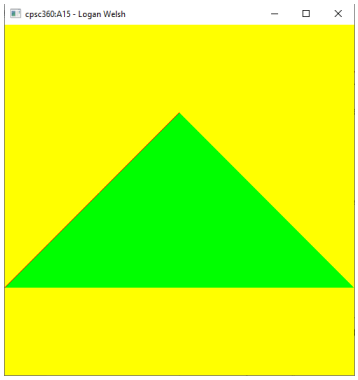
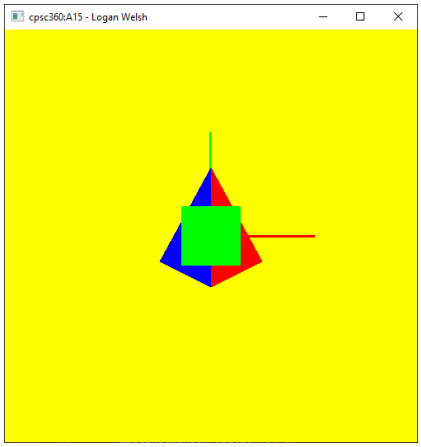
16: Triangle Meshes

1. The two major considerations when assessing mesh representations are efficiency of memory storage and efficiency of queries. Anything we can do to represent the same information with less memory resources will allow us to do more with the same memory hardware. On the other hand, retaining more information allows us to make more efficient use of the system's processing hardware.
2. Mesh topology concerns only the edges connecting vertices, whereas mesh geometry concerns only the locations of the vertices.
3. Data structures:
    a. {(v0, v3, v2), (v0, v1, v3), (v1, v4, v3), (v2, v3, v5), (v3, v6, v5), (v3, v7, v6), (v4, v7, v3)}
    b. {v0, v1, v2, v3, v4, v5, v6, v7}; {(0, 3, 2), (0, 1, 3), (1, 4, 3), (2, 3, 5), (3, 6, 5), (3, 7, 6), (4, 7, 3)}
    c. {v0, v1, v2, v3, v4, v5, v6, v7}; {3, 2, 5, 6, 7, 4, 1, 0, 2}

15: Projection and viewing transformations

4. Tetrahedron Back Face:
    a. We want to move the eye position to the opposite side of the tetrahedron [eye=(0, 0, 0) -> eye=(0, 0, -100)] and reverse our look direction [lookat=(0, 0, -1) -> lookat=(0, 0, 1)].
    b. 

5. Cube at Origin:
    a. Transform the cube
        i. We want to move the cube towards the tetrahedron, so that it is in-between the camera and the tetrahedron [glTranslatef(0, 0, -20)].
        ii. 

    b. Move the camera position
        i. We want to move the camera away from the tetrahedron, so that the cube is in-between the camera and the tetrahedron [lookat=(0, 0, 0) -> lookat=(0, 0, 20)].
        ii.