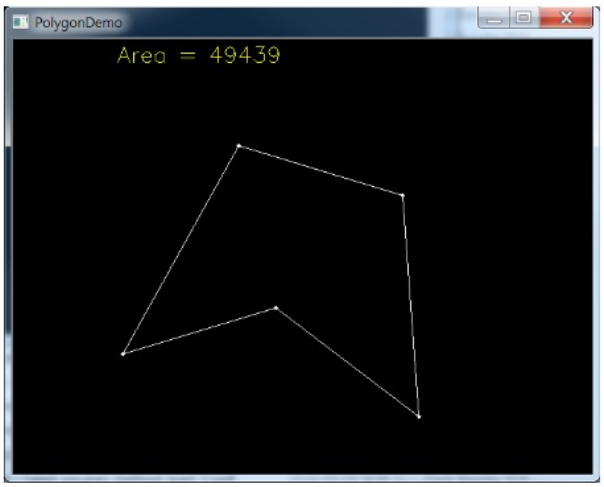
HW #4 벡터를 이용한 기하학 계산 (프로그래밍)

GitHub Link

02221081 황지현

- Q1. 사용자로부터 n각형의 꼭지점의 좌표를 입력받아 이 다각형의 면적을 출력하는 프로그램을 작성하시오
  - 제공되는 GitHub 템플릿 코드 활용 (cpp_template)
  - 오목, 볼록, 교차인 경우 정상 동작 여부 확인
  - 리포트(Slack) + 구현코드(GitHub 링크) 제출



* 리포트 포함 내용: 실행화면 캡처
* 계산 결과가 맞는지 스스로 검증

Q1-1. 입력한 도형의 좌표 (5 각형)

```
Adding point #0 with position(437,317)
Adding point #1 with position(270,90)
Adding point #2 with position(129,128)
Adding point #3 with position(222,219)
Adding point #4 with position(158,338)
Complete polygon with 4 points
Same point input
Completing polygon with 5 points.
points:[(437, 317), (270, 90), (129, 128), (222, 219), (158, 338)]
```

Q1-2. 도형의 면적 계산 (sheolace fomula)

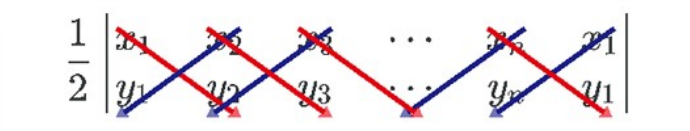
사선식, 신발끈 공식(sheolace formula)

$X_1(x_1, y_1)$, $X_2(x_2, y_2)$, $X_3(x_3, y_3)$가 좌표평면 위의 서로 다른 세 점, $\triangle X_1X_2X_3$의 넓이를 S라고 하자. 그러면 $S=\frac{1}{2}\left|(x_1y_2+x_2y_3+x_3y_1)-(y_1x_2+y_2x_3+y_3x_1)\right|$ 이다. 이를 일반화하면 다음과 같다.

$n \geq 3$에 대하여 좌표평면 위에 놓인 n각형의 꼭짓점들을 반시계 방향(혹은 시계 방향) 순서대로 $X_1(x_1, y_1)$, $X_2(x_2, y_2)$, $X_3(x_3, y_3)$, $\cdots$, $X_n(x_n, y_n)$라고 하자. 이 n각형의 넓이를 S라고 하면 다음과 같다.

$$S=\frac{1}{2}\left|(x_1y_2+x_2y_3+\cdots+x_{n-1}y_n+x_ny_1)-(y_1x_2+y_2x_3+\cdots+y_{n-1}x_n+y_nx_1)\right|$$

위 공식은 아래 이미지에서 붉은 화살표 방향으로는 곱해서 더하고, 푸른 화살표 방향으로는 곱해서 뺀 값에 절댓값을 취한 뒤 $\frac{1}{2}$를 곱한 것으로 기억하면 사용하기 편리하다.



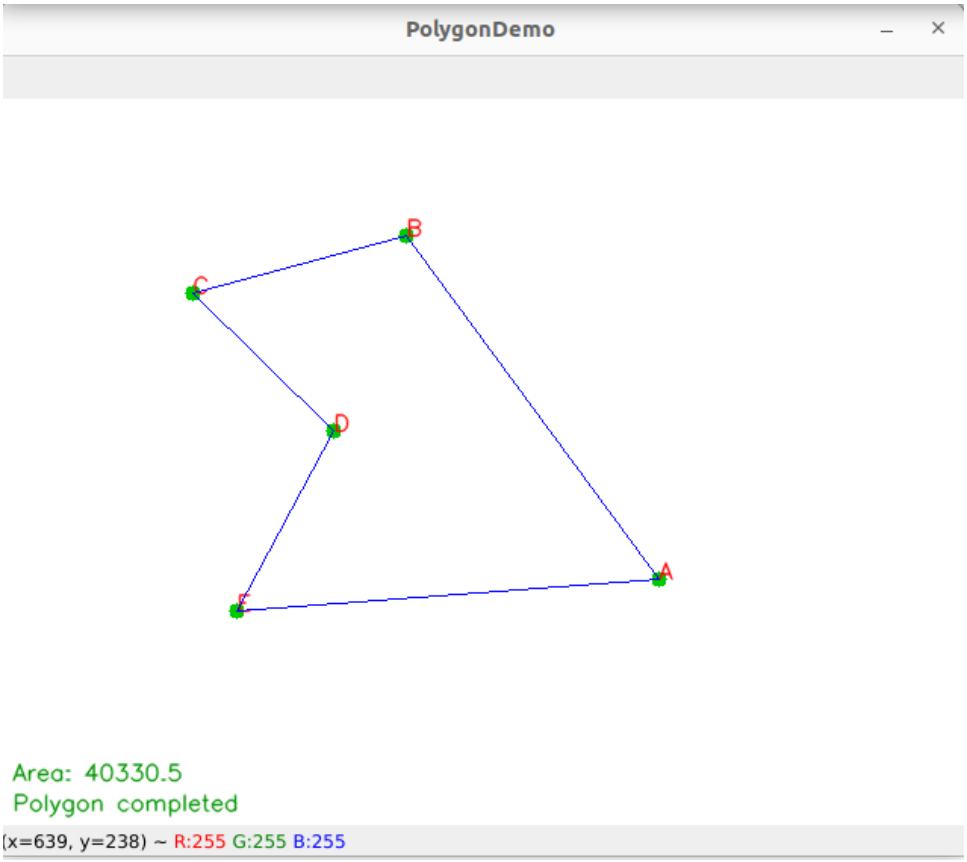
$$\frac{1}{2}\begin{vmatrix} x_1 & x_2 & x_3 & x_4 & x_5 \\ y_1 & y_2 & y_3 & y_4 & y_5 \end{vmatrix}$$

$$=\frac{1}{2}\begin{vmatrix} 437 & 270 & 129 & 222 & 150 \\ 317 & 90 & 128 & 219 & 338 \end{vmatrix}$$

$$=\frac{1}{2}\left|(39330+34560+28251+75036+50086)-(85590+11610+28416+34602+144006)\right|$$

$$=\frac{1}{2}\left|227263-307924\right|=\frac{80661}{2}=40330.5$$

Q1-3. 프로그램 결과값과의 비교

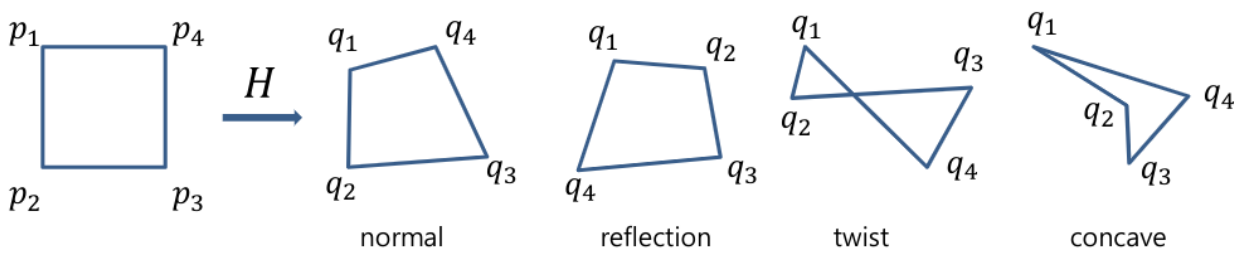


Q1-4. 구현코드

```
def polyArea(points):
    if type(points) == np.ndarray:
        return polyArea_vector(points)
    n = len(points)
    area = 0.0
    for i in range(n):
        j = (i + 1) % n
        area += points[i][0] * points[j][1]
        area -= points[j][0] * points[i][1]
    area = abs(area) / 2.0
def polyArea_vector(points: np.ndarray):

    right_shift_points = np.roll(points, 1, axis=0)
    area = np.cross(points, right_shift_points)
    return abs(area.sum()) / 2.0
```

- Q2. 사용자가 입력한 네 점에 의해 결정되는 homography 변환의 종류를 정상(normal), 뒤집힘(reflection), 뒤틀림(twist), 오목(concave), 뒤집힌 오목(reflective concave) 중의 하나로 구분하는 프로그램을 작성하시오.
  - 제공되는 GitHub 코드 활용 (cpp_template, feature_matching)
  - 1) check_homography = true로 설정, classifyHomography() 함수 구현
  - 2) 구현 결과를 feature_matching 예제에 적용하여 잘못된 homography 제거
  - 리포트(Slack) + 구현코드(GitHub 링크) 제출
    실행화면 캡처 포함



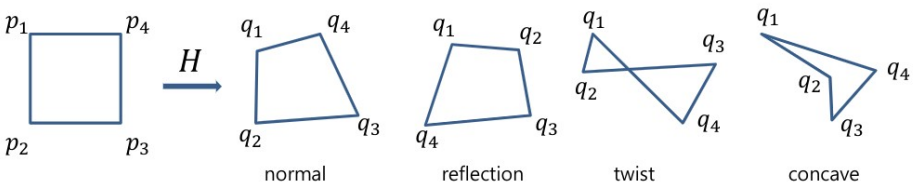
Q2-1. classifyHomograghy() 함수 구현

```
def classifyHomography(pts1: np.ndarray, pts2: np.ndarray) -> int:
    if len(pts1) != 4 or len(pts2) != 4:
        return HomographyType.UNKNOWUN

    pt1 = np.cross(pts1 - np.roll(pts1, -1, axis=0), pts1 - np.roll(pts1, 1, axis=0))
    pt2 = np.cross(pts2 - np.roll(pts2, -1, axis=0), pts2 - np.roll(pts2, 1, axis=0))

    pts = pt1 * pt2
    h_type = (pts < 0).sum()
    if h_type == 4:
        return HomographyType.REFLECTION
    elif h_type == 2:
        return HomographyType.TWIST
    elif h_type in [1, 3]:
        return HomographyType.CONCAVE
    return HomographyType.NORMAL
```

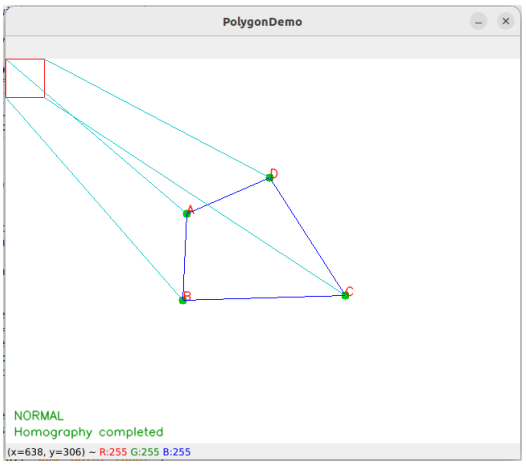
cf) 잘못된 Homograghy 판단하기



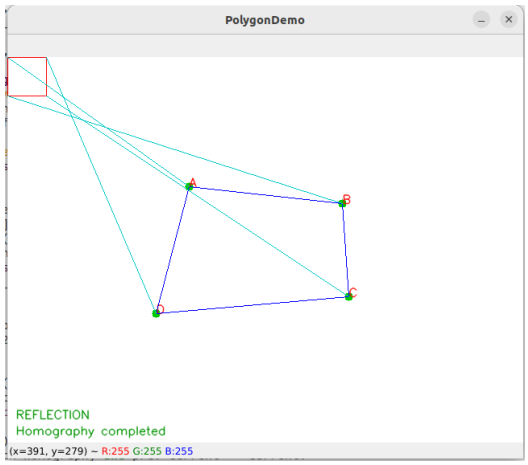
- q_1, q_2, q_3, q_4
  - H에 의해 변환된 네 꼭지점 영상좌표
  - $q_i = Hp_i$
- for each i = 1, 2, 3, 4
  - $p_ip_{i-1} \times p_ip_{i+1} = (0, 0, J_p)$
  - $q_iq_{i-1} \times q_iq_{i+1} = (0, 0, J_q)$
  - $J_pJ_q<0$ 면 잘못된 호모그래피

Q2-2.

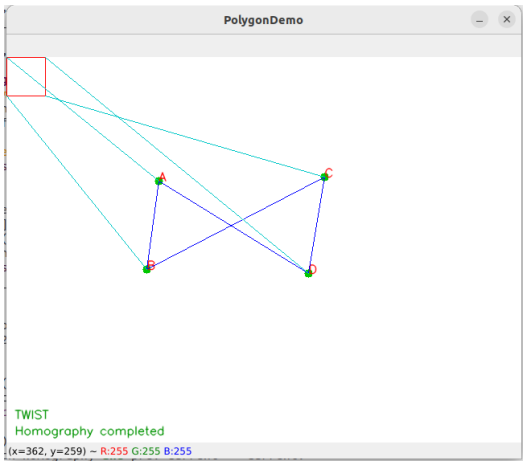
- 실행화면 (cpp_template)



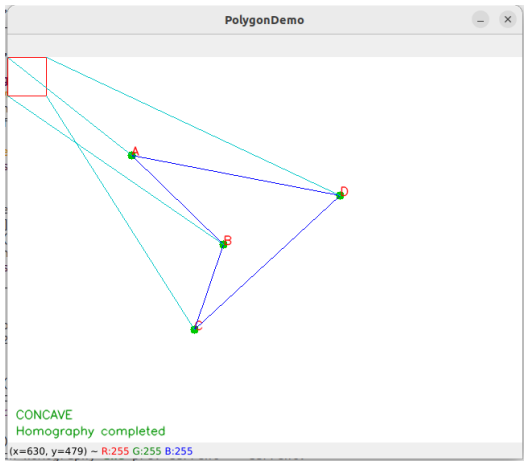
1. normal



2. reflection



3. twist



4. concave

- 실행화면 (feature_matching)