

미니 프로젝트 Ver 1.0 스핀오프

디지털 영상 처리 w. Python

[Intel] 엡지 AI SW 아카데미 - 절차지향 프로그래밍

Contents

- 프로젝트의 목표
- 개발 환경
- 화면 구성 및 기능
- 부가 기능
- 마치며

프로젝트의 목표



비전

C언어 이외의 언어를 사용해
각 언어의 유사점과 차이점을
직접 체감해 보는 것



미션

C언어로 구현했던
영상 처리 프로그램을
Python으로 구현하는 것

개발 환경

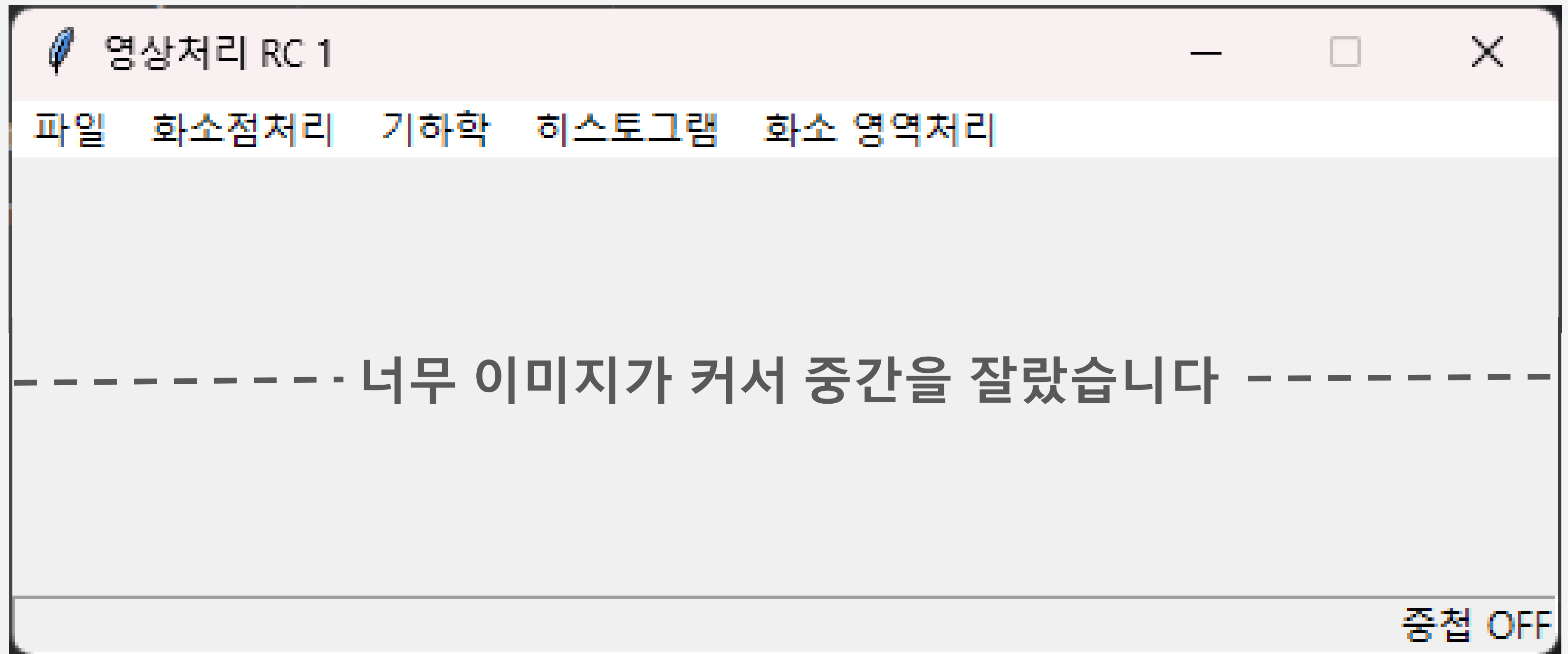


PyCharm 2023

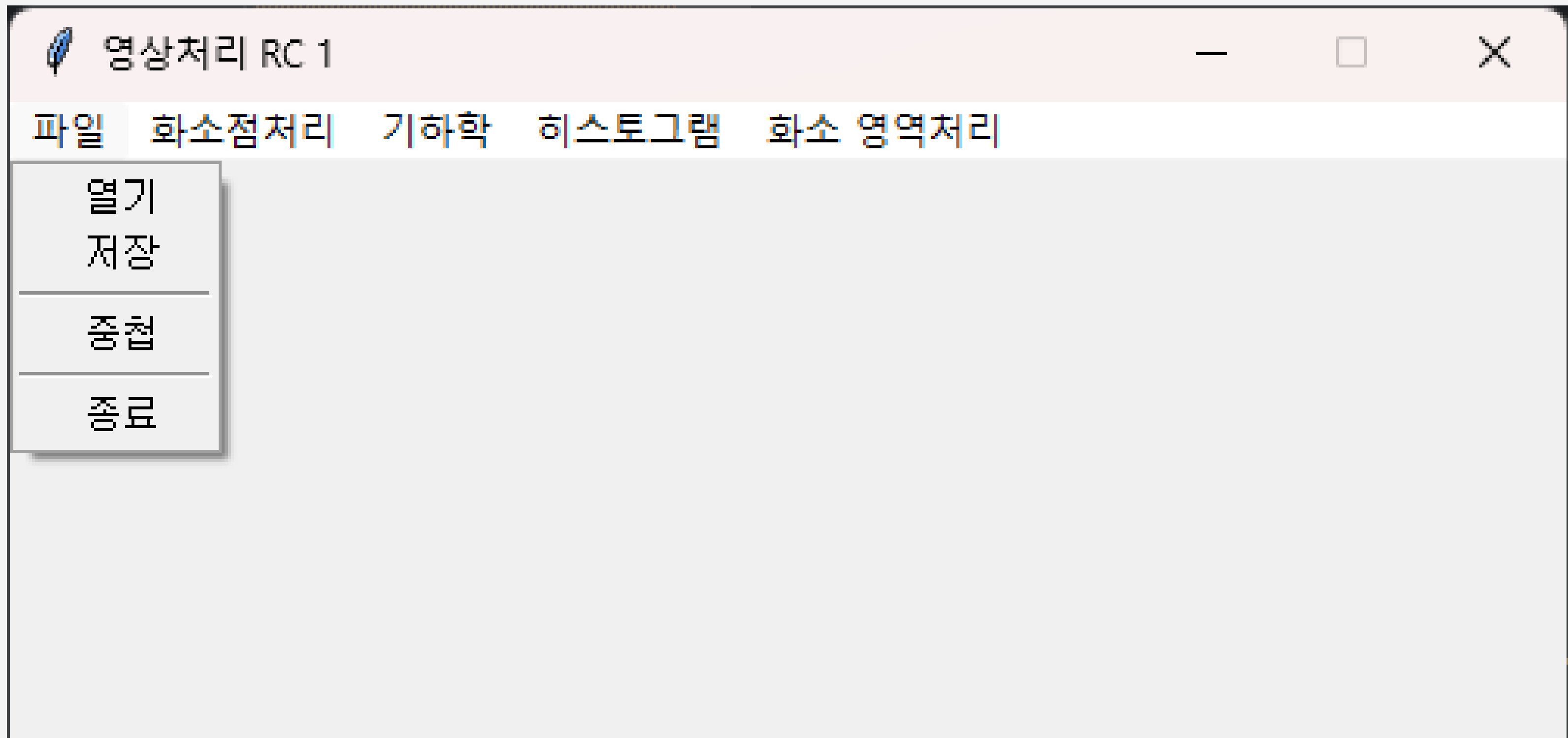


Python 3.12

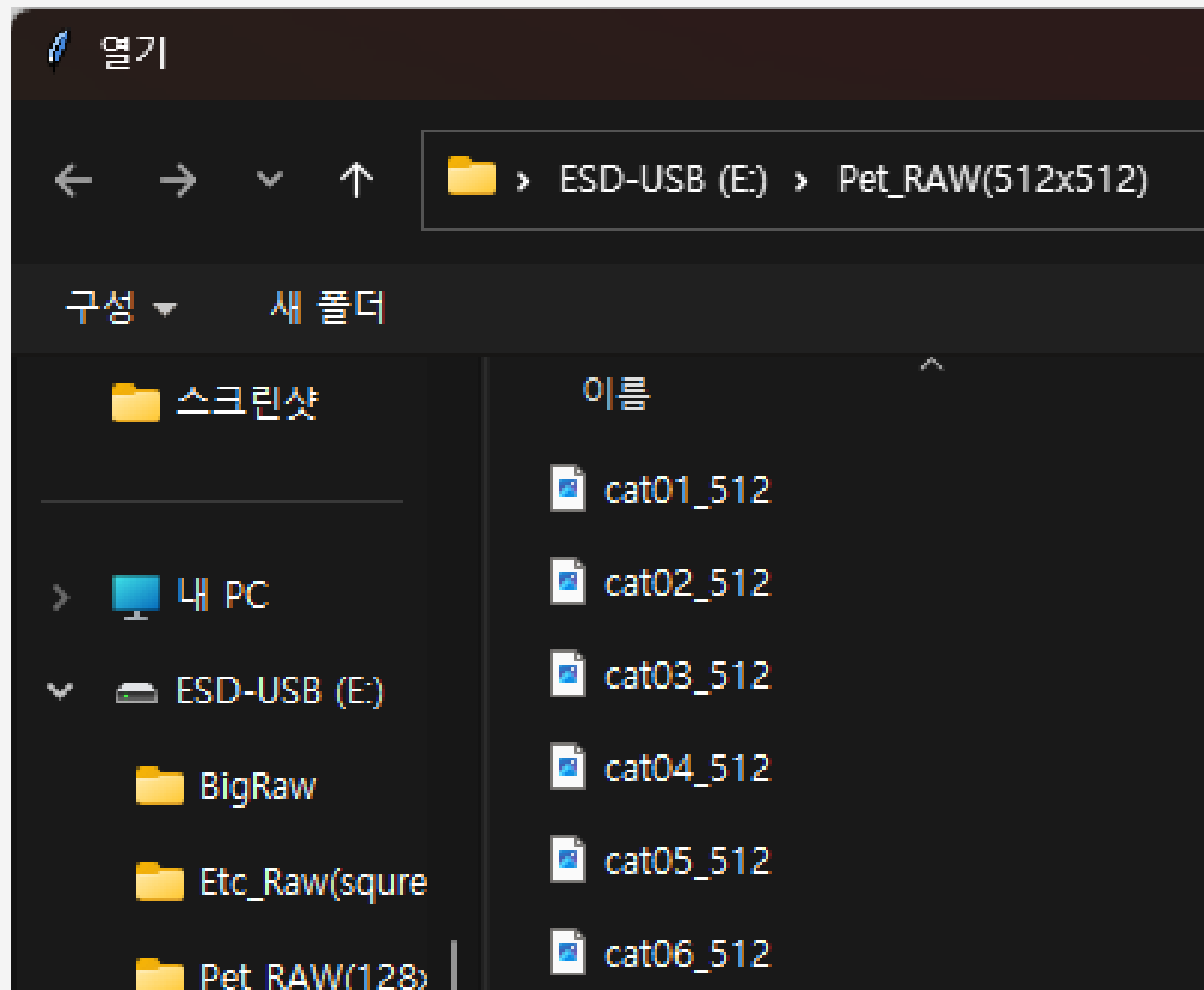
■ 화면 구성 및 기능 초기 화면



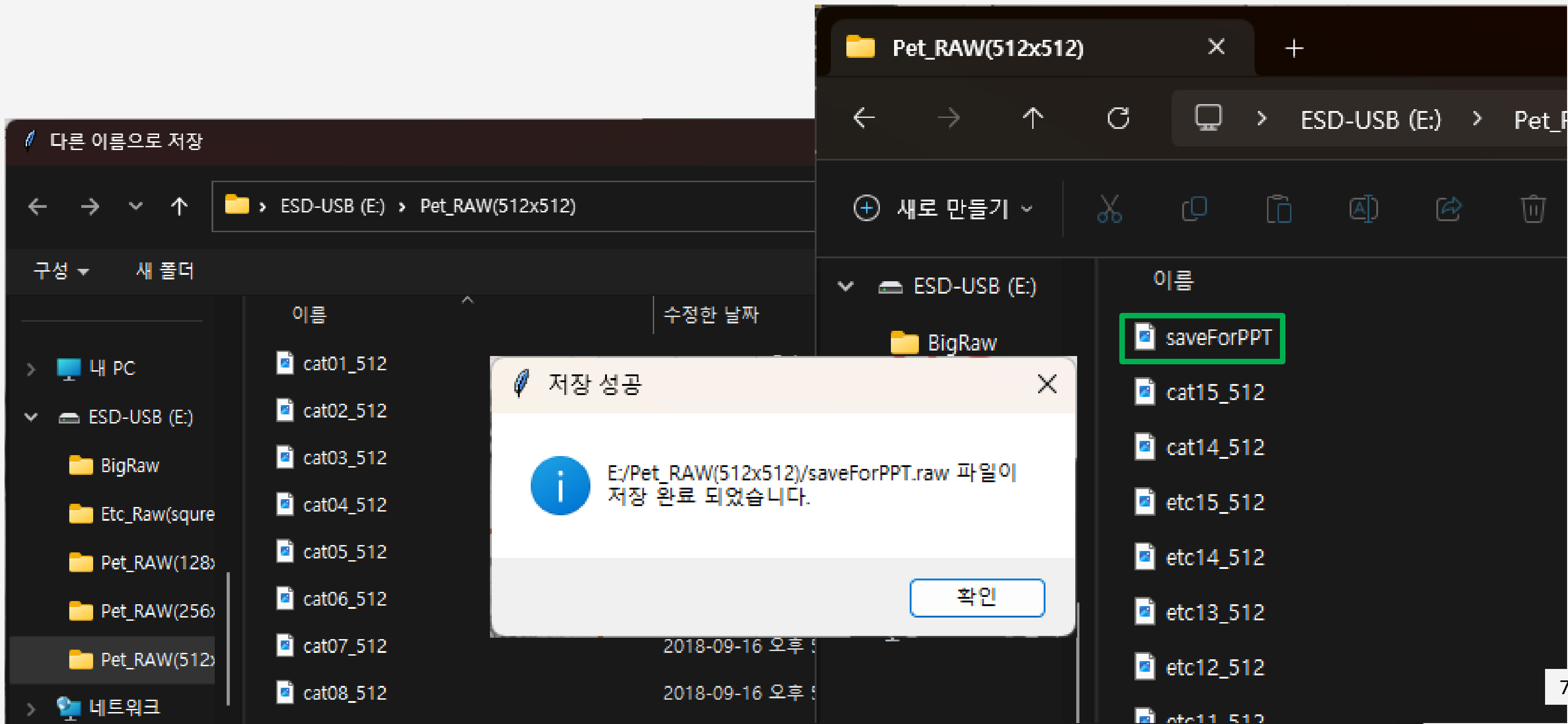
■ 화면 구성 및 기능 파일 메뉴



화면 구성 및 기능 열기 창으로 파일 가져오기



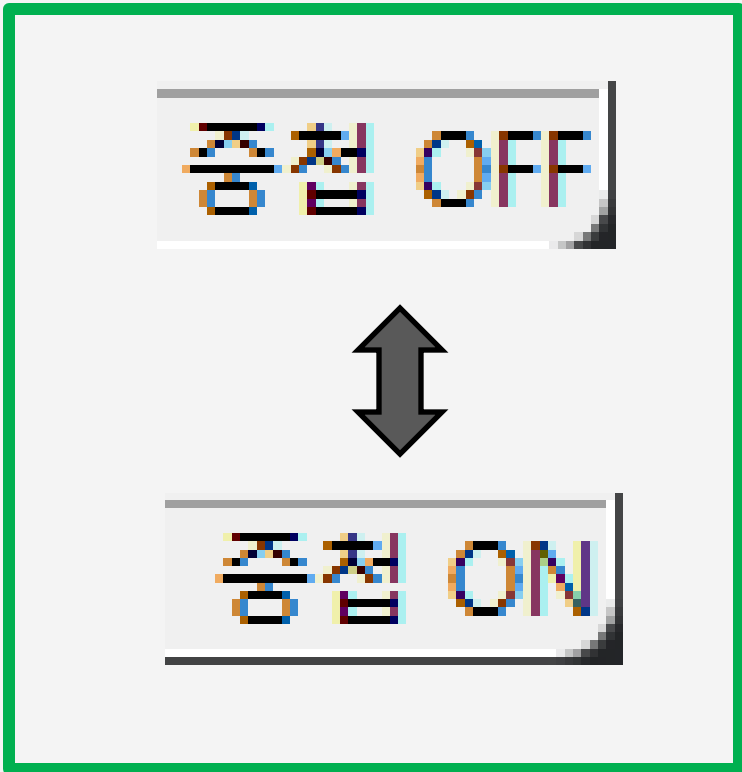
화면 구성 및 기능 저장 창으로 파일 저장하기



화면 구성 및 기능 효과 중첩 ON / OFF 및 확인

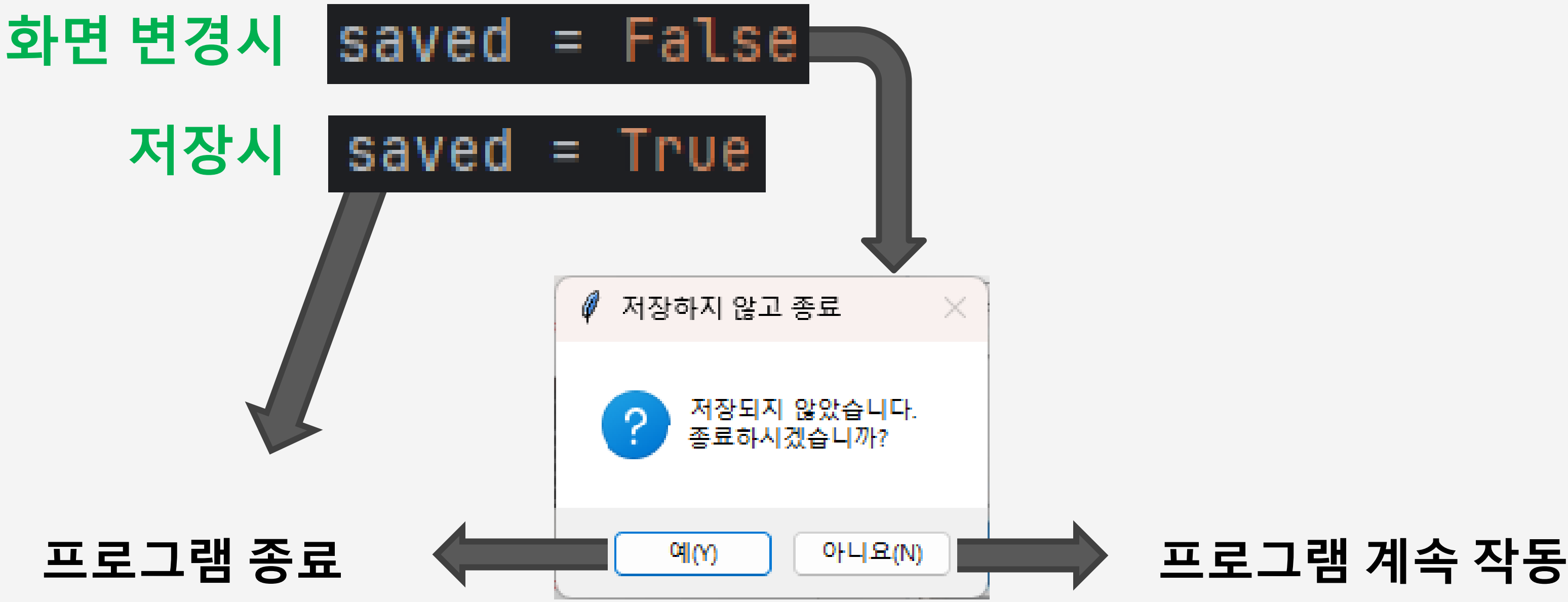


클릭 시 변경

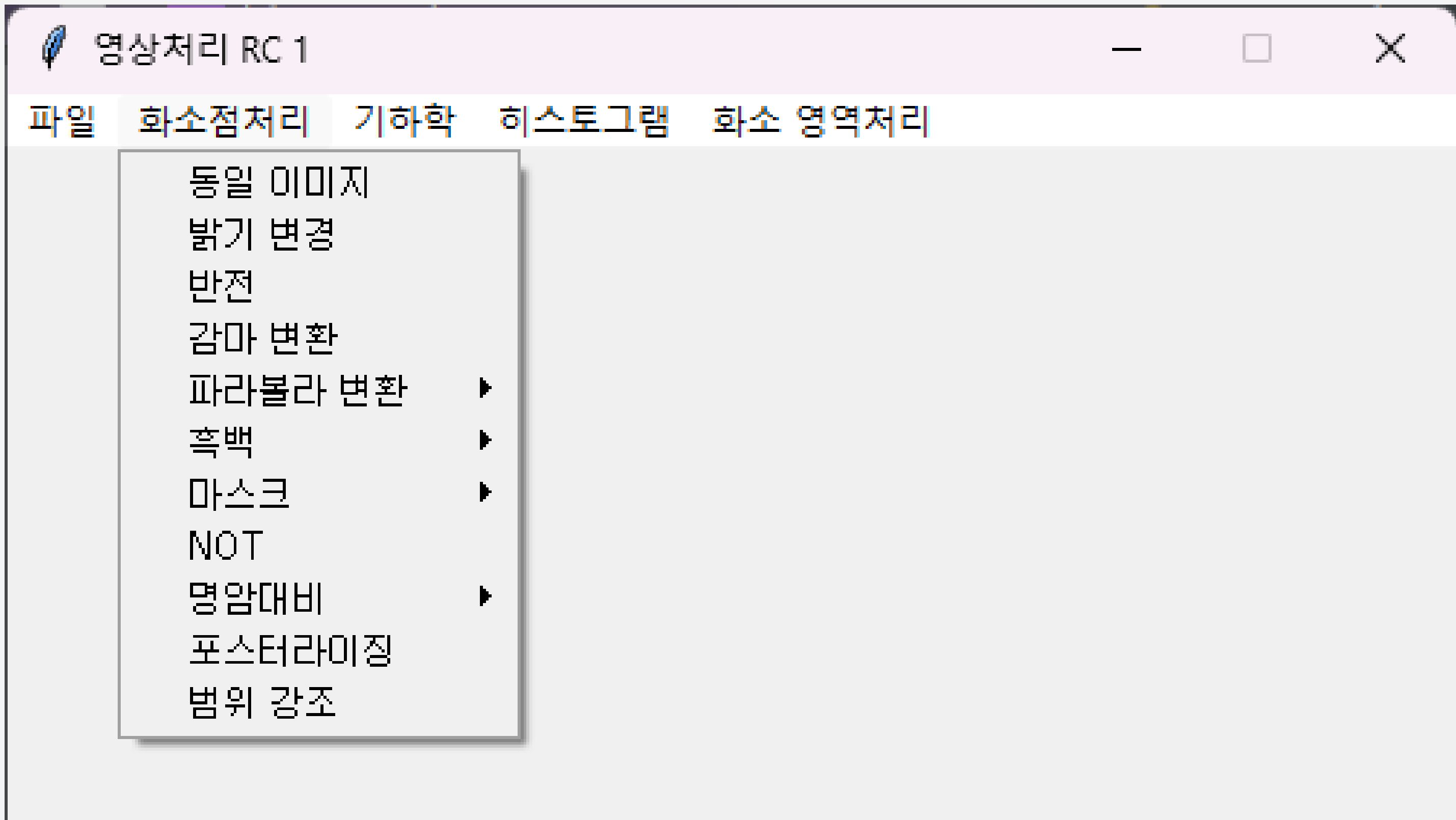


```
global statusbar, accumulation
statusbar.config(text=accumul)
```

■ 화면 구성 및 기능 저장 여부를 확인하고 종료하기



■ 화면 구성 및 기능 화소점처리 메뉴



화면 구성 및 기능

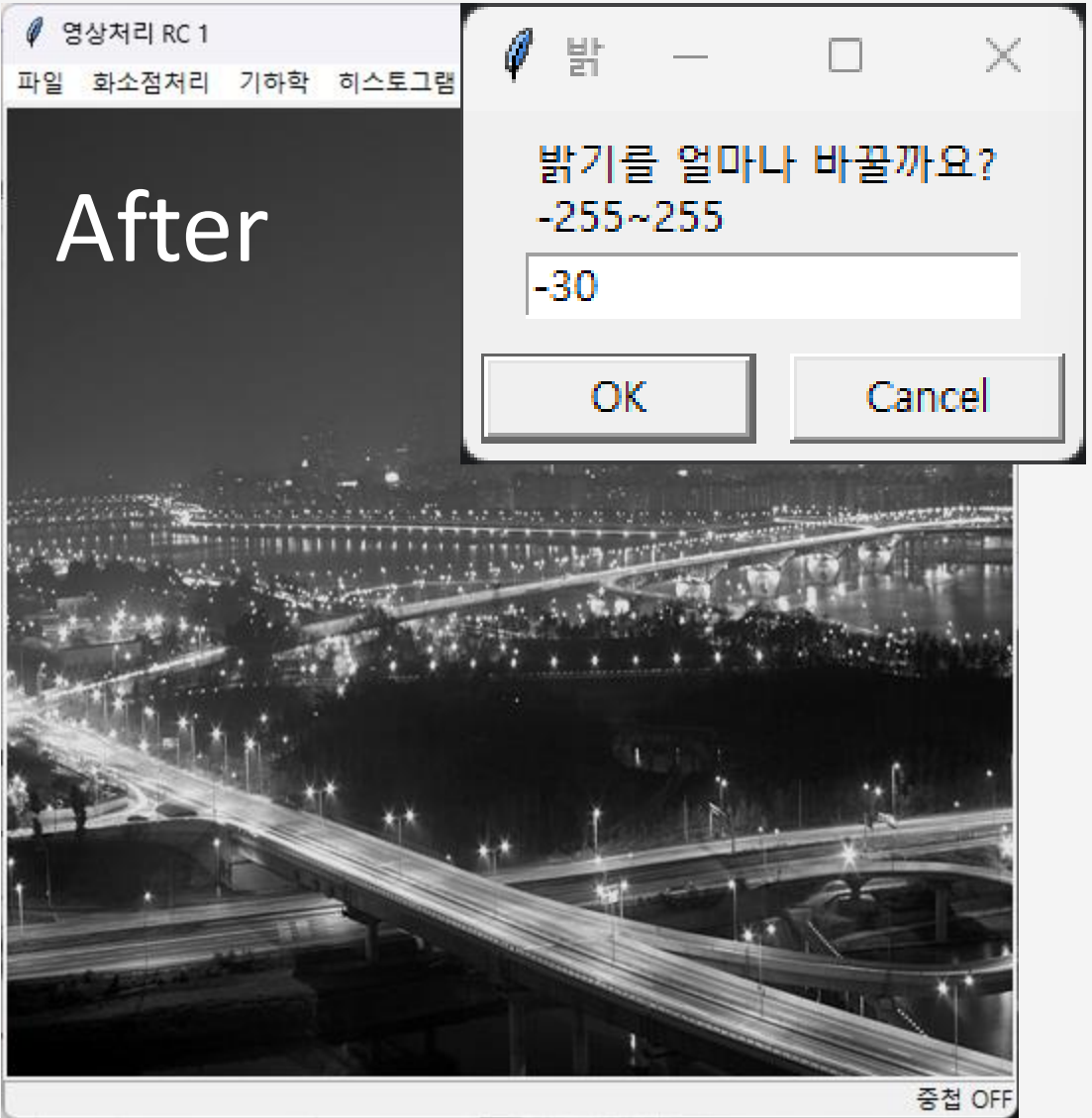
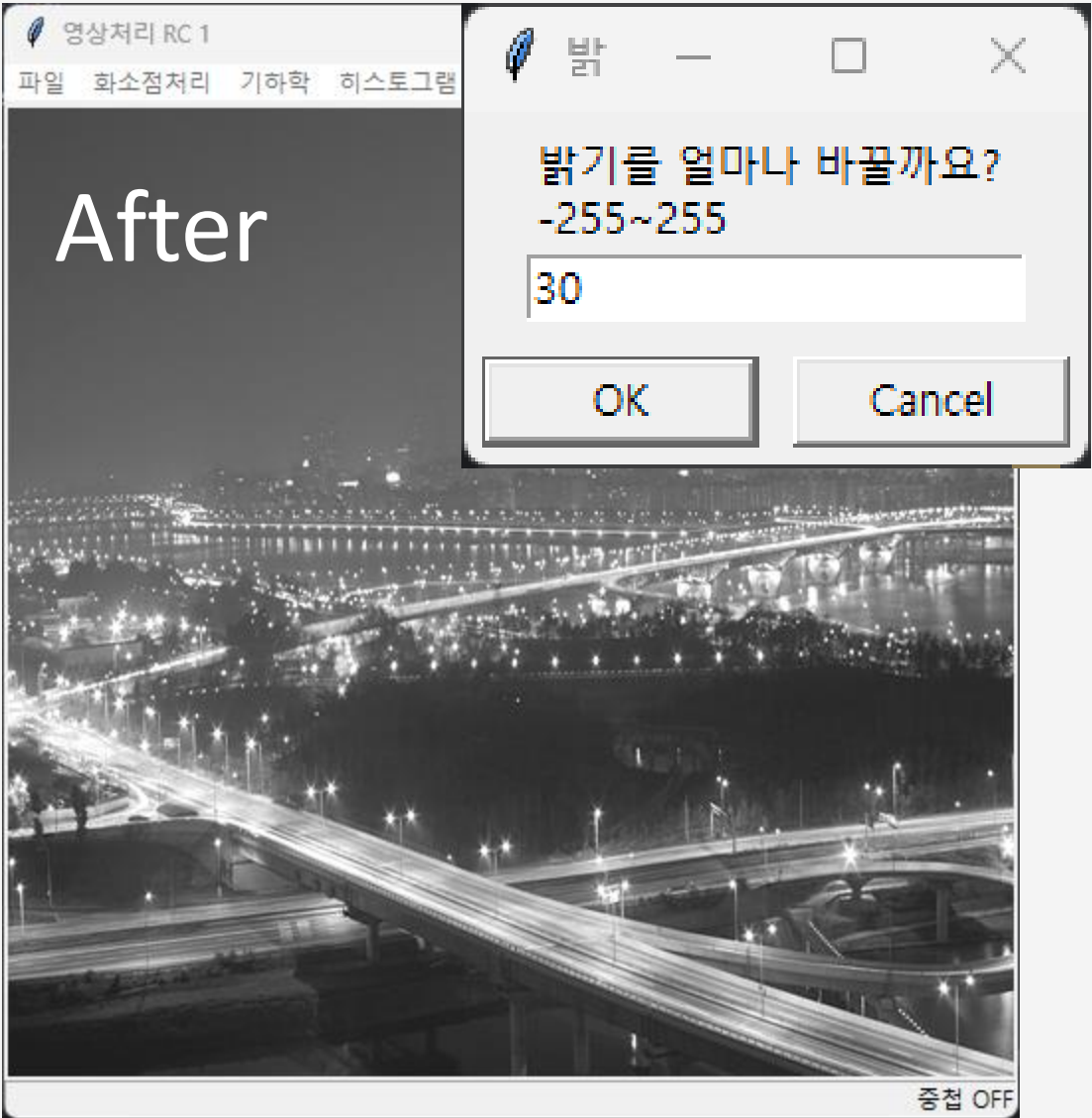
화소점처리 > 동일영상 알고리즘



```
outImage[i][k] = inImage[i][k]
```

화면 구성 및 기능

화소점처리 > 밝기 변경

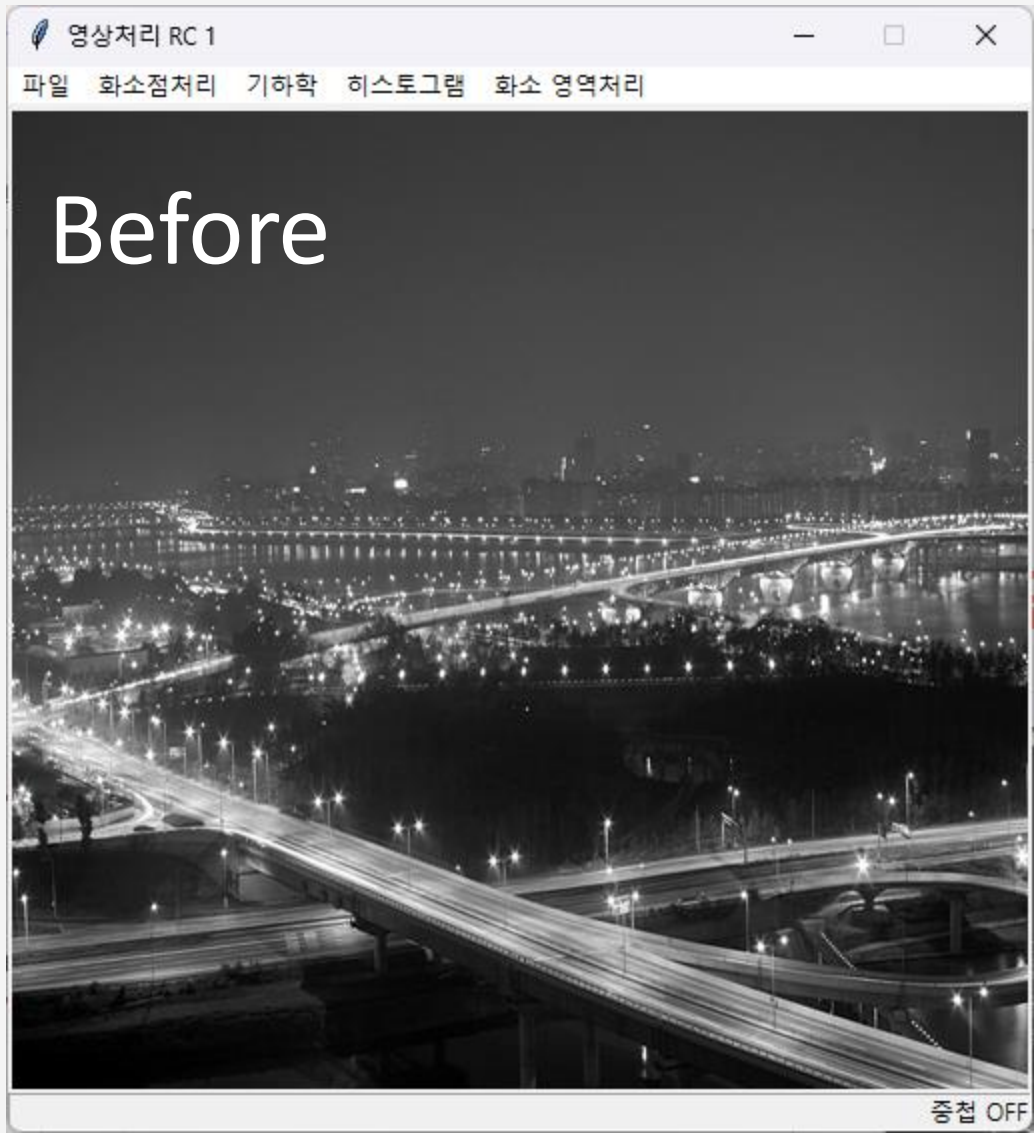


```
px = inImage[i][k] + scale
if (px > 255):
    px = 255
```

```
if (px < 0):
    px = 0
outImage[i][k] = px
```


화면 구성 및 기능

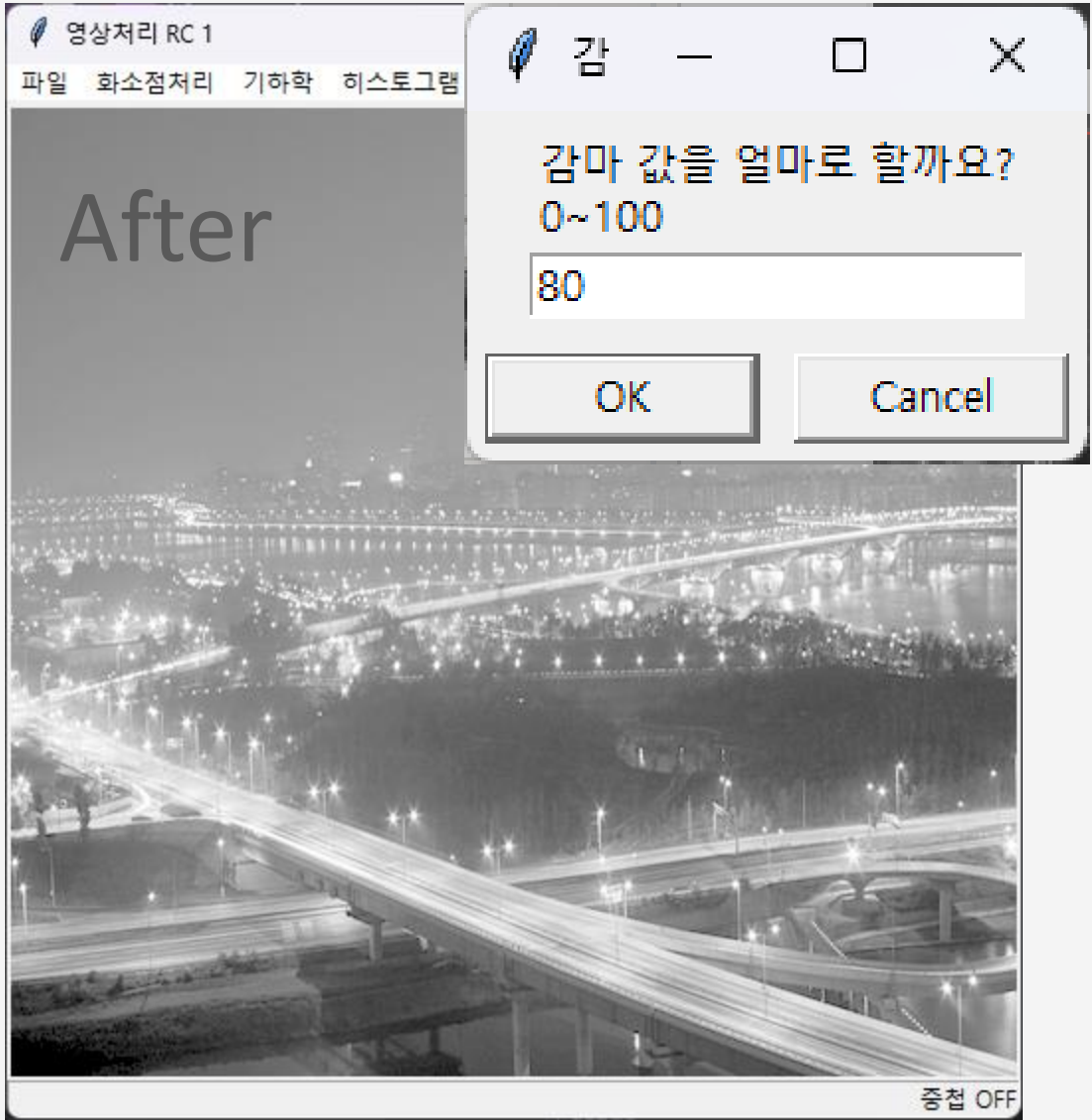
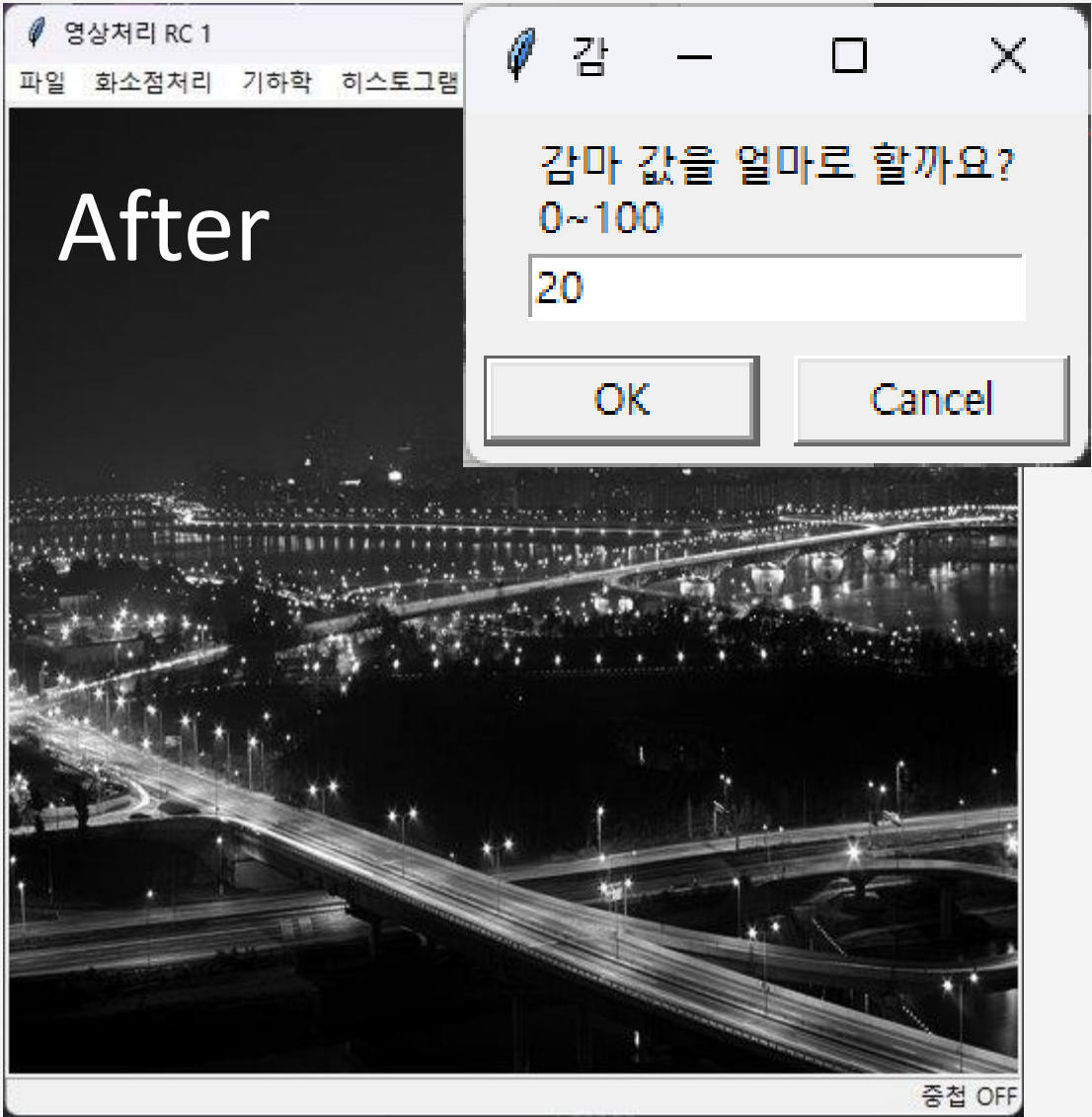
화소점처리 > 반전



```
outImage[i][k] = 255 - inImage[i][k]
```

화면 구성 및 기능

화소점처리 > 감마값 변환



$$\text{OutImage} = 255 * \left(\frac{\text{InImage}}{255} \right)^{\frac{30}{\text{scale}}}$$

입력 받은 수치를 유의미하게 관찰할 수 있는 정도

```
outImage[i][k] = int(255 * (inImage[i][k] / 255) ** (30 / scale))
```


화면 구성 및 기능

화소점처리 > 감마값 변환



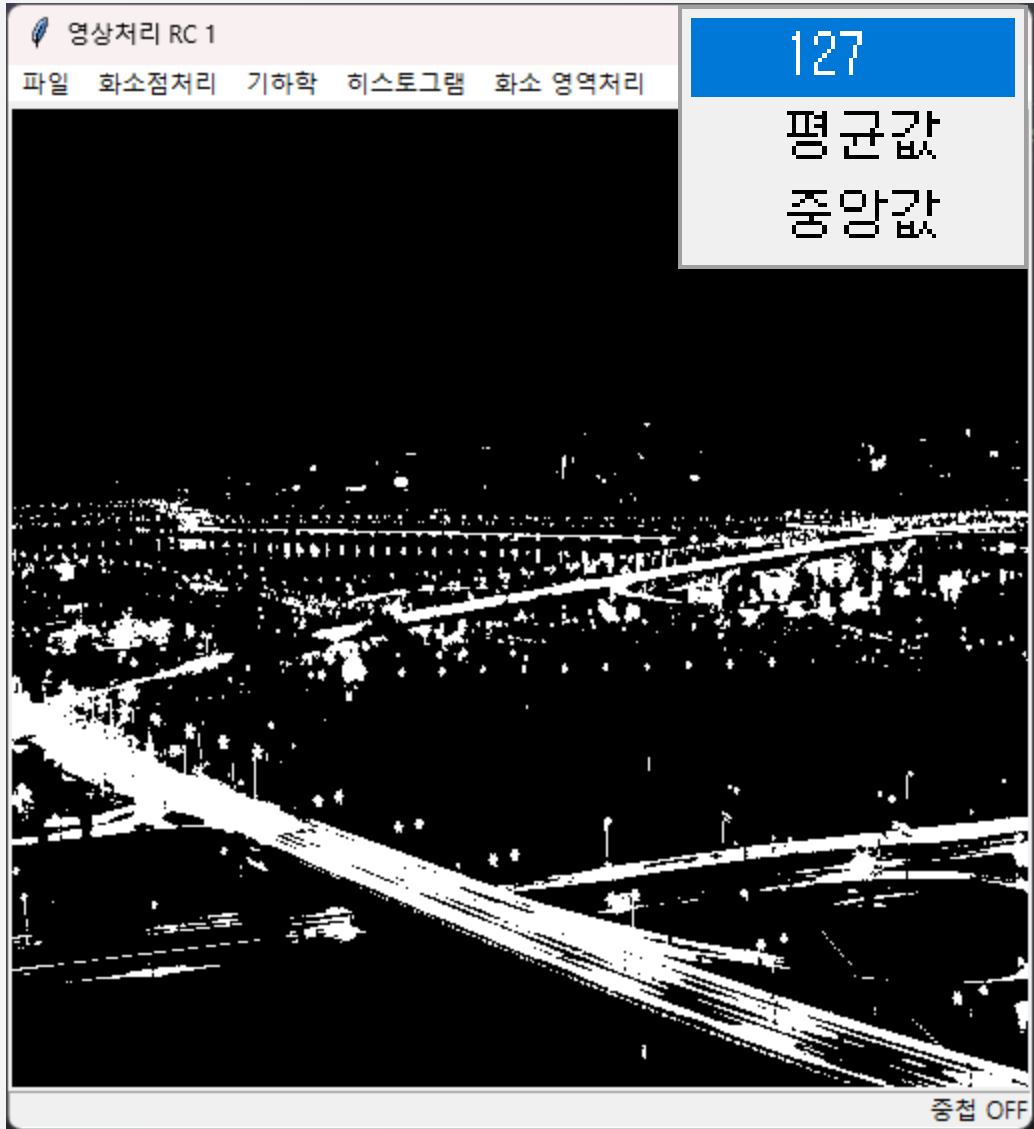
`int(255. * ((inImage[i][k] / 127.) - 1) ** 2)`



`int(255 - 255 * ((inImage[i][k] / 127) - 1) ** 2)`

화면 구성 및 기능

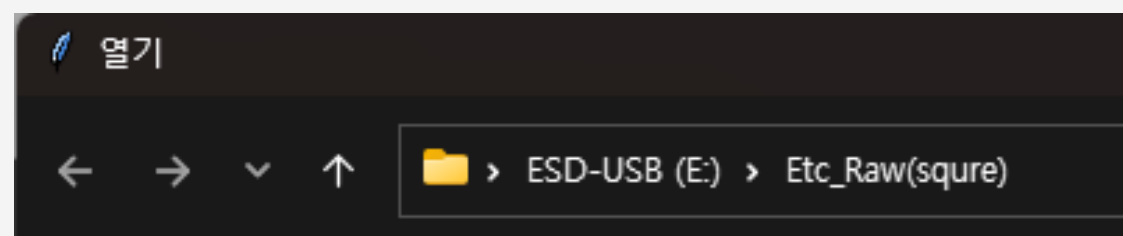
화소점처리 > 감마값 변환



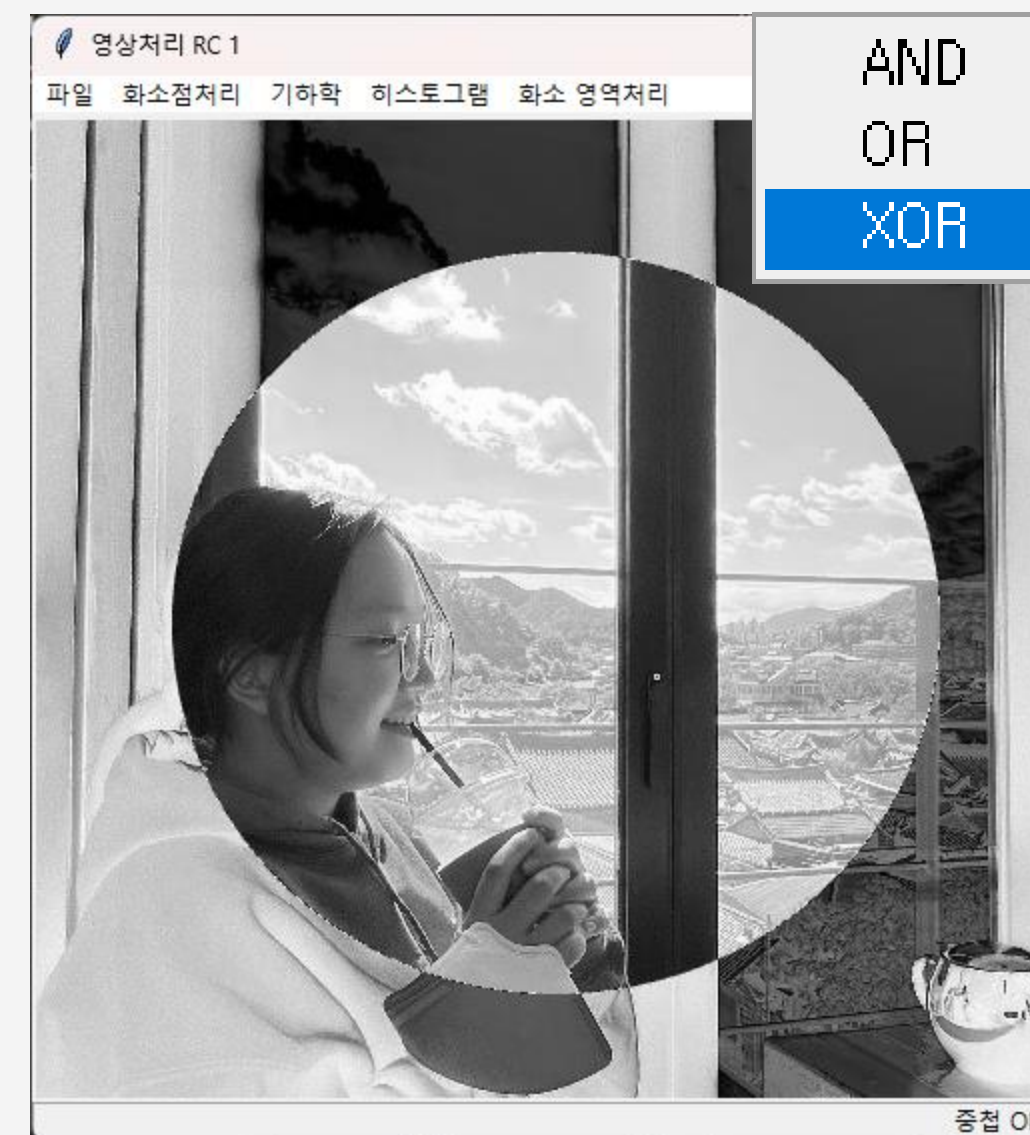
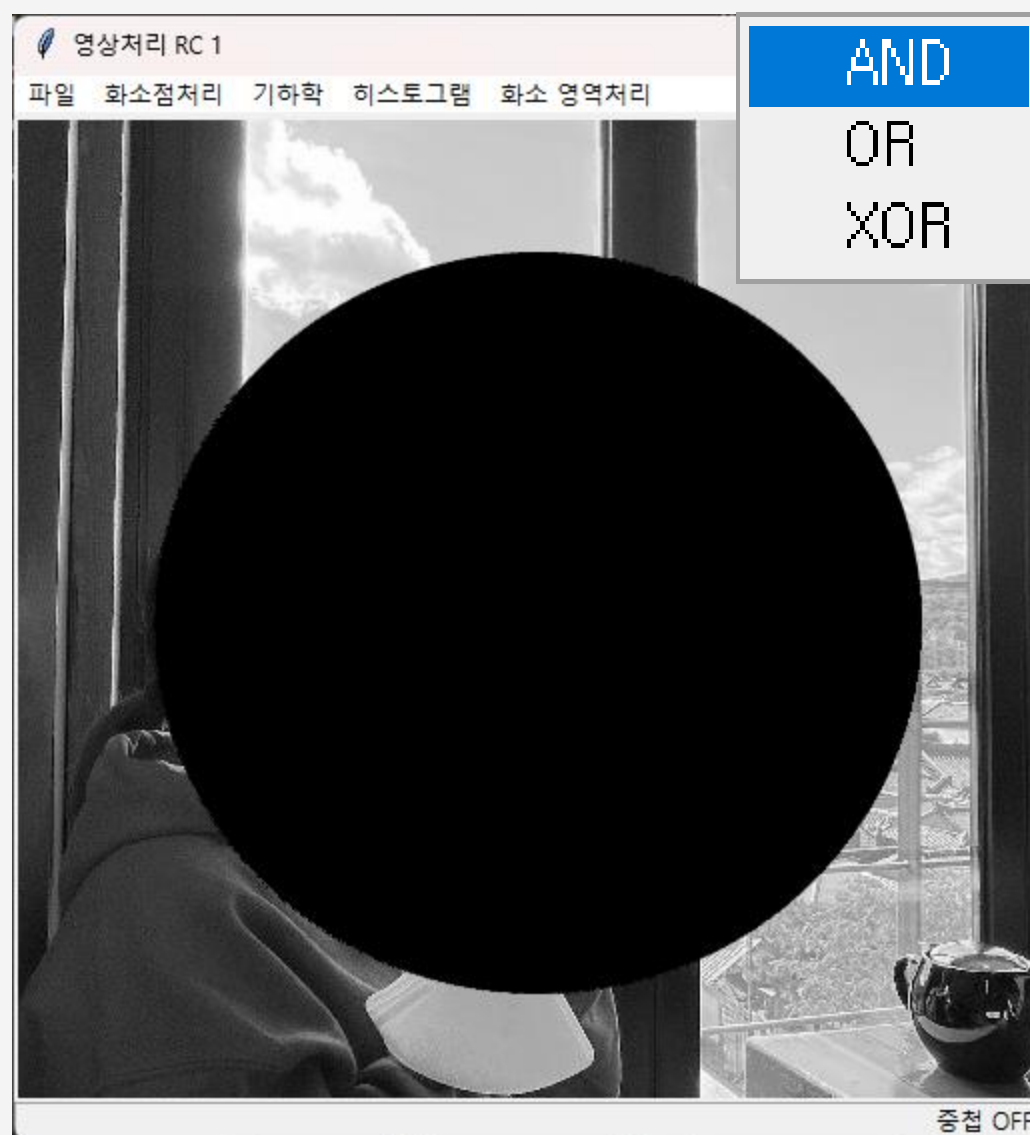
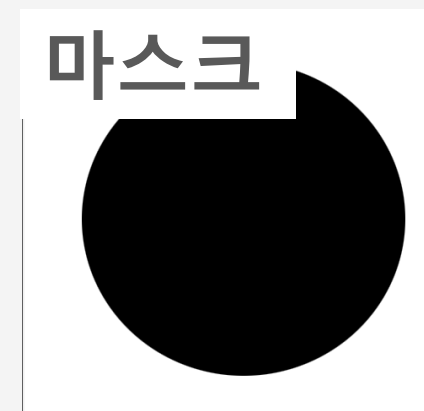
화면 구성 및 기능

화소점처리 > 감마값 변환

- 동일 이미지
- 밝기 변경
- 반전
- 감마 변환
- 파라볼라 변환
- 흑백
- 마스크**
- NOT
- 명암대비
- 포스터라이징
- 범위 강조



열기로 마스크 이미지 결정



화면 구성 및 기능

화소점처리 > NOT 반전

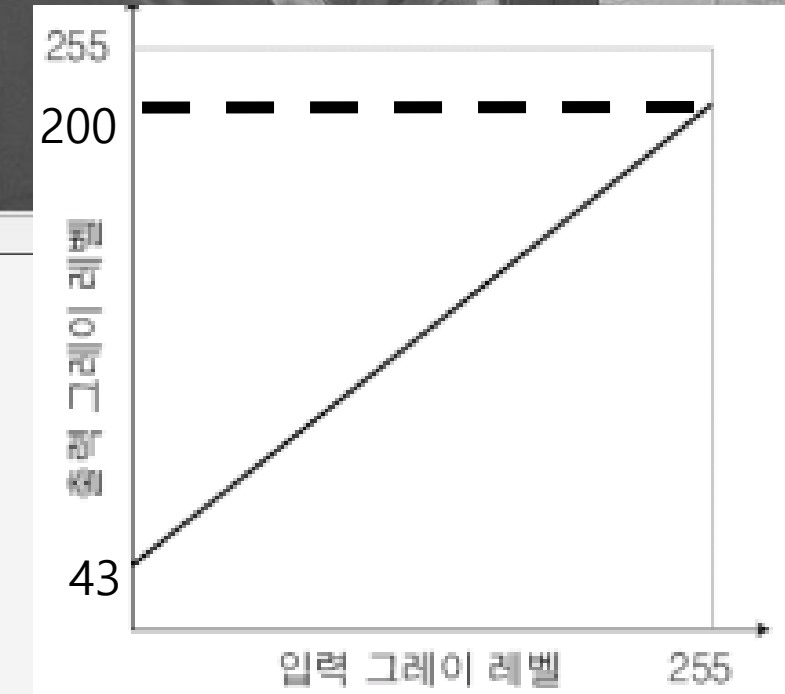
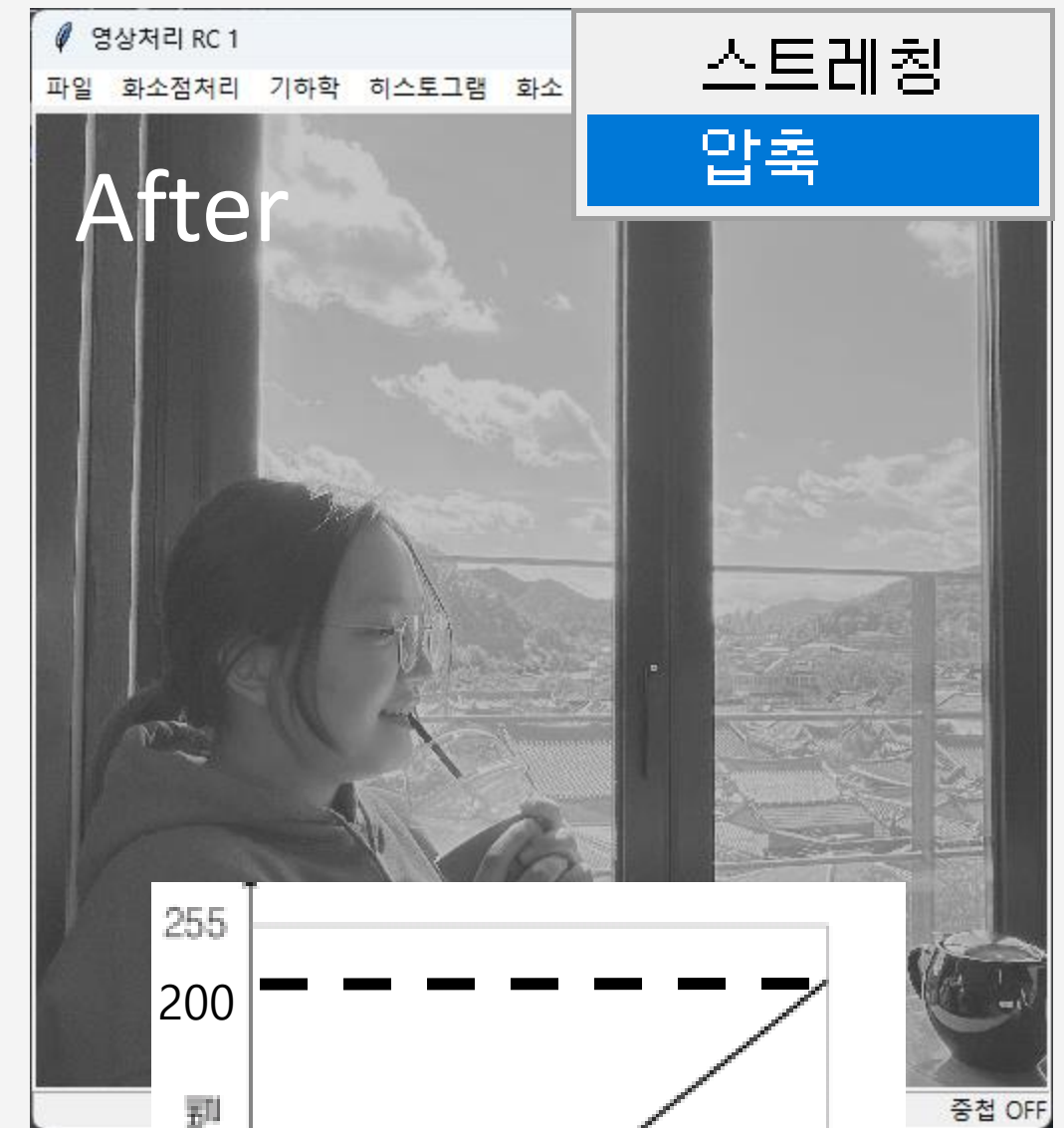
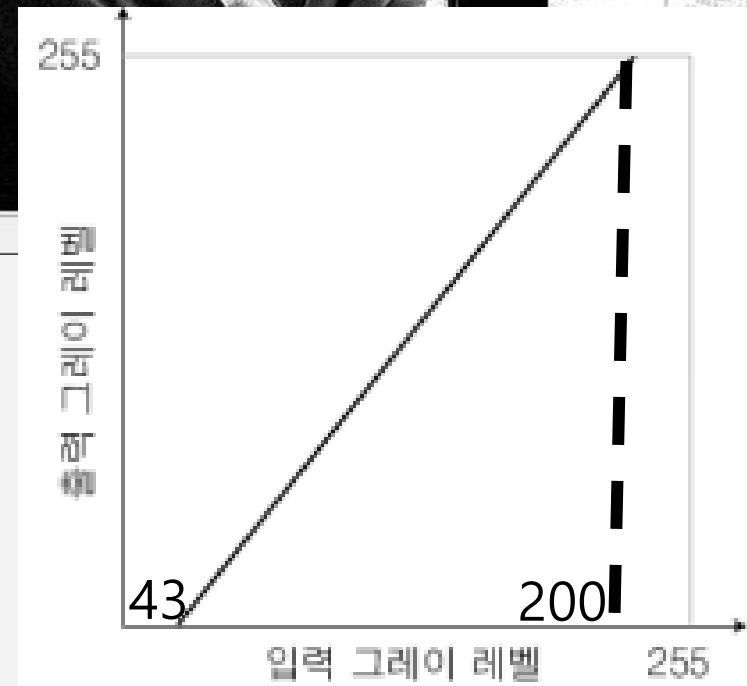


파이썬의 데이터 타입은 int로
음수로 넘어가기도 하고
255를 넘기도 하여
둘 모두를 처리하도록 함

```
val = ~inImage[i][k]
if (val < 0):
    val = abs(val % 255)
outImage[i][k] = val
```

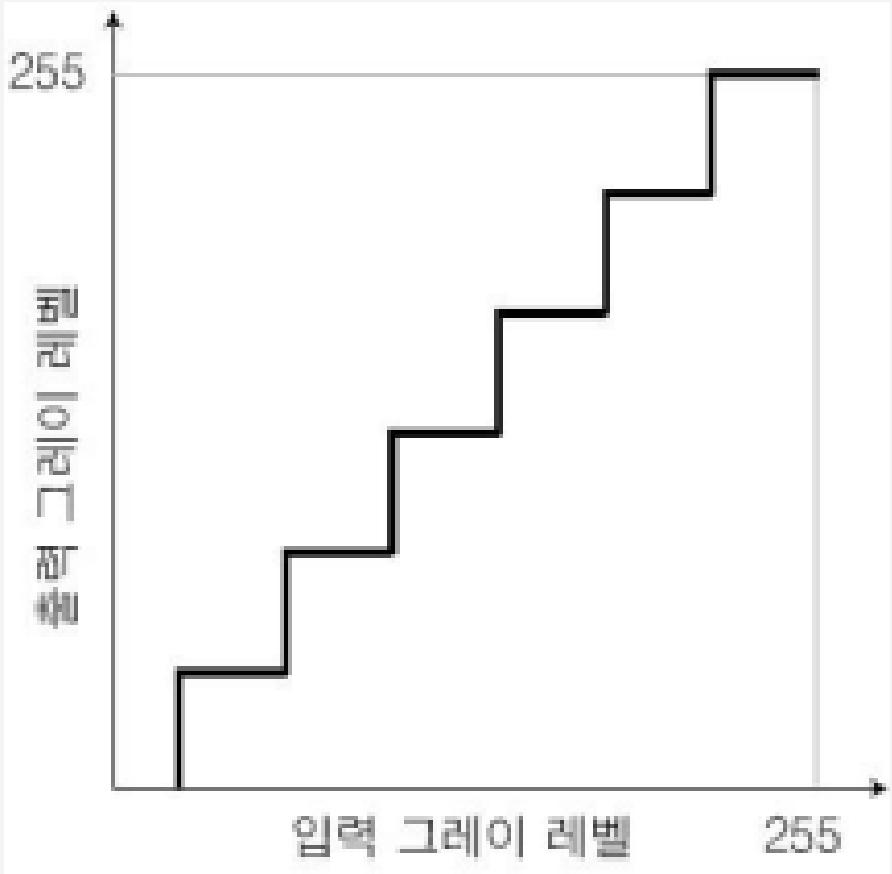
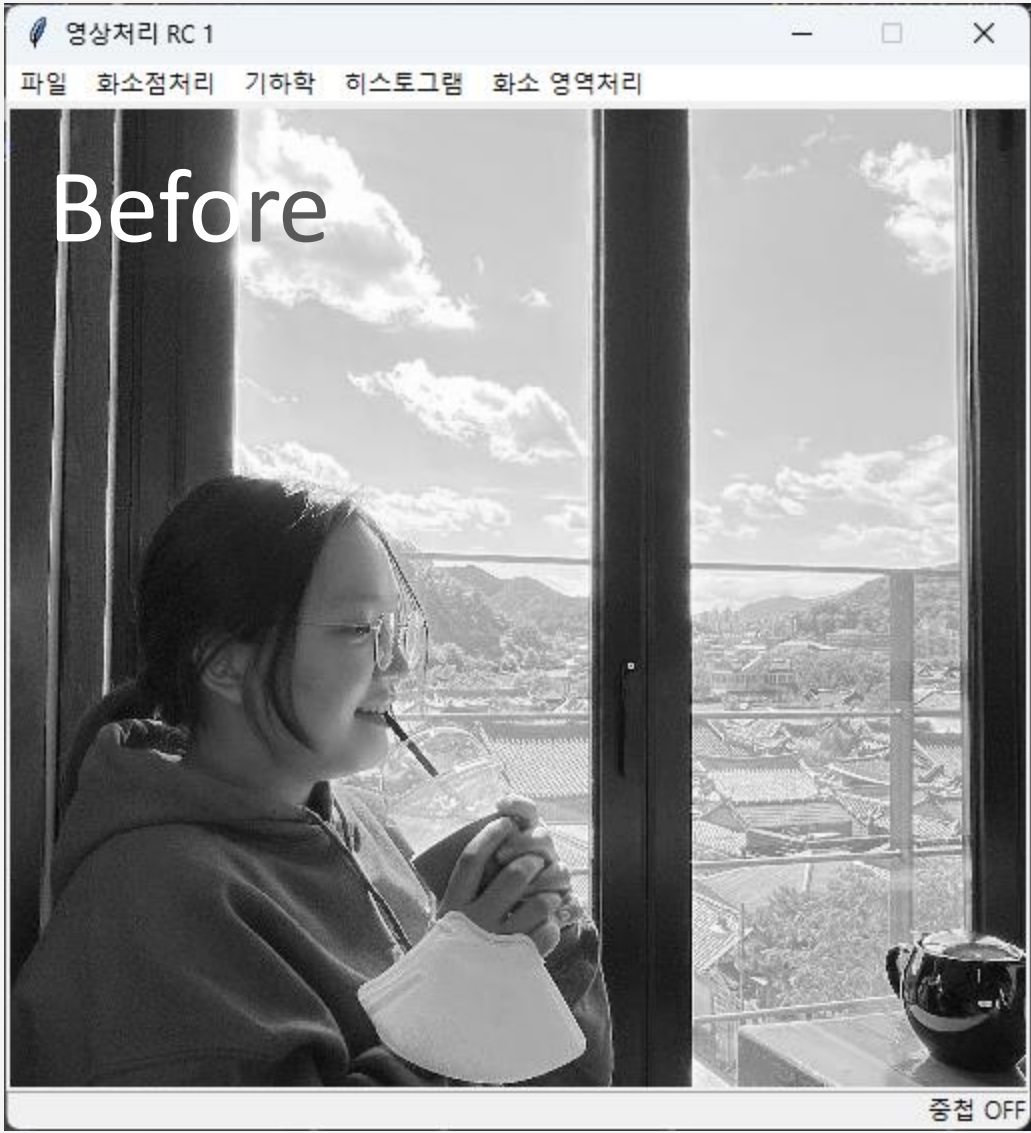

화면 구성 및 기능

화소점처리 > 명암대비

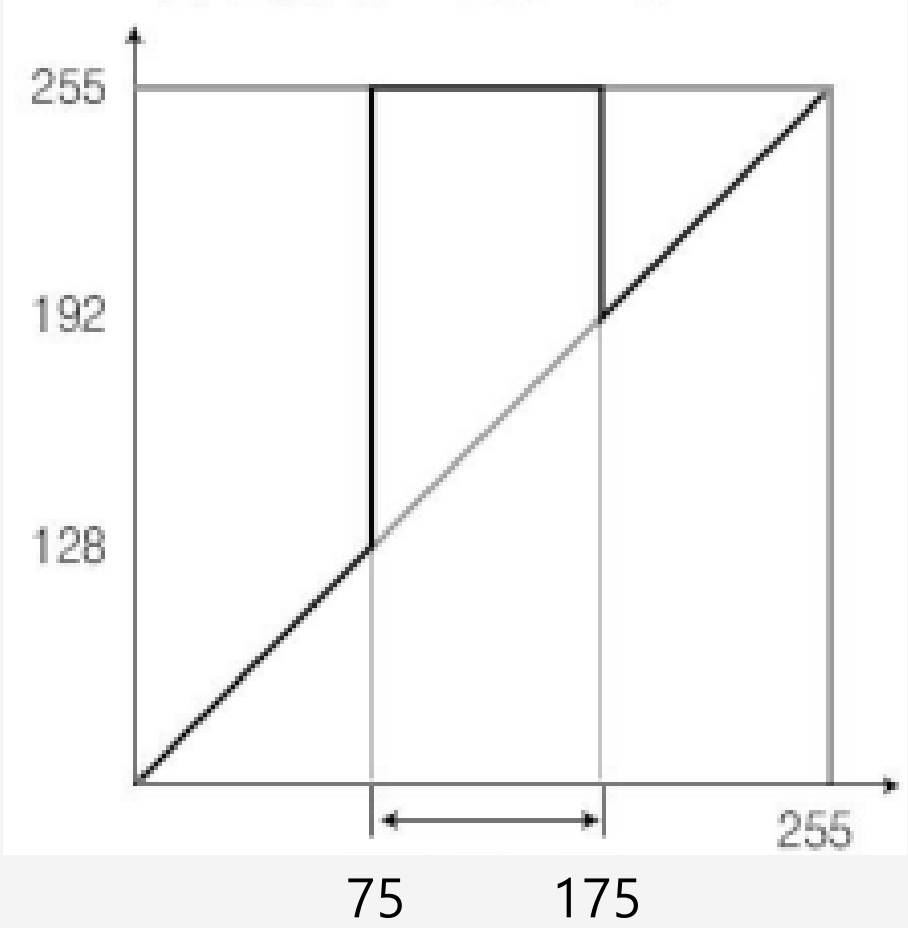
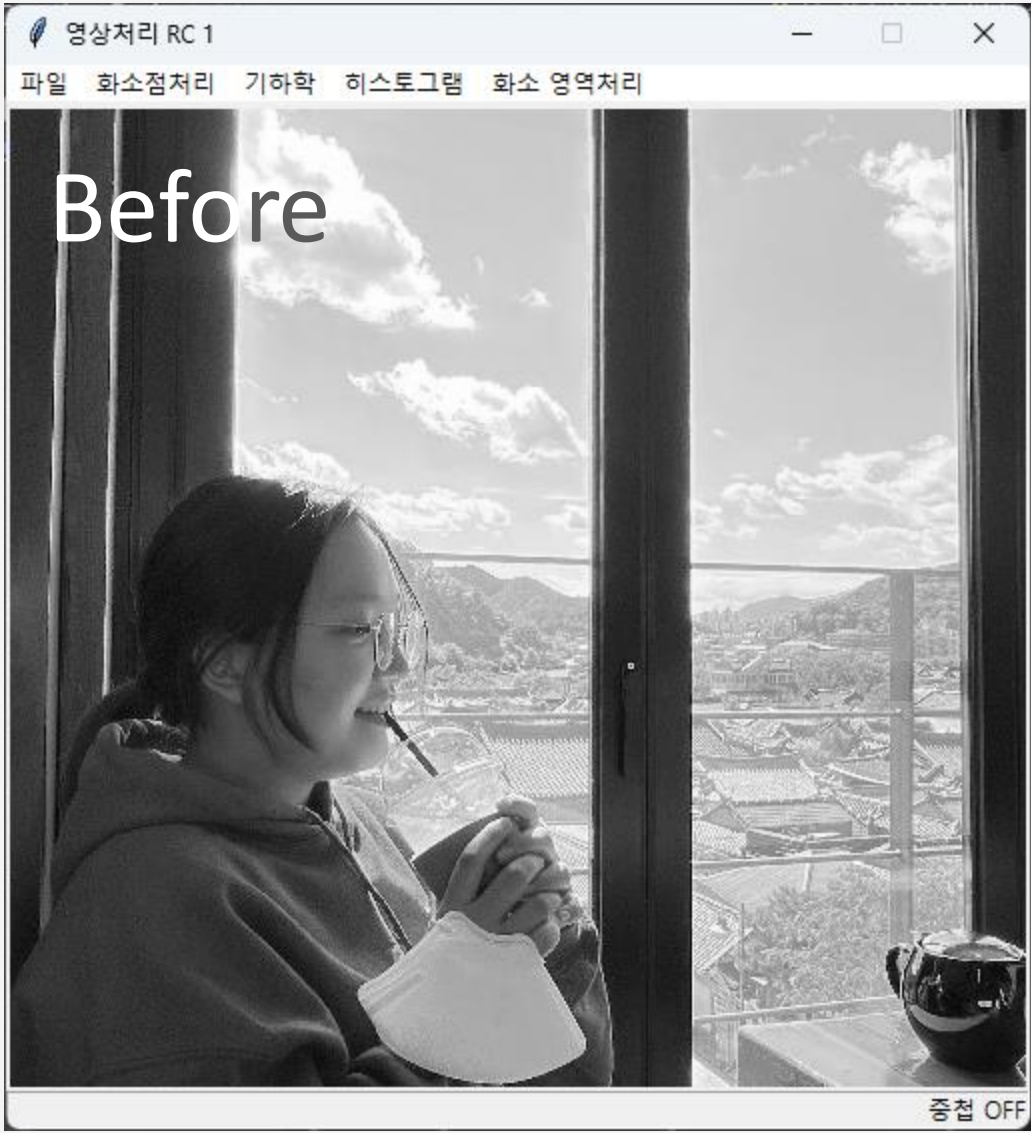


화면 구성 및 기능

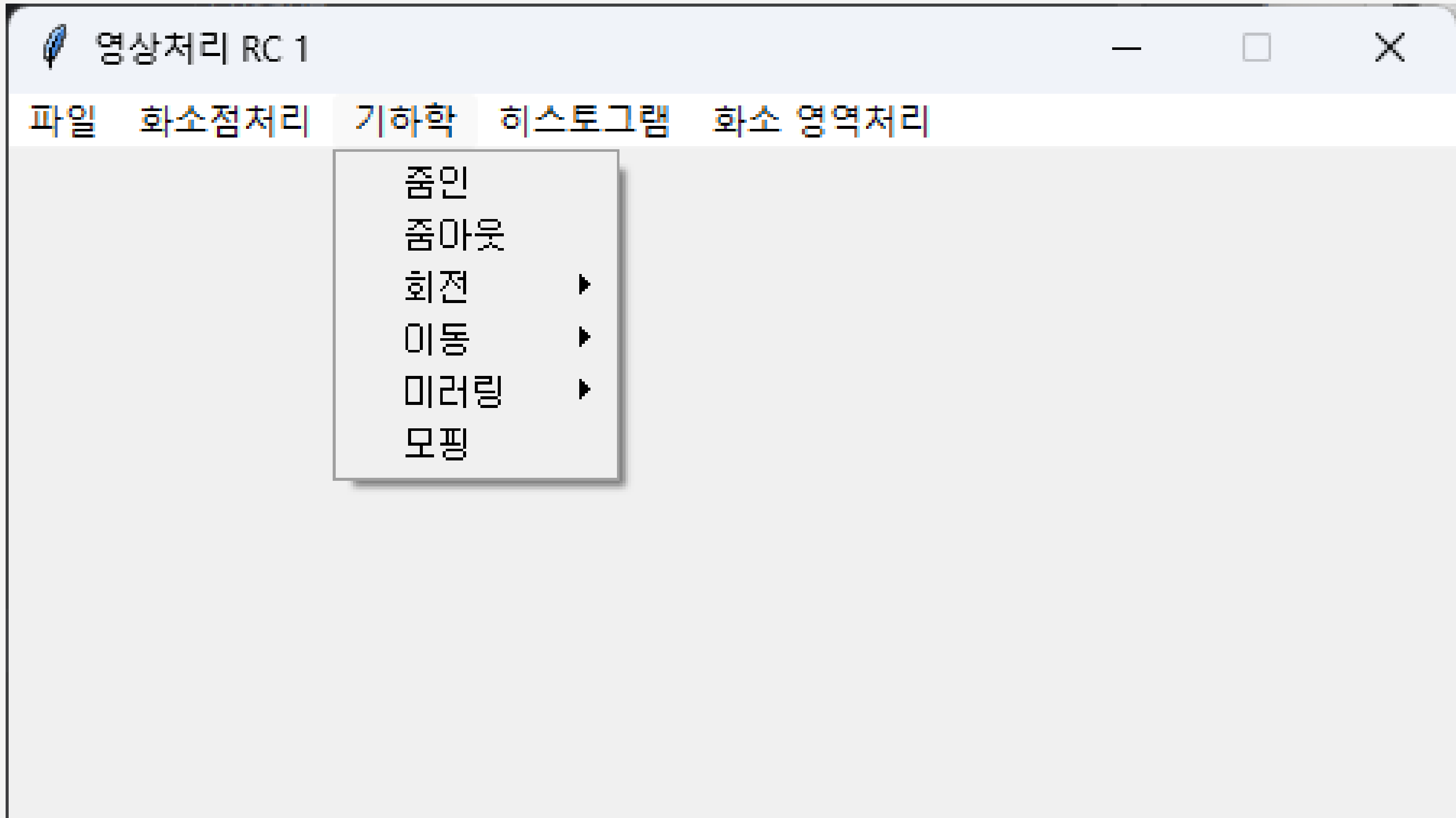
화소점처리 > 포스터라이징



화면 구성 및 기능 화소점처리 > 포스터라이징

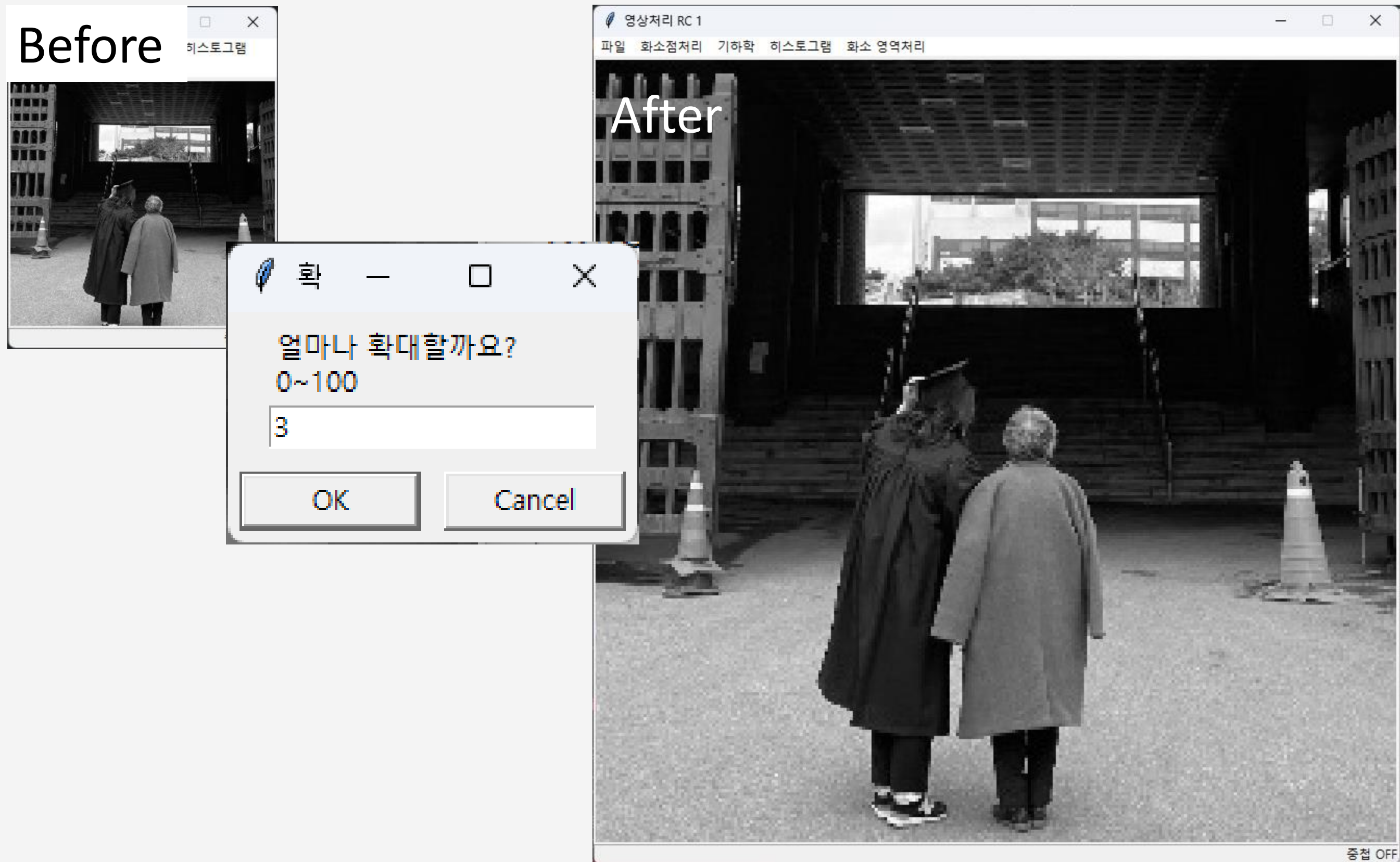


■ 화면 구성 및 기능 기하학 처리 메뉴



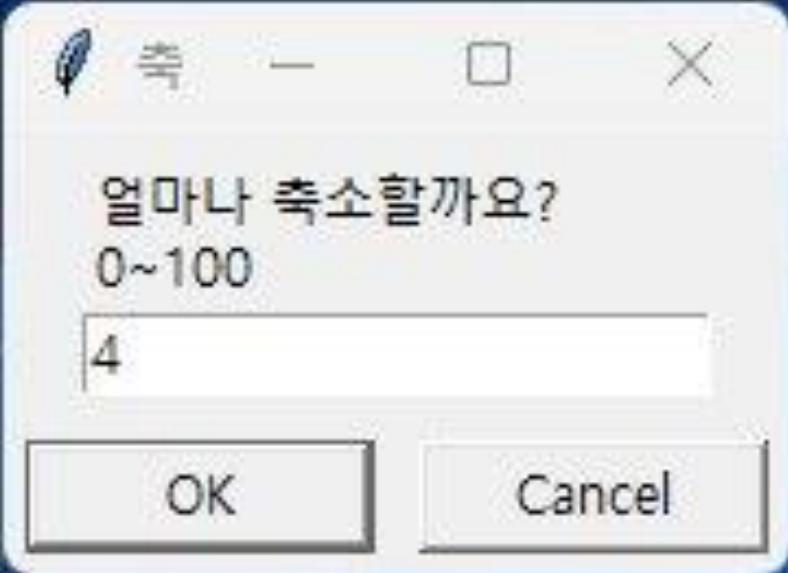
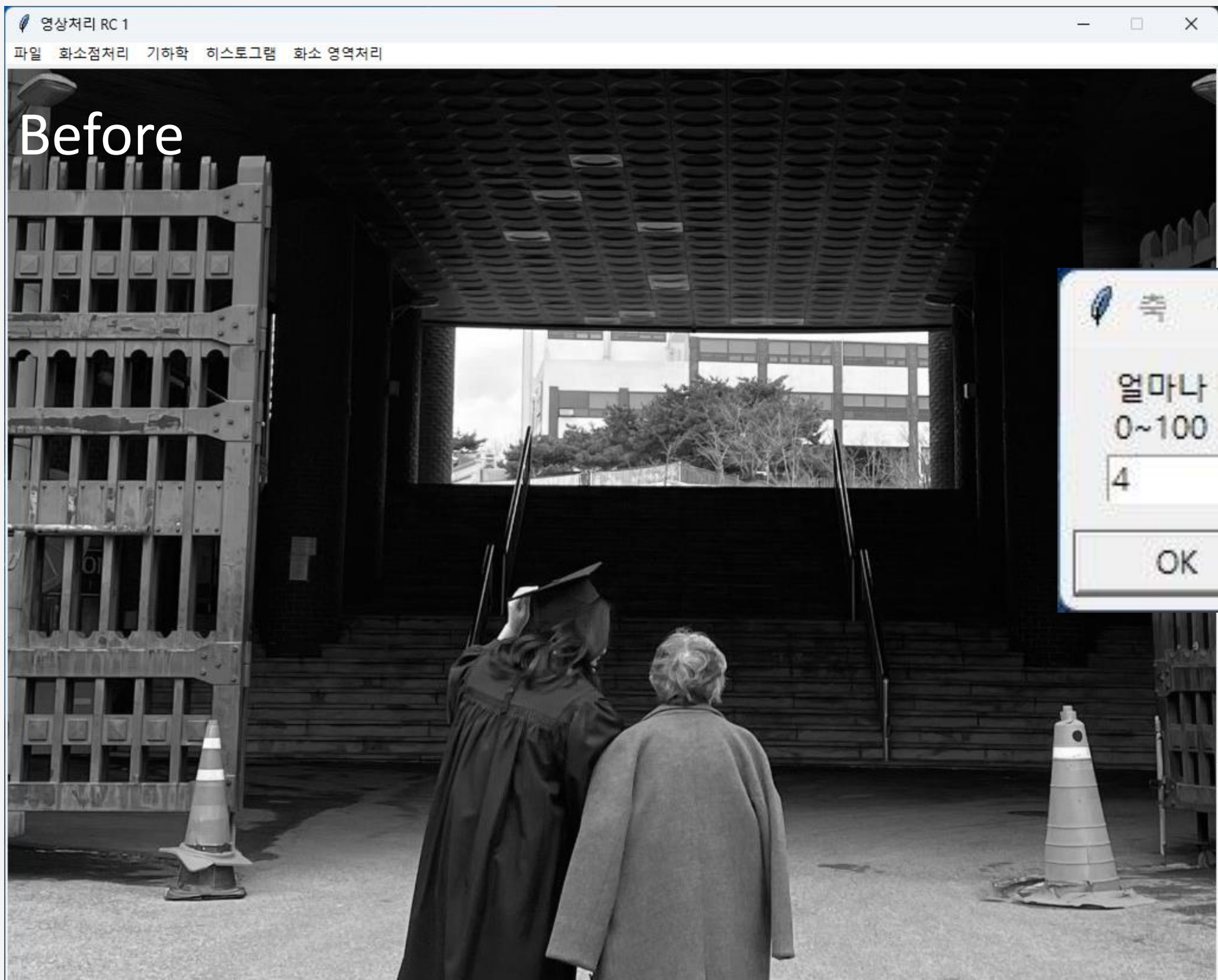
| 화면 구성 및 기능

기하학처리 > 줌인

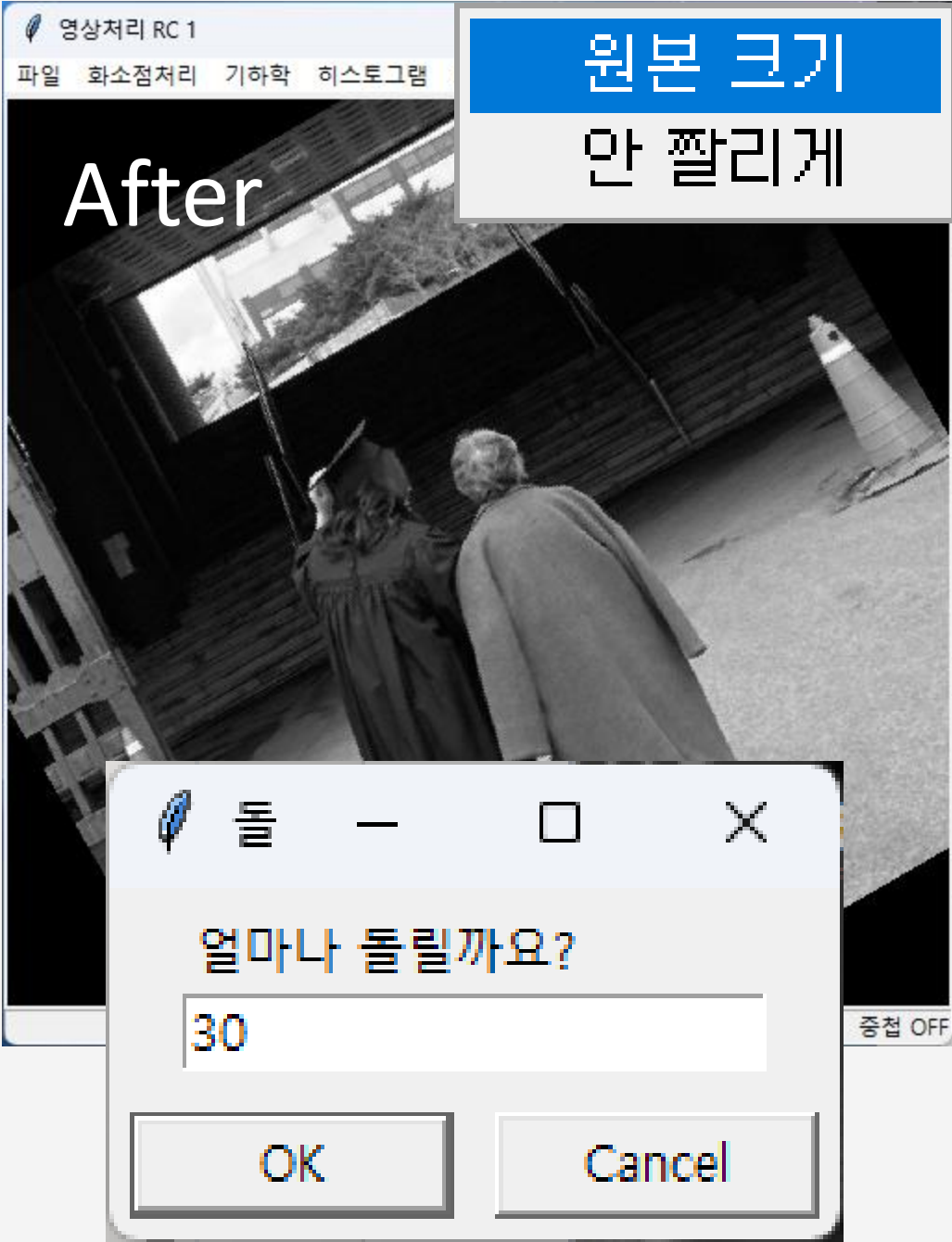


화면 구성 및 기능

기하학처리 > 줌아웃

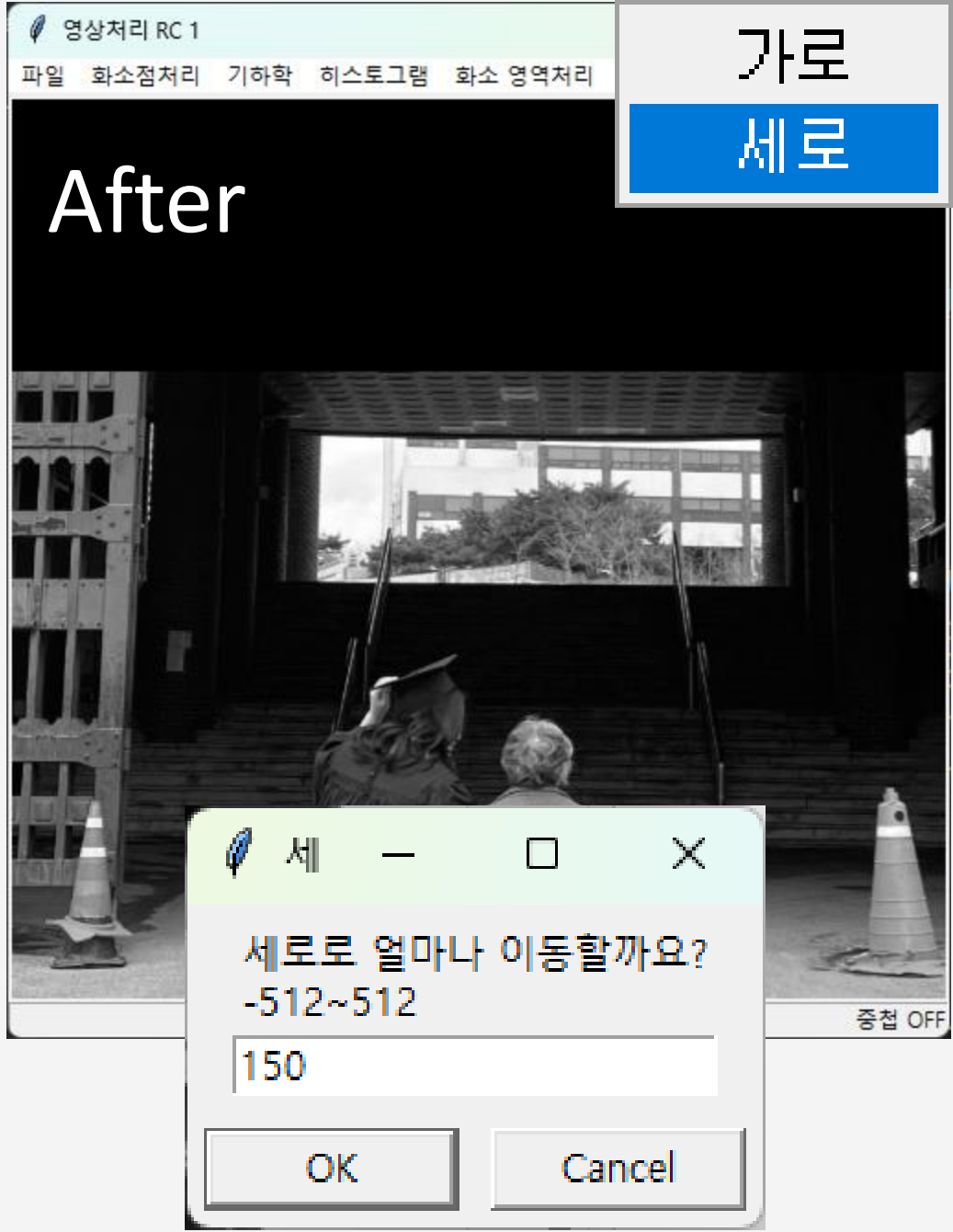


| 화면 구성 및 기능 기하학처리 > 회전



화면 구성 및 기능

기하학처리 > 이동



■ 화면 구성 및 기능



화면 구성 및 기능 기하학처리 > 모핑

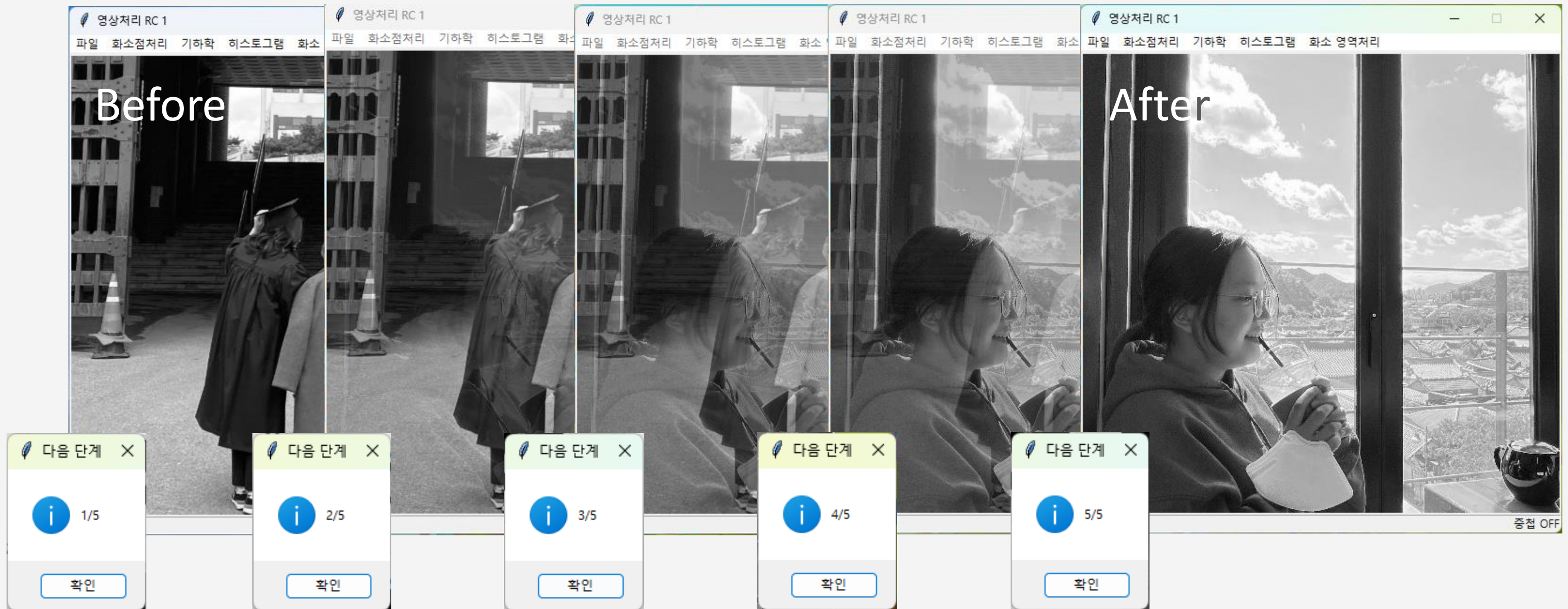
- 줌인
- 줌아웃
- 회전
- 이동
- 미러링
- 모핑

모 - □ ×

몇 단계가 필요한가요?
0~100

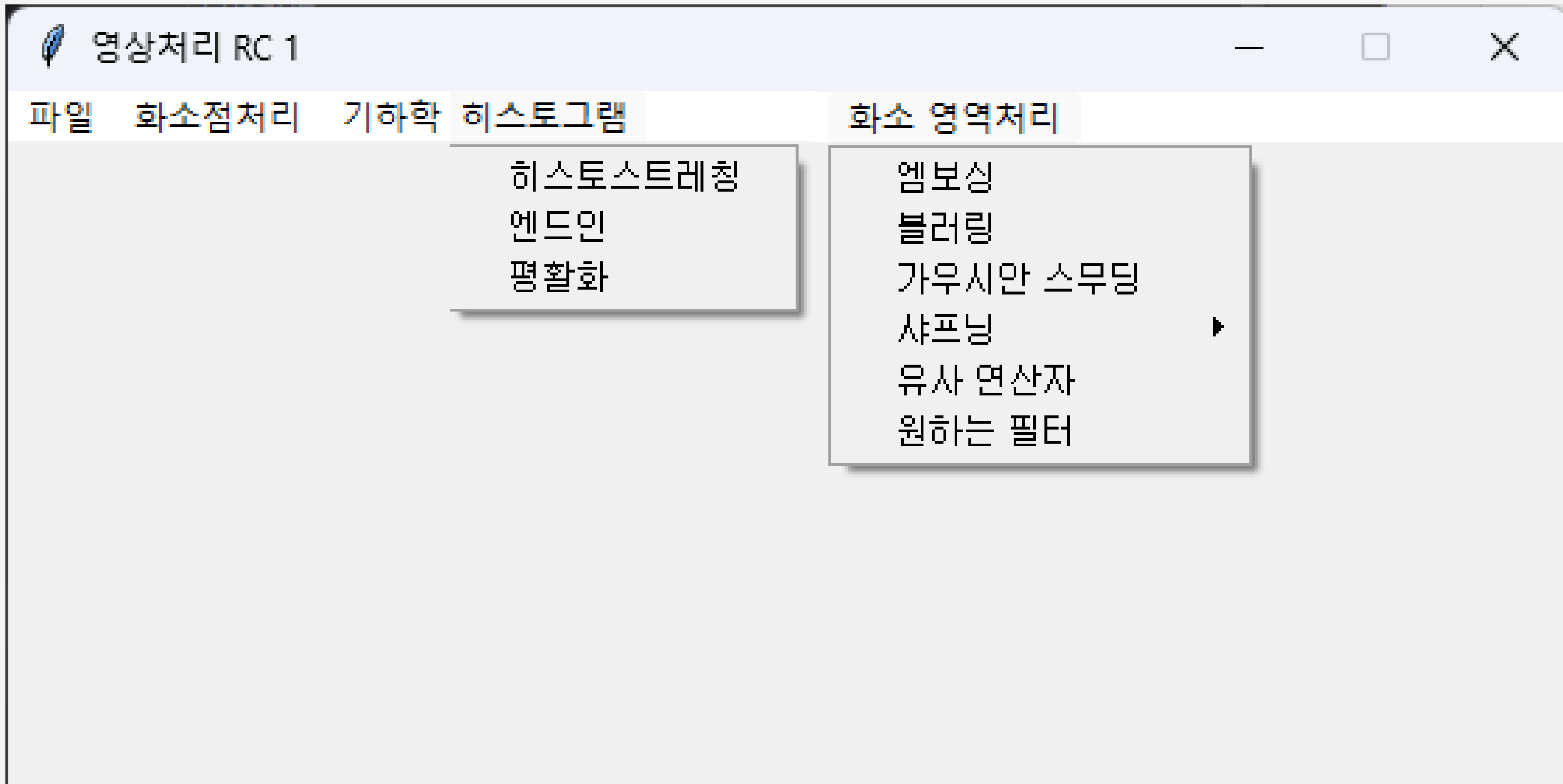
5

OK Cancel



■ 화면 구성 및 기능

히스토그램 & 화소 영역 처리 메뉴



화면 구성 및 기능

히스토그램 > 히스토스트레칭



해당 사진의 픽셀 중
최대 밝기와 최저 밝기를 찾음

```
if (inImage[i][k] < low):  
    low = inImage[i][k]  
if (inImage[i][k] > high):  
    high = inImage[i][k]
```

최대 밝기와 최저 밝기를 기반으로
기존 픽셀 값을 갱신

```
new = (int)((old - low) / (high - low) * 255.0)  
if (new > 255):  
    new = 255  
if (new < 0):  
    new = 0  
outImage[i][k] = new
```


화면 구성 및 기능

히스토그램 > 엔드인



해당 사진의 픽셀 중
최대 밝기와 최저 밝기를 찾음

최대 밝기와 최저 밝기를
상황에 맞게 변경

```
high -= 50  
low += 50
```

구조적으로 히스토스트레칭과 거의 동일

최대 밝기와 최저 밝기를 기반으로
기존 픽셀 값을 갱신

화면 구성 및 기능

히스토그램 > 평활화



각 밝기의 빈도를 일정하게
만드는 처리

밝은 사진은
전체적으로 어두워짐

어두운 사진은
전체적으로 밝아짐

히스토그램 생성
각 픽셀값의 빈도를 셈

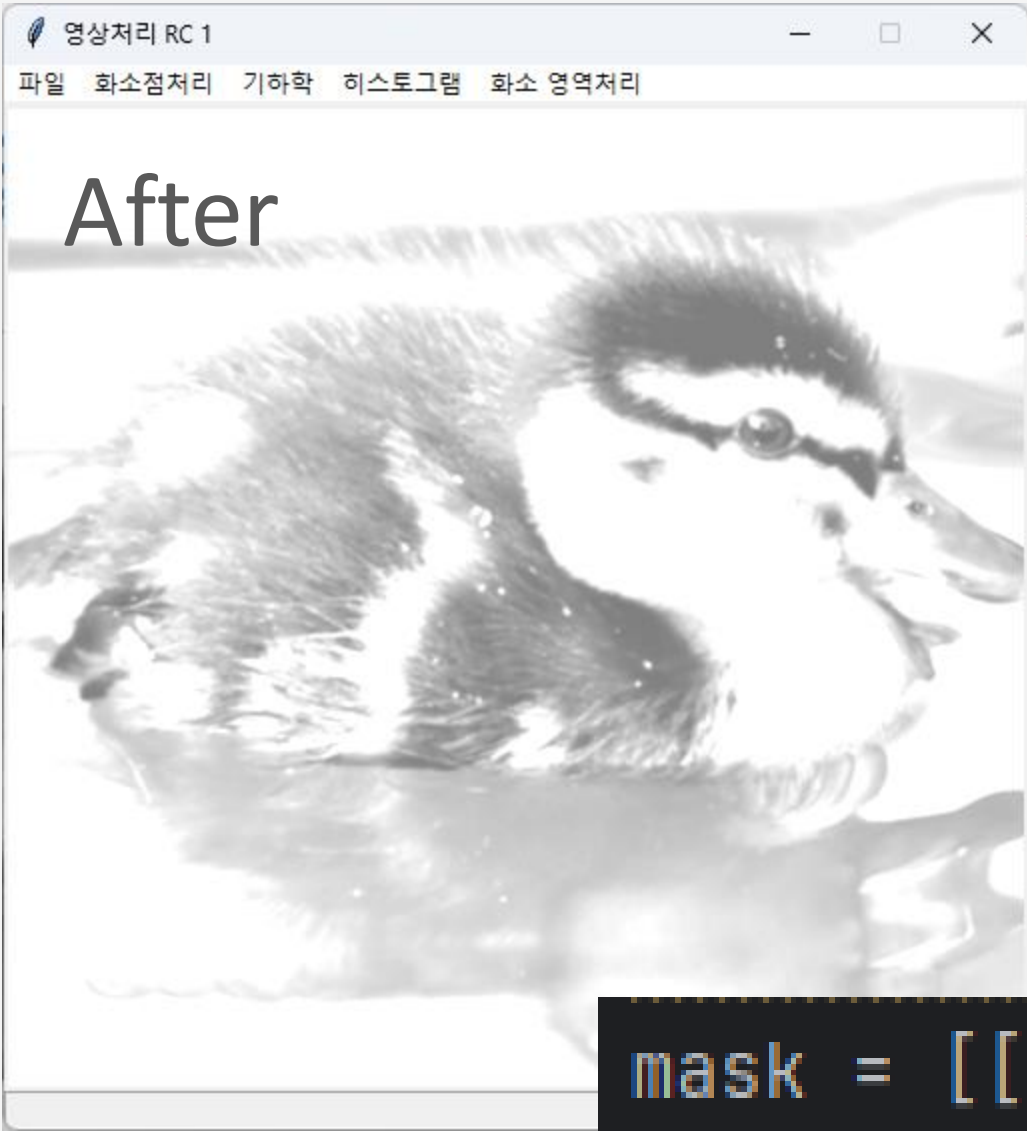


누적 히스토그램 생성
히스토그램 누적합 배열 생성



히스토그램 정규화
누적합 배열을 평균냄

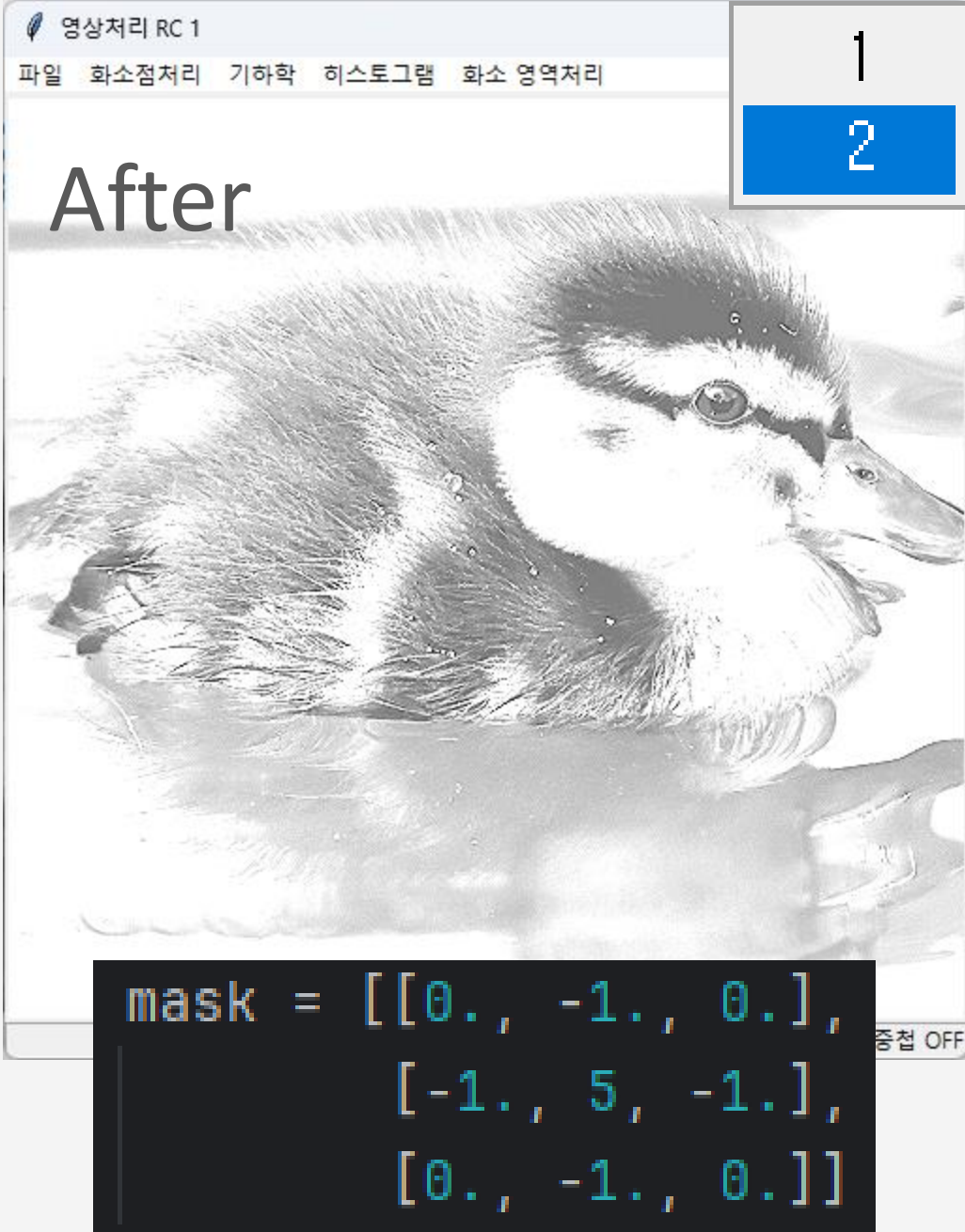
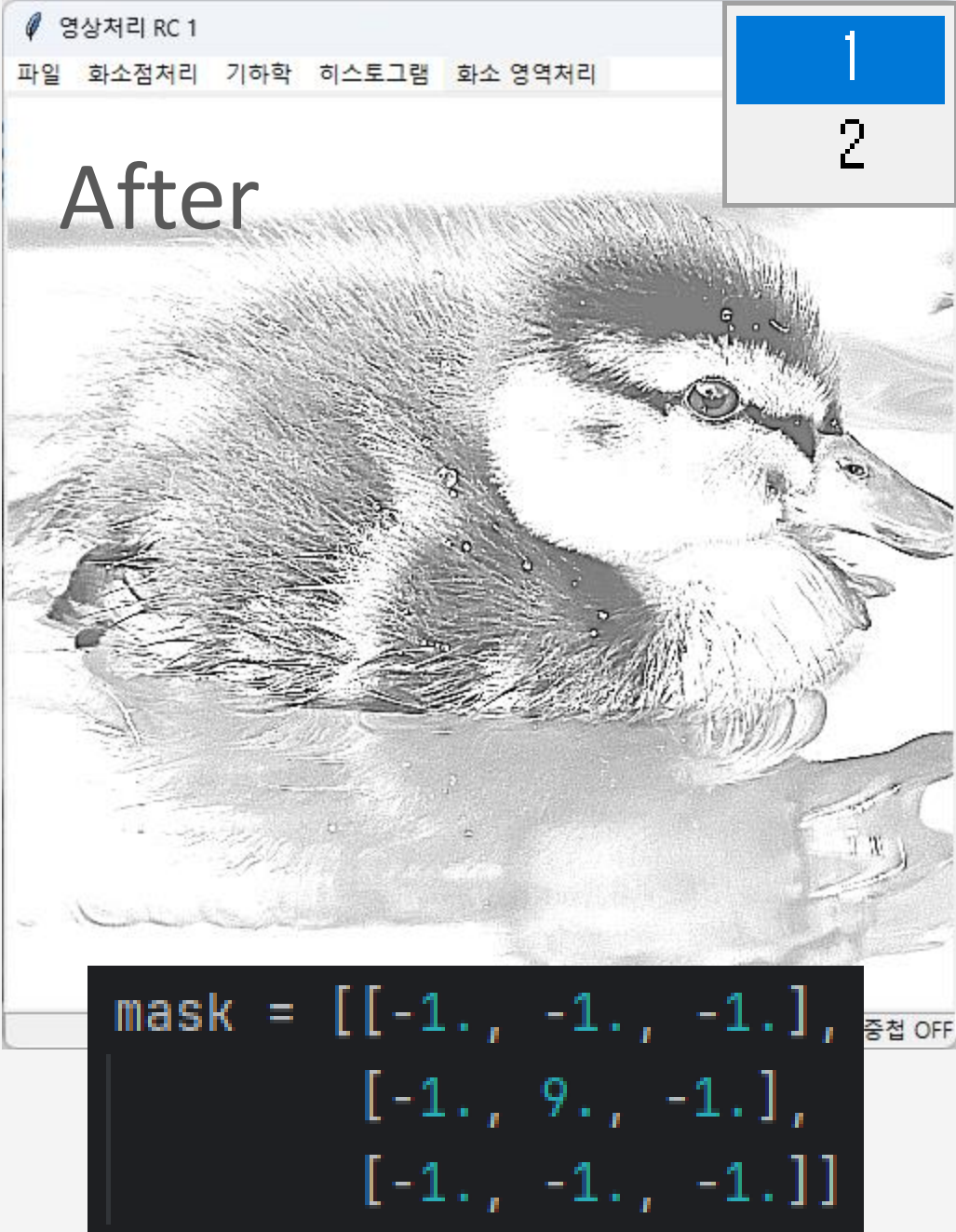
화면 구성 및 기능 화소영역처리 > 가우시안 스무딩



```
mask = [[1. / 16, 1. / 8, 1. / 16],
        [1. / 8, 1. / 4, 1. / 8],
        [1. / 16, 1. / 8, 1. / 16]]
```

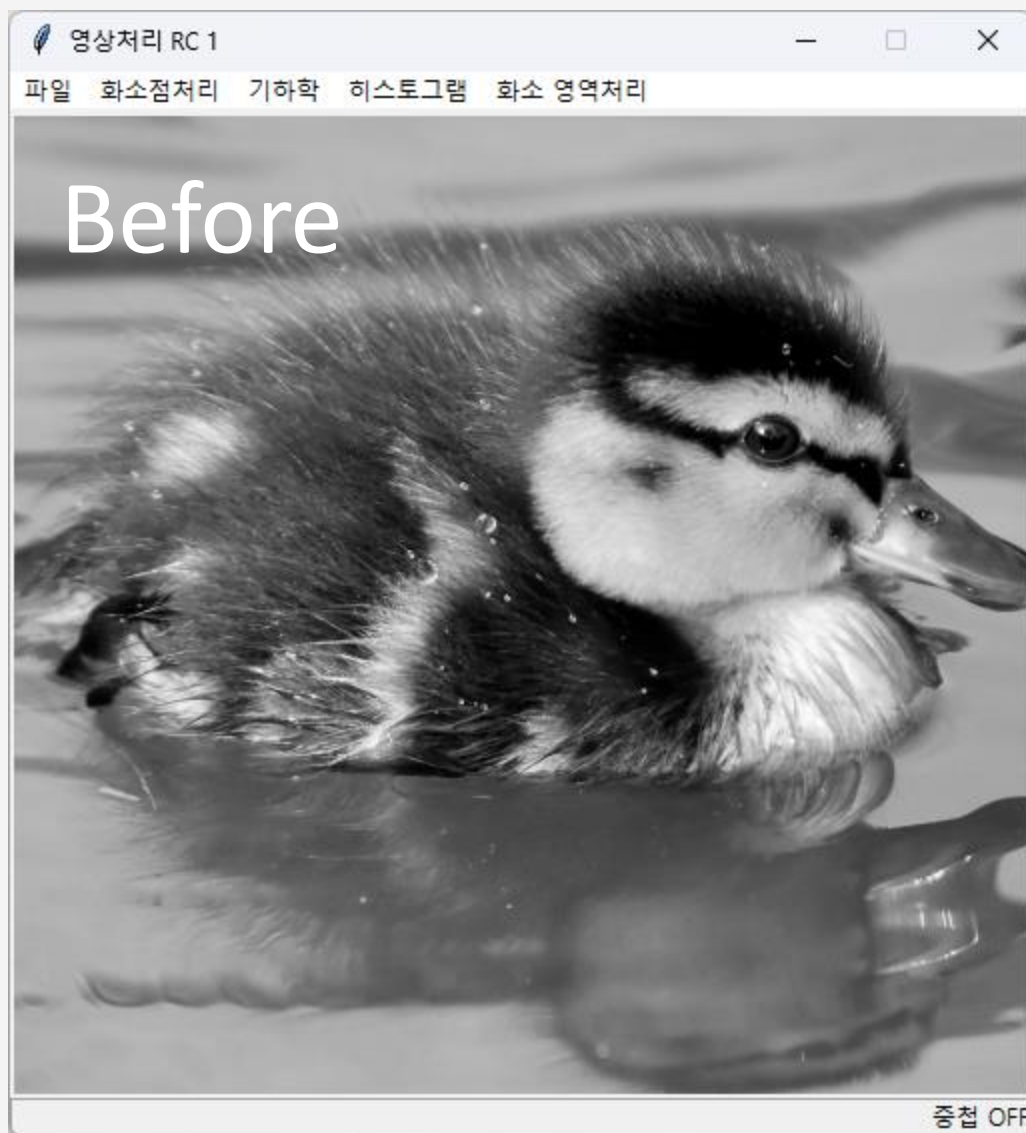
화면 구성 및 기능

화소영역처리 > 샤프닝



화면 구성 및 기능

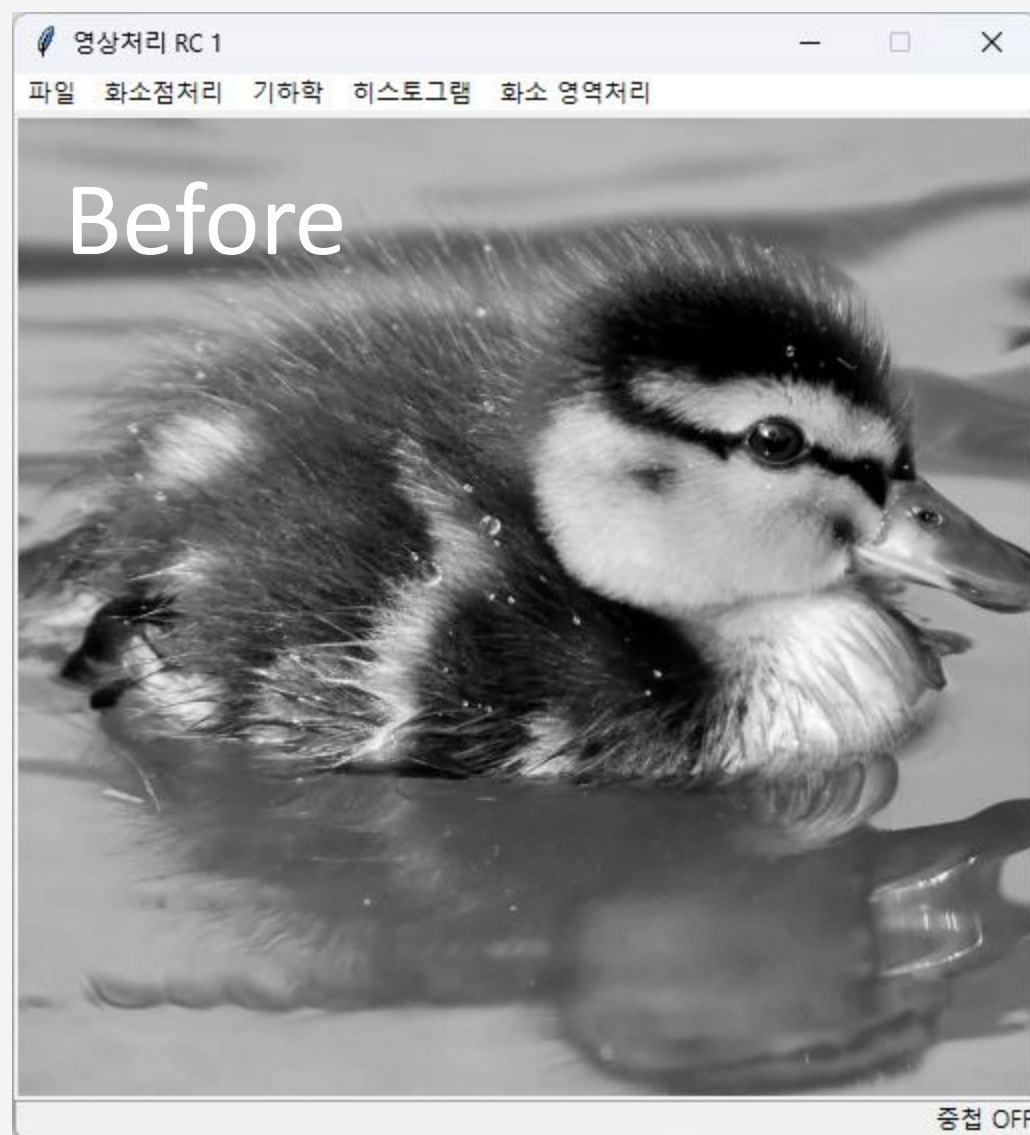
화소영역처리 > 유사 연산자



3x3으로 픽셀 주변의 값과
자신의 차이 중
가장 큰 값의 절댓값을
출력 이미지의 값으로 하는 연산

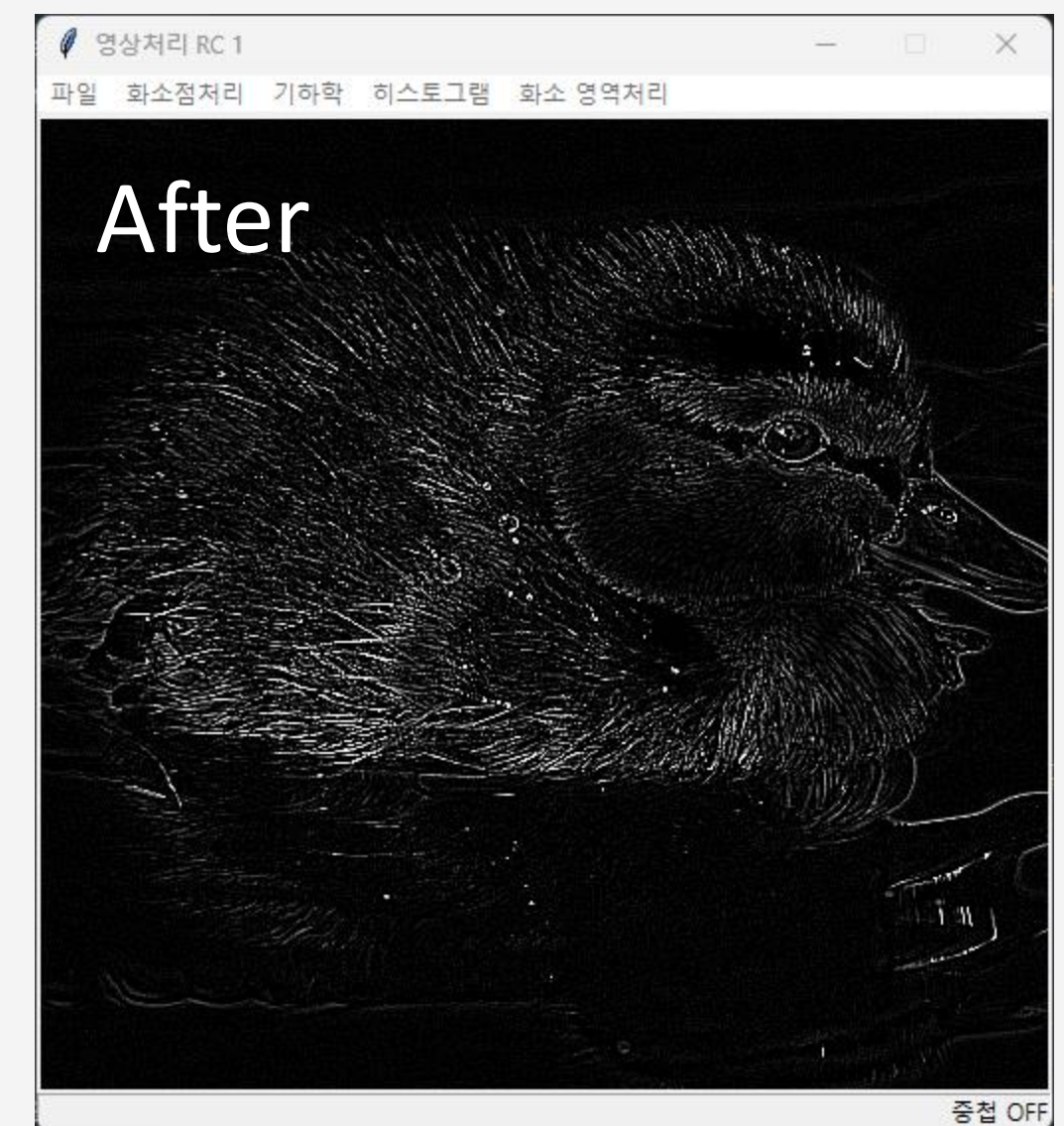
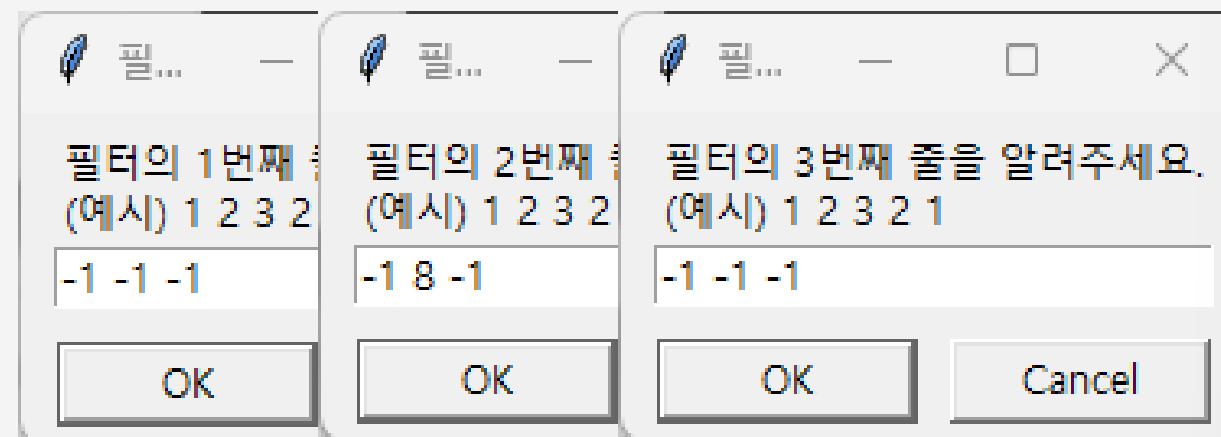
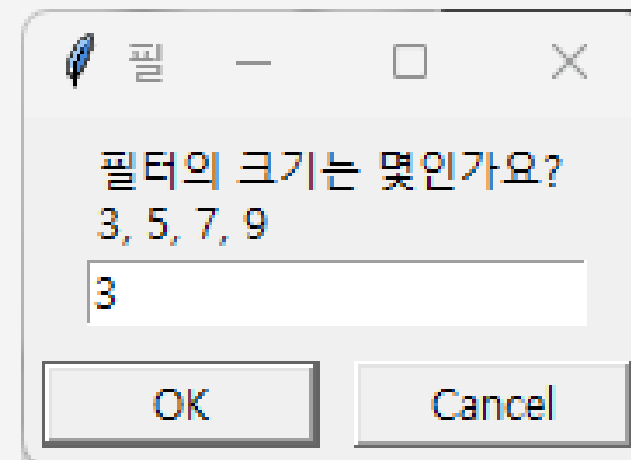
```
S = abs(tmpInImage[i + 1][k + 1] - tmpInImage[i][k])
if ((m != 1) & (n != 1)):
    S = max(abs(tmpInImage[i + 1][k + 1] - tmpInImage[i + m][k + n]), S)
tmpOutImage[i][k] = S
```

화면 구성 및 기능

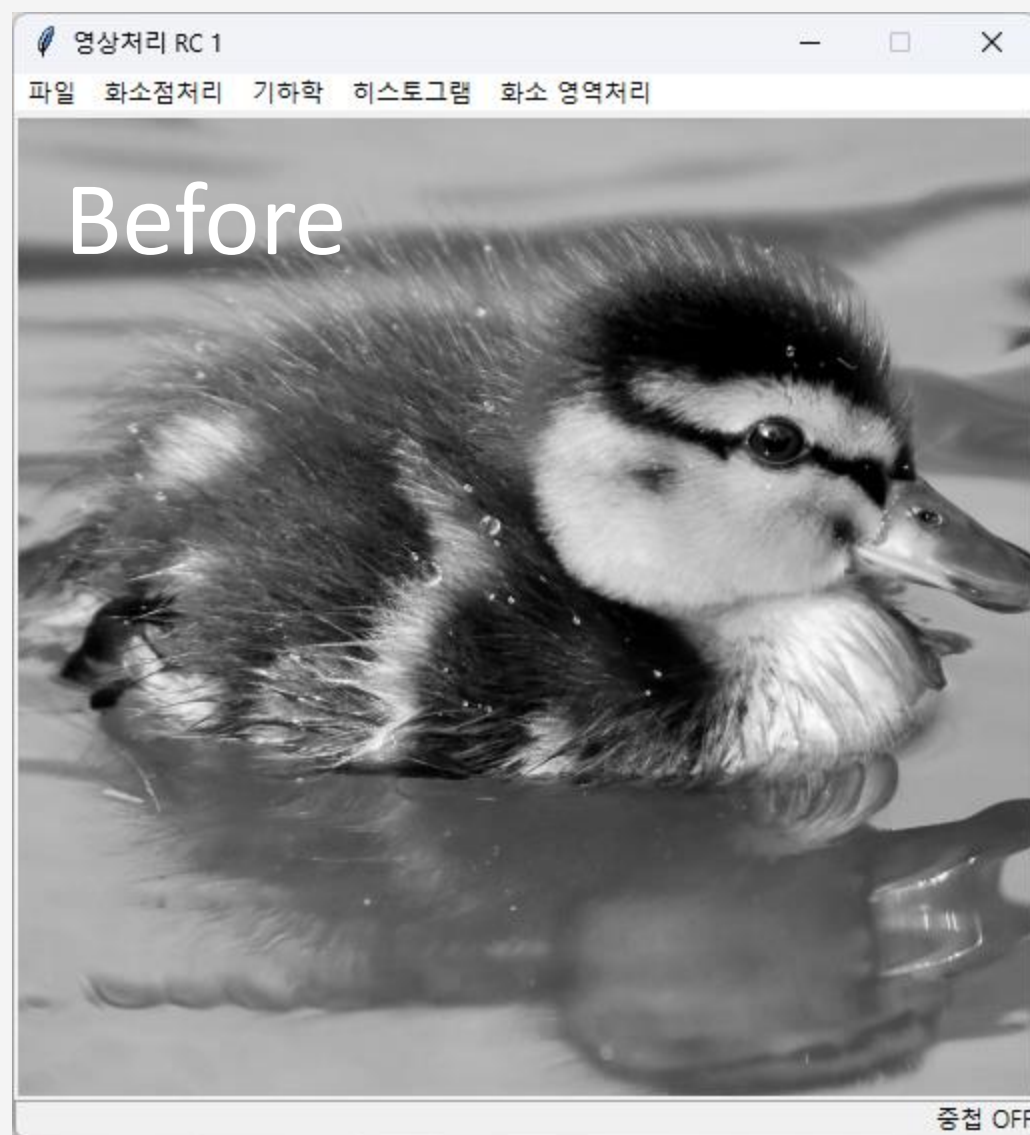


고주파 필터 -


-1	-1	-1
-1	8	-1
-1	-1	-1



화면 구성 및 기능



LoG 필터 -



0	0	-1	0	0
0	-1	-2	-1	0
-1	-2	16	-2	-1
0	-1	-2	-1	0
0	0	-1	0	0

<p>필터의 1번째 줄을 알려주세요. (예시) 1 2 3 2 1</p> <p>0 0 -1 0 0</p>	<p>필터의 3번째 줄을 알려주세요. (예시) 1 2 3 2 1</p> <p>-1 -2 16 -2 -1</p>
<p>필터의 2번째 줄을 알려주세요. (예시) 1 2 3 2 1</p> <p>0 -1 -2 -1 0</p>	
<p>필터의 4번째 줄을 알려주세요. (예시) 1 2 3 2 1</p> <p>0 -1 -2 -1 0</p>	<p>필터의 5번째 줄을 알려주세요. (예시) 1 2 3 2 1</p> <p>0 0 -1 0 0</p>



화면 구성 및 기능

4. 2번과 3번을 필터 크기만큼 반복

0	0	-1	0	0
0	-1	-2	-1	0
-1	-2	16	-2	-1
0	-1	-2	-1	0
0	0	-1	0	0

5. 필터의 합을 확인하고 0이 아닐 경우, 1이 되도록 연산

```
for i in range(scale):
    for k in range(scale):
        sum += mask[i][k]
if (sum != 0):
    for i in range(scale):
        for k in range(scale):
            mask[i][k] /= sum
```

6. 입력 이미지에 필터를 씌움

7. 필터의 합이 0이었다면 각 픽셀에 127

```
if (sum != 0):  
    for i in range(outH):  
        for k in range(outW):  
            tmpOutImage[i][k] += 127.0
```

부가 기능

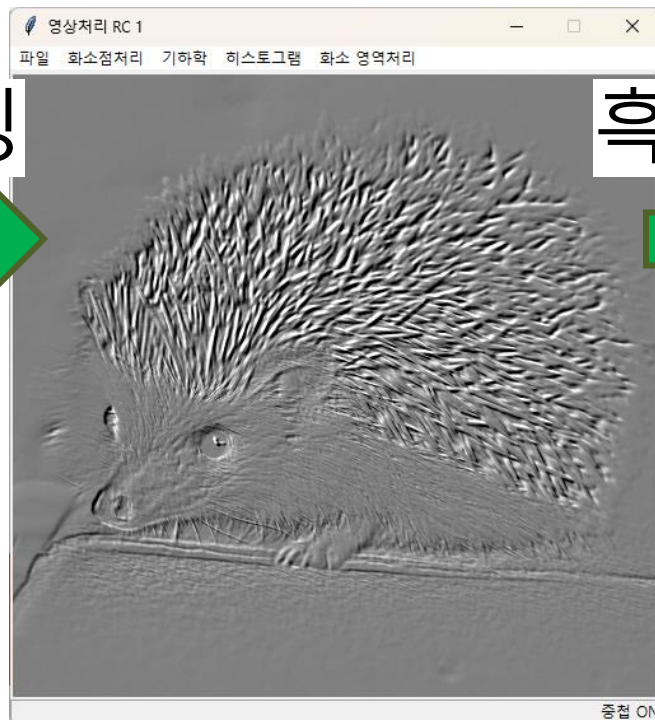


효과 중첩

전역 변수를 통해 효과 중첩을 관리합니다.



엠보싱



흑백(평균)



마치며

좋았던 점

프로젝트에서 파이썬을 사용해본 점

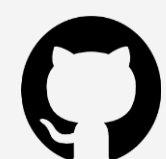
아쉬운 점

기간이 비교적 짧아, None 입력에 예외 처리를 못한 점

추후 계획

되돌리기 기능을 어떻게 추가할지 생각 해보자

장혜원 .



https://github.com/Jang-HW/Intel_Edge_AI_SW_Academy



hw11435@naver.com