Demo2-切换流程和场景

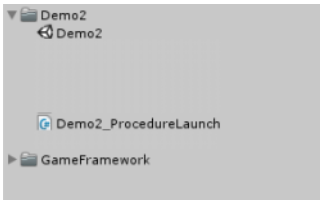
2018年3月5日 7:17

这个Demo是为了和大家展示Game Framework切换场景的方式，但是切换场景又涉及到切换流程，所以就一起介绍了。

1.创建初始场景（Demo2）

这个Demo需要三个场景：初始场景（永久存在，不卸载）、菜单场景、游戏场景。

我们先来创建初始场景，创建一个项目，新建一个场景，命名为Demo2，然后创建一个Demo2_ProcedureLaunch.cs脚本：

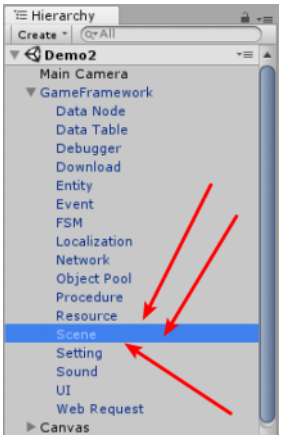


脚本内如如下：

```
using System.Collections;
using System.Collections.Generic;
using GameFramework;
using GameFramework.Procedure;
using UnityEngine;
using UnityGameFramework.Runtime;
using ProcedureOwner = GameFramework.Fsm.IFsm<GameFramework.Procedure.IProcedureManager>;
public class Demo2_ProcedureLaunch : ProcedureBase {
    protected override void OnEnter(ProcedureOwner procedureOwner)
    {
        base.OnEnter(procedureOwner);

        Log.Debug("初始！！")

        SceneComponent scene
            = UnityGameFramework.Runtime.GameEntry.GetComponent<SceneComponent>();

        // 切换场景
        scene.LoadScene("Demo2_Menu", this);

        // 切换流程
         ChangeState<Demo2_ProcedureMenu>(procedureOwner);
    }
}
```

这是一个很简单的流程类，只要继承ProcedureBase，并且重写OnEnter函数即可。

SceneComponent是专门用于处理场景逻辑的框架组件，其实就是我们一直看到的这个东西：



而LoadScene就是用于切换场景的函数（Demo2_Menu是另外一个场景，等会再说），ChangeState函数用于切换流程。
按照官方的示例项目，应该是先在OnEnter切换场景，然后在Update里切换流程，而且有一个专门的流程用于处理这些切换。
本Demo仅作演示功能，不代表最佳实践，请不要随意学习。

我们在进入初始场景时，立刻就切换到菜单场景（旁白：那为什么不直接把菜单场景设为初始场景？）

对，为什么不把菜单场景设为初始场景呢？因为初始场景是用来存放框架基础组件的，所以这个场景是不会被卸载的。在切换场景的时候，实际上只是加载了新场景，初始场景并不会消失（否则那些基础组件不就没了）。

2.创建菜单流程

我们的菜单场景只需要一个功能：点击菜单里的一个按钮后，进入到游戏场景。

新建一个场景，命名为Demo2_Menu，然后创建一个Demo2_ProcedureMenu.cs脚本.
脚本的内容如下：

```
using System.Collections;
using System.Collections.Generic;
using GameFramework;
using GameFramework.Procedure;
using UnityEngine;
using ProcedureOwner = GameFramework.Fsm.IFsm<GameFramework.Procedure.IProcedureManager>;
public class Demo2_ProcedureMenu : ProcedureBase {
    protected override void OnEnter(ProcedureOwner procedureOwner)
    {
        base.OnEnter(procedureOwner);
        Log.Debug("进入菜单流程，可以在这里加载菜单UI。");
    }
}
```

这是一个很简单的流程类，只要继承ProcedureBase，并且重写OnEnter函数即可。
因为只是Demo，我们只在进入这个流程时打印一条日志。

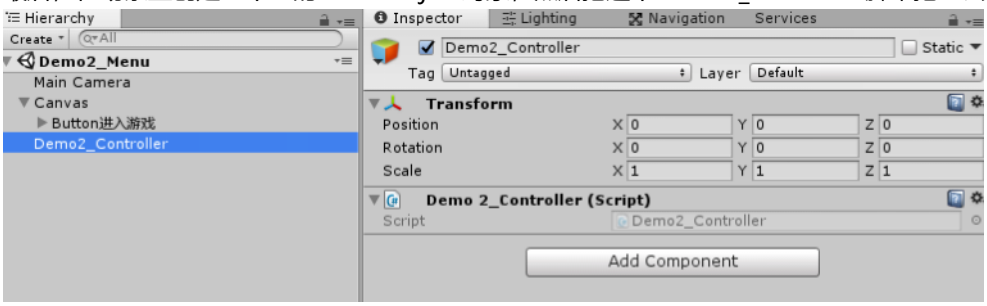
3.创建测试按钮

因为这个Demo仅展示切换流程和场景，所以不进行加载UI的演示，我们直接用Unity本身的方式创建一个按钮（希望大家不要问我是怎么创建的）：

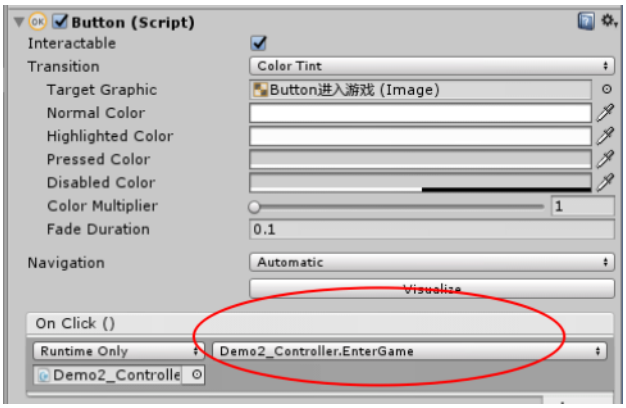
然后再创建一个脚本，用来处理按钮事件，我们命名为Demo2_Controller.cs吧：

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityGameFramework.Runtime;
public class Demo2_Controller : MonoBehaviour {
    public void EnterGame() {
        SceneComponent Scene
            = UnityGameFramework.Runtime.GameEntry.GetComponent<SceneComponent>();
        Scene.LoadScene("Demo2_Game", this);
    }
}
```

最后，在场景里创建一个空的GameObject对象，然后把这个Demo2_Controller脚本拖上去：



啊，不对，还有一个最后，那就是给按钮添加一个点击事件：



这个点击事件自然是我们的EnterGame函数。

4.创建游戏流程

用同样的方式，创建一个新的场景，命名为Demo2_Game，然后创建一个游戏流程脚本（Demo2_ProcedureGame.cs）：

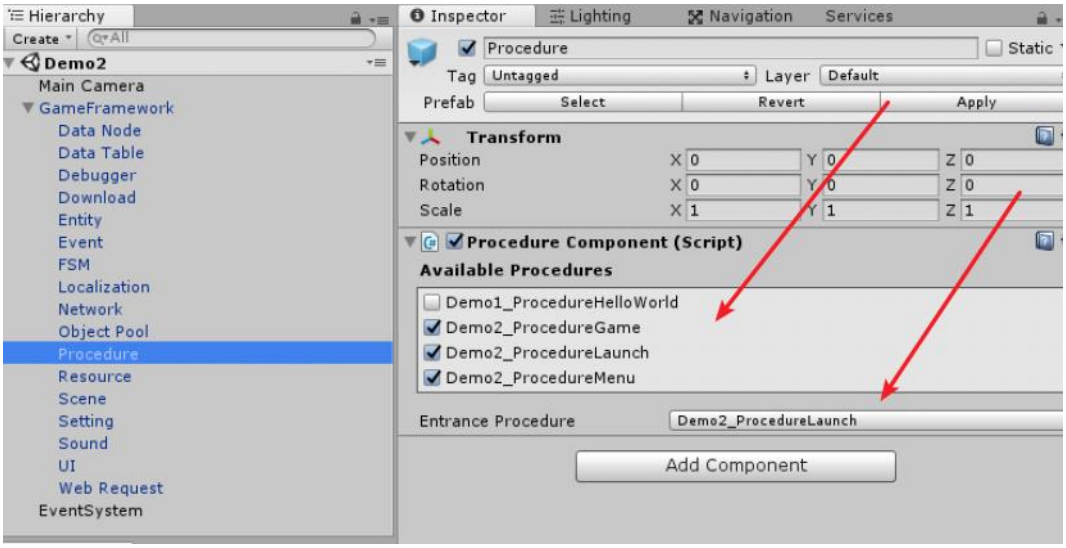
```
using System.Collections;
using System.Collections.Generic;
using GameFramework;
using GameFramework.Procedure;
using UnityEngine;
```

```
using ProcedureOwner = GameFramework.Fsm.IFsm<GameFramework.Procedure.IProcedureManager>;
public class Demo2_ProcedureGame : ProcedureBase {
    protected override void OnEnter(ProcedureOwner procedureOwner)
    {
        base.OnEnter(procedureOwner);
        Log.Debug("进入游戏流程，可以在这里处理游戏逻辑，这条日志不会打印，因为没有切换到Game流程");
    }
}
```

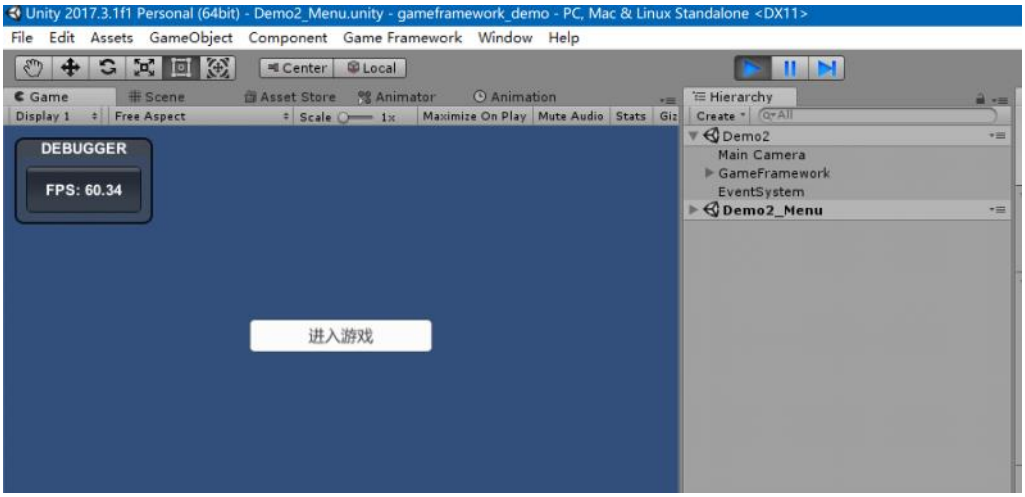
由于切换流程需要在继承了ProcedureBase的类里操作，而我们的Demo2_Controller类只是一个普通的MonoBehavior类，所以没法在切换场景的时候同时切换流程。
只好作罢，我们下一个Demo再一起看看怎么用框架的方式加载UI，那样就能进行比较完整的流程和场景切换演示了。本Demo不想太复杂，点到即止。

5.测试一下

为了验证创建的流程是否正常，我们按照Demo1的方式，把Demo2_ProcedureLaunch流程设为初始流程：



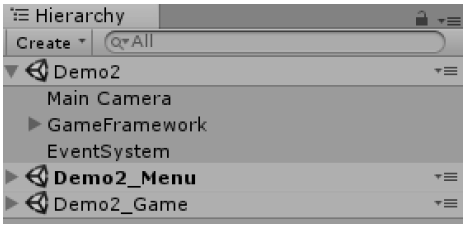
OK，运行游戏（Demo2场景），正常情况下是这个效果：



留意右边的Hierarchy视图，Demo2场景和Demo2_Menu场景是同时存在的。
证明之前所说的初始化场景Demo2是不会被卸载的。

6.卸载场景

现在，点击进入游戏，我们将切换到Demo2_Game场景，但是，请看下图：



Demo2_Menu场景和Demo2_Game场景竟然同时存在，这是因为框架只加载场景，而不是单纯地切换场景，这不是我们想要的。虽然我暂时不知道框架有没有直接切换场景的功能，但是，按照官方的示例项目，在切换场景前，需要先卸载场景。

打开我们的Demo2_Controller类，改动如下：

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityGameFramework.Runtime;
public class Demo2_Controller : MonoBehaviour {
    public void EnterGame() {
        // 获取框架场景组件
        SceneComponent Scene
            = UnityGameFramework.Runtime.GameEntry.GetComponent<SceneComponent>();

        // 卸载所有场景
        string[] loadedSceneAssetNames = Scene.GetLoadedSceneAssetNames();
        for (int i = 0; i < loadedSceneAssetNames.Length; i++)
        {
            Scene.UnloadScene(loadedSceneAssetNames[i]);
        }

        // 加载游戏场景
        Scene.LoadScene("Demo2_Game", this);
    }
}
```

在加载新的场景之前，我们先读取所有已存在的场景，然后主动卸载掉（初始场景会被排除），这样就不会同时出现多个场景了。

7.结束

OK，本Demo到此结束，由于不想涉及太多内容，所以在某些操作上是不太符合框架的思路的，仅供参考。