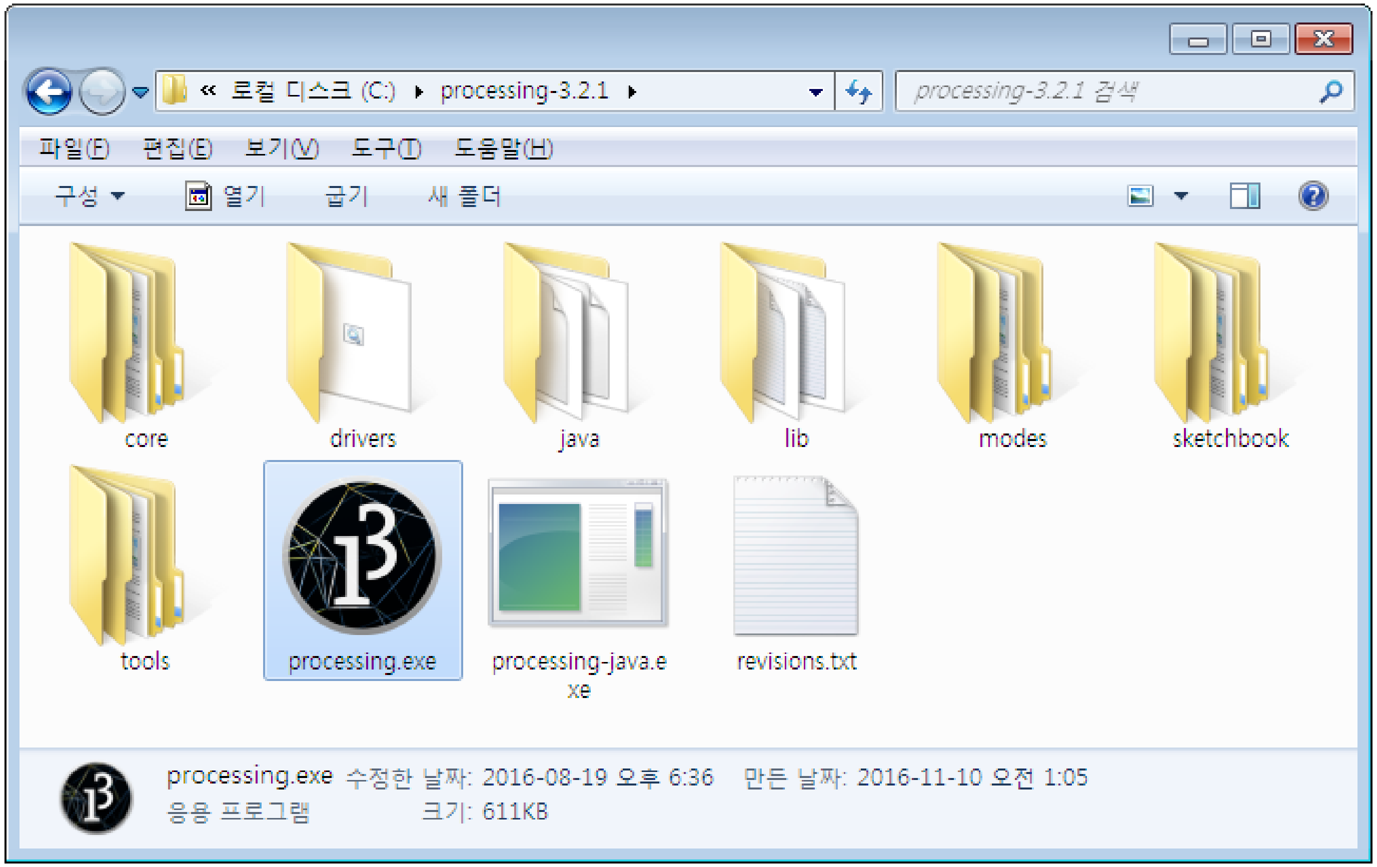
프로세싱

광운대학교 로봇학부
박광현

프로세싱 실행

- C:₩processing-3.2.1 폴더

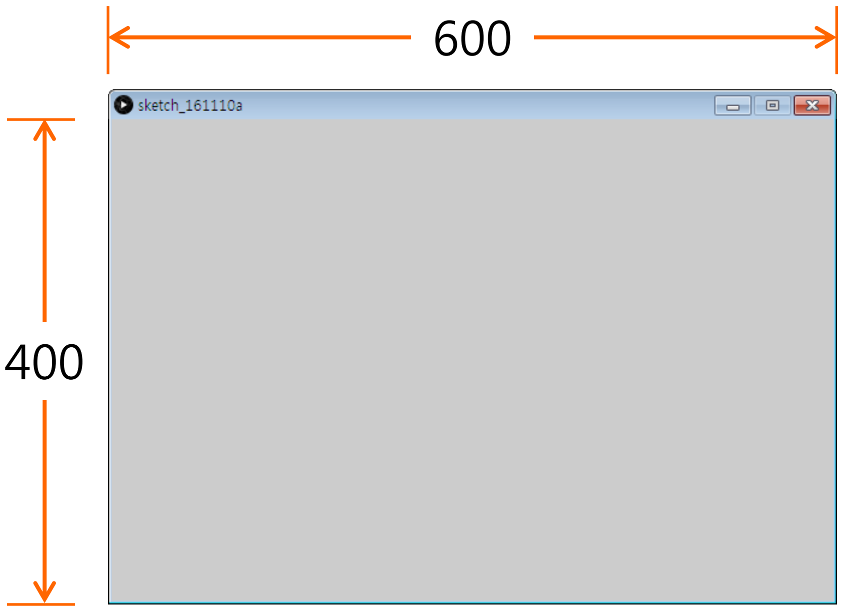


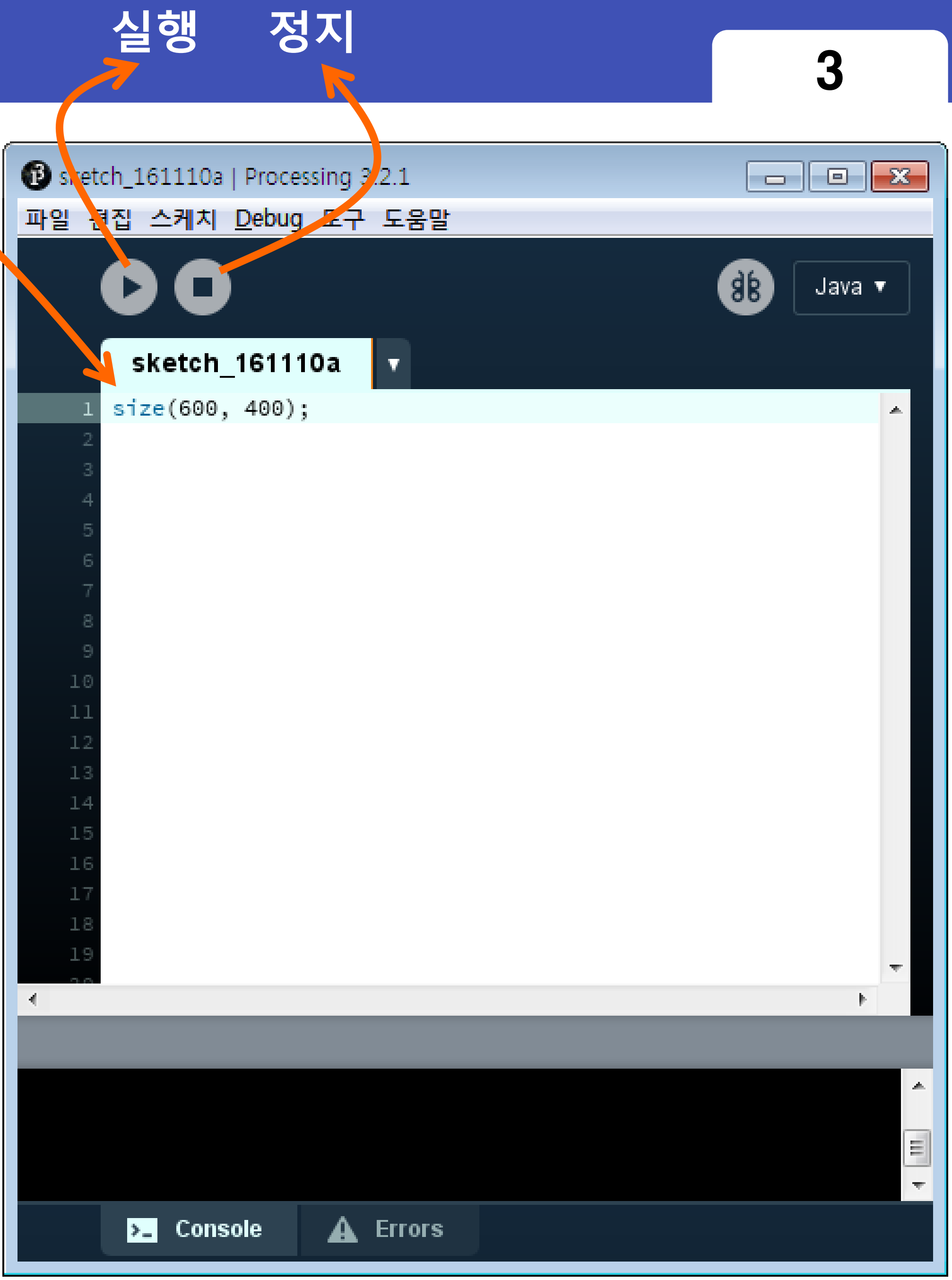
processing.exe 수정한 날짜: 2016-08-19 오후 6:36 만든 날짜: 2016-11-10 오전 1:05
응용 프로그램 크기: 611KB

창 나타내기

```
size(600, 400);
```

폭 높이




실행
정지
sketch_161110a | Processing 3.2.1
파일 편집 스케치 Debug 도구 도움말
Java ▼
sketch_161110a
Console
Errors

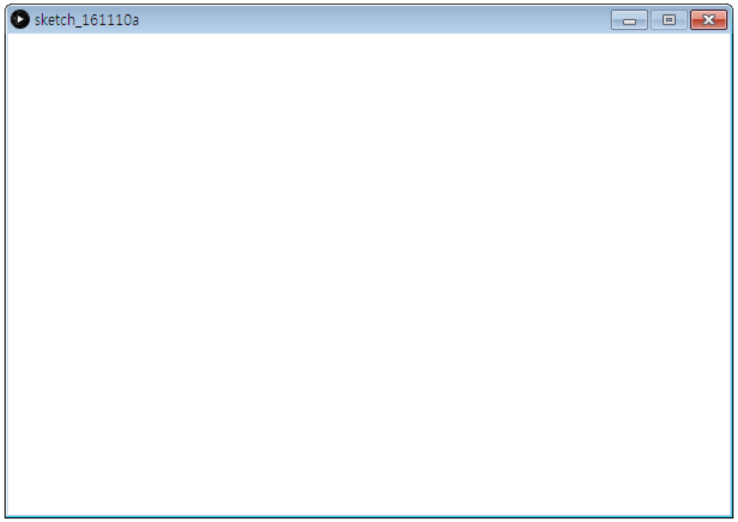
```
size(600, 400);
```

도형 그리기

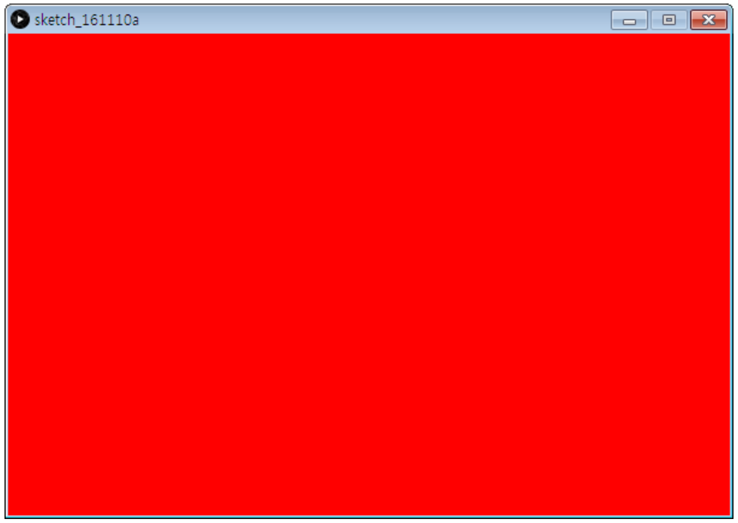
배경 칠하기

```
size(600, 400);
background(255, 255, 255);
```

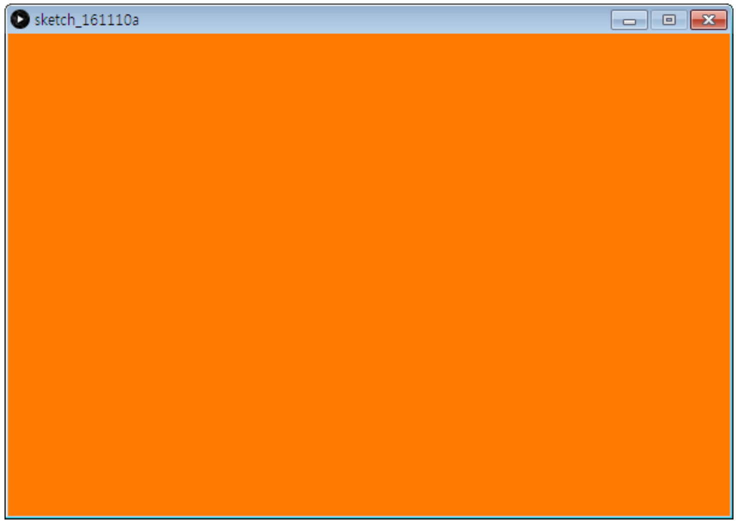
R G B



```
size(600, 400);
background(255, 0, 0);
```



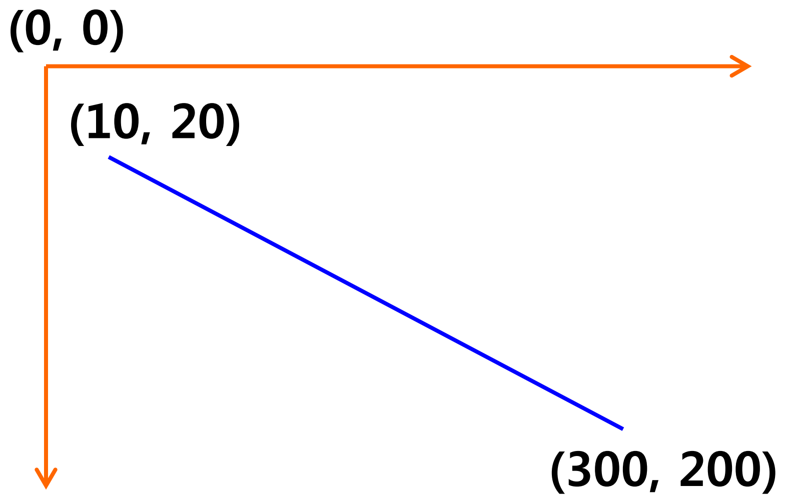
```
size(600, 400);
background(255, 122, 0);
```

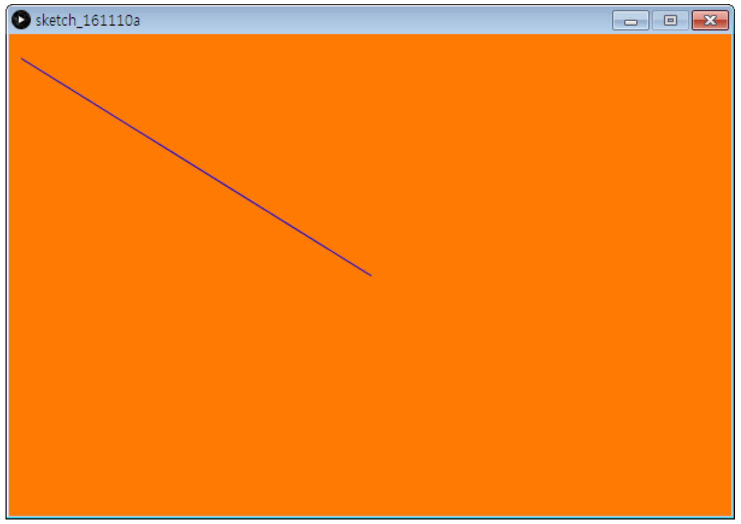


선 그리기

```
size(600, 400);
background(255, 122, 0);
stroke(0, 0, 255);
line(10, 20, 300, 200);
```

$\rightarrow$ 선 색깔 (그리기 전에 설정한다)

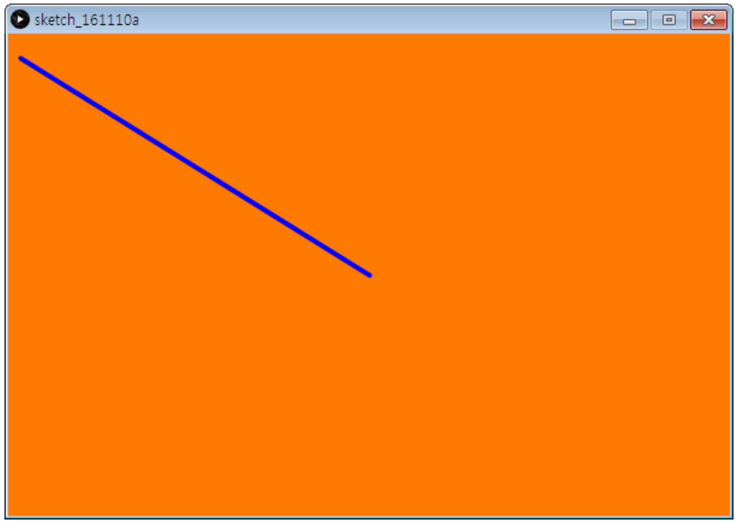




선 두께

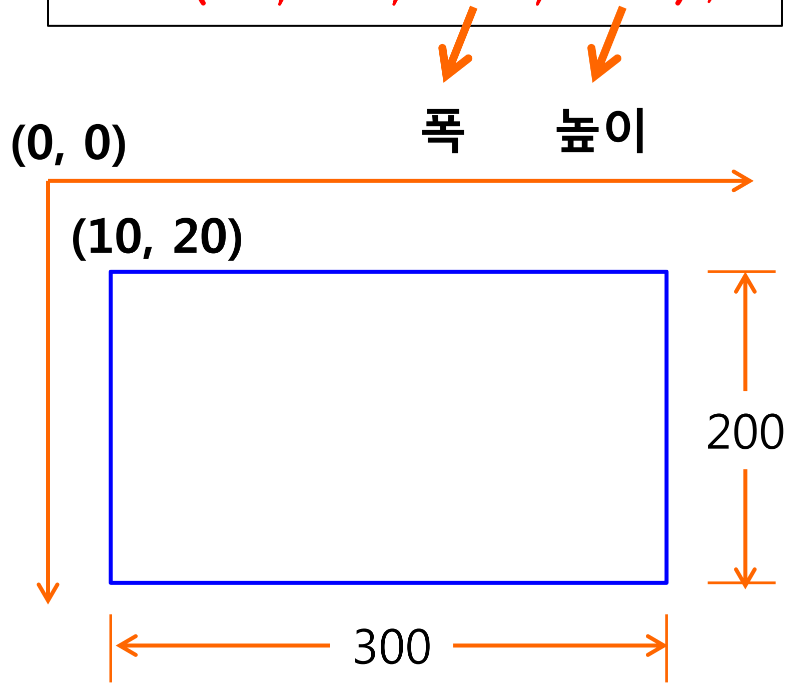
```
size(600, 400);
background(255, 122, 0);
stroke(0, 0, 255);
strokeWeight(4);
line(10, 20, 300, 200);
```

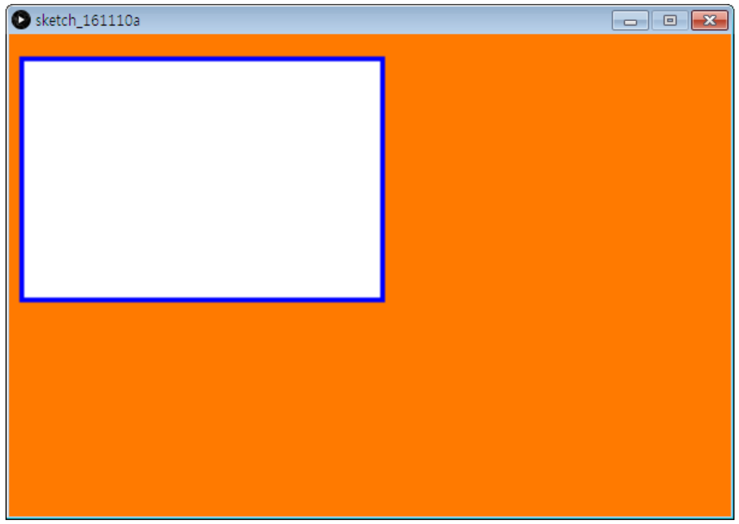
→ 선 두께 (픽셀) (그리기 전에 설정한다)



직사각형 그리기

```
size(600, 400);
background(255, 122, 0);
stroke(0, 0, 255);
strokeWeight(4);
rect(10, 20, 300, 200);
```

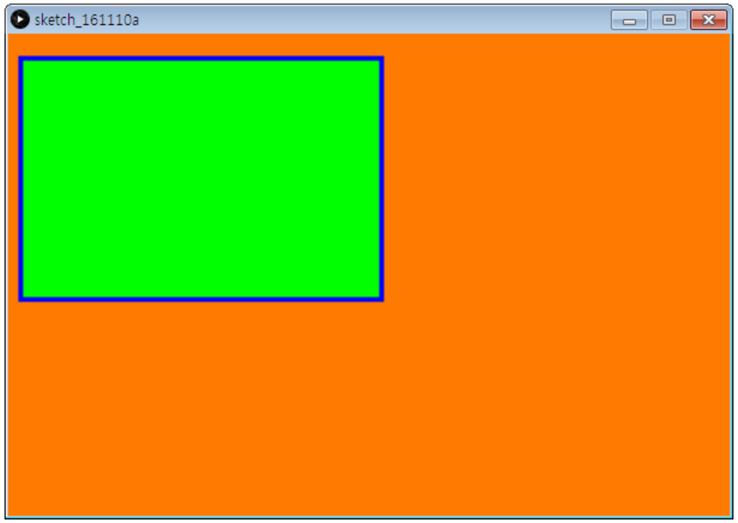




색깔 채우기

```
size(600, 400);
background(255, 122, 0);
stroke(0, 0, 255);
strokeWeight(4);
fill(0, 255, 0);
rect(10, 20, 300, 200);
```

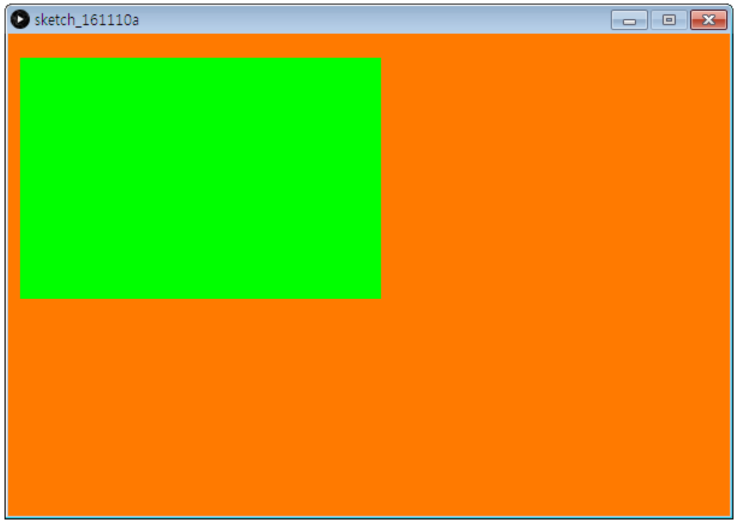
→ 채우기 색깔 (그리기 전에 설정한다)

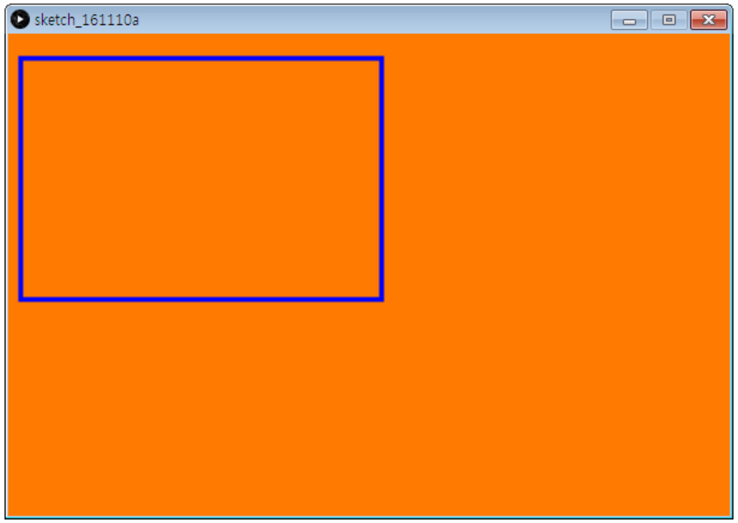


선/채움 없애기

```
size(600, 400);
background(255, 122, 0);
stroke(0, 0, 255);
strokeWeight(4);
fill(0, 255, 0);
noStroke();
rect(10, 20, 300, 200);
```

```
size(600, 400);
background(255, 122, 0);
stroke(0, 0, 255);
strokeWeight(4);
fill(0, 255, 0);
noFill();
rect(10, 20, 300, 200);
```

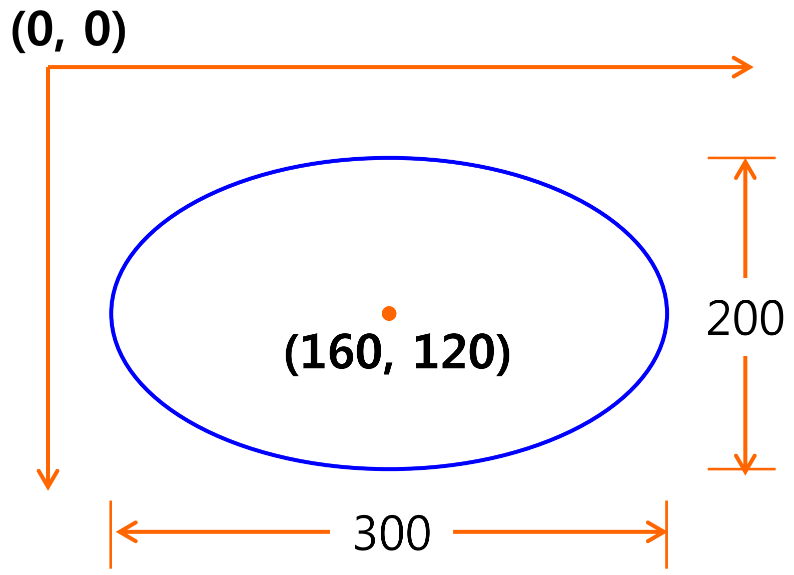


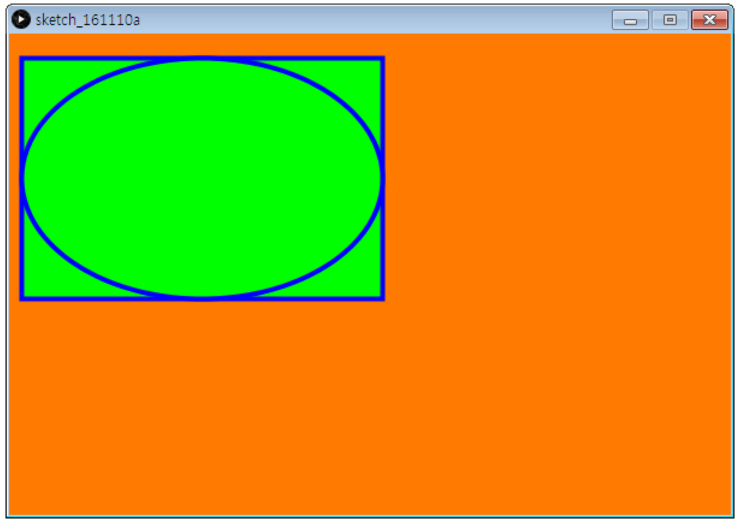


타원 그리기

```
size(600, 400);
background(255, 122, 0);
stroke(0, 0, 255);
strokeWeight(4);
fill(0, 255, 0);
rect(10, 20, 300, 200);
ellipse(160, 120, 300, 200);
```

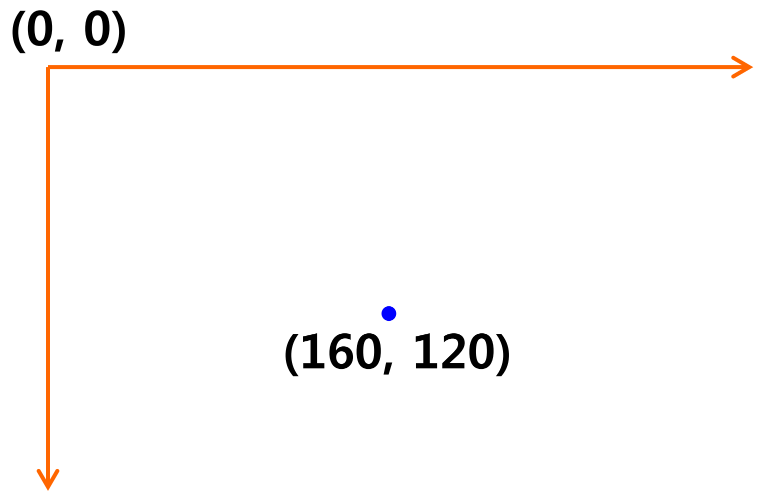
폭　　높이

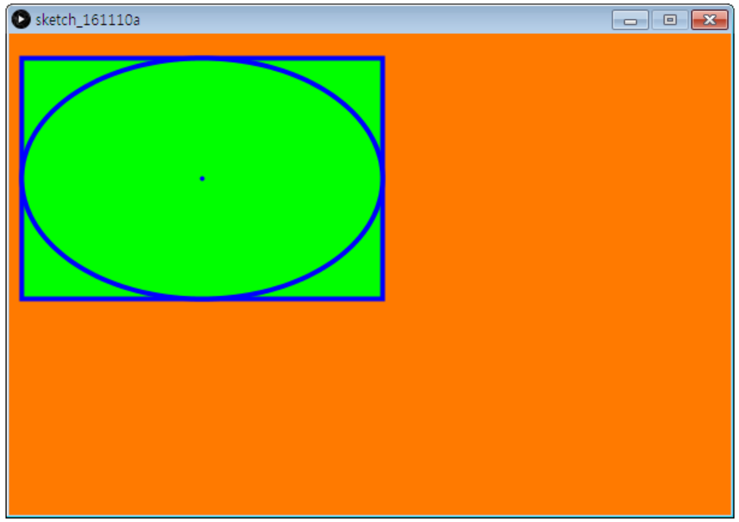




점 그리기

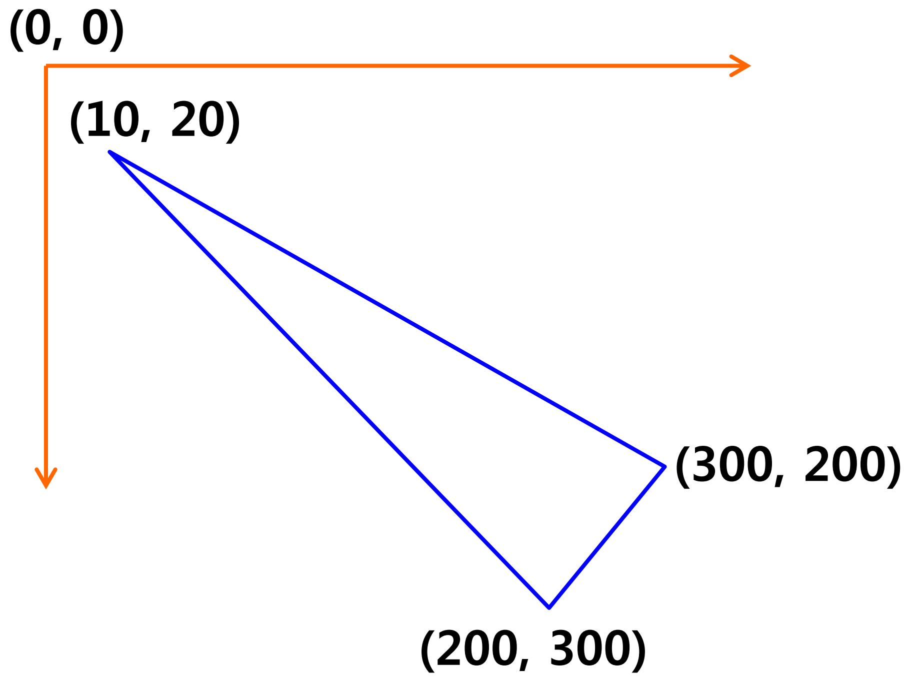
```
size(600, 400);
background(255, 122, 0);
stroke(0, 0, 255);
strokeWeight(4);
fill(0, 255, 0);
rect(10, 20, 300, 200);
ellipse(160, 120, 300, 200);
point(160, 120);
```

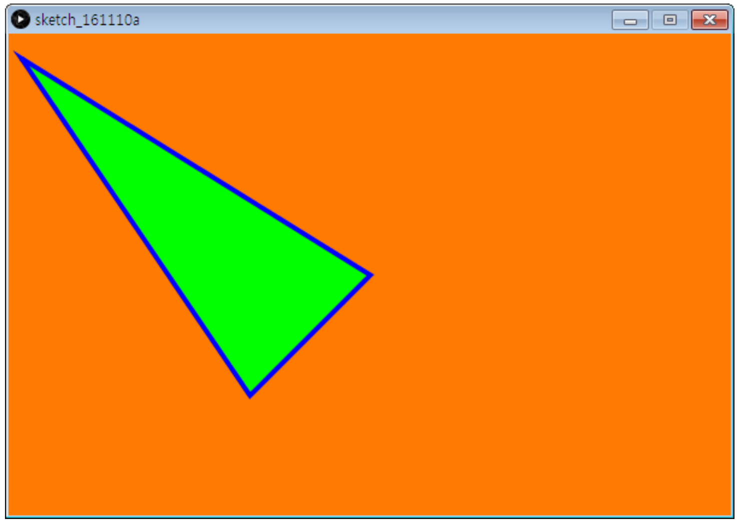




삼각형 그리기

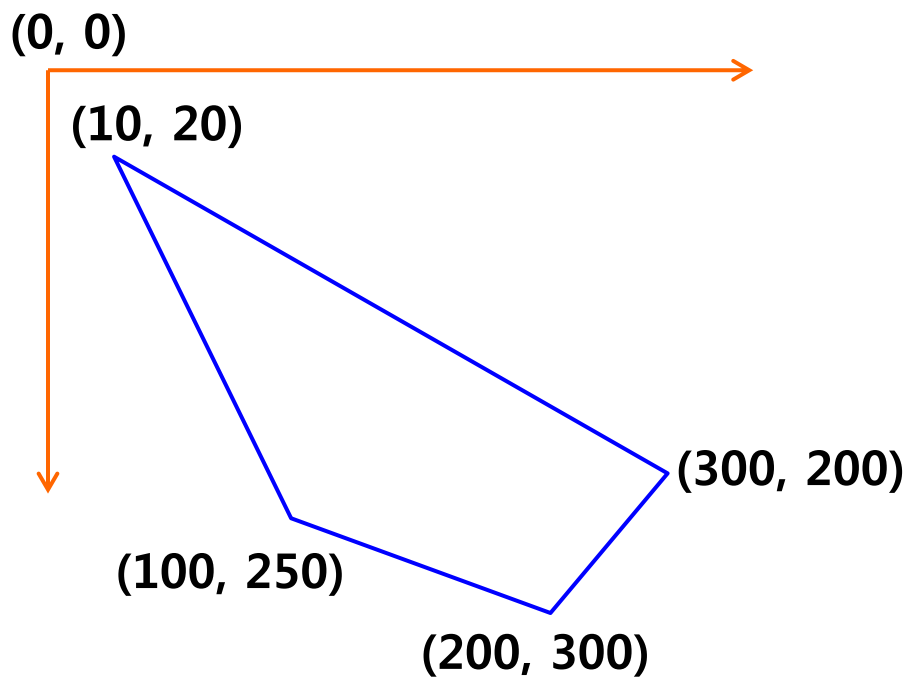
```
size(600, 400);
background(255, 122, 0);
stroke(0, 0, 255);
strokeWeight(4);
fill(0, 255, 0);
triangle(10, 20, 300, 200, 200, 300);
```

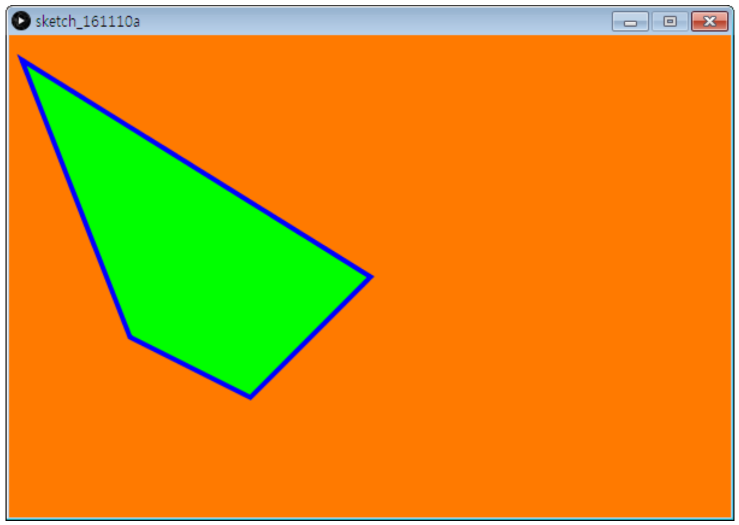




사각형 그리기

```
size(600, 400);
background(255, 122, 0);
stroke(0, 0, 255);
strokeWeight(4);
fill(0, 255, 0);
quad(10, 20, 300, 200, 200, 300, 100, 250);
```





애니메이션

기본 형태

```
void setup() {

}

void draw() {

}
```

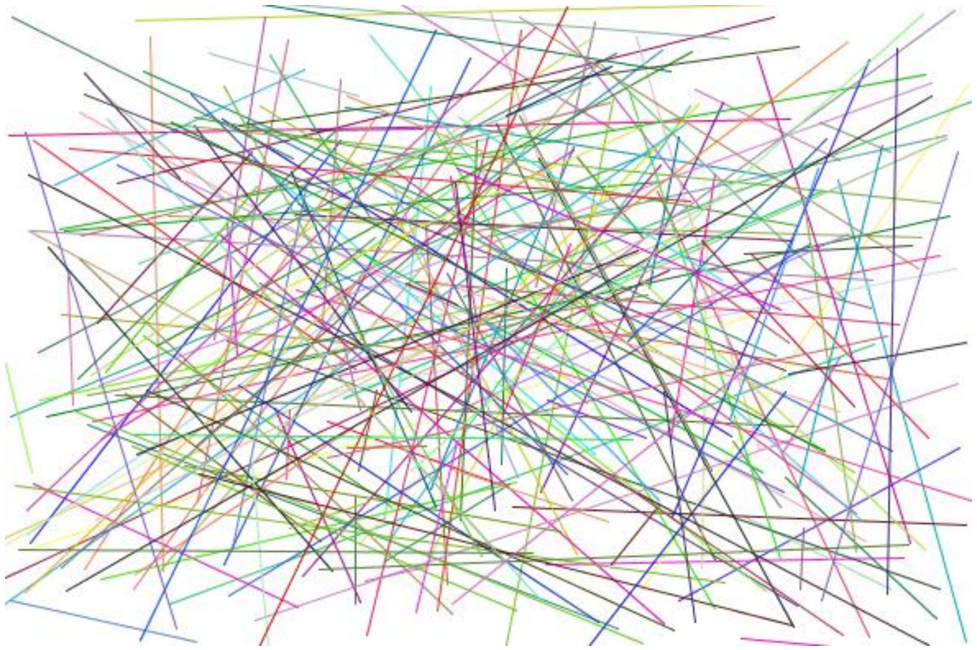
색깔 애니메이션

```
void setup() {
  size(600, 400);
  background(255, 122, 0);
  stroke(0, 0, 255);
}

void draw() {
  int c = frameCount % 256;
  fill(c, 0, 0);
  rect(10, 20, 300, 200);
}
```

1부터 시작
draw() 호출하고 1씩 증가

직선 예술

```
void setup() {
  size(600, 400);
  background(255, 255, 255);
}

void draw() {
  stroke(random(256), random(256), random(256));
  line(random(width), random(height), random(width), random(height));
}
```



- random(end);
- random(start, end);

end는 포함 안 됨

마우스

```
void setup() {
  size(600, 400);
  background(255);
  stroke(0, 0, 255);
}

void draw() {
  line(200, 200, mouseX, mouseY);
}
```

$\rightarrow$ background(255, 255, 255) 와 같음

```
void setup() {
  size(600, 400);
  stroke(0, 0, 255);
}

void draw() {
  background(255);
  line(200, 200, mouseX, mouseY);
}
```

마우스

```
void setup() {
  size(600, 400);
  stroke(0, 0, 255);
}

void draw() {
  line(200, 200, mouseX, mouseY);
}

void mousePressed() {
  background(255);
}
```

마우스

```
void setup() {
  size(600, 400);
  stroke(0, 0, 255);
}

void draw() {
  line(200, 200, mouseX, mouseY);
}

void mousePressed() {
  if(mouseButton == LEFT)
    background(255);
  else
    background(255, 122, 0);
}
```

마우스

- `mouseButton`
- `mouseClicked()`
- `mouseDragged()`
- `mouseMoved()`
- `mousePressed()`
- `mouseReleased()`
- `mouseWheel()`
- `mouseX`
- `mouseY`
- `pmouseX`
- `pmouseY`

마우스

```
void setup() {
  size(600, 400);
  background(255);
  stroke(0, 0, 255);
}

void draw() {
}

void mouseDragged() {
  line(pmouseX, pmouseY, mouseX, mouseY);
}
```

마우스

```
void setup() {
  size(600, 400);
  background(255);
  stroke(0, 0, 255);
  fill(0, 255, 0);
}

void draw() {
  background(255);
  rect(mouseX, mouseY, 30, 30);
}
```

키보드

```
void setup() {
  size(600, 400);
  background(255);
  stroke(0, 0, 255);
}

void draw() {
  line(200, 200, mouseX, mouseY);
}

void keyPressed() {
  if(key == 'a')
    background(255);
  else
    background(255, 122, 0);
}
```

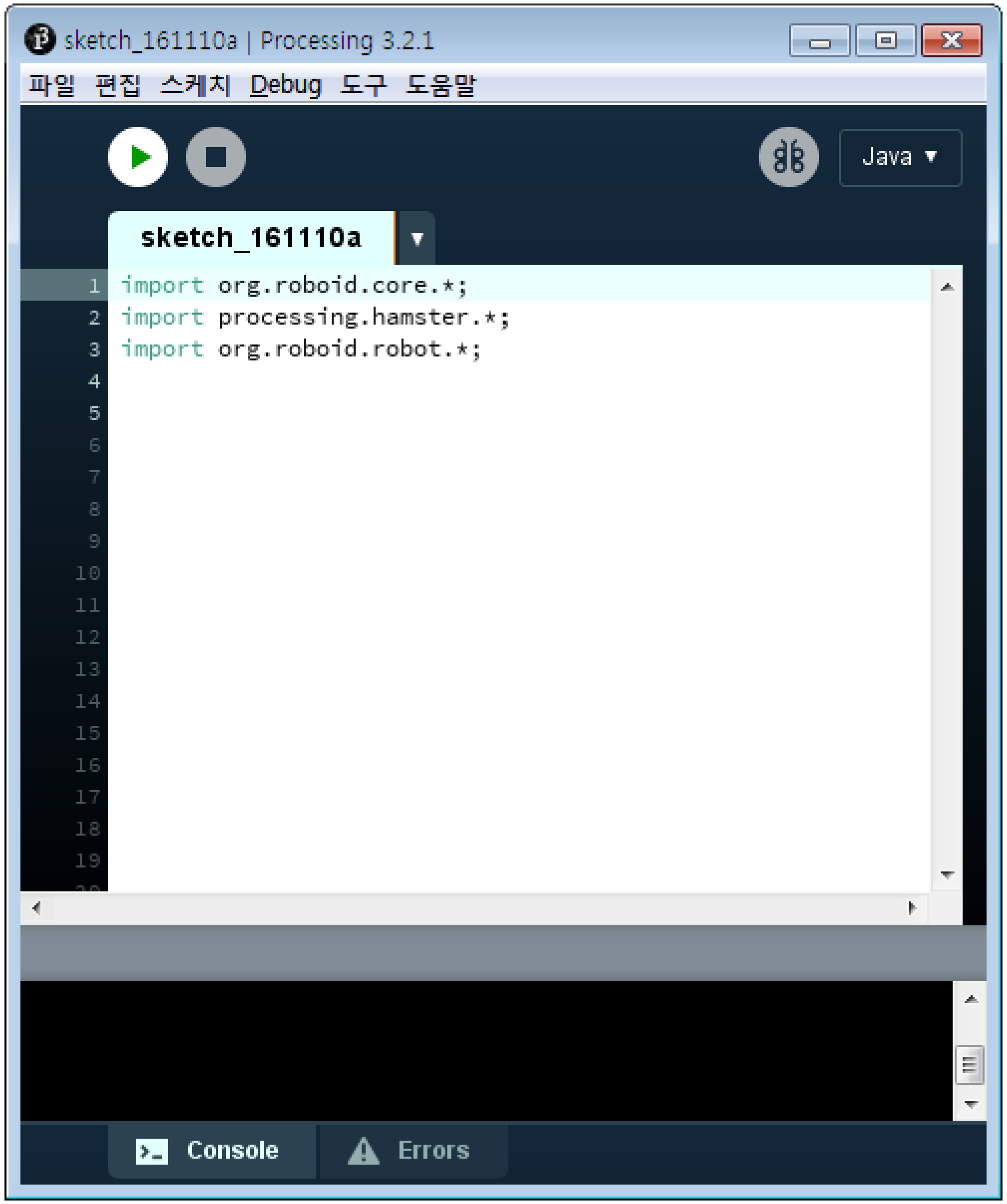
키보드

- `key`
- `keyCode`
- `keyPressed()`
- `keyPressed`
- `keyReleased()`
- `keyTyped()`

햄스터

라이브러리 사용

- 스케치
  > 내부 라이브러리...
  > Roboid


sketch_161110a | Processing 3.2.1
파일 편집 스케치 Debug 도구 도움말
Java
sketch_161110a
Console
Errors

```
import org.roboid.core.*;
import processing.hamster.*;
import org.roboid.robot.*;
```

1초 앞으로 이동하기

```
import org.roboid.core.*;
import processing.hamster.*;
import org.roboid.robot.*;

Hamster hamster;

void setup() {
  hamster = Hamster.create(this);
}

void draw() {
}

void run() {
  hamster.wheels(30, 30);
  delay(1000);
  hamster.stop();
}
```

왼쪽 바퀴 속도 (-100 ~ 100) %

오른쪽 바퀴 속도 (-100 ~ 100) %

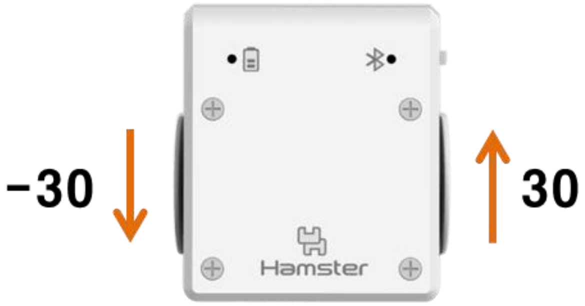
msec (1000분의 1초 단위)

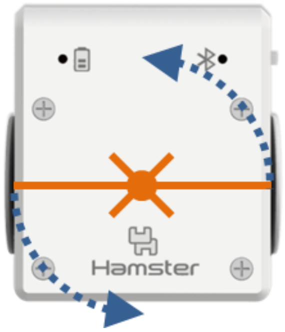
1초 뒤로 이동하기

```
import org.roboid.core.*;
import processing.hamster.*;
import org.roboid.robot.*;

Hamster hamster;

void setup() {
  hamster = Hamster.create(this);
}

void draw() {
}

void run() {
  hamster.wheels(-30, -30);
  delay(1000);
  hamster.stop();
}
```

제자리 돌기 (스핀)

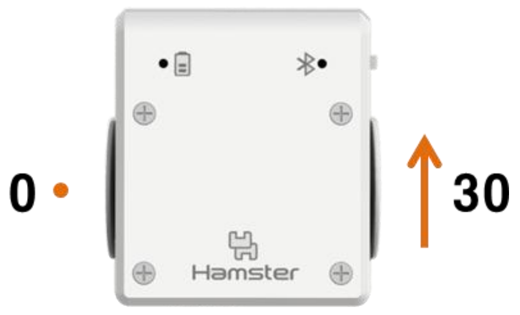
```
import org.roboid.core.*;
import processing.hamster.*;
import org.roboid.robot.*;

Hamster hamster;

void setup() {
  hamster = Hamster.create(this);
}

void draw() {
}

void run() {
  hamster.wheels(-30, 30);
  delay(1000);
  hamster.stop();
}
```

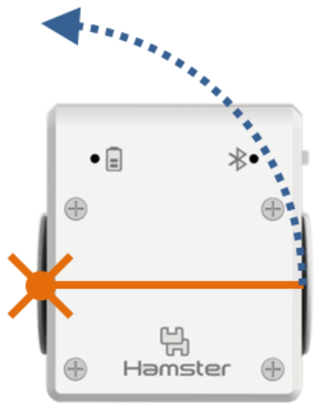




한쪽 바퀴를 축으로 회전하기 (피봇 턴)

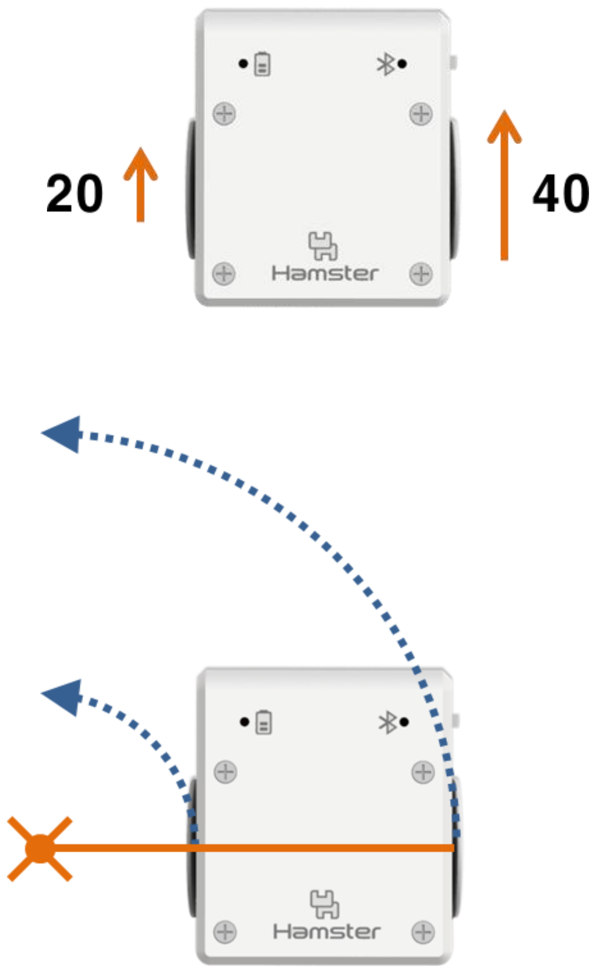
```
import org.roboid.core.*;
import processing.hamster.*;
import org.roboid.robot.*;

Hamster hamster;

void setup() {
  hamster = Hamster.create(this);
}

void draw() {
}

void run() {
  hamster.wheels(0, 30);
  delay(1000);
  hamster.stop();
}
```





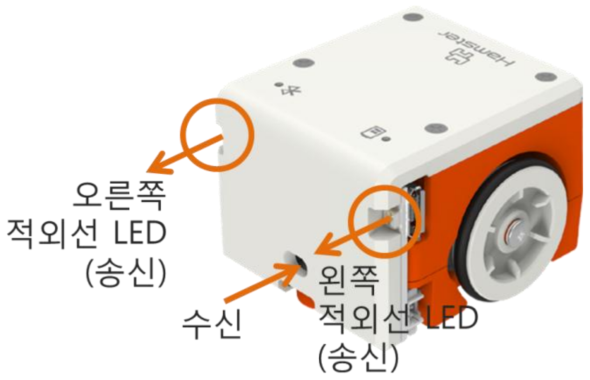
둥글게 회전하기 (라운드 턴)

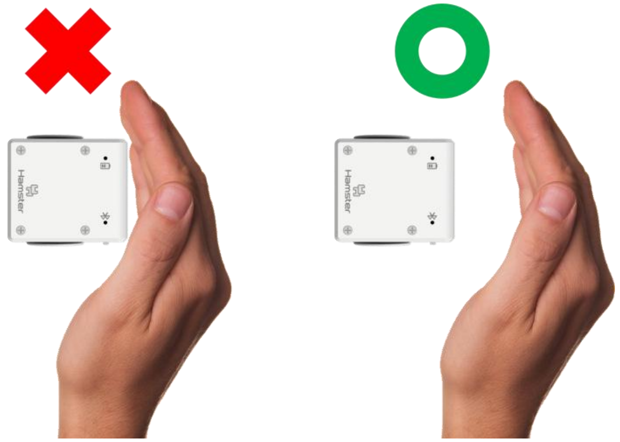
```
import org.roboid.core.*;
import processing.hamster.*;
import org.roboid.robot.*;

Hamster hamster;

void setup() {
  hamster = Hamster.create(this);
}

void draw() {
}

void run() {
  hamster.wheels(20, 40);
  delay(1000);
  hamster.stop();
}
```



근접 센서 값 관찰하기

```
import org.roboid.core.*;
import processing.hamster.*;
import org.roboid.robot.*;

Hamster hamster;

void setup() {
  hamster = Hamster.create(this);
}

void draw() {
}

void run() {
  while(true) {
    println(hamster.leftProximity());
    delay(20);
  }
}
```





손을 가져 가면 뒤로 도망가기

```
import org.roboid.core.*;
import processing.hamster.*;
import org.roboid.robot.*;

Hamster hamster;

void setup() {
  hamster = Hamster.create(this);
}

void draw() {
}

void run() {
  while(hamster.leftProximity() < 50) {
    delay(20);
  }
  hamster.wheels(-30, -30);
}
```

밀당 로봇

```
...

void run() {
  while(true) {
    if(hamster.leftProximity() > 40) {
      hamster.wheels(-30, -30);
    } else {
      hamster.wheels(30, 30);
    }
    delay(20);
  }
}
```

LED 켜고 끄기

```
...

void run() {
  while(true) {
    if(hamster.leftProximity() > 40) {
      hamster.leds(Hamster.LED_RED, Hamster.LED_RED);
      hamster.wheels(-30, -30);
    } else {
      hamster.leds(0, 0);
      hamster.wheels(30, 30);
    }
    delay(20);
  }
}
```

오른쪽 LED 색깔

왼쪽 LED 색깔

버저 소리내기

```
...

void run() {
  while(true) {
    if(hamster.leftProximity() > 40) {
      hamster.leds(Hamster.LED_RED, Hamster.LED_RED);
      hamster.buzzer(1000);
      hamster.wheels(-30, -30);
    } else {
      hamster.leds(0, 0);
      hamster.buzzer(0);
      hamster.wheels(30, 30);
    }
    delay(20);
  }
}
```

Hz

햄스터 + 그래픽

키보드 조종기

```
import org.roboid.core.*;
import processing.hamster.*;
import org.roboid.robot.*;

Hamster hamster;

void setup() {
  hamster = Hamster.create(this);
}

void draw() {
}
```

```
void keyPressed() {
  if(key == ' ')
    hamster.stop();
  else if(key == CODED) {
    if(keyCode == UP)
      hamster.wheels(30, 30);
    else if(keyCode == DOWN)
      hamster.wheels(-30, -30);
    else if(keyCode == LEFT)
      hamster.wheels(-30, 30);
    else if(keyCode == RIGHT)
      hamster.wheels(30, -30);
  }
}
```

마우스 조종기

```
import org.roboid.core.*;
import processing.hamster.*;
import org.roboid.robot.*;

Hamster hamster;

void setup() {
  size(200,200);
  hamster = Hamster.create(this);
}

void draw() {
  background(255);
  fill(0);
  ellipse(100,100, 30, 30);
  line(100 ,100, mouseX, mouseY);
}
```

```
void run() {
  while(true) {
    if(mousePressed) {
      int dx = (100 - mouseX) / 2;
      int dy = (100 - mouseY) / 2;

      if(dy < 0)
        hamster.wheels(dy + dx, dy - dx);
      else
        hamster.wheels(dy - dx, dy + dx);
    } else {
      hamster.stop();
    }
    delay(20);
  }
}
```

수고하셨습니다.

http://hamster.school

akaii@kw.ac.kr