Algorithm 3: Sceneview

Comp175: Introduction to Computer Graphics – Spring 2014

Algorithm due: **Monday March 3th** at 11:59pm
Project due: **Monday March 10th** at 11:59pm

```
Your Names: Louis Rassaby
            Jayme Woogerd

Your CS Logins: lrassa01
                jwoogerd
```

1 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.

2 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?

```
____ glEnd()
____ glMatrixMode(GL_MODELVIEW)
____ glBegin()
____ glLoadMatrix()
____ glVertex4fv()
```

Solution:
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?

Solution:
This won't do anything because the same matrix transformation needs to be applied to all vertices created between the `glBegin()` and `glEnd()` tags. Thus, if it were possible to modify the matrix in the middle of drawing vertices, it would compromise the integrity of the model view.

This is also proven by looking at the openGL documentation, which states that "if any other GL command is executed between `glBegin` and `glEnd`, the error flag is set and the command is ignored."

See *https://www.opengl.org/sdk/docs/man2/xhtml/glBegin.xml*

3 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' = ___________________________ * C

Solution:
Assuming C starts at the origin,

$$C' = T * S * R * C$$

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.

Solution:
The transformation is

$$\mathrm{CTM}_1 * \mathrm{CTM}_2$$

So,

$$\mathrm{sphere}' = \mathrm{CTM}_1 * \mathrm{CTM}_2 * \mathrm{sphere}$$

4 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?

Solution:
The tree traversal need only be done once. Traversing the tree every time we draw would be extremely inefficient.

[**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?

Solution:
We would use a `std::vector` from the C standard library. Any sort of list structure would work.

[**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.

Solution:

You need the type of primitive object, the color, and the matrix of transformations associated with the object.

5 How to Submit

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

```
provide comp175 a3-alg
```