UI框架设计原则：

对场景中所有显示的或者需要显示的界面进行统一的管理

（管理者模式 Manager类 单例 提供打开界面的方法 存储所有已打开的界面）

所有可以被单独打开和关闭的界面 做成prefab(存储在Resources文件下)

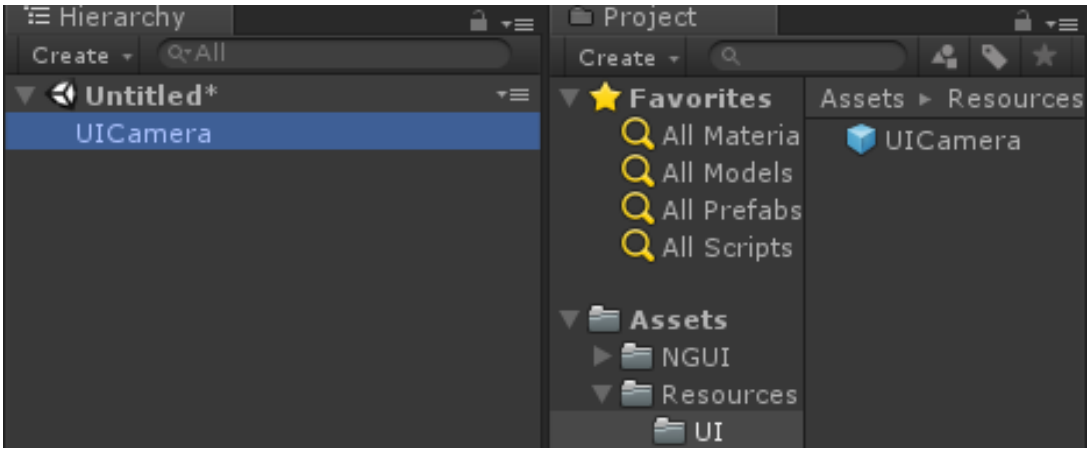
加载和销毁需要考虑减少UI渲染压力 提高渲染效率和对内存的优化

（减少实例化次数和销毁次数 合理分配图集）

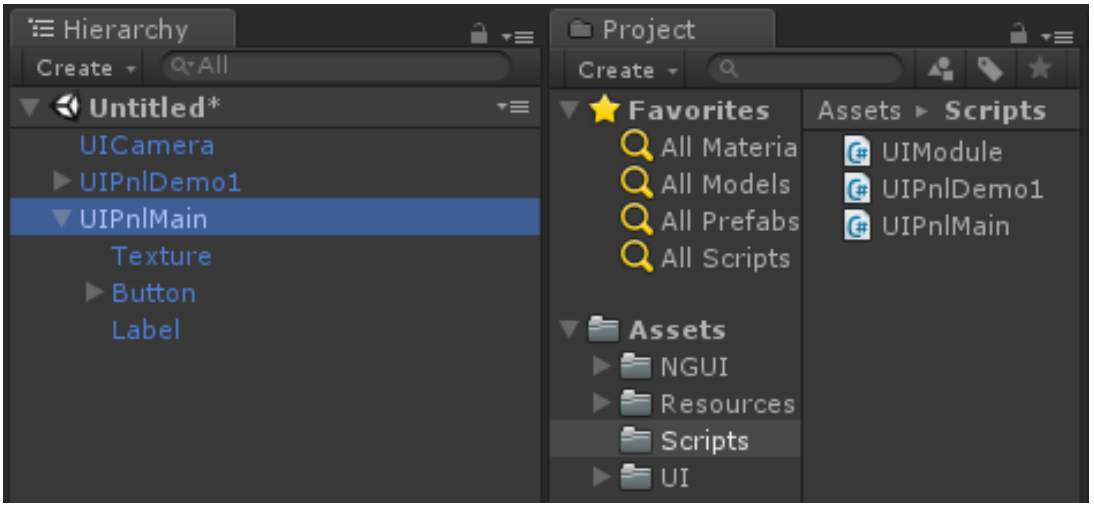
设计时 考虑不同类型界面的显示顺序问题以及不同界面之间的关联问题

（界面 对话框 跑马灯）

把UI摄像机制作成预设 保证所有场景使用UICamera就可以渲染出UI界面



在当前场景中创建两个UI界面 绑定脚本



UI父类 UIModule

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

//UI模块类  不管是界面 对话框 还是静态界面 都需要继承于此类
public class UIModule : MonoBehaviour {

    public bool DontHide;//不会被隐藏
    public bool dialog;//是否是对话框

    private GameObject uiPrefab;//UI界面的预制体

    //OnShow 在打开UI的时候调用
    public virtual bool OnShow(params object[] userdatas)
    {
        return true;
```

```
    }

    //OnHide 在隐藏UI时调用 不直接Destory 不显示使用设置非激活
    public virtual void OnHide(bool hide)
    {

    }

}
```

UI管理类 SysUIManager

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

//UI界面枚举 一般一个界面对应一个枚举
//同时保证 枚举 预制体 脚本名 统一
public enum UIType
{
    UIPnlMain,
    UIPnlDemo1,
    UIDlgDemo2
};

//UI模块数据类 做数据绑定 比如把枚举和对应的Prefab进行关联
public class UIModuleData
{
    //静态数据
    public string prefabName;//预设名字
    public UIType type;//UI对应的枚举类型
    public bool hideOtherModule;//当UI打开时 是否需要隐藏其它的界面

    //动态数据
    public GameObject uiObj;//UI预设实例化后生产的克隆对象(clone)
    public UIModule uiClass;//UI物体身上绑定的继承于UIModule的类的对象

    public UIModuleData(string name,UIType type,bool hide)
    {
```

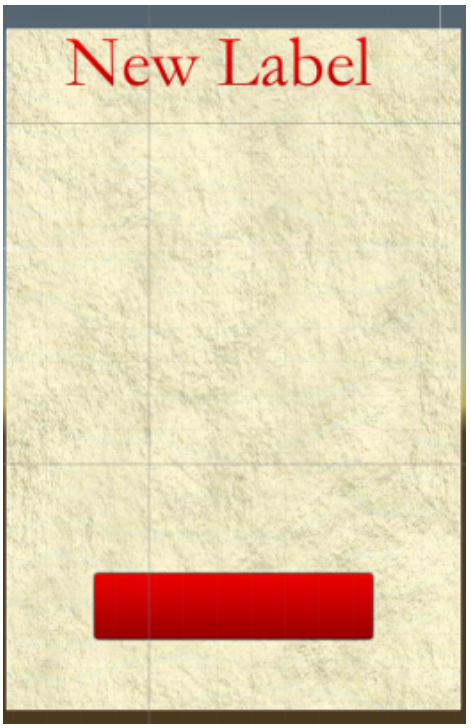
```
        this.prefabName = name;
        this.type = type;
        this.hideOtherModule = hide;
    }
}



//UI管理类
public class SysUIManager : MonoBehaviour {

    private static SysUIManager instance;
    public static SysUIManager Instance
    {
        get { return instance; }
    }

    //UI界面信息类型的数据 存储所有需要在项目中使用的UI界面信息
    private UIModuleData[] uiModules;

    //当前场景中所有已经被打开过的UI界面 集合
    private Dictionary<UIType, UIModuleData> shownUIModules = new
Dictionary<UIType, UIModuleData>();



    void Awake()
    {
        DontDestroyOnLoad(gameObject);
        instance = this;
        OnInit();
    }

    void OnInit()
    {
        //完成所有界面信息的初始化
        uiModules = new UIModuleData[]
        {
            //预设名字 预设的枚举类型 是否隐藏其它界面
            new UIModuleData("UIPnlMain",UIType.UIPnlMain,true),
            new UIModuleData("UIPnlDemo1",UIType.UIPnlDemo1,true),
            new UIModuleData("UIDlgDemo2",UIType.UIDlgDemo2,false),
        };
```

```
69
70          ShowUIModule(UIType.UIPnlMain, "大家好");
71      }
72
73      //显示UI
74      public void ShowUIModule(UIType type,params object[] datas)
75      {
76          UIModuleData data;
77          //需要打开的界面 打开过
78          if (shownUIModules.TryGetValue(type, out data))
79          {
80              //如果需要隐藏其它界面 就隐藏
81              if (data.hideOtherModule)
82                  HideAllModules();
83
84              //让UI从隐藏状态 切换回显示状态
85              HideUIModule(type, false);
86              data.uiClass.OnShow(datas);
87          }
88          else
89          {
90              //打开一个新的界面
91              data = GetUIModuleDataByType(type);
92              if (data == null)
93              {
94                  Debug.LogError("GameObject---Missing--");
95                  return;
96              }
97
98              //如果需要隐藏其它界面 就隐藏
99              if (data.hideOtherModule)
100                 HideAllModules();
101
102             //设置UI数据 显示 调用回调 返回UI物体 并赋值给UIModuleData
103             data.uiObj=SetUIInfo(data, datas);
104         }
105
106         //设置Panel的Depth 调整 界面和对话框的层级关系
107         UIPanel panel = data.uiObj.GetComponent<UIPanel>();
108
109         if (data.uiClass.dialog)
110             panel.depth = 100;
```

```
111
112     }
113
114     //隐藏UI
115     public void HideUIModule(UIType type,bool hide)
116     {
117         UIModuleData data = shownUIModules[type];
118
119         if (hide)
120         {
121             //调用隐藏的回调函数
122             data.uiClass.OnHide(hide);
123             data.uiObj.SetActive(!hide);//setActive（false）
124         }
125         else
126         {
127             data.uiObj.SetActive(!hide);//setActive（true）
128             //调用隐藏的回调函数
129             data.uiClass.OnHide(hide);
130         }
131     }
132
133     //根据界面枚举值 得到对应的界面信息对象
134     private UIModuleData GetUIModuleDataByType(UIType type)
135     {
136         for (int i = 0; i < uiModules.Length; i++)
137             if (uiModules[i].type == type)
138                 return uiModules[i];
139         return null;
140     }
141
142     private GameObject SetUIInfo(UIModuleData data,params object[]
    userdatas)
143     {
144         //实例化UI界面
145         GameObject uiObj=GameObject.Instantiate(Resources.Load("UI/" +
    data.prefabName)) as GameObject;
146
147         //设置UI界面在UICamera子节点 并且设置好位置和缩放
148         uiObj.transform.parent =
    GameObject.FindGameObjectWithTag("UICamera").transform;
149         uiObj.transform.localScale = Vector3.zero;
```

```
150         uiObj.transform.localPosition = Vector3.zero;
151 
152         UIModule module = uiObj.GetComponent<UIModule>();
153         data.uiClass = module;
154 
155         //调用OnShow的回调函数
156         module.OnShow(userdatas);
157 
158         //把打开的界面存储在已打开界面字典中
159         shownUIModules.Add(data.type, data);
160 
161         return uiObj;
162     }
163 
164 
165     //隐藏所有已打开的界面
166     private void HideAllModules()
167     {
168         foreach (var it in shownUIModules)
169         {
170             //如果本身UI是不允许被隐藏的 则过滤掉
171             if (!it.Value.uiClass.DontHide)
172             {
173                 HideUIModule(it.Value.type, true);
174             }
175         }
176     }
177 }
```

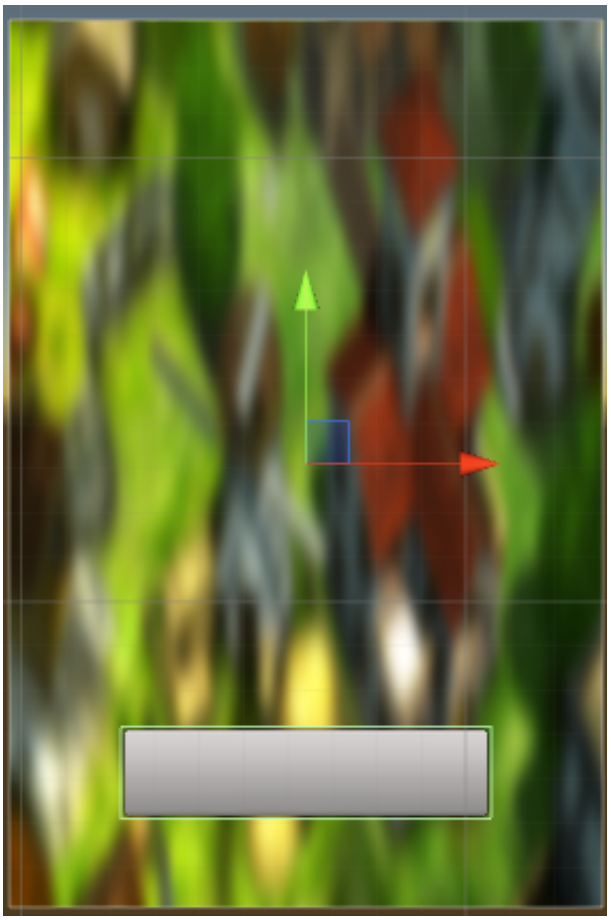
UIPnlMain




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

//UI界面子类 继承于UIModule
public class UIPnlMain : UIModule {

    public UILabel label;

    //重写OnShow回调
    public override bool OnShow(params object[] userdatas)
    {
        if(base.OnShow(userdatas)==false)
            return false;


        label.text = userdatas[0] as string;

        return true;
    }

    //重写OnHide回调
    public override void OnHide(bool hide)
    {
        base.OnHide(hide);
```

```
    }

    //界面中所带按钮的回调函数
    public void OnClickBtn()
    {
        //加载新界面
        SysUIManager.Instance.ShowUIModule(UIType.UIPnlDemo1);
    }
}
```

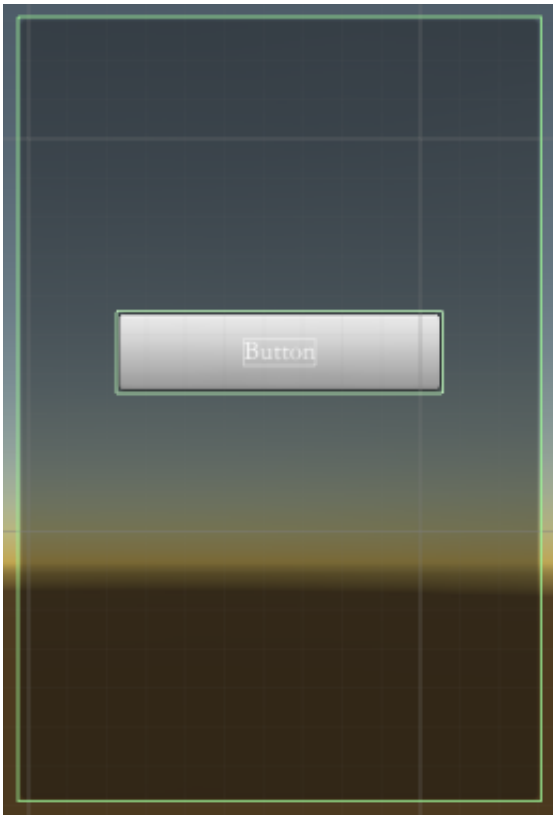
UIPnlDemo1



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

//UI界面子类 继承UIModule
public class UIPnlDemo1 : UIModule{

    public override bool OnShow(params object[] userdatas)
    {
```

```
        if(base.OnShow(userdatas)==false)
            return false;

        return true;
    }

    public override void OnHide(bool hide)
    {
        base.OnHide(hide);
    }

    public void OnClick()
    {
        SysUIManager.Instance.ShowUIModule(UIType.UIDlgDemo2);
    }
}
```

UIDlgDemo2




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

```

```
public class UIDlgDemo2 : UIModule {

    public override bool OnShow(params object[] userdatas)
    {
        if(base.OnShow(userdatas)==false)
            return false;

        return true;
    }

    public override void OnHide(bool hide)
    {
        base.OnHide(hide);
    }

    public void OnClickClose()
    {
        SysUIManager.Instance.HideUIModule(UIType.UIDlgDemo2, true);
    }

}

```