Shapes

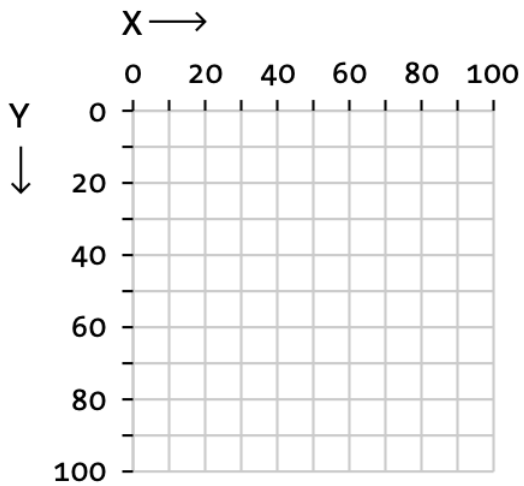
`size(width, height);`

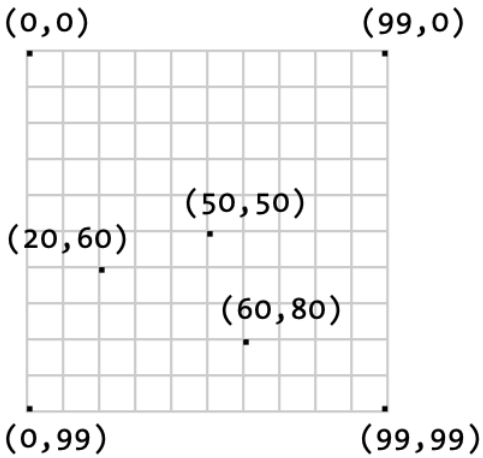
```
size(800, 400);
```

After you specify the size of your canvas, you will have access to the global variables `width` and `height`;

```
println("The size of my canvas is "
    + width + " by "
    + height + " pixels");
```

Coordinates



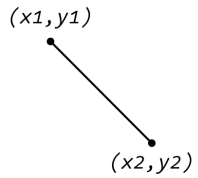


Basic shapes

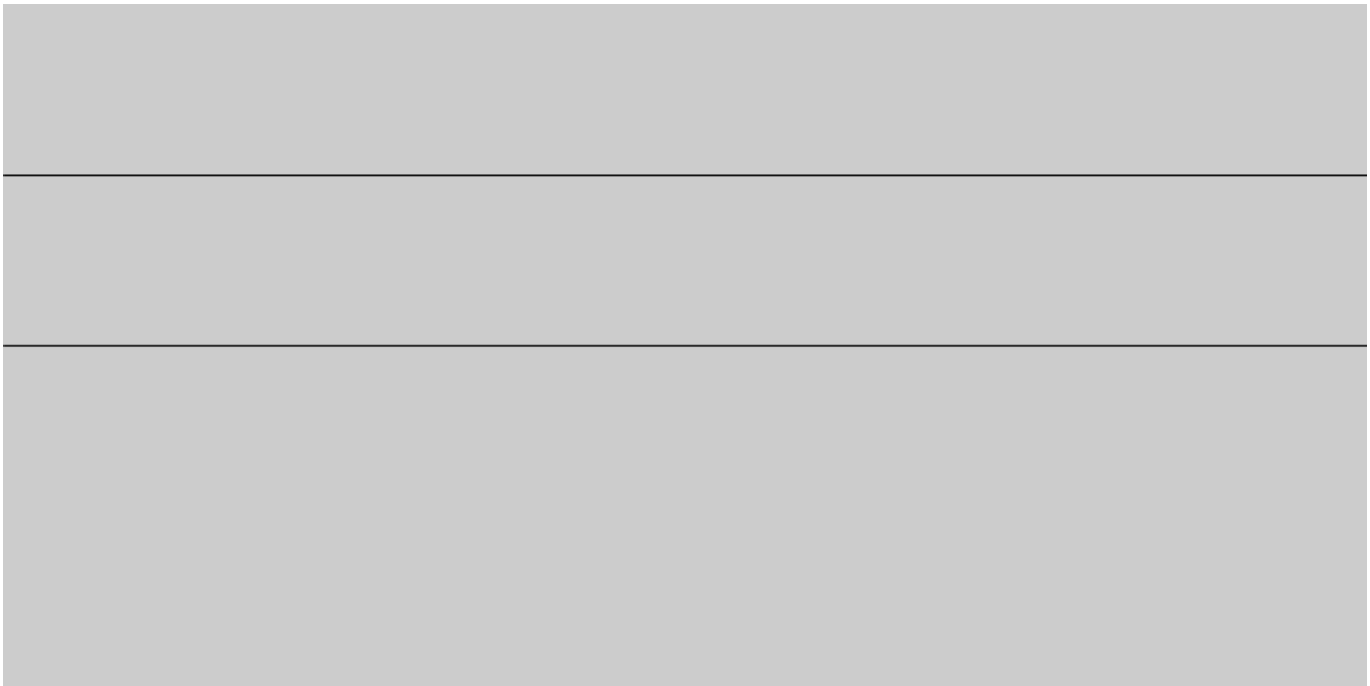
`point(x, y);`

```
//A point drawn in the middle of the screen
point(width/2, height/2);
```

`line(x1, y1, x2, y2);`

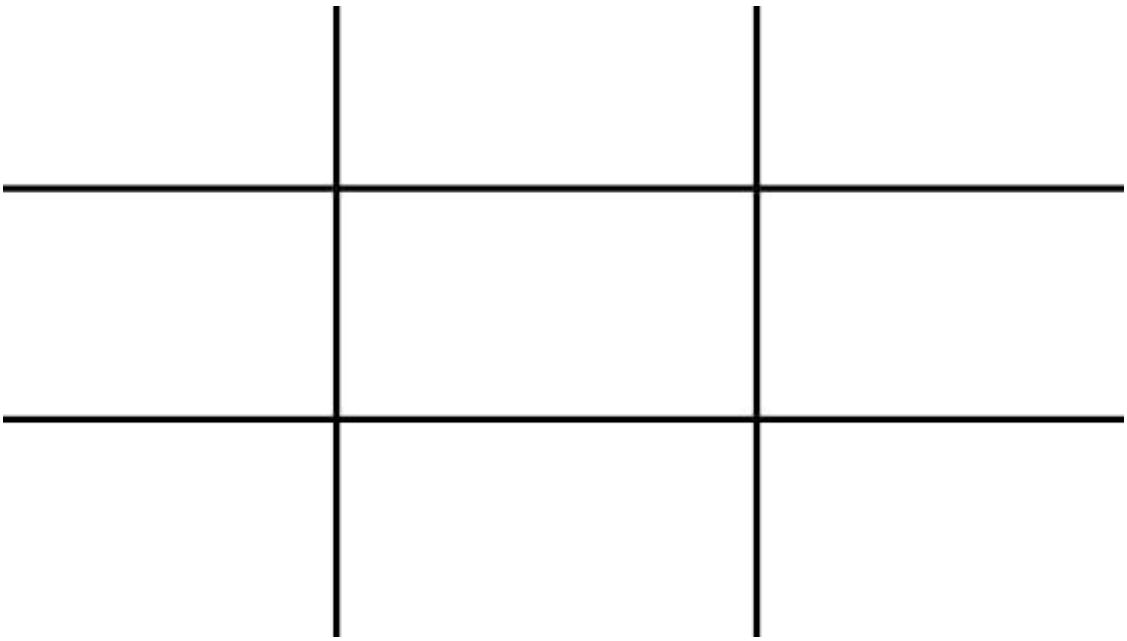



```
//Two horizontal parallel lines
line(0, 100, width, 100);
line(0, 200, width, 200);
```

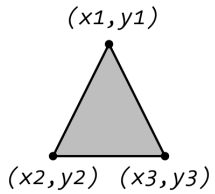


Exercise

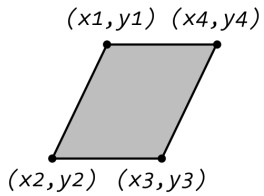
Implement the two-thirds rule in photography for your canvas.



```
triangle(x1, y1, x2, y2, x3, y4);
```

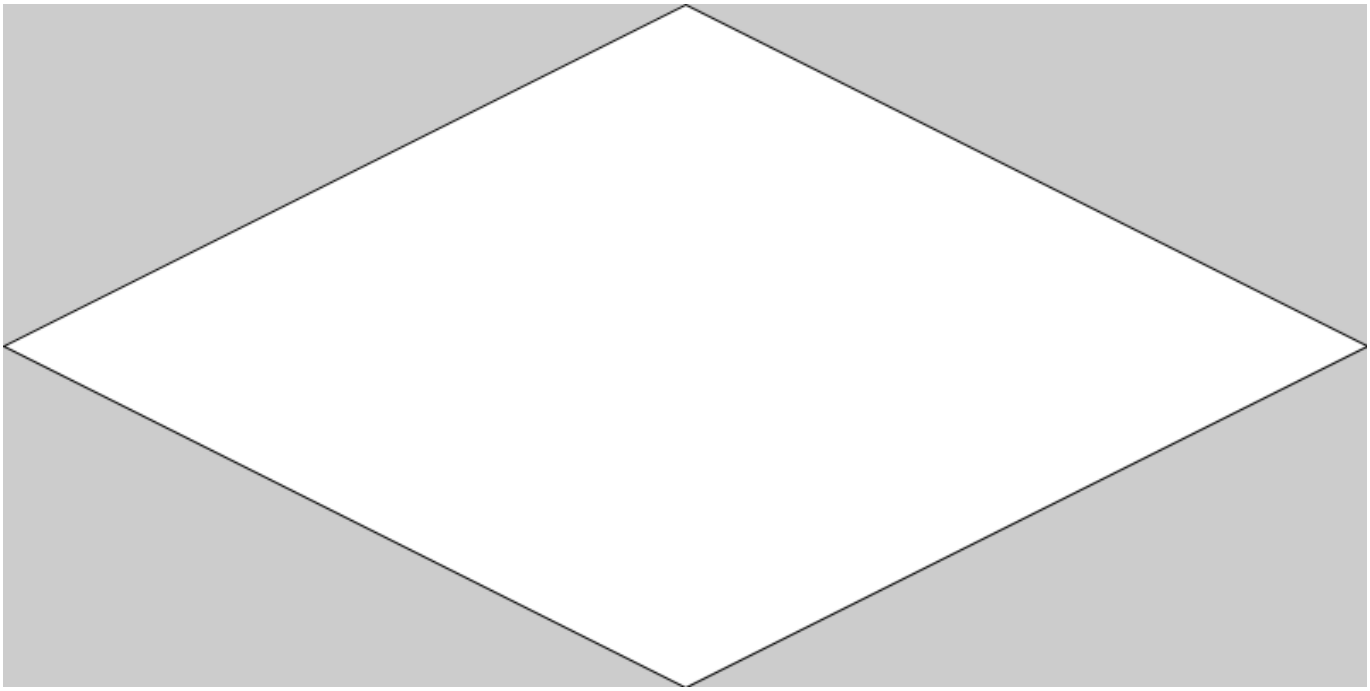



```
quad(x1, y1, x2, y2, x3, y3, x4, y4);
```

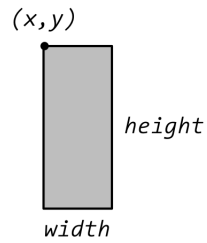


Exercise

Draw a diamond shape like you see in the picture below by using only the variables `width` and `height`.

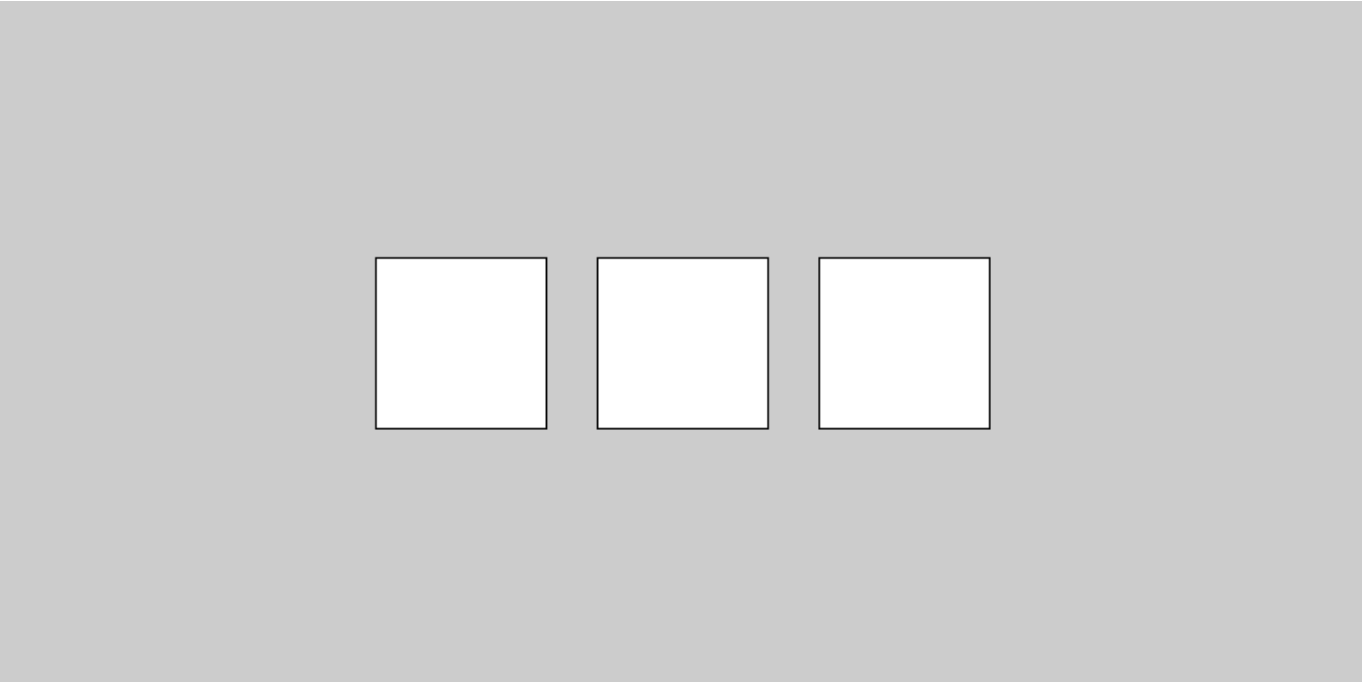


`rect(x, y, width, height)`

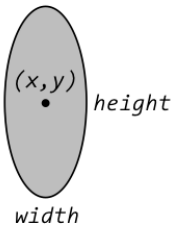


Exercise

Draw 3 squares measuring 100 pixels on the side, spaced by 30 pixels like in the picture below.

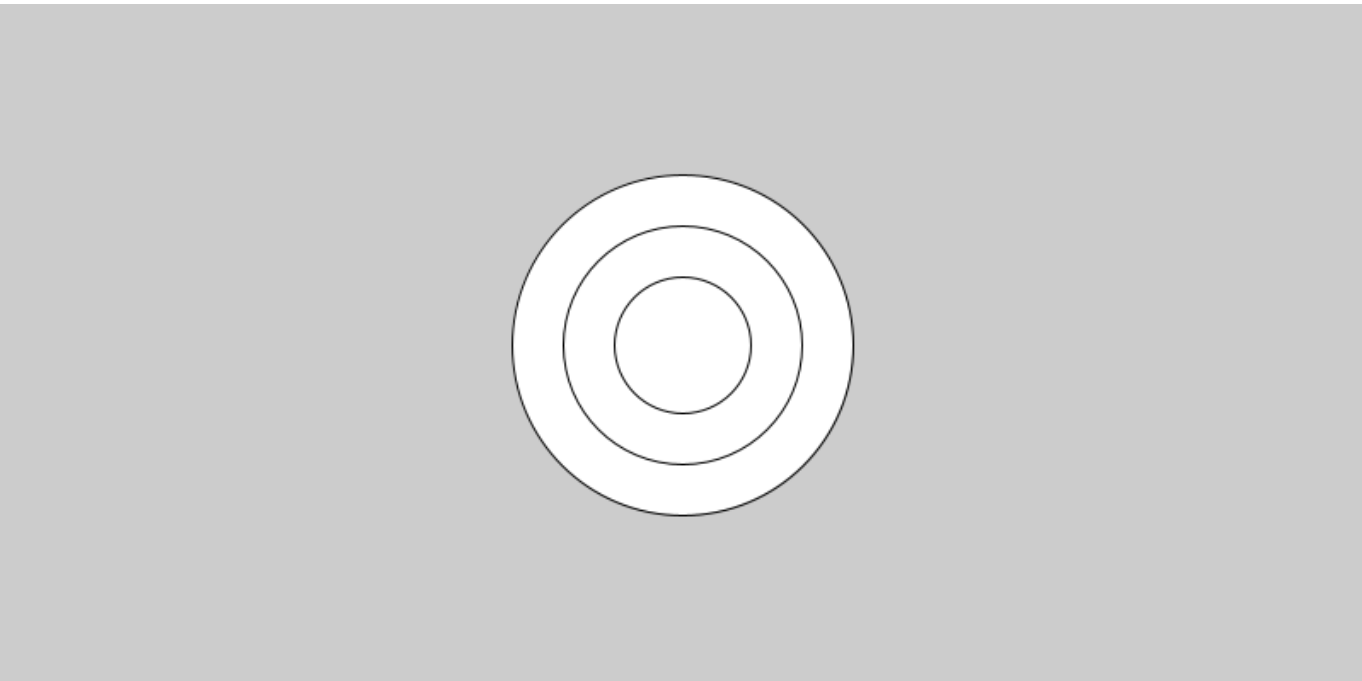


```
ellipse(x, y, width, height)
```



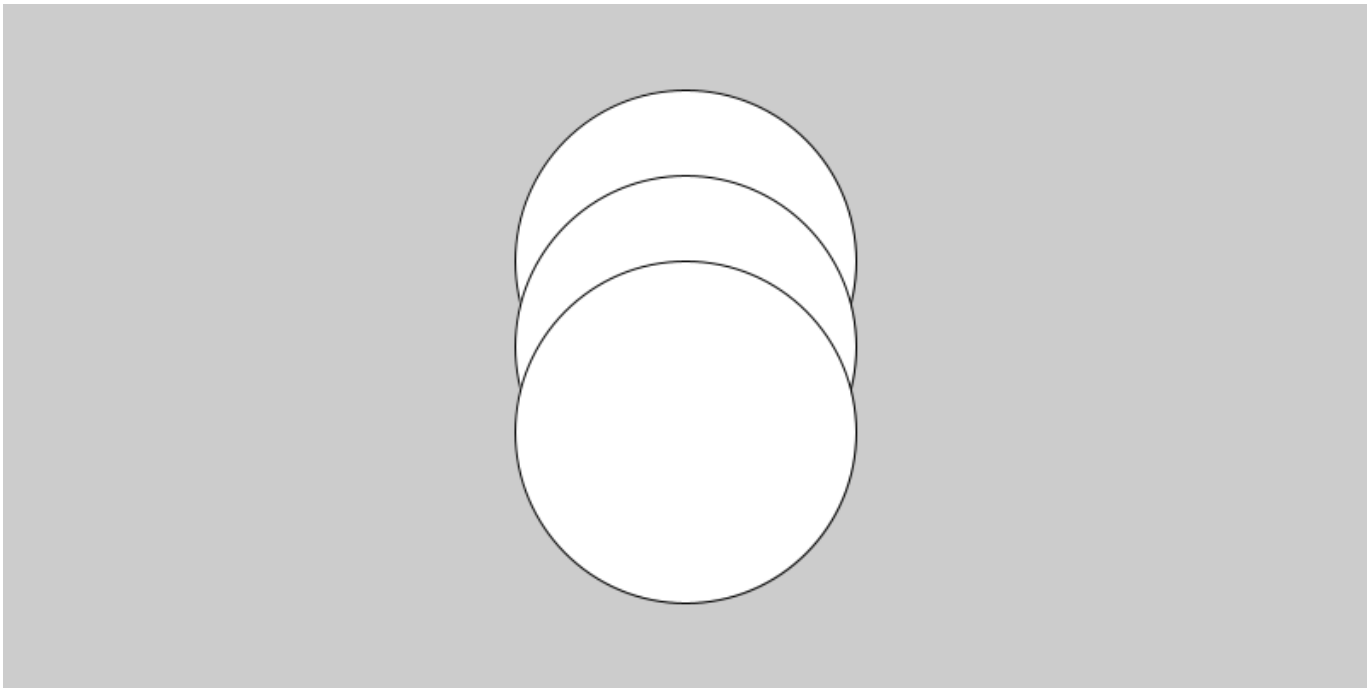
Exercise

Draw 3 concentric circles with radiuses of 100, 70 and 40, just as you see below.

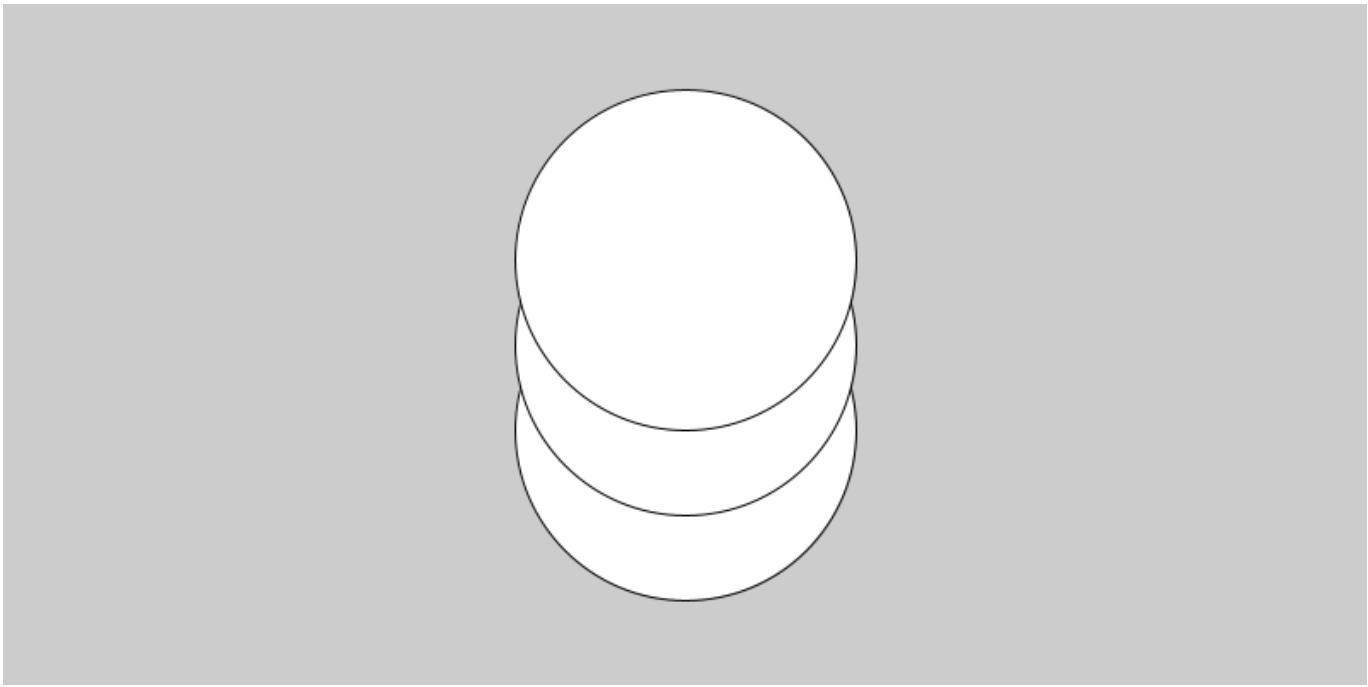


Order matters

```
ellipse(width/2, height/2 - 50, 200, 200);
ellipse(width/2, height/2, 200, 200);
ellipse(width/2, height/2 + 50, 200, 200);
```



```
ellipse(width/2, height/2 + 50, 200, 200);
ellipse(width/2, height/2, 200, 200);
ellipse(width/2, height/2 - 50, 200, 200);
```



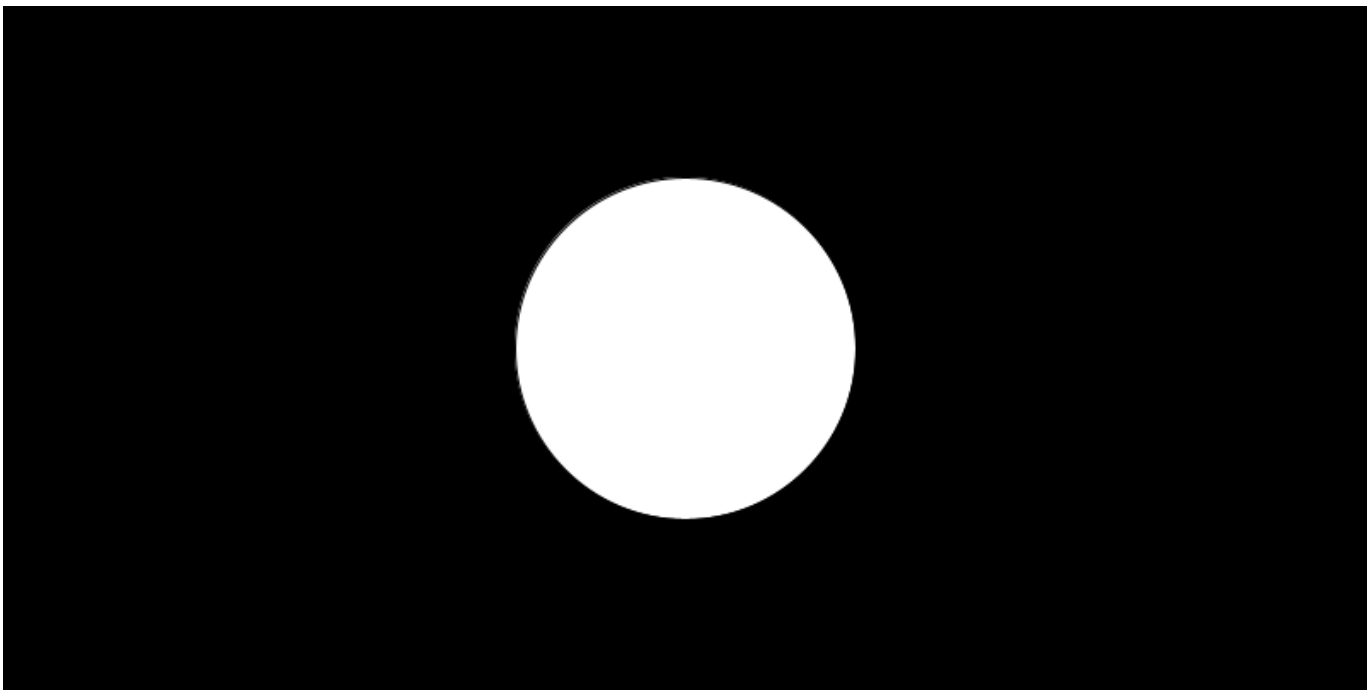
Background

Gray values in the range of [0-255].

0 – black

255 – white

```
size(800, 400);
background(0);
ellipse(width/2, height/2, 200, 200);
```



Fill and stroke

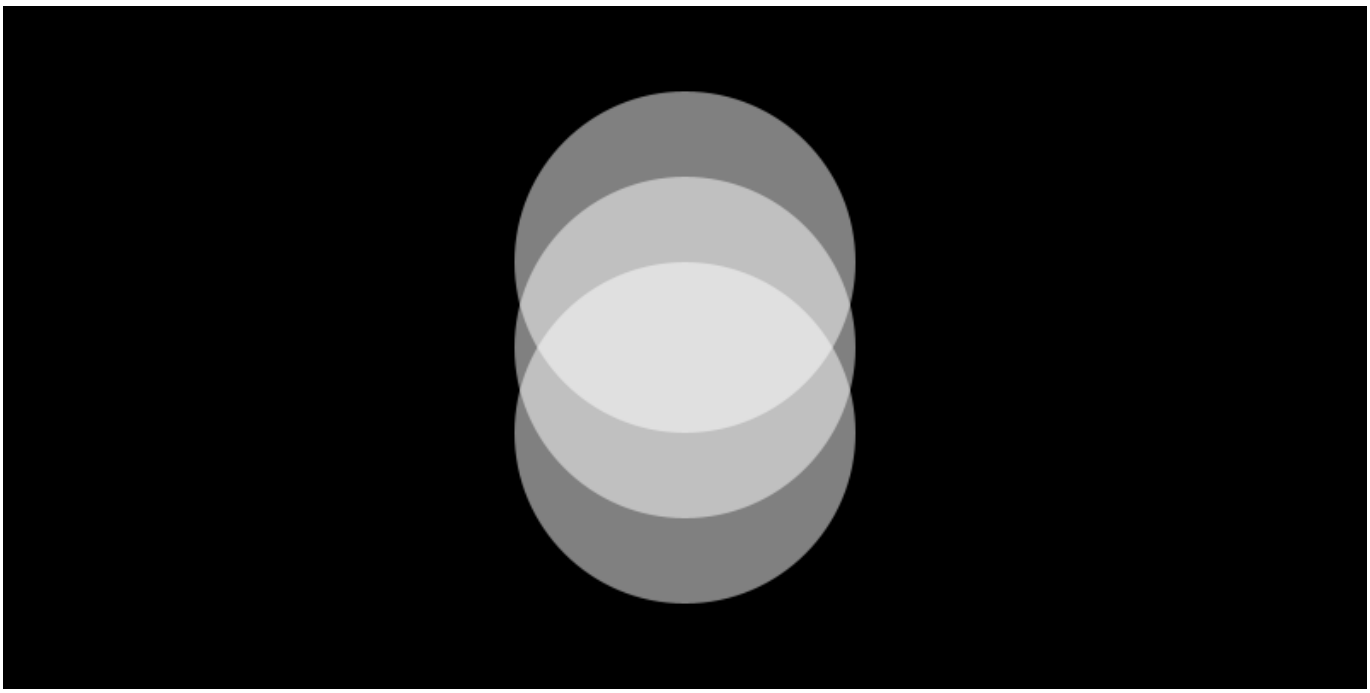
The `fill()` function sets the fill value of shapes, and the `stroke()` function sets the outline value of the drawn shapes. If no fill value is defined, the default value of 255 (white) is used. If no stroke value is defined, the default value of 0 (black) is used. Once a fill or stroke value is defined, it applies to all shapes drawn afterward. To change the fill or stroke value, use the `fill()` or `stroke()` function again. The stroke and fill of a shape can be disabled. The `noFill()` function stops Processing from filling shapes, and the `noStroke()` function stops lines from being drawn and shapes from having outlines.

```
int s = width/5;
int step = 255/6;

noStroke();

fill(step);
rect(0, 0, s, height);

fill(step*2);
rect(s, 0, s, height);

fill(step*3);
rect(s*2, 0, s, height);

fill(step*4);
rect(s*3, 0, s, height);

fill(step*5);
rect(s*4, 0, s, height);
```

Transparency

An optional second parameter to `fill()` and `stroke()` controls transparency. Setting the parameter to 255 makes the shape entirely opaque, and 0 is totally transparent.

```
fill(255, 128);
```

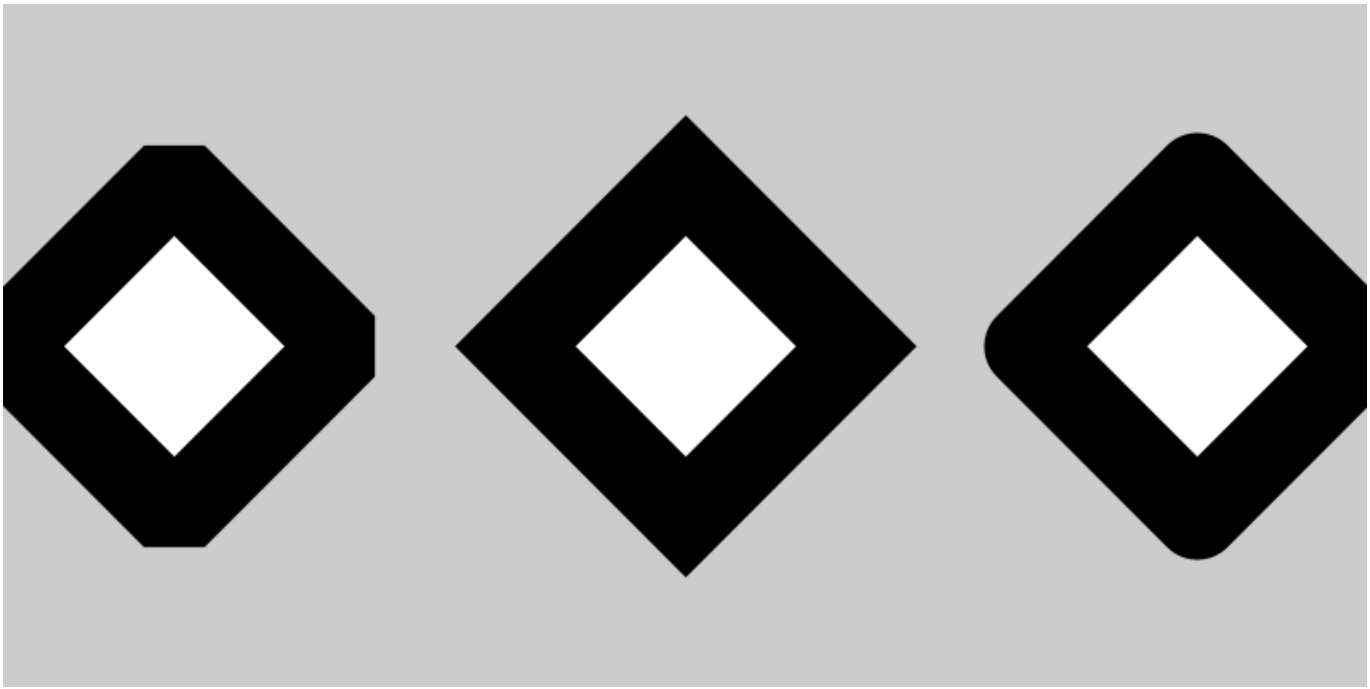


Stroke attributes

Line attributes are controlled by the `strokeWeight()`, `strokeCap()`, and `strokeJoin()` functions. The `strokeWeight()` function has one numeric parameter that sets the thickness of all lines drawn after the function is used. The `strokeCap()` function requires one parameter that can be either `ROUND`, `SQUARE`, or `PROJECT`. The `strokeJoin()` function has one parameter that can be either `BEVEL`, `MITER`, or `ROUND`.

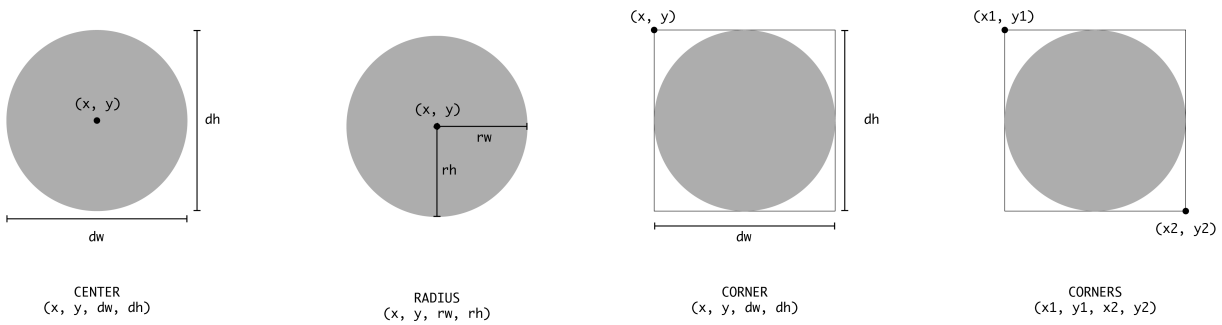
```
int s = 100;
int cx = width/2;
int cy = height/2;

strokeWeight(50);

strokeCap(ROUND);
stroke(0);
line(cx - s, cy + s, cx + s, cy - s);

strokeCap(PROJECT);
stroke(255);
line(cx + s, cy + s, cx - s, cy - s);

strokeCap(SQUARE);
stroke(0);
line(cx + s, cy + s, cx - s, cy - s);
```

```
int s = 100;
int cx = width/2 - 3*s;
int cy = height/2;

strokeWeight(50);

strokeJoin(BEVEL);
quad(cx, cy - s, cx - s, cy, cx, cy + s, cx + s, cy);

cx = width/2;
strokeJoin(MITER);
quad(cx, cy - s, cx - s, cy, cx, cy + s, cx + s, cy);

cx = width/2 + 3*s;
strokeJoin(ROUND);
quad(cx, cy - s, cx - s, cy, cx, cy + s, cx + s, cy);
```



Drawing modes

By default, the parameters for `ellipse()` set the x-coordinate of the center, the y-coordinate of the center, the width, and the height. The `ellipseMode()` function changes the way these parameters are used to draw ellipses. The `ellipseMode()` function requires one parameter that can be either `CENTER`, `RADIUS`, `CORNER`, or `CORNERS`. The default mode is `CENTER`. The `RADIUS` mode also uses the first and second parameters of `ellipse()` to set the center, but changes the third parameter to half of the width and the fourth parameter to half of the height. The `CORNER` mode makes `ellipse()` work similarly to `rect()`. It makes the first and second parameters to position the upper-left corner of the rectangle that encapsulates the ellipse and uses the third and fourth parameters as the width and height. The `CORNERS` mode has a similar affect to CORNER, but turns the third and fourth parameters of `ellipse()` to the lower-right corner of the rectangle.



In a similar fashion, the `rectMode()` function defines how rectangles are drawn. It requires one parameter that can be either `CORNER`, `CORNERS`, or `CENTER`. The default mode is `CORNER`, and `CORNERS` turns the third and fourth parameters of `rect()` into the coordinates of the other

corner of the rectangle. The `CENTER` mode turns the first and second parameters into the center of the rectangle and uses the third and fourth parameters as the width and height.

Exercise

Try to replicate the image below in Processing.