Computer Science 180, Programming Assignments #1-2

Due:
Part 1: 11:59pm, Wednesday, October 18th
Part 2: 11:59pm, Sunday, November 5th

Important notes:

- No late assignments accepted without a documented emergency.
- You can code and debug your assignments using your own PC. However, ***it is imperative that you compile and run your programs on machines in CSIL (CSIL-xx.cs.ucsb.edu, where xx is from 00 to 48) at least once before you turn in your source codes electronically for grading***. This step is essential to make sure that no incompatibility (in header and library files) exists that prevents your programs from being compiled and run in CSIL.
- For all programming assignments, you must keep a copy of your codes in CSIL. ***You must not edit or change these backup copies in any way after you turn them in electronically***. A backup copy on your own computer is ***not*** acceptable.
- If you are missing glut libraries, you can download precompiled glut library from http://www.opengl.org/resources/libraries/glut/

The first two assignments are related and build upon each other. Hence, it is important that you keep the deadlines. Failing to complete an assignment on time will affect your ability to complete the later assignment.

You are the producer of children's video and you would like to generate an animation sequence with several floating, rotating blocks and some numbers flying through the scene (see the animation sequences from Brainy Baby www.small-fry.com on Youtube at http://www.cs.ucsb.edu/~cs180/videos.html).

Important Notes: The Brainy Baby video serves as a "sample implementation." You are encouraged to exercise your imagination to come up with fancier 3D scenes and animation sequences. The requirements below are the *minimum* requirements (e.g., you must have at least 5 blocks with protruding eaves and 3D characters or shapes on each of the surfaces, but you can have more) that you must meet.

In the first assignment, you need to:

1. Generate a *static* scene with at least five blocks, the blocks should have "protruding eaves" along the borders and should have 3D letters, shapes, and/or numbers on each face. You could follow the spatial layout of blocks and letters/numbers on them as shown in the sample video, or design a new scene. Do note that all faces of all blocks must have some 3D patterns, numbers, or letters on them - this applies to the top and bottom faces and other surfaces that are not visible in the video. (These faces will become visible in your animation sequence, as we will allow the user to adjust the camera aim and rotate and translate these blocks in the 2nd assignment.) Furthermore, all these letters and shapes have "volume," that is, they are 3D shapes, not 2D shapes.
2. Generate a string of 3D numbers, 0 to 9 inclusive, and place them in the scene - no animation/movement is required in the first assignment, but they must be clearly visible and be three-dimensional.
3. Generate background polygons of a white color.
4. Everything should be opaque and colored (using GL_FLAT shading without light being turned on). No texture mapping is needed here.

NOTE:

- To be fair to everyone, you are not allowed to use any modeling packages (e.g., Maya and StudioMax) to generate the 3D models and then use an OpenGL plugin to read the 3D model files.

All 3D models (blocks, shapes, letters, and numbers on the blocks) must be generated using OpenGL commands in your programs.

- You need to use two 3D primitive: a cube (for straight edges) and a curved piece (for modeling the curvy parts of numbers like 2, 3, and 8, and alphabets like, B, C, and Q, etc.). You must have curved pieces in your scene; not everything modeled using cubes.
- You must use different shapes, letters and numbers for different faces of all the blocks.
- You must use the hierarchical modeling technique (discussed in class) to model the 3D scene. Do not write your program in such a way that you use thousands of lines of codes to manually position each and every 3D piece in the scene!

In the second assignment, you need to:

1. Animate a block by rotating it about the vertical axis passing through the center of the block. You should rotate some blocks clockwise and other counterclockwise. All five blocks must be in motion. Again, fancier motion involving, say rotation, translation and scaling are allowed.
2. Animate the string of numbers (from 0 to 9) into the scene. The numbers must be opaque and colored. You can design your own animation trajectory.
3. Apply a wood texture to all faces of all blocks.
4. Turn on the light (there should be a single light source on top of the blocks). Use an appropriate OpenGL lighting model to shade the scene.
5. Add the ability to change the viewpoint. The user can move the mouse up and down to move the camera up or down, and move the mouse left and right to move the camera left and right. In any case, the camera's look-at point should be in the center of the scene. The easiest way to implement change of view point is through “press-and-drag.” That is, the user presses down on the left mouse button and drags the mouse to a new location. You should interactively update the viewpoint based on some function of the cursor displacement until the mouse button is released.
6. Add the ability to have shadow. Only “fake” shadow is needed (e.g., shadow cast on a big background plane).

NOTE: As we do not have access to the models used in Brainy Baby, no precise measurement or comparison of your animation sequence with the one in Brainy Baby is possible. Your job is to design your scene models (size of the blocks, trajectory of the numbers and the blocks, camera pose, etc.) to produce an animation sequence that resembles the one in Brainy Baby or come up with your own design with a complexity matches or exceeds the specifications here.