목차

1. 문제에 대한 분석 및 해결 방법 2

   실습 3.1: 컴퓨터에 GLUT 설치하기 2

   실습 3.4: 자신의 이름 그리기 2

   실습 3.5: Sierpinski Gasket 2

2. 자신이 구현한 주요 코드 2

   실습 3.4: 자신의 이름 그리기 (convert.ipynb, main.cpp) 2

   실습 3.5: Sierpinski Gasket (main.cpp) 3

3. 테스트 결과 5

   실습 3.4: 자신의 이름 그리기 5

   실습 3.5: Sierpinski Gasket 5

4. 느낀 점 6

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

1. 문제에 대한 분석 및 해결 방법

실습 3.1: 컴퓨터에 GLUT 설치하기

온라인 강의를 참조하여 GLUT 라이브러리를 다운받고 Windows 환경에서 설치하였다.

실습 3.4: 자신의 이름 그리기

다양한 방법이 있겠지만, 직접 선을 하드코딩하여 그리기에는 내 이름의 "형"자가 너무 복잡하다는 것을 깨닫고, 다른 이미지 편집 툴에서 이미지를 생성하여 해결할 수 있겠다는 생각이 들었다. 실습 시간에, 모든 도형은 삼각형으로 표현할 수 있다는 조교님의 이야기가 생각나서 단색으로 이름이 적힌 이미지를 좌표 행렬로 표현하고, 들로네 삼각분할를 이용하여 삼각형 좌표의 배열로 만든 뒤 OpenGL 코드로 렌더링하려고 했으나, 관련 라이브러리 사용이 미숙하여 실패하였다. (옵션 설정 문제인지, 삼각형 메시로 글자가 원하는 대로 표현되지 않았음) 결국 파이썬을 이용하여, 내 이름이 적힌 이미지를 압축된 희소행렬 형태로 나타낸 뒤, OpenGL에서 사각형을 그리는 함수를 정의하고, 이를 이용하여 이미지를 렌더링하였다.

실습 3.5: Sierpinski Gasket

이전에 다른 프로그래밍 수업에서 프랙탈 도형을 그려본 경험에서 삼각형을 그리는 함수와 재귀함수 두개로 문제를 표현할 수 있음이 직감적으로 느껴졌다. 정삼각형을 그리는 함수와, 재귀함수를 이용한 시에르핀스키 삼각형을 그리는 함수 두개를 구현하여 시에르핀스키 삼각형을 렌더링하였다.

2. 자신이 구현한 주요 코드

실습 3.4: 자신의 이름 그리기 (convert.ipynb, main.cpp)

```
…
image = Image.open('name.png')
…
image_array = asarray(image)
compressed_array = []
for r in range(image.size[0]):
    for c in range(image.size[1]):
        pixel = image_array[r][c]
        if pixel[0] != 255:
            compressed_array.append([c, r])

image_array = np.array(compressed_array)
trimmed_image_array = image_array - np.array([[np.min(image_array[..., 0]),
np.min(image_array[..., 1])]])
raw_trimmed_image_array = trimmed_image_array.tolist()
print(len(raw_trimmed_image_array))
print(np.array([[np.max(trimmed_image_array[..., 0]),
np.max(trimmed_image_array[..., 1])]]))
```

```
print('{' + ','.join(list(map(lambda e: str(e).replace('[', '{').replace(']', '}'),
raw_trimmed_image_array))) + '}')
```

↑ 주어진 이미지를 압축된 희소행렬 형태로 변환한 뒤, C++ 스타일의 배열로 출력한다.

```
/**
 * 주어진 중심 좌표에 주어진 변 크기를 가지는 정사각형을 그려주는 함수
 * @param center 중심 좌표
 * @param edgeSize 변 크기
 */
void drawRect(Point2D center, float edgeSize) {
    edgeSize /= 2;
    Point2D v1 = {center.x - edgeSize, center.y - edgeSize};
    Point2D v2 = {center.x + edgeSize, center.y - edgeSize};
    Point2D v3 = {center.x + edgeSize, center.y + edgeSize};
    Point2D v4 = {center.x - edgeSize, center.y + edgeSize};

    glBegin(GL_QUADS);
    glVertexP2D(v1);
    glVertexP2D(v2);
    glVertexP2D(v3);
    glVertexP2D(v4);
    glEnd();
}
```

```
/**
 * 내 이름을 화면에 그리는 함수
 */
void drawMyName() {
    const float startX = -RECT_SIZE * NAME_WIDTH / 2;
    const float startY = RECT_SIZE * NAME_HEIGHT / 2;
    for (int i = 0; i < NAME_SIZE; i++) {
        drawRect({startX + RECT_SIZE * NAME[i][0], startY + -RECT_SIZE * NAME[i][1]},
RECT_SIZE);
    }
}
```

```
float color[3] = {0.1f, 0.3f, 0.1f};
bool increaseGreen = true;

/**
 * 색깔 그라데이션을 계산하여 적용하는 함수
 */
void applyGradation() {
    if (color[1] <= 0.3f) {
        increaseGreen = true;
    } else if (color[1] >= 1) {
        increaseGreen = false;
    }
    color[1] += (increaseGreen ? 1 : -1) * 0.01f;
    glColor3f(color[0], color[1], color[2]);
}
```

실습 3.5: Sierpinski Gasket (main.cpp)

```
/**
 * 주어진 중심 좌표에 주어진 변 크기를 가지는 정삼각형을 그리는 함수
 * @param center 중심 좌표
```

```
 * @param edgeSize 변 크기
 * @param inverted 뒤집힌 정삼각형을 그릴지 결정하는 플래그
 */
void drawEquilateralTriangle(Point2D center, float edgeSize, bool inverted = false)
{
    int invertedFactor = !inverted ? 1 : -1;
    float height = SQRT3 / 2 * edgeSize;
    Point2D v1 = {center.x, center.y + invertedFactor * height * 2 / 3};
    Point2D v2 = {center.x - edgeSize / 2, center.y + -invertedFactor * height /
3};
    Point2D v3 = {center.x + edgeSize / 2, center.y + -invertedFactor * height /
3};

    glBegin(GL_TRIANGLES);
    glVertexP2D(v1);
    glVertexP2D(v2);
    glVertexP2D(v3);
    glEnd();
}
```

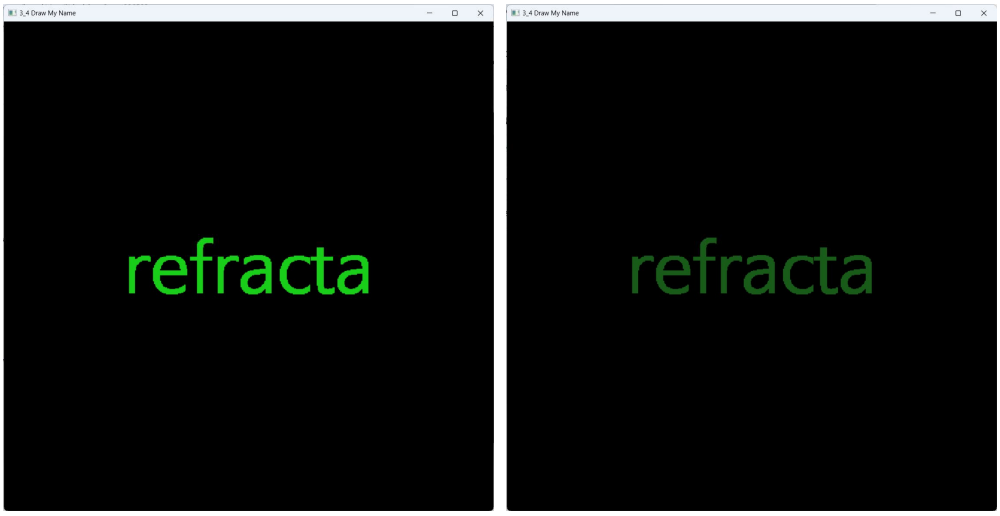
```
/**
 * 주어진 중심 좌표에 주어진 변 크기를 가지는 시에르핀스키 삼각형을 그리는 함수
 * @param center 중심 좌표
 * @param edgeSize 변 크기
 * @param limit 재귀 연산 한계
 * @param depth 재귀 연산의 깊이
 */
void drawSierpinski(Point2D center, float edgeSize, int limit = 7, int depth = 0)
{
    if (limit < depth)
        return;

    if (depth == 0) {
        glColor3f(1, 1, 1);
        drawEquilateralTriangle(center, edgeSize);
    }

    edgeSize /= 2;
    glColor3f(depth * 0.1f, depth * 0.1f, depth * 0.1f);
    drawEquilateralTriangle(center, edgeSize, true);
    drawSierpinski({center.x, center.y + (1 / SQRT3) * edgeSize}, edgeSize, limit,
depth + 1);
    drawSierpinski({center.x - edgeSize / 2, center.y - (1 / (2 * SQRT3)) * edgeSize},
edgeSize, limit, depth + 1);
    drawSierpinski({center.x + edgeSize / 2, center.y - (1 / (2 * SQRT3)) * edgeSize},
edgeSize, limit, depth + 1);
}
```

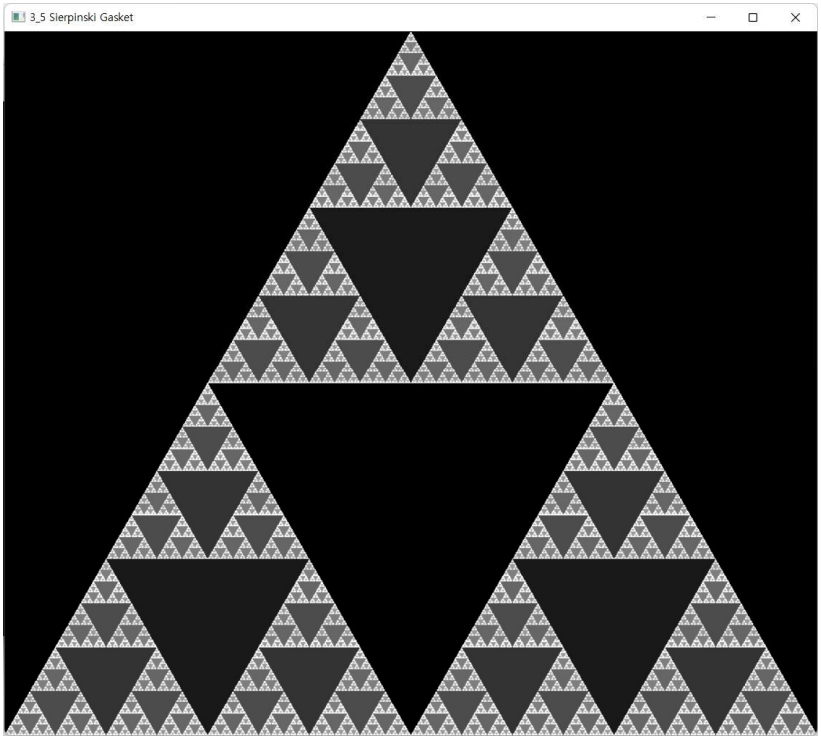
3. 테스트 결과

실습 3.4: 자신의 이름 그리기



시각적 효과가 하나 있었으면 좋겠다는 생각이 들어, 시간이 지날 때마다 초록색 계열 색깔로 그라데이션이 글씨에 적용되는 효과를 구현하였다.

실습 3.5: Sierpinski Gasket



시에르핀스키 삼각형을 그리는 과정 중 다음 단계의 삼각형을 그릴 때마다 더 옅은 색으로 삼각형이 그려지는 색깔 효과를 추가하였다.

4. 느낀 점

수업 시간에 배운 OpenGL 라이브러리 기능을 이용하여, 이름과 도형을 직접 출력해보니 익숙치 않았던 사용법이나 함수들을 사용하는 것에 자신감이 생겼다. 이번 과제에서는 정지된 2차원 물체만 렌더링 해보았는데 3차원 물체를 렌더링하거나, 움직이는 애니메이션을 구현할 수 있다면 더 재밌을 것 같다는 생각이 들었다.