README

作业2

1. 实现功能

本次作业涉及两个部分：第一个部分需要完成显示骨骼并完成转换；第二部分需要完成SSD。

1.1第一部分

功能全部实现。

1. 矩阵栈实现。
2. 读入.skel文件，解析装入。
3. 绘制关节。
4. 绘制骨骼。
5. 实现旋转骨骼接口。

1.2第二部分

功能全部实现。

1. 文件输入：绑定姿态网格。读入并解析文件中的点面数据。
2. 网格渲染。实现Mesh.cpp中的draw，用保存的currentVertices中的数据进行网格渲染。
3. 文件输入: 附加权重。读入.attach中的权重信息，解析并保存。
4. 世界坐标系与局部坐标系互转。
5. 对于模型进行变换，及时进行权重计算并更新。
6. 实现过程

**2.1第一部分**

**1.矩阵栈**

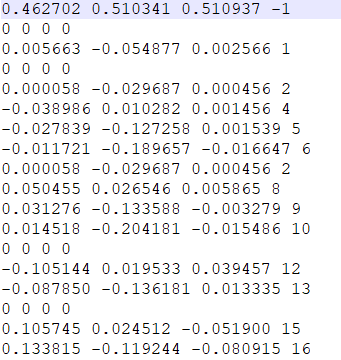
* 栈具有两个基本功能，push(压栈)，pop(出栈)。在本作业中还需要实现clear(清空矩阵栈)，top(返回栈顶元素)。为了模拟栈的操作，使用vector容器可以很容易的实现上述功能，因为vector自带的push\_back,pop,back，与栈的机制类似。
* 初始化中，根据注解“Initialize the matrix stack with the identity matrix.”的提示，得知在栈初始化将identity matrix放入栈中。
* Clear注解“Revert to just containing the identity matrix.”，即clear容器之后，将identity matrix放入栈中。
* Top和pop直接调用vector中的back和pop方法即可。
* 需要注意的是push操作，因为是层次进行建模，最底层的节点需要进行所有祖先节点的叠加，即乘积，所以进行压栈时需要将压栈元素与栈顶相乘。

**2.skel文件解析并装入**

运行程序发现执行的文件名为



打开指定的文件



前三个字段是浮点数给出关节相对其父关节的平移。第四个字段是其父关节的标号。

即将读入的三个字段进行平移后转化成Matrix4f其次矩阵，作为变换矩阵，然后保存到相应父关节标号下的children即可。如果标号是-1则保存到m\_rootJoint（根节点）。

**3.绘制关节**

层次结构保存关节，需要递归绘制。Opengl绘制的过程为先将变换压入栈，等到所有变换都压入栈之后，再从栈顶逐一弹栈操作。这里的原理相同，不过需要递归操作，递归过程为，先将根节点的变换压入栈，然后寻找根节点的子节点，压入子节点的变换入栈，以此类推，遍历所有子树之后，再弹栈，用glutSolidSphere(0.025f, 12, 12)进行绘制。

**4.绘制骨骼**

递归过程大致与绘制关节相同，首先将所有的变换压入栈中，然后pop出变换。接下来开始对变换进行修改。先沿z轴平移1个单位，直接调用scaling返回一个Matrix4f作为变换矩阵m1。接下来进行缩放[0.025, 0.025,L]T，L下一关节与当前关节的距离，可以直接通过变换矩阵保存的平移量表示，因为代表的是相对父节点的平移量，分别平方和开放则为与父节点的距离，实现为获取transform的getCol中的第三列的所有元素，在调用abs求距离，最后调用scaling(0.025, 0.025, L)，返回一个Matrix4f作为变换矩阵m2。最后进行z轴旋转直接调用normalize，根据老师建议直接调用映射y =(z × rnd).normalized(), 而 x =(y × z).normalized(), 其中rnd 为 [0, 0, 1]T，生成旋转矩阵m3，三个变化矩阵都存入栈中。使用glutSolidCube(1.0f)画图，再将三个变化矩阵都弹栈。

**5.旋转骨骼**

使用rotateX，rotateY，rotateZ分别将相对x,y,z轴的旋转量转化成相应的旋转变化矩阵。然后将原变化矩阵的旋转部分通过setSubmatrix3x3修改相应的值。

**2.2第二部分**

**1.读点面数据并绘制**

这里的读点面操作和绘画操作和作业0相似，只不过不同的是作业0没有法向量，所以当时绘制出的图形显示时，并不能看出来兔子是以三角形进行拼接而成。本作业原始代码已经提供了叉乘的方法直接调用，法向量的计算方法即一个面上的任意两个向量进行叉乘，这里使用(v2-v1)X(v3-v2)，代码表示为：Vector3f::cross(v1 - v2, v1 - v3).normalized();

**2. attach文件解析保存**

这里注意attach文件对关节1进行缺省操作，所以在保存关节权重值时，关节数会少1，并且根关节的权重为0。

**3.世界坐标与局部坐标互转**

* 世界坐标转化成局部坐标。因为目前的模型坐标是基于当前世界坐标进行变换而成，如果要将世界坐标转化成局部坐标，那么局部坐标里的点就要以局部坐标系为世界坐标，即映射到世界坐标中，那么就是进行逆操作T^(-1)，可以想成局部坐标系原点移动到世界坐标系的原点中。
* 想象如果局部坐标系已经移动到世界坐标系原点，那么如果想要回到原处，就要进行上一操作的逆操作，即T变换，这就完成了局部坐标转化成世界坐标的过程。

**4. 变换网格**

只需要读入权重数据，通过加权操作进行赋值即可，公式如下：

ΔP = W \* CurrentJointToWorldTransform \* BindWorldToJointTransform \* P

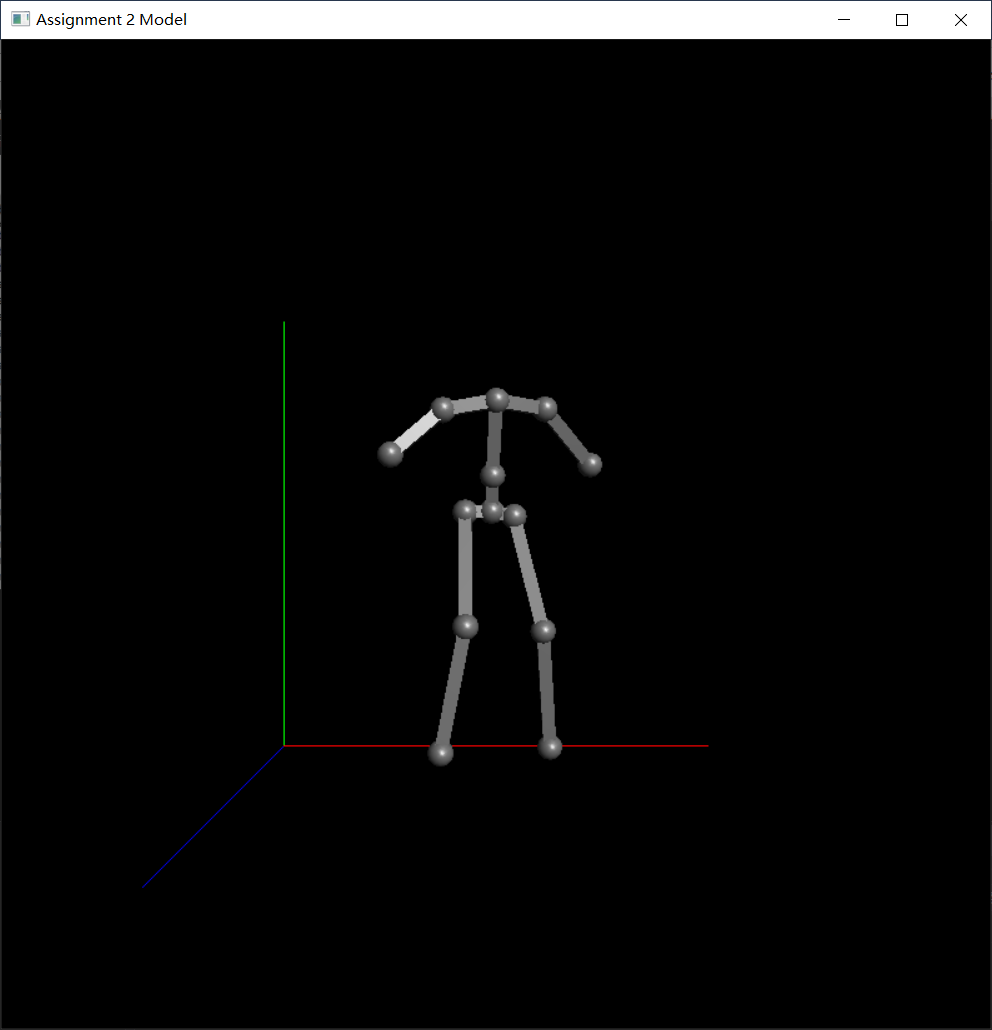
则具体的变换过程为先将世界坐标转化成局部坐标，然后进行变换完成之后再转化回世界坐标，之后再复制权重。

1. 问题与解决

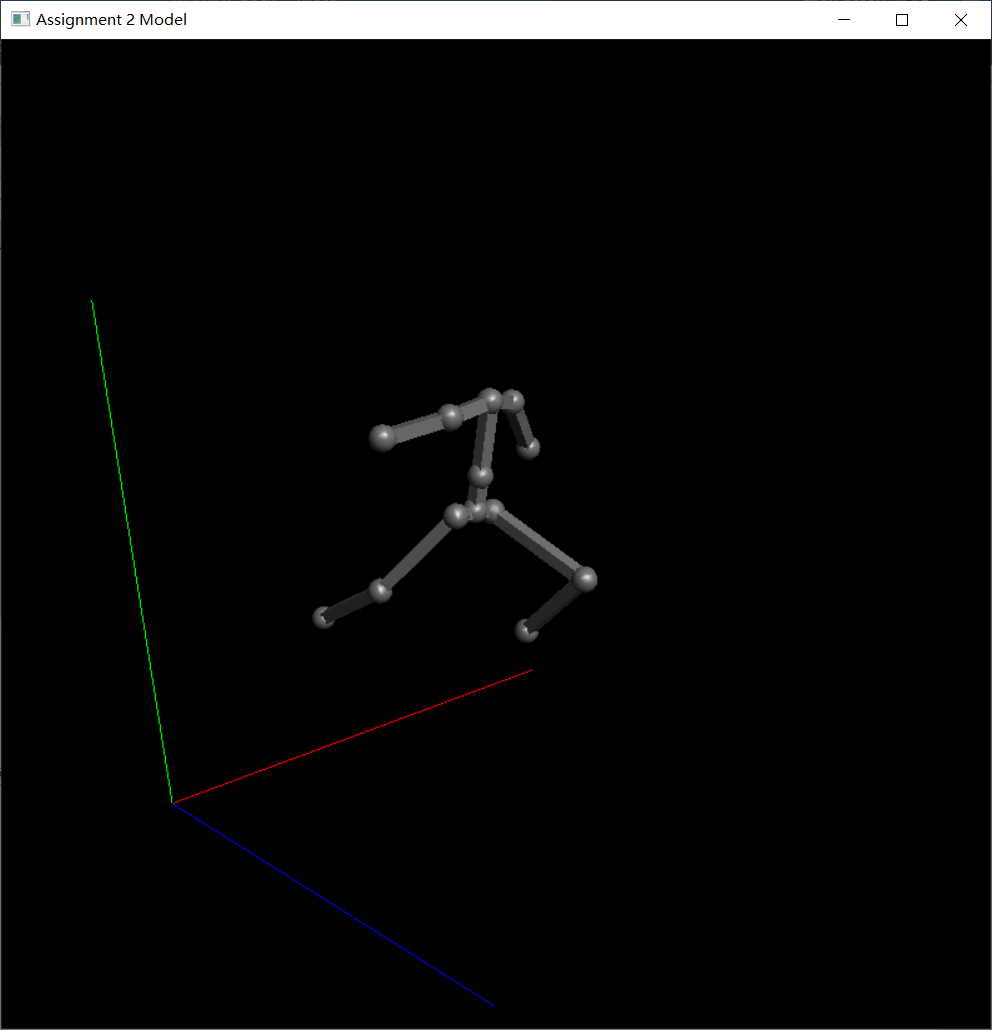
* 在实现第一部分的步骤5中，如果直接将旋转矩阵3x3赋值给关节的变换矩阵中，会发错误因为先前已经进行了相对于原点的平移操作，直接赋值模型就会回到坐标原点，并且都进行旋转操作，解决方法就是仅修改旋转矩阵控制的左上方3x3部分。4x4矩阵其他部分不修改。
* 纠结了很久世界坐标与局部坐标的相互转换的关系，思路在第二部分的第三点进行说明。
* 在进行旋转时，因为要进行计算权重等信息，所以要花费一些时间，所以会发生卡顿现象。
* 附加部分没有实现。

1. 效果展示

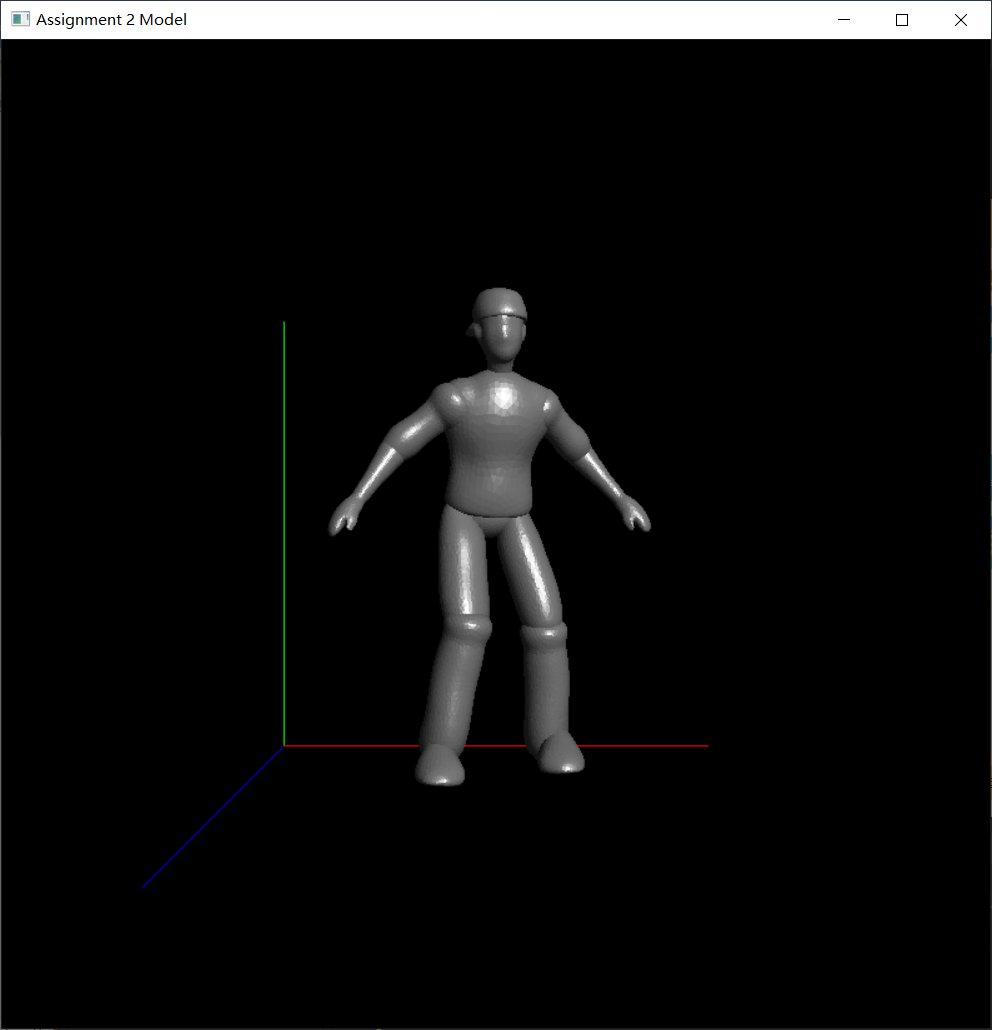
* 未经旋转骨骼模型



* 进行旋转后的骨骼模型



* 未进行旋转的模型



* 旋转后的模型

