A/Coding Tips

Coding is a type of writing. Like all types of writing, code has specific rules. For comparison, we'll quickly mention some of the rules for English that you probably haven't thought about in a while because they are second nature. Some of the more invisible rules are writing from left to right and putting a space between each word. More overt rules are spelling conventions, capitalizing the names of people and places, and using punctuation at the end of sentences to provide emphasis! If we break one or more of these rules when writing an email to a friend, the message still gets through. For example, "hello ben. how r u today" communicates nearly as well as, "Hello, Ben. How are you today?" However, flexibility with the rules of writing don't transfer to programming. Because you're writing to communicate with a computer, rather than another person, you need to be more precise and careful. One misplaced character is often the difference between a program that runs and one that doesn't.

Processing tries to tell you where you've made mistakes and to guess what the mistake is. When you click the Run button, if there are grammar (syntax) problems with your code (we call them *bugs*), then the Message Area turns red and Processing tries to highlight the line of code that it suspects as the problem. The line of code with the bug is often one line above or below the highlighted line, though in some cases, it's nowhere close. The text in the Message Area tries to be helpful and suggests the potential problem, but sometimes the message is too cryptic to understand. For a beginner, these error messages can be frustrating. Understand that Processing is a simple piece of software that's trying to be helpful, but it has a limited knowledge of what you're trying to do.

Long error messages are printed to the Console in more detail, and sometimes scrolling through that text can offer a hint. Additionally, Processing can find only one bug at a time. If your

program has many bugs, you'll need to keep running the program and fix them one at a time.

Please read and reread the following suggestions carefully to help you write clean code.

Functions and Parameters

Programs are composed of many small parts, which are grouped together to make larger structures. We have a similar system in English: words are grouped into phrases, which are combined to make sentences, which are combined to create paragraphs. The idea is the same in code, but the small parts have different names and behave differently. *Functions* and *parameters* are two important parts. Functions are the basic building blocks of a Processing program. Parameters are values that define how the function behaves.

Consider a function like `background()`. Like the name suggests, it's used to set the background color of the Display Window. The function has three parameters that define the color. These numbers define the red, green, and blue components of the color to define the composite color. For example, the following code draws a blue background:

```
background(51, 102, 153);
```

Look carefully at this single line of code. The key details are the parentheses after the function name that enclose the numbers, the commas between each number, and the semicolon at the end of the line. The semicolon is used like a period. It signifies that one statement is over so the computer can look for the start of the next. All of these parts need to be there for the code to run. Compare the preceding example line to these three broken versions of the same line:

```
background 51, 102, 153; // Error! Missing the parentheses
background(51 102, 153); // Error! Missing a comma
background(51, 102, 153) // Error! Missing the semicolon
```

The computer is very unforgiving about even the smallest omission or deviation from what it's expecting. If you remember these parts, you'll have fewer bugs. But if you forget to type them, which we all do, it's not a problem. Processing will alert

you about the problem, and when it's fixed, the program will run well.

Color Coding

The Processing environment color-codes different parts of each program. Words that are a part of Processing are drawn as blue and orange to distinguish them from the parts of the program that you invent. The words that are unique to your program, such as your variable and function names, are drawn in black. Basic symbols such as `()`, `\[\]`, and `>` are also black.

Comments

Comments are notes that you write to yourself (or other people) inside the code. You should use them to clarify what the code is doing in plain language and provide additional information such as the title and author of the program. A comment starts with two forward slashes (`//`) and continues until the end of the line:

```
// This is a one-line comment
```

You can make a multiple-line comment by starting with `/*` and ending with `*/`. For instance:

```
/* This comment
   continues for more
   than one line
*/
```

When a comment is correctly typed, the color of the text will turn gray. The entire commented area turns gray so you can clearly see where it begins and ends.

Uppercase and Lowercase

Processing distinguishes uppercase letters from lowercase letters and therefore reads "Hello" as a distinct word from "hello". If you're trying to draw a rectangle with the `rect()` function and you write `rect()`, the code won't run. You can see if Processing recognizes your intended code by checking the color of the text.

Style

Processing is flexible about how much space is used to format your code. Processing doesn't care if you write:

```
rect(50, 20, 30, 40);
```

or:

```
rect (50,20,30,40);
```

or:

```
rect     (       50,20,
 30,   40)            ;
```

However, it's in your best interest to make the code easy to read. This becomes especially important as the code grows in length. Clean formatting makes the structure of the code immediately legible, and sloppy formatting often obscures problems. Get into the habit of writing clean code. There are many different ways to format the code well, and the way you choose to space things is a personal preference.

Console

The Console is the bottom area of the Processing Development Environment. You can write messages to the Console with the `println()` function. For example, the following code prints a message followed by the current time:

```
println("Hello, Processing.");
println("The time is " + hour() + ":" + minute());
```

The Console is essential to seeing what is happening inside of your programs while they run. It's used to print the value of variables so you can track them, to confirm if events are happening, and to determine where a program is having a problem.

One Step at a Time

We recommend writing a few lines of code at a time and running the code frequently to make sure that bugs don't accumulate without your knowledge. Every ambitious program is written one line at a time. Break your project into simpler subprojects and complete them one at a time so that you can have many

small successes, rather than a swarm of bugs. If you have a bug, try to isolate the area of the code where you think the problem lies. Try to think of fixing bugs as solving a mystery or puzzle. If you get stuck or frustrated, take a break to clear your head or ask a friend for help. Sometimes, the answer is right under your nose but requires a second opinion to make it clear.