02
기본도형 그리기

02.01
도형그리기 기초

프로세싱 프로그램의 기본구조

지난 시간을 떠올려 볼까요? 원(타원)과 사각형을 그릴 수 있는 함수를 이용해 간단한 도형을 그려보았습니다. 그리고 mouseX와 mouseY를 이용해 마우스를 따라다니는 도형을 그렸는데 void draw(){...}가 등장했습니다. 이것의 정체를 알려면 프로세싱 프로그램의 기본 구조를 알아야 합니다.

모든 프로세싱 프로그램은 setup()안에 작성된 부분과 draw() 함수 안에 작성된 부분으로 나눌 수 있습니다. 이것을 프로세싱 프로그램의 기본 구조라고 부르겠습니다.

```
void setup() {
...
}

void draw() {
...
}
```

우선 setup() 함수 내부의 명령이 프로그램이 시작할 때 단 한번 실행됩니다. 이런 이유로 프로그램이 처음 실행될 때 필요한 명령들을 setup() 함수 내부에 작성합니다. 예를 들어 화면의 크기를 정하는 size() 함수라든지, 도형을 매끄럽게 그리는 명령어 smooth()와 같은 함수는 setup() 함수 안에 둡니다. 이런 이유로 setup() 이라는 이름을 얻었습니다.

draw() 한수는 setup() 함수가 실행된 이후에 다음으로 실행됩니다. draw() 함수도 모두 실행되면 어떻게 될까요? 프로그램이 종료할까요?

아닙니다. draw() 함수 내부의 명령이 한 번 모두 실행된 다음에는 draw() 함수 내부의 첫 줄로 돌아가 다시 명령문을 실행하고 끝에 돌아가면 다시 처음으로 다시 끝으로... 이런 과정을 프로그램이 실행되는 동안 계속됩니다. 1/60초마다 draw() 함수를 실행하는 것이 기본 설정입니다. 그런데 왜 이렇게 할까요?

이렇게 반복해야 애니메이션을 그릴 수 있고 마우스나 키보드 입력에 반응할 수 있습니다. 지난 시간에 mouseX와 mouseY를 처음 소개할 때 이전에 하던 것처럼

```
ellipse(mouseX, mouseY, 60, 60);
```

이라고 입력하면 우리가 원하는 결과가 나오지 않고 원 하나만 화면에 떠있게 됩니다. draw() 함수 안에서 사용하지 않았기 때문에 반복실행되지 않고 단 한번만 실행된 이후 프로그램이 종료되기 때문에 처음 마우스 위치에서 원을 그리고는 더이상 처리할 코드가 없어 프로세싱이 가만히 있는 것입니다.

지금부터 작성할 모든 프로그램에 setup()과 draw()를 적용하겠습니다.

좌표

기본도형을 그리는 여러 함수가 있지만 일단 도형의 위치와 크기를 정하는 값을 입력받는 것을 기본으로 합니다.

도형의 위치는 왼쪽 상단을 (0, 0)으로 삼습니다. 많은 그래픽 프로그램이 이런 좌표체계를 따릅니다. 위치와 크기의 단위는 픽셀(pixel)입니다. 화면의 점 하나가 픽셀 하나입니다.

모니터의 크기에 맞게 실행창 크기를 정하세요. 실행창의 크기는 size()함수를 이용해 정할 수 있습니다. 예를 들어 가로 640픽셀, 세로 480픽셀 크기의 실행창을 정의하려면

```
void setup() {
    size(640, 480);
}
```

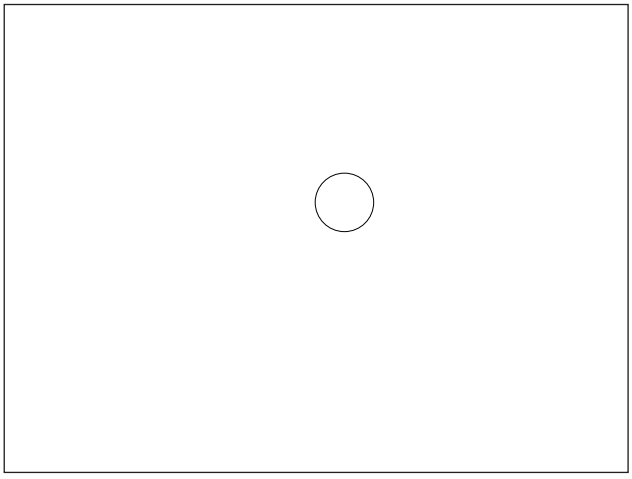
처럼 setup() 함수 내부에 size() 함수를 실행하세요.

background() 함수

지난 시간을 떠올려보면 나중에 그린 도형이 이전 도형 위에 누적되어 그려집니다. 나중에 심해지면 화면을 까맣게 모두 채워 더이상 구별이 힘들어지는 경우도 있습니다. 이를 해결하는 방법을 살펴봅시다. 이것은 draw() 함수를 활용하는 방법입니다. 아이디어는 간단합니다. 매번 draw() 함수가 새로운 도형을 그릴 때마다 우선 바탕화면을 다시 하얗게 그립니다. 그리고 새로운 도형을 그리면 매 프레임마다 새로 그려지는 도형만 보게 됩니다.

배경화면을 칠하는 함수는 background()입니다. 아래의 코드에서 background()의 draw() 함수 내부에서의 위치를 잘 살펴보세요.

```
void setup() {
    size(640, 480);
}

void draw() {
    background(255); //흰 바탕화면
    ellipse(mouseX, mouseY, 60, 60);
}
```



draw() 함수 가장 상단에 background() 함수가 위치해 매 프레임을 그릴 때마다 바탕화면은 흰 색으로 우선 칠하고 원을 그립니다. 그래서 움직이는 원만 볼 수 있습니다. draw() 함수가 어떻게 동작하는지 알게되면 쉽게 이해할 수 있는 기술입니다.

02.02
기본 도형 그리기

이제 여러 기본도형을 알아보고 어떤 함수와 어떤 입력값을 이용해 원하는 모양을 그릴 수 있는지 살펴봅시다.

점

point() 함수는 화면에 점을 찍습니다. x좌표와 y좌표를 입력값으로 가집니다.

기본 형식: point(x1, y1);

```
void setup() {
    size(480, 360);
    strokeWeight(5);
}
void draw() {
    point(120, 120);
    point(120, 240);
    point(360, 120);
    point(360, 240);
}
```

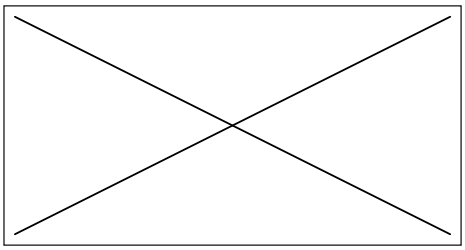
화면에 먼지묻는 것 아닙니다. 크게 눈을 뜨고 찾아봅시다.

선

직선은 두 개의 점으로 특정됩니다. line() 함수는 두 개의 점을 잇는 선분을 그립니다.

기본 형식: line(x1, y1, x2, y2);

```
void setup() {
    size(480, 360);
}

void draw() {
    line(120, 120, 360, 240);
    line(120, 240, 360, 120);
}
```

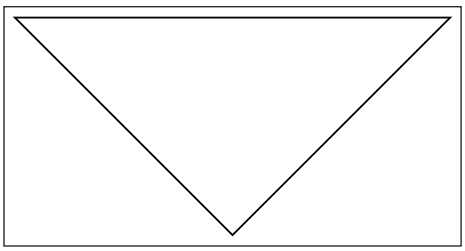


삼각형

하나의 삼각형은 세 개의 점으로 특정할 수 있습니다. triangle() 함수는 세 점의 위치를 입력값으로 받습니다. 점의 순서는 상관없습니다.

기본 형식: triangle(x1, y1, x2, y2, x3, y3);

```
void setup() {
    size(480, 360);
}

void draw() {
    triangle(120, 120, 360, 120, 240, 240);
}
```



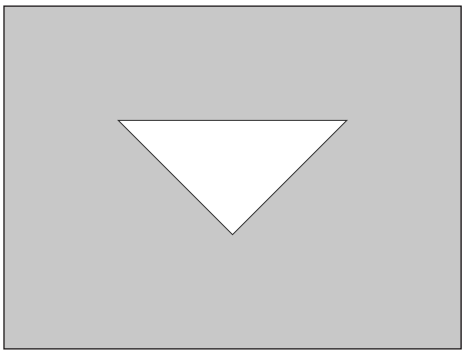
삼각형 꾸미기

지금까지 도형을 그릴 때 도형의 획이나 색을 따로 정하지 않고 기본값을 사용했습니다. 이번 기회에 획의 두께를 바꾸고 다른 색으로 도형 내부를 채우는 등 도형을 꾸미는 방법을 알아봅시다.

도형의 외각선을 획이라고 합니다. 획의 색을 바꿀 때는 stroke() 함수를 사용합니다. stroke() 함수는 획의 색을 입력값으로 받습니다. 예를 들어 흰 색은 숫자 255에 대응됩니다. 검은색은 반대로 숫자 0에 대응됩니다. 0과 255사이의 값을 이용해 회색의 진하기를 표현할 수 있습니다.

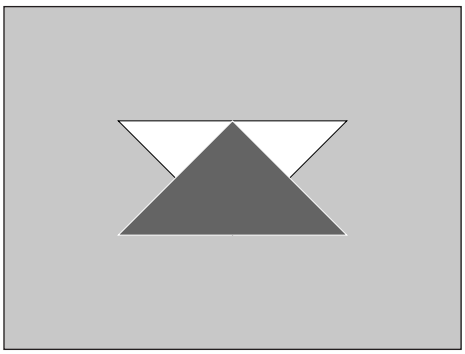
바탕화면을 어둡게 하고 삼각형 획을 밝은 색으로 정해봅시다. 획을 뚜렷하게 보기 위해 도형의 내부가 색으로 채워지지 않게 함수 noFill()을 setup() 내부에 둡니다.

```
void setup() {
    size(480, 360);
    noFill();
}

void draw() {
    background(200);
    stroke(50);
    triangle(120, 120, 360, 120, 240,
240);
}
```



이제 삼각형 내부를 밝은 회색으로 칠합니다. filll() 함수를 이용합니다. fill() 함수도 stroke() 함수와 마찬가지로 채우기 색을 숫자로 입력받습니다.

```
void setup() {
    size(480, 360);
}

void draw() {
    background(200);
    stroke(0);
    triangle(120, 120, 360, 120, 240,
240);

    stroke(255);
    fill(100);
    triangle(120, 240, 360, 240, 240,
120);
}
```



stroke()과 fill()이 한 번 실행된 이후에 그려지는 도형은 다시 stroke()과 fill()을 실행하지 않더라도 계속 영향을 끼칩니다.

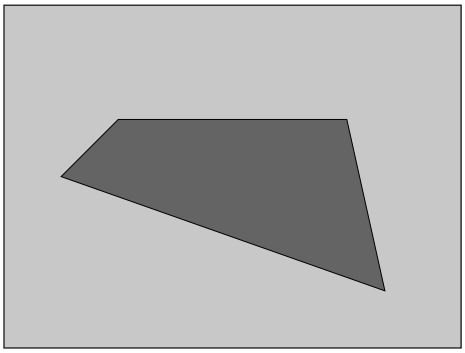
사변형

사변형은 네 개의 점으로 정의되는 도형입니다. quadrilateral(사변형)의 처음 4자를 따서 quad()라고 함수의 이름을 지었습니다. triangle() 함수와 마찬가지로 점을 차례로 입력합니다. 주의할 점은 quad() 함수에 점을 입력할 때는 입력하는 점의 순서의

차이로 다른 도형이 그려질 수 있다는 사실입니다. 점을 입력할 때는 시계방향이나 반시계방향으로 차례로 입력합니다.

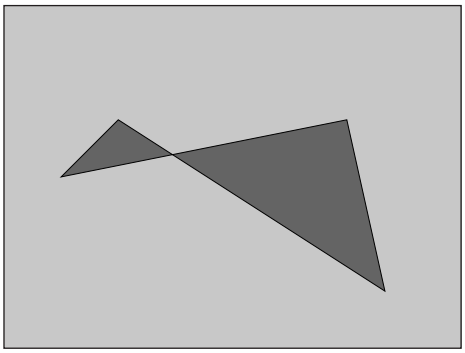
기본 형식: quad(x1, y1, x2, y2, x3, y3, x4, y4);

```
void setup() {
    size(480, 360);
}

void draw() {
    background(200);
    stroke(0);
    fill(100);
    quad(120, 120, 360, 120, 400, 300, 60, 180);
}
```



입력순서를 바꾸면 의도하지 않은 도형이 그려집니다.

```
void setup() {
    size(480, 360);
}

void draw() {
    background(200);
    stroke(0);
    fill(100);
    quad(120, 120, 400, 300, 360, 120, 60, 180);
}
```



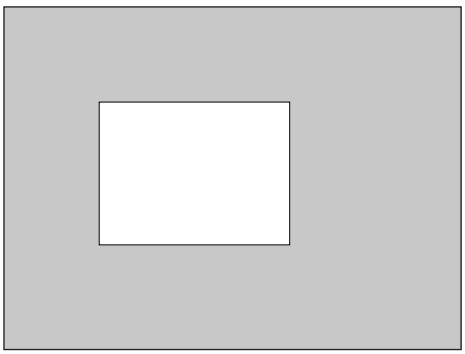
사각형

사각형을 그리는 함수는 rect()입니다. rectangle의 처음 4자를 따온 이름입니다. quad() 함수를 사용해 사각형을 그릴 수도 있지만 자주 사용하는 도형인만큼 더 편하고 직관적인 방식을 사용합니다. rect() 함수는 위치를 나타내는 x, y를 입력받고 사각형의 너비와 높이를 정의하는 방식을 따릅니다.

기본 형식: rect(x, y, width, height);

```
void setup() {
    size(480, 360);
}
```

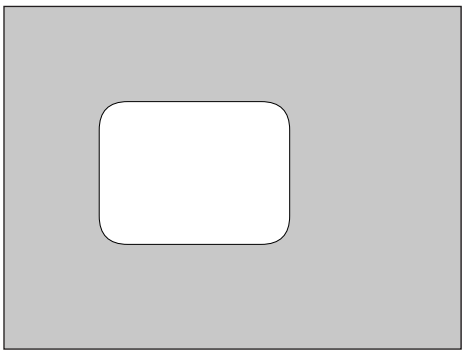
```
void draw() {
    background(200);
    stroke(0);
    fill(255);
    rect(100, 100, 200, 150);
}
```



rect() 함수에 입력을 추가할 수 있습니다. 5번째 입력 값은 네 모서리의 반지름값으로 해석됩니다. 값이 클수록 둥근 모서리를 가지는 사각형을 그립니다.

기본 형식: `rect(x, y, width, height, radius);`

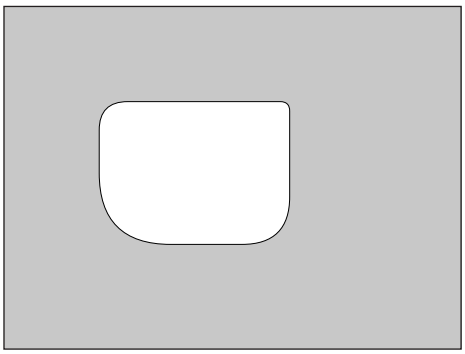
```
void setup() {
    size(480, 360);
}

void draw() {
    background(200);
    stroke(0);
    fill(255);
    rect(100, 100, 200, 150, 30);
}
```



각 모서리마다 다른 반지름을 지정할 수 있습니다. 이런 경우 rect() 함수는 총 8개의 입력값에 대응해 사각형을 그립니다.

기본 형식: `rect(x, y, width, height, top-left, top-right, bottom-right, bottom-left);`

```
void setup() {
    size(480, 360);
}

void draw() {
    background(200);
    stroke(0);
    fill(255);
    //모서리 radius는 시계방향으로 입력됩니다
    rect(100, 100, 200, 150, 30, 10, 50,
```

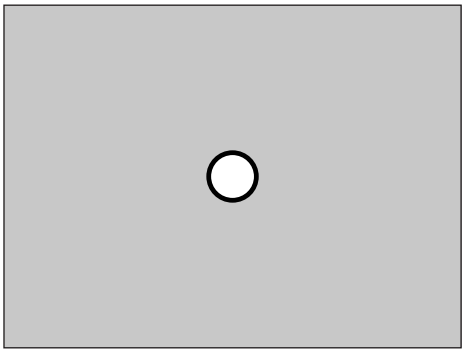


```
100);
}
```

원

첫번째 프로세싱 프로그램에서 이미 사용한 적이 있습니다. 프로세싱은 원을 따로 정의하지 않고 폭과 너비가 같은 특별한 타원형으로 다룹니다.

```
기본 형식: ellipse(x, y, width, height);

void setup() {
    size(480, 360);
}

void draw() {
    background(200);
    stroke(0);
    strokeWeight(5);
    fill(255);
    ellipse(240, 180, 50, 50);
}
```

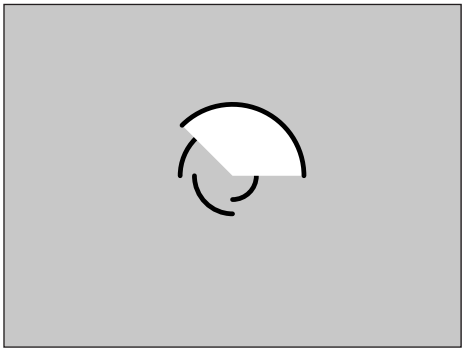


strokeWeight() 함수를 사용해 획(stroke)의 두께를 5픽셀로 고쳤습니다. 두꺼운 선을 이용해 원을 그리는 것을 관찰할 수 있습니다.

호(arc)

호는 원의 일부를 말합니다. arc() 함수로 호를 그릴 수 있습니다. ellipse() 함수의 입력값을 기본으로 호를 그리기 시작하는 각도와 마치는 각도를 추가로 지정해야 합니다.

```
기본 형식: arc(x, y, width, height, start,
stop);

void setup() {
    size(480, 360);
}

void draw() {
    background(200);
    stroke(0);
    strokeWeight(5);
    noFill();
    arc(240, 180, 50, 50, 0, HALF_PI);
    arc(240, 180, 80, 80, HALF_PI, PI);
    arc(240, 180, 110, 110, PI, PI +
QUARTER_PI);
```



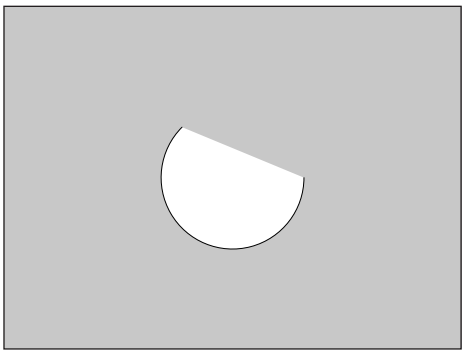
```
fill(255);
arc(240, 180, 150, 150, PI + QUARTER_PI,
TWO_PI);
}
```

7번째 매개변수 mode를 이용해 호를 그리는 방법을 달리 할 수 있습니다.

기본 형식: `arc(x, y, width, height, start, stop, mode);`

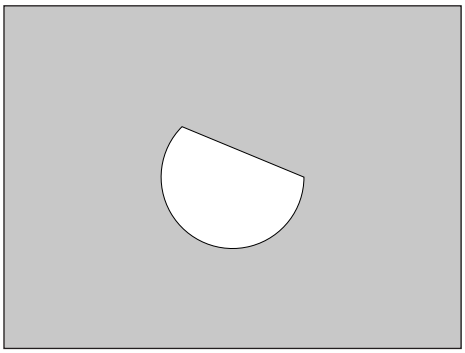
OPEN

```
void setup() {
    size(480, 360);
}

void draw() {
    background(200);
    arc(240, 180, 150, 150, 0, PI +
QUARTER_PI, OPEN);
}
```



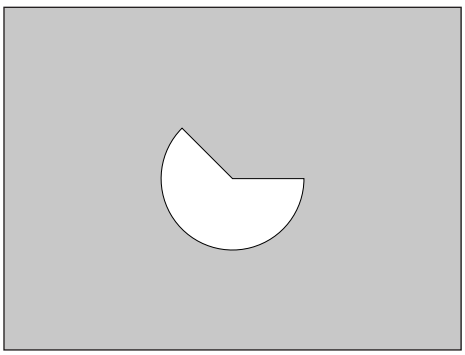
CHORD (시위 혹은 시위선)

```
void setup() {
    size(480, 360);
}

void draw() {
    background(200);
    arc(240, 180, 150, 150, 0, PI +
QUARTER_PI, CHORD);
}
```



PIE

```
void setup() {
    size(480, 360);
}

void draw() {
    background(200);
    arc(240, 180, 150, 150, 0, PI +
QUARTER_PI, PIE);
}
```



사용자 정의 도형(vertex)

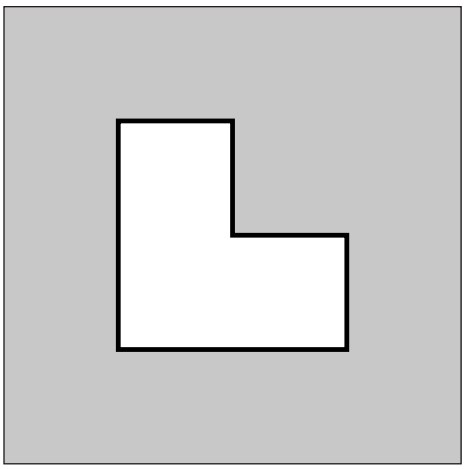
프로세싱은 기본도형을 그리는 다양한 함수를 제공하고 있지만 좀 더 복잡한 도형을 그리는 데는 부족함이 있습니다. 복잡한 도형에 대해 일일이 함수를 하나씩 만드는 대

신 프로세싱은 범용으로 쓸 수 있는 함수 vertex() 를 제공합니다. vertex()는 도형의 꼭지점을 차례로 지정합니다. 꼭지점을 하나씩 지정할 수 있으면 아무리 복잡한 도형이라도 vertex() 함수를 적용할 수 있습니다.

vertex()를 적용할 때는 나열하는 꼭지점이 하나의 도형이라는 것을 반드시 알려주어야 합니다. 나열하는 꼭지점을 beginShap()과 endShape() 사이에 둡니다.

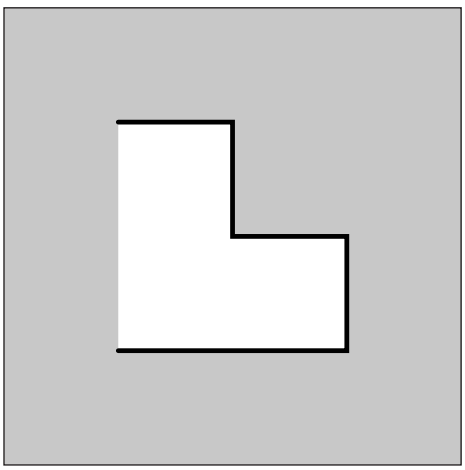
기본 형식:
```
beginShape();
      vertex(x1, y1);
      ...
      vertex(xn, yn);
endShape(CLOSE);
```

```
void setup() {
    size(480, 480);
    strokeWeight(5);
}

void draw() {
    background(200);
    beginShape();
          vertex(120, 120);
          vertex(240, 120);
          vertex(240, 240);
          vertex(360, 240);
          vertex(360, 360);
          vertex(120, 360);
    endShape(CLOSE);
}
```



만약 마지막 endShape()에 CLOSE를 넣지 않고 비워두면 다른 도형이 그려집니다. 차이를 이해하세요.

```
void setup() {
    size(480, 480);
}

void draw() {
    background(200);
    beginShape();
          vertex(120, 120);
          vertex(240, 120);
          vertex(240, 240);
          vertex(360, 240);
          vertex(360, 360);
          vertex(120, 360);
    endShape();
}
```

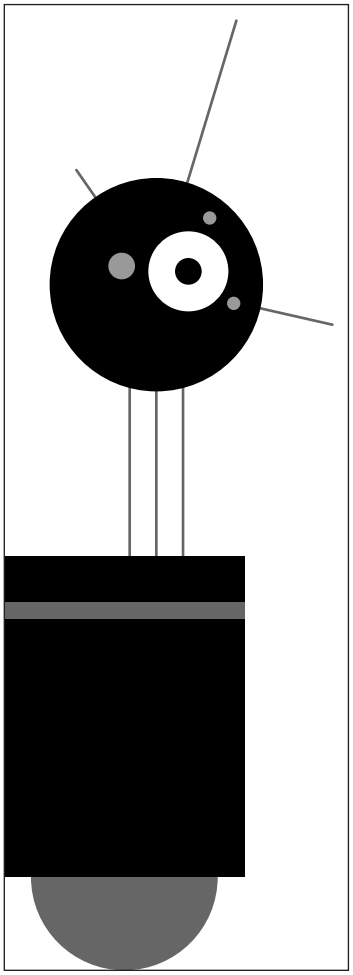


02.03
도형 그리기 실습

P5 그리기

기본 도형을 활용해 인공위성을 그려봅시다.

```
void setup() {
 size(640, 480);
 smooth();
 strokeWeight(2);
 background(204);
}

void draw() {
 //neck
 stroke(102);
 line(266, 257, 266, 162);
 line(276, 257, 276, 162);
 line(286, 257, 286, 162);

 //antenna
 line(276, 155, 246, 112);
 line(276, 155, 306, 56);
 line(276, 155, 342, 170);

 //body
 noStroke();
```



```
    fill(102);
    ellipse(264, 377, 70, 70);
    fill(0);
    rect(219, 257, 90, 120);
    fill(102);
    rect(219, 274, 90, 6);

    //head
    fill(0);
    ellipse(276, 155, 80, 80);
    fill(255);
    ellipse(288, 150, 30, 30);
    fill(0);
    ellipse(288, 150, 10, 10);
    fill(153);
    ellipse(263, 148, 10, 10);
    ellipse(296, 130, 5, 5);
    ellipse(305, 162, 5, 5);
}
```

과제

아래의 도형을 기본도형을 이용해 그려보세요.