1. 播放音效.

在ClipManager里的clipNames添加音效的名字

调用方法：AudioManager.Instance.PlayClip(name); //name为对应音效名字

1. GameDefine

这个脚本保存一些常用到的游戏物体或者文件夹的路径名

1. BaseRoleController,

每个游戏角色继承这个脚本

（待更新）

1. RoleManager

可以获取到玩家的transform，调用：RoleManager.Instance.GetPlayer()

（待更新）

1. UIManager

（待更新）

6、事件中心

EventType.cs,EventCenter.cs,Callback.cs

使用：1、在EventType里加一个enum的消息类型

2、监听消息，EventCenter.AddListener<T>(EventType type,Callback<T> cb);

3、移除消息，EventCenter.RemoveListener<T>(EventType type,Callback<T> cb);

4、广播，EventCenter.Broadcast<T>(EventType t);

<T>C#的泛型编程。目前Callback的类型支持最多五个参数的泛型

使用例子：

1、（无参数传递）玩家监听虚拟摇杆的输入：

EventCenter.AddListener(EventType.OnJumpBtnClick, Jump);

AddListener的第一个参数为对应的消息类型，要在EventType里添加，第二个参数为自定义的方法，意思是，监听到了对应的消息类型就会调用这个方法，类似回调

发消息：虚拟摇杆脚本中， EventCenter.Broadcast(EventType.OnJumpBtnClick);

意思是，广播了对应的消息类型出去，

注意：AddListener要和RemoveListener对应使用。

EventCenter.RemoveListener(EventType.OnJumpBtnClick, Jump);一般在生命周期结束时调用RemoveListener。Monobehavior的生命周期结束，OnDestroy方法。具体unity的生命周期详细可百度。

2、（一个参数传递）玩家监听点击到了交互物体：

EventCenter.AddListener<GameObject>(EventType.OnClickInteractive, Interactive);

<>括号里的是泛型，即传递的参数类型，这里OnClickInteractive是传递一个GameObject类型，意思是，监听到了对应的消息类型吧就会调用Interactive方法，但这个方法需传递一个GameObject类型的参数。

传递方法也类似：EventCenter.Broadcast<GameObject>(EventType.OnClickInteractive, hitInfo.transform.gameObject);

在BroadCast后面的泛型<>括号里声明对应的参数类型，然后在参数列表的第二个参数里传递对应的实际的参数。

RemoveListener，类似

EventCenter.RemoveListener<GameObject>(EventType.OnClickInteractive, Interactive);

3、多参数传递：类比一个参数传递，目前最多支持5个参数传递。

1. BaseInteractive

每个可交互物体继承这个脚本

如：箱子：添加一个控制脚本，BoxInteractive，继承自BaseInteractive

然后重写它的InteractiveLogic方法。

1. ConfigManager

配置管理器

1. 添加配置：在项目的文件夹下有一个excel2Json的文件夹。在里面的excel文件夹可以添加一个excel表，表的第一个子表即生成的json的名称。

详细使用规则：https://blog.csdn.net/ylbs110/article/details/82755822

1. Excel表的规则：不要有多余子表，每一列的数据类型默认为string，如果想是int类型，可以，name#int这样，详细可以参考RoleConfig这一个表
2. 读取：ConfigManager有一个LoadJson<T>的泛型函数，T为对应的数据类，如RoleConfig表的数据类就是RoleData，第一个参数是一个对应数据类的字典。可以参考写法，第二个参数jsonPath，只需要填生成好的json文件名就好了
3. 实现接口，1、要声明一个对应数据类的字典如

private Dictionary<int, RoleData> roleDict = new Dictionary<int, RoleData>();

实现获取接口：

public RoleData GetRoleData(int cfgId)

{

if (!this.roleDict.ContainsKey(cfgId))

{

return null;

}

return this.roleDict[cfgId];

}

然后在构造方法里调用LoadJson<T>(dict,path);

数据类的写法，必须保证继承IcfgId这一个接口，这是配置名属性，主要用来获取对应的配置。

1. 调用ConfigManager.Instance.GetXXXData(xxx);
2. GameManager

单例，控制游戏的过程，如StartGame开始游戏，GameOver,游戏结束，RestartGame

调用： GameManager.Instance.GameWin();