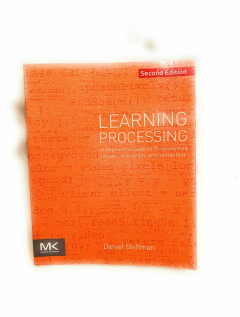

Programowanie Grafiki w Processing: Podstawowy Przewodnik dla Początkujących

Indeks: **751055** Producent: **Morgan Kaufmann Publishers** Kod producenta: **36199232**

Cena: **211.51 zł**

Opis

Learning Processing: A Beginner's Guide to Programming Images, Animation, and Interaction

Producent: Morgan Kaufmann Publishers

Zanurz się w fascynujący świat programowania grafiki i animacji z książką \

Learning Processing: A Beginner's Guide to Programming Images, Animation, and Interaction\

Znajdziesz w niej kompleksowe informacje na temat tworzenia obrazów, animacji i interakcji, które rozbudzą Twoją kreatywność i umożliwią eksplorację nowych możliwości programowania. Zapraszamy do odkrywania świata cyfrowej sztuki!

Parametry

Wydawca	Morgan Kaufmann Publishers
ISBN	978-0-123456-78-9

Zdjęcia

eclare an array of three `Zoog` *objects. Initialize each spot in the array with a Zoog from*

```
oogs = new _______[___];

[_______] = _______ _______(100, 100, 50, 60, 15);

[_______] = _______ _______(_________________);

_[_______] = _______ _______(_________________);
```

vaches are not commonly used and you will not see them in most of the exam
ecle. In fact, neither initialization method has really solved the problem posed at
hapter. Imagine initializing each element individually with a list of 1,000 or per
000,000 elements.

ill of your woes involves a means for *iterating* through the elements of the arr
ctually a loud bell is ringing in your head. Loops! (If you're lost, revise Chapter

operations

a moment, the following problem:

n array of 1,000 floating point numbers.
e every element of that array with a random number between 0 and 10.

ready know how to do.

```
[] values = new float[1000];
```

ant to avoid is having to do this for Part 2:

```
ers[0] = random(0, 10);
ers[1] = random(0, 10);
ues[2] = random(0, 10);
ues[3] = random(0, 10);
lues[4] = random(0, 10);
(lues[5] = random(0, 10);
/ etc. etc.
```

describe in English what I want to program.

every number n from 0 to 999, initialise the nth element stored in array as a random
l 10. Translating into code, I have:


```
int n = 0;
values[n] = random(0, 10);
values[n + 1] = random(0, 10);
values[n + 2] = random(0, 10);
values[n + 3] = random(0, 10);
values[n + 4] = random(0, 10);
values[n + 5] = random(0, 10);
```

Unfortunately, the situation has not improved. I have, nonetheless, taken a big leap forward. By using a variable (*n*) to describe an index in the array, I can now employ a `while` loop to initialize every *n* element.

Example 9-4. Using a `while` loop to initialize all elements of an array

```
int n = 0;
while (n < 1000) {
  values[n] = random(0, 10);
  n = n + 1;
}
```

A `for` loop allows you to be even more concise, as Example 9-5 shows.

Example 9-5. Using a `for` loop to initialize all elements of an array

```
for (int n = 0; n < 1000; n++) {
  values[n] = random(0, 10);
}
```

What was once 1,000 lines of code is now three!

I can exploit the same technique for any type of array operation I might like to do beyond simply initializing the elements. For example, I could take the array and double the value of each element. (I will use `i` from now on instead of `n` as it is more commonly used by programmers.)

Example 9-6. An array operation

```
for (int i = 0; i < 1000; i++) {
  values[i] = values[i] * 2;
}
```

There is one problem with Example 9-6: the use of the hard-coded value 1,000. Striving to be better programmers, we should always question the existence of a hard-coded number. In this case, what if you wanted to change the array to have 2,000 elements? If your program was very long with many array operations, you would have to make this change everywhere throughout your code. Fortunately, JavaScript offers a nice means for accessing the size of an array dynamically, using the dot syntax you learned to use with objects in Chapter 8. `length` is a property of every array and you can access it by saying

`arrayName.length`.