【Unity基础教程】重点知识汇总

（七）

Unity同步与异步加载场景

同步加载（Synchronous Loading）

概念：同步加载场景意味着在加载场景时，游戏的主线程会被暂停**（阻塞主线程）**，直到场景完全加载完毕后，才会继续执行后续代码。在这个过程中，游戏的其它部分（比如动画、音频、UI等）也会被冻结**（暂停一切其他操作）。**这种方式最直接，但也可能导致游戏卡顿，尤其是当场景较大或物体较多时。

优点：

- 简单直接，代码量少。
- 场景加载完成后，所有内容都已准备好。

缺点：

- 阻塞主线程，可能导致游戏卡顿（时间停滞），影响用户体验。特别是在加载大型场景时。

同步加载（Synchronous Loading）

使用方法：

```
public class LoadSceneExample : MonoBehaviour
{
    private void Update()
    {
        // 按下1键时
        if (Input.GetKeyDown(KeyCode.Alpha1))
        {
            // 同步加载到Scene01或0号场景(场景的名称或者索引)
            SceneManager.LoadScene("Scene01");
            // SceneManager.LoadScene(0);
            // SceneManager.LoadScene("Scene01", LoadSceneMode.Additive);
        }
        // 按下2键时
        if (Input.GetKeyDown(KeyCode.Alpha2))
        {
            // 同步加载到Scene02或1号场景(场景的名称或者索引)
            SceneManager.LoadScene("Scene02");
            // SceneManager.LoadScene(1);
            // SceneManager.LoadScene("Scene02", LoadSceneMode.Additive);
        }
    }
}
```

LoadSceneMode（可选）：加载模式，可以是LoadSceneMode.Single（默认）或LoadSceneMode.Additive（一般很少使用）。

Single：加载新场景并卸载当前场景。

Additive：加载新场景时保留当前场景。

异步加载（Asynchronous Loading）

概念：异步加载场景意味着在加载场景的同时，游戏的主线程依然可以继续执行其他操作（例如显示加载动画、播放背景音乐等）。加载操作在**后台进行**，**不会阻塞主线程**。

优点：

- 不会阻塞主线程，游戏仍然可以响应用户操作，例如显示加载动画或提示加载进度。
- 更适合大型场景或需要平滑过渡的情况，提供给用户更好的体验。

缺点：

- 需要更多代码逻辑来处理加载完成后的激活等操作。
- 如果处理不当，可能导致复杂的逻辑错误。

异步加载（Asynchronous Loading）

使用方法：

```
public class LoadSceneAsyncExample : MonoBehaviour
{
    private void Update()
    {
        // 按下1键时
        if (Input.GetKeyDown(KeyCode.Alpha1))
        {
            // 开始异步加载Scene01场景
            StartCoroutine(LoadScene("Scene01"));
        }
        // 按下2键时
        if (Input.GetKeyDown(KeyCode.Alpha2))
        {
            // 开始异步加载Scene02场景
            StartCoroutine(LoadScene("Scene02"));
        }
    }
    private IEnumerator LoadScene(string SceneName)
    {
        // 异步加载场景
        AsyncOperation asyncOperation = SceneManager.LoadSceneAsync(SceneName);
        // 设置场景为非激活状态，直到加载完成
        asyncOperation.allowSceneActivation = false;
        // 检查加载进度
        while (!asyncOperation.isDone)
        {
            Debug.Log($"加载进度: {asyncOperation.progress * 100}%");
            // 当加载进度达到 90% 时，激活场景
            if (asyncOperation.progress >= 0.9f)
            {
                asyncOperation.allowSceneActivation = true;
            }
            yield return null;
        }
    }
}
```

AsyncOperation：一个异步操作对象，包含加载状态的信息。

isDone：场景是否加载完成。

progress：加载进度，范围为[0, 1]。注意，**加载进度最大为0.9，剩下的0.1是场景激活的阶段。**

allowSceneActivation：控制场景是否立即激活**（场景的激活需要手动控制）**。

同步与异步加载场景区别（总结）

3. 同步 vs 异步 加载场景对比

特性	同步加载	异步加载
阻塞主线程	是	否
加载进度控制	无	可获得加载进度信息
用户体验	可能出现卡顿	平滑加载，适合大场景
代码复杂性	低	高
适用场景	小型场景，快速加载	大型场景，复杂过渡

- **小型场景或简单项目：使用同步加载，效率高且实现简单。**
- **大型场景或复杂项目：使用异步加载，并结合进度条或加载动画提供更好的用户体验。**
- 场景加载模式：根据需求选择Single或Additive模式，Additive模式适合子场景加载或场景叠加。

【Unity基础教程】重点知识汇总

（七）

Unity同步与异步加载场景