The following file explains the code structure of our project.

1. How to use our project

When the program is run, you will be given the option to input a single command line.
Please type in the name of the file you wish to see.

1. Input_config_ymca.txt
2. Input_config_building.txt

When the program executes, the camera will be set to the global camera. By pressing 't' the camera orientation can be changed to the one that revolves around the scene. This can be reversed by pressing 'g' which will return you to the global camera.

2. Scenegraphs: What are they and how are they used in our project?

Scenegraphs are a collection of nodes and branch out similar to a tree with a root up top and dividing further and further into its base components with the leaves of this tree are the basic objects that we will use to make our models.
Scenegraphs described above are in the forms of multiple XML files for our project and are parsed by the program when we provide the files. The header files Group node, leaf node, transform node is used to create and store the respective types of nodes. We use the functions provided by these headers to create, name and add child nodes to them.

Each node will contain the information required by OpenGL to draw the models. The transform nodes will contain the transformations we wish to apply to the child node of the group node. The labels in the XML files are used to distinguish and create the various types of nodes we require.
The YMCA and the two_buildings files are the topmost scenegraphs which include its constituent scenegraph XML file paths. In our scenegraphs, each individual logical group is labeled and then divided into parts. While traversing from the top the program first encounters the global transformations that we wish to apply to the world. Next node is the first logical group node. The first transform node it encounters is the last transformation it must apply while going from child to parent. This part should be carefully written as transformations are not commutative. The leaf nodes of the scenegraph are the basic objects that will use to create the model. For our projects, we use ‘box’, ‘cylinder’, ‘cone’ and ‘sphere.’
The scenegraphs have been built with a "bottom-top" approach. First, we shall look at the design of the Y-M-C-A model.
The file is divided into two parts:

1. The first part consists of 4 files. Each file is used to represent one of the 4 poses. From Y to A and also applies a translation to each of them to create adequate spacing between each humanoid.

2. The humanoid files contain 2 main groupings: upper body and lower body.
3. The upper body is further divided into the pelvis, right arm and left arm along. Transformations are applied to each child node as we make our way from 'palm' to 'shoulder' on both sides. We applied rotations on forearms to create the various shapes for the different letters.
4. The lower body does not have any rotations applied as we do not require to orient them in different ways.

For the buildings, the two_buildings file is the top model scenegraph.

1. It describes the two buildings and a translation to create adequate space and also a ground of different color. It also has a ground file that is used to create the ground for the buildings to “stand” on.
2. The building is then further divided into floors and each floor is described/constructed in a sperate file.
3. The floor.xml file contains the bottom-most scenegraph. It describes each component of the wall we create for the floor. We use blocks to represent the top portion, bottom portion and middle portion of the wall.
4. For the left wall, we have also added a recessed window by describing another separate group.

The Scenegraphrender is a top level program that uses the files described above to render the whole scenegraph. It traverses each node of the scenegraph that we have created. It takes the leaf nodes and adds them to a mesh. Once the mesh is made it adds texture. Then starting from the root of the scenegraph it starts drawing each node with the modelview matrix we provide.