Assignment3: Flat Shading

Linear Algebra

Due: November 25, 2024

Introduction

Flat shading is a basic technique used in computer graphics to render three-dimensional objects. In flat shading, each polygon of an object is rendered with a single color, typically based on the color of its surface. This means that all points on a polygon receive the same color, regardless of their position relative to the light source or other factors. While simple and computationally efficient, flat shading can produce a faceted appearance, especially on curved surfaces, due to the lack of shading gradients across the polygons. (ChatGPT)

Figure 1: Ambient Light

Figure 2: Specular Light (Under the assumption that both of the light source and the viewer are infinitely far away from the object)

There are two commonly used light effect to determine the color of a polygon. The first one is the ambient light (natural light), which is directionless and everywhere. An object of the same color with ambient light only has the one color for every polygon, as the image shown on the left. The second one is called the specular light, where the light intensity on different polygons varies based on the reflection direction and the viewer's position. An example of specular lighting is shown in the right image above.

The intensity of the specular light is determined by several factors, including the light source, surface normal, and viewer's position. As shown in Figure 3, retrieved from https://math.hws.edu/graphicsbook/c7/s2.html, vector N is the surface normal; vector V indicates the position of the viewer; vector L represents the light source, and vector R is the reflection of the light. The law of reflection states that the angle between L and N, α, is the same as the angle between R and N, and they lie on the same plane. Moreover, the intensity of the specular light is proportional to the cosine of the angle between R and V, $\cos(\beta)$, raised to the power n to control the sharpness of the reflection.

In this project, we will learn how to compute the flat shading of a given object. The object we use comes from https://github.com/nopjia/tracer/blob/master/data/icosahedron.obj, which is an .obj file. An

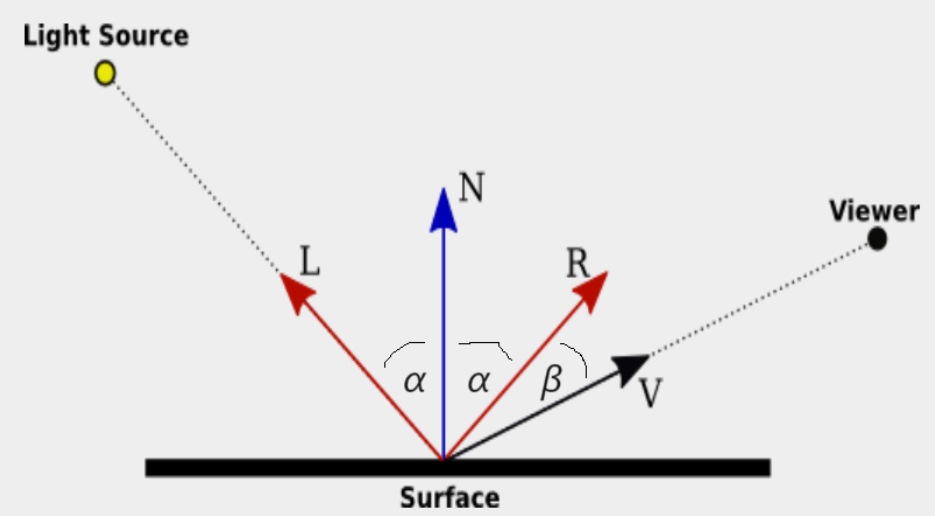

Figure 3: reflection

.obj file provides the 3D coordinates of vertices and records the faces using the bounding vertices in order. The animation (icosahedron_noframe.gif, please check on eeclass) shows how the intensity of the faces changes when the light source moves around the icosahedron.

Assignments

1. (15%) Complete the function **compute_lookat(azim, elev)**, which computes the viewer's direction V. Originally, the viewer is looking from the direction $[x, y, z] = [1, 0, 0]$, but when the plot is shown, the viewer's direction will be changed. How the position changes is characterized by two variables: azimuth and elevation, whose definition can be found at [here].
   Please normalize the vector V when returning it.

2. (15%) Complete the function **compute_normal(P1, P2, P3)**, which computes the surface normal N (outward-facing normal) of each face from three vertices $P1, P2, P3$.
   For faces, .obj files use counter-clockwise ordering by default, which helps determine the orientation of the face normals.

3. (15%) Complete the function **visible(face_normal, lookat)**, which determines whether a face is visible from the viewer's direction. The argument *face_normal* is N and *lookat* is V.

4. (15%) Complete the function **compute_intensity(face_normal, lookat, lightsource)**, which calculates the intensity of the specular lighting.

5. (20%) Perform at least two tasks from the following list, and describe what you have done:

   - Change a model (please choose a convex object).
   - Give each face of the model a different color.
   - Change the movement of the light source.
   - Other interesting changes.

6. (20%) There is a type of terms, called **vn** (vertex normal), in the obj file.

   A common approach assigns vertex normals directly from face normals. While this method is straightforward, it may not yield the smooth transitions necessary for smooth shading.

   In problem 6, please look up the definition of vertex normal and implement it accordingly. The OBJ file with the corrected vertex normals is icosahedron_modified.obj.

Submission

You need to submit

1. HW3_studentID.ipynb
2. HW3_studentID.png
   which is the output from problem 5
3. (Optional) HW3_inputQ5.obj
   if you didn't use the icosahedron_modified.obj in problem 5