Lab #3 –Matrix Push And Pop

컴퓨터 그래픽스 00

남병규 교수님

201002506 진재연

2014. 10. 15

# Lab의 목적

물체의 구조적 변환을 위해 행렬 스택을 이용해 단계별 행렬을 적용할 수 있도록 하는 것이 목적입니다. 이에 따라 OpenGL 함수 glPushMatrix와 glPopMatrix를 사용해볼 수 있습니다.

# 소스코드

|  |
| --- |
| //  // 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);  // 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, 24, 72, 0, 0, 0, 0, 1, 0);  // 구 A  glPushMatrix();  {  // 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 7도  glRotatef(7, 0, 0, 1);    // 화면에 그리는 물체의 색상은 R:50, G:51, B:255  glColor3f(0.2f, 0.2f, 1);  // 구를 그린다  glutWireSphere(3, 10, 10);  }  glPopMatrix();  // 자전을 위해 y축으로 회전  glRotatef(angle\_A, 0, 1, 0);  // x축으로 90도 회전  glRotatef(90, 1, 0, 0);  // 자전축 회전을 위해 z축으로 회전을 3도  glRotatef(3, 0, 0, 1);  // 화면에 그리는 물체의 색상은 R:255, G:127, B:76  glColor3f(1, 0.5f, 0.3f);  // 구를 그린다  glutWireSphere(7, 10, 10);  }  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);  glutInitWindowSize(640, 480);  glutCreateWindow("201002506 Jin Jae-yeon");  // 콜백 함수 등록  glutDisplayFunc(display);  glutReshapeFunc(reshape);  glutKeyboardFunc(keyboardDown);  glutKeyboardUpFunc(keyboardUp);  glutTimerFunc(1000 / 60, timer, 0);  // GLUT 메시지 루프 시작  glutMainLoop();  return 0;  } |

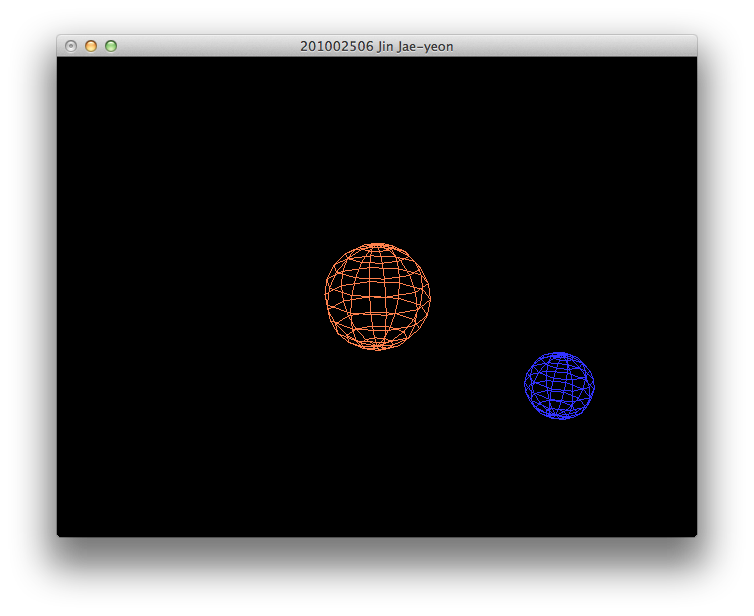
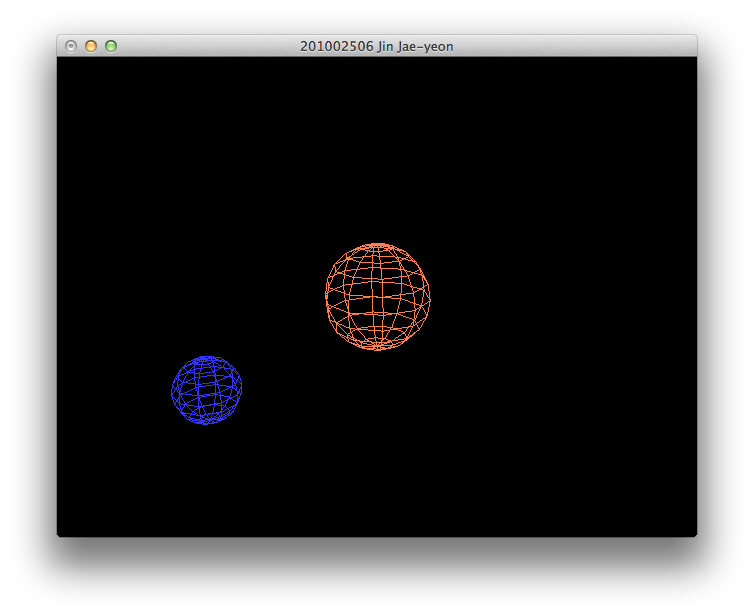
소스 코드 작성은 OS X 10.9.5에서 Xcode 6 상에서 작성되었습니다.

이 프로젝트에서 가장 중요한 코드는 display 함수 내의 코드로, 지정한 조건에 맞는 태양계의 구현을 요구하는 과제에 대한 코드가 작성되었습니다. 단, 과제에서 요구하는 변환 값만큼으로 구현하면 화면상에 잘 보이지 않는 것이 있어 일부 값은 수정했습니다.

키보드에서 A키를 누르고 있으면 자전, B키를 누르고 있으면 공전을 하도록 작성했습니다.

구 A 하위에 구 B가 있는 것으로 하여 구 A의 행렬을 먼저 Push 한 뒤 구 B의 행렬을 Push한 후 구 B를 그리고 구 B의 행렬을 Pop한 뒤 구 A를 그리도록 했습니다.

# 결과



주황색 구가 구 A이고, 파란색 구가 구 B입니다.

# 논의

## 이 Lab의 키는 무엇인가?

이번 Lab은 **OpenGL에서의 행렬 스택을 이용한 구조적 행렬 적용**을 다루는 것이 키였습니다.

## 무슨 실수를 하고, 무엇을 배웠는가?

따로 실수를 한 것은 없었습니다.

## 프로그램을 어떻게 향상시킬 수 있겠는가?

태양계(Solar System)를 작성하는 프로그램이었으므로 태양과 행성A가 아닌 태양계를 실제로 구현하고, 각 행성의 위성을 구현하면 더 재미있을 것입니다