付録A

コーディングの心得
Coding Tips

プログラムを書くことも、手紙を書くことも、書くという意味では同じですが、コンピュータを相手にするときはルールがより厳密に適用されます。友達に送るメールなら“Hello Ben. How are you today?”と書くかわりに、“hello ben. how r u today”と書くことも許されるでしょう。しかし、融通の効かないコンピュータに読ませるプログラムは1文字でも間違えば動きません。

Processingはプログラマーがどこでどんなミスをしたかを告げようとします。コードに何らかの間違い（バグ）があると、Runボタンを押した直後、メッセージエリアが赤くなって、Processingが疑わしいと判断した行がハイライト表示されます。ただし、その行にバグがあるとは限りません。1行前の間違いが原因ということはよくありますし、ときには、ひどく離れた行にバグが潜んでいることもあります。メッセージエリアには問題点を説明し、解決策を示す文章（エラーメッセージ）が表示されますが、それも暗号のようにわかりにくいときがあります。初心者にとって、こうしたエラーメッセージは間違いなくイライラのもとでしょう。Processingはバグの発生時に有益な情報を提供しようとしますが、プログラマーがやろうとしていることに関しては極めて限られた知識しか持っていないことを理解してください。

Processingが一度に見つけることのできるバグは1個だけです。たくさんのバグがあるときは、プログラムの実行と修正を何度も繰り返す必要があります。1つのバグに関する長いエラーメッセージがコンソールに表示されたときは、スクロールして最後まで目を通すとヒントが見つかるかもしれません。

ここからは、確実に動くコードを書くために覚えておいてほしいことを整理していきます。

カッコ、カンマ、セミコロン

プログラムはたくさんの小さなパーツでできています。小さなものが集まって大きな構造が生み出されます。パーツのなかでもっとも重要なのが関数とパラメータです。関数はProcessingの基本的な構成要素であり、パラメータは関数のふるまいを決定づける値のことです。

background()のような関数を考えてみましょう。その名前から、ウィンドウの背景（background）の色を変えるために使われることが推測できます。この関数は色を定義する3つのパラメータを持っていて、赤、緑、青の3色を混ぜ合わせることで、ひとつの色を作り出します。たとえば、青い背景を描きたいときは次のようにします。


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

この1行のコードをよく見て、大事なことを確認してください。まず、関数名の後ろのカッコの中に数字が全部入っていますね。それぞれの数字はカンマで区切られています。行の末尾にはセミコロンがあります。コンピュータはこのセミコロンを見て、文の終わりを判断します。カッコ、カンマ、セミコロンといった部品はひとつでも欠けるとプログラムは動作しません。次の3行を、前の1行と比べてみましょう。

```
background 51, 102, 153;     // エラー! カッコがひとつありません
background(51 102, 153);     // エラー! カンマがひとつありません
background(51, 102, 153)     // エラー! セミコロンが見つかりません
```

コンピュータはたった1文字の誤字や脱字にも容赦ありませんが、カッコ、カンマ、セミコロンの3つを忘れないようにするだけでもバグはだいぶ減らせます。エラーが発生したときは、この3つを忘れていないか確認しましょう。間違いが見つかったときは、それを修正してからまた実行します。問題が消えれば、プログラムは動きだします。

コードの色分け

Processingの開発環境はプログラムを部分ごとに色分けして表示します。青かオレンジになっている部分はProcessingに組み込まれている単語で、プログラマーが自分でつけた名前と区別できるようになっています。プログラマーがつけた関数名や変数名は黒で表示されます。()や[]、>といった記号も黒です。

コメント

コメントはコードの中に書くメモです。自分のための覚え書きであり、そのコードを読む他者に向けての説明書でもあります。そのプログラムが何をしているのかを、普通の言葉で明解に示しましょう。コメントはプログラムのタイトルや作者に関する情報を記録するためにも使われます。

コメントの書き方は2通りあります。ひとつがスラッシュ（//）を2個書く方法で、そこから行末までがコメントになります。

```
// これは1行コメントです
```

複数の行に渡るコメントも書けます。/*と*/で挟まれた部分がコメントとなります。

```
/* 複数の行にわたる
    コメントの例です
*/
```

正しくコメントが入力されるとテキストがグレーに変わり、どこからどこまでがコメントなのかが分かりやすく表示されます。

大文字、小文字

Processingは大文字と小文字を別の文字として認識します。つまり、"Hello"と"hello"は別の単語です。もし、コーディングの途中で、rect()と書くべきところをRect()と書いてしまうと正常に動作しません。入力したコードをProcessingが正しく認識しているかどうかは、文字の色からも判断できます。

スタイル

Processingはコードの中のスペースの数については柔軟です。下記のコードはどれも同じ意味に解釈されます。

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

rect (50,20,30,40);

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

読みやすいコードを書くよう心がけましょう。コードが長くなってくると、それがとくに重要です。適切な空白のあるコードは読みやすいものです。逆に、スペースの設け方が雑なコードは読みにくく問題が埋もれがちです。すっきりしたコードを書く習慣を持ちましょう。

コンソール

Processing開発環境の一番下のエリアがコンソールです。println()関数を使って、そこにメッセージを出力することができます。例をあげましょう。次のコードは挨拶と現在時刻を表示します。

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

実行中のプログラムの内部で起きていることを知るためにコンソールは不可欠です。変数の値を出力することで、動作状態を確認したり、問題発生時の原因を探ることができます。

一歩ずつ

プロジェクトを細かいサブプロジェクトに分割し、それらをひとつひとつ片付けていくようにすれば、小さな成功を何度も味わえます。バグが知らないうちに溜まってしまうことのないよう、コードは数行ずつ書き、頻繁に実行して正常に動くことを確かめながら進めていきましょう。もしバグが出てしまったら、コードを切り分けて、そのバグが潜んでいる場所を特定し解決します。とはいっても、バグ取りはパズルのようなもので、ときにはイライラしたり手詰まりになってしまうこともあります。そういうときは、休憩を取って頭をスッキリさせましょう。友人に助言を求めるのもいい手です。セカンドオピニオンによって、問題がクリアになることがあります。