CS606 Computer Graphics

Term 2 2021-22

Programming Assignment 3

3D Rendering with Lighting, Shading, and Rotation using Quaternions

Contributors:

ADRIJ SHARMA - IMT2019004

SAMAKSH DHINGRA - IMT2019075

ABHAY ARAVINDA - MT2021002

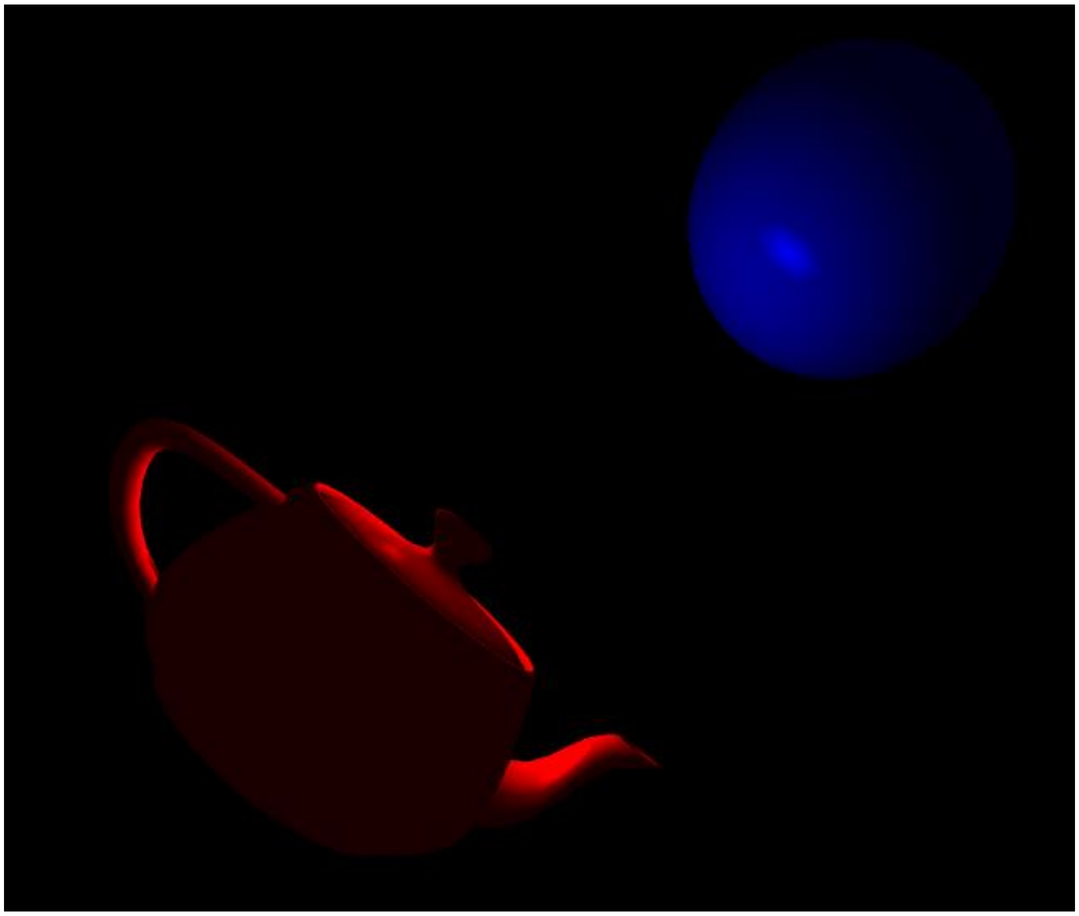
ANSWERS

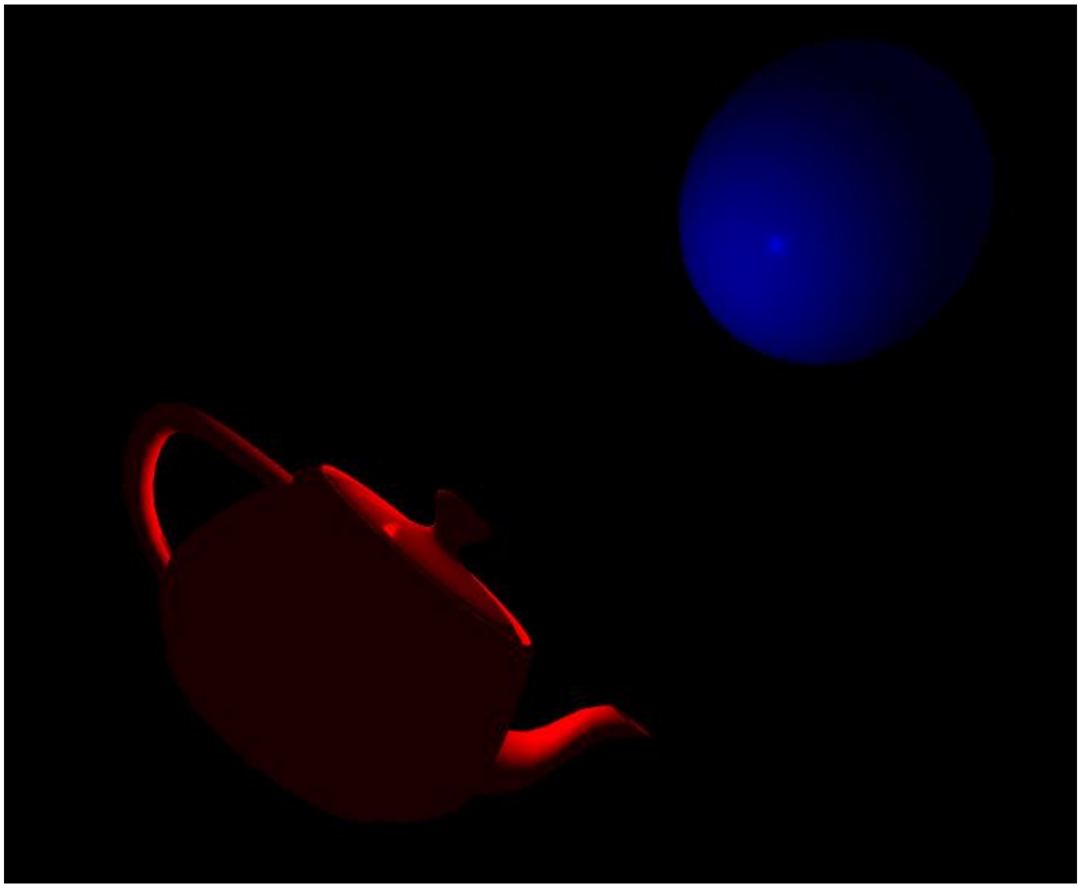
1. ***What are your observations of the distance attenuation terms used for lighting on the sphere and teapot/urn models?***
   - Attenuation calculates the fraction of the original light intensity that is utilized to colour a pixel.
   - The longer and further light go from its source, the weaker it becomes.
   - The primary logic behind attenuation is to create a function that is inversely proportional to a point's distance from the light source. The output would decrease as the distance between the two points grew.
   - The attenuation is proportional to $1/d^2$. However, employing the $1/d^2$ function causes light to progressively dim very rapidly.
   - Instead of $1/d^2$, we use $1/(a + bd + cd^2)$. We strive to set the values of a, b, and c in a range such that the attenuation term becomes proportional to $1/d$. It took a lot of trial and error to figure out the right coefficient values for a, b and c.

2. ***What are your observations about the change in the shading model on the two meshes?***
   - The approach of two shading approaches highlights the differences between them.
   - The surfaces of the objects appear to be smoother and shinier in the Phong model as compared to the Gouraud model.
   - The specular highlight spot on the objects appeared to be sharper in the case of the Phong model as compared to the Gouraud model. The spot was very much distorted (polygons) in the Gouraud model.
   - During the use of mesh for the 3-D models –
   - In Gouraud shading, specular highlights present in the middle of a triangle are not visible. This is because Gouraud shading interpolates the effects of light on the vertices.
   - This problem is resolved in the case of the Phong shading model because the calculation of effects of light is being calculated at each potential fragment of the surface.
   - Phong is computationally more expensive because the effect of light is computed at each point instead of using linear interpolation of light effects calculated at the vertices only.
   - Overall, Phong shading gives comparatively more accurate results than Gouraud shading.

3. ***What are your observations of the individual components of reflection in the Blinn-Phong model for each of the mesh models you have used?***

   Phong lighting model diffuse reflection of rough surfaces and specular reflection of shiny surfaces.
   - Ambient: The ambient lighting is always visible and is not affected by attenuation. If the coefficient of ambient light is 1, the model appears flat. Ambient lighting is independent of the model we are using.
   - Diffuse: A property of diffuse light is that the effect of light on a surface that can be seen better if the light source is kept parallel to the normal to the surface rather than at an angle to the normal. This difference cannot be easily observed in the case of a sphere because the normal at a point will be parallel to the light source even when the sphere is rotated. This is not true in the case of the teapot. The effect of diffuse light can be seen on the bottom surface of the sphere on rotating.
   - Specular: When the surface is less shiny, the specular reflection results in a rough specular area. Causes a bright spot of light to occur on both objects. The spot looks flat in Gouraud shading, whereas, it looks smoother in Phong shading in both the teapot and the sphere.


Gouraud Shading Model – Polygon like (distorted) specular highlight spot


Phong Shading Model – Smooth specular highlight spot

CODE SPECIFICS & APPROACH:

1. ***Lighting and Shading***
   - Point lighted on a single object – Interpolation of the normals for each of the points present on the mesh enables the effect of a more natural object reflecting light from its surface.
   - We made a separate light class for handling the effects of lighting.
   - Each light object has its colour (ambient, diffuse, specular), position, and attenuation factor.
   - Each mesh object has its colour and coefficients of light colours k_a, k_d, k_s.
   - The combination of the colour of light and the colour of the object will give the resultant colour.
   - Directional lighting –
     - Here we learnt about the use of normal in determining the effect of lighting.
     - Directional lighting acts on each face of the object.
     - We calculate the dot product of the normal to a face with the direction of the light and add this result to the colour of the object to observe the overall effect.
   - In the Blinn Phong model, we also have to pass the position of the eye/camera. The shader implemented for this model corresponds to the Phong shading method, in which we calculate the normals and the colour values in the fragment shader.
   - In the Gouraud shading, we perform all the colour calculations in the vertex shader and then send the final colour after the completion of all the calculations calculated colour to the fragment shader.

2. ***Quaternions***
   - The rotation has been implemented using quaternions
   - a system of rotation for the cameras using mouse drag in all directions, though that used basic geometry to move a point along the sphere.
   - Two things were required – axis of rotation, angle to rotate the vector about that
   - Logic is similar to camera rotation. When our mouse from $A(X_1, Y_1)$ to $B(X_2, Y_2)$ on the screen and we have a sphere of unit radius present at the origin, we need to find the corresponding points on the sphere to A and B.
   - We assume the screen is a plane sheet present at some positive Z value. The points on this plane can be mapped onto the sphere by normalizing the coordinates A and B with z-coordinate as Z.
   - Now, we have two points A' and B' on the sphere, between which we have to perform the transformation. These two points are vectors with one endpoint as the centre of the sphere.
   - We can find the axis of rotation to go from A' to B' by taking the cross product of these 2 vectors. The angle of rotation can be found using the dot product of these two vectors.
   - For using the mouse as a trackball, we keep track of the starting point of the mouse down A and the other point B.

3. ***Multiple Light Sources***
   - We need to access the position of all the lights and the colours as well.
   - The position values of the lights get automatically updated while translating.
   - In the shader, we created a struct for holding all these values to make it more modular.
   - We created an array of structs of all the light sources and passed it to each of the mesh objects.
   - To get the resultant colour, we kept adding colours from all the light sources.

4. ***Bounding Box Constraints***
    - We use the inverse of the transformation matrix and multiply it with the position vector of the light source to change it into model coordinates.
    - Then, we compute the bounding box in model coordinates by finding the minimum and maximum x, y, and z values among all vertices in the mesh and multiplying it by 1.25. Direct multiplication is fine because the object is originally centred at the origin in model coordinates.
    - Comparison of light coordinates with the bounding box coordinates checks whether the light is inside the bounding box.

LIST OF THINGS WE LEARNT

1. The lighting of 3D models.
2. Properties of light – specular, diffuse, ambient.
3. One single source of light can change the animation from 2D to 3D.
4. Manipulation of models using the two different shading methods and the difference between the two shading models.
5. Debugging.

REFERENCES

1. ***Documentations:***
    - https://marketplace.visualstudio.com/items?itemName=ritwickdey.LiveServer
    - https://glmatrix.net/docs/module-mat4.html
    - https://glmatrix.net/docs/module-mat4.html#.perspective
    - https://glmatrix.net/docs/module-mat4.html#.lookAt
    - https://webglfundamentals.org/webgl/frustum-diagram.html
    - https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API/Tutorial/Creating_3D_objects_using_WebGL
    - http://www.songho.ca/opengl/gl_transform.html
    - http://www.songho.ca/opengl/gl_projectionmatrix.html
    - https://www.scratchapixel.com/lessons/3d-basic-rendering/3d-viewing-pinhole-camera
    - https://learnopengl.com/Lighting/Basic-Lighting
    - https://learnopengl.com/Lighting/Multiple-lights
    - http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-17-quaternions/

2. ***Videos***
    - https://www.youtube.com/watch?v=-nqIzV9OuQM&list=RDCMUCRQCzMftIUElX03kHjV4rmQ&start_radio=1
    - https://www.youtube.com/watch?v=-nqIzV9OuQM&t=810s
    - https://www.youtube.com/watch?v=Y5eHAfuO9aY&t=491s

- https://www.youtube.com/watch?v=MF1qEhBSfq4&list=PLa1F2ddGya_-UvuAqHAksYnB0qL9yWDO6&index=1
- https://www.youtube.com/watch?v=nIoXOplUvAw&list=PLjEaoINr3zgFX8ZsChQVQsuDSjEqdWMAD

3. ***Repositories***
- https://github.com/Amit-Tomar/T2-21-CS-606
- https://github.com/invent-box/Learn-WebGL
- https://github.com/frenchtoast747/webgl-obj-loader