# README 作业二 part 1

## 一、文件目录

MatrixStack.cpp
SkeletalModel.h
SkeletalModel.cpp
README.pdf

（说明：提交的文件为修改过的工程文件。因为只做了 part 1，仅修改了 MatrixStack.cpp, SkeletalModel.h, SkeletalModel.cpp 三个文件，及 README.pdf 文件。）

## 二、编译环境

操作系统 Windows 10，IDE 为 Visual studio 2017。

## 三、实现功能

Part 1 功能全部实现：

1. 矩阵栈实现，相应修改为 MatrixStack.cpp 文件各函数的实现。
2. .skel（骨架数据文件）的解析装入，相应实现见 SkeletalModel.cpp 文件中 SkeletalModel::loadSkeleton 函数的实现代码。
3. 关节绘制。在 SkeletalModel.h 文件中添加了 SkeletalModel::drawJointsHelper 函数用于辅助递归绘制，在 SkeletalModel::drawJoints 函数中调用该函数。
4. 骨骼绘制。在 SkeletalModel.h 文件中添加 SkeletalModel:: drawSkeletonHelper 函数用于辅助递归绘制，在 SkeletalModel:: drawSkeleton 函数中调用该函数。
5. 旋转骨架接口，实现为 SkeletalModel.cpp 中 SkeletalModel::setJointTransform 的代码实现部分。

## 四、实现过程

1. 矩阵栈部分。头文件给出了栈的存储形式是 STL 中的 vector 容器，对应栈的 push, pop, top 都是栈数据结构的常规操作，容易实现。Push 的话注意先将现在的栈顶矩阵乘以参数中的 m 矩阵再压栈，这也对应着 OpenGL 的风格。Clear 实现需要注意 clear 完 vector 后再压入一个单位矩阵，这也是对应 OpenGL 的。
2. .skel（骨架数据文件）的解析装入。对应的需要读什么样的文本文件，在执行参数处和框架代码已经做好了，调试一下发现 loadSkeleton 函数的字符串参数就是文件的路径，因此用 fstream 读入即可。解析思路就是构造 Joint 并存入模型，对于每一个 Joint 相对于父节点是一个平移变换，文件给出了平移变换的向量和相对点，用 Matrix4f 中的函数将向量转化为变换矩阵，存储至对应关节的 transform 矩阵即可。
3. 关节绘制。题目提示了用递归生成，这里我构造了一个递归函数 drawJointHelper 递归绘制，在 drawJoint 函数中调用以下即可。递归过程为：从根关节开始，压入当前关节的变换矩阵，绘制当前关节，再递归绘制所有子关节树，然后矩阵栈 pop 操作。递归返回的终点为当前关节无子关节。

   这里注意 push 后 glLoadMatrixf(m_matrixStack.top())，pop 后也要 glLoadMatrixf(m_matrixStack.top())。
4. 骨骼绘制。这里我也同样是构造了一个递归函数 drawSkeletonHelper 递归绘制，在 drawSkeleton 函数中调用以下。这里需要注意的问题是，当前关节可能不仅相

对于父节点做了对应的平移变换，而且可能发生了旋转（在用户操作过程中），即 joint->transform 在用户连续旋转操作中不仅仅是平移变换矩阵了。这样造成的问题是，当前节点到父节点的骨骼朝向是仅平移的向量，但是当前节点的子节点树上的节点是在该节点做出 transform 矩阵变换后绘制的，这个很重要。因此，我分成了两个部分进行绘制：

（1）先将当前节点的 transform 矩阵的平移部分压入矩阵栈，这样就找到了当前节点绘画的位置，然后就可以求得当前节点相对于父节点的 offset 向量，继而画出骨骼，矩阵栈 pop；

（2）然后将当前节点的 transform 矩阵压入矩阵栈，递归绘制子节点相对于当前节点的骨骼，最后矩阵栈 pop。

这样做的好处在于，没有旋转变换当前节点，便于寻找当前节点相对于父节点的 offset 向量，便于寻找骨骼朝向继而画出骨骼。

5. 寻找骨骼朝向的过程。首先是从当前关节的 tranform 矩阵中提取平移的部分。根据矩阵相乘的规律，平移矩阵的样子是这样的：

$$\begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

这一部分在之后和旋转矩阵相乘后，并不会改变最右列的红色三个参数的值，因此通过提取 transform 的最右列三个数放入一个单位矩阵即可提取出平移矩阵。将平移矩阵压入栈中，画出 Z 轴朝向的合适大小的骨骼，然后通过四元数旋转该骨骼至刚刚求得的偏移向量处，（这里还需要旋转以下 Y 轴，让骨骼放置的更自然，和给出的实例效果一样）即画出当前节点到父节点的骨骼，然后 pop 矩阵栈。之后进行 4(2)部分即可递归画出所有骨骼。

6. 旋转骨架接口封装。这个函数需要注意的问题是，如果单纯的将给定标号的关节的 transform 矩阵乘以三个方向的旋转矩阵的话，不可避免地出现错误的结果。事实是，调用该函数时三个旋转轴的角度都是相对于原先的平移矩阵的，因此这里调用时，必须先将现在的 transform 矩阵提取出平移矩阵（方法在上面第 5 点），然后与三个旋转矩阵相乘作为现在的 transform 矩阵，这样才是正确的。

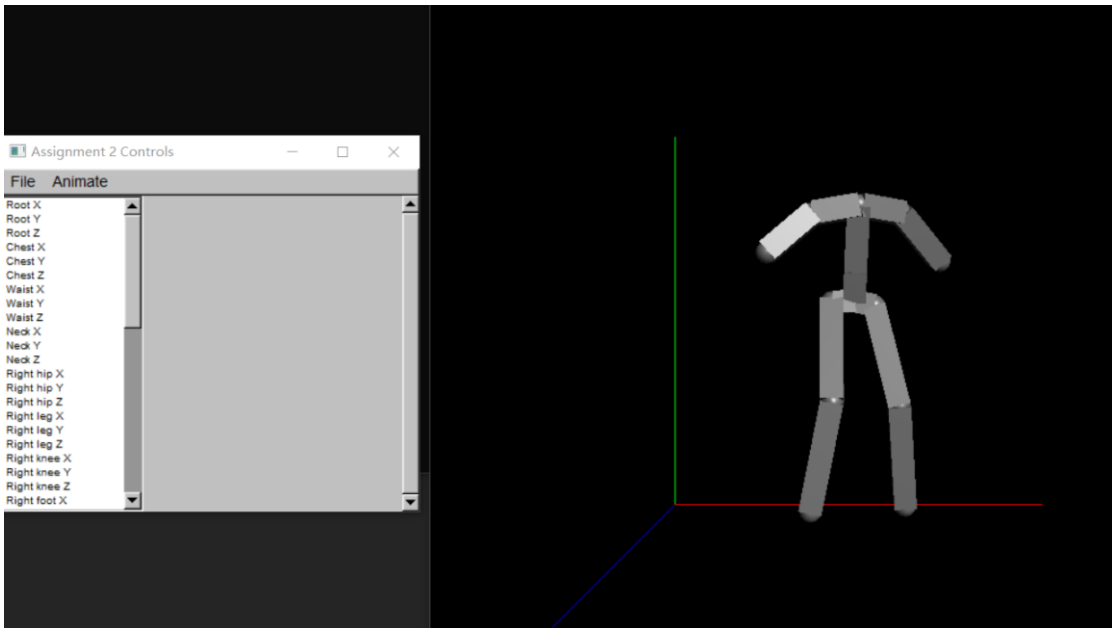
## 五、问题与解决

1. 骨骼连接的过程，寻找 parentOffset 很是让我费解，最后想到了先找出平移矩阵画出骨骼，在用 transform 矩阵递归画子骨骼。过程上述以给出。
2. 旋转骨架接口函数，这个原先没有注意到上述（四.6）讲出的问题，每次发现旋转两次以上就会转过了，转了好几圈，注意到了才进行的改进。

作者：AnDJ

# 效果展示

Model1 未进行旋转操作效果：




根据左边参数进行旋转效果：