Recognize triangle and circle

1. Outline of program

- Requirement :

Implement a program which distinguish whether the Shape in the picture is circle or triangle.

Except input and output functions in OpenCV, Do not use the any pre-implemented function in libraries.

Sample Images are provided and project result will be evaluated with several different images.

Each of images contain one shape. (Circle or Triangle)

2. Design of program

designed this program as below.

- Assume the input image is circle.
- We can get pixels which are highest, lowest, farthest to the right and left points.
- After that, we can get a central point and a radius by using those pixels.
- Since we supposed that the image is circle, some points should exist upwards, downwards, right and left as much as radius from the central point.
- If there are all points, we conclude that it is circle image.
- Because of noises, we assume the error range (-10~10).

3. The spending time

took about 2 days.

4. Source code explanation

- There are two source codes(Points.cpp, main.cpp) and one header file (Header.h).
- Points class stores high, low, left, right, central points and length of radius.

  This class also have two methods of finding points and determining circle.

- Header.h

```
        #include <iostream>
#include <opencv2\highgui\highgui.hpp>
```

```
using namespace std;

#ifndef __HEADER_H__
#define __HEADER_H__

class Points {
private:
	CvPoint high, low, left, right, central;
	int radius = 0;
	int cnt = 0;

public:
	Points() {};
	void findPoints(IplImage* image);
	bool isCircle(IplImage* image);

};

	#endif
```

- Points.cpp

```
	#include "Header.h"

void Points::findPoints(IplImage* image) {
	high = cvPoint(image->height, image->width);
	low = cvPoint(0, 0);
	left = cvPoint(image->height, image->width);
	right = cvPoint(0, 0);
	central = cvPoint(0, 0);

	// find high, low, left, and right point
	for (int y = 0; y < image->height; y++) {
		for (int x = 0; x < image->width; x++) {
			if (cvGet2D(image, y, x).val[0] == 0 && cvGet2D(image, y, x).val[1]
== 0 && cvGet2D(image, y, x).val[2] == 0) {
				if (y < high.y) {
					high.x = x;
					high.y = y;
				}
				if (y > low.y) {
					low.x = x;
					low.y = y;
				}
				if (x < left.x) {
					left.x = x;
					left.y = y;
				}
				if (x > right.x) {
					right.x = x;
					right.y = y;
				}
			}
		}
	}
```

```
        central.x = (left.x + right.x) / 2;
        central.y = (high.y + low.y) / 2;

        radius = (right.x - left.x) / 2;


}
// Determine the image is circle
bool Points::isCircle(IplImage* image) {
        for (int i = -10; i < 10; i++) {
                if (cvGet2D(image, central.y + radius + i, central.x).val[0] == 0) {
                        cnt++;
                        break;
                }
        }
        for (int i = -10; i < 10; i++) {
                if (cvGet2D(image, central.y, central.x - radius + i).val[0] == 0) {
                        cnt++;
                        break;
                }
        }
        for (int i = -10; i < 10; i++) {
                if (cvGet2D(image, central.y - radius + i, central.x).val[0] == 0) {
                        cnt++;
                        break;
                }
        }
        for (int i = -10; i < 10; i++) {
                if (cvGet2D(image, central.y, central.x + radius + i).val[0] == 0) {
                        cnt++;
                        break;
                }
        }
        if (cnt == 4)
                return true;
        else
                return false;

}
```

- main.cpp

  load four images named a, b, c, d, and print if images are circle or triangle.

```
#include "Header.h"

int main() {

        // Load four images named a, b, c, d
        IplImage* arr_image[4];
        for (int i = 0; i < 4; i++) {
                switch (i) {
                case 0:
                        arr_image[i] = cvLoadImage("C:₩₩a.jpg", CV_LOAD_IMAGE_COLOR);
                        if (arr_image[i] == NULL)
                                return -1;
                        break;
```

```cpp
		case 1:
			arr_image[i] = cvLoadImage("C:₩₩b.jpg", CV_LOAD_IMAGE_COLOR);
			if (arr_image[i] == NULL)
				return -1;
			break;
		case 2:
			arr_image[i] = cvLoadImage("C:₩₩c.jpg", CV_LOAD_IMAGE_COLOR);
			if (arr_image[i] == NULL)
				return -1;
			break;
		default:
			arr_image[i] = cvLoadImage("C:₩₩d.jpg", CV_LOAD_IMAGE_COLOR);
			if (arr_image[i] == NULL)
				return -1;
		}
	}

	cout << "Result : " << endl;

	// First, make Points objects
	// And find high, low, right, left points
	// Finally, If the image is circle, print "Circle!!" or not print "Triangle"
	Points imgA = Points();
	imgA.findPoints(arr_image[0]);
	if (imgA.isCircle(arr_image[0])) {
		cout <<"a : Circle!!" << endl;
	}
	else {
		cout<< "a : Triangle!!" << endl;
	}

	Points imgB = Points();
	imgB.findPoints(arr_image[1]);
	if (imgB.isCircle(arr_image[1])) {
		cout << "b : Circle!!" << endl;
	}
	else {
		cout << "b : Triangle!!" << endl;
	}


	Points imgC = Points();
	imgC.findPoints(arr_image[2]);
	if (imgC.isCircle(arr_image[2])) {
		cout << "c : Circle!!" << endl;
	}
	else {
		cout << "c : Triangle!!" << endl;
	}


	Points imgD = Points();
	imgD.findPoints(arr_image[3]);
	if (imgD.isCircle(arr_image[3])) {
		cout << "d : Circle!!" << endl;
```

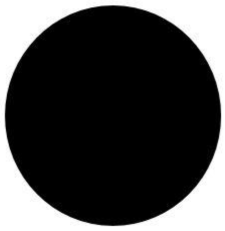
```
}
else {
        cout << "d : Triangle!!" << endl;
}
-   }
```

5. Result

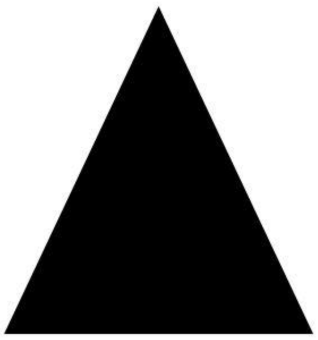
- we tested four images. Two are sample images provided, and the others are images made by us.

- images

-> a

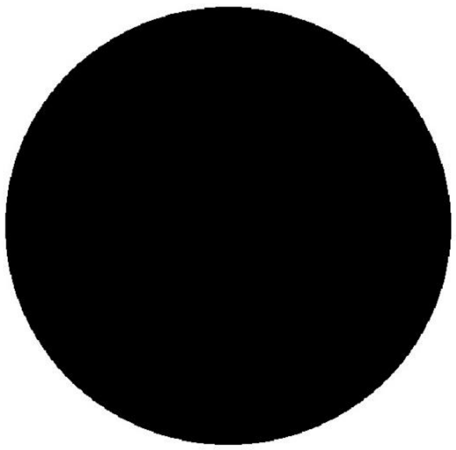


-> b

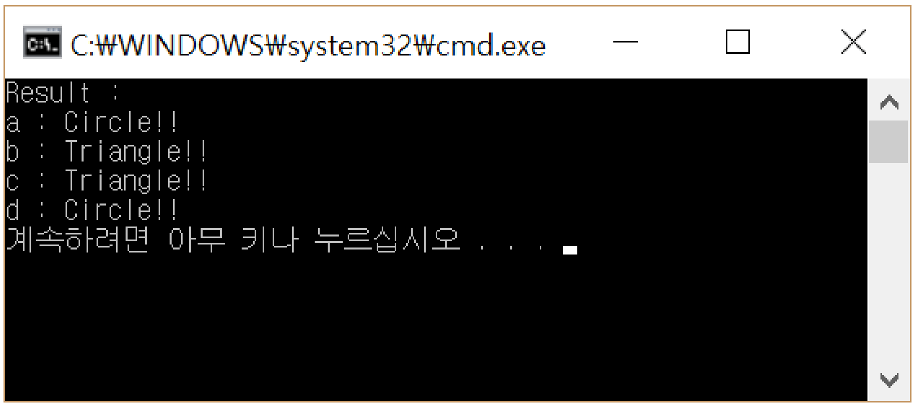


-> c

-> d



-> result



```
Result :
a : Circle!!
b : Triangle!!
c : Triangle!!
d : Circle!!
계속하려면 아무 키나 누르십시오 . . .
```