README 作业二 part 2

一、 文件目录

1. MatrixStack.cpp
2. Mesh.cpp
3. SkeletalModel.cpp
4. SkeletalModel.h
5. README.pdf

说明:之前提交的part 1部分修改的MatrixStack.cpp文件用于矩阵栈,Mesh.cpp,SleletalModel.cpp,SkeletalModel.h三个文件包含part 1和part 2两个部分的修改的源代码。README.pdf文件为本说明文件。

二、 编译环境

操作系统为Windows 10(64位)，IDE为visual studio 2017。

三、 实现功能

Part 2功能全部实现：

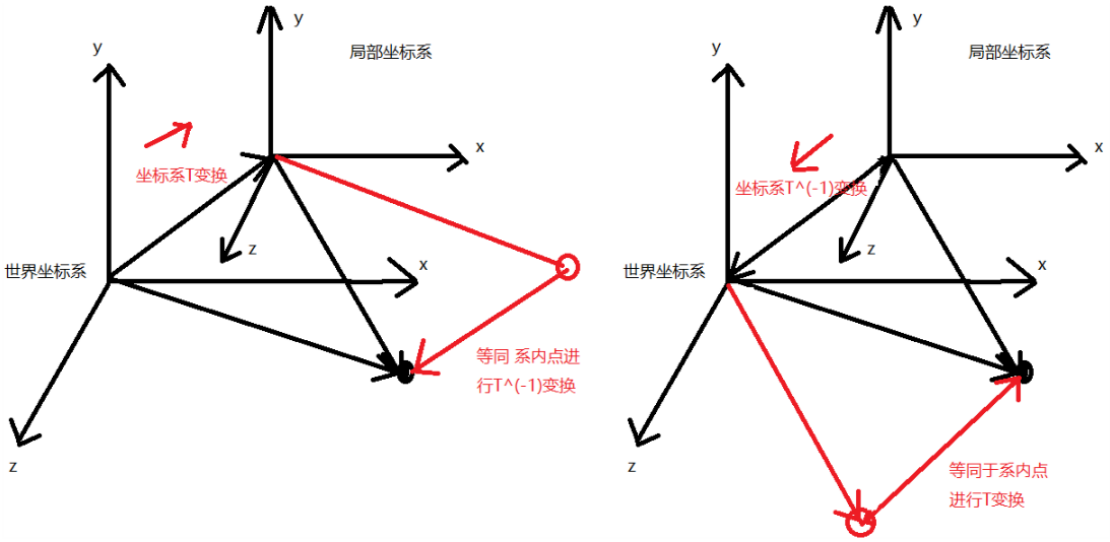
1. 文件输入：绑定姿态网格(File Input:Bind Pose Mesh)，相应源代码修改及实现为Mesh.cpp中Mesh::load函数体的实现。
2. 网格渲染(Mesh Rendering):相应源代码修改及实现为Mesh.cpp文件中Mesh::draw函数体的实现。
3. 文件输入：附加权重(File Input:Attachment Weights)：相应源代码修改及实现为Mesh.cpp文件中Mesh::loadAttachments函数体的实现。
4. 计算变换：实现函数为SkeletalModel::computeBindWorldToJointTransforms和SkeletalModel::updateCurrentJointToWorldTransforms两个函数。这两个函数调用两辅助递归函数SkeletalModel::computeBindWorldToJointTransformsHelper和SkeletalModel::updateCurrentJointToWorldTransformsHelper,这两个函数的实现部分及上述两个函数均在SkeletalModel.cpp文件中。
5. 变换网格(Deforming the Mesh)：相应源代码修改及实现为SkeletalModel.cpp文件中SkeletalModel::updateMesh函数体的实现。

附加部分没有实现。

四、 实现过程

1. 文件输入：绑定姿态网格(File Input:Bind Pose Mesh)。这一部分就是载入OBJ文件，这一部分和作业0的部分差不多，循环载入内存。
2. 网格渲染(Mesh Rendering)。这一部分解决了我作业0的困惑。作业0中我通过画三角形面从而画出来的兔子,其光影效果很差,完全看不到每个三角形之间的拼接。这个题目提示了，在给定的数据下，隐含了每一个面的法向量，即face在文件中给定的三个点是有固定顺序的，如果按照一种既定的方式计算该面的法向量（如normal=(v2-v1)X(v3-v2)),那么必定是相邻的面之间法向量朝向大致相反,从而产生凹凸不平的拼接效果。glNormal3f函数针对每次绘制三角形进行法向量的修改。

3. 文件输入：附加权重(File Input:Attachment Weights)。这一部分需要注意的是 attachment 文件的针对关节 1 的缺省，单行只有 17 个数（针对 model1 来说），关节 1 缺省值为 0。文件中的数据即一张“点数量”*“关节数量-1”）的二维表。
4. 计算变换(Computing Transforms)。这个地方涉及世界坐标和局部坐标之间的转换，思想是全局点放在世界坐标上，每一个点受到所有关节的影响，因此计算单个点受单个关节影响后的位置的过程分三步：(1)将点世界坐标转换为关节局部坐标。(2)将点位置进行关节的 transform 变换。(3)将点变换后的局部坐标再转换为世界坐标。之后进行 SSD 的思想进行加权即可。关于两个坐标系之间的变换见下一点。
5. 两个坐标系之间（世界坐标系和局部坐标系）点的坐标变换。以平移示例，设 T 矩阵为局部坐标系原点相对于世界坐标系矩阵原点的变换矩阵，下图左为世界坐标转局部坐标，下图右为局部坐标转世界坐标示例。



从示例看出，世界坐标转换为局部坐标，变换矩阵为 T^{-1}，P’ =T^{-1}*P；局部坐标转换为世界坐标，变换矩阵为 T， P’ =T*P。针对旋转也可以得出同样的结论。因此，针对单个关节来说：

$$BindWorldToJointTransform=(T_1*T_2*T_3*\cdots T_n)^{-1}$$
$$CurrentJointToWorldTransform=T_1*T_2*T_3*\cdots T_n$$

6. 变换网格(Deforming the Mesh)。更新 Mesh 即可，用 SSD 算法加权计算点的位置，单个关节产生的作用分量：

$$\Delta P = W * CurrentJointToWorldTransform * BindWorldToJointTransform * P$$

累加其，然后更新 currentVertices 即可。

五、 问题与解决

1. 法向量的问题。之前实习 0 时没有做过这个，导致画出来的图形“辨识度”很差，这次实习学习到了这个。
2. 世界坐标到局部坐标之间的变换，这个花费的时间较长，思路在 四(5)做出了解释。
3. 每次旋转计算的时间较长，导致出现些许卡顿，拖动滑块不太顺畅，这个还需要优化。

六、 效果展示

以上是一个滑稽的 pose。

作者：AnDJ