1. M1 = S, M2 = R, m3 = T

Multiplying by M3 moves the point so it is at the origin. Multiplying by M2 then aligns the axes to align wih the x y and z axes. Multiplying by M1 scales the object.

2. M1 * M2
3. A) It's inefficient because it's several unneeded computations. A scenegraph could potentially be very, very long. Additionally, an object might be drawn several times in a short span of time. The combination of reading the graph numerous times within a short amount of time is inefficient. A more efficient version could be if each object knows the transformations that will be applied to it, or if there is a TransformationCoordinator of some sort that saves the information from the scene graph. Basically, I want that information stored somewhere so I don't have to keep reading it.
   B) I'd probably use a tree of some sort. It's efficient because I can traverse it quickly and store information in the nodes. It also has a recursive structure, where every node can have children. This is similar to how the scenefile is set up.
   C) Each node will represent a transblock. It will have the total transofmation matrix from the list of transformations applied to the object within the transblock. There will also be information about the object within the transblock: he primitive type, color, ambient color, reflected color, specular color, specular exponent, transparency, and index of fraction. These will be stored as std::vectors of floats for all of these values. We also will as a Boolean indicating if it will have texture. Each transformation block in the groupings will be children nodes of a particular node.