# 一：资源分类

Texture：纹理

Sprites：精灵

Prefabs：预设

Shaders：着色器

Materials：材质

Mesh：网格

AnimationClip：动画片段

Animator：动画控制器

Timeline：时间轴

Skeleton：骨骼

Font：字体

AudioClip：音效

Config：配置文件

Models：模型 fbx

Scenes：场景

Scripts：代码脚本

... ... 待添加

# 二：文件夹

**Gizmos：**Gizmos 绘图相关资源放这里，Gizmos 绘制的效果不会在 Game 视图中显示，除非在 Game视图勾选 Gizmos。该目录的资源不会被打进安装包

**Plugins：**插件目录，项目需要用到的一些第三方SDK，比如广告SDK，Facebook 登录 等等，使用C/C++编写的 DLL 库

**Resources：**动态加载的资源可以放这里，这个文件夹里的所有文件，不管有没有用，都会自动打进安装包，并且Unity会执行资源压缩，该目录下的资源文件通过Resources.Load 方法加载，它是一个只读文件夹，实际项目中一般只会放一些必要且少量的资源在这个目录

**StreamingAssets：**流资源目录，这个文件夹里的所有文件，不管有没有使用，都会打进安装包，该目录下的文件打包时不会被压缩，它是一个只读文件夹，并且它在各个平台：iphone、Android、PC 等路径是不同的，可以使用 Application.streamingAssetsPath 访问该路径

**Editor：**编辑器目录，该目录下的代码调用UnityEditor 的API，存放扩展编辑器的代码，该目录中的代码不会被打进安装包内，比如打包AB工具，自己/第三方的插件，并且项目中任何层级目录下的名字为 Editor 的目录，都是一样的功能，比如一个编辑器脚本放在目录 A/XXX/Editor，另一个编辑器脚本放在 B/XXX/Editor，都是不影响编辑器功能的，所以当引入第三方SDK 的时候如Facebook，导入进来可能会在 Assets目录下有一个 Assets/Facebook 这是Facebook SDK 的目录，使用这个目录就行了，下面还可能会有 Assets/Facebook/Editor 第三方插件的都不需要管，放着就是了

**Scripts：**脚本目录，C#代码放在这里，上边的 Editor目录也可以放在 Scripts 下作为一个子目录 **Scripts\Editor**，不做强制要求

**Scenes：**场景目录，场景文件放这个目录

**RawAssets：**原始资源，比如美术贴图、模型、动画、骨骼、音效等这种未经加工的资源，而预设就不是原始资源，预设需要将模型、贴图、动画等组合包装在一起行程一个新的资源。AnimatorController 动画控制器也不是原始资源，是需要将多个动画帧组合包装在一起的资源，Timeline 等等

**GameAssets：**与RawAssets相反，**GameAssets** 目录下存放的都是经过制作的资源，如 Prefabs、AnimatorController、Timeline 等等，在编辑器模式下可以使用AssetDatabase.LoadAssetAtPah加载该目录下的资源，建议仅仅在一些自定义编辑器内使用这个方法

... ...待添加

# 三：RawAssets 文件夹内结构

RawAssets/Texture：存放贴图类图片，模型贴图、特效图片资源等

Texture/Effect/：特效贴图总目录

Texture/Effect/xxx：具体特效的贴图，也可以没有这么小的目录

Texture/Scene/：场景编辑，地图编辑用到的贴图

Texture/Scene/Common/：多个场景共同用到的放这里

Texture/Scene/Scene\_main/：主城用到的贴图

RawAssets/Sprites：存放UI 精灵相关

Sprites/UI：存放UI 精灵图片

Sprites/UI/Common：多界面通用精灵图片目录，通用按钮、一些九宫格图等

Sprites/UI/Main：主界面独立使用的精灵图片

Sprites/UI/Rank：排行榜界面独立使用的精灵图片

Sprites/XXX: 存放某些不是UI 但要将图片 TextureType 设置为 Sprite(2D and UI)

RawAssets/Shaders：存放着色器代码

根据情况决定是否需要创建子目录

RawAssets/Materials：存放材质球

Materials/Effect/：特效材质球总目录

Materials/Effect/xxx：具体材质球，也可以没有这么小的目录

Materials/xxx/ 不好确认怎么划分的材质球可以建这样的目录，一些共用，至少有两个以上引用这个材质球的可以建一个共用的目录

RawAssets/Mesh：存放网格

Mesh/xxx 存放具体功能的网格

RawAssets/AnimationClip：存放动画的目录

AnimationClip/UI：UI动画的目录

AnimationClip/UI/xxx：根据情况创建子目录

AnimationClip/Model/模型动画

AnimationClip/Model/xxx：根据情况创建子目录

RawAssets/Skeleton：存放骨骼

Skeleton/xxx 看情况创建子目录

RawAssets/Font：字体目录，一般就这一个目录就够了

RawAssets/AudioClip：存放音效

AudioClip/SoundEffect/：比较短的音效

AudioClip/Background/：较长的背景音乐

RawAssets/Config：存放配置文件

Config/BehaviorTree/ 行为树的配置文件

Config/xxx/ 具体功能的配置文件

RawAssets/Models：存放模型

Models/xxx：具体模型的目录

**RawAssets/Models特殊性**：因为模型一般拥有 Texture、Shaders、Materials、Mesh、Animation、Skeleton，这里边依赖关系比较多，所以如果一个模型，比如角色模型确实是关联了多个资源**（注意：预设跟这些资源是不一样的，所以预设还是要放在 Prefabs目录下的），**可以直接在RawAssets/Models下建立这样一个目录，还以 LvJuRen 为例

(1)

RawAssets/Models/Character/ 所有角色的总目录

RawAssets/Models/Character/LvJuRen 绿巨人的所有相关的都放这里

RawAssets/Models/Character/LvJuRen/Fbx 模型fbx文件放这里

RawAssets/Models/Character/LvJuRen/Animations 动画资源放这里

RawAssets/Models/Character/LvJuRen/Skin 贴图、材质球、Shaders、Mesh 等可以统一放这样一个目录，如果需要也可以细分目录

(2)

如果角色模型是一个系列，比如绿巨人1、绿巨人2、绿巨人3 有相互共用的资源，可以这么建立目录

RawAssets/Models/Character/LvJuRen/LvJuRen1

RawAssets/Models/Character/LvJuRen/LvJuRen2

RawAssets/Models/Character/LvJuRen/LvJuRen3

上边三个目录内可以将自己独立使用的资源建立如 (1) 的目录，而公共资源可以建立

RawAssets/Models/Character/LvJuRen/Common

RawAssets/Models/Character/LvJuRen/Common/Skin

RawAssets/Models/Character/LvJuRen/Common/Material

等等

注：把一个模型相关的多种RawAssets 类型资源放一起有什么好处？

1. 方便查找、查看，当找一个角色资源的时候，它们全部在一个目录内，它包含的 Texture、Animation、Fbx 等资源一目了然
2. 关于热更资源打包如AB包，可以将一个模型的所有资源打包到一个AB资源内，减少了AB资源的个数，方便管理

# 四：GameAssets文件夹内结构

GameAssets/Atlas：存放UI 图集

GameAssets/Prefabs：存放预设

根据不同资源建立不同的目录

Prefabs/UI：存放UI 界面预设总目录

Prefabs/UI/xxx：根据具体功能建立更小粒度的目录

Prefabs/Character：存放角色预设总目录

Prefabs/Character/LvJuRen：存放绿巨人预设的目录

如果绿巨人有不同的等级如 lv\_ju\_ren1、lv\_ju\_ren2... 将所有绿巨人的预设一同放入Prefabs/Character/LvJuRen

Prefabs/Effects/：存放特效预设

Prefabs/Scene/：存放场景地编的一些预设

GameAssets/Animator：存放动画控制器文件

GameAssets/Timeline：存放Timeline文件

# 五：命名规范

1. 资源文件名均采用驼峰式命名，每一个单词的首字母小写，中间可添加下划线，且使用能够描述其功能或意义的英文单词或词组
2. 根据功能属性和资源类型添加不同前/后缀

|  |  |  |  |
| --- | --- | --- | --- |
| 功能属性/资源类型 | 前缀 | 后缀 | 例子 |
| 角色：属性 | ch |  |  |
| 按钮图片 |  | btn |  |
| UI背景图 |  | bg |  |
| Collision：碰撞盒 |  | collision |  |
| Texture：纹理 |  | tex | ch\_lv\_ju\_ren\_model\_tex |
| Sprites：精灵 |  | spt | ui\_icon\_lv\_ju\_ren\_spt |
| Materials：材质 |  | mat | stone\_model\_mat |
| AnimationClip：动画片段 |  | ani | ch\_lv\_ju\_ren\_idle\_ani |
| Animator：动画控制器 |  | ac | ch\_lv\_ju\_ren\_ac |
| 特效 |  | fx | ui\_xxx\_panel\_open\_fx |
| Font：字体 |  | font | xxx\_font\_cn，xxx\_font\_en |
| 模型 |  | mod/fbx | ch\_lv\_ju\_ren\_fbx |
| AudioClip：音效 | sound | music | sound短音效，music长音乐 |
| Models：模型 |  | fbx | ch\_lv\_ju\_ren\_fbx |
| Scenes：场景 | scene |  | scene\_main |

没有严格意义上的前/后缀区分

(1)如果只需要使用一个标示，该标示可以放在前缀，也可以放在后缀

如：spt\_ui\_icon\_lv\_ju\_ren、ui\_icon\_lv\_ju\_ren\_spt、ui\_icon\_lvjuren\_spt

spt ：这是一个Sprite

ui\_icon ：这是一个UI 的图标

lv\_ju\_ren 、lvjuren、lvJuRen：资源名

所以从名字就能知道这是：绿巨人UI 上使用的图标

(2)如果需要同时用到功能属性和资源类型，可以将一个加在前边，一个加在后边

如：绿巨人的碰撞盒：ch\_lv\_ju\_ren\_collision

ch ：这是一个角色

collision：这是一个碰撞盒

lv\_ju\_ren：资源名

所以这是：角色绿巨人的碰撞盒

spt\_main\_panel\_close\_btn

spt ：这是一个 Sprite

btn ：这是一个按钮

main\_panel\_close：资源名，包含信息主界面关闭

所以这是：主界面关闭按钮图标

关于热更新部分资源依赖的要求，如果一个角色相关的 fbx、贴图、材质球、动画等是只被这一个资源使用，也可以简历一个类似 Common 的目录

Common/LvJuRen

Common/LvJuRen/Texture：纹理

Common/LvJuRen/Prefabs：预设

Common/LvJuRen/Shaders：着色器

Common/LvJuRen/Materials：材质

Common/LvJuRen/AnimationClip：动画片段

Common/LvJuRen/Animator：动画控制器

Common/LvJuRen/Models：模型 fbx

打 AB 包的时候将 Common/LvJuRen 目录内的所有资源打包为一个 ab资源