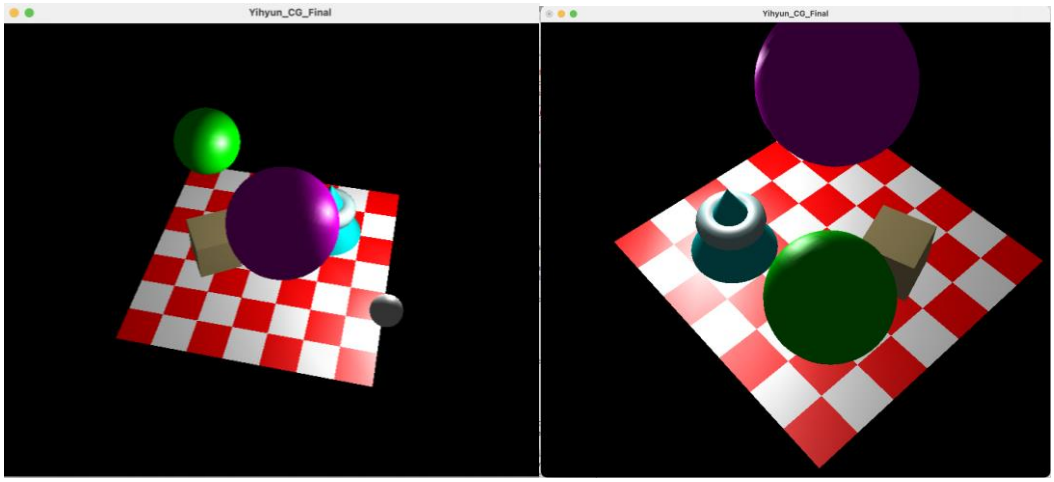
Computer Graphics

Final Project

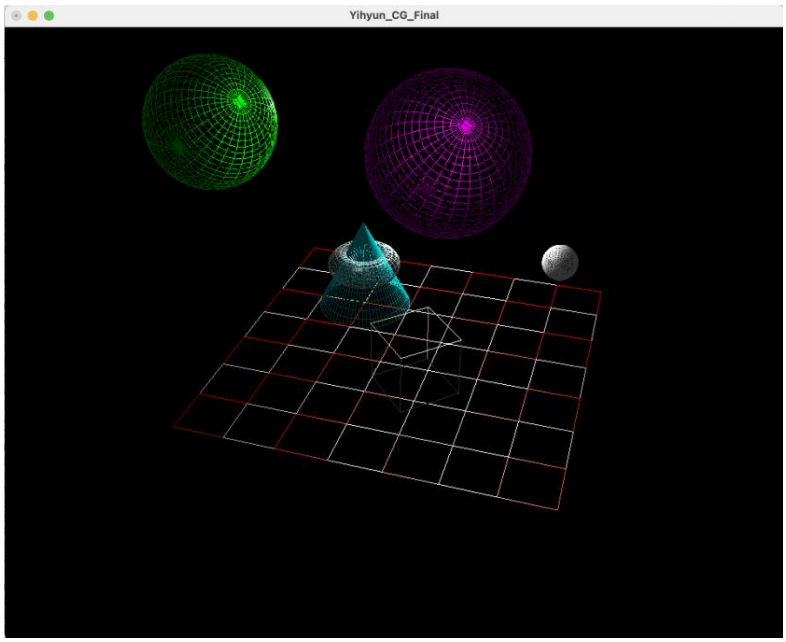
20194308 김이현

목차

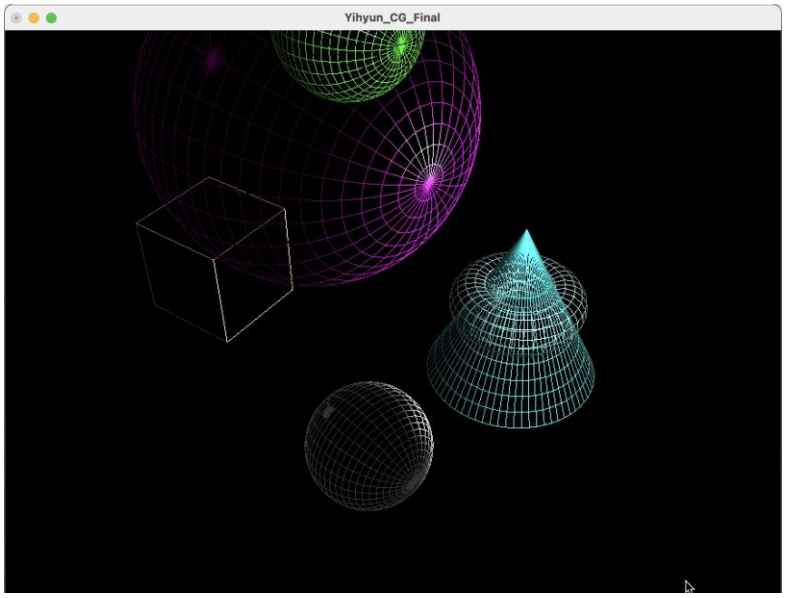
1. 프로그램 동작 과정



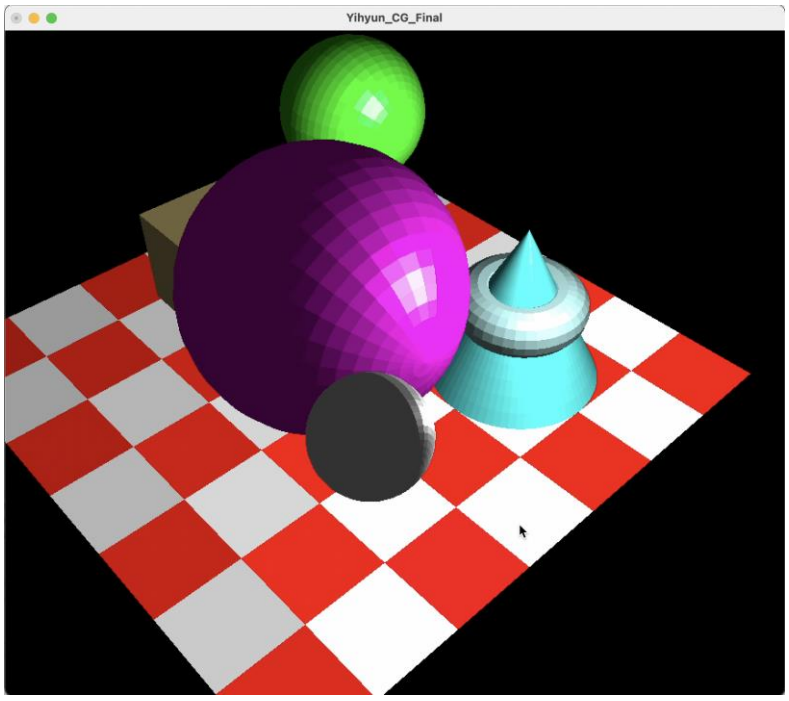
- 전체 Scene(원근 투영) / Lighting



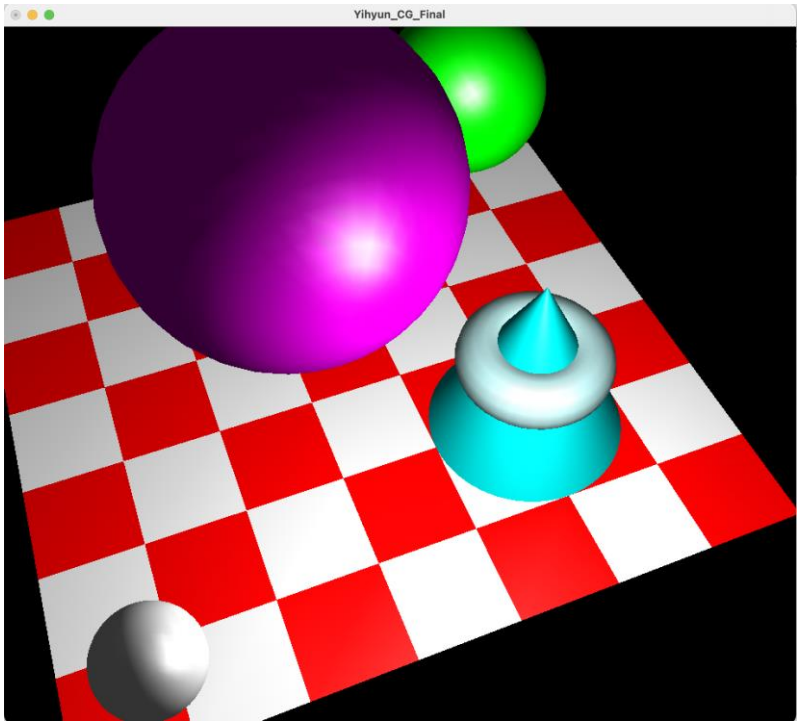
- Wireframe (Default value & Key 'w')



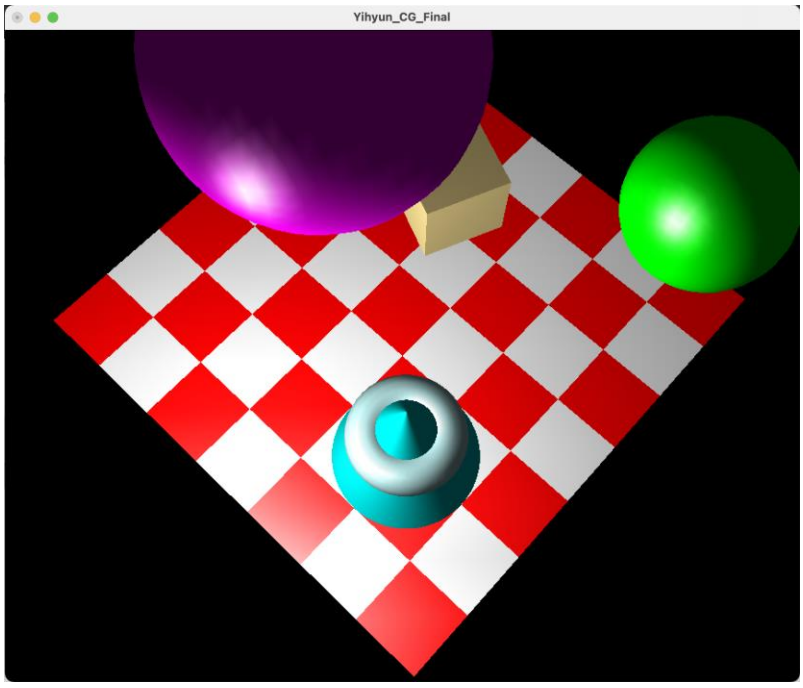
- Backface Culling (Mouse 좌클릭시 ON)



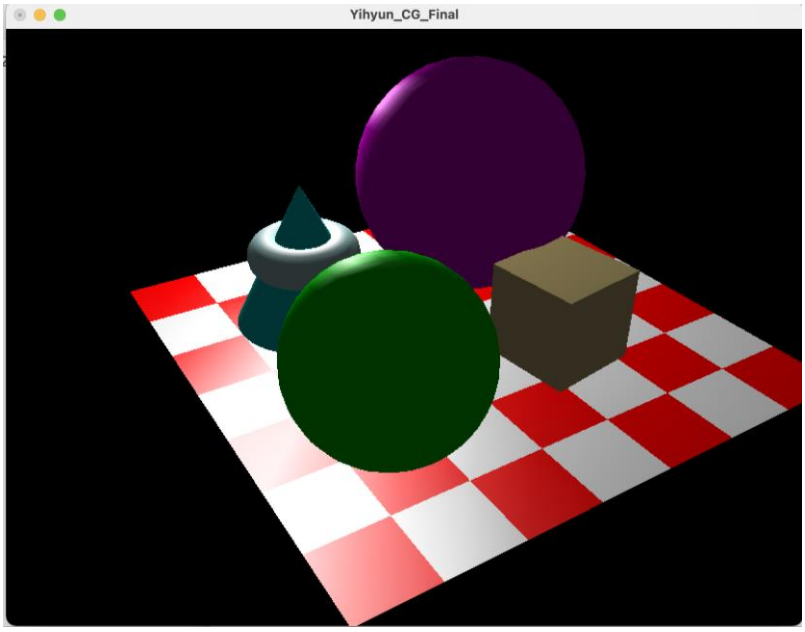
- Flat Shading (Mouse 우클릭시 ON)



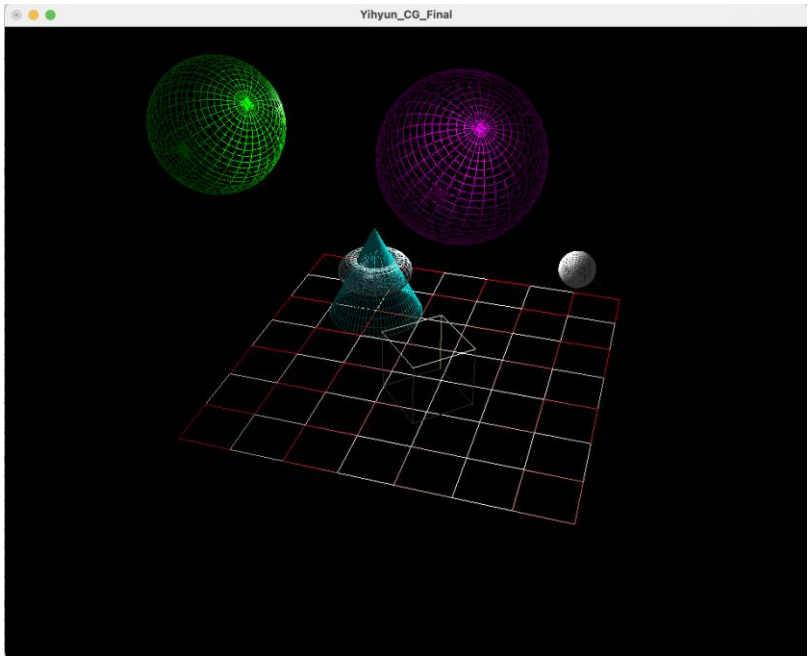
- Smooth Shading (Key 'S')



- Ball/Cube 등의 움직이는 objects 구현



- Texture Mapping / Lighting을 이용한 그림자



- 방향키를 이용한 카메라 시점 전환

2. 프로그램 구현 과정

- 전체 코드 및 각 함수의 기능 소개

```
#define _CRT_SECURE_NO_WARNINGS

#include <stdio.h>
#include <math.h>
#include <iostream>
#ifdef __APPLE_CC__
#include <GLUT/glut.h>
#else
#include <GL/glut.h>
#endif

#define WIRE 0
#define SHADE 1

#define PI 3.141592
#define R 10



using namespace std;

//큐브 – 회전 애니메이션을 위해 true로 설정
static bool spinning = true;
//큐브의 현재 방향을 추적하기 위해 정의
static GLfloat currentAngleOfRotation = 0.0;


typedef struct {
    float x;
    float y;
    float z;
} Point;

typedef struct {
    unsigned int ip[3];
} Face;

int pnum;
int fnum;

Point* mpoint = NULL; Face* mface = NULL;

GLfloat angle = 0;

int moving;
int mousebegin;
int light_moving;

int scaling = 0;
int status = 0; // WIRE or SHADE

```

- 프로그램에 사용될 변수, 구조체, 함수 선언 및 정의

```
// Colors
GLfloat WHITE[] = {1, 1, 1};
GLfloat RED[] = {1, 0, 0};
GLfloat GREEN[] = {0, 1, 0};
GLfloat MAGENTA[] = {1, 0, 1};
GLfloat Lemon[] = {1, 0.9, 0.6};
GLfloat CYAN[] = {0, 1, 1};
GLfloat lightCYAN[] = {0.8, 1, 1};

```

- 바닥, object에 입힐 컬러를 RGB 값의 조절을 통하여 지정

```
class Camera {
  double theta;       // x, z 위치
  double y;           // 현재의 y 위치
  double dTheta;      // 카메라 – 여러 방향으로 움직이기 위한 조절
  double dy;          // 카메라 – 상하 조절을 위한 조절
public:
  Camera(): theta(0), y(3), dTheta(0.04), dy(0.2) {}
  double getX() {return 10 * cos(theta);}
  double getY() {return y;}
  double getZ() {return 10 * sin(theta);}
  void moveRight() {theta += dTheta;} //카메라 우로 이동
  void moveLeft() {theta -= dTheta;} //카메라 좌로 이동
  void moveUp() {y += dy;} //카메라 높이 조절 – 상향
  void moveDown() {if (y > dy) y -= dy;} //카메라 높이 조절 – 하향
};
```

- Camera 클래스 정의
- MoveRight(), MoveDown() 등의 함수를 정의하여 카메라 이동 방향을 설정

```
class Ball {
  double radius;
  GLfloat* color;
  double maximumHeight;
  double x;
  double y;
  double z;
  int direction;

public:
  Ball(double r, GLfloat* c, double h, double x, double z):
      radius(r), color(c), maximumHeight(h), direction(-1),
      y(h), x(x), z(z) {
      }
  void update() {
    y += direction * 0.05;
    if (y > maximumHeight) {
      y = maximumHeight; direction = -1;
    } else if (y < radius) {
      y = radius; direction = 1;
    }
    glPushMatrix();
    glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, color);
    glTranslated(x, y, z);
    glutSolidSphere(radius, 30, 30);
    glPopMatrix();
  }
};
```

- Ball 클래스 정의
- Maximum height & xz plane 사이에서 bouncing하는 공들을 구현
- Update() 함수 : 한 프레임마다 0.05씩 up/down하는 애니메이션을 구현하기 위한 함수

```
class CheckFloor {
  int displayListId;
  int width;
  int depth;
public:
    CheckFloor(int width, int depth): width(width), depth(depth) {}
  double centerx() {return width / 2;}
  double centerz() {return depth / 2;}
  void create() {
    displayListId = glGenLists(1);
    glNewList(displayListId, GL_COMPILE);
    GLfloat lightPosition[] = {4, 3, 7, 1};
    glLightfv(GL_LIGHT0, GL_POSITION, lightPosition);
    glBegin(GL_QUADS);
    glNormal3d(0, 1, 0);
    for (int x = 0; x < width - 1; x++) {
      for (int z = 0; z < depth - 1; z++) {
        glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE,
                     (x + z) % 2 == 0 ? RED : WHITE);
        glVertex3d(x, 0, z);
        glVertex3d(x+1, 0, z);
        glVertex3d(x+1, 0, z+1);
        glVertex3d(x, 0, z+1);
      }
    }
    glEnd();
    glEndList();
  }
```

- CheckFloor 클래스 정의
- 1*1 사이즈의 square들로 xz plane 상에 RED&WHITE 색상의 바닥을 생성
- (4,3,7) 좌표에 Spotlight을 배치

```
void draw() {
    glCallList(displayListId);
  }
};

//카메라, 바닥, 공 -> 글로벌 변수로 선언
CheckFloor checkfloor(8, 8);
Camera camera;
Ball balls[] = {
  Ball(1, GREEN, 7, 6, 1),
  Ball(1.5, MAGENTA, 6, 3, 4),
  Ball(0.4, WHITE, 5, 1, 7)
};
```

Draw()함수 : displayListId값을 받아와 리스트 내에 정의된 사항을 그릴 수 있게 하는 함수

- 화면에 나타내야 할 바닥, 카메라, 공은 사용이 용이하도록 글로벌 변수로 선언

```
void init() {
  glEnable(GL_DEPTH_TEST);
  glLightfv(GL_LIGHT0, GL_DIFFUSE, WHITE);
  glLightfv(GL_LIGHT0, GL_SPECULAR, WHITE);
  glMaterialfv(GL_FRONT, GL_SPECULAR, WHITE);
  glMaterialf(GL_FRONT, GL_SHININESS, 30);
  glEnable(GL_LIGHTING);
  glEnable(GL_LIGHT0);
  checkfloor.create();
}
```

Init() 함수 : Lighting 효과를 위한 라obal parameter들을 세팅하고 checkfloor를 생성하는 함수

```
167 void mouse(int button, int state, int x, int y) {
168     if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN) {
169         glEnable(GL_CULL_FACE); glCullFace(GL_FRONT); glFrontFace(GL_CW);
170     } //왼쪽 마우스 버튼 누르면 backface culling ON
171     if (button == GLUT_LEFT_BUTTON && state == GLUT_UP) {
172         glDisable(GL_CULL_FACE);
173     } // 왼쪽 버튼 떼면 backface culling OFF
174     if ((button == GLUT_RIGHT_BUTTON && state == GLUT_DOWN)) {
175         // 오른쪽 버튼 누르면 FLAT shading
176           glShadeModel(GL_FLAT);
177     }
178     if ((button == GLUT_RIGHT_BUTTON && state == GLUT_UP)) {
179         // Default 값 & 오른쪽 버튼 떼면 SMOOTH shading
180           glShadeModel(GL_SMOOTH);
181     }
182 }
```

Mouse() 함수 : 마우스 좌클릭, 우클릭에 따른 기능을 구현하기 위한 함수

- line 168-9) Backface culling을 위한 기존 코드에서는 glCullFace의 parameter 값이 GL_BACK이지만,

  GL은 반시계방향으로 그려진 대상을 전면으로 인식하기 때문에,

  본 코드에서 Backface Culling을 구현하기 위해서는

  해당 parameter 값을 GL_FRONT로 설정해야 후면 제거가 가능

- 마우스 좌클릭) BackFace Culling 기능 ON
- 마우스 좌클릭 해제) BackFace Culling 기능 OFF
- 마우스 우클릭) Flat Shading 적용
- 마우스 우클릭 해제) Smooth Shading 적용

```
186 void display(void) {
187   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
188   if (status == WIRE)
189       glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
190   else if (status == SHADE)
191       glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
192 
193   glLoadIdentity();
194   gluLookAt(camera.getX(), camera.getY(), camera.getZ(),
195             checkfloor.centerx(), 0.0, checkfloor.centerz(),
196             0.0, 1.0, 0.0);
197     checkfloor.draw();
198   for (int i = 0; i < sizeof balls / sizeof(Ball); i++) {
199     balls[i].update();
200 }
201     glPushMatrix();
202     glTranslatef(5, 1, 5);
203     glRotatef(90.0, 1.0, 0.0, 0.0);
204     glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, lightCYAN);
205     glutSolidTorus(0.275, 0.5, 16, 40);
206     glPopMatrix();
207 
208     glPushMatrix();
209     glTranslatef(5, 0, 5);
210     glRotatef(270.0, 1.0, 0.0, 0.0);
211     glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, CYAN);
212     glutSolidCone(1.0, 2.0, 70, 12);
213 
214     glPushMatrix();
215     glTranslatef(-2, 3, 1);
216     glRotatef(currentAngleOfRotation, 0.0, 0.0, 1.0);
217     glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, Lemon);
218     glutSolidCube(1.2);
219     glPopMatrix();
220 
221     glFlush();
222     glutSwapBuffers();
223 }
```

Display() 함수 : 실제로 화면에 그리는 함수

- 현재 카메라 포지션 상에서 checkfloor -> balls 순서로 그림
- Special() 함수 상에서 w, s키로 변환되는 wireframe/shading의 State를 실질적으로 반영,
- 이를 위해 각각의 face는 앞/뒷면에 모두 적용하기 위해 GL_FRONT_AND_BACK을 사용,
- 각각의 모드는 GL_LINE / GL_FILL을 사용하여 그림
- GluLookAt()을 통해 카메라 상에서 어떻게 관찰 대상을 바라볼지 설정함
- glMaterialfv를 통해 면의 재질과 속성 등을 지정

- GlPushmatrix ~ GlPopmatrix 사이에 있는 부분이 화면에 그려지기 때문에

  도형마다 각각의 속성과 컬러, 사이즈 등을 개별 지정

- Torus) xz plane 상에 0.5만큼 떠있을 수 있도록 그렸으며,

  기본적인 glut lib의 torus는 수많은 원들이 x축을 에워싸는 방식으로 그려지기 때문에

  90도 회전을 통해 화면에 나타냄

- Cone) 기존과 달리 y축을 포인팅하기 위하여 x축 기준 270도 회전을 통해 화면에 나타냄

- Cube) 일정 각도로 회전하는 큐브 구현, 기존에 정의한 static 변수를 통해 방향을 추적함

```
234 void reshape(GLint w, GLint h) {
235   glViewport(0, 0, w, h);
236   GLfloat aspect = (GLfloat)w / (GLfloat)h;
237   glMatrixMode(GL_PROJECTION);
238   glLoadIdentity();
239   gluPerspective(40.0, GLfloat(w) / GLfloat(h), 1.0, 150.0);
240   glMatrixMode(GL_MODELVIEW);
241 }
```

Reshape() 함수 : 윈도우 크기에 변화가 있을 때의 변경사항을 반영해주는 함수

- glViewport를 통해 변화된 윈도우 사이즈로 설정
- glLoadIdentity를 통해 원하는 위치에 그릴 수 있도록 단위행렬로 초기화함
- gluPerspective를 통해 시야각과 종횡비, near/far 클리핑 평면의 거리 등을 이용한 원근투영

  (glFrustum과 같은 관측공간을 생성하더라도 gluPerspective가 보다 직관적)

```
227 void timer(int v) {
228   glutPostRedisplay();
229   glutTimerFunc(1000/60, timer, v);
230 
231     if (spinning) {
232         currentAngleOfRotation += 1.0;
233         if (currentAngleOfRotation > 360.0) {
234           currentAngleOfRotation -= 360.0;
235         } //회전하는 큐브를 그리는 코드
236 
237         glutPostRedisplay();
238       }
239 }
```

timer() 함수 : 일정한 시간마다 애니메이션을 동작시키기 위한 Callback 함수 구현

- glutPostRedisplay를 통해 애니메이션 수행 시에 윈도우를 다시 그려줌

- line 229)다음 타이머 이벤트는 1000/60ms 후에 호출됨

- line 231-235) 회전하는 큐브를 그리기 위한 각도를 계산

```
241 void normalKey(unsigned char key, int x, int y)
242 {
243     printf("key %d\n", key);
244     switch (key) {
245     case 'w':
246         status = WIRE;  glutPostRedisplay(); break;
247     case 's':
248         status = SHADE; glutPostRedisplay(); break;
249     glutPostRedisplay();
250 }
```

NormalKey() 함수 : 키보드 상의 키를 이용하여 부가적인 기능을 수행할 수 있게 하는 함수

- 'w' 누르기) Wireframe 모드 적용

- 's' 누르기) Shading 모드(Smooth shading) 모드 적용

- glutpostredisplay를 통해 변경사항을 화면에 반영

```
252 void special(int key, int, int) {
253     printf("key %d\n", key);
254     switch (key) {
255     case GLUT_KEY_LEFT: camera.moveLeft(); break;
256     case GLUT_KEY_RIGHT: camera.moveRight(); break;
257     case GLUT_KEY_UP: camera.moveUp(); break;
258     case GLUT_KEY_DOWN: camera.moveDown(); break;
259   }
260   glutPostRedisplay();
261 }
```

special() 함수 : 키보드 상의 키를 이용하여 카메라를 움직이는 함수

- 이동키 좌) 카메라 좌로 이동

- 이동키 우) 카메라 우로 이동

- 이동키 상) 카메라 위로 이동

- 이동키 하) 카메라 아래로 이동

- glutpostredisplay를 통해 변경사항을 화면에 반영

```
257 int main(int argc, char** argv) {
258   glutInit(&argc, argv);
259   glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
260   glutInitWindowPosition(80, 80);
261   glutInitWindowSize(800, 600);
262   glutCreateWindow("Yihyun_CG_Final");
263   glutDisplayFunc(display);
264   glutMouseFunc(mouse);
265   glutKeyboardFunc(normalKey);
266   glutReshapeFunc(reshape);
267   glutSpecialFunc(special);
268   glutTimerFunc(100, timer, 0);
269   init();
270   glutMainLoop();
271
272   return 0;
273 }
```

Main() 함수

- GlutInit을 통해 OS와 세션을 연결
- GlutInitDisplayMode를 통해 디스플레이 모드를 설정
- GlutInitWindowPosition을 통해 윈도우의 위치를, GlutInitWindowSize를 통해 윈도우 크기를 설정
- GlutDisplayFunc() : 화면에 그려질 사항들을 정의할 함수를 display() 함수로 설정
- display(), mouse(), reshape() => 콜백 함수로 작동

3. 부가 기능 소개

- 모든 object 및 바닥을 직접 제작, Color 지정
- 키보드 화살표를 이용한 공간 이동 (위/아래/왼쪽/오른쪽)
- 바닥 부분 Texture Mapping
- Lighting을 이용한 바닥 그림자 생성
- 가상공간에 움직이는 Object (위아래로 움직이는 공 / 회전하는 큐브) 구현