# 목차


1. 문제에 대한 분석 및 해결 방법 ..... 2

   실습 과제 04: GLUT 객체 그리기 및 여러 콜백 함수 테스트 ..... 2

2. 자신이 구현한 주요 코드 ..... 2

   실습 과제 04: GLUT 객체 그리기 및 여러 콜백 함수 테스트 ..... 2

3. 테스트 결과 ..... 5

   실습 과제 04: GLUT 객체 그리기 및 여러 콜백 함수 테스트 ..... 5

4. 느낀 점 ..... 5

5. 질문 및 건의사항 ..... **오류! 책갈피가 정의되어 있지 않습니다.**

## 1. 문제에 대한 분석 및 해결 방법

### 실습 과제 04: GLUT 객체 그리기 및 여러 콜백 함수 테스트

이번 과제는 수업 시간에 배웠던 여러 glut의 함수들을 이용하여, 기존 과제들을 한 번에 볼 수 있게 병합하는 것으로 파악했다. 따라서 요구사항에 맞게 필요한 glut의 유틸리티 함수들을 적재적소에 사용하고, 기존 과제의 구현 함수들을 호출하여 과제를 해결할 수 있었다.

## 2. 자신이 구현한 주요 코드

### 실습 과제 04: GLUT 객체 그리기 및 여러 콜백 함수 테스트

```
enum Menu {
    MENU_WIRE_CUBE,
    MENU_WIRE_SPHERE,
    MENU_WIRE_CONE,
    MENU_WIRE_TORUS,
    MENU_WIRE_TETRAHEDRON,
    MENU_WIRE_ICOSAHEDRON,
    MENU_WIRE_TEAPOT,
    MENU_MY_NAME,
    MENU_SIERPINSKI_GASKET,
    MENU_EXIT
} typedef Menu;
```

메뉴 처리를 위한 열거형 선언

```
/**
 * 메뉴 변경시 호출되는 콜백
 * @param id 메뉴 ID
 */
void handleMenu(int id) {
    currentMenu = (Menu) id;
    if (currentMenu == MENU_EXIT) {
        exit(0);
    } else {
        isAutoUpdateMode = false;
    }

    glutPostRedisplay();
}

/**
 * 메뉴 초기화 함수
 */
void initMenu() {
    GLint MyMainMenuID = glutCreateMenu(handleMenu);
    glutAddMenuEntry("Draw WireCube", MENU_WIRE_CUBE);
    glutAddMenuEntry("Draw WireSphere", MENU_WIRE_SPHERE);
    glutAddMenuEntry("Draw WireCone", MENU_WIRE_CONE);
    glutAddMenuEntry("Draw WireTorus", MENU_WIRE_TORUS);
    glutAddMenuEntry("Draw WireTetrahedron", MENU_WIRE_TETRAHEDRON);
    glutAddMenuEntry("Draw WireTeapot", MENU_WIRE_TEAPOT);
    glutAddMenuEntry("Draw MyName", MENU_MY_NAME);
    glutAddMenuEntry("Draw SierpinskiGasket", MENU_SIERPINSKI_GASKET);
    glutAddMenuEntry("Exit", MENU_EXIT);
    glutAttachMenu(GLUT_RIGHT_BUTTON);
}
```

메뉴 핸들러 및 메뉴 초기화

```
/**
 * 주어진 메뉴에 맞는 도형을 그리는 함수
 */
void draw() {
    switch (currentMenu) {
        case MENU_WIRE_CUBE:
            glutWireCube(1.0);
            break;
        case MENU_WIRE_SPHERE:
            glutWireSphere(0.9, 20, 20);
            break;
        case MENU_WIRE_CONE:
            glutWireCone(1.0, 1.0, 20, 20);
            break;
        case MENU_WIRE_TORUS:
            glutWireTorus(0.1, 0.75, 20, 20);
            break;
        case MENU_WIRE_TETRAHEDRON:
            glutWireTetrahedron();
            break;
        case MENU_WIRE_ICOSAHEDRON:
            glutWireIcosahedron();
            break;
        case MENU_WIRE_TEAPOT:
            glutWireTeapot(0.5);
            break;
        case MENU_MY_NAME:
            drawMyName();
            break;
        case MENU_SIERPINSKI_GASKET:
            drawSierpinski({0, -0.2}, 1.75);
            break;
    }
}
```

실제로 도형을 그리는 함수 선언

```
/**
 * 창 크기 변경시 호출되는 콜백
 */
void reshape(int width, int height) {
    glViewport(0, 0, width, height);

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();

    glOrtho(-width / (double) SCREEN_SIZE,
            width / (double) SCREEN_SIZE,
            -height / (double) SCREEN_SIZE,
            height / (double) SCREEN_SIZE, -1, 1);

    glMatrixMode(GL_MODELVIEW);

    glutPostRedisplay();
}
```

창 크기 변경시 크기가 유지되도록 구현한 함수

```
/**
 * 마우스 클릭시 호출되는 콜백
 * @param button 클릭한 버튼
 * @param state 상태
 * @param x 가로 좌표
 * @param y 세로 좌표
 */
void mouseClick(int button, int state, int x, int y) {
    if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN) {
        prevLocation = {x, y};
    }
}
```

```
/**
 * 마우스 모션시 호출되는 콜백
 * @param x 가로 좌표
 * @param y 세로 좌표
 */
void mouseMotion(GLint x, GLint y) {
    glRotated(x - prevLocation.x, 0, 1, 0);
    glRotated(y - prevLocation.y, 1, 0, 0);
    prevLocation = {x, y};
}
```

마우스 상호작용 처리 함수

```
/**
 * 키보드 사용시 호출되는 콜백
 * @param key 누른 키
 * @param x 마우스 가로 좌표
 * @param y 마우스 세로 좌표
 */
void keyboard(unsigned char key, int x, int y) {
    if ('0' <= key && key < '0' + MENU_EXIT) {
        isAutoUpdateMode = false;
        currentMenu = (Menu) (key - '0');
    } else if (key == 'a') {
        isAutoUpdateMode = !isAutoUpdateMode;
    } else if (key == 'i') {
        glLoadIdentity();
    } else if (key == 'q') {
        exit(0);
    }
}
```

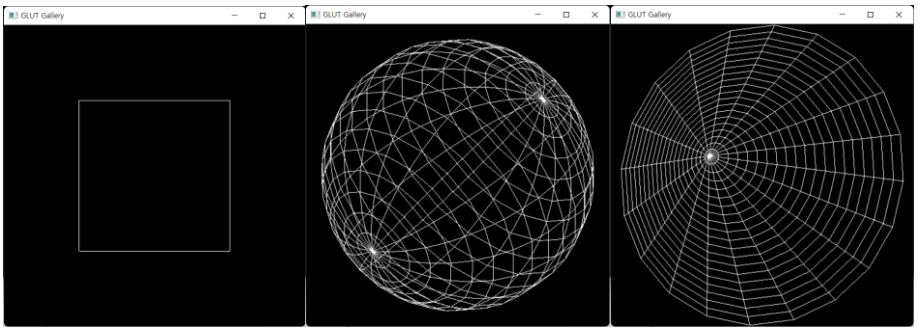
키보드 상호작용 처리 함수

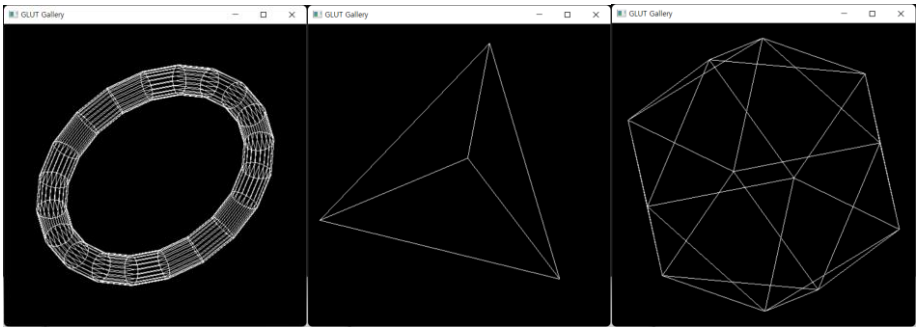
```
/**
 * 자동 모드인 경우, 0.5초마다 메뉴 상태를 변경하는 타이머 콜백
 * @param value 타이머 값
 */
void updateTimer(int value) {
    if (isAutoUpdateMode) {
        currentMenu = (Menu) ((currentMenu + 1) % MENU_EXIT);
    }
    glutTimerFunc(AUTO_UPDATE_TICK, updateTimer, 0);
}

/**
 * 화면 갱신 및 그라데이션 효과를 적용하는 타이머 콜백
 * @param value 타이머 값
 */
void drawTimer(int value) {
    if (currentMenu == MENU_MY_NAME) {
        applyGradation();
    } else {
        glColor3f(1.0f, 1.0f, 1.0f);
    }
    glutPostRedisplay();
    glutTimerFunc(DRAW_TICK, drawTimer, 0);
}
```

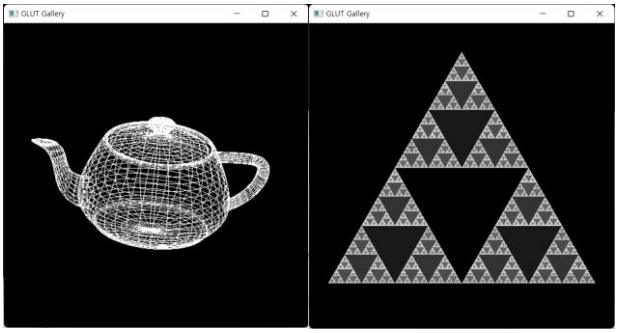
자동 넘기기 처리 및 화면 갱신을 위한 타이머 함수

## 3. 테스트 결과

### 실습 과제 04: GLUT 객체 그리기 및 여러 콜백 함수 테스트








## 4. 느낀 점

GLUT의 여러 함수를 이용하여, 주어진 과제의 요구사항에 맞는 프로그램을 만들어보니까, GLUT를 이용한 프로그래밍에 조금 더 자신감이 생겼다. 앞으로는 조금 더 복잡한 요구사항의 OpenGL 프로그램도 만들어보고 싶다는 생각이 들었다.