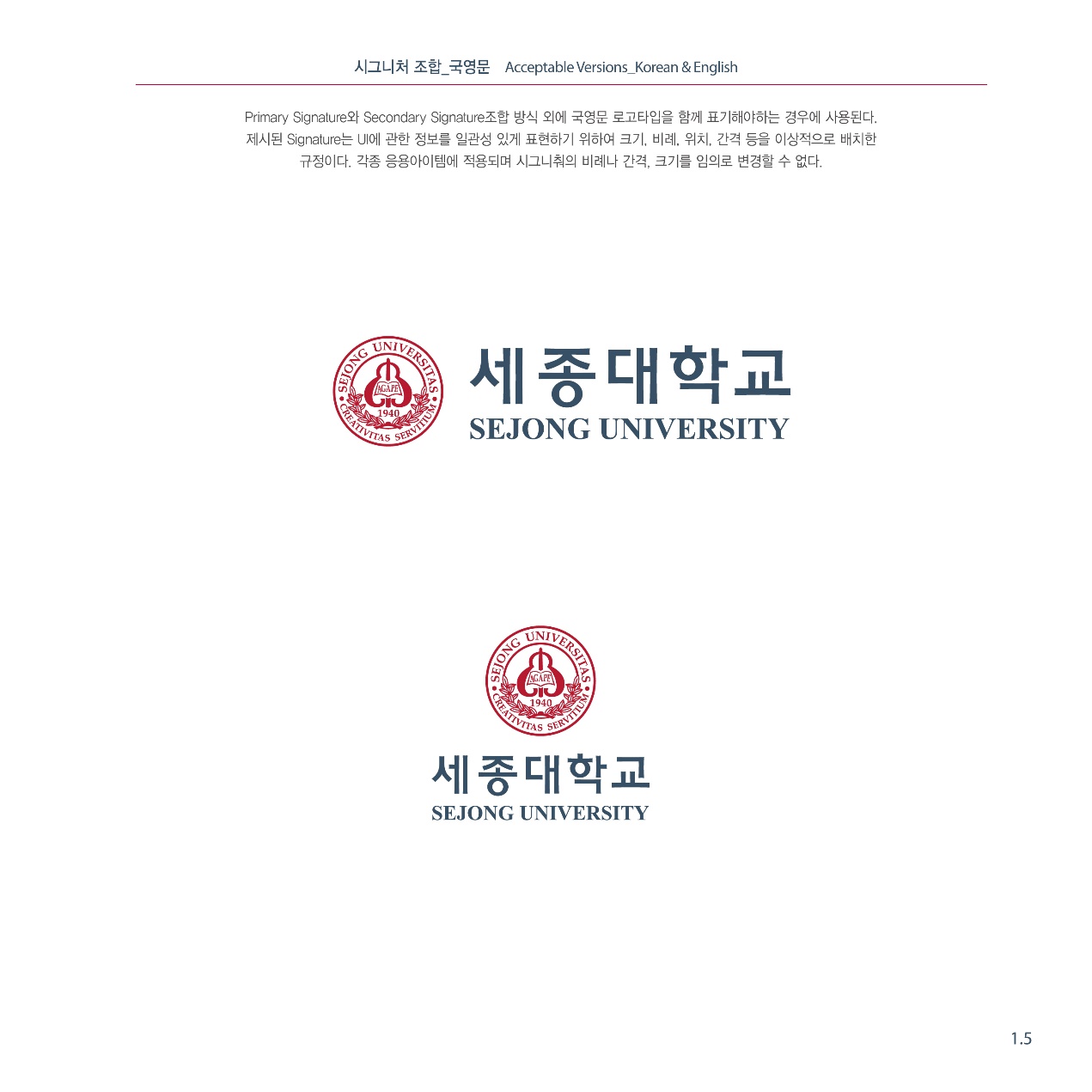


|  |
| --- |
| **멀티미디어 프로그래밍 과제 보고서**  과제4: Painerly Rendering |

|  |  |
| --- | --- |
| 전공 | 소프트웨어학과 |
| 학번 | 21011778 |
| 이름 | 염지환 |
| 제출일 | 2024.05.13 |



**[ 목차 ]**

내용

1. **논문 분석3**
2. **프로그램 구현 과정**3
   1. 구조체, 함수 선언 및 Main 함수 정의3
   2. paint함수 정의 8
   3. paintLayer함수 정의9
   4. makeStroke함수 정의10
   5. makeSplineStroke 함수 정의11
3. **결과**15
4. **논문 분석**

주어진 논문은 사진 이미지를 마치 화가가 그린 그림처럼 표현하도록 하는 Painterly Rendering에 대해 다루고 있다. 이 논문에서는 그림의 요소를 각기 다른 Brush Size를 가진 붓들을 이용한 Brush Stroke, 즉 붓질로 보고. 큰 Brush Size를 가진 붓부터 작은 Brush Size를 가진 붓 순으로 무작위 순서로 붓질을 그려내 그림을 그려내려고 한다.

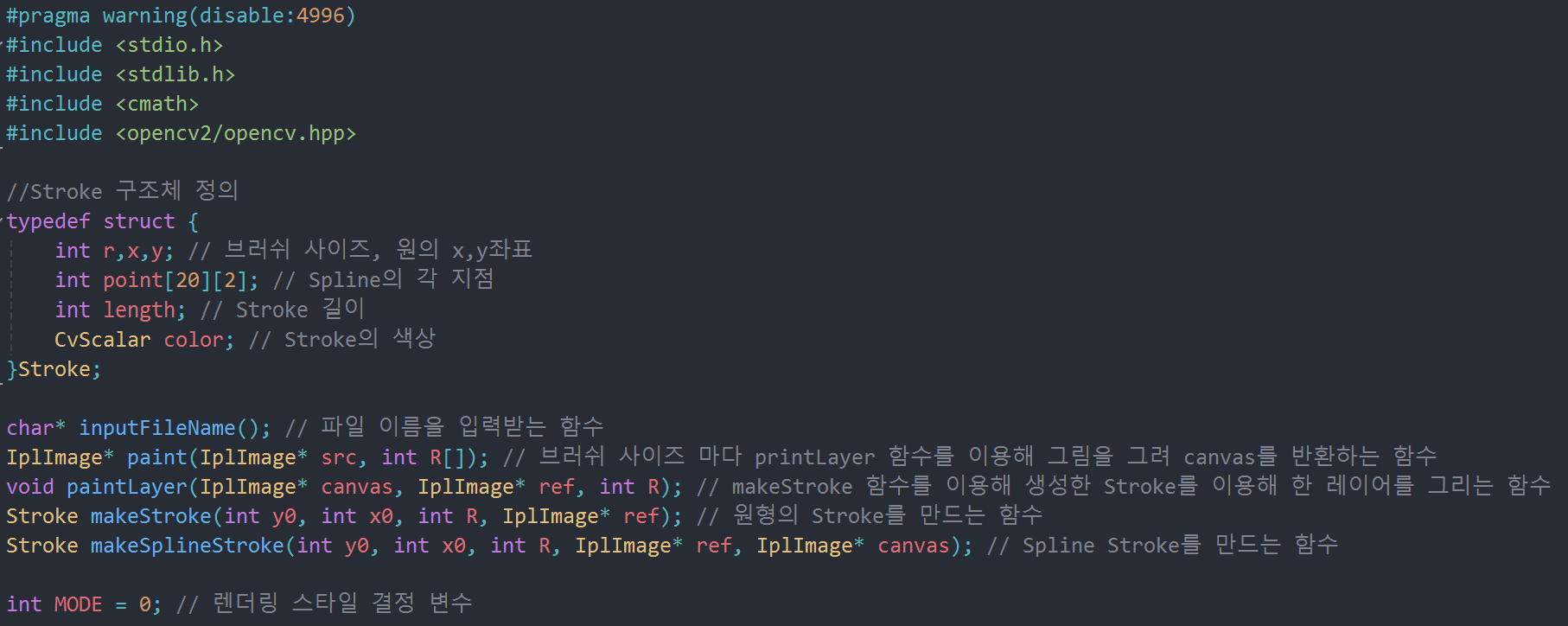
2.1에선 Brush Size 선언과 paint함수, paintLayer 함수에 대해 다루고 있으며 paint 함수는 각 Brush Size마다 Brush Size에 비례하는 크기의 커널 사이즈로 가우시안 필터링을 진행한 reference 이미지를 만들어 paintLayer함수에 매개변수로 주어 해당 Brush Size를 가진 Stroke들을 이용해 그림을 그린다. paintLayer함수는 paint 함수에서 생성된 canvas 이미지와 reference 이미지, 그리고 Brush Size를 매개변수로 받아 호출되며 Stroke의 배열 S를 만들고, canvas와 reference이미지의 오차를 저장할 배열 D를 만든다. 그리고 가상의 그리드를 만드는데, 이때 한 그리드 칸의 크기는 Brush Size에 비례하도록 만든다. 이후 각각의 그리드를 순회하며 그리드 안에서 가장 오차가 큰 지점을 Stroke를 그릴 지점으로 정하고 그리드의 오차의 평균이 임계값 T를 넘어갈 경우 makeStroke 함수를 통해 S에 Stroke를 추가한다. 모든 그리드의 순회가 끝나면 무작위 순서로 S에 저장된 Stroke들을 통해 canvas에 원, 또는 Spline Stroke를 그린다.

예시 프로그램에서 원을 이용한 그림 스타일은 지금까지의 내용으로 구현할 수 있으며 곡선을 만들기 위해서는 makeStroke 함수에 더 복잡한 기능이 필요하다. 이 논문의 2.2 부분에선 어떻게 해야 Curved Brush Stroke를 구현할 수 있는지 설명한다. 먼저 스트로크의 색상은 reference 이미지 중 paintLayer함수에서 매개변수로 받는 좌표의 픽셀에서 가져온다. 그리고 Spline Stroke의 점의 집합에 매개변수로 받은 좌표를 추가하고, 현재 좌표 역시 매개변수로 받은 좌표로 초기화 한다. 그리고 직전 포인트의 (dx,dy)를 저장하기 위한 lastDx, lastDy를 정의한다. 이후 반복문을 통해 Spline Stroke의 직선을 연결할 점들을 생성한다. 먼저 현재 지점에서 reference 이미지와 canvas 사이의 색상 오차가 reference 이미지와 Stroke 색상의 색상 오차보다 작을 경우, 즉 Stroke보다 canvas가 색상이 reference와 더 유사할 경우 함수를 종료한다. 또한 현재 좌표를 기준으로 reference에서 현재 픽셀의 상 하, 좌 우 색상 변화가 0일 경우, 즉 변화가 없을 경우 역시 함수를 종료한다. 이후 픽셀의 상 하, 좌 우 색상 변화량을 각각 gy, gx라고 하여 가상의 벡터 (gx, gy)를 정의한다. 이 벡터의 방향은 색상이 더 밝게, 더 많이 변하는 방향이다. 우리는 Spline Stroke를 만들 때 색상이 비슷한 부분을 선으로 이어야 하므로, 색상이 변하는 방향을 가진 벡터 (gx,gy)에 수직인 벡터를 구하여 색상이 변하지 않는 방향의 벡터 또한 얻을 수 있다. 색상이 변하지 않은 방향의 벡터를 (dx,dy) 라고 했을 때 dx = -gy, dy = gx로 정의하여 (dx, dy)를 (gx, gy)에 수직인 벡터로 정의해 주었다. 이때 (dx, dy)가 Stroke 진행 방향의 거의 정 반대 방향일 경우 (dx, dy)를 (-dx, -dy)로 정의하여 반향을 반대로 바꿔준다. 그리고 (dx, dy)와 (lastDx, lastDy)를 특정 비율로 혼합하여 곡률을 바꿔주는 작업을 한다. Spline Stroke의 점과 점 사이의 거리는 Brush Size로 결정되므로 벡터의 크기가 필요 없기에 (dx, dy)를 단위벡터 로 바꾸어 준다. 그리고 x와 y를 (dx, dy)에 Brush Size를 곱한 만큼 이동한 지점을 점들의 배열에 저장한다. 모든 점들의 저장이 끝나면 Stroke를 반환한다.

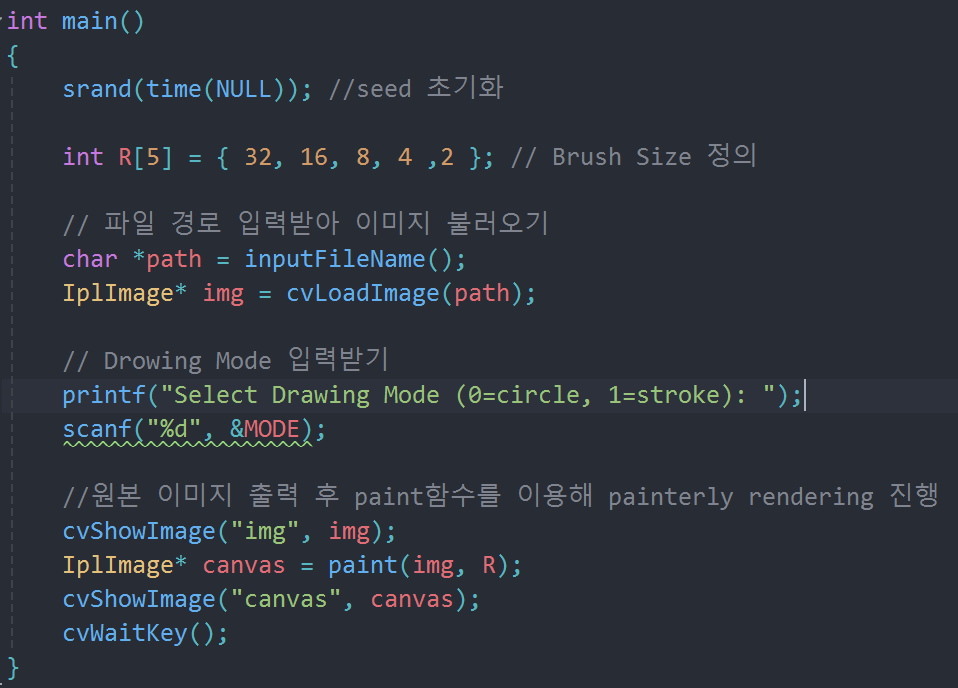
마지막으로 논문의 3.1 에선 그림 스타일을 바꾸는 몇가지 매개변수들을 설명해 두었다. T는 임계값으로 오차가 얼마 이상일 때 Stroke를 생성할 건지 정하는 값이다. Brush Size는 말그대로 Stroke가 그려질 때 반지름이다. Curvature Filter, fc는 2.2에서 곡률을 결정할 때 쓰인다. Blur Factor는 paint 함수에서 Brush Size의 몇배 사이즈의 커널로 가우시안 필터링을 진행시킬 지 결정하는 변수이다. Stroke Length의 최소값과 최댓값도 스타일을 변경시키는 매개 변수이며, Grid Size, fg는 그리드 하나의 크기가 Brush Size의 몇배인지 결정하는 변수이다. 이런 매개변수들을 변경하여 그림의 스타일을 변경할 수 있다.

1. **프로그램 구현 과정**

I. 구조체, 함수 선언 및 Main 함수 정의

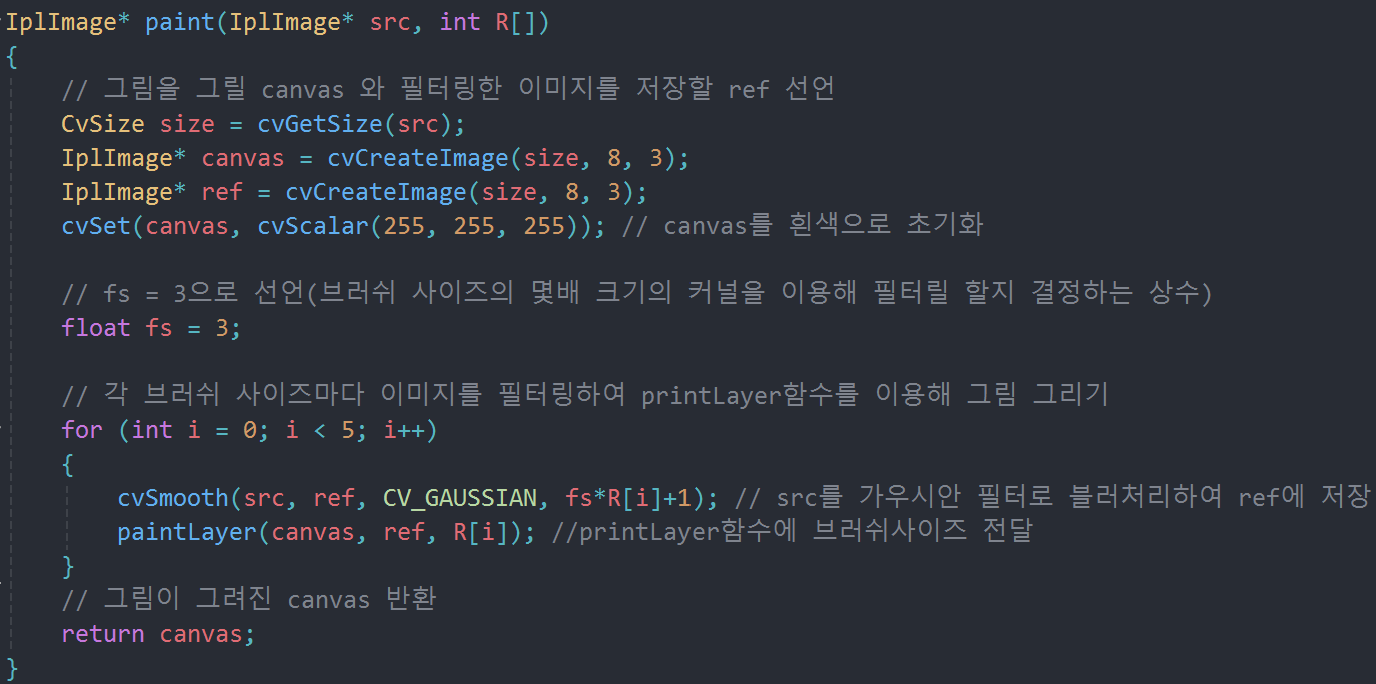


먼저 프로그램을 구현하는데 필요한 구조체와, 함수들을 정의하였다. Stroke를 구조체로 정의하여 Brush Size를 저장할 변수 r, 그리고 Stroke의 위치를 저장할 변수 x와 y, 그리고 Stroke의 색상을 저장할 CvScalar 변수를 선언하였다. 처음 원형 Stroke만을 이용해 이미지를 그릴 때는 위의 4가지 변수만 사용하여 Stroke를 구성했지만 Spline Stroke를 이용해 이미지를 그리기 위해선 spline을 구성하는 점들을 저장할 배열과 spline의 점들의 개수, 즉 Stroke의 길이가 필요했다. 따라서 좌표들의 배열 point와 Stroke의 길이 length를 추가로 정의해 주었다



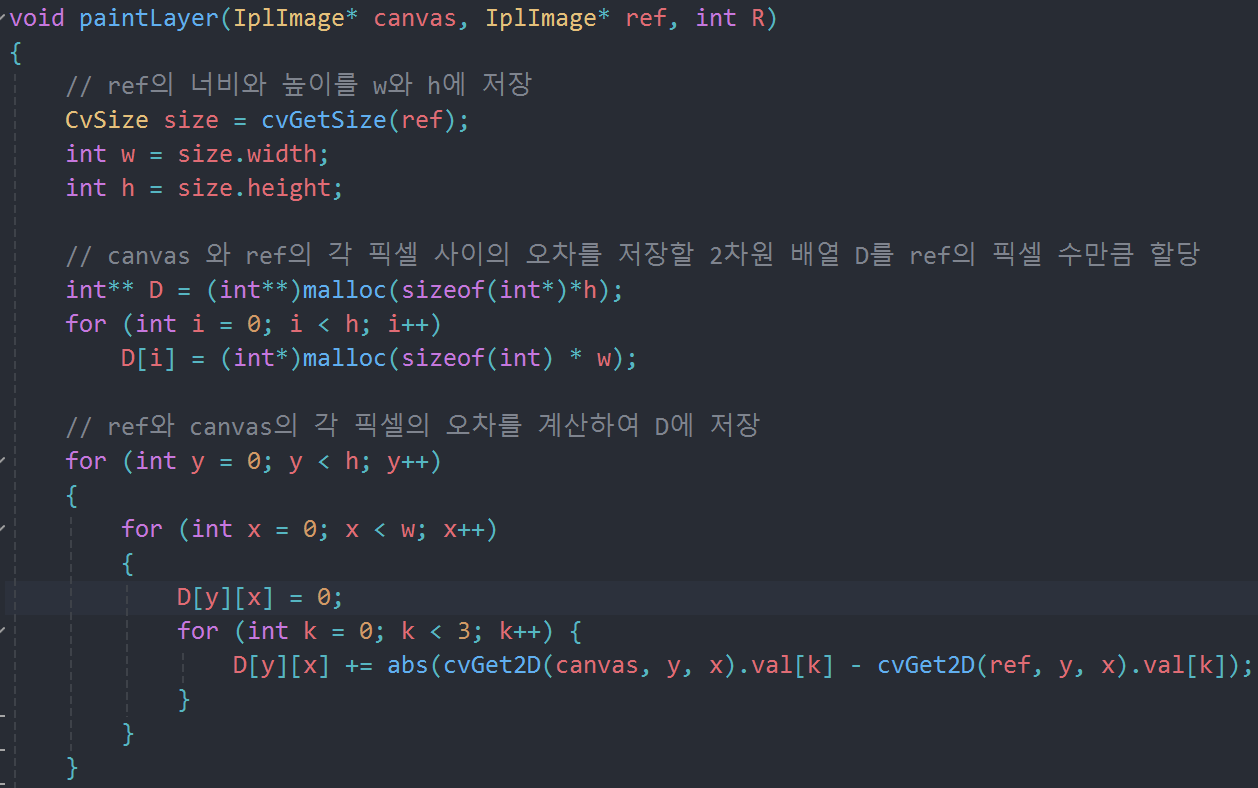
main함수에서는 먼저 무작위로 Stroke를 그릴 때 rand()함수를 사용하기 위해 srand()를 사용하여 시드를 변경 해주었다. 그 후 Brush Size를 지정해주었다. 주어진 예시 프로그램과 최대한 유사하게 동작하도록 각각 32, 16, 8, 4, 2로 정해주었다. 다음으론 그림의 스타일을 변경하는 변수인 MODE를 입력받아 사용자가 모드를 선택할 수 있도록 하였다. 그 후 inputFileName을 이용해 파일 경로를 입력받아 원본 이미지 img를 불러왔다. 그 후 img를 출력하고 paint 함수를 이용해 렌더링이 끝난 함수를 canvas에 저장하여 출력한다.

II. paint함수 정의

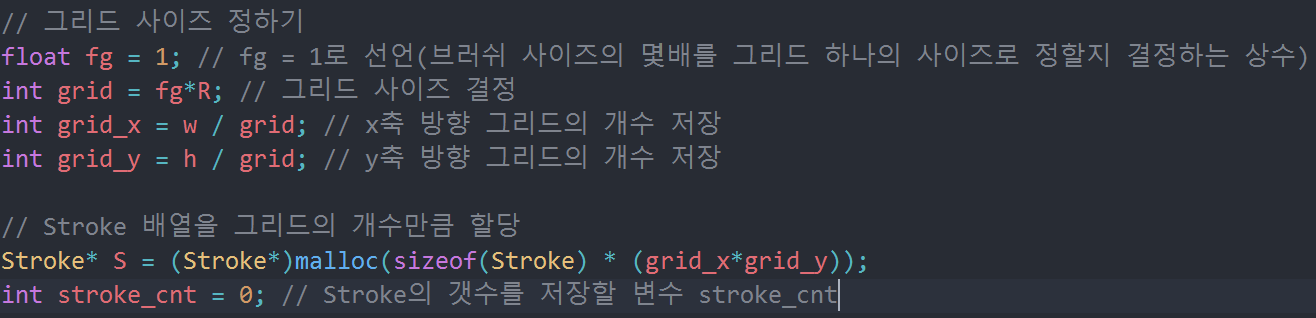


Paint 함수는 원본 이미지 src와 Brush size 배열 R을 전달받아 호출된다. 먼저 그림을 그릴 이미지인 canvas와 원본을 필터링하여 저장할 ref 이미지를 원본 이미지와 같은 크기로 생성해주고 canvas를 흰색으로 초기화해주었다. 그 후 가우시안 필터의 커널 크기를 Brush Size의 몇 배로 할지 결정하는 상수인 fs를 선언했고 위 소스코드에서는 3으로 정의했다. 그 후 반복문을 통해 각 Brush Size와 그에 맞게 필터링된 ref 이미지와 canvas 이미지를 매개변수로 하여 printLayer 함수를 호출한다. 마지막으로 반복문이 끝나면 완성된 이미지 canvas를 반환한다.

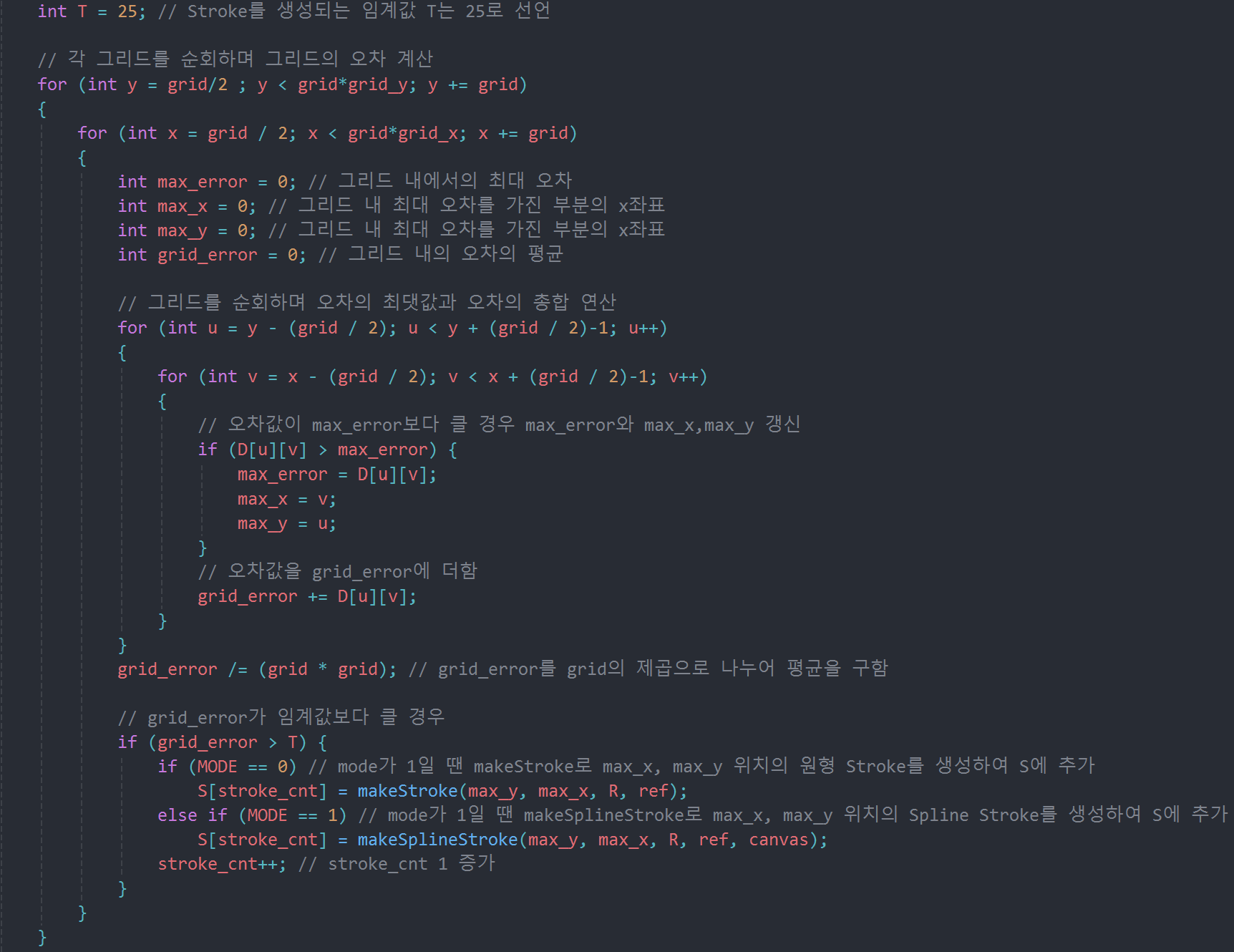
III. paintLayer함수 정의

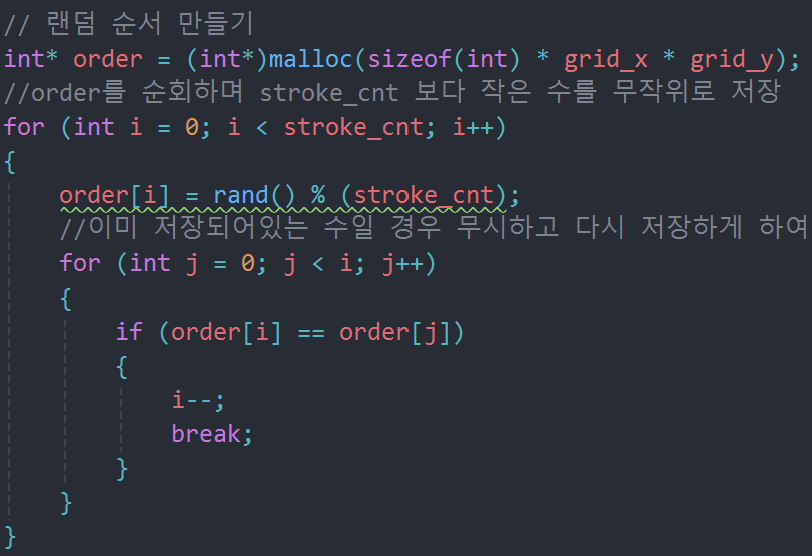


paintLayer함수는 canvas와 ref 이미지, 그리고 Brush size를 매개변수로 받아 호출된다 먼저 원활한 이미지 처리를 위해 ref의 가로와 세로 길이를 각각 w와 h에 저장한다. 그리고 canvas와 ref의 각 픽셀 사이의 오차를 저장하기 위한 2차원 int 배열 D를 이미지의 크기에 맞게 동적으로 할당해준다. 그 후 각 ref와 canvas를 순회하며 각 픽셀의 오차를 계산하여 D에 저장한다.

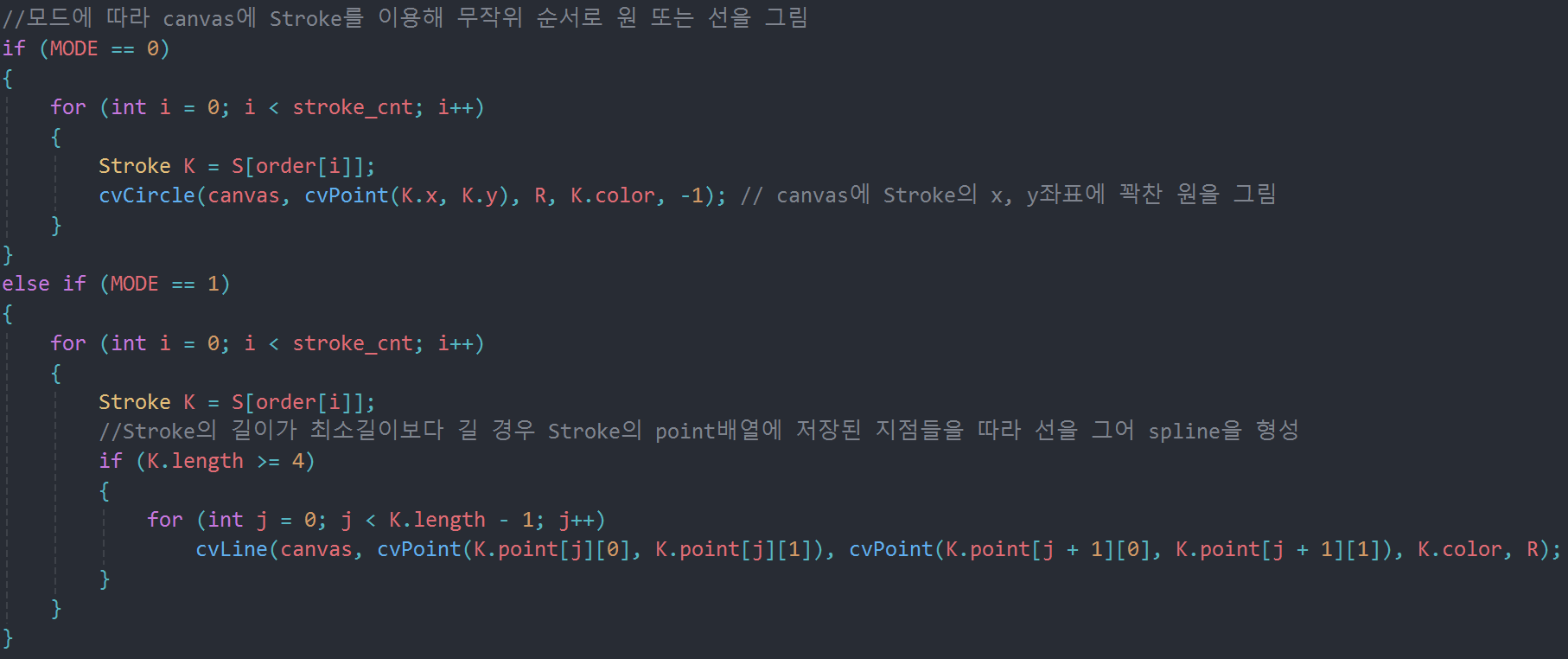


그 후 가상의 그리드를 생성하기 위해 그리드 사이즈를 정한다. fg는 그리드의 크기를 Brush size의 몇배로 할 지 결정하는 상수로 위 소스코드에서는 1로 설정하였다. 그 후 grid에 R에 fg를 곱한 값을 저장하여 그리드의 크기를 저장해주고, w와 h를 grid로 나누어 그리드의 가로 칸의 개수를 grid\_x, 세로 칸의 개수를 grid\_y로 저장한다. 그리고 Stroke배열 S를 동적으로 할당하는데 이때 크기는 grid\_x \* grid\_y로 그리드의 전체 칸의 개수만큼 할당한다.

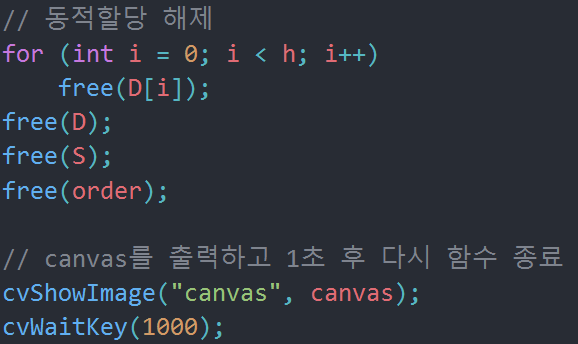
다음으로는 각 그리드를 순회하며 앞서 만든 2차원 배열 D를 이용해 그리드 내에서 ref와 canvas사이의 오차값이 가장 큰 지점의 좌표를 찾아내고, 그리드 내 픽셀의 오차들을 전부 더해 그리드의 총 픽셀 수로 나누어 그리드의 평균 오차를 구했다. 이렇게 구한 평균 오차가 임계값 T보다 클 경우에 Stroke를 생성하도록 하였다. 위 소스코드에서는 임계값을 25로 정의하였다. Stroke를 생성할 때 MODE가 0,일 경우 원형 스트로크를 만드는 makeStroke를 이용해 Stroke를만들었고, MODE가 1일 경우 Spline Stroke를 만드는 makeSplineStroke함수를 이용해 Stroke를 만들어 S에 저장하고 stroke\_cnt 를 증가시켜 Stroke의 총 개수를 카운트하였다.



Stroke 배열 S를 완성한 뒤에는 Stroke배열을 무작위로 순회하기 위해 무작위로 정렬된 인덱스들의 집합 order을 만들었다. Order의 크기는 그리드의 개수가 되도록 동적할당 하였다. 그 후 반복문을 통해 stroke\_cnt보다 작은 수들을 무작위로 order에 추가하였으며, 만약 중복되는 숫자가 추가될 경우 그 추가를 무시하고 다시 추가하는 방식으로 인덱스의 중복을 피하여 무작위로 정렬된 인덱스들의 배열을 만들 수 있었다.

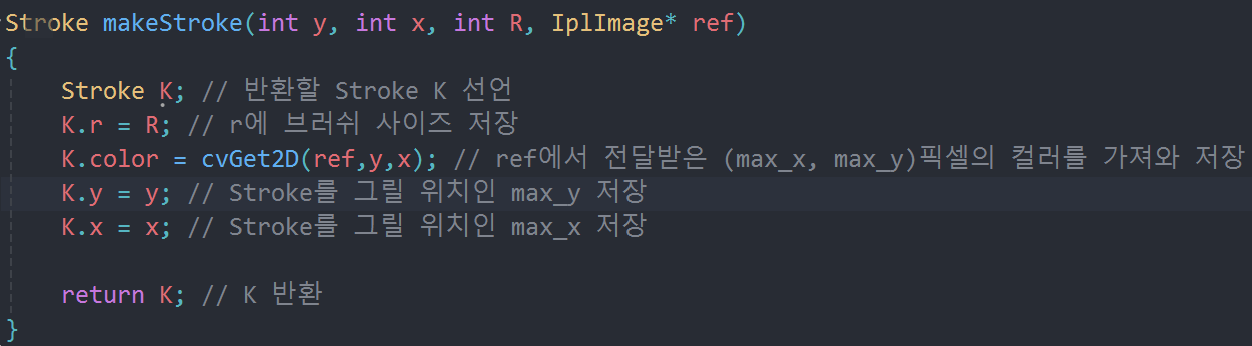


이후 MODE값에 따라 Stroke배열 S를 canvas에 적용하였다. MODE가 0일 경우 order에 저장된 무작위 순서에 따라, makeStroke으로 만들어진 원형 Stroke들을 cvCircle 함수를 이용해 꽉찬 원으로 canvas위에 그리게 하였다. MODE가 1일 경우 역시 무작위 순서에 따라 그렸으며 makeSplineStroke로 만들어진 Spline Stroke를 각 Stroke의 길이가 최소 길이보다 클 경우, point에 저장된 점들을 따라 그려지는 직선을 canvas 위에 그리게 하였다.



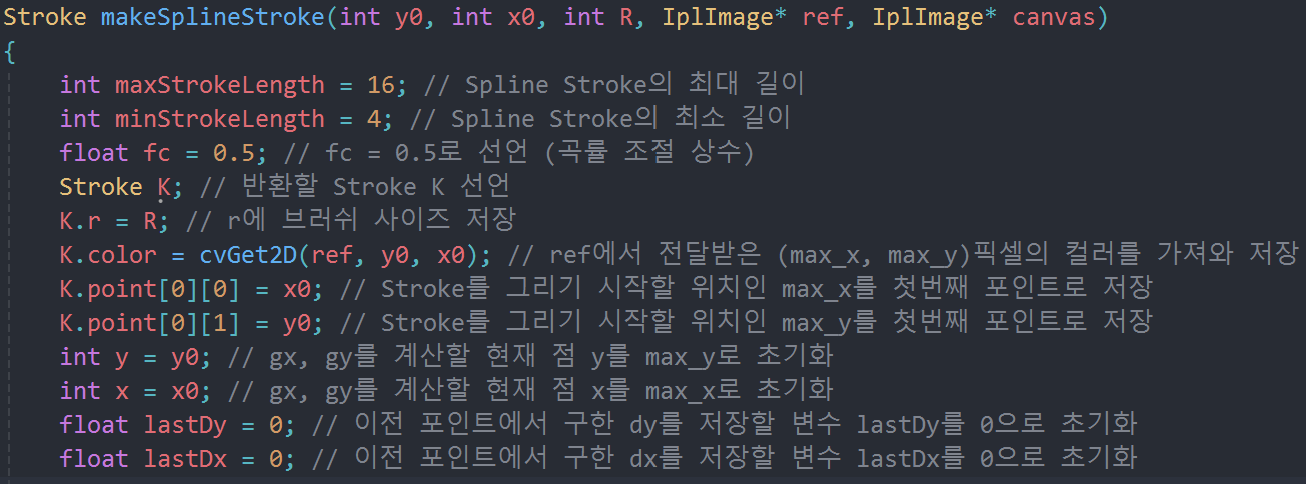
마지막으로 동적할당 되었던 배열 D, S, order의 할당을 해제하고 cvShowImage를 통해 각 레이어가 칠해질 때 마다 canvas를 출력하게 하였으며 cvWaitKey(1000)을 이용해 레이어가 칠해지는 반복문이 1초 간격을 두고 실행되도록 하였다.

IV. makeStroke함수 정의

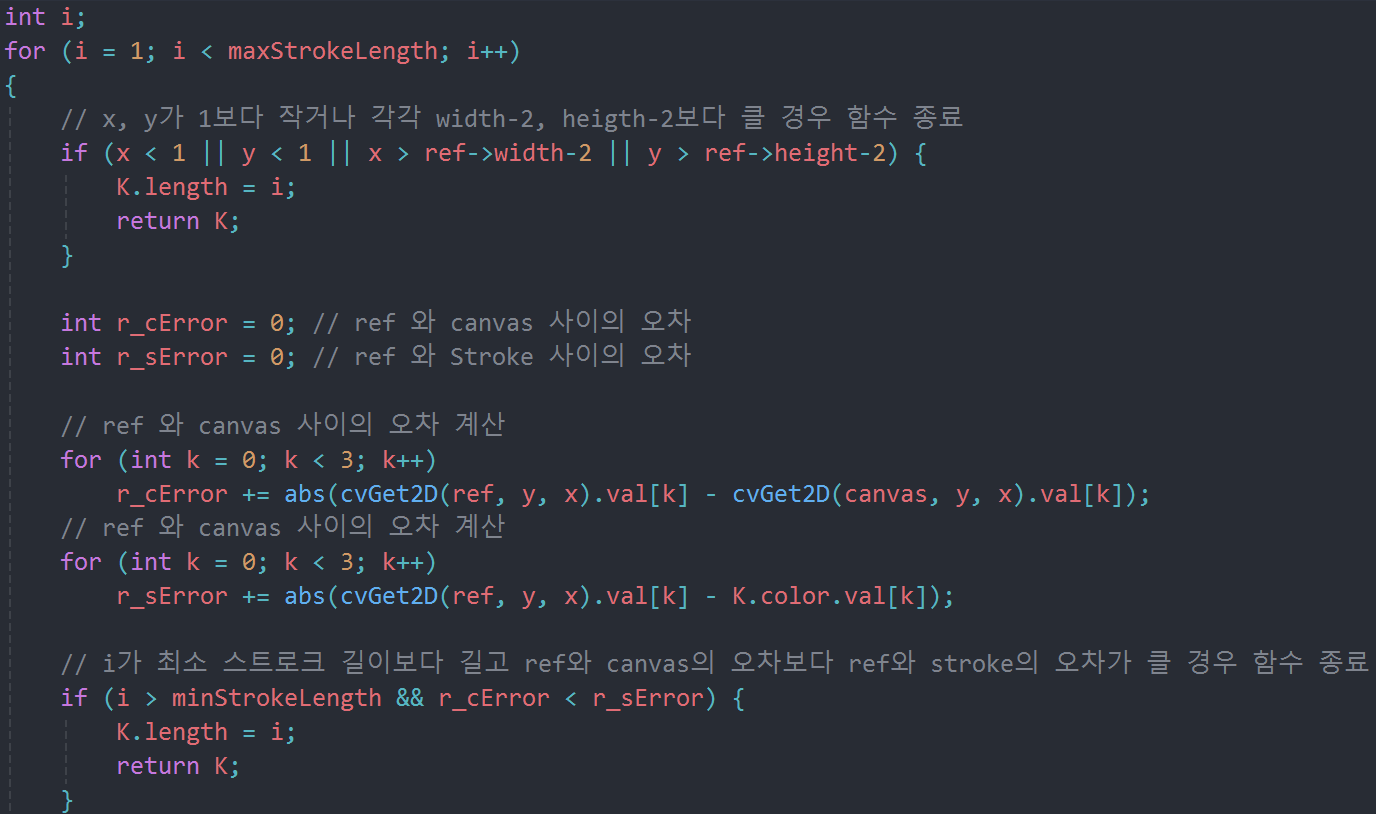


원형의 Stroke를 생성하는 makeStroke 함수는 위와 같다. 먼저 새로운 Stroke K를 선언하고 paintLayer에서 구한 그리드 내에서 오차가 가장 큰 픽셀의 x좌표와 y좌표를 Stroke의 위치 K.x, K.y로 저장하고 Brush Size는 K.r에 저장, ref이미지에서 K.x, K.y 에 위치한 픽셀의 색상을 불러와 K.color에 저장하였다. 그리고 K를 반환하여 반환된 Stroke 값을 배열에 추가할 수 있도록 하였다.

V. makeSplineStroke 함수 정의



makeStroke함수를 이용해 원형 Stroke를 이용해 그림을 그리는데 성공한 뒤에는 makeSplineStroke함수를 정의하였다. makeSplineStroke 함수는makeStroke가 받는 매개변수들을 받고 이에 더해 canvas이미지도 매개변수로 받는다. 함수의 첫 부분에서는 SplineStroke를 만들기 위한 변수들을 선언한다. Stroke 길이의 최댓값과 최솟값, Stroke의 곡률을 조정할 상수 fc를 선언하고, 새 Stroke 구조체 변수 K를 선언하여 Brush Size와 color를 저장한 뒤에 Spline을 이루는 점들의 배열인 K.point 배열의 첫번째 인덱스에 x0, y0을 저장한다. 그리고 gx, gy를 계산할 현재 위치의 좌표를 x0, y0으로 초기화하고 직전의 dx와 dy값을 저장할 변수 lastDx와 lastDy변수를 선언하였다.

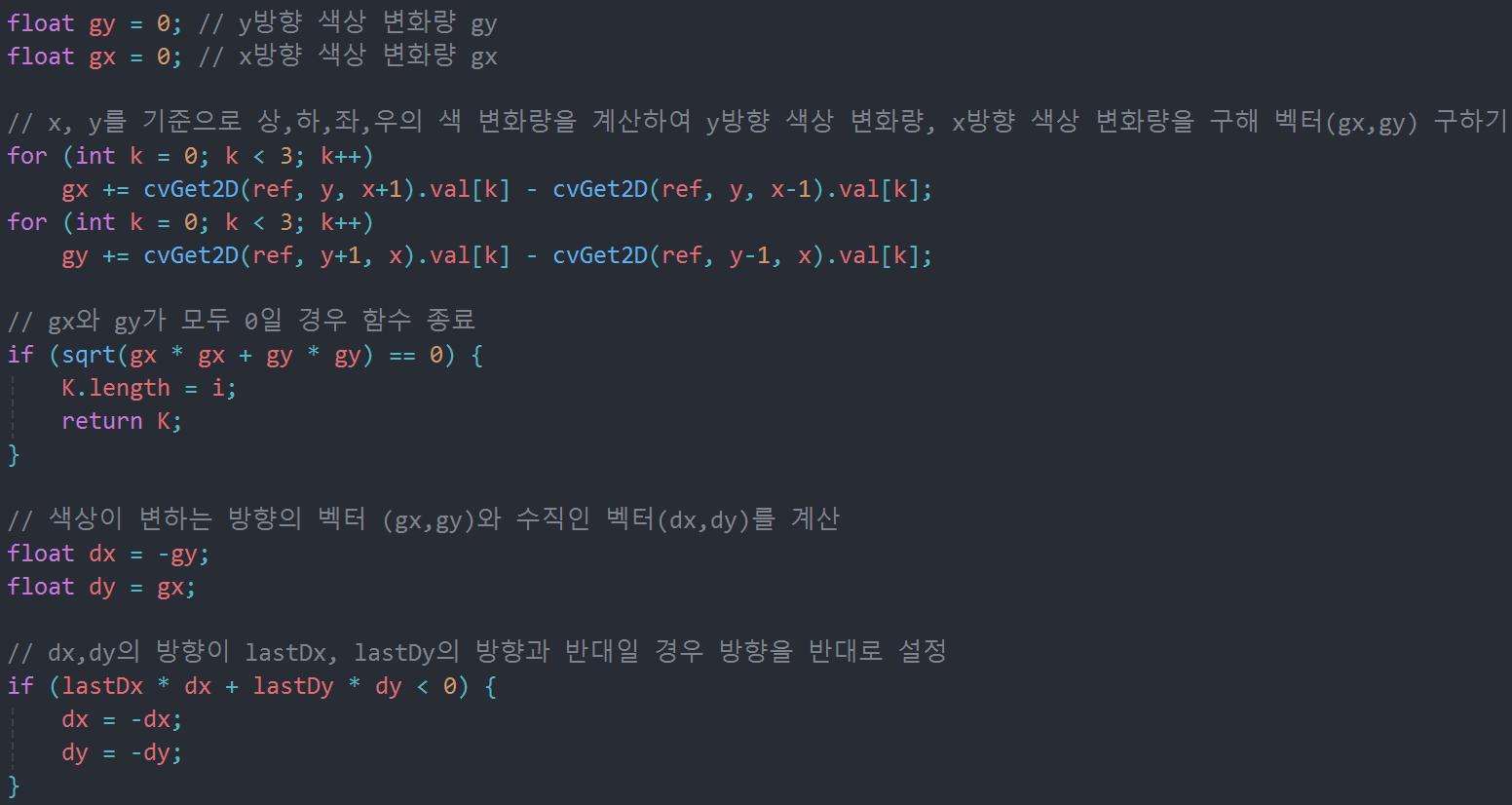


변수 선언을 마친 뒤에는 반복문을 통해 spline을 이루는 점들을 배열에 추가해가며

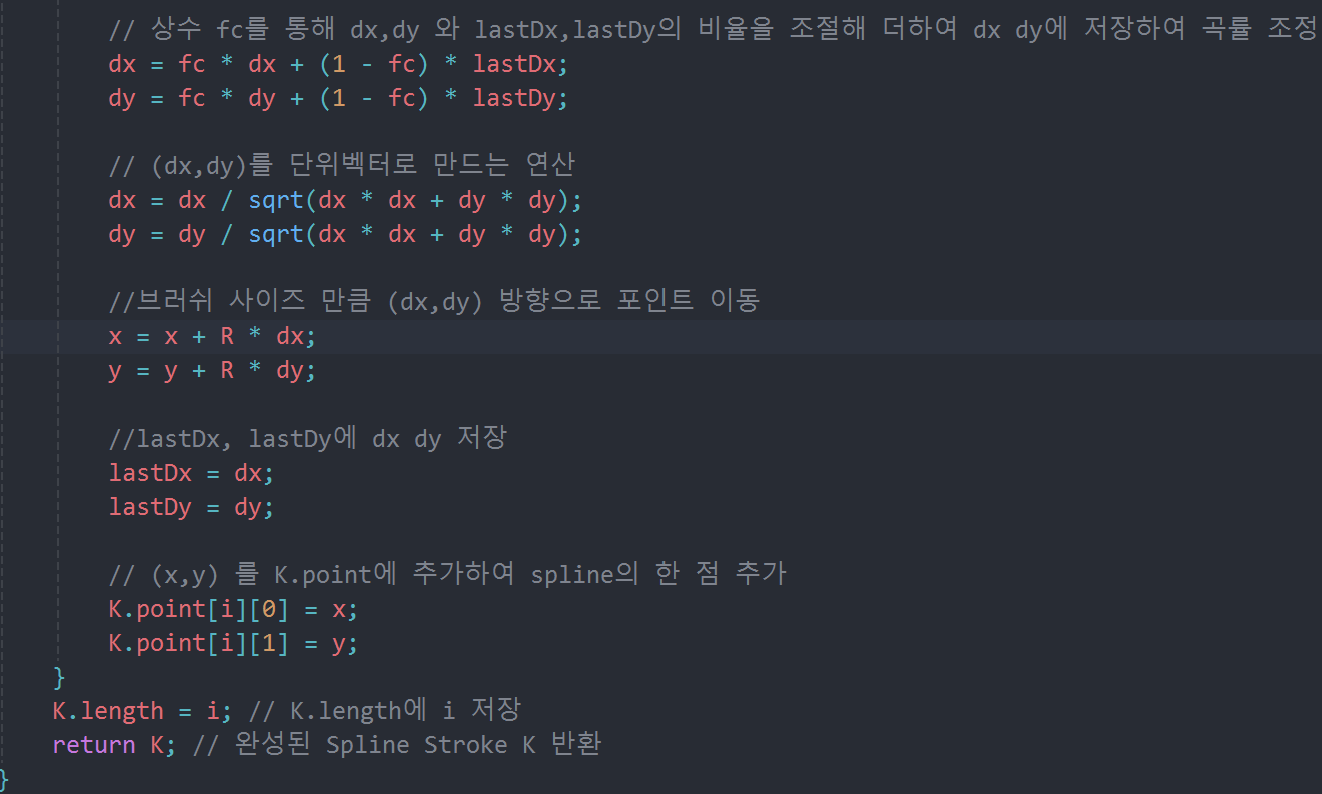
Spline Stroke를 만들어가야 했다. 우선, (x,y)에서 상하 좌우의 색상 변화량 (gx,gy)를

계산할 때 변수가 이미지 범위를 벗어나거나 Spline Stroke의 point가 이미지 밖에

위치하는 경우를 막기 위해 특정 범위가 넘어가는 경우 바로 함수를 종료하게 하였다 그 후 (x,y)에서 ref와 canvas, ref와 Stroke 색상 사이의 오차를 비교하여 Stroke의 길이가 최솟값보다 길고 ref의 색상이 Stroke보다 canvas에 가까울 경우, 함수를 종료하고 K를 반환하도록 하였다.



이후 (x,y)를 기준으로 y방향 색상 변화량을 gy, x방향 색상 변화량을 gx로 선언하였다. 그리고 계산을 통해 (gx,gy)의 값을 얻었다. 그 후 만약 gx와 gy가 둘 다 0, 즉 상하좌우 픽셀의 색상에 변화가 없을 경우에도 함수를 종료시키고 K를 반환하였다. 이후 벡터 (gx,gy)와 뱡향이 수직인 벡터를 저장하기 위해 (dx,dy)를 선언하고 dx = -gy, dy = gx 로 초기화 하여 색상이 변하지 않는 방향을 가리키는 벡터인 (dx,dy)를 얻어내었다. 이때 (dx,dy)의 방향이 Stroke의 진행방향과 정 반대일 경우 dx = -dx, dy = -dy로 (dx,dy)의 방향을 반대로 바꿔주었다.



그 뒤에는 상수 fc를 이용해 (dx,dy)와 (lastDx,lastDy)를 특정 비율로 더하여 SplineStroke의 곡률을 정하도록 하였다. 그 뒤 (dx,dy)를 벡터의 크기만큼 나누어 단위벡터로 만들었다. 이후 x와 y에 각각 R\*dx, R\*dy를 더하여 (x,y)에서 (dx,dy)방향으로 Brush Size만큼 이동한 곳을 새로운 (x,y)로 지정하고 이 좌표를 point배열에 저장하여 Spline을 이루는 점을 추가하였다. 이후 i를 K.length에 저장하여 Stroke의 길이를 저장하고 완성된 Stroke K를 반환한다

1. **결과**

논문에서 주어진 상수들을 조절하여 최대한 주어진 예시 코드와 유사하게 동작하도록 프로그램을 구성하도록 하였고 그 결과 최종적으로 결정된 상수는 위의 소스코드와 같이 fg = 1, fs = 1, fc = 0.5, maxStrokeLength = 16, minStrokeLength = 4, T = 25였다.

다음은 프로그램을 통해 변형한 이미지들의 예시이다, 모든 예시는 SplineStroke를 이용해 그린 이미지들이다.



Source 1 Result 1



Source 2 Result 2



Source 3 Result 3