CIS 441/541: Project #1F
Due May 5th, 2021 (which means 6am May 6)
Worth 8% of your grade

Instructions:

You will add shading to your program and also generate a movie.

1) NOTE: there is a new data member, normal, for the Triangle class.

```
class Triangle
{
  public:
      double       X[3];
      double       Y[3];
      double       Z[3];
      double       colors[3][3];
      double       normals[3][3];
};
```

Normals is indexed by the vertex first and the dimension second.

```
 int vertexId = 0;
 int x = 0, y = 1, z =2;
 normals[vertexId][y] = …;
```

Note: I also added a "double shading[3];" data member to Triangle. I found this to be a helpful location to store per-vertex shading information.

2) You can use the same reader routine from 1E. HOWEVER: you must add "#define NORMALS" at the top of your project1F.cxx file. This will enable "#ifdef NORMALS" commands in the GetTriangles function. You also should change the string that says "proj1e_geometry.vtk" within the GetTriangles() function to be "proj1f_geometry.vtk".

3) Download the file shading.cxx. This file defines a data structure that contains the parameters for shading. I pasted the contents of this file into my code, and encourage you all to do the same. This file also contains a function called GetLighting. This function should be called for every render, since the light position updates with the camera.

4) Extend your code to do Phong shading. Use one-side lighting for both diffuse and specular components. Note: in class on Tuesday April 27th, I said we would use two-sided lighting. I am reversing myself and we will use one-sided lighting. This makes conventions for vector directions very important. See the bottom of this document for more detail.

5) The correct image for GetCamera(0,1000) is posted to the website, as well as the correct images when using only ambient, diffuse, and specular.

You have two deliverables:

1) Your code. This is the only thing you upload to Canvas
2) A movie. This movie should be posted to a website (YouTube, ix.cs.uoregon.edu/~<yourname>, or something else) or shared via the cloud (Google Drive, etc.)

Finally, the very first line of your project1F.cxx code should be:
// Access my movie at: <link>

Being extra clear, you only turn in your source code, and your source code will have a link to your movie.

Note: incorrect images are likely to earn less than half credit. I'd rather have correct submissions late than incorrect submissions on time.

My implementation notes:

- my first step was to add shading as a data member to Triangle.
- I added a fake function that would calculate the shading for a vertex. The function returned 0.5.
- I then added code to LERP the per-vertex shading as I did scanlines, and to modify the output color for a fragment using the shading info.
- I then tested and confirmed it looked right.
- After all of that worked, I implemented the shading equations.

Movie encoders: I imagine most will use ffmpeg. I used mpeg2encode, since I can access it easily through other software I use.

Grading rubric:

- Code with everything correct: 5.5 points
- Movie (on a website or the cloud): 2.5 points

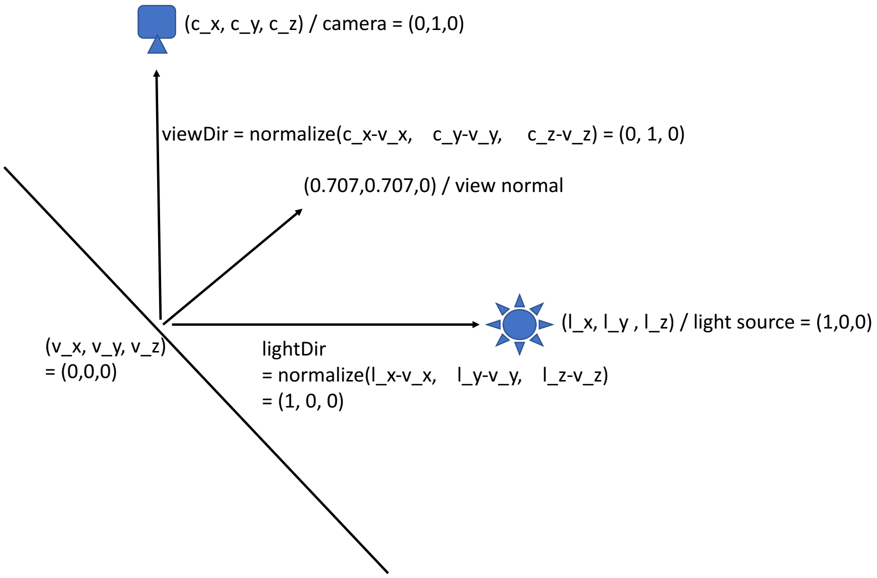
(Part of this assignment is learning a movie encoder … install software, learn to use it, etc. If you want to skip that, you will lose 2.5 points.)

== Convention on vector directions ==

(1) The light source is coming from the triangles. Explicitly, if a triangle vertex is at (0,0,0) and if the light source is at (10,0,0), then the light direction is (1, 0, 0).

(2) The view direction is coming from a triangle vertex. Explicitly, if a triangle vertex is at (0,0,0) and if the camera is at (0,10,0), then the view direction is (0, 1, 0).

This image further clarifies:


(c_x, c_y, c_z) / camera = (0,1,0)
viewDir = normalize(c_x-v_x, c_y-v_y, c_z-v_z) = (0, 1, 0)
(0.707,0.707,0) / view normal
(l_x, l_y , l_z) / light source = (1,0,0)
(v_x, v_y, v_z) = (0,0,0)
lightDir = normalize(l_x-v_x, l_y-v_y, l_z-v_z) = (1, 0, 0)

This example has a triangle vertex, v, at the origin, the camera one unit along the Y-axis and the light source one unit along the X-axis.

The lightDir and viewDir formulas show the conventions we should use for direction for general positions.