Computer Graphics HW1

문제 1: Disc Triangulation

1.1 더 좋은 Triangulation 방식과 이유

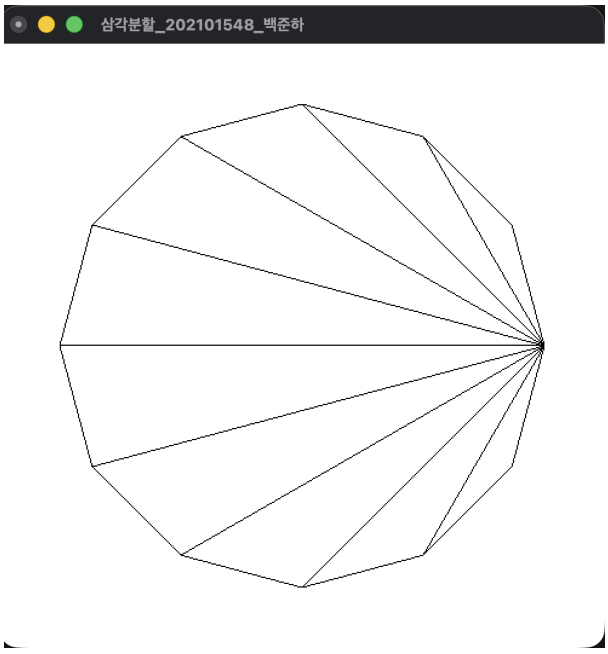
(a)와 (b) 중에 더 좋은 triangulation은 **(a)이다.**

일반적으로 정삼각형보다 길고 얇은 삼각형(long thin triangles)들이 픽셀 정확도 저하, 렌더링 성능 저하, 부동 소수점 오차 등에 더 많은 영향을 받기 때문이다. (a)의 삼각형들은 (b)보다 상대적으로 정삼각형에 가까우며 더 균일한 삼각형을 만들어 낸다.

1.2 (b) 방식 Triangulation 구현 및 설명

disc를 이루는 boundary vertex들의 세계 좌표와 `glOrtho` projection을 정하고, (b)와 같이 triangulation을 수행하여 disc 모양을 만든다. `glPolygonMode`를 설정하여 triangulation을 와이어프레임으로 확인했다.

렌더링 결과



Triangulation 및 Projection 수행 방식

1. 꼭짓점 좌표 계산 각 꼭짓점의 좌표는 반복문을 활용하여 공식 ***$x = r |cdot |cos(|theta)$, $y = r |cdot |sin(|theta)$*** 을 적용하여 구한다. 정12각형이므로 총 12번을 실행한다.

```
float vertices[numSegments][2];
   for (int i = 0; i < numSegments; ++i)
   {
       float angle = (30.0f * i * PI) / 180.0f;
       vertices[i][0] = centerX + radius * cos(angle);
```

```
        vertices[i][1] = centerY + radius * sin(angle);
    }
```

2. Triangulation 수행 `GL_TRIANGLE_FAN`을 사용하면 첫 번째 꼭짓점(중심점)을 기준으로 부채꼴처럼 삼각 분할을 수행할 수 있다. `GL_TRIANGLE_FAN`은 처음 입력된 중심점을 계속 활용하여 삼각형을 만들기 때문에 (P0, P1, P2) - (P0, P2, P3) - (P0, P3, P4) ... 와 같은 순서로 삼각형들이 생성된다.

문제 2: Triangulation 품질의 수치적 분석

2.1 `equilateralMetric` 함수를 이용한 수치적 분석

첨부한 코드의 `equilateralMetric` 함수를 사용하여 1번 문제의 두 triangulation (a), (b) 각각에 대해 모든 삼각형의 metric 값을 출력하고, 어떤 방식이 수치적으로 더 좋은 triangulation인지 분석한다.

`equilateralMetric` 함수는 삼각형이 정삼각형에 가까울수록 1에 근접한 값을 반환한다.

결과 분석

```
--- Triangulation Method (b) Metrics ---
Triangle (v0, v1, v2): 0.517638
Triangle (v0, v2, v3): 0.366025
Triangle (v0, v3, v4): 0.298858
Triangle (v0, v4, v5): 0.267949
Triangle (v0, v5, v6): 0.258819
Triangle (v0, v6, v7): 0.258819
Triangle (v0, v7, v8): 0.267949
Triangle (v0, v8, v9): 0.298858
Triangle (v0, v9, v10): 0.366025
Triangle (v0, v10, v11): 0.517636

--- Triangulation Method (a) Metrics ---
Triangle (center, v0, v1): 0.517638
Triangle (center, v1, v2): 0.517638
Triangle (center, v2, v3): 0.517638
Triangle (center, v3, v4): 0.517638
Triangle (center, v4, v5): 0.517638
Triangle (center, v5, v6): 0.517637
Triangle (center, v6, v7): 0.517638
Triangle (center, v7, v8): 0.517638
Triangle (center, v8, v9): 0.517638
Triangle (center, v9, v10): 0.517638
Triangle (center, v10, v11): 0.517638
Triangle (center, v11, v0): 0.517642
Program ended with exit code: 0
```

- **위 (b) 방식**: 균등하게 삼각분할을 해냈기에 일관된 수치를 보이지만, 1에 근접하지는 못한다. x축을 기준으로 대칭이므로 대응되는 삼각형끼리는 같은 `equilateralMetric` 함수값을 가진다. (예: 삼각형(v0,v1,v2) 와 (v0,v10,v11)은 합동이기에 함수값 또한 동일)
- **아래 (a) 방식**: (b) 방식보다 metric 값이 1에 더 가깝고 균일하므로, 수치적으로도 더 좋은 triangulation임을 알 수 있다.

2.2 완벽한 정삼각형 구성 및 증명

1의 (a) 방식을 사용하여 모든 삼각형이 완벽한 정삼각형이 되도록 정점들의 좌표를 설정하고, 이때 모든 삼각형의 metric 값이 1이 나옴을 출력하여 증명한다. 어떻게 좌표들을 정했는지 설명한다.

(a) 방식에서 모든 삼각형을 정삼각형으로 만들려면, 중심점과 두 경계 정점이 이루는 삼각형의 세 변의 길이가 모두 같아야 한다. 이는 중심점과 각 경계 정점 간의 거리(반지름)와 인접한 두 경계 정점 간의 거리가 모두 같아야 함을 의미한다.

이 조건을 만족하는 다각형은 **정육각형**이다.

따라서 12각형 대신 정육각형의 꼭짓점을 계산하여 `GL_TRIANGLE_FAN`으로 렌더링하면 6개의 완벽한 정삼각형이 생성된다. 아래 결과는 모든 metric 값이 1로 출력됨을 보여준다.

문제 3: Steiner Vertex를 이용한 Annulus Triangulation

3.1 Steiner Vertex란?

원래 객체를 구성하는 정점이 아니지만, Triangulation 품질을 높이기 위해(예: 길고 얇은 삼각형 방지) 추가하는 정점을 말한다. 객체 내부 또는 경계선 위에 위치할 수 있다.

3.2 요구사항

- 12개의 정점으로 구성된 Annulus(고리) 모양의 도형을 2개의 Steiner Vertex를 추가하여 총 14개의 정점으로 Triangulation 한다.
- 길고 얇은 삼각형이 최소화되도록 Steiner Vertex의 위치를 정하고 그 이유를 설명한다.
- `glPolygonMode`를 설정하여 Triangulation 결과를 확인한다.
- 모든 삼각형의 Orientation(방향성)이 동일한지 확인한다.

3.3 해결 전략 및 구현

3.3.1 Steiner Vertex 위치 선정

- **선정 위치**: `(50, 70)`과 `(50, 10)`
- **선정 이유**: Annulus 도형의 대칭성을 고려하여, 위쪽과 아래쪽 경계선의 중앙에 각각 Steiner Vertex를 추가했다. 이 두 점을 기준으로 도형을 분할하면, 각 부분의 Triangulation 시 생성되는 삼각형들이 비교적 정삼각형에 가까운 균일한 형태를 가지게 되어 길고 얇은 삼각형의 생성을 최소화할 수 있다.

3.3.2 Triangulation 방법

14개의 정점(기존 12개 + Steiner 2개)을 사용하여 도형 전체를 16개의 삼각형으로 분할했다. 모든 삼각형은 반시계 방향으로 정점을 정의하여 `GL_FRONT` 방향성을 갖도록 했다.

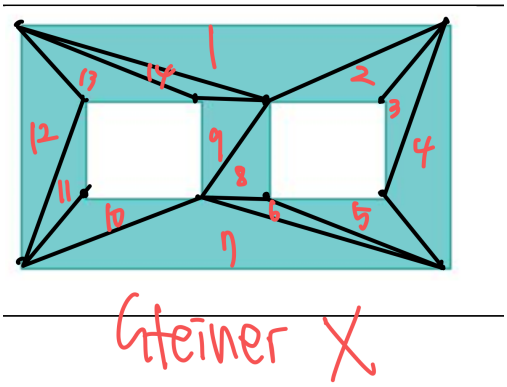
3.4 Triangulation 결과 비교

코드 주석을 통해 알 수 있지만, steiner vertex 추가 전 코드와 추가 후 코드에 같은 번호의 삼각형은 합동이거나 아래 코드의 경우가 더 정삼각형에 상대적으로 근접한다. **(구분을 쉽게 하기 위해서 합동인 삼각형들은 동일한 색깔을 채워주거나, 비슷한 위치의 삼각형은 회색과 검은색으로 칠해주었다.)**

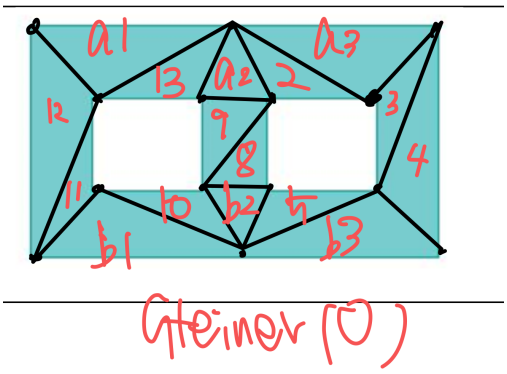
예를 들어:

1. 두 코드의 3번과 4번 삼각형은 합동이다.
2. (2번, a1, a2, a3, 13번) 삼각형은 위 코드의 (1번, 2번, 13번, 14번) 삼각형에서 steiner vertex 추가를 통해 바뀐 것으로 볼 수 있으며, steiner vertex 추가 후 **상대적으로** 모두 각각 정삼각형에 근접해간다.

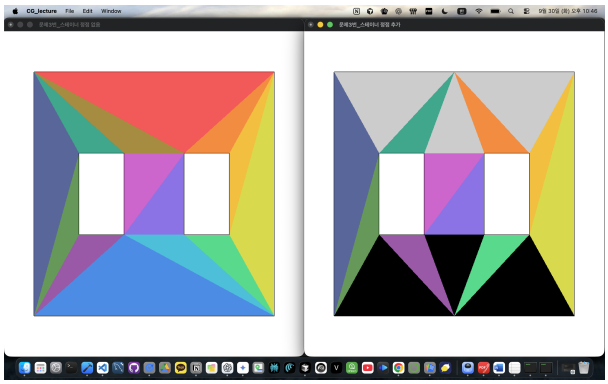
3.4.1 Steiner vertex 적용 전 (상상도)



3.4.2 Steiner vertex 적용 후 (상상도)



3.4.3 실제 렌더링 결과



위 사진은 실제 정점을 반시계 방향으로 설정하여 최대한 같은 구역을 합동이면서 같은 색깔인 삼각형들로 구성하고, steiner vertex를 추가한 영역의 삼각형들은 색깔이 회색 혹은 검정색으로 칠하여 전후 비교를 쉽게 하기 위해 설정하였다.

예를 들어 왼쪽 끝과 오른쪽은 합동인 삼각형들로 구성 및 색칠하였고, 커다란 빨간 삼각형은 회색 부분과, 커다란 파란 삼각형은 검정색 삼각형들과 대응된다고 이해하면 될 것이다.