数据结构 Project 实验报告

程礼彬 19300740005

一、 实验思路

本次实验完成的是 Project 的 bonus 部分。

本次 Project 的目的是写一个解析器去解析 bvh 格式的文件，并输出一个 json 格式的文件。实验分为两步，第一步是将 bvh 文件中的数据整理成一个 joint 结构体，然后再将这个 joint 用 json 格式输出。

二、 实验步骤

1、搭建实验环境

本次实验代码在本地的 VSCode 上编写，在 WSL 上测试。由于之前没有用过 WSL，所以在报告要求的提示下自己安装了 WSL 和 Ubuntu。一开始安装的时候遇到错误，降低了 WSL 的安装版本之后问题得到解决。在 WSL 上配置了代码存放路径，代码成功运行，至此实验环境搭建完成。

2、bvh 格式文件特点分析

观察 bvh 格式的文件不难发现，bvh 文件的存储结构和 json 文件的存储结构十分接近，都类似于树状存储结构，这启示我 joint 也应该构建成树状结构。然后，观察得到，motion 数据和其他数据是分开的，其行数 frame 会给出。所以，我们还需要单独处理 motion 数据，使其可以和 joint 中其他数据联系起来。

如何处理 motion 数据呢？实际上只需要定义一个行数为 frame 的二维向量，将数据转换成浮点数之后一个一个 push_back 进对应的一维分量即可。为了获取数据，我们需要读入 bvh 文件。读入文件采用 ifstream file 方法，然后我们定义一个 line 的字符串，采用 `getline(file, line)`方法一行一行地读，这样就完成了文件的读入，同时也构建了 motion_tmp 数组，保存了所有的 motion 数据。

3、构建 META 结构体

META 结构体包含数据的总帧数和单帧时间两个数据，这两个数据在 bvh 文件中都可以直接找到。读入 bvh 文件以后，只要找到 frame 和 frame_time 两个字符串即可。这里我采用的方法是：利用 `if (line.find("frame") != line.npos)` 以及 `(line.find("frame_time") != line.npos)`判断（以下简称 `find` 函数），后面的数据就是对应的结果，再用 `atoi((line.substr(x)).c_str())`，即先截取数据对应的字符串，再将字符串转化为整数或者浮点数。总之，META 结构体的构造比较容易。

4、构建 joint 结构体

joint 结构体里面需要包含以下数据，下面根据这些数据类型，依次讨论怎么把它们保存到 joint 数组里。

1、 type: 即关节类型，是字符串，有 ROOT、JOINT 和 End 三种类型。这三种类型都在“{”的上面一行。例如，`if (line.find("End") != line.npos)`，那么就可以写 `p->joint_type = "End"`。

2、 其后空格跟着的就是 name，即关节名，也是字符串。但是要注意到，如果这个关节的类型为 End，则其 name 字段应为父关节名+“_End”，所以遍历到 End 类型时，要把父关节名保存下来。事实上，为了方便起见，在遍历的时候都可以保存父关节名。处理的时候可以定义一个类型为 string 的 vector，结合 split 函数获得 name。但是，这里以及后面都要注意到的一点是，由于制表符（缩进）以及可能的多个空格的存在，对 line 这个字符串不能简单地用空格 split 后直接取第二个值（第一个值是 type），应该要观察数组的结构。Split 操作之后得到的 vector 数组前面几个很有可能为空。不过对于 name，只要取数组的最后一个值即可。

3、 然后是 offset，它是一个长为 3 的浮点数数组，表示该关节相对父关节的偏移。这个在“{”下面一行，也很容易找到。同样定义一个类型为 string 的 vector（Data），结合 split 函数获得 offset 的字符串表示类型，然后用 `node.offset_x = stod(Data[Data.size()-3].c_str())`转换成浮点数。

4、 然后是 channels，它表示在 motion 数据中该关节包含的通道，是字符串数组。这个处理方法是要先读取 CHANELLS 后面跟着的整数，它代表着通道的个数，也是 channels 数组的长度。获得整数之后就可以构建 channels 数组了，处理方法跟上面类似。同时还应该注意到，对于 End 类型而言，channels 为空数组。

5、 接着是 motion。它是一个二维嵌套数组，数组长度为总帧数，每个元素是一个长度和通道数一致的数组，为这个关节在相应帧每个通道的值。这里就要用到前面处理得到的 motion_tmp 二维数组。同时，对于 End 和一些 JOINT 类型而言，若关节通道数为 0，则每个元素都是空数组。但是，在实际上在检查时，不会访问 End 关节的 motion 数据，故 End 关节该字段为空不会影响结果，所以对于这部分处理，只需要生成空数组即可。

那么，如何确定 motion 数组的列数呢？motion 的列数实际上就是 channels 的个数。根据 motion_tmp 二维数组的结构特点，在构建 joint 结构体之前，需要定义一个变量（本实验定义的是 col）去记录 motion 数组里已经遍历的列数。例如，ROOT 里有六个 channel，那么 ROOT 的 motion 数组就是六列，下一次就要从 motion_tmp 第七列开始算。

6、 最后是 children 数组。数组元素嵌套一个 joint，递归表示这个关节的子关节信息，End 关节该字段应为一个空数组。一般来说，都会默认“{”表示一个新的 child，但是这里为了方便起见，每出现一个 type（JOINT、End），就新建一个 child。同理，每出现一个“}”，都代表这个 child 的结束。

讨论完了数据的处理，下面阐述建立 joint 结构体的思路。

首先要确立基本的框架，即确立根节点 ROOT 的一系列参数，包括 type、name、

offset、channels、motion 以及 children 数组（空数组）。在这个框架的基础上，再添加类型为“JOINT”和“End”的数据。

对于如何搭建 joint 结构体，本次实验借鉴了利用广义表构建二叉树的思想，即建立一个存储 joint 数据类型的栈(s)，先将 ROOT 节点进栈，并将栈顶的节点作为母节点，之后，读入的每一行遇到“JOINT”或者“End”，就新建一个 joint 结构体 p，表示 child 节点依次进栈（即 `s.top()->children.push_back(p);s.push(p)`）。遇到“}”时，就把栈顶节点弹出（即 `s.pop()`）。这样一行一行地读，读到出现“MOTION”为止，这样就完成了 joint 结构体的构建。

5、构建 json 文件

获得了 joint 结构体之后，就需要把它转换成 json 文件。json 文件中所需要的参数就在 META 结构体和 joint 结构体里面。为此，转换成 json 文件的处理函数里只需要包括 META 和 joint 这两个参数。

和前面的处理流程类似，先打开一个空的 json 文件，先搭建基本的 json 文件框架。根据 json 的文件要求，先输出 frame、frame_time 的数据，然后再输出具体的信息。构建 json 文件没有必要考虑缩进。

由于 json 文件本身也是树状结构，所以构建 json 文件，再一次用到了递归的思想，依次输入 type、name、offset、channels。输入这些数据的时候要严格按照字符串的格式输入，并且双引号要按字符输入，避免产生编译错误。

接下来要处理 motion 数据。motion 数据的输出依照 type 的不同分为两部分，一个是“JOINT”，一个是“End”。End 类型的构造比较简单，只需要输出 frame 行的二维空数组即可。JOINT 类型的构建也只需要把 joint 中对应的数据输进去即可。唯一要注意的是逗号的处理。需要注意到数组的最后一个元素之后不能有多余的逗号。之前输出元素都是连后面的逗号一并输出，在输出最后一个元素时需要做特殊处理。

最后是处理 children 数据。类似广义表，对于某个母节点，用循环对子节点遍历，重复上面的操作即可。这里需要注意的是由于 children 也是一个数组，所以输出的元素（joint 结构体）之间需要用逗号隔开，同时最后一个元素之后不能有逗号。遍历会在遇到 type 为 End 的元素停止，这样即可输出最终的 json 文件。

6、解析器鲁棒性测试

为了测试程序的鲁棒性，除了用已有的样例文件测试，本实验还参考了以下几个网址，获取了一系列 bvh 格式的文件进行测试。

1、https://blog.csdn.net/u012336923/article/details/50972968

2、http://mocap.cs.cmu.edu/

在测试的过程中，有时候遇到的问题一个问题是输出的字符串结尾包含了空字符，导致后面的双引号和逗号换行输出，使得 json 文件格式错误。后来发现了字符串的输出长度有误，截取了前面部分，修改之后，这个问题得到解决。

三、 实验感想

通过这次实验，我对数据结构中树、栈、广义表的理解更加深刻，对字符串的处理更加熟练，对递归的理解更加深刻，同时也复习了文件的读写操作，了解了命令行和虚拟机的基本知识。当然最重要的还是增强了利用已知知识去解决未知问题的能力，提高了自己的代码能力，意识到了数据结构在计算机科学中的巨大作用，即使是一个文件解析器这种听上去非常高级的东西，用基本的数据结构知识也可以写出来。

四、 参考资料

1、打包文件里面的 DSProject.pdf
2、数据结构 PPT 第五章