Assignment #3

Computer Graphics, Spring 2025

ID	201920702
Name	**H**yun**b**in **K**im(김현빈)
Prof.	Ri Yu(유리)

1 Overview

Figure 1: Overall view

The implemented program includes new features such as interactive camera view modes and the Blinn-Phong lighting model.

Changing Camera View Mode

- By Button

  Click the `{Front, Side, Top} View` buttons in the `[Camera View]` section.

- By Mouse Interaction

  Moving the mouse within the canvas dynamically changes the camera view by repositioning the camera toward the pointer location.

Light Shading Model

Based on the Blinn-Phong lighting model, two features have been implemented:

- Adjusting Light Properties: `Ambient, Diffuse, Specular, Shininess`

  You can control the lighting characteristics by adjusting the sliders in the `[Lighting]` section.

- Changing Light Source Position

  You can change the light source's position using the sliders in the `[Light Position]` section.

2 Implementation

2.1 `assignment3.html`

There are two main updates: one for controlling the camera view, and the other for adjusting the position and properties of the lighting. Both features are controlled via sliders.

2.2 `assignment3.js`

2.2.1 `Declaration, Variables`

- `ambientStrength, specularStrength, diffuseStrength, shineStrength, uAmbientStrength, uSpecularStrength, uShininess, uDiffuseStrength`

  These are the parameters used in the Blinn-Phong lighting model. The Blinn-Phong model calculates lighting as the sum of three components: Ambient, Diffuse, and Specular.

  $$I = I_{\text{ambient}} + I_{\text{diffuse}} + I_{\text{specular}}$$

  - `ambientStrength, uAmbientStrength`

    $$I_{\text{ambient}} = k_a I_a$$

    In this formula, k_a corresponds to `ambientStrength`, and I_a represents the color or intensity of the light source. `uAmbientStrength` is the uniform location that passes `ambientStrength` to the shader.

  - `diffuseStrength, uDiffuseStrength`

    $$I_{\text{diffuse}} = k_d \cdot I_l \cdot \max(0, N \cdot L)$$

    In this formula, k_d corresponds to `diffuseStrength`. N is the surface normal vector, and L is the normalized vector pointing toward the light source. `uDiffuseStrength` is the uniform location used to pass `diffuseStrength` to the shader.

  - `specularStrength, shineStrength, uShininess, uSpecularStrength`

    $$I_{\text{specular}} = k_s \cdot I_l \cdot \max(N \cdot H, 0)^{\alpha}$$

    Here, k_s corresponds to `specularStrength`, and α to `shineStrength`. $H = \frac{L+V}{\|L+V\|}$ is the half vector between the light direction L and the view direction V. `uSpecularStrength` and `uShininess` are used to pass their respective values to the shader.

- `lightPos, uLightPos`

  `lightPos` stores the current position of the light source, while `uLightPos` is the corresponding uniform location used to pass that position to the shader.

- `normalBuffer, vNormal`

  `normalBuffer` stores normal vectors for each vertex. These normals are used in lighting calculations to determine how light interacts with the surface. `vNormal` is linked to the `normalBuffer` and is passed to the GPU. It is then used to compute lighting effects in the fragment shader.

2.2.2 shaders

- Vetex Shader(`vs`)

The vertex shader was updated to include normal vector processing by adding a `vNormal` attribute and a `normalMatrix` uniform. These additions allow transformed normal vectors to be passed to the fragment shader via the `fNormal` varying variable, enabling accurate lighting calculations. `fNormal` is used to store the normalized surface normal vectors after applying transformations such as translation, rotation, and scaling.

- Fragment Shader(`fs`)

- Fragment Shader (`fs`)

  The fragment shader was extended to support the Blinn-Phong lighting model. It uses `fNormal` and `fPosition`, passed from the vertex shader, to compute lighting vectors `L`, `V`, and `H`. Using these vectors and user-controlled coefficients, the shader calculates the final color based on the Blinn-Phong equation:

$$I = I_{\text{ambient}} + I_{\text{diffuse}} + I_{\text{specular}}.$$

2.2.3 Functions

- Adding normal vectors of triangles (Line 368–377)

  For each triangle in `triangle_vertices`, extract three points and calculate the surface normal vector. Let $u = p_1 - p_2$ and $v = p_3 - p_1$, then the normal is given by $n = \frac{u \times v}{\|u \times v\|}$.

- Parsing coefficients of the Blinn-Phong lighting model (Line 424–430)

  Retrieves the values of ambient, diffuse, specular, and shininess coefficients from the sliders, along with the position of the light source.

- Camera view mode change via buttons (Line 476–495)

  Updates the values of `eyeX, eyeY, eyeZ` based on the selected view mode (Front, Side, Top), which changes the camera perspective.

- Camera view interaction using mouse movement (Line 497–517)

  Computes the mouse position relative to the canvas using `rect, x, y`. Since normalized device coordinates(NDC) range from $[-1, 1]$, `x` and `y` are normalized to `ndcX` and `ndcY`. These NDC values are then scaled by a factor of 3 for a more noticeable camera movement. Finally, the values of `eyeX, eyeY, eyeZ` are updated accordingly.

2.2.4 `render`

- Calculating `NormalMatrix`

  A 3×3 matrix is sliced from the Model-View matrix (`mvMatrix`) to construct the `normalMatrix`, which is used to transform surface normals correctly after model transformations.

- Updating lighting-related uniforms

  The values of `ambientStrength`, `diffuseStrength`, `specularStrength`, `shininess`, and `lightPos` are passed to the shader via their corresponding uniform variables in each render call.

- Enabling or disabling normal attribute

  When rendering in `TRIANGLES` or `ALL` mode, the normal buffer is bound and `vNormal` is enabled. Otherwise, a fixed normal vector `(0, 0, 1)` is used and `vNormal` is disabled.

- Updating camera position

  The current eye position(`eye`) is passed to the shader as `cameraPos` to support specular lighting effects.

3 Results and Demo

The demo includes all implemented features described above.(Watch)

Figure 2: Camera View Interaction – Side View(Button)

Figure 3: Camera View Interaction – Mouse on Upper Right

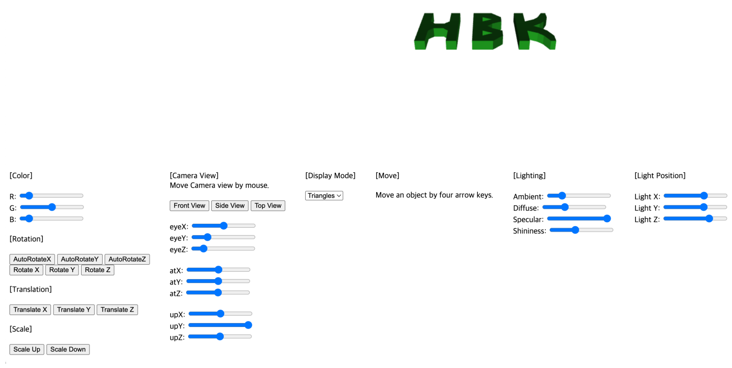

Figure 4: Camera View Interaction – Mouse on Lower Center

Figure 5: Blinn-Phong Model – High Ambient

Figure 6: Blinn-Phong Model – High Diffuse