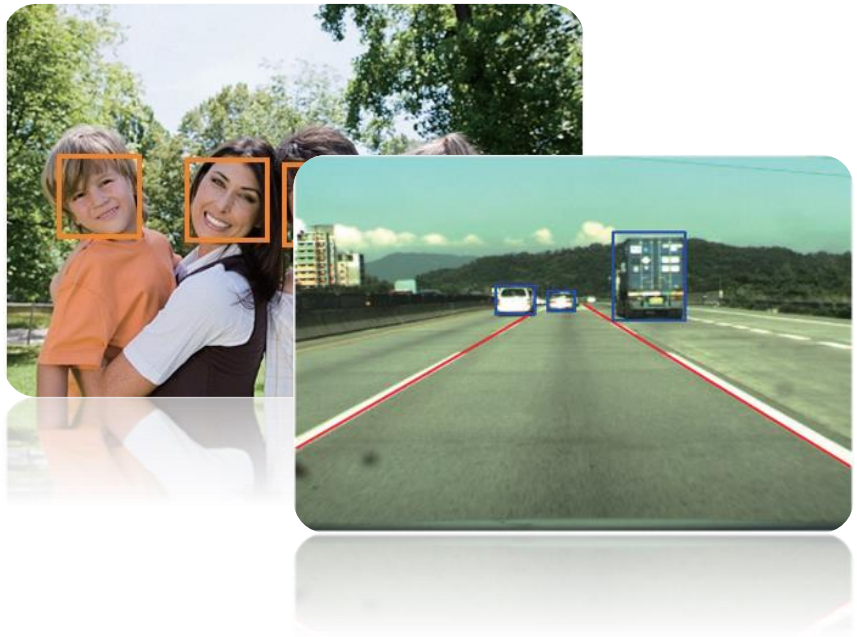
# 그리기 함수



# 직선 및 사각형 그리기

- cv::line()과 cv::rectangle()

**함수명과 반환형 및 인수 구조**

```
void line(Mat& img, Point pt1, Point pt2, const Scalar& color, int thickness = 1, int
        lineType = 8, int shift = 0)
void rectangle(Mat& img, Point pt1, Point pt2, const Scalar& color, int thickness = 1, int
        lineType = 8, int shift = 0)
void rectangle(Mat& img, Rect rec, const Scalar& color, int thickness = 1, int lineType = 8,
        int shift = 0)
```

| 인수 | 설명 |
|---|---|
| • Mat& img | 그릴 대상 행렬(영상) |
| • Point pt1, pt2 | 시작 좌표와 종료 좌표 |
| • Rect rect | 그릴 영역을 나타내는 사각형 자료 형 |
| • Scalar color | 선의 색상 |
| • int thickness | 선의 두께 , FILLED(-1)일 경우 지정된 색으로 내부를 채움 |
| • int lineType | 선의 형태 |

| 옵션 | 값 | 설명 |
|---|---|---|
| LINE_4 | 4 | 4-방향 연결선(4-connected line), |
| LINE_8 | 8 | 8-방향 연결선(8-connected line) |
| LINE_AA | 16 | 계단현상 감소(안티에일리싱) 선 |

| | |
|---|---|
| • int shift | 입력 좌표(pt1, pt2)에 대해서 오른쪽으로 비트시프트(>>) 연산한 결과를 좌표로 지정해서 직선을 그림 |

# 직선 및 사각형 그리기

★실습

예제 4.3.1 직선 & 사각형 그리기 – draw_line_rect.cpp

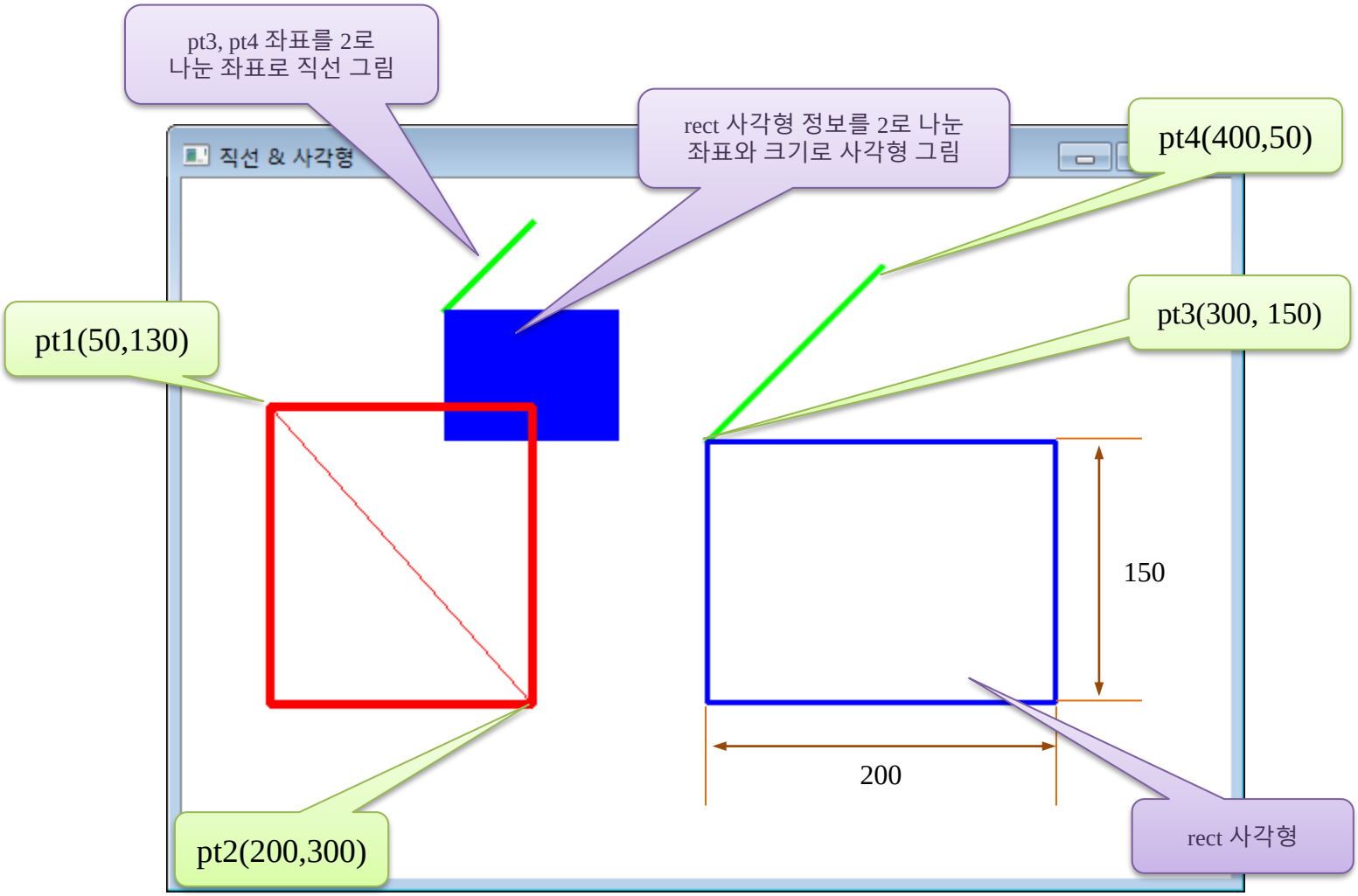
```
01  #include <opencv2/opencv.hpp>
02  using namespace cv;
03  using namespace std;
04  int main()
05  {
06      Scalar blue(255, 0, 0), red(0, 0, 255), green(0, 255, 0);        // 색상 선언
07      Scalar white = Scalar(255, 255, 255);                            // 흰색 색상
08      Scalar yellow(0, 255, 255);
09
10      Mat image(400, 600, CV_8UC3, white);
11      Point pt1(50, 130), pt2(200, 300), pt3(300, 150), pt4(400, 50); // 좌표 선언
12      Rect  rect(pt3, Size(200, 150));

        line(image, pt1, pt2, red);                                      // 직선 그리기
        line(image, pt3, pt4, green, 2, LINE_AA);                        // 안티에일리싱 선
16      line(image, pt3, pt4, green, 3, LINE_8, 1);        // 8방향 연결선 , 1비트 시프트
17
18      rectangle(image, rect, blue, 2 );                                // 사각형 그리기
19      rectangle(image, rect, blue, FILLED, LINE_4, 1);   // 4방향 연결선, 1비트 시프트
20      rectangle(image, pt1, pt2, red, 3 );
21
22      imshow("직선 & 사각형", image);
23      waitKey(0);
24      return 0;
25  }
```

3채널 ucahr 행렬 선언 및 흰색 (255) 초기화

사각형 선언 - 300, 150 에서 200x150 크기

파란색, 채움 사각형

# 직선 및 사각형 그리기


pt3, pt4 좌표를 2로
나눈 좌표로 직선 그림
rect 사각형 정보를 2로 나눈
좌표와 크기로 사각형 그림
pt4(400,50)
직선 & 사각형
pt1(50,130)
pt3(300, 150)
150
200
pt2(200,300)
rect 사각형

# 글자 쓰기

표시 문자열

2줄 산세리프 폰트

확대 비율

```
putText(image, "DUPLEX", pt1, FONT_HERSHEY_DUPLEX, 2, Scalar(128, 128, 0));
putText(image, "TRIPLEX", pt2, FONT_HERSHEY_TRIPLEX, 3, Scalar(221, 160, 221));
```

색상

3줄 세리프 폰트

시작좌표(pt1)

시작좌표(pt2)

# 글자 쓰기

| 함수명과 반환형 및 인수 구조 |
|---|
| void putText(Mat& img, const string& text, Point org, int fontFace, double fontScale, Scalar color, int thickness = 1, int lineType = 8, bool bottomLeftOrigin = false) |

| 인수 | 설명 |
|---|---|
| Mat& img | 문자열을 작성할 대상 행렬(영상) |
| string& text | 작성할 문자열 |
| Point org | 문자열의 시작 좌표, 문자열에서 가장 왼쪽 하단을 의미 |
| int fontFace | 문자열에 대한 글꼴 |
| double fontScale | 글자 크기 확대 비율 |
| Scalar color | 글자의 색상 |
| int thichness | 글자의 굵기 |
| int lineTpye | 글자 선의 형태 |
| bool bottomLeftOrigin | 영상의 원점 좌 |

〈표 4.3.1〉 문자열의 폰트(fontFace)에 대한 옵션과 의미

| 옵션 | 값 | 설명 |
|---|---|---|
| FONT_HERSHEY_SIMPLEX | 0 | 중간 크기 산세리프 폰트 |
| FONT_HERSHEY_PLAIN | 1 | 작은 크기 산세리프 폰트 |
| FONT_HERSHEY_DUPLEX | 2 | 2줄 산세리프 폰트 |
| FONT_HERSHEY_COMPLEX | 3 | 중간 크기 세리프 폰트 |
| FONT_HERSHEY_TRIPLEX | 4 | 3줄 세리프 폰트 |
| FONT_HERSHEY_COMPLEX_SMALL | 5 | COMPLEX보다 작은 크기 |
| FONT_HERSHEY_SCRIPT_SIMPLEX | 6 | 필기체 스타일 폰트 |
| FONT_HERSHEY_SCRIPT_COMPLEX | 7 | 복잡한 필기체 스타일 |
| FONT_ITALIC | 16 | 이탤릭체를 위한 플래그 |

# 글자 쓰기

★실습

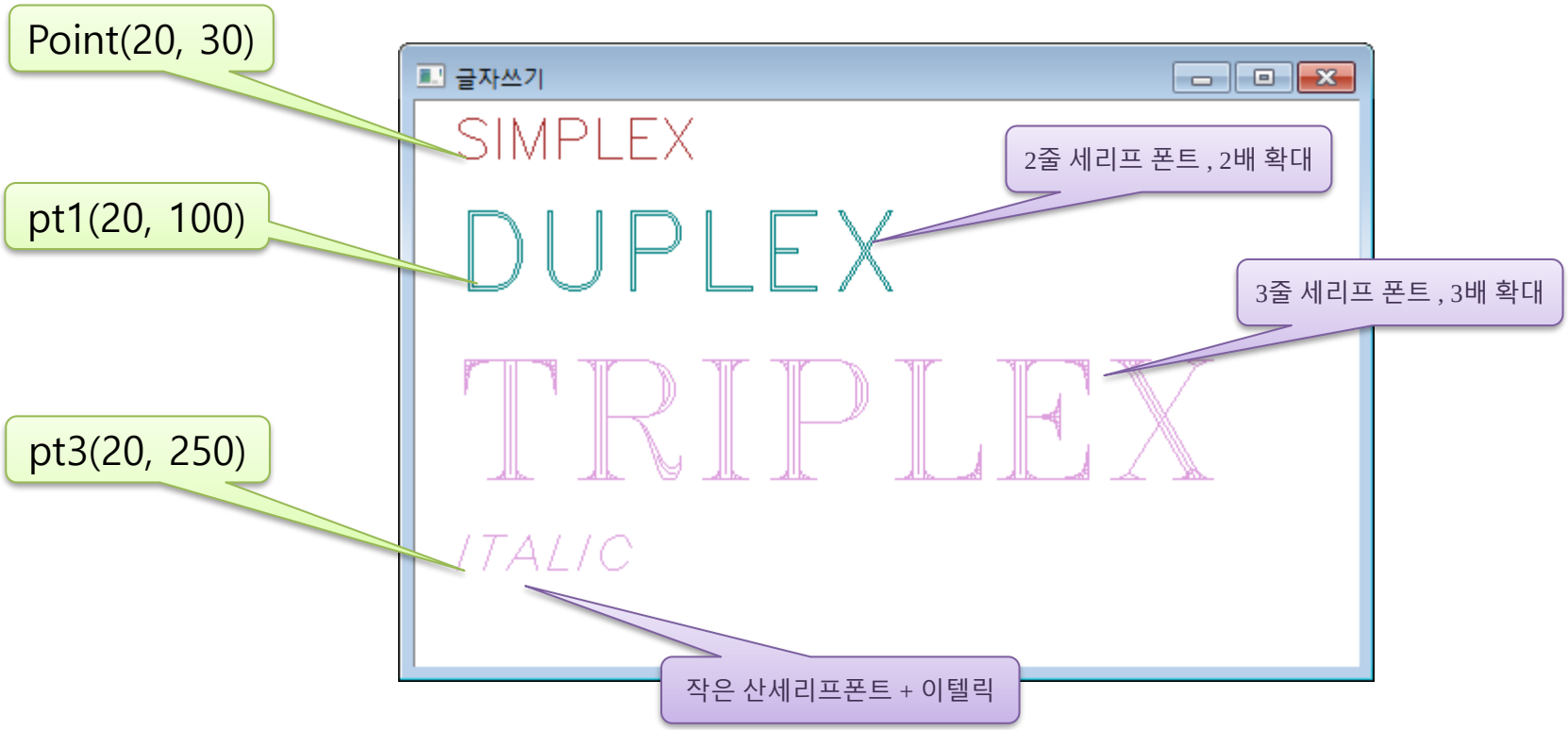
예제 4.3.2 글자 쓰기 – draw_line_rect.cpp

```
01  #include <opencv2/opencv.hpp>
02  using namespace cv;
03  using namespace std;
04  int main()
05  {
06      Scalar olive(128, 128, 0), violet(221, 160, 221), brown(42, 42, 165);
07      Point pt1(20, 100), pt2(20, 200), pt3(20, 250) ;            // 문자열 위치 좌표
08
09      Mat image(300, 500, CV_8UC3, Scalar(255, 255, 255));
10
11      putText(image, "SIMPLEX", Point(20, 30), FONT_HERSHEY_SIMPLEX, 1, brown);
12      putText(image, "DUPLEX", pt1, FONT_HERSHEY_DUPLEX, 2, olive);
13      putText(image, "TRIPLEX", pt2, FONT_HERSHEY_TRIPLEX, 3, violet);
14      putText(image, "ITALIC" , pt3, FONT_HERSHEY_PLAIN | FONT_ITALIC, 2, violet);
15
16      imshow("글자쓰기", image);
17      waitKey(0);
18      return 0;
19  }
```

작은 산세리프폰트 + 이텔릭

# 글자 쓰기

- 실행결과


Point(20, 30)
pt1(20, 100)
pt3(20, 250)
글자쓰기
SIMPLEX
DUPLEX
TRIPLEX
ITALIC
2줄 세리프 폰트 , 2배 확대
3줄 세리프 폰트 , 3배 확대
작은 산세리프폰트 + 이텔릭

# 원 그리기

| 함수명과 반환형 및 인수 구조 | |
|---|---|
| void circle(Mat& img, Point center, int radius, const Scalar& color, int thickness = 1, int lineType = 8, int shift = 0) | |
| **인수** | **설명** |
| • Mat& img | 원을 그릴 대상 행렬(영상) |
| • Point center | 원의 중심 좌표 |
| • int radius | 원의 반지름 |
| • Scalar& color | 선의 색상 |
| • int thickness | 선의 두께 |
| • int lineType | 선의 형태 , cv::line() 함수의 인자와 동일 |
| • int shift | 좌표에 대한 비트 시프트 연산 |

# 원 그리기

★실습

예제 4.3.3 원 그리기 – draw_circle.cpp

```
01  #include <opencv2/opencv.hpp>
02  using namespace cv;
03  using namespace std;
04  int main()
05  {
06      Scalar orange(0, 165, 255), blue(255, 0, 0), magenta(255, 0, 255);
07      Mat image(300, 500, CV_8UC3, Scalar(255, 255, 255));        // 흰색 컬러 영상
08
09      Point center = (Point)image.size() / 2;                     // 영상 중심좌표
10      Point pt1(70, 50), pt2(350, 220);
11
12      circle(image, center, 100, blue);                           // 원 그리기
13      circle(image,  pt1 ,  80, orange, 2);
14      circle(image,  pt2 ,  60, magenta, -1);                     // 원 내부 채움
15
16      int  font = FONT_HERSHEY_COMPLEX;
17      putText(image, "center_blue", center, font, 1.2, blue);
18      putText(image, "pt1_orange", pt1, font, 0.8, orange);
19      putText(image, "pt2_magenta", pt2 + Point(2, 2), font, 0.5, Scalar(0,0,0), 2);
20      putText(image, "pt2_magenta", pt2, font, 0.5, magenta, 1);
21
22      imshow("원그리기", image);
23      waitKey(0);
24      return 0;
25  }
```

원 중심 좌표

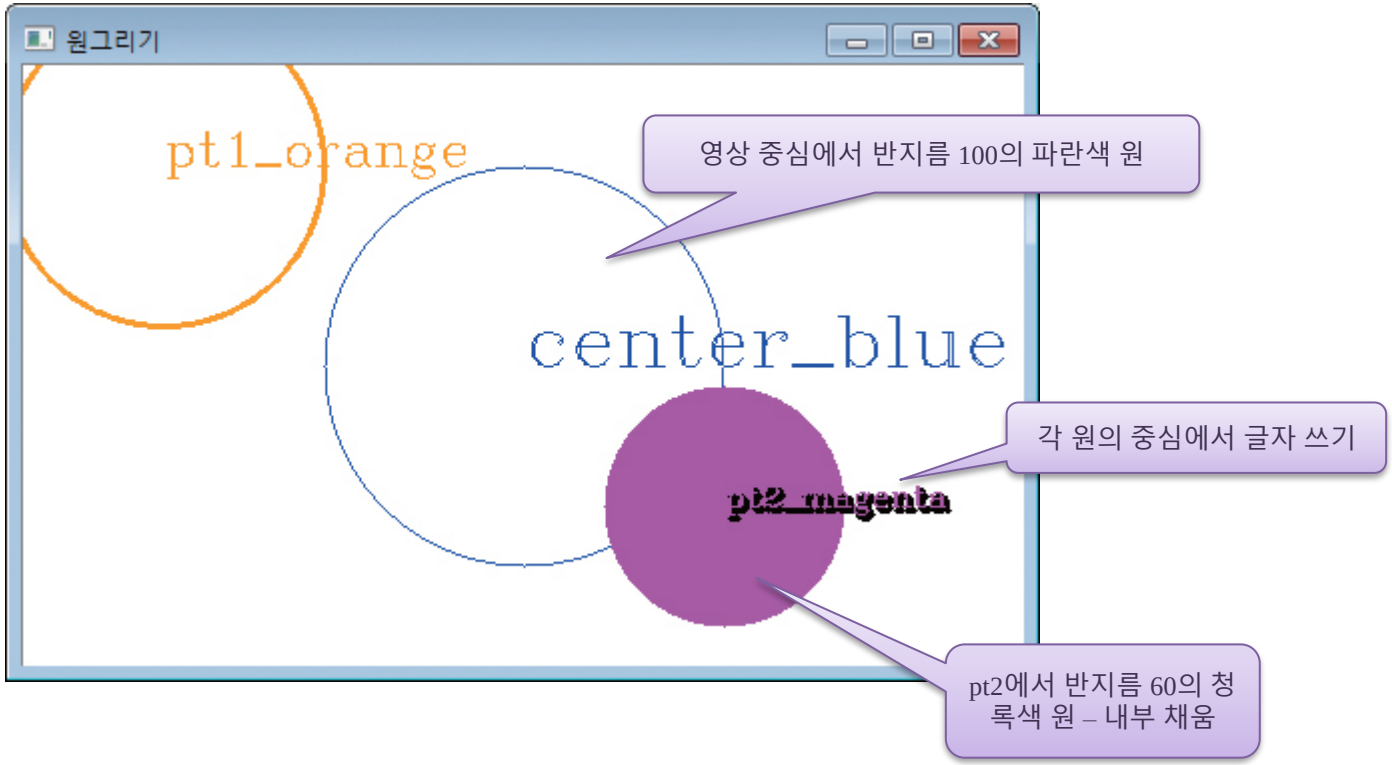
영상 중심에서 반지름 100의 파란색 원

pt2에서 반지름 60의 청록색 원 – 내부 채움

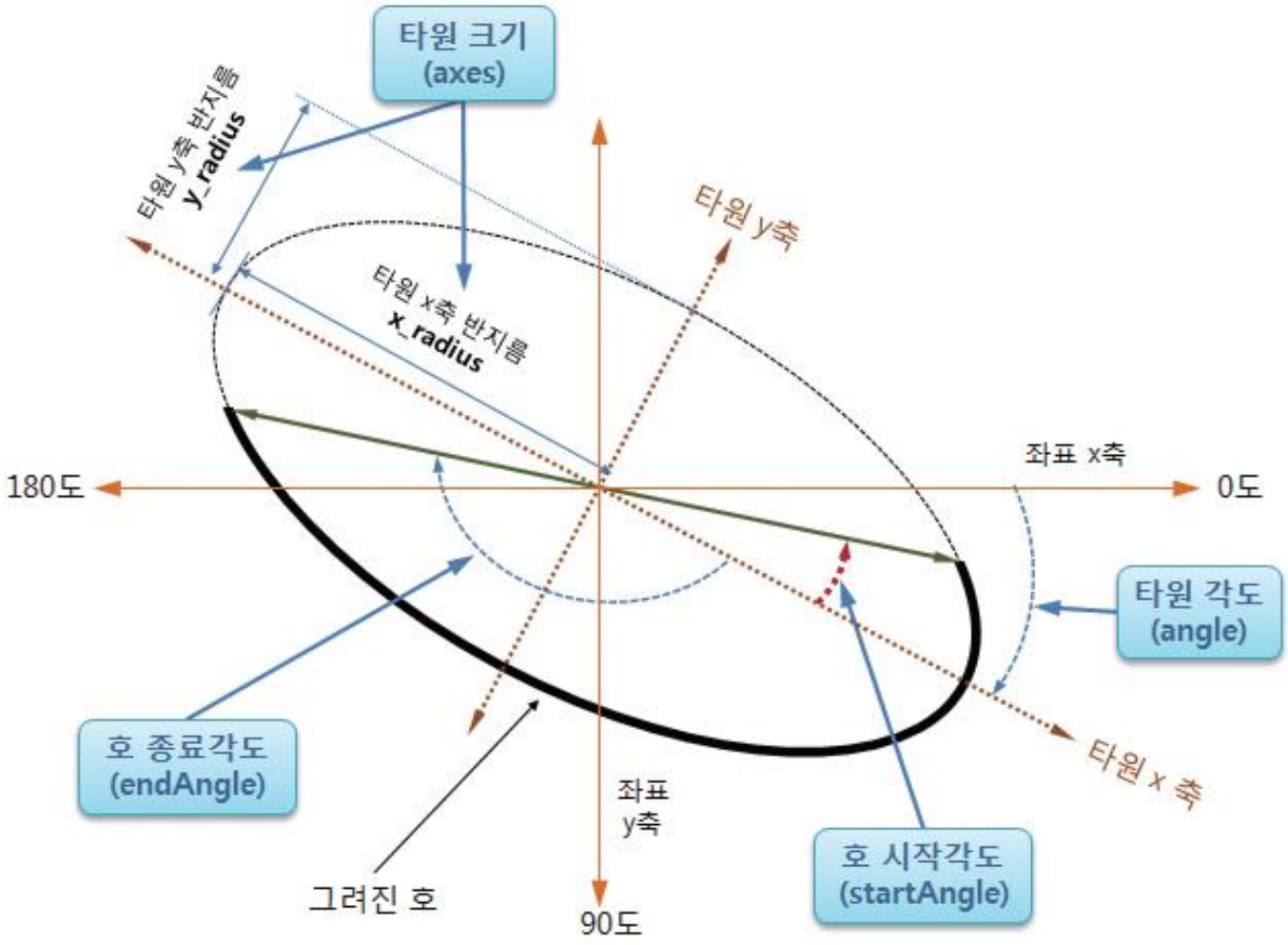
원의 중심에서 글자 쓰기

# 원 그리기

- 실행결과


원그리기
pt1_orange
center_blue
pt2_magenta
영상 중심에서 반지름 100의 파란색 원
각 원의 중심에서 글자 쓰기
pt2에서 반지름 60의 청록색 원 – 내부 채움

# 타원 그리기

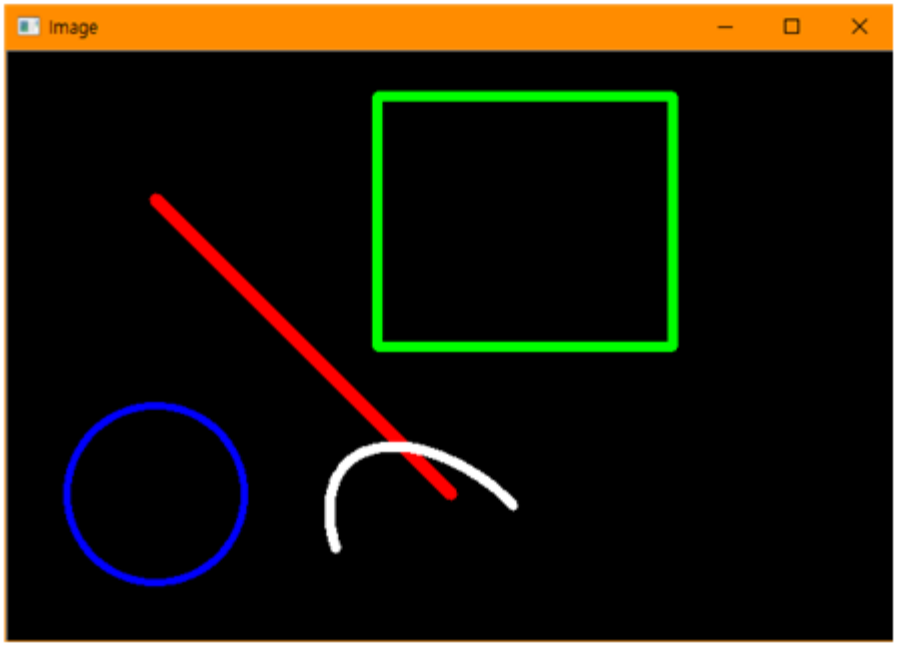

타원 크기
(axes)
타원 y축 반지름
y_radius
타원 y축
타원 x축 반지름
x_radius
좌표 x축
180도
0도
타원 각도
(angle)
호 종료각도
(endAngle)
타원 x 축
좌표
y축
그려진 호
호 시작각도
(startAngle)
90도

# 타원 그리기

- 타원 그리기 함수

| 함수명과 반환형 및 인수 구조 | |
|---|---|
| void ellipse(Mat& img, Point center, Size axes, double angle, double startAngle, double endAngle, const Scalar& color, int thickness = 1, int lineType = 8, int shift = 0)<br>void ellipse(Mat& img, const RotatedRect& box, const Scalar& color, int thickness = 1, int lineType = 8) | |
| 인수 | 설명 |
| • Mat& img | 그릴 대상 행렬(영상) |
| • Point center | 원의 중심 좌표 |
| • Size axes | 타원의 크기(x축 반지름, y축 반지름) |
| • double angle | 타원의 각도(3시 방향이 0도, 시계방향 회전) |
| • double startAngle | 호의 시작 각도 |
| • double endAngle | 호의 종료 각도 |
| • Scalar& color | 선의 색상 |
| • RotatedRect& box | 회전사각형(중심점, 회전 각도, 크기) 객체로 타원 그림 |

# OpenCV를 이용하여 도형 그리기



# 소스 분석

★실습

```
#include <opencv2/core.hpp>
#include <opencv2/imgproc.hpp>
#include <opencv2/highgui.hpp>
using namespace cv;
using namespace std;

int main()
{
	// 검정색으로 초기화된 600×400 크기의 영상 생성
	Mat image = Mat(400, 600, CV_8UC3, Scalar(0, 0, 0));

	line(image, Point(100, 100), Point(300, 300), Scalar(0, 0, 255), 7);
	rectangle(image, Point(250, 30), Point(450, 200), Scalar(0, 255, 0), 5);
	circle(image, Point(100, 300), 60, Scalar(255, 0, 0), 3);
	ellipse(image, Point(300, 350), Point(100, 60), 45, 130, 270,
		Scalar(255, 255, 255), 5);

	imshow("Image", image);
	waitKey(0);
	return(0);
}
```

# HW

1. 800 x 600 크기의 윈도우를 만들고, (100,100) 위치에 반지름이 50 인 파란색 원을 그리시오

2. 가로 600, 세로 400 크기 영상에 200x200 크기의 태극 문양을 그리시오 (힌트: 타원그리기 함수 사용)