使用手册

一、Util工具

- ***概率函数 MathUtil***

```
***概率函数 MathUtil
//从若干值中随机取出一个值
var randomAge = MathUtil.GetRandomValueFrom(new int[] { 1, 2, 3 });
//输入百分比返回命中概率
MathUtil.Percent(50);
```

- ***单例 MonoSingleton 和 Singleton***

```
  //所要单例的Mono类继承 MonoSingleton<所要单例的Mono类>就可以使用
MonoSingletonExample.Instance;
   public class MonoSingletonExample : MonoSingleton<MonoSingletonExample>{}
  //Singleton也是一样，所要单例的Mono类继承 Singleton<所要单例的Mono类>
    public class SingletonExample : Singleton<SingletonExample>{}
```

- ***MsgDispatcher消息机制***

```
     private static void MenuClicked()
    {
    MsgDispatcher.UnRegisterAll("消息1");//移除消息（包括key）
     MsgDispatcher.Register("消息1", OnMsgReceived);//注册消息
    MsgDispatcher.Send("消息1", "1");//发送消息
    MsgDispatcher.UnRegister("消息1", OnMsgReceived);//移除指定消息 消息key还在
    }
    static void OnMsgReceived(object data)
    {
     ebug.LogFormat("消息{0}", data);
    }
```

- ***ResolutionCheck分辨率检查***

```
//判断当前设备是否是Pad
ResolutionCheck.IsPadResolution();
//判断当前设备是否是Phone
ResolutionCheck.IsPhoneResolution();
//判断当前设备是否是4s
ResolutionCheck.IsPhone15Resolution()
//判断当前设备是否是iphoneX
ResolutionCheck.IsiPhoneXResolution()
```

- ***SimpleObjectPool简单对象池使用***

```
 class Fish { }
//实例出100的对象 参数一是在实例的时候执行的函数FuncFunc<T>,参数二是需要重置实例的函数Action<T>参数三是实例的个数
var fishPool = new SimpleObjectPool<Fish>(() => new Fish(), null, 100);
//取出一个一个对象实例
var fishOne = fishPool.Allocate();
//对象实例回收
fishPool.Recycle(fishOne);
```

- ***TransformExtension Transform的扩展***

```
GameObject gameObject = new GameObject();
gameObject.transform.SetLocalPosX(5.0f);//直接设置X值
gameObject.transform.SetLocalPosY(5.0f);//直接设置Y值
gameObject.transform.SetLocalPosZ(5.0f);//直接设置Z值
gameObject.transform.SetLocalPosXY(5.0f,5.0f);//直接设置XY值
gameObject.transform.Identity();//重置transform
```

二、Manager框架结构

- ***MonoBehaviourSimplify 对MonoBehaviour的扩展***

```
//Delay定时功能，定时执行方法内部使用了协程
  Delay(5.0f, () =>
            {
                Debug.Log(" 5 s 之后");
                this.Hide();
            });
//继承MonoBehaviourSimplify后可直接使用消息机制
 private void Awake()
 {
   RegisterMsg("Do", DoSomething);//注册消息
   RegisterMsg("OK", data =>
   {
     Debug.Log(data);
     UnRegisterMsg("OK");//移除消息
   });
   SendMsg("Do", "hello");
   SendMsg("OK", "hello1");//发送消息
}
void DoSomething(object data)
{
   // do something
   Debug.LogFormat("Received Do msg:{0}", data);
}
```

- ***AudioManager 音效管理***

```
//播放音效 Resources内音效
AudioManager.Instance.PlaySound("coin");
//所有音效暂停并静音
```

```
AudioManager.Instance.SoundOff();
//所有音效停止暂停并且关闭静音
AudioManager.Instance.SoundOn();

//循环播放音乐 一般指背景音乐 只有一个实例
AudioManager.Instance.PlayMusic("home", true))
//背景音乐暂停
AudioManager.Instance.PauseMusic();
//背景音乐继续播放
AudioManager.Instance.ResumeMusic();
//背景音乐停止播放
AudioManager.Instance.StopMusic();
//背景音乐暂停并且静音
AudioManager.Instance.MusicOff();
//背景音乐停止暂停并且关闭静音
AudioManager.Instance.MusicOn();
```

- ***GUIManager UI界面管理***

```
//设置分辨率 权重为0以宽度进行适配
GUIManager.SetResolution(1280, 720, 0);
//Resources内加载界面到哪个层级
GUIManager.LoadPanel("HomePanel",UILayer.Common);
//移除界面
GUIManager.UnLoadPanel("HomePanel")
```

- ***LevelManager 关卡管理***

```
//初始化 添加两个场景的名字
LevelManager.Init(new List<string>
            {
                "Home",
                "Level",
            });
//加载当前Index的场景
LevelManager.LoadCurrent();
//设置当前场景的Index
LevelManager.Index = 1;
//加载下一个场景
LevelManager.LoadNext
```

- ***MainManager 主入口管理继承了MonoBehaviour***

```
//分成三个模式
protected override void LaunchInDevelopingMode(){}//开发模式
protected override void LaunchInTestMode(){}//测试模式
protected override void LaunchInProductionMode(){}//发布完成模式
```