Algorithm 3: Sceneview

Comp175: Introduction to Computer Graphics – Spring 2015

Algorithm due: **Wednesday March 25th** at 11:59pm
Project due: **Monday March 0th** at 11:59pm

Your Names: ____Winston Tan___________

____Holt Spalding_____

Your CS Logins: ___hspald01___________

___wtan02_____

Instructions

Complete this assignment only with your teammate. You may use a calculator or computer algebra system. All your answers should be given in simplest form. When a numerical answer is required, provide a reduced fraction (i.e. 1/3) or at least three decimal places (i.e. 0.333). Show all work; write your answers on this sheet. This algorithm handout is worth 3% of your final grade for the class.

OpenGL Commands

[$\frac{1}{2}$ **point each**] Suppose you want to apply a transformation matrix to some vertices. In what order should you use the following five OpenGL commands?

```
5 - glEnd()
1 - glMatrixMode(GL_MODELVIEW)
3 - glBegin()
2 - glLoadMatrix()
4 - glVertex4fv()
```

[$1\frac{1}{2}$ **points**] Suppose your program contained a block of code which sent vertices to OpenGL, delimited by `glBegin()` and `glEnd()`. What would be the effect of a call to `glLoadMatrix()` within this block?

An error of GL_INVALID_OPERATION is generated when glLoadMatrix is executed between glBegin() and glEnd().s

Scenefiles

Consider the following excerpt from a scenefile.

```
<transblock>
     <translate x="0" y=".5" z="0"/>
     <scale x=".05" y="1.0" z=".05"/>
     <rotate x="1" y="0" z="0" angle="45"/>
     <object type="primitive" name="cylinder">
          <diffuse r="1" g="1" b="1"/>
     </object>
</transblock>
```

[$1\frac{1}{2}$ **points**] To transform a point on the cylinder C into the desired cylinder C', in which order would you multiply the three transformations: translate (T), rotate (R), and scale (S) to achieve the desired effect?

```
C' = R * S * T * C
```

In a previous question you described how to compose transformations within a single transformation block (trans block). When coding `Sceneview`, you will also have to compose transformations whenever there is an object tree block contained

within a trans block. Consider the following contrived excerpt from a scenefile:

```
<transblock>
  <rotate x="0" y="1" z="0" angle="60"/>
  <scale x=".5" y=".5" z=".5"/>
  <object type="tree">
    <transblock>
      <translate x="0" y="2" z="0"/>
      <scale x="1" y=".5" z="1"/>
      <object type="primitive" name="sphere">
        <diffuse r="1" g="1" b="0"/>
      </object>
    </transblock>
  </object>
</transblock>
```

[$1\frac{1}{2}$ **points**] Suppose you composed the two transformations in the outer trans block, calling the result CTM1, and composed the transformations in the inner trans block, calling the result CTM2. Show the order in which you must multiply these matrices to obtain a single composite matrix with the desired effect on the sphere.

$CTM1 = S_{.5,.5,.5} * R_{0,1,0,60}$
$CTM2 = S_{1,.5,1} * T_{0,2,0}$
Total Composite Matrix (TCM):
$TCM = CTM1 * CTM2$

Parse Tree

Being sure of the order in which matrices must be multiplied puts you well on your way to completing `Sceneview`. The other major hurdle is deciding how you will traverse the parse tree provided by `SceneParser`.

[**1 point**] In your most efficient program design, when and how many times should you traverse the original parse tree?

Assuming the tree provided by SceneParser is n-ary and not threaded in any way, in order to transform a given node and all of its children in the scenegraph, we would traverse the tree using a simple recursive depth-first traversal that might look something like this:

void traverse(Node *node):
** apply relevant transformation to node.**
** for all children of node:**
**   traverse(child of node);**

The nodes of the tree produced by SceneParser only tells us how each object is transformed relative to its parent node. Therefore, we must use a stack in order to keep track of the current composite transformation matrix at any given node, in order to apply our transformation.

[**1 point**] Flattening the parse tree makes it quicker and easier to traverse when drawing the scene. What type of data structure will you use for this flattened tree?

We will use an C++ STL vector to represent our serialized n-ary tree. The node at each index is described below.

[**1 point**] What information will you store at each of the nodes in the flattened tree? Please give valid types and descriptions of any types you are defining yourself.

Each node of the STL vector will contain a SceneNode pointer and an integer representing the number of children a node has, which will help with indexing. Nodes will be stored using the pre-order convention. Also, we will not store the transformation matrices of each object relative to its parents, but the absolute/total composite transformation matrix of each node in the scene. These two things in combination will make applying transformations to a given node and its children relatively easy. For example, if we wanted to apply a transformation X to a node E and all of its children, we need only to multiply E's TCM by X, and multiply the TCM's of all its children nodes by X, which are the nodes to the right within the vector. The number of children of each SceneNode stored in each node of the STL vector

allows us to do this easily, and even in parallel if necessary.

How to Submit

Hand in a PDF version of your solutions using the following command:

```
provide comp175 a3-alg
```