1、做一个公共二次确认界面

1）定义一个内部参数类 ParamClass，用于接收调用方传递的参数和显示样式

2）参数类有多个构造函数，可以适应多种定义

3）将确认和取消事件以委托的方式传入

例：ConfirmFormController.cs

4）扩展

A、可以定义枚举，然后根据枚举显示不同的二次确认界面的样式

5）好处：

定义公共二次确认界面，方便所有界面调用该界面，若需要调用该界面

2、Class类

2.1类的定义

1、若一个类功能很多，则要使用partial方法，将类根据不同的功能进行分开，放到不同的文件中

2、类的继承有两种：基础父类和接口

3、要考虑清楚哪些继承基础父类，哪些时候要继承接口

4、类需不需要静态变量和静态方法

5、需不需要扩展类

6、初始化哪些变量

7、销毁哪些变量

8、需不要定义管理类（工厂模式）

3、时钟

3.1 定义一个实际用来实现时钟任务的类：TimeTask

3.2关闭这些时钟任务：ShutDown，具体的关闭处理也在Time Task中来实现

3.3 定义一个循环遍历函数：Update()就是通过这个函数来增加时钟走时，然后判断是否到了执行任务的时间点。当某个任务时钟到了就在这个函数中删除掉这个任务。

3.4 定义增加时钟任务的函数：AddTask()，若有相同的任务，则将以前的删除，用新的

3.5 定义增加时钟任务带参数的AddTask(flaot timeDelay, bool repeat, Action<T> callback, T param)

3.6 定义删除时钟任务 RemoveTask(Action callBack);

3.7 定义销毁所有时钟任务的函数：Destroy()

TimeTask

3.8 延迟时间，是否重复，当前时间，回调函数

3.9 创建：Create(timeDelay, repeat,callBack);从池子中获取

3.10 释放：Release（TimeTask T）:放到池子中

3.11 初始化：Init（）

3.12 实际判断是否到执行时间的函数：Tick

3.13 实际执行任务的函数：RunTask()

3.14 清理函数：Clear()

4、abd

5、GitHub

6、对单个功能类的处理

6.1 该类主要用于实现某一个具体的功能，比如生成一个小人动画，生成一个特效显示

6.2 加一个参数，用于向类中传入具体想要实现的类功能的样子

6.3 写一个辅助类，该类有静态方法，可以加池子处理功能类，其他类通过这个辅助类对功能类进行使用

7、游戏模块分布

8、邮件头显示处理

8.1SLGMailReportFormController中会确定此次显示的邮件头是那个类型邮件的（战斗，采集，系统通知等等）

8.2 然后在创建邮件头里面加一个创建参数，然后调用创建子邮件头类

9、行为的处理

9.1行为分级：行为组--🡪行为

9.2行为组分为：Click, Use, Cast, Equip

9.3行为：

A、IBehavior：写一个接口

B、BehaviorParam:写一个行为参数，带有这个行为是哪个实体的行为

C、Behavior Base：IBehavior :写一个行为基类，带有行为参数

D、BehaviorState：行为状态，成功，失败，正在运行

9.4行为组

A、List<IBehavior>：管理行为

B、List<ICondition> :管理条件

C、BehaviorGroupState:行为组状态，正在运行，已处理

D、Handle：处理行为函数，若一个组多个行为，则只有全部成功或者其中有一个失败就结束这个组的执行

9.5实体中处理行为组（EntityBase.Event）

A、*Dictionary*<BehaviorGroupEvent, List<BehaviorGroup>> 行为组字典

B、AddBehaviorGroup:增加行为组

C、OnBehaviorGroup:添加行为组

D、ExecuteBehaviorGroup:执行行为组 C

E、OnItemClickEvent():点击事件 D

10、PopTips冒泡提示

10.1 一个用于显示提示文字和图标的类TipItem

10.2 在该类中实现文字和图标的赋值，并且加一个渐隐渐显的效果

10.3 一个用于对该类进行创建和使用的类PopTip

10.4 在该类中加入一个协程，通过协程来实现顺序冒泡提示

10.5技术关键：功能类，管理类，协程

11、UIEntityBase：用于加载预制体的方式

11.1定义预制体的名字

11.2 定义加载预制体成功之后的回调函数

11.3 定义一个Pop函数，用来创建或者获取预制体池子，然后若池中没有定义的预制体，则就重新创建一个，并且在池子中注册，并且对该创建出来的对象进行实例化InitInstance()

11.4 实例化的工作就是加载该预制体，并且将加载出来的预制体进行初始化处理