计算机系科协暑培 5.0

Unity 项目结构与项目管理

计算机系学生科协 谢语桐

2024 年 8 月 14 日

本讲内容

① Unity 开发常用技巧

- 在 Unity 中处理 JSON 数据
- 在 Unity 中使用 WebSocket 进行通信
- Unity 异步与多线程

② Unity 项目管理

- 使用 Git 管理 Unity 项目

③ Unity 项目结构

- 项目实例

在 Unity 中处理 JSON 数据

JSON 文件是一种轻量级的数据交换格式，它使用键值对的方式来表示数据，常用于客户端和服务器之间的数据传输。

一个例子：

```
[
    {
        "name": "cxk",
        "age" : 25
    },
    {
        "name": "Bob",
        "age" : 30
    }
]
```

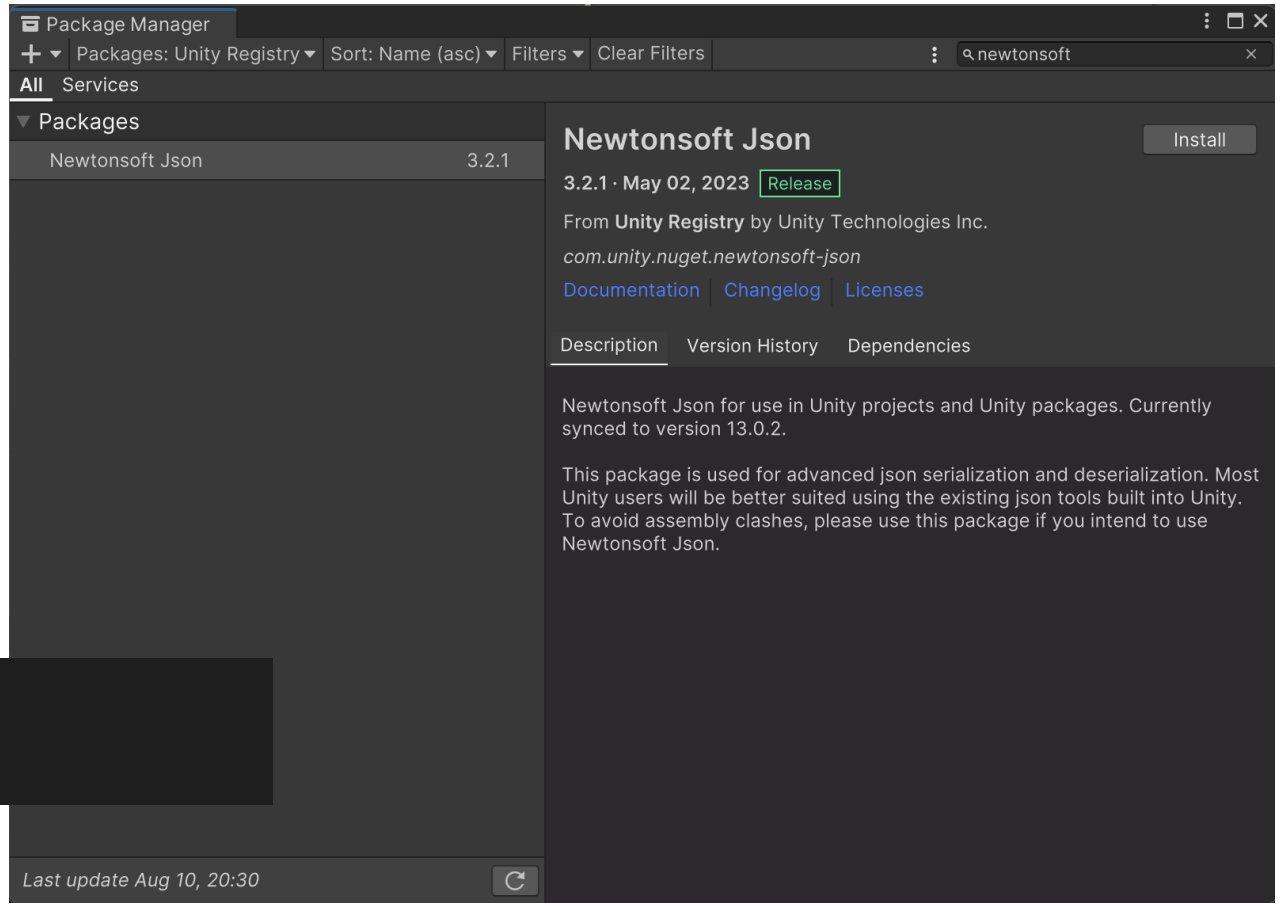
- 用 [] 表示数组，用 {} 表示对象
- 键值对结构中的值可以是数组或其他对象，从而实现 JSON 嵌套
- 键值对可以携带的基本类型：字符串、数字、布尔值

在 Unity 中处理 JSON 数据

Unity 中有一个 Newtonsoft.Json 库，可以用于处理 JSON 格式数据的序列化和反序列化。

安装和使用该库的方法：

- 在 Unity 编辑器的 Package Manager 中搜索 Newtonsoft 并安装。
- 在脚本文件中包含该库




```
3    using UnityEngine;
4    using Newtonsoft.Json;
5
```

在 Unity 中处理 JSON 数据

Newtonsoft 的基本使用方法：

```
T res = JsonConvert.Deserialize<T>(string);
string json = JsonConvert.Serialize<T>(object);
```

```
[
    {
        "name": "cxk",
        "age" : 25
    },
    {
        "name": "Bob",
        "age" : 30
    }
]
```

例如要解析或生成左侧文件，可以在 C# 中定义右侧所示的类：

```
public class NameCard {
    public string name {get;set;}
    public int age {get;set; }
}
```

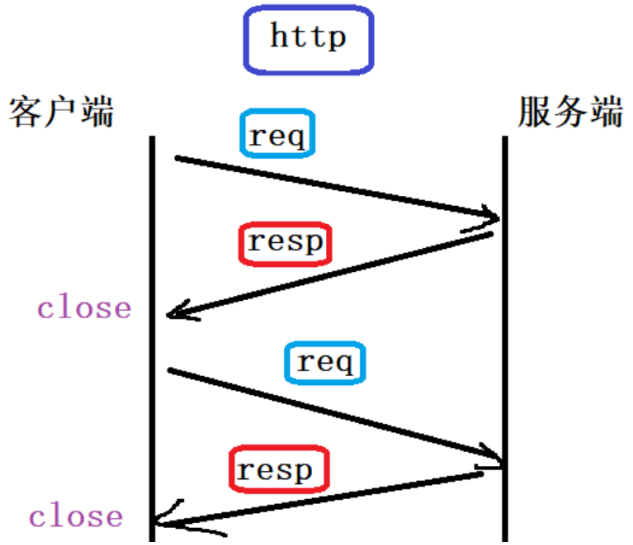
在 Unity 中使用 WebSocket 进行通信

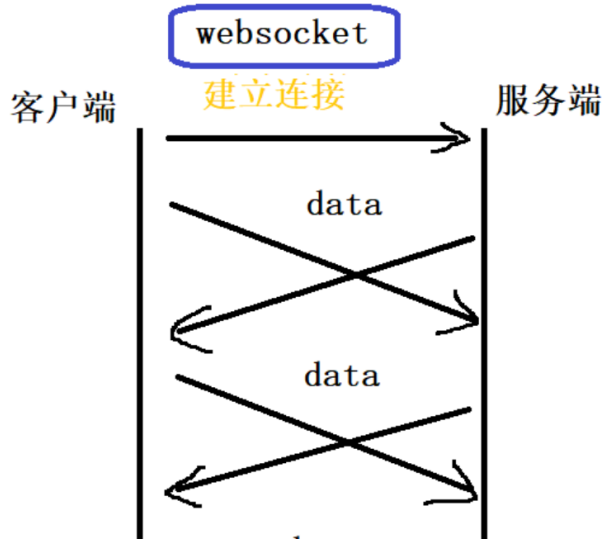
- 什么是 WebSocket？

  同学们在小学期中接触过 HTTP 协议，它是单向的，通常是一方请求、另一方响应。

  设想一种情况：服务端想要主动给客户端发送信息，使用 HTTP 很难实现。

  WebSocket 是一种双向的通信协议，支持双向发送和接收信息。它的协议名为 ws。





在 Unity 中使用 WebSocket 进行通信

- 一个例子

  假设现在想使用 Unity 开发一款双人对战的游戏。游戏双方在客户端上的操作需要发送给服务端，而服务端需要将操作的结果发回客户端。

  假设玩家 A 进行了一次操作，这个操作需要同时显示在玩家 B 的屏幕上，这就需要玩家 B 的客户端设法获取玩家 A 操作的信息。但是 B 客户端并不知道玩家 A 什么时候进行操作，因此也就不知道什么时候向服务端请求关于 A 的操作的数据。

  通过 WebSocket 这种双向的通信协议，客户端 B 不再需要考虑何时主动请求数据，只需要等待服务端将数据发送到客户端 B 即可。

在 Unity 中使用 WebSocket 进行通信

- 在 Unity 中使用 WebSocket

  在 Unity 中我们可以使用 System.Net.WebSockets 类来创建 ClientWebSocket 对象，从而实现客户端到服务端的通信。具体步骤如下：

  ① 初始化、建立连接

```
var ws = new ClientWebSocket();
await ws.ConnectAsync(new Uri("ws://" + uri),
Cancellation.None);
```

  连接失败时将会抛出异常。可以用 try catch 语句处理异常。

微软官方文档链接：https://learn.microsoft.com/zh-cn/dotnet/api/system.net.websockets.clientwebsocket?view=net-7.0

在 Unity 中使用 WebSocket 进行通信

- 在 Unity 中使用 WebSocket

② 监听接收窗口

```
                while (ws.State == WebSocketState.Open)
                {
                    WebSocketReceiveResult result = await ws.ReceiveAsync(new ArraySegment<byte>(buffer),
CancellationToken.None);
                    [redacted]
                }
```

WebSocketState.Open 指示握手完成后的状态，也即正常连接的状态。

buffer 为接受信息的缓冲区。（注意这里接受的是 byte 不是字符串，需要转换）

在 Unity 中使用 WebSocket 进行通信

- 在 Unity 中使用 WebSocket

③ 按需调用 Send 方法与服务端通信

```
try
{
    byte[] bytes = Encoding.UTF8.GetBytes(infromation);
    await ws.SendAsync(new ArraySegment<byte>(bytes), WebSocketMessageType.Text, true, CancellationToken.None);
}
catch (Exception e)
{
    Debug.Log($"Failed to send WebSocket message: {e.Message}");
}
```

④ 关闭连接

使用 Abort() 和 Dispose() 方法，详见文档。

Unity 异步与多线程

- Unity 中的异步编程

  常用于多人线上游戏中监听网络接口、执行网络请求（与网页前后端的原理类似），或加载场景等资源、处理大量数据时。

  C# 中，常用 async 和 await 来实现异步编程，使用 try...catch 进行错误处理。

```
public async Task Example()
{
    SyncMethodA();
    await Task.Delay(1000); // 第一个异步操作
        SyncMethodB();
    await Task.Delay(2000); // 第二个异步操作
        SyncMethodC();
    await Task.Delay(3000); // 第三个异步操作
}
```

```
var thread = new Thread(new ThreadStart(Read));
    // 启动线程
    thread.Start();
    // 等待线程结束
    thread.Join();
    void Read() {
    // Do something
    }
```

Unity 异步与多线程

- Unity 中的异步编程

在 Unity 中使用异步或多线程技术处理大量数据时，要注意 Unity 实际上基于单线程的执行模型，它的主要执行线程是主线程（也称为渲染线程），负责处理游戏逻辑、渲染和用户输入等。Unity 的很多 API 并不是线程安全的，也即不能被多线程调用。

在 Unity 中使用多线程时，需要注意 UnityEngine 的一些 API（比如场景相关的运算等）不能在分线程运行，否则程序会报错或者出现不在期望中的行为；但对于只涉及到基本数据类型（如 int, float, struct 等）的大量运算任务（例如计算相机旋转角度等），可以使用多线程来处理，加快运行速度。

Unity 项目管理

Unity 有一些配套的版本管理工具，如 PlasticSCM 等，这些工具和 Unity 的适配性较强（有些甚至集成到 Unity 编辑器中）。相对来说更方便。

但是同学们最常用也最熟悉的版本管理工具是 Git，而 Git 对 Unity 的适配性比较一般（如果有同学尝试过直接使用 Git 对 Unity 项目进行版本管理，就能明白这句话的意思）。

为了让同学们有较好的开发体验，本课程将详细讲解如何使用 Git 对 Unity 进行版本管理。

Unity 项目管理

① 创建 Git 项目和 Unity 工程（不再赘述）

② 对 Unity 工程进行配置

- 将资产序列化模式设置为强制文本文件
  - 这是因为 Git 只能处理文本文件的合并。对于产生冲突的二进制文件，只能二选一保留，这不利于多人开发。
- 将 meta 文件设置为可见
  - meta 文件包含资产文件之间的关联等信息，如果删除或设置为不生成这些文件，就会造成关联丢失等问题。

Unity 项目管理

③ 设置 .gitignore

- 建议直接按照该仓库中的模板来设置：

  https://github.com/github/gitignore/blob/main/Unity.gitignore

- 实际上在 Unity 工程自动生成的文件夹中只有以下目录必须加入版本控制（即不能被 .gitignore 忽略）

  - Assets
  - ProjectSettings
  - Packages

Unity 项目管理

④ 处理合并冲突

- 如果 .unity、.prefab 等格式的文件产生合并冲突，需要将代码 clone 到本地，使用 IDE 打开这些文件手动处理冲突。不能在 Git 的网页编辑器中处理这些冲突。
- Unity 使用一串无意义的数码识别游戏对象和组件，手动处理冲突后，很难保证处理过后的场景正常运行，因此最好避免场景和预制体等文件产生合并冲突，即参与开发的两个人不要同时修改同一个此类文件。
- 当然，对于脚本代码的修改不受限制，因为这部分冲突的处理方式与普通代码文件相同。

Unity 项目管理

对于 Unity 项目中的文件架构，推荐的组织方式如下：

- 所有与 Unity 控制逻辑直接关联的部分，都放在 Assets 文件夹内
- 项目目录的一个典型的组织方式：
  - Materials
  - Prefabs
  - Scenes
  - Scripts
  - Resources
  - Settings
- 尽量不要把过大的文件和美术素材等中间文件放在工程中，否则可能会造成 Git 仓库过大等问题

Unity 项目实例

- 如何使用 Unity 做出一个真正属于自己的游戏？

① 游戏策划：构思游戏玩法、背景故事、美术风格等

② 按照游戏玩法和游戏元素设计主模块和其他模块

- 一般来说一个游戏项目会有一个游戏主控，负责维护游戏数据、调度其他模块
- 外围模块一般有 UI 模块（监控用户输入和用户操作、显示必要的信息等）、网络通信模块（负责与后端服务器连接，发送和接收信息）、渲染器模块（负责控制游戏对象的状态和位置等），还有场景管理模块、资源管理模块、游戏过程监控模块等等。这些模块不都是必需的，可以按照开发需要自行设计

③ 制作相关的游戏元素，如各种预制体、UI 美术素材等。

④ 开发并测试游戏

Unity 项目实例

- 实例：双人对战井字棋游戏中的模块设计

① 游戏过程监控模块：负责保存历史操作

- 维护一个数组，记录双方对战过程中每一步的下子方位，并使用规范化的格式来表示这些数据，从而实现游戏对局的回放
- 通常可以提供保存回放数据、加载回放数据、跳转到某一步操作等接口

② 渲染器模块：负责控制游戏对象的状态和位置、播放动画特效等

- 从主控模块获得棋子的位置等数据后，将棋子的抽象坐标转换到 Unity 世界空间，然后在该坐标处实例化棋子对象（通常是预制体）
- 也可以实现更复杂的效果，比如对象移动时的动画、放置时的特效等

Unity 项目实例

- 实例：双人对战井字棋游戏中的模块设计

③ UI 模块或用户输入监控模块：负责处理用户输入和展示信息

- 该模块需要使用 UI 对象构造一系列用户界面窗口、处理窗口的动画效果和窗口之间的逻辑，以及在玩家作出操作或输入数据的时候将接收到的操作和数据交由主控模块处理（对于在线游戏，往往会通过通信模块发给后端）。

④ 网络通信模块：使用 WebSocket 或其他方式与在线服务端交换信息

- 由于玩家需要看到对方的落子位置，所以需要使用 WebSocket 实现网络通信。
- 该模块具体需要实现的功能就是前面所说的四步：创建、监听和关闭 WebSocket，并将接收到的网络数据交由主控模块处理（如通过渲染器显示落子的位置）。

Unity 项目实例

- 实例：双人对战井字棋游戏中的模块设计

⑤ 场景管理模块：负责处理场景切换

- 假设我们的游戏需要支持段位，那么可以通过切换到不同的场景来表示玩家参与不同段位的对局。场景切换模块就需要通过玩家的段位正确处理场景的切换。
- 另一个例子：闯关类游戏切换关卡也可以通过场景管理模块实现。

⑥ 资源管理模块：负责控制游戏资源

- 例如我们的井字棋可以有悔棋道具、金币和经验值等设定。
- 资源管理模块中可以判定是否能够使用某个道具，在商店购买时检查玩家的金币是否足够等。

Unity 项目实例

- 实例：其他模块

⑦ 成就或游戏进度管理模块

- 假设现在要开发一款单机游戏，其中有一些成就设定，需要在玩家每次启动时正确显示玩家锁定和已经解锁的成就，并且还要显示玩家的游戏进度。
- 可以将成就和游戏进度设计成一种格式（如 JSON 文件）保存在本地，游戏启动时由进度管理模块读取该文件并设置主控中的相关数据，从而由 UI 模块正确显示信息。

以上仅为一些举例。实际开发中，大家可以根据自己的理解和需求，自行设计和实现各个模块的功能。但是请注意尽量降低模块之间的耦合性，便于修改和维护。

参考材料

WebSocket：

https://blog.csdn.net/qq_54773998/article/details/123863493

异步与多线程：

https://blog.csdn.net/qq_42345116/article/details/122055201

https://blog.csdn.net/2401_82584055/article/details/140021442

Unity 版本管理：

https://zhuanlan.zhihu.com/p/57468011

https://github.com/github/gitignore/blob/main/Unity.gitignore

谢谢