Lab #5 – Lighting

Computer Graphics

컴퓨터 그래픽스

남병규 교수님

201002506 진재연

2014. 10. 29

# Lab의 목적

이번 과제의 목적은 OpenGL의 빛을 설정하여 광원에 의한 물체의 쉐이드 표현을 이론에서 벗어나 실제로 다뤄보는 것이 목적입니다.

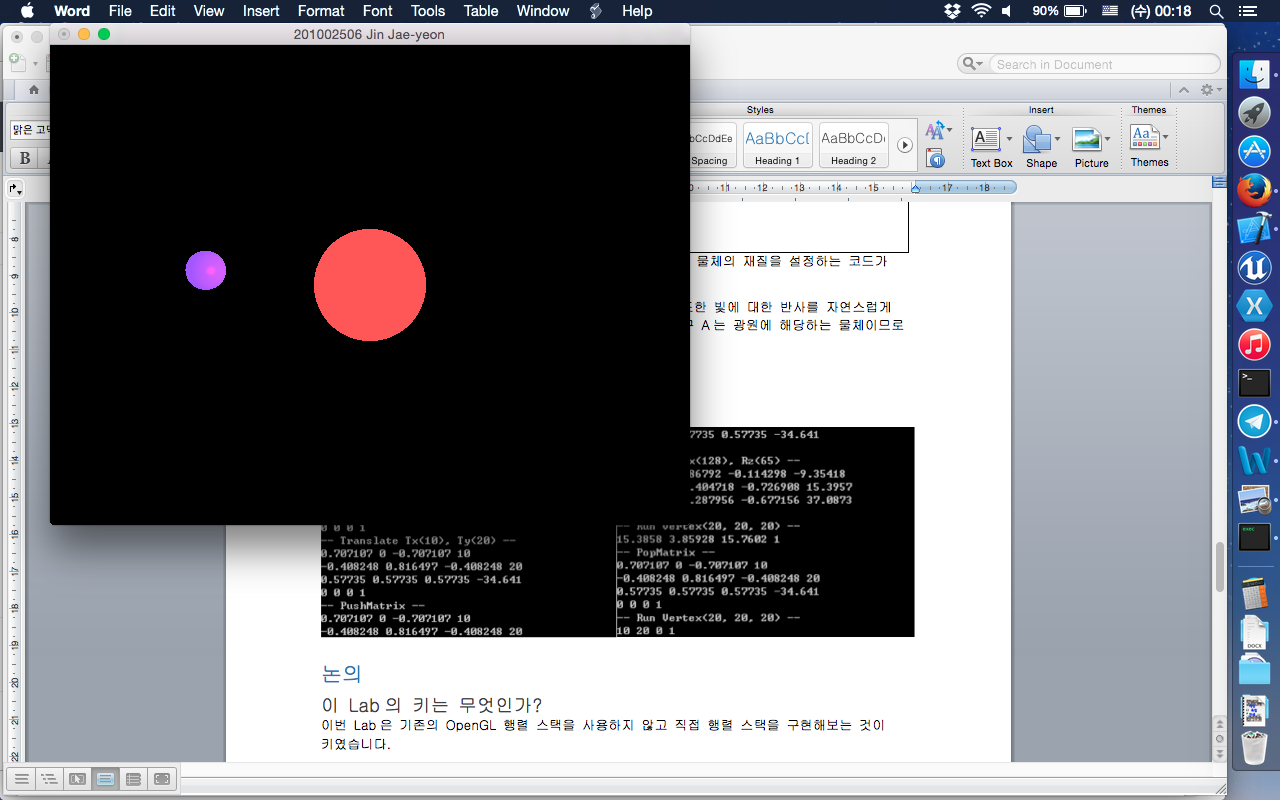
# 소스 코드

|  |
| --- |
| ***main.cpp*** |
| //  // main.cpp  // OpenGLProgramming  //  // Created by 진재연 on 2014. 10. 1..  // Copyright (c) 2014년 진재연. All rights reserved.  //  #include <GLUT/GLUT.h>  #include <OpenGL/OpenGL.h>  float angle\_A, angle\_B;  bool rotation\_A, rotation\_B;  void display() {  // 화면 색상을 A:255, R:0, G:0, B:0으로  // 초기화하도록 설정  glClearColor(0, 0, 0, 1);  // 색상 버퍼를 초기화  glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);    // 행렬 모드를 모델 및 뷰 행렬 모드로 전환  glMatrixMode(GL\_MODELVIEW);  // 단위 행렬을 로드  glLoadIdentity();  // 뷰 행렬에 위치 (0, 24, 72), 방향 (0, 0, 0), 카메라 상위 (0, 1, 0)  // 인 카메라를 적용  gluLookAt(0, 10, 72, 0, 0, 0, 0, 1, 0);    // 구 A  glPushMatrix();  {  // 첫 번째 광원 설정  // 붉은색 빛을 띈다  glEnable(GL\_LIGHT0);  GLfloat diffuse0[]={1.0, 0.2, 0.2, 1.0};  GLfloat ambient0[]={1.0, 1.0, 1.0, 1.0};  GLfloat specular0[]={1.0, 0.0, 0.0, 1.0};  GLfloat light0\_pos[]={0, 0, 0, 1.0};  glLightfv(GL\_LIGHT0, GL\_POSITION, light0\_pos);  glLightfv(GL\_LIGHT0, GL\_AMBIENT, ambient0);  glLightfv(GL\_LIGHT0, GL\_DIFFUSE, diffuse0);  glLightfv(GL\_LIGHT0, GL\_SPECULAR, specular0);    // 광원이 앞면과 뒷면 모두에 적용되도록 한다  glLightModeli(GL\_LIGHT\_MODEL\_TWO\_SIDE, GL\_TRUE);    // 구 B  glPushMatrix();  {  // 자전을 위해 y축으로 반대 회전  glRotatef(angle\_B, 0, 1, 0);  // 구 A로부터 떨어지기 위해 (30, 0, 0)만큼 이동  glTranslatef (30, 0, 0);  // 자전을 위해 y축으로 회전  glRotatef(-angle\_B + angle\_A, 0, 1, 0);  // x축으로 90도 회전  glRotatef(90, 1, 0, 0);  // 자전축 회전을 위해 z축으로 회전을 70도  glRotatef(70, 0, 0, 1);    // 구 B의 재질을 설정  GLfloat ambient[] = {0.2, 0.2, 1, 1.0};  GLfloat diffuse[] = {0.2, 0.2, 1, 1.0};  GLfloat specular[] = {1.0, 1.0, 1.0, 1.0};  GLfloat shine = 100.0f;  glMaterialfv(GL\_FRONT, GL\_AMBIENT, ambient);  glMaterialfv(GL\_FRONT, GL\_DIFFUSE, diffuse);  glMaterialfv(GL\_FRONT, GL\_SPECULAR, specular);  glMaterialf(GL\_FRONT, GL\_SHININESS, shine);    // 구를 그린다  glutSolidSphere(3, 60, 60);  }  glPopMatrix();    // 자전을 위해 y축으로 회전  glRotatef(angle\_A, 0, 1, 0);  // x축으로 90도 회전  glRotatef(90, 1, 0, 0);  // 자전축 회전을 위해 z축으로 회전을 30도  glRotatef(30, 0, 0, 1);    // 구 A의 재질을 설정  GLfloat ambient[] = {1, 0.2, 0.2, 1.0};  GLfloat diffuse[] = {1, 0.2, 0.2, 1.0};  GLfloat specular[] = {1.0, 1.0, 1.0, 1.0};  GLfloat shine = 10.0f;  glMaterialfv(GL\_FRONT, GL\_AMBIENT, ambient);  glMaterialfv(GL\_FRONT, GL\_DIFFUSE, diffuse);  glMaterialfv(GL\_FRONT, GL\_SPECULAR, specular);  glMaterialf(GL\_FRONT, GL\_SHININESS, shine);  GLfloat emission[] = {0.4, 0.1, 0.1, 1};  glMaterialfv(GL\_FRONT, GL\_EMISSION, emission);    // 구를 그린다  glutSolidSphere(7, 60, 60);  }  glPopMatrix();    // 지금까지의 작업 결과를 화면에 출력  glFlush ();  }  void reshape (int w, int h) {  // 화면비 계산  float ratio = w / (float)h;  // 뷰포트를 화면 크기만큼 적용  glViewport (0, 0, w, h);  // 행렬 모드를 투영 행렬 모드로 전환  glMatrixMode (GL\_PROJECTION);  // 단위 행렬을 로드  glLoadIdentity ();  // 45도 각도로 내려보고, 위에서 계산된 화면비를 사용하며  // 1만큼 가까이 있고 1000만큼 멀리 있는 물체를 출력하도록  // 투영 행렬을 적용  gluPerspective (45, ratio, 1, 1000);  }  void keyboardDown ( unsigned char key, int x, int y )  {  switch (key) {  case 'a': rotation\_A = true; break;  case 'b': rotation\_B = true; break;  }  }  void keyboardUp ( unsigned char key, int x, int y )  {  switch (key) {  case 'a': rotation\_A = false; break;  case 'b': rotation\_B = false; break;  }  }  void timer ( int x )  {  if(rotation\_A)  angle\_A += 3;  if(rotation\_B)  angle\_B += 3;    glutPostRedisplay();  glutTimerFunc(1000 / 60, timer, 0);  }  int main(int argc, const char \* argv[]) {  // GLUT 초기화  glutInit(&argc, (char\*\*)argv);  glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB | GLUT\_DEPTH);  glutInitWindowSize(640, 480);  glutCreateWindow("201002506 Jin Jae-yeon");    // 콜백 함수 등록  glutDisplayFunc(display);  glutReshapeFunc(reshape);  glutKeyboardFunc(keyboardDown);  glutKeyboardUpFunc(keyboardUp);  glutTimerFunc(1000 / 60, timer, 0);    // 라이팅 켜기  glEnable(GL\_LIGHTING);  // 깊이 테스트 켜기  glEnable(GL\_DEPTH\_TEST);  glDepthFunc(GL\_LESS);    // GLUT 메시지 루프 시작  glutMainLoop();    return 0;  } |

Lab3 과제인 Solar System을 이어서 작성한 것으로, 광원과 물체의 재질을 설정하는 코드가 추가되었습니다.

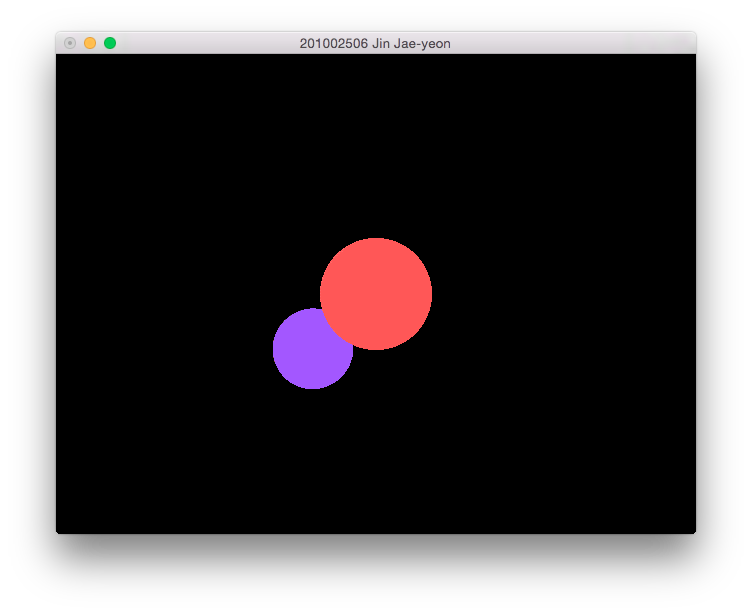
Lighting을 켜고, 1번 라이트에 광원을 설정해주었습니다. 또한 빛에 대한 반사를 자연스럽게 처리할 수 있도록 각 구마다 재질을 입혀주었습니다. 특히 구 A는 광원에 해당하는 물체이므로 이에 대한 처리(Emission)를 추가로 해주었습니다.

# 결과

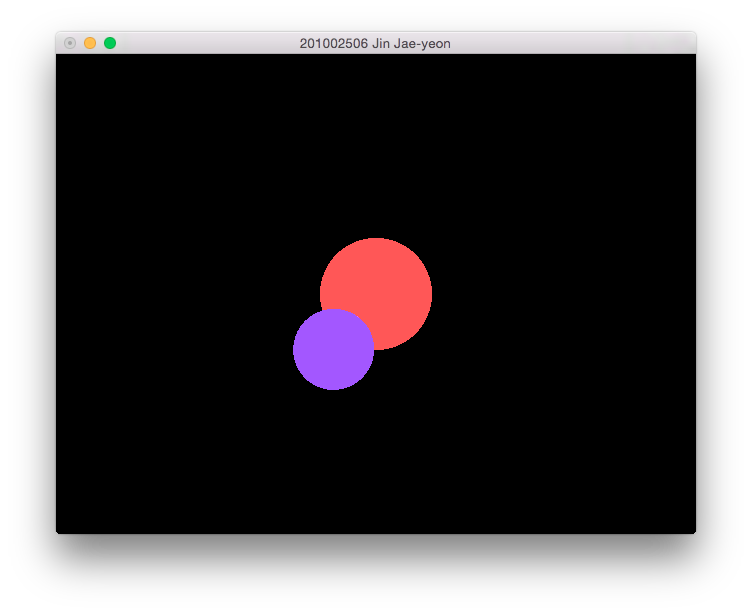


구 B는 원래 푸른색이지만 구 A의 빛에 의해 보랏빛과, 정반사된 붉은 빛이 구 B에 표현됩니다.

단, 이 과제에서 Depth Test 처리를 하지 않는 경우 그리는 순서대로 그려버리기 때문에 구 B가 구 A보다 앞에 있는 경우 아래와 같이 그리게 됩니다.



따라서 GL\_DEPTH\_TEST를 켜고, Z가 작은 것이 앞에 오도록 하면 아래와 같이 나중에 그리더라도 앞에 있는 것이 앞에 그려지게 됩니다.



# 논의

## 이 Lab의 키는 무엇인가?

이번 Lab은 광원과 재질에 필요한 Diffuse, Specular, Ambient 값을 설정하여 빛 효과를 구현해보는 것이 키였습니다.

## 무슨 실수를 하고, 무엇을 배웠는가?

GLUT에서 Depth를 켜지 않으면 GL\_DEPTH\_TEST가 적용되지 않는 점을 몰라 헤매게 되었습니다. 또한 Depth Test를 사용하면 glClear에서 GL\_DEPTH\_BUFFER\_BIT 값도 넣어 깊이 버퍼를 초기화해야 한다는 점도 배우게 되었습니다.

## 프로그램을 어떻게 향상시킬 수 있겠는가?

진짜 항성계처럼 항성의 빛(광원)이 다른 행성들에 반사되고, 다시 행성에서 반사된 빛이 다른 행성이나 위성에 영향을 끼치는 형태로 작성하는 것도 재밌을 것입니다.