CSE471 Introduction to Computer Graphics (fall 2019)
Due date: Nov 10, 2019, 11:59 pm

Assignment 2: GLSL Shaders

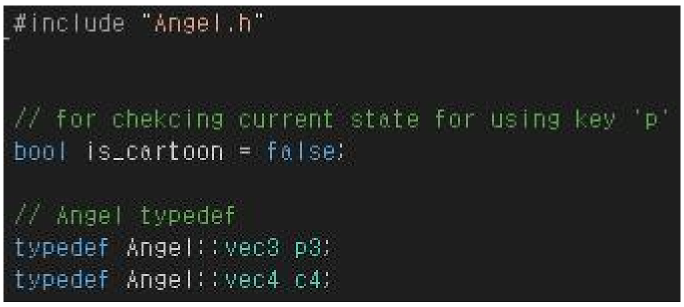
20131218
Park KeeHun

1. Introduction

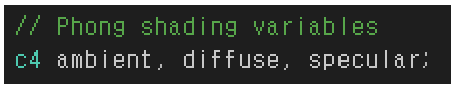
In this assignment, I implement 2 shadings(Phong, (silhouette), cartoon), which are special render effect, by using GLSL shaders. I use the basic teapot model to show these shadings, and this viewer can have several functions. By using mouse control, trackball, zooming, panning function is available, and by using keyboard input, changing parameters of each shading is available.

2. Method

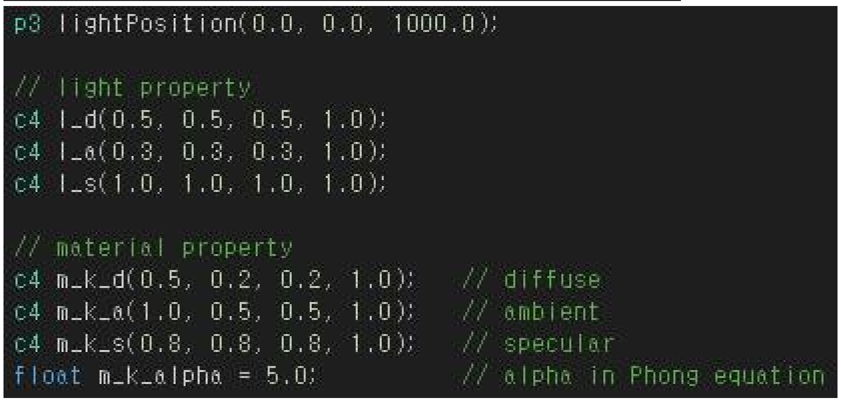
2-1. Phong Shading



```
#include "Angel.h"


// for chekcing current state for using key 'p'
bool is_cartoon = false;

// Angel typedef
typedef Angel::vec3 p3;
typedef Angel::vec4 c4;
```

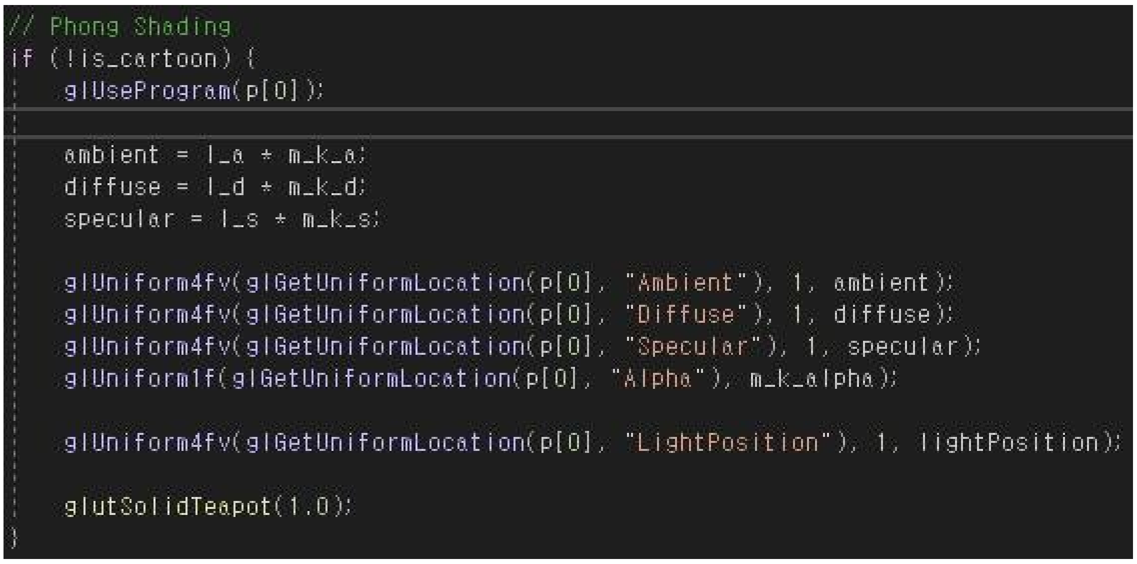


```
// Phong shading variables
c4 ambient, diffuse, specular;
```



```
p3 lightPosition(0.0, 0.0, 1000.0);

// light property
c4 l_d(0.5, 0.5, 0.5, 1.0);
c4 l_a(0.3, 0.3, 0.3, 1.0);
c4 l_s(1.0, 1.0, 1.0, 1.0);

// material property
c4 m_k_d(0.5, 0.2, 0.2, 1.0);   // diffuse
c4 m_k_a(1.0, 0.5, 0.5, 1.0);   // ambient
c4 m_k_s(0.8, 0.8, 0.8, 1.0);   // specular
float m_k_alpha = 5.0;          // alpha in Phong equation
```

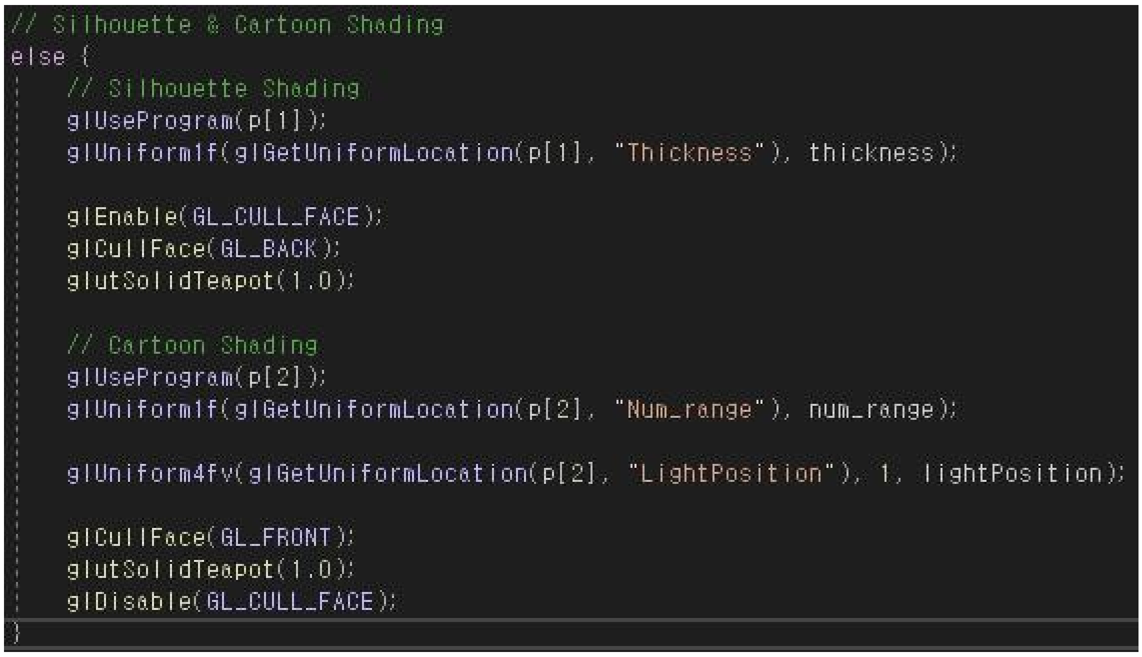
I added the certain header file, “Angel.h”, which can help to deal with vertex and fragment files. So I should add some header files, “Angle.h”, “CheckError.h”, “mat.h”, and “vec.h”. For setting lightPosition, I do typedef vec3 as p3, and for setting parameters of light, material property, I do typedef vec4 as c4. And using boolean variable, is_cartoon, I check whether current shading is phong or cartoon.



```
// Phong Shading
if (!is_cartoon) {
    glUseProgram(p[0]);

    ambient = l_a * m_k_a;
    diffuse = l_d * m_k_d;
    specular = l_s * m_k_s;

    glUniform4fv(glGetUniformLocation(p[0], "Ambient"), 1, ambient);
    glUniform4fv(glGetUniformLocation(p[0], "Diffuse"), 1, diffuse);
    glUniform4fv(glGetUniformLocation(p[0], "Specular"), 1, specular);
    glUniform1f(glGetUniformLocation(p[0], "Alpha"), m_k_alpha);

    glUniform4fv(glGetUniformLocation(p[0], "LightPosition"), 1, lightPosition);

    glutSolidTeapot(1.0);
}
```

In renderScene(), by using if(!is_cartoon), I distinguish phong shading with cartoon. And calculate ambient, diffuse, specular by multiplying light property and material property, which after be sent to phong.frag. In phong.vert, I give values to fragN, fragE, fragL, and setting gl_Position by multiplying gl_ModelViewprojectionMatrix and gl_Vertex. In phong.frag, I take 3 inputs, fragN(normal), fragE(eye), and fragL(light), which are used to calculate diffuse and specular. Then, fragColor is determined by adding ambient, diffuse and specular, which becomes color of object.

2-2. Silhouette / Cartoon Shading

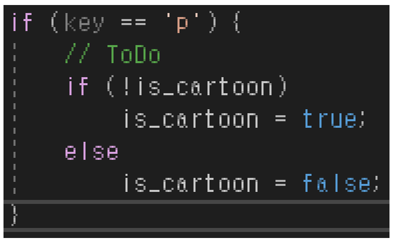


```
// Silhouette & Cartoon Shading
else {
    // Silhouette Shading
    glUseProgram(p[1]);
    glUniform1f(glGetUniformLocation(p[1], "Thickness"), thickness);

    glEnable(GL_CULL_FACE);
    glCullFace(GL_BACK);
    glutSolidTeapot(1.0);

    // Cartoon Shading
    glUseProgram(p[2]);
    glUniform1f(glGetUniformLocation(p[2], "Num_range"), num_range);

    glUniform4fv(glGetUniformLocation(p[2], "LightPosition"), 1, lightPosition);

    glCullFace(GL_FRONT);
    glutSolidTeapot(1.0);
    glDisable(GL_CULL_FACE);
}
```

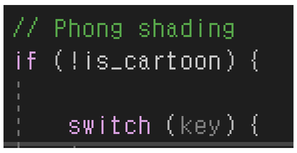
In renderScene(), by using else of if(!is_cartoon), I distinguish cartoon from phong shading. And in this section, I draw the two teapots sequently, first is silhouette shading and cartoon is next. By using glCullFace(), I draw the back_face polygons of the slightly larger object in black and then draw the front-face polygons of the original sized object in white.

In silhouette shading, I send the ‘Thickness’ data to silhouette.vert, and in silhouette.vert, I move the vertex along its vertex normal by adding each component(x, y, z) to Thickness * each normal component. Then, the silhouette.frag will make the black line of the silhouette of the object.
In cartoon shading, I send the ‘Num_range’ data to toon.frag, and the ‘LightPosition’ data to toon.vert. In toon.vert, I give values to fragN, fragL, and calculate gl_Position. In toon.frag, I take 2 inputs, fragN and fragL,and normalize them into N and L. Then I calculate angle by using this equation, angle = abs(acos(dot(N,L)))*180/3.14. After calculating angle, by using for loop, I devide sections (a1 < angle < a2) and assign a constant color value for a certain angle range.

2-3. Keyboard

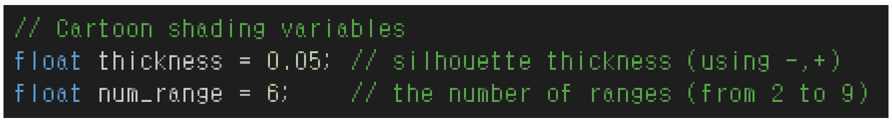


```
if (key == 'p') {
    // ToDo
    if (!is_cartoon)
        is_cartoon = true;
    else
        is_cartoon = false;
}
```



```
// Phong shading
if (!is_cartoon) {

    switch (key) {
```

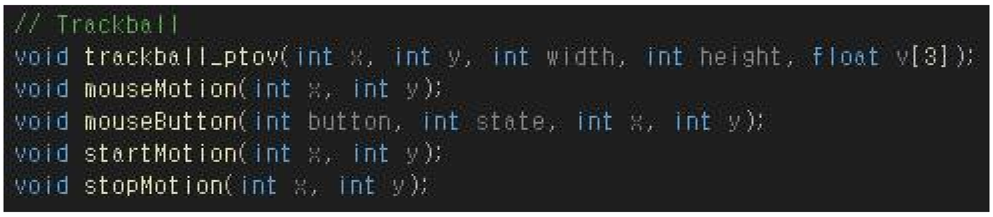
In main function(), I take keyboard callback through glutKeyboardFunc() and keyboard(). In keyboard function, ‘p’ input toggles between phong shader and cartoon shader through boolean variable, is_cartoon.



```
// Cartoon shading variables
float thickness = 0.05; // silhouette thickness (using -,+)
float num_range = 6;    // the number of ranges (from 2 to 9)
```

In phong shading, I can control diffusion, ambient, specular, and alpha. Each key 1 and 3 decreases and increases the diffuse constant, each key 4 and 6 decreases and increases the ambient constant, each key 7 and 9 decreases and increases the specular constant, and lastly each key − and + decreases and increases the alpha in Phong equation.
In cartoon shading, I can control thickness and the number of ranges. Each key − and + decreases and increases thickness of silhouette. And I can choose the number of ranges from 2 to 9, then shader assigns different shades of the color.

2-4. Virtual Trackball

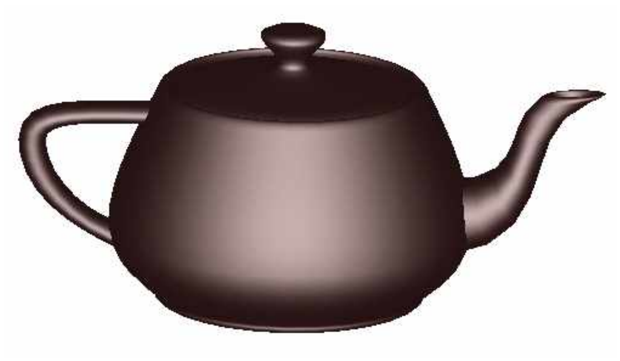


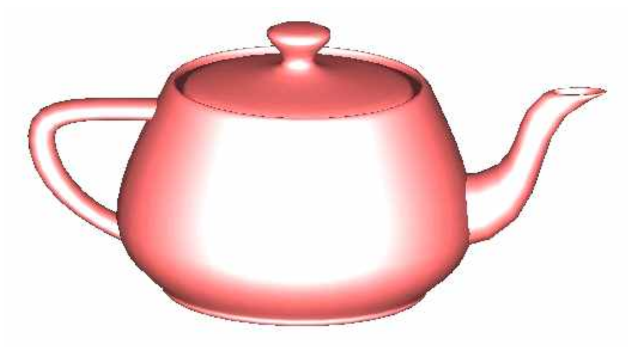
```
// Trackball
void trackball_ptov(int x, int y, int width, int height, float v[3]);
void mouseMotion(int x, int y);
void mouseButton(int button, int state, int x, int y);
void startMotion(int x, int y);
void stopMotion(int x, int y);
```

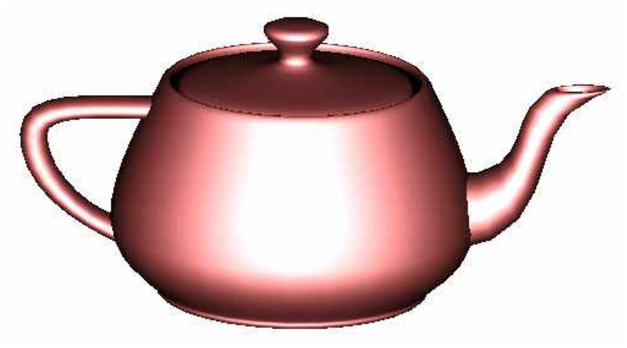
As I refer to Lecture 6 presentation, I make trackball function with mouse callback functions. In assignment 1, I didn’t use the hemi-sphere algorithm, so I had some problems in rotating along z-axis. But in this assignment, I resolve former problems. Left button is for rotation, right button is for zooming along x-axis, and middle button is for panning. I use the boolean variable, trackingMouse, in startMotion() and stopMotion() to treat all mouse buttons in mouseMotion() function, which is updating the position. And I treat the middle button in renderScene().

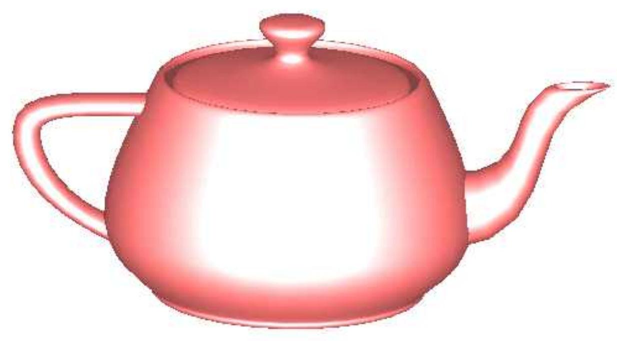
3. Result

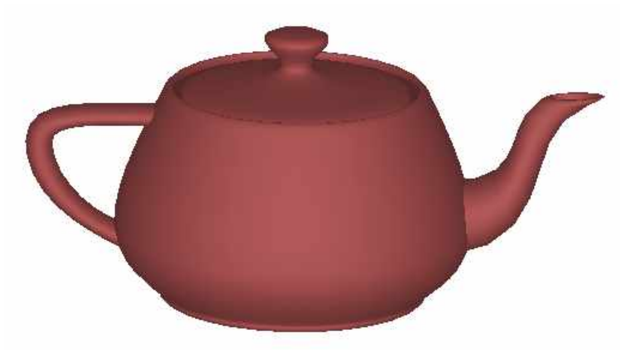
3-1. Phong Shading

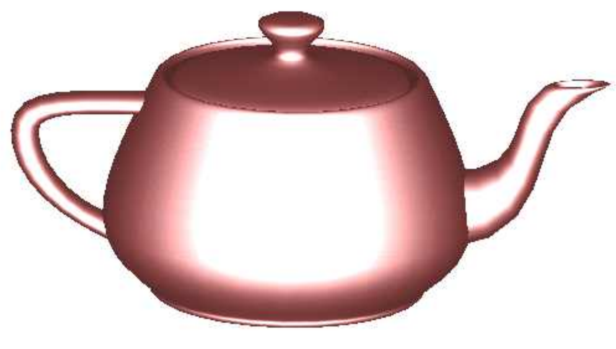

Low Diffusion


High Diffusion


Low Ambient


High Ambient


Low Specular


High Specular

Low Alpha
High Alpha

3-2. Silhouette / Cartoon Shading

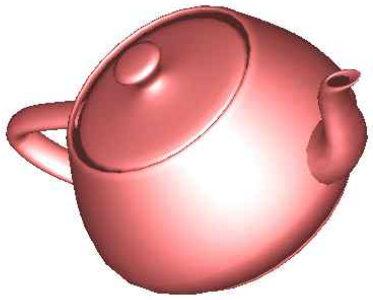
Low Thickness

High Thickness

Number of Range : 2

Number of Range : 9

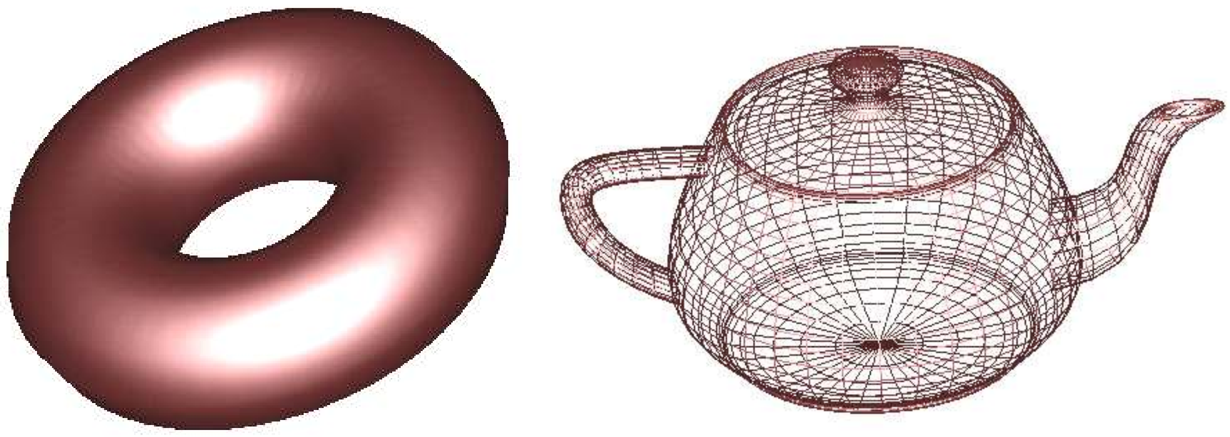
3-3. Virtual Trackball


Rotation

Zooming

Panning(Translation)

4. Conclusion



Through this assignment implementation, I can learn and understand how to use the GLSL shaders. Moreover, through “Angle.h“, I learn that it is even easier to calculate, such as dot product, between vectors. From sending needful informations into shader by using glUniform function to dealing with these informations in frag and vert shader files, I can know about overall process of GLSL shader usage. It is nice to improve the performance of trackball than assignment 1. I think that learning this shading skill must be very helpful to implement our final project, because phong shading is high-quality shader.