框架说明：

架构说明：

GameFramework说明：

GameFramework代码UML说明：

代码规范说明：

**一．架构说明：**

GameFramework三层架构：

**1.底层：**

**说明**：各个核心功能的实现，作为游戏的基础服务Service类；比如Entity，表现为

EntityManager相关的实现

具体见GameFramework说明：https://github.com/EllanJiang/GameFramework

**补充说明**：

**Entity**：特效，角色，坐骑，武器等都可以作为Entity，他们按Group分类；

也可以相互绑定，比如角色骑上坐骑；Entity在底层托管给EntityManager管理，而不要自己实现类似逻辑和Manager。（讨论）

**GameFrameworkModule**: 每个XXManager是一个Module

**GameFrameworkEntry**: 管理所有Module（也就是管理所有XXManager）

**包括创建，获取，关闭，更新；**

**游戏逻辑层**如果要实现新的Manager，也应该通过GameFrameEntry管理

比如SkillManager：

继承：GameFrameworkModule

使用：GameFrameworkEntry.GetModule<SkillManager>()

所以不需要通过**Singleton**管理

**命名空间**：GameFramework

**是一个DLL**



**2.中间层：**

说明：

**框架GameFramework .prefab**：GameFramework/Builtin下挂载各个Component，

直接在这做一些初始设置，可以细看下; 游戏启动也必须使用这个prefab；

比如Entity：EntityGroup创建Effect组，Character组，还有设置EntityHelper

（负责创建和销毁）**需要了解下这块的设置和相关类设计**

**XXComponent**：对底层的封装，上层业务层不能直接调用底层XXManager，必须通

过XXComponent中转调用，另外使用Component时，需要做类似：

GameEntry.Resource.LoadAsset的调用方式（Resource就是ResourceComponnet）

**XXHelper**：对底层使用的接口的实现类，比如DefaultLoadResourceAgentHelper完成

ResourceManager的最终资源加载实现，ResourceManager会调用这个Helper加载资源；Helper放在中间层，开放给用户修改或添加实现类

**GameEntry**：游戏入口，注册所有XXComonent，游戏中获取和使用XXComponent，

都要GameEntry.XX（比如GameEntry.Entity.ShowEffect）

**BaseComponent**：游戏最外层的管理，类似GameApplication

**ResourceComponent：**

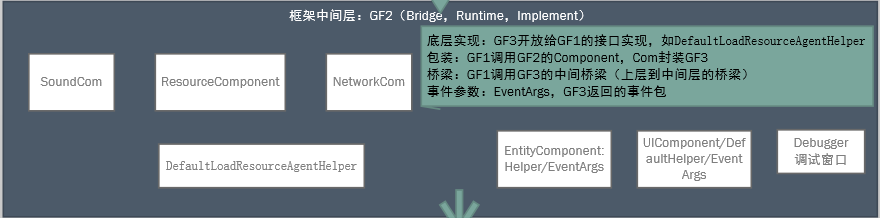
资源的加载（UML结构图和时序图见GameFrameworkUML目录）

案例：

1. ProcedurePreload.LoadFont：调用GameEntry.Resource.LoadAsset
2. DataTable：DataTableExtension.LoadDataTable，不过是在底层完成资源加载

**命名空间**：UnityGameFramework.Runtime

**目录**：Assets/Scripts/GF\_Bridge



**3.业务层：**

**GameFramework相关**：通过调用XXComponent调用底层的核心功能完成业务逻辑。

放在Assets/scripts/Logic/GF\_Logic

**其他业务逻辑**：放在Assets/Scripts/Logic里

**命名空间**：Qin

**目录**：Assets/scripts/Logic/GF\_Logic；Assets/scripts/Logic 

（放大看说明）

**二．代码规范说明：（业务层， 未定）**

类：前缀c; cSkillManager, cCharacter

枚举：前缀e；ePlayer\_Job

结构体：前缀s；sItemInfo

Partial：代码多用C#的Partial机制，尽量在一个类的管理下，避免代码之间过于零散无

关联不好管理；比如PlayerManager分为PlayerManager, PlayerManagerNet,

PlayerManagerLoad

命名空间：代码过多，需要按命名空间组织代码，注意层次

**三．目录结构**

**RawRes**:

原始资源目录，先按模块分，再按内容分，最后按类型分；

比如Character和Windows，Scene

Character包含character001，character002；

character001包含Model,Animation,Texture,Material等；

Windows包含LoginUI，RoleUI；

LoginUI包含：Texture，Material, Effect, Animation等

Scene包含SelectScene, GameScene

GameScene包含：Model，Texture，Material，Scene文件等

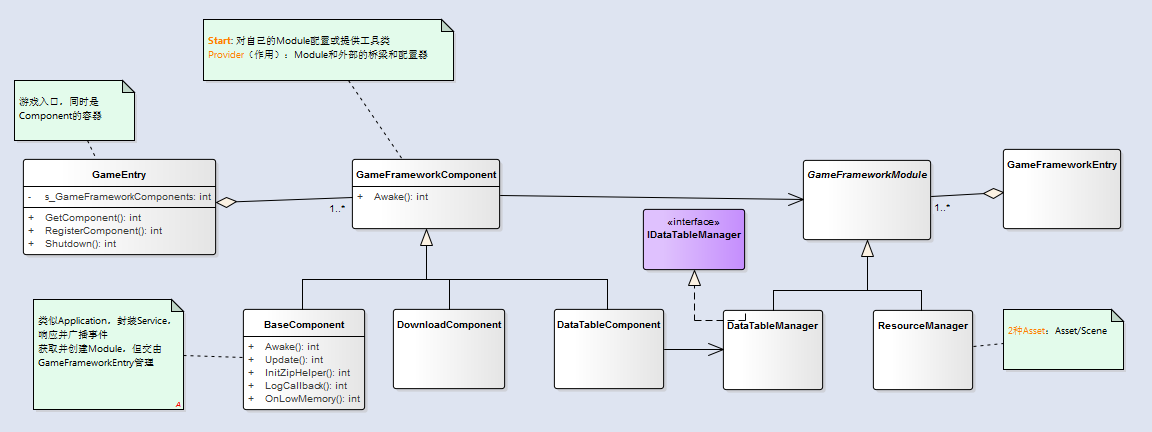
**Prefab**：Prefab文件

比如Windows包含LoginUI，RoleUI的Prefab文件都放在此

**Scripts**：逻辑代码；Common, Logic, GF\_Bridge

**四．GameFramework代码UML图：（**见doc/GameFrameworkUML目录**）**

1. **主框架：**



1. 资源加载时序图（Table表）



3.其他：见doc/GameFrameworkUML目录