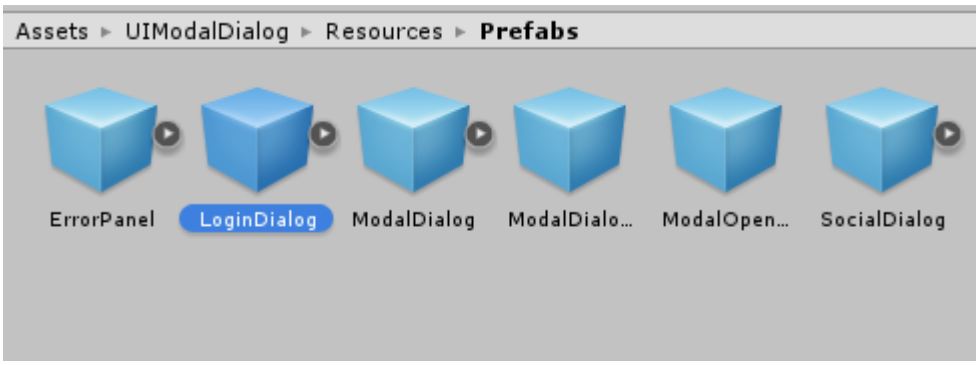
1.6 Close Notification

If you just want to be notified when a modal dialog has been closed you can bind the event *DialogClosed* of the *ModalDialogManager* like this:

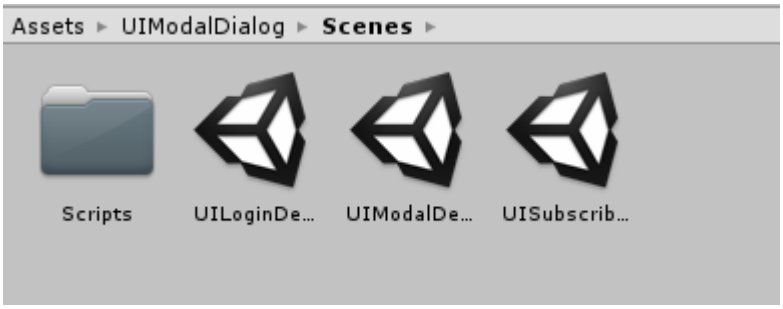
```
using UnityEngine;
using System.Collections;
using UnityEngine.UI;

public class BtnOpenModalDialog : MonoBehaviour
{

    // Use this for initialization
    void Awake()
    {
        ModalDialogManager.Instance.DialogClosed += Instance_DialogClosed;
    }

    void Instance_DialogClosed(object sender,
Assets.UIModalDialog.Scripts.DialogClosedEventArgs e)
    {
        Debug.Log("Dialog " + e.DialogName + " is closed by button " +
e.CloseButton.name);
    }

    public void ShowDialog()
    {
        ModalDialogManager.Instance.ShowDialog();
    }
}
```

2 A MODAL LOGIN DIALOG

An example of a dialog in which the principles above are implemented is a Login dialog, which is also part of the package. You can find it in the folder UIModalDialog -> Resources -> Prefabs.

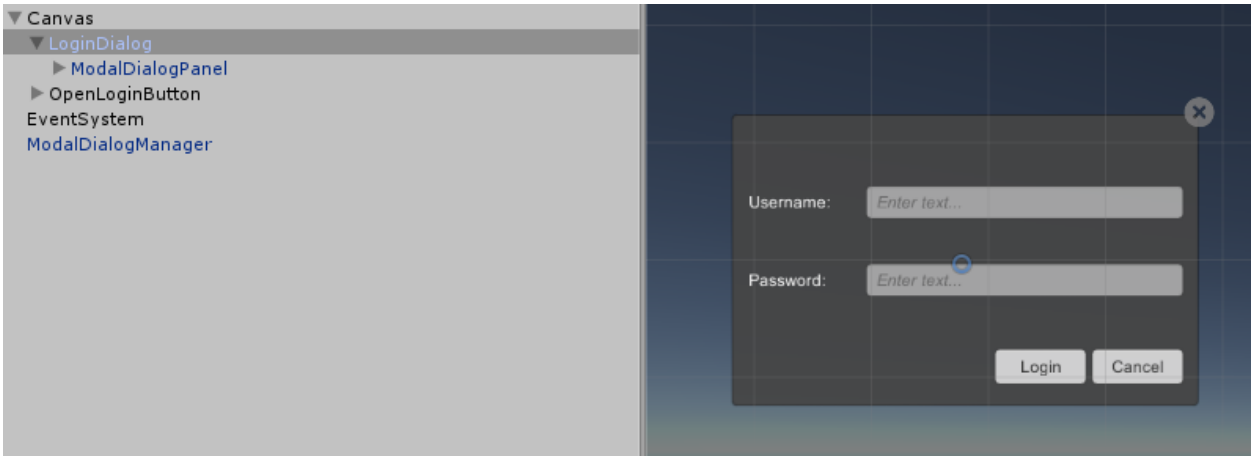


A sample scene in which it is used is the scene *UILoginDemo*:



2.1 ADD A LOGIN DIALOG

To use the *LoginDialog* gameobject you can do the same as described in the first chapter but after adding the Canvas you just drag and drop the LoginDialog.prefab to the Canvas in your hierarchy window:



2.2 USE THE LOGIN DIALOG

Opening the modal dialog is done in the same way as described in chapter 1, but for the Login Dialog we added a *LoginCallback*-object. This is used to get access to the error-fields when the dialog is about to be closed. Closing is prevented for example because of wrong user input:

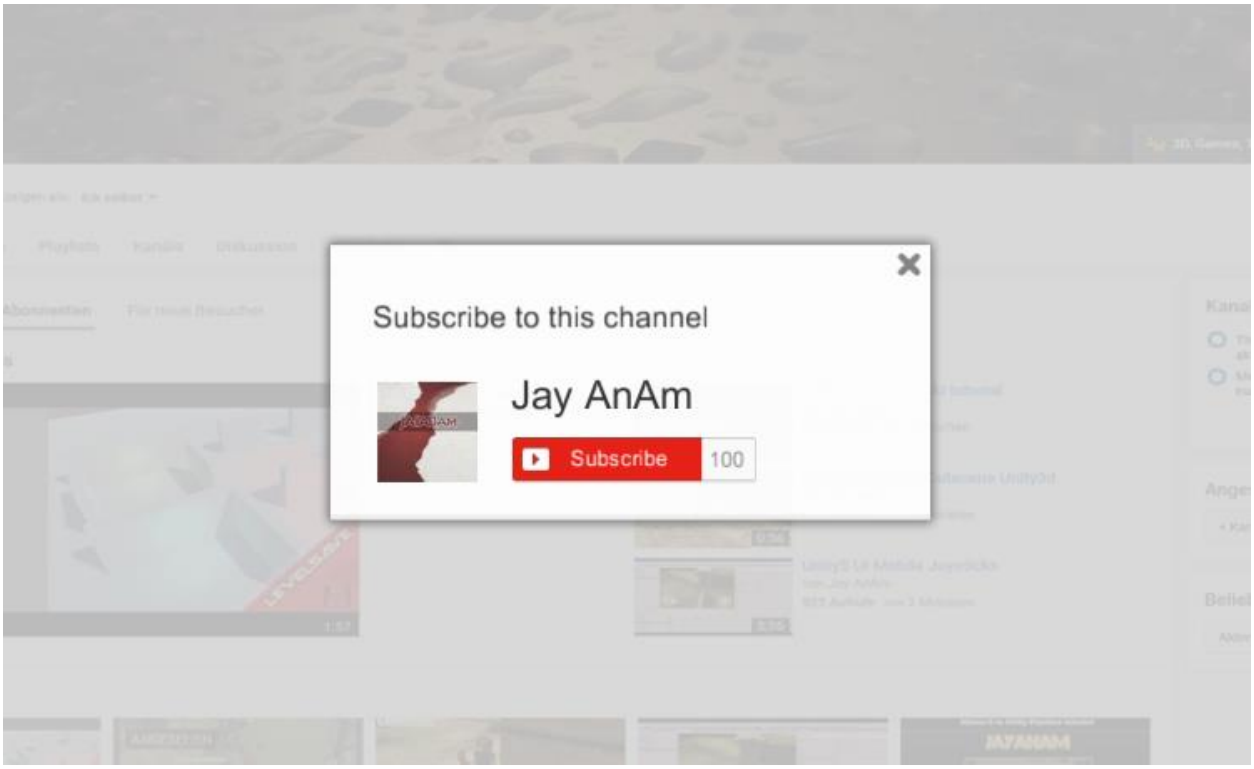
```
using System;
using UnityEngine;
using System.Collections;
using UnityEngine.UI;

public class BtnOpenLoginDialog : MonoBehaviour {

    // Use this for initialization
    void Awake()
    {
        ModalDialogManager.Instance.DialogOpened += Instance_DialogOpened;

        ModalDialogManager.Instance.DialogClosing += Instance_DialogClosing;
    }

    void Instance_DialogOpened(object sender,
Assets.UIModalDialog.Scripts.DialogEventArgs e)
    {
        LoginCallback loginCallBack = e.DialogPanel.GetComponent<LoginCallback>();
        loginCallBack.ClearInput();
    }


    void Instance_DialogClosing(object sender,
Assets.UIModalDialog.Scripts.DialogClosingEventArgs e)
    {
        if (e.CloseButton.name == "LoginButton")
        {
            // Access Login callback scripton Login
            LoginCallback loginCallback = e.DialogPanel.GetComponent<LoginCallback>();
            loginCallback.ClearErrors();

            if(String.IsNullOrEmpty(loginCallback.GetUsername()))
            {
                loginCallback.SetErrorUsername("Please enter a username");
            }

            if (String.IsNullOrEmpty(loginCallback.GetPassword()))
            {
                loginCallback.SetErrorPassword("Please enter a password");
            }

            e.Cancel = loginCallback.HasError;
        }
    }

    public void ShowDialog()
    {
        ModalDialogManager.Instance.ShowDialog();
    }
}
```

As you can see when the dialog will be opened, the input fields for entering the username and password are cleared. When hitting the Login-button, the input fields are checked. If these are empty an error is displayed and the dialog won't be closed.

Of course you can add any kind of validation logic here.

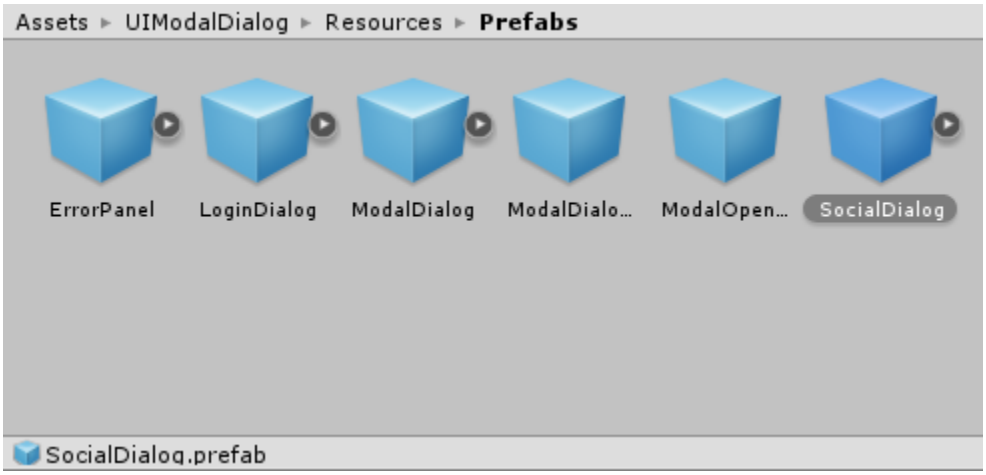
3 A SOCIAL MEDIA DIALOG

The modal dialog system is very extendable. In the scene *UISubscribeDemo* we changed the modal dialog to a style you might know from youtube-channels. You start the scene and after some time an overlay for subscribing to the channel appears:

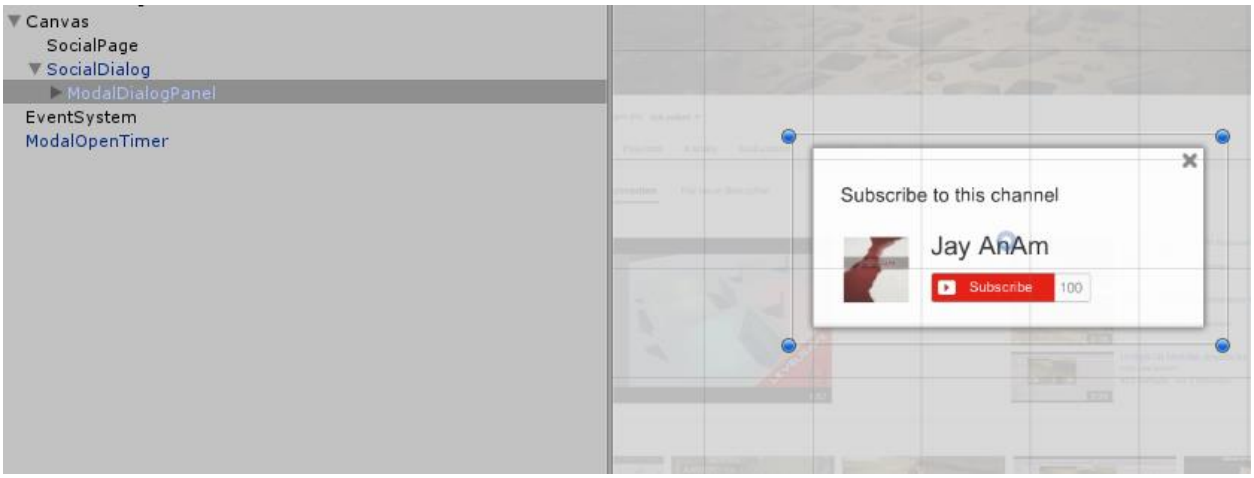


3.1 ADDING A SOCIAL MEDIA OVERLAY

The overlay can be found in the prefabs folder:



Drag and drop it to your canvas just like in the examples above:



As background an image of our youtube channel is used but you can use any other image.

To let the overlay appear after a certain time (e.g. 2 seconds after starting the scene) we added a script called *OpenSocialTimer.cs*: