

컴퓨터그래픽스및실습

실습과제 03



과목명/분반	컴퓨터그래픽스및실습/ 01	담당교수	최영규 교수님
학 과 명	컴퓨터공학부	제 출 일	2023 / 09 / 23
학번	2019136098	이름	이우진

3.1

실습 3.4 : 자신의 이름 그리기

각 문제에 대한 분석과 자신이 해결한 방법 설명

우선 이름을 도형으로 표현을 더 편하게 하기 위해 성, 이름1, 이름2를 함수로 설정해서 그 안에 코드를 넣었습니다. 각각 First_Name, Last_Name_1, Last_Name_2로 설정했습니다.

First_Name(이)에는 원 도형을 2개 만들어서 큰 원은 배경과 다른 색, 작은 원은 배경과 같은 색으로 해서 **o**을 표현했습니다. 그리고 **l**는 직사각형 도형으로 만들었습니다.

Last_Name_1(우)에는 First_Name 때와 마찬가지로 원 도형을 2개 만들어서 큰 원은 배경과 다른 색, 작은 원은 배경과 같은 색으로 해서 **o**을 표현했습니다. 그리고 **ㅜ**는 직사각형 2개를 만들어서 표현했습니다.

Last_Name_2(진)에는 우선 **ㅈ**을 만들 때 직사각형 도형으로 **ㅡ**를 만들고 삼각형 도형을 2개 만들어서 큰 삼각형은 배경과 다른 색, 작은 삼각형은 배경과 같은 색으로 해서 **ㅈ**을 표현했습니다. **ㅣ**는 직사각형 도형으로 만들었습니다. 그리고 **ㄴ**은 직사각형 2개를 만들어서 큰 직사각형은 배경과 다른 색, 작은 직사각형은 배경과 같은 색으로 만들어서 **ㄴ**을 표현했습니다.

자신이 구현한 주요 코드 및 코드 설명

이름 이우진은 **o**, **l**, **ㅈ**, **ㄴ**으로 조합할 수 있어서 주요 코드로 4글자 도형을 만든 코드로 설정했습니다. 코드 설명은 주석으로 했으며 주석 포함 주요 코드를 입력했습니다.

```
< o >
double circleX = -0.8;
double rad = 0.2;
double rad2 = 0.1;
...
glColor3f(1.0, 1.0, 1.0); // 색은 하얀색
glBegin(GL_POLYGON); // 다각형 그리기
for (int i = 0; i < 360; i++) {
    double angle = i * 3.141592 / 180;
    double x = rad * cos(angle) + circleX; // circleX를 통해 원의 x위치 조절,
    double y = rad * sin(angle);
    // 반지름 * 코사인(angle)을 x값, 반지름 * 사인(angle)을 y값으로 해서 선으로 원을
    채웁니다.
    // i값 0 ~ 360으로 angle 구합니다. 예로 i = 0일때 angle은 pi / 180이고
    // 이를 x 값을 구하면 rad가 2일 때 2 * cos(pi / 180) 해서 근사값 2
    // y 값을 구하면 근사값 0으로 해서 도형을 채웁니다.
    glVertex3f(x, y, 0.0);
}
glEnd();

glColor3f(0.0, 0.0, 0.0);
glBegin(GL_POLYGON);
for (int i = 0; i < 360; i++) {
    double angle = i * 3.141592 / 180;
```

```

        double x = rad2 * cos(angle) + circleX;
        double y = rad2 * sin(angle);
        glVertex3f(x, y, 0.0);
    }
    glEnd();
    // 위에 방식과 같지만 반지름을 작게해서 도형 원을 작게 만들고 이를 같은 곳에 둔 후
    // 색을 배경색과 같게 한다음 위치하면 가운데가 비어있는 진짜 o상태가 됩니다.
< | >
    glColor3f(1.0, 1.0, 1.0); // 색은 하얀색
    glBegin(GL_POLYGON);
    glVertex3f(-0.5, -0.4, 0.0); // 왼쪽 아래
    glVertex3f(-0.4, -0.4, 0.0); // 오른쪽 아래
    glVertex3f(-0.4, 0.4, 0.0); // 오른쪽 위
    glVertex3f(-0.5, 0.4, 0.0); // 왼쪽 위
    glEnd();
< ㅈ >
    glColor3f(1.0, 1.0, 1.0); // 색은 하얀색
    glBegin(GL_POLYGON);
    glVertex3f(0.4, 0.5, 0.0);
    glVertex3f(0.8, 0.5, 0.0);
    glVertex3f(0.8, 0.4, 0.0);
    glVertex3f(0.4, 0.4, 0.0);
    glEnd();
    // 우선 ㅈ의 —는 기존 직사각형을 만드는 방법과 같습니다.

    glColor3f(1.0, 1.0, 1.0);
    glBegin(GL_POLYGON);
    glVertex3f(0.4, 0.0, 0.0); // 삼각형 왼쪽 아래
    glVertex3f(0.8, 0.0, 0.0); // 삼각형 오른쪽 아래
    glVertex3f(0.6, 0.5, 0.0); // 삼각형 위 중간
    glEnd();
    // 이렇게 삼각형을 먼저 만들었습니다.

    glColor3f(0.0, 0.0, 0.0); // 색은 검은색
    glBegin(GL_POLYGON);
    glVertex3f(0.5, 0.0, 0.0);
    glVertex3f(0.7, 0.0, 0.0);
    glVertex3f(0.6, 0.25, 0.0);
    glEnd();
    // 그 후 같은 방식으로 작은 삼각형을 만들고
    // 원을 만든 방식처럼 큰 삼각형은 하얀색, 작은 삼각형은 배경과 같은 검은색으로 해서
    // ㅈ의 ㅅ을 만들었습니다.
    // 좌표를 직사각형의 위쪽 중간과 큰 삼각형의 가장 높은 점을 같은 위치로 해서

```

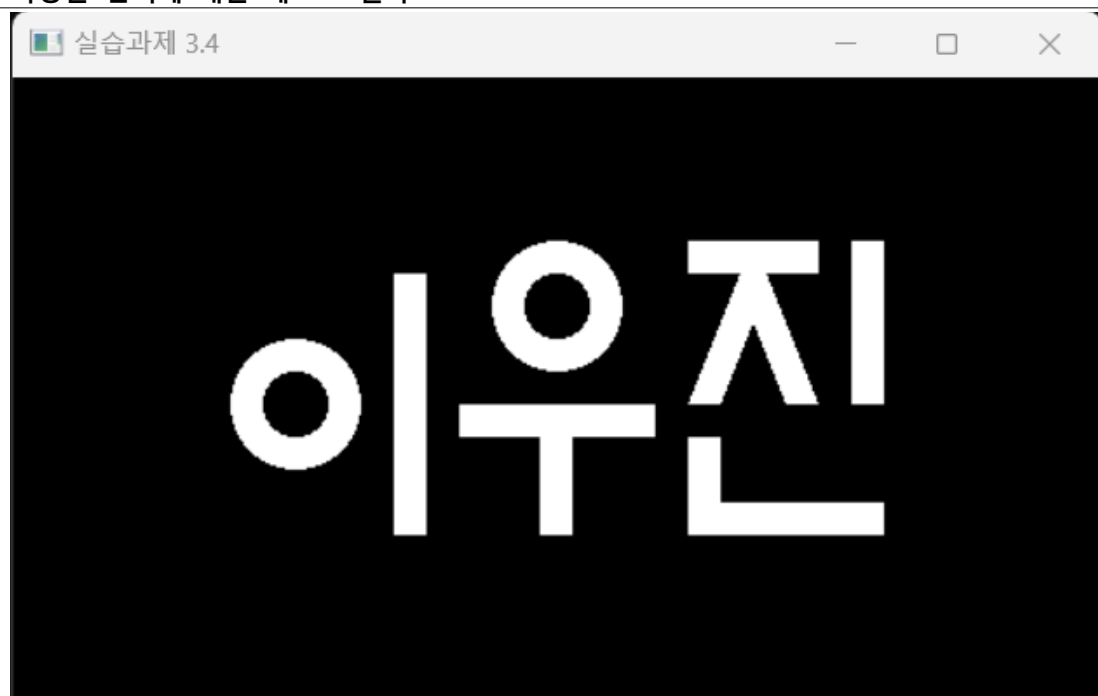
```

// ㅈ의 모양을 조정했습니다.
<┐>
glColor3f(1.0, 1.0, 1.0); // 색은 하얀색
glBegin(GL_POLYGON);
glVertex3f(0.4, -0.4, 0.0);
glVertex3f(0.4, -0.1, 0.0);
glVertex3f(1.0, -0.1, 0.0);
glVertex3f(1.0, -0.4, 0.0);
glEnd();
// 큰 직사각형을 만들었습니다.

glColor3f(0.0, 0.0, 0.0); // 색은 검은색
glBegin(GL_POLYGON);
glVertex3f(0.5, -0.3, 0.0);
glVertex3f(0.5, -0.1, 0.0);
glVertex3f(1.0, -0.1, 0.0);
glVertex3f(1.0, -0.3, 0.0);
glEnd();
// 작은 사각형을 만들었습니다.
// ㅇ, ㅈ 때 처럼 작은 삼각형을 배경색과 같은 색으로 해서 ㄴ을 표현했습니다.

```

다양한 입력에 대한 테스트 결과



해당 문제에 대한 고찰

처음에 어떻게하면 도형을 이동시킬 수 있을까 고민을 했고, 이동을 했을 때 글씨가 잘리는 현상이 발생했었는데, 이유가 $(-1, -1) \sim (1, 1)$ 이 범위이기 때문에 그 이상을 넘어가면 잘린다는 것을 알게 되었습니다. 또한 이름을 적으려면 동그라미를 그려야하는데 동그라미를 그리는 법을 직각삼각형을 여러개 이어서 원 모양을 만들었고, 이를 어떻게

이용하면 \circ 을 만들 수 있을지 고민을 했고, 작은 원을 큰 원 안에 넣고 작은 원을 배경 색이랑 같게하면 \circ 을 만들어진다는 것을 깨닫게 되었습니다. 이러한 방식을 이용해서 \times 의 \swarrow 부분과 \searrow 을 만들었습니다.

난이도는 어렵지는 않았지만 어떻게 하면 한번에 한 화면에 나올수 있을지, 이름을 만들 수 있을지 고민하는 시간이 오래걸렸던 거 같습니다.

3.2

실습 3.5: Sierpinski Gasket

각 문제에 대한 분석과 자신이 해결한 방법 설명

Sierpinski Gasket 문제는 작은 삼각형의 꼭짓점이 큰 삼각형 한 변의 중점에 있어야 합니다. 그래서 삼각형을 분할할 때 큰 삼각형의 꼭짓점 2개의 중점을 작은 삼각형의 꼭짓점으로 계산을 해서 하였습니다. 이 반복을 처음에 분할 횟수를 정한 후 분할 횟수만큼 분할을 하여 Sierpinski Gasket을 구합니다.

자신이 구현한 주요 코드 및 코드 설명

```
GLfloat v[3][2] = { {-1.0, 0}, {1.0, 0}, {0.0, 1.73} };
// 2차원 배열 v 선언 후 이 좌표를 삼각형 꼭짓점으로 합니다.
// 삼각형의 좌표를 이렇게 설정한 이유는 한 변의 길이가 2인 정삼각형이고,
// 정삼각형의 높이를 구하는 공식인 "h = (루트3 / 2) * 한 변의 길이"를 이용하여
// 높이는 루트 3이라는 값을 구했기에 근사값인 1.73을 높이로 설정하였습니다.
int n = 6; // 분할 횟수

void triangle(GLfloat* a, GLfloat* b, GLfloat* c)
{
    glVertex2fv(a);
    glVertex2fv(b);
    glVertex2fv(c);
    // divide_triangle 함수에서 받아온 a, b, c 값을 받아와서
    // 받아온 값을 꼭짓점으로 하여 삼각형을 그립니다.
}

void divide_triangle(GLfloat* a, GLfloat* b, GLfloat* c, int m)
// 삼각형을 분할하는 함수, a, b, c는 삼각형의 꼭짓점 좌표, m 은 분할 횟수
// a = v[0], b = v[1], c = v[2], m = n
{
    GLfloat v0[2], v1[2], v2[2]; // 분할된 삼각형 꼭짓점 좌표를 저장하기 위해 선언
    int j; // 반복문에서 사용하기 위해 선언
    if (m > 0) // m > 0일 때 삼각형 분할
    {
        for (j = 0; j < 2; j++)
```

```

        v0[j] = (a[j] + b[j]) / 2;
    for (j = 0; j < 2; j++)
        v1[j] = (a[j] + c[j]) / 2;
    for (j = 0; j < 2; j++)
        v2[j] = (b[j] + c[j]) / 2;
    // 예시를 들어 설명하면
    // v0은 a와 b의 중점을 계산하여 분할할 삼각형의 꼭짓점 좌표 중 하나로 설정합니
다.

    // 같은 방식으로 v1, v2의 꼭짓점 좌표를 구합니다.

    divide_triangle(a, v0, v1, m - 1);
    divide_triangle(c, v1, v2, m - 1);
    divide_triangle(b, v2, v0, m - 1);
    // 위에서 구한 v0, v1, v2 값을 이용하여 divide_triangle 함수를 재귀적으로 호출합
니다.

    // 호출할 때 a,b,c와 v0, v1, v2로 구성된 하위 삼각형을 분할합니다.
    // m이 0이 될 때까지 과정을 반복하여 작은 삼각형으로 분할합니다.
}
else // m > 0이 아닐 때 분할하지 않고 삼각형을 그립니다.
    triangle(a, b, c); // m이 0이 되면 triangle함수를 호출하여 삼각형을 그립니다.
}

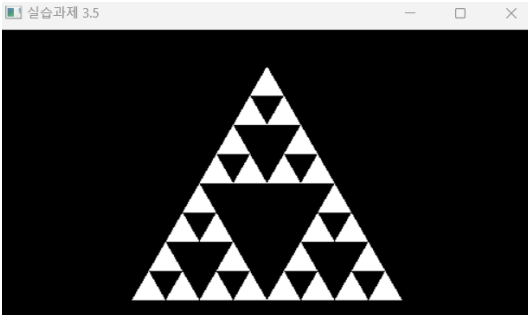
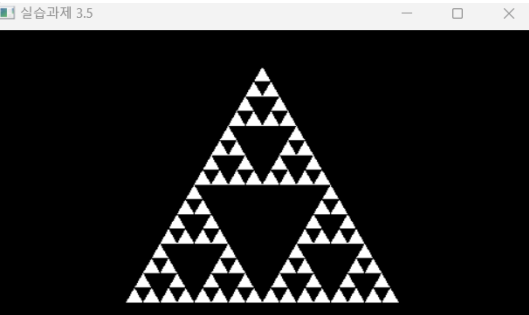
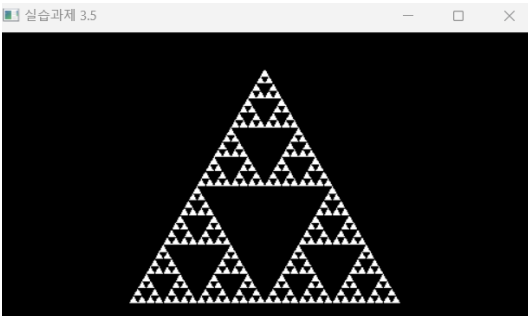
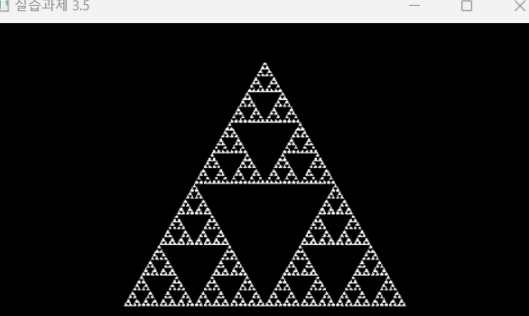
void display()
{
    ...

    divide_triangle(v[0], v[1], v[2], n);
    // 삼각형을 분할하고 그리기 위해 divide_triangle 함수를 호출합니다.
    ...
}

void main(int argc, char** argv) {
    ...

    glutDisplayFunc(display); // display함수를 호출하도록 설정합니다.
    ...
}

```

다양한 입력에 대한 테스트 결과	
<div>분할 횟수 = 3</div> 	<div>분할 횟수 = 4</div> 
<div>분할 횟수 = 5</div> 	<div>분할 횟수 = 6</div> 
해당 문제에 대한 고찰 이 과제를 통해 glut에 대해 좀 더 이해를 하게 된 시간이 된거 같습니다. 처음에는 막막했지만 단순히 glut 기능이 들어간 것일 뿐이지 c 계열 코딩을 하면 되는 것이여서 딱히 어렵지 않았던 거 같습니다.	

이번 과제에 대한 느낀점
처음 glut를 한다고 했을때는 자주 보던게 아니여서 헛갈리기도 했지만 많이 검색해보고 glut에서 사용하는 기능들을 찾아보고 하면서 조금이나마 익숙해진 시간이 되었던 거 같습니다.
궁금한 점이나 건의사항
궁금한 점이나 건의사항은 없습니다.
자신이 구현한 전체 코드 (3.1)
<pre> #include <GL/glut.h> #include <math.h> double circleX = -0.8; double circleY2 = 0.3; double rad = 0.2; double rad2 = 0.1; void First_Name(); void Last_Name_1(); </pre>

```

void Last_Name_2();

void MyDisplay() {
    glClear(GL_COLOR_BUFFER_BIT);
    glViewport(100, 0, 300, 300);

    First_Name();
    Last_Name_1();
    Last_Name_2();

    glFlush();
}

void First_Name() {
    glColor3f(1.0, 1.0, 1.0);
    glBegin(GL_POLYGON);
    for (int i = 0; i < 360; i++) {
        double angle = i * 3.141592 / 180;
        double x = rad * cos(angle) + circleX;
        double y = rad * sin(angle);
        glVertex3f(x, y, 0.0);
    }
    glEnd();

    glColor3f(0.0, 0.0, 0.0);
    glBegin(GL_POLYGON);
    for (int i = 0; i < 360; i++) {
        double angle = i * 3.141592 / 180;
        double x = rad2 * cos(angle) + circleX;
        double y = rad2 * sin(angle);
        glVertex3f(x, y, 0.0);
    }
    glEnd();

    glColor3f(1.0, 1.0, 1.0);
    glBegin(GL_POLYGON);
    glVertex3f(-0.5, -0.4, 0.0);
    glVertex3f(-0.4, -0.4, 0.0);
    glVertex3f(-0.4, 0.4, 0.0);
    glVertex3f(-0.5, 0.4, 0.0);
    glEnd();
}

```



```

void Last_Name_1() {
    glColor3f(1.0, 1.0, 1.0);
    glBegin(GL_POLYGON);
    for (int i = 0; i < 360; i++) {
        double angle = i * 3.141592 / 180;
        double x = rad * cos(angle);
        double y = rad * sin(angle) + circleY2;
        glVertex3f(x, y, 0.0);
    }
    glEnd();

    glColor3f(0.0, 0.0, 0.0);
    glBegin(GL_POLYGON);
    for (int i = 0; i < 360; i++) {
        double angle = i * 3.141592 / 180;
        double x = rad2 * cos(angle);
        double y = rad2 * sin(angle) + circleY2;
        glVertex3f(x, y, 0.0);
    }
    glEnd();

    glColor3f(1.0, 1.0, 1.0);
    glBegin(GL_POLYGON);
    glVertex3f(-0.3, 0.0, 0.0);
    glVertex3f(0.3, 0.0, 0.0);
    glVertex3f(0.3, -0.1, 0.0);
    glVertex3f(-0.3, -0.1, 0.0);
    glEnd();

    glColor3f(1.0, 1.0, 1.0);
    glBegin(GL_POLYGON);
    glVertex3f(-0.05, -0.4, 0.0);
    glVertex3f(0.05, -0.4, 0.0);
    glVertex3f(0.05, -0.1, 0.0);
    glVertex3f(-0.05, -0.1, 0.0);
    glEnd();
}

void Last_Name_2() {
    glColor3f(1.0, 1.0, 1.0);
    glBegin(GL_POLYGON);
    glVertex3f(0.4, 0.5, 0.0);
    glVertex3f(0.8, 0.5, 0.0);
}

```

```
glVertex3f(0.8, 0.4, 0.0);  
glVertex3f(0.4, 0.4, 0.0);  
glEnd();
```

```
glColor3f(1.0, 1.0, 1.0);  
glBegin(GL_POLYGON);  
glVertex3f(0.4, 0.0, 0.0);  
glVertex3f(0.8, 0.0, 0.0);  
glVertex3f(0.6, 0.5, 0.0);  
glEnd();
```

```
glColor3f(0.0, 0.0, 0.0);  
glBegin(GL_POLYGON);  
glVertex3f(0.5, 0.0, 0.0);  
glVertex3f(0.7, 0.0, 0.0);  
glVertex3f(0.6, 0.25, 0.0);  
glEnd();
```

```
glColor3f(1.0, 1.0, 1.0);  
glBegin(GL_POLYGON);  
glVertex3f(0.9, 0.0, 0.0);  
glVertex3f(1.0, 0.0, 0.0);  
glVertex3f(1.0, 0.5, 0.0);  
glVertex3f(0.9, 0.5, 0.0);  
glEnd();
```

```
glColor3f(1.0, 1.0, 1.0);  
glBegin(GL_POLYGON);  
glVertex3f(0.4, -0.4, 0.0);  
glVertex3f(0.4, -0.1, 0.0);  
glVertex3f(1.0, -0.1, 0.0);  
glVertex3f(1.0, -0.4, 0.0);  
glEnd();
```

```
glColor3f(0.0, 0.0, 0.0);  
glBegin(GL_POLYGON);  
glVertex3f(0.5, -0.3, 0.0);  
glVertex3f(0.5, -0.1, 0.0);  
glVertex3f(1.0, -0.1, 0.0);  
glVertex3f(1.0, -0.3, 0.0);  
glEnd();
```

```
}
```

```

int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_RGB);
    glutInitWindowSize(500, 300);
    glutInitWindowPosition(0, 0);
    glutCreateWindow("실습과제 3.4");
    glClearColor(0.0, 0.0, 0.0, 1.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-1.0, 1.0, -1.0, 1.0, -1.0, 1.0);
    glutDisplayFunc(MyDisplay);
    glutMainLoop();
    return 0;
}

```

자신이 구현한 전체 코드 (3.2)

```

#include <stdio.h>
#include <stdlib.h>
#include <GL/glut.h>

GLfloat v[3][2] = { {-1.0, 0}, {1.0, 0}, {0.0, 1.73} };
int n = 6;

void triangle(GLfloat* a, GLfloat* b, GLfloat* c)
{
    glVertex2fv(a);
    glVertex2fv(b);
    glVertex2fv(c);
}

void myinit()
{
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(-2.0, 2.0, -2.0, 2.0);
    glMatrixMode(GL_MODELVIEW);
    glClearColor(0.0, 0.0, 0.0, 1.0);
    glColor3f(1.0, 1.0, 1.0);
}

void divide_triangle(GLfloat* a, GLfloat* b, GLfloat* c, int m)
{
    GLfloat v0[2], v1[2], v2[2];

```

```

int j;
if (m > 0)
{
    for (j = 0; j < 2; j++)
        v0[j] = (a[j] + b[j]) / 2;
    for (j = 0; j < 2; j++)
        v1[j] = (a[j] + c[j]) / 2;
    for (j = 0; j < 2; j++)
        v2[j] = (b[j] + c[j]) / 2;

    divide_triangle(a, v0, v1, m - 1);
    divide_triangle(c, v1, v2, m - 1);
    divide_triangle(b, v2, v0, m - 1);
}
else
    triangle(a, b, c);
}

void display()
{
    glClear(GL_COLOR_BUFFER_BIT);
    glBegin(GL_TRIANGLES);
    divide_triangle(v[0], v[1], v[2], n);
    glEnd();
    glFlush();
}

void main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitWindowPosition(100, 100);
    glutInitWindowSize(500, 500);
    glutCreateWindow("실습과제 3.5");
    glutDisplayFunc(display);
    myinit();
    glutMainLoop();
}

```