Unity 游戏框架搭建 2017（一）概述

为了重构手头的一款项目,翻出来当时未接触Unity时候收藏的视频 《Unity项目架构设计与开发管理》 对于我这种初学者来说全是干货。简单的总结了一下,以后慢慢提炼。

关于Unity的架构有如下几种常用的方式。

1.EmptyGO:

在 Hierarchy 上创建一个空的GameObject,然后挂上所有与 GameObject 无关的逻辑控制的脚本。使用GameObject.Find() 访问对象数据。

缺点:逻辑代码散落在各处,不适合大型项目。

2.Simple GameManager:

所有与 GameObject 无关的逻辑都放在一个单例中。

缺点:单一文件过于庞大。

3.Manager Of Managers:

将不同的功能单独管理。如下:

- MainManager: 作为入口管理器。

- EventManager: 消息管理。
- GUIManager: 图形视图管理。
- AudioManager: 音效管理。
- PoolManager: GameObject管理（减少动态开辟内存消耗,减少GC)。

实现一个简单的 PoolManager:

```
 // 存储动可服用的GameObject。
private List<GameObject> dormantObjects = new List<GameObject>();
// 在dormantObjects获取与go类型相同的GameObject,如果没有则new一个。
public GameObject Spawn(GameObject go)
{
     GameObject temp = null;
     if (dormantObjects.Count > 0)
     {
          foreach (GameObject dob in dormantObjects)
          {
               if (dob.name == go.name)
               {
                    // Find an available GameObject
                    temp = dob;
                    dormantObjects.Remove(temp);
                    return temp;
               }
          }
     }
     // Now Instantiate a new GameObject.
     temp = GameObject.Instantialte(go) as GameObject;
     temp.name = go.name;
     return temp;
}

// 将用完的GameObject放入dormantObjects中
public void Despawn(GameObject go)
{
     go.transform.parent = PoolManager.transform;
```

```
        go.SetActive(false);
        dormantObject.Add(go);
        Trim();
}

//FIFO 如果dormantObjects大于最大个数则将之前的GameObject都推出来。
public void Trim()
{
        while (dormantObjects.Count > Capacity)
        {
                GameObject dob = dormantObjects[0];
                dormantObjects.RemoveAt(0);
                Destroy(dob);
        }
}
```

缺点:

- 不能管理prefabs。
- 没有进行分类。

更好的实现方式是将一个PoolManager分成:

- 若干个 SpawnPool。
- 每个 SpawnPool 分成 PrefabPool 和 PoolManager。
    - PrefabPool 负责 Prefab的加载和卸载。
    - PoolManager 与之前的 PoolMananger 功能一样,负责 GameObject 的 Spawn、Despawn 和 Trim。

要注意的是:

- 每个 SpawnPool 是 EmeptyGO。
- 每个 PoolManager 管理两个 List (Active,Deactive)。

讲了一堆,最后告诉有一个NB的插件叫 PoolManager- -。

- LevelManager: 关卡管理。

推荐插件: MadLevelManager。

GameManager: 游戏管理。

C#程序员整理的Unity 3D笔记（十二）：Unity3D之单体模式实现GameManager

- SaveManager: 配置管理。
- 实现 Resume,功能玩到一半数据临时存储。

  推荐 SaveManager 插件。可以 Load、Save 均采用二进制(快!!!)

  所有 C# 类型都可以做 Serialize。

  数据混淆,截屏操作。

  MenuManager 菜单管理。

4.将 View 和 Model 之间增加一个媒介层。

MVCS:Strange IOC 插件。

MVVM:uFrame 插件。

5. ECS(Entity Component Based System)

Unity 是基于 ECS,比较适合 GamePlay 模块使用。

还有比较有名的 Entitas-CSharp

此篇的内容就这些。

更多内容

- QFramework 地址：https://github.com/liangxiegame/QFramework
- QQ 交流群：623597263
- **Unity 进阶小班**：
  - 主要训练内容：
    - 框架搭建训练（第一年）
    - 跟着案例学 Shader（第一年）
    - 副业的孵化（第二年、第三年）
  - 权益、授课形式等具体详情请查看《小班产品手册》：https://liangxiegame.com/master/intro
- 关注公众号：liangxiegame 获取第一时间更新通知及更多的免费内容。

-