Sceneview Algorithm

1. M_1 = Scale
   M_2 = Translate
   M_3 = Rotate

   M_3 first rotates the cone about y axis by 30 degrees, then M_2 translates 0.4 in x and y direction (movement NE). M_1 then shrinks the cone shape only in y direction.
   For a more logical transformation, we should do transformations in following order: scale → rotate → translate, ordering the transformations in transblock in that order. The coordinates of desired cone will be given by $C' = M_{translate} * M_{rotate} * M_{scale} * C$

2. $M_1 * M_2$

3.
   a. Since the parse tree generates a static scene, it is unnecessary and inefficient to traverse the scenegraph and re-generate the scene every time an object is drawn. A more efficient solution for drawing a static scene is using a different data structure to store the object for that particular object, so that the object data can be accessed without having to traverse the scenegraph.
   b. I would use a hashmap to store object data. This hashmap would use objects as keys to map them to their respective object data, including the transformations and lights. This data structure has a constant lookup runtime, which would speed up obtaining necessary information for rendering objects.
   c. Each node of my data structure will store transformation, light and other object data for an object in an array, mapped to the object key. Transformation data would be resulting composite matrix computed after performing matrix multiplications of translation, rotation, scale matrices in correct order. Then, it will also contain light data such as lighttype, position vec4, direction vec4, and id integers. Finally, the value array in hashmap will also contain object data such as primitive type and material properties.