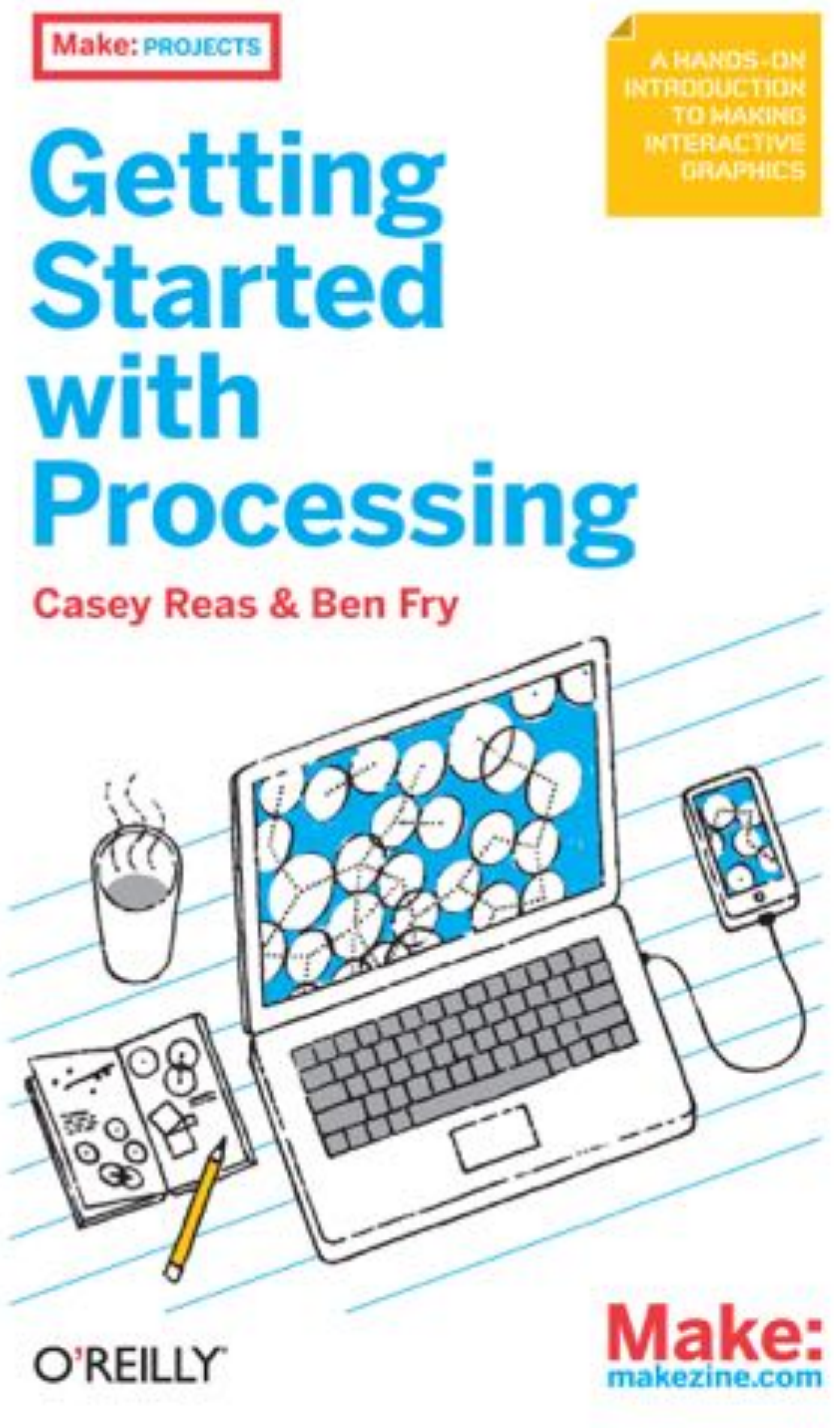
USC IML 288 __ 2018.09.24

5/Input

all code illustrations taken from the processing.org website +



Make: PROJECTS

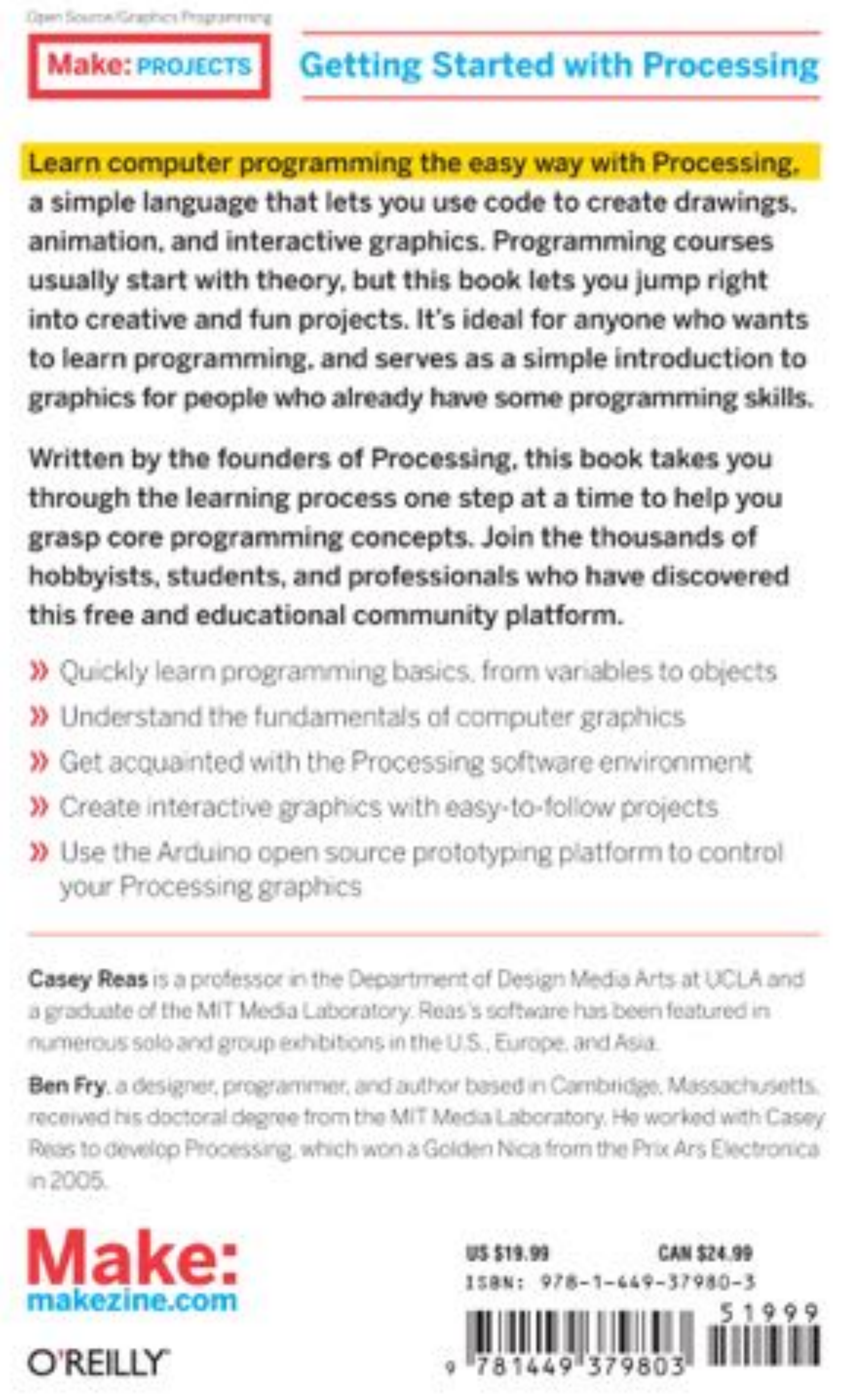
A HANDS-ON INTRODUCTION TO MAKING INTERACTIVE GRAPHICS

Getting Started with Processing

Casey Reas & Ben Fry

O'REILLY

Make: makezine.com



Open Source/Graphics Programming

Make: PROJECTS

Getting Started with Processing

Learn computer programming the easy way with Processing, a simple language that lets you use code to create drawings, animation, and interactive graphics. Programming courses usually start with theory, but this book lets you jump right into creative and fun projects. It's ideal for anyone who wants to learn programming, and serves as a simple introduction to graphics for people who already have some programming skills.

Written by the founders of Processing, this book takes you through the learning process one step at a time to help you grasp core programming concepts. Join the thousands of hobbyists, students, and professionals who have discovered this free and educational community platform.

» Quickly learn programming basics, from variables to objects
» Understand the fundamentals of computer graphics
» Get acquainted with the Processing software environment
» Create interactive graphics with easy-to-follow projects
» Use the Arduino open source prototyping platform to control your Processing graphics

Casey Reas is a professor in the Department of Design Media Arts at UCLA and a graduate of the MIT Media Laboratory. Reas's software has been featured in numerous solo and group exhibitions in the U.S., Europe, and Asia.

Ben Fry, a designer, programmer, and author based in Cambridge, Massachusetts, received his doctoral degree from the MIT Media Laboratory. He worked with Casey Reas to develop Processing, which won a Golden Nica from the Prix Ars Electronica in 2005.

Make: makezine.com

O'REILLY

US $19.99 CAN $24.99
ISBN: 978-1-449-37980-3

5/Response

Code that responds to input from the mouse, keyboard, and other devices has to run continuously. To make this happen, place the lines that update inside a Processing function called *draw()*.

```
void setup() {
  println("I’m starting");
}

void draw() {
  println("I’m running");
}
```

Draw 100 random lines every frame.

Explore the use of the Background () function in the Draw () loop.

Using *mouseX* and *mouseY*, draw an ellipse where the mouse is.

Explore the use of the Background () function in the Draw () loop.

Using *mouseX* and *mouseY*, draw an ellipse where the mouse is.

Explore the use of the Background () function in the Draw () loop.

Using *pmouseX* and *pmouseY*, change the size of the ellipse based on how quickly the cursor is moving.

easing

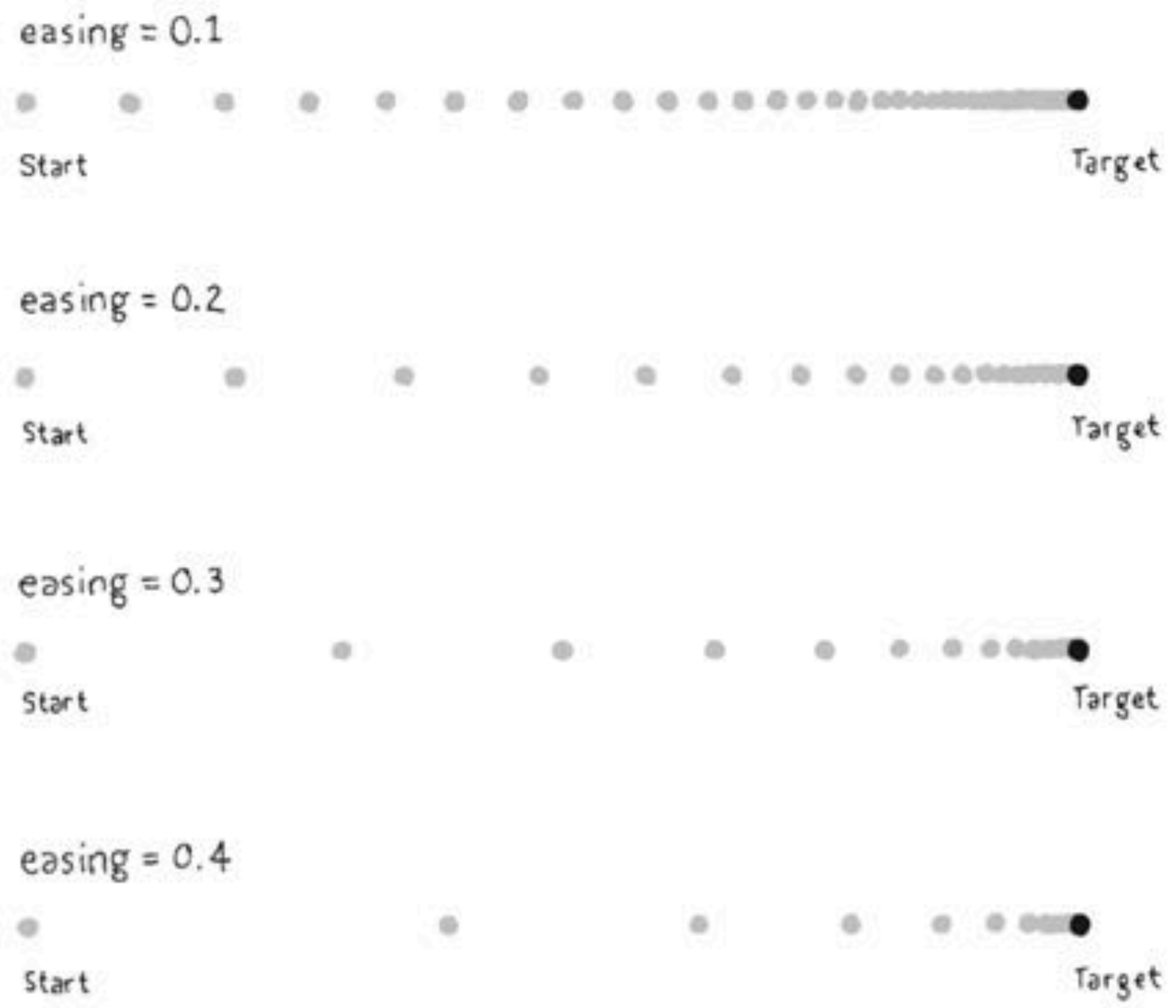


Figure 5-1. Easing.

```
float x;
float easing = 0.01;

void setup() {
  size(220, 120);
  smooth();
}

void draw() {
  float targetX = mouseX;
  x += (targetX - x) * easing;
  ellipse(x, 40, 12, 12);
  println(targetX + " : " + x);
}
```

Using *mouseX*, *mouseY*, *pmouseX* and *pmouseY*, draw a line that follows the mouse with an ease of 0.05.

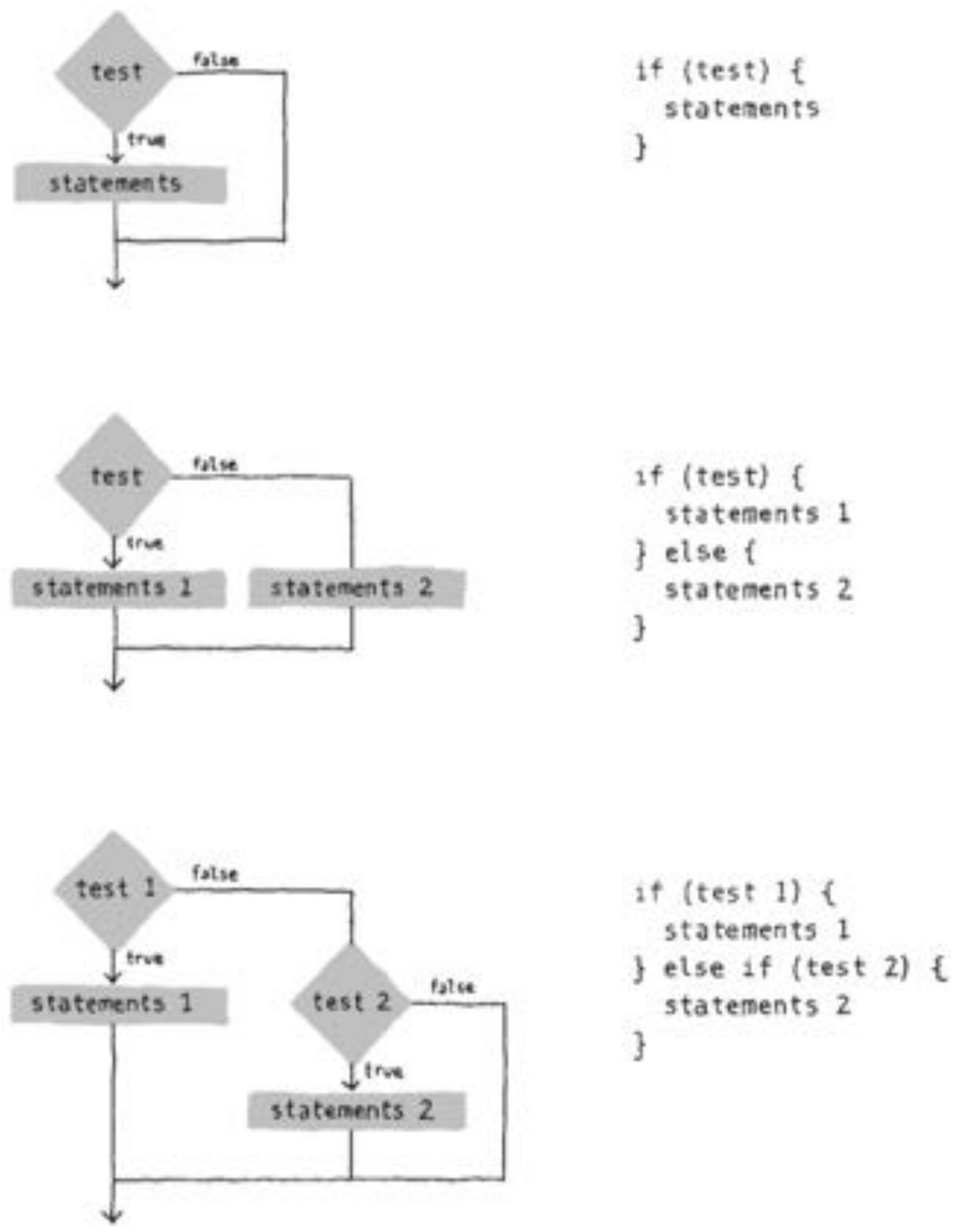
hint, calculate x and y using:

```
    x += (targetX - x) * easing;
    y += (targetY - y) * easing;
```

and store *x* and *y* somehow

Draw an ellipse that follows the mouse using *mouseX* + *mouseY.* Using the *map()* function, change the color of the ellipse based on the position of the mouse.

conditional statements



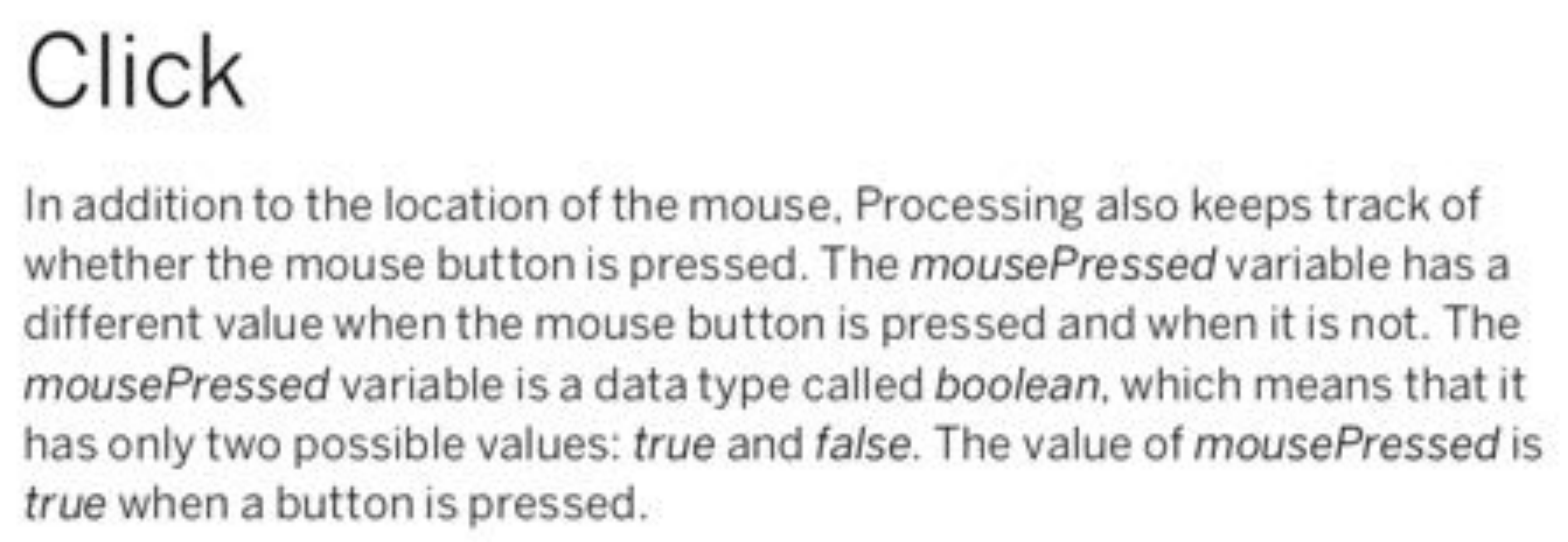
```
if (test) {
  statements
}
```

```
if (test) {
  statements 1
} else {
  statements 2
}
```

```
if (test 1) {
  statements 1
} else if (test 2) {
  statements 2
}
```

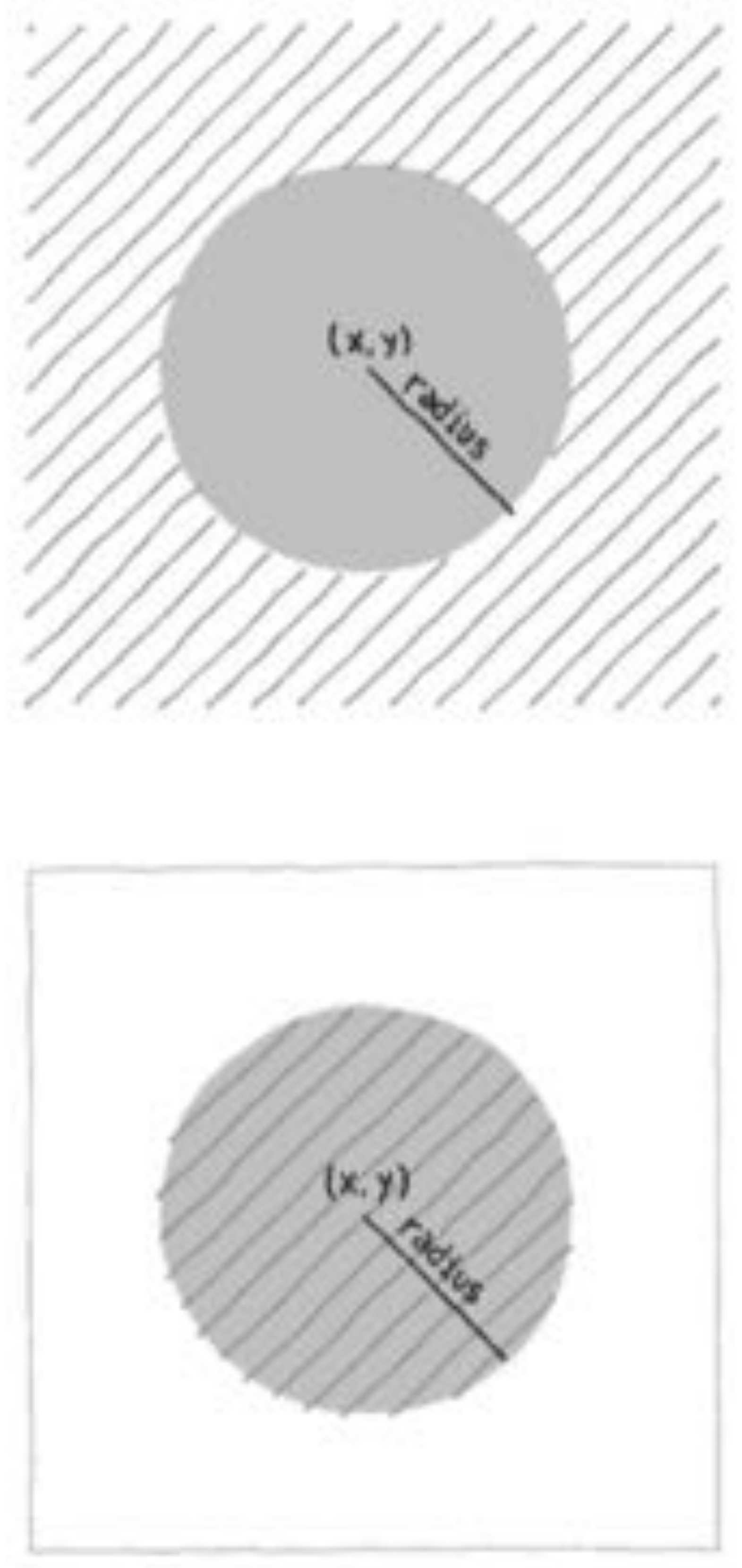
Figure 5-2. The if and else structure makes decisions about which blocks of code to run.

a simple hit check



Click

In addition to the location of the mouse, Processing also keeps track of whether the mouse button is pressed. The *mousePressed* variable has a different value when the mouse button is pressed and when it is not. The *mousePressed* variable is a data type called *boolean*, which means that it has only two possible values: *true* and *false*. The value of *mousePressed* is *true* when a button is pressed.



```
dist(x, y, mouseX, mouseY) > radius
```

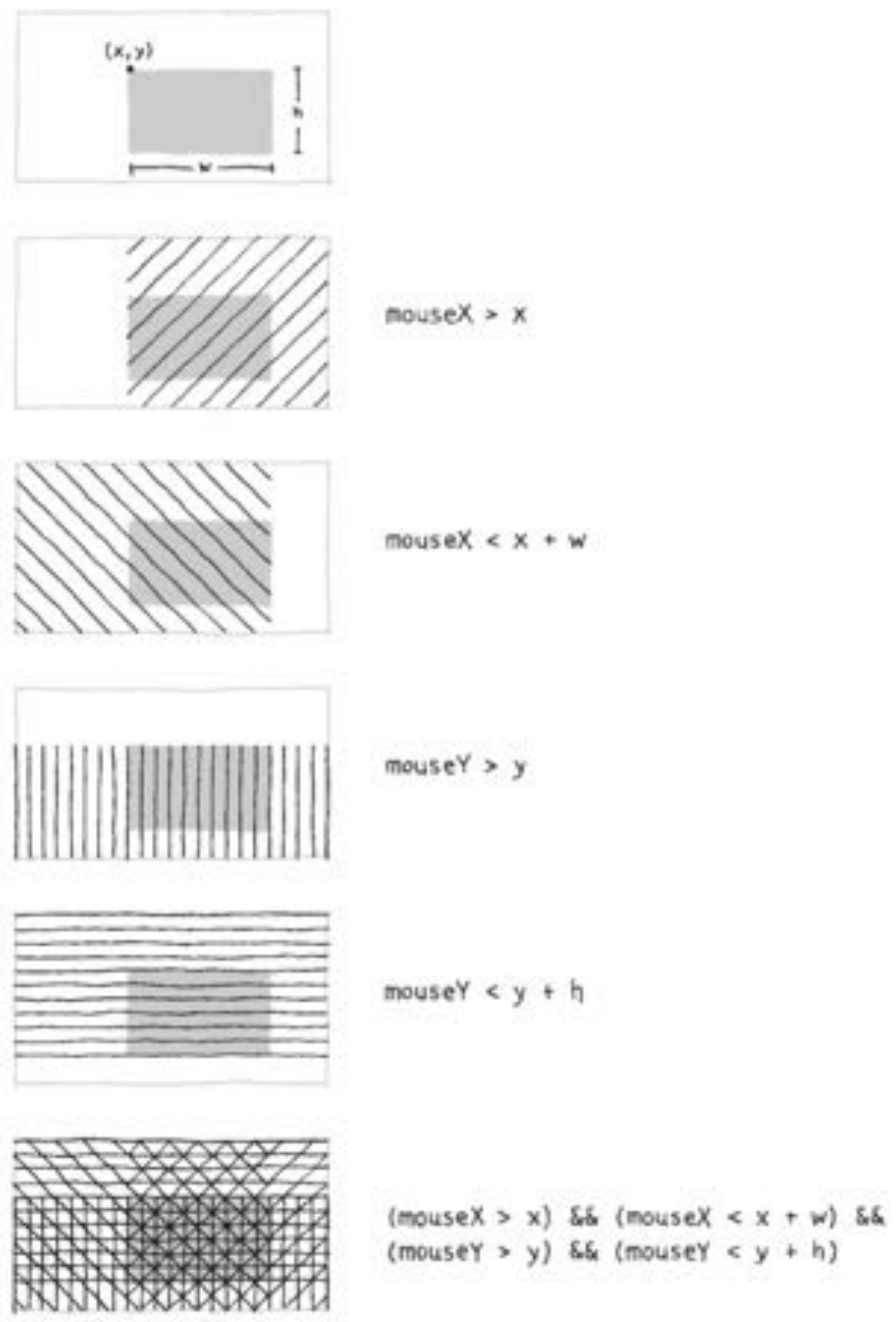
```
dist(x, y, mouseX, mouseY) < radius
```

Figure 5-3. Circle rollover test.

Draw an ellipse and change its appearance :

a) when the mouse is over it, and
b) when the mouse presses it

hit check on a rectangle



```
mouseX > x
```

```
mouseX < x + w
```

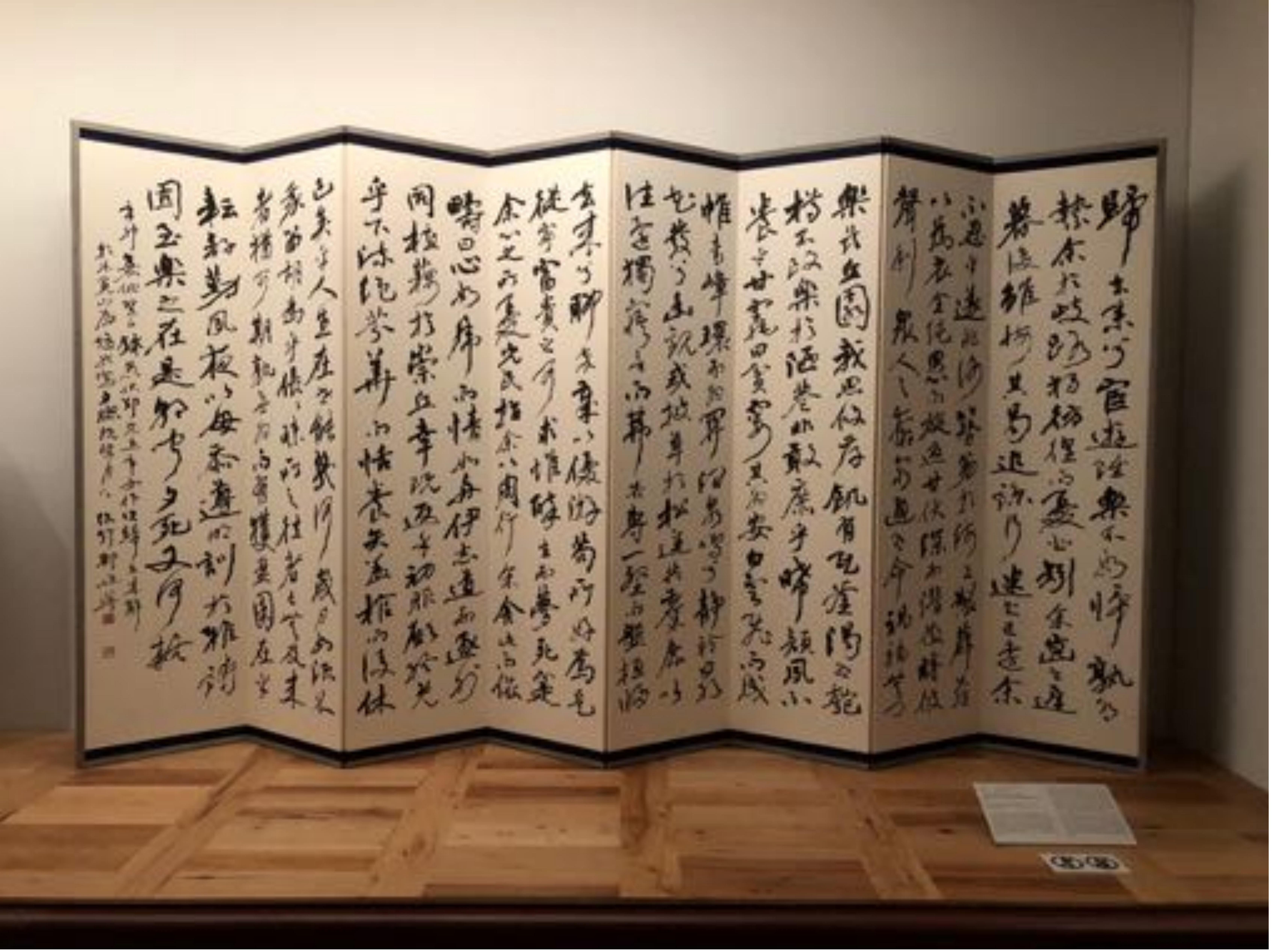
```
mouseY > y
```

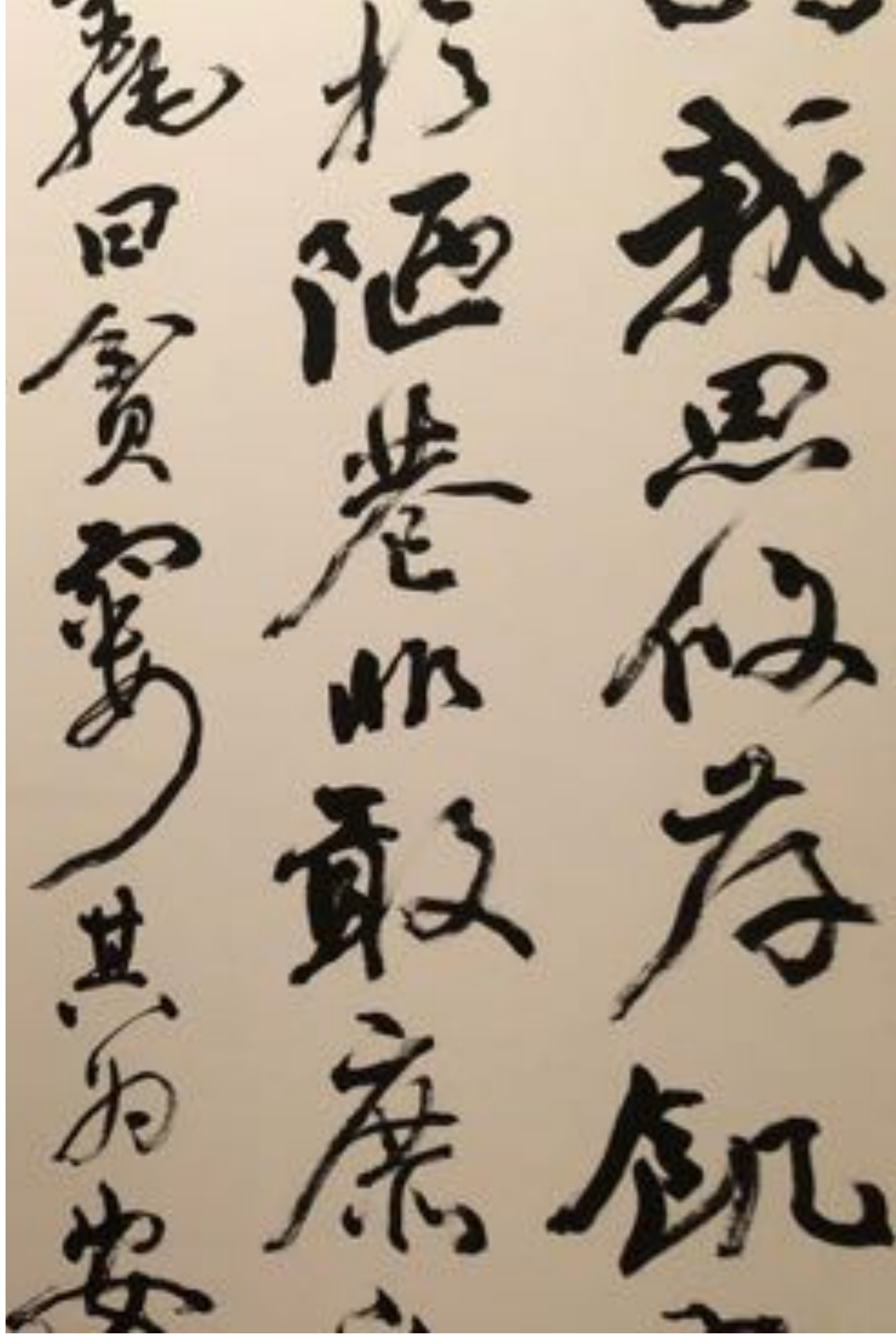
```
mouseY < y + h
```

```
(mouseX > x) && (mouseX < x + w) &&
(mouseY > y) && (mouseY < y + h)
```

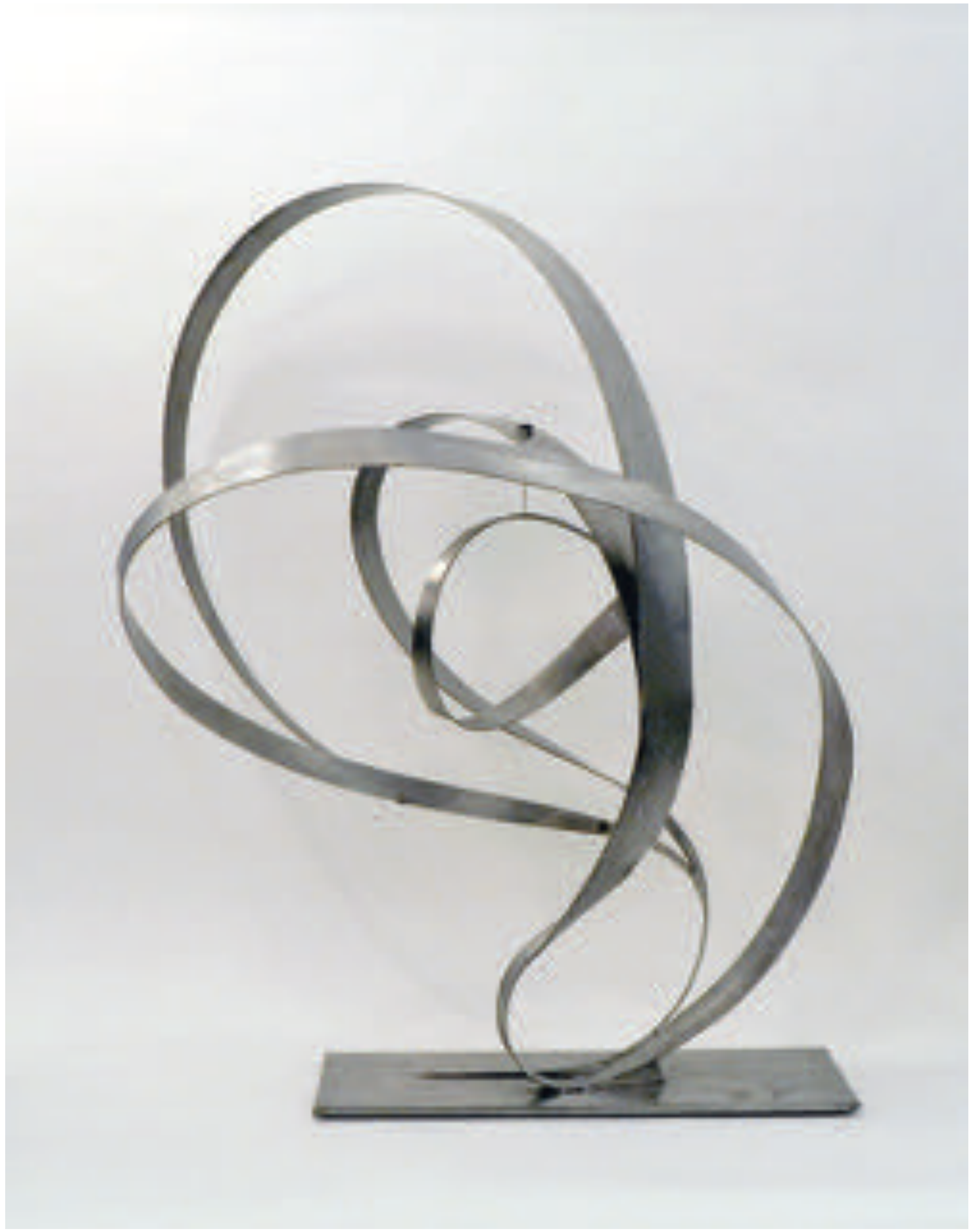
Figure 5-4. Rectangle rollover test.

Move an ellipse around on the screen using the arrow keys using a test for *key == CODED*, and *keyCode == LEFT*, *keyCode == RIGHT*, and so on…





Verse after Tao Yuanming's *Returning Home* by Jung Do-jun (2011)
http://portlandartmuseum.us/mwebcgi/mweb.exe?request=record;id=20252;type=701



Beverly Pepper, Early Sculpture with Kinetic Element (1962)
https://www.artsy.net/article/artsy-editorial-11-female-minimalists-you-should-know

Design I/O, Connectd Worlds, 2015
http://design-io.com/projects/ConnectedWorlds/

teamLab, *Graffiti Nature* at the Walker Art Museum (2017)
https://walkerart.org/press-releases/2017/virtual-ecosystems-dissolve-boundaries-betwee

Homework 04 // Due 2018.10.01

Part I. Create a drawing tool
Using some of the things we discussed in class today (*mouseX, mouseY, pmouseX, pmouseY*, etc)

Size: 1280x720 pixels

Part II. Prototype an interaction with some digital organism
Some options: have the organism follow the mouse (easing? rotate?), move the organism around with the arrow keys, change the appearance or behavior of the organism on hover

Size: appropriate to the work (see above size guidelines)