はじめに
Preface

私たちがProcessingを作った理由は、インタラクティブグラフィックスのプログラミングをもっとやさしくするためです。かつて使っていたC++やJavaといった言語でこの種のプログラムを開発することはとても大変で、フラストレーションのもとでした。子どものころはBASICやLogoで気軽に面白いプログラムを組んだことを思い出したものです。Processingはこの2つの言語からインスパイアされています。ただし、もっとも強い影響を受けた言語は別にあって、それは私たちの恩師でもあるJohn Maedaが開発し、私たちが保守し教えていた言語、Design By Numbers（DBN）です。

1枚の紙を使って行ったブレインストーミングの結果、Processingは誕生しました。2001年春のことです。当時取り組んでいたソフトウェアをスケッチ（試作）する道具を作ることが目標でした。私たちが作りたかったソフトウェアは画面全体を使うインタラクティブなものだったので、コードを書いて構想を検証しようにも、C++では時間がかかりすぎて現実的ではなかったのです。口先の議論だけで我慢するか、もっと良い方法を見つける必要がありました。もうひとつの目的は、学生たちにとって使いやすい言語を開発することでした。アートやデザインの学生にプログラムの作り方を教える際の教材として、あるいはもう少し技術的なクラスで学生がグラフィックスを扱うときの道具として使える言語を作るため、それまでのプログラミングの教え方から離れて、グラフィックスとインタラクションに焦点を絞って開発を始めました。

Processingは長い幼年期を経験しました。アルファ版の期間は2002年8月から2005年4月まで続き、その後2008年11月まではパブリックベータ版だったのです。この間、世界中のたくさんのユーザーに使用され、言語仕様、開発環境、そして教授法が継続的に見直されました。私たち開発者が下した決定の多くが強化され、また多くが変更されました。ライブラリと呼ばれるソフトウェア拡張機能が加えられると、当初は想定していなかった驚くような用途に向けてProcessingが成長を始めました。現在、100以上のライブラリが存在します。7年間の開発を経て、安定性に重点を置いたバージョン1.0が公開されたのは2008年11月29日のことです。

始まりから9年が経過し、成長したProcessingは当初の目的を果たしました。そして、教育以外の文脈でも、その有用性が知られるようになりました。そうした状況に対応するため、この本は新たな読者に向けて書かれています。カジュアルプログラマー、ホビースト、分厚い教科書を読まずにProcessingのおいしいところを体験したい人も歓迎します。プログラミングを楽しんで、これからもずっと続けたいと思うくらい刺激を受けてください。この本は出発点に過ぎません。

私たち（CaseyとBen）は9年間、Processingという船を進めてきましたが、このプロジェクトはコミュニティの成果であることを強調しておきます。ソフトウェアを拡張するためにライブラリを書いたり、オンラインでコードを配布して他者の学習を助けたりといった、コミュニティの人々によるさまざまな貢献が、当初のコンセプトを越える領域にまでProcessingを推し進めました。この努力なくして、今日のProcessingはありえません。

本書の構成

本書の各章は次のような構成になっています。

- 1章「ようこそProcessingへ」 Processingの基礎について学びます。
- 2章「コードを書いてみよう」 最初のProcessingプログラムを作ります。
- 3章「描く」 基本的な図形を描きます。
- 4章「変数」 データを記憶し、変更し、再利用します。
- 5章「反応」 マウスとキーボードに反応するプログラムを作ります。
- 6章「メディア」 画像、フォント、ベクタ画像などを読みこんで表示します。
- 7章「動き」 図形を意図したとおりに動かします。
- 8章「関数」 新たなコードモジュールを作ります。
- 9章「オブジェクト」 変数と関数が結合したコードモジュールを作ります。
- 10章「配列」 変数のリストを扱いやすくします。
- 11章「拡張」 3次元画像、画像の書き出し、Arduinoボードからの読み込み。

対象読者

この本は気軽に読める簡潔なコンピュータプログラミングの入門書を求める人のために書かれています。とくにグラフィックスやインタラクションに興味がある人にとって有用でしょう。Processingの膨大なオンラインリファレンスと数千に及ぶサンプルコードを理解するためのとっかかりを必要としている人も、この本から始めることをおすすめします。

『Processingをはじめよう』は教科書ではありません。タイトルが示すように、プログラミングを始めるための本であり、中高生、ホビースト、おじいさん、おばあさんを含むあらゆる人を対象にしています。

この本はインタラクティブなコンピュータグラフィックスの基礎を学びたいプログラミング経験者にも適しています。ゲーム、アニメーション、ユーザーインタフェイスの開発に応用可能なテクニックが見つかるでしょう。

本書の表記について

本書では、以下の表記を使用しています。

» 等幅文字（Constant Width）：プログラムリストを表します。

[NOTE] このタイプの項目は一般的な注意事項を表しています。

サンプルコードの使用について

本書の目的は、読者の仕事の手助けをすることです。一般に、本書に掲載しているコードは各自のプログラムやドキュメントに使用してかまいません。コードの大部分を転載する場合を除き、私たちに許可を求める必要はありません。たとえば、本書のコードブロックをいくつか使用するプログラムを作成するために、許可を求める必要はありません。なお、O'Reillyから出版されている書籍のサンプルコードをCD-ROMとして販売したり配布したりする場合には、そのための許可が必要です。本書や本書のサンプルコードを引用して問題に答える場合、許可を求める必要はありません。ただし、本書のサンプルコードのかなりの部分を製品マニュアルに転載するような場合には、そのための許可が必要です。

作者の帰属を明記する必要はありませんが、そうしていただければ感謝します。帰属を明記する際には、Casey Reas、Ben Fry著『Processingをはじめよう』（オライリー・ジャパン）のように、タイトル、著者、出版社、ISBNなどを盛り込んでください。サンプルコードの使用について、正規の使用の枠を超える、またはここで許可している範囲を超えると感じる場合は、permissions@oreilly.comまでご連絡ください。

意見をお聞かせください。

この本に関するコメントや質問は、出版社までお願いします。

株式会社オライリー・ジャパン
〒160-0002　東京都新宿区坂町26番地27　インテリジェントプラザビル1F
電話：03-3356-5227　FAX：03-3356-5261

この本のウェブサイトには、正誤表などの追加情報が掲載されています。URLは以下のとおりです。

http://oreilly.com/catalog/0636920000570
http://www.oreilly.co.jp/books/9784873115153

この本に関するコメントや質問を電子メールで送るには、以下のアドレスへお願いします。

japan@oreilly.co.jp
bookquestions@oreilly.com（英文）

謝辞

Brian Jepsonの情熱と助力と洞察に感謝します。Nancy Kotary、Rachel Monaghan、Sumita Mukherjiが本書を完成に導いてくれました。

Tom Sgourosが本書を完璧に編集し、David Humphreyは的確なテクニカルレビューを提供してくれました。

この本の原型はMassimo Banziの素晴らしい著書『Getting Started with Arduino』（O'Reilly、日本語訳『Arduinoをはじめよう』オライリー・ジャパン）です。彼の本がなければ、本書もなかったでしょう。

何年にも渡ってProcessingのために時間とエネルギーを割いて貢献した人たちがいます。ウェブハッキングとデザインの能力を発揮してくれたFlorian Jenett、ライブラリ構築の基盤を作ってくれたAndreas Schlegel、そして素晴らしいサンプルを書きチュートリアルを整理してくれたDan Shiffmanに感謝の意を表します。Karsten Schmidt、Eric Jordan、Jonathan FeinbergはProcessingソフトウェアに対して長い間貢献してくれました。フォーラムを通じた議論が機能し続けるために、PhiLho、Cedric、antiplastikといった管理者たちの働きが重要でした。

私たちは、ライブラリを作成し、作品をコミュニティに提供してくれた人々の仕事ぶりに驚嘆しました。全員にありがとう！　なかでも、Andres ColubriのGLGraphicsとGSVideoライブラリ、Damien Di FedeのMinimライブラリ、Karsten Schmidtの膨大なtoxiclibsは特筆に値します。

Processing 1.0のリリースにあたっては、マイアミ大学とOblong Industriesからのサポートを得ています。マイアミ大学のArmstrong Institute for Interactive Media StudiesはOxford Project（Processing開発に関する一連のワークショップ）に対して資金を提供しました。このワークショップはIra Greenbergの尽力によって実現したものです。2008年11月、オハイオ州オクスフォードとペンシルバニア州ピッツバーグで開催された4日間のミーティングの場でProcessing 1.0は公開されました。Oblong Industriesは、2008年の夏の間、Ben FryがProcessing開発を行うための資金を提供しましたが、これはバージョン1.0のリリースにとって決定的に重要でした。

本書はUCLAで行った講義をもとにしています。その内容を決定するにあたってはChandler McWilliamsの協力がありました。CaseyはUCLA Design Media Artsの学部学生が発揮してくれた情熱と熱意に感謝しています。このときのCaseyのアシスタントたちは、Processingの教育方法を定める上で大きな貢献を果たしました。Tatsuya Saito、John Houck、Tyler Adams、Aaron Siegel、Casey Alt、Andres Colubri、Michael Kontopoulos、David Elliot、Christo Allegra、Pete Hawkes、そしてLauren McCarthyに敬意を表します。

OpenProcessingはオープンソースなProcessingのコードを共有する場として登場しました。コミュニティに素晴らしいリソースを提供してくれているSinan Asciogluに感謝します。

Processing.jsはProcessingとオープンウェブの未来形です。John Resig、Al MacDonald、David Humphrey、Seneca College's Centre for Development of Open Technology (CDOT)、Robert O'Rourke、そしてMozilla Foundationに拍手を。

John MaedaはMITのAesthetics and Computation Group (1996-2002) の設立によって、これらすべてのことを可能にしてくれました。

1

ようこそProcessingへ

Hello

Processingはイメージ、アニメーション、そしてインタラクションを生み出すソフトウェアを書くためにあります。画面に円を描きたいとき、Processingならコードを1行書くだけです。そのコードに何行か付け加えると円はマウスカーソルを追って動き出し、さらにもう1行追加するとボタンを押すたびに色が変わるでしょう。このように1行、また1行とコードを書き加えながらプログラムを作っていくことを、我々は「スケッチング」と呼んでいます。

典型的なプログラミングの授業では、はじめに理論と構造を習います。アニメーションやユーザーインタフェイスといった視覚的な要素はいつも食後のデザート扱いで、数週間に渡ってアルゴリズムを勉強した後にだけ登場します。何年もの間、私たちが目にしたのは、そうした授業から脱落していった友人たちの姿です。ある人は最初の授業のあと、またある人は最初の課題の締切前夜に、挫折感とともに辞めていきました。コンピュータを使って作りたいものと、そのために学ばなくてはいけないことのギャップが大きすぎて、学び始めのころは持っていた好奇心を失ってしまったのです。

Processingが提案するのは、インタラクティブな映像の創作を通じてプログラミングを学ぶ方法です。視覚的なフィードバックがすぐに返ってくれば、それが刺激となり、やる気を保ちながら課題に取り組むことができるでしょう。

この後の数ページを使って、フィードバックを重視するProcessingがプログラミングの学習に向いている理由と、イメージ、スケッチング、コミュニティといった概念の重要性について議論します。

Sketching and Prototyping

スケッチングとプロトタイピング

速く楽しく考えるための手段がスケッチングです。短時間に多くのアイデアを試すことが重要で、新しい仕事に取りかかるときは、まず紙にスケッチを描き、それをコードへ移していきます。アニメーションやインタラクションのアイデアは絵コンテのようなスケッチになるでしょう。ソフトウェアによるスケッチをいくつか作ったあと、最良のアイデアを選び出し、それらを組み合わせてプロトタイプにまとめます（図1-1）。紙と画面の間を行き来しながら、作り、テストし、改良することを繰り返します。

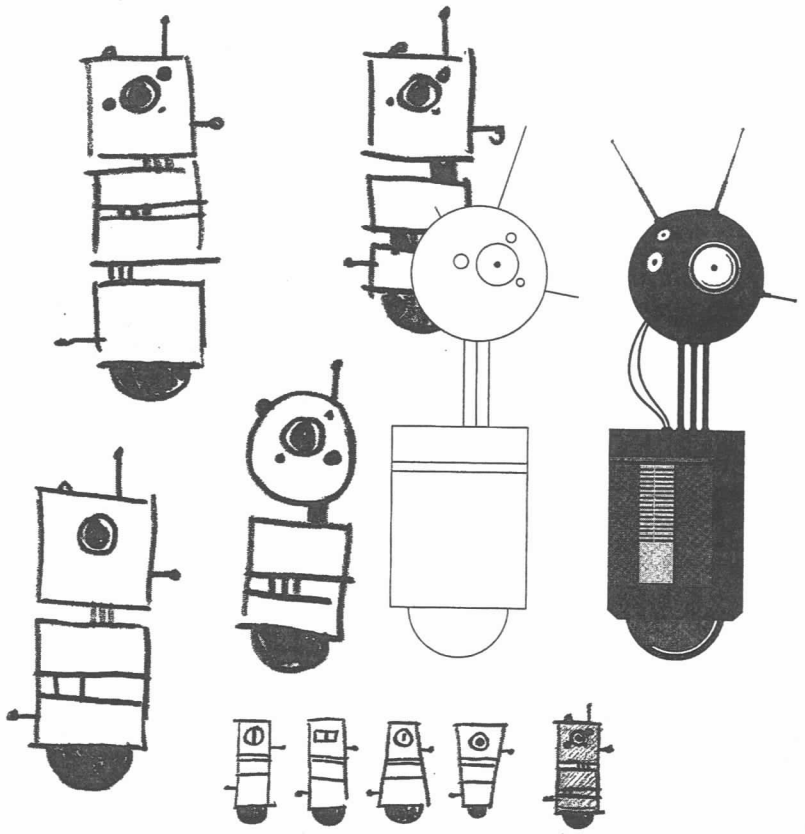


図1-1 スケッチブックに描いた絵をスクリーンに移すことで、新たな可能性が生まれます。

Flexibility

柔軟性

大工さんの腰にぶらさがっている工具入れのように、Processingにはたくさんのツールが備わっています。目的に応じてそれらの組み合わせ方を変えることで、手早く済ませたいハックから本格的な研究まで、さまざまな用途に対応できます。最小のプログラムは1行、大きいものは数千行に及びます。それだけ成長と変化の余地があるわけです。100種類以上あるライブラリでProcessingを拡張すれば、サウンド、コンピュータビジョン、デジタルファブリケーションといった、より高度な領域にも応用できるでしょう（図1-2）。

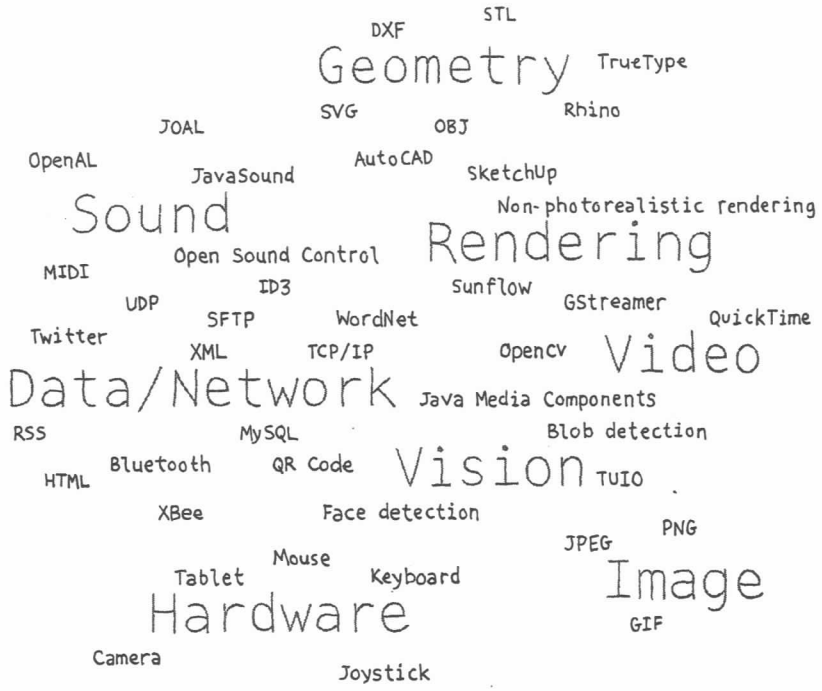


図1-2 Processingはさまざまな情報を扱うことができます。

Giants

巨匠たち

1960年代から人々はコンピュータを使って絵を描いてきました。先人が築き上げたこの歴史から多くを学び取ることができます（図1-3）。Processingも、デザイン、コンピュータグラフィック、アート、建築、統計学といった分野の思想家を含む数多の巨匠たちから影響を受けています。Ivan Sutherlandの"Sketchpad"（1963）、Alan Kayの"Dynabook"（1968）、Ruth Leavittの"Artist and Computer 1"[*1]（Harmony Books、1976）などを調べてみましょう。ACM SIGGRAPHのアーカイブは、ソフトウェアとグラフィックスに関する魅力的な資料を提供しています。

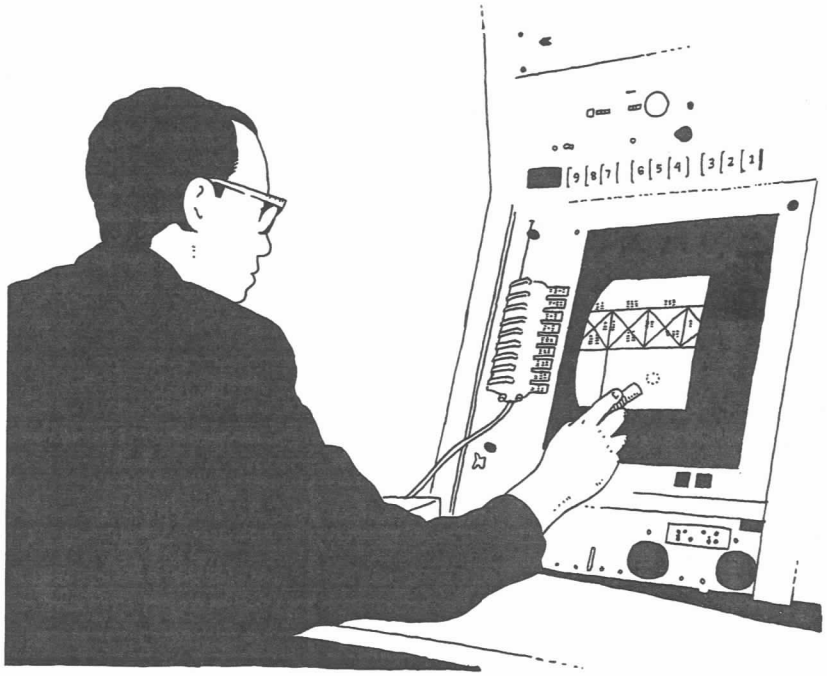


図1-3 Processingは過去半世紀の偉大な巨匠たちの発想からインスパイアされました。

*1 原注：http://www.atariarchives.org/artist/

Family Tree

家系図

人間の言語と同様に、コンピュータ言語にも語族があります。Processingは Javaの方言で、その構文はほぼ同一です。ただし、Processingにはグラフィックとインタラクションに関連するいくつかの独自構文が追加されています（図1-4）。グラフィックに関してはPostScript（PDFの基盤）とOpenGL（3Dグラフィックの規格）の親戚といえるでしょう。Processingを学ぶことは、そういった他の言語やソフトウェアツールの基本を習得することにもつながります。

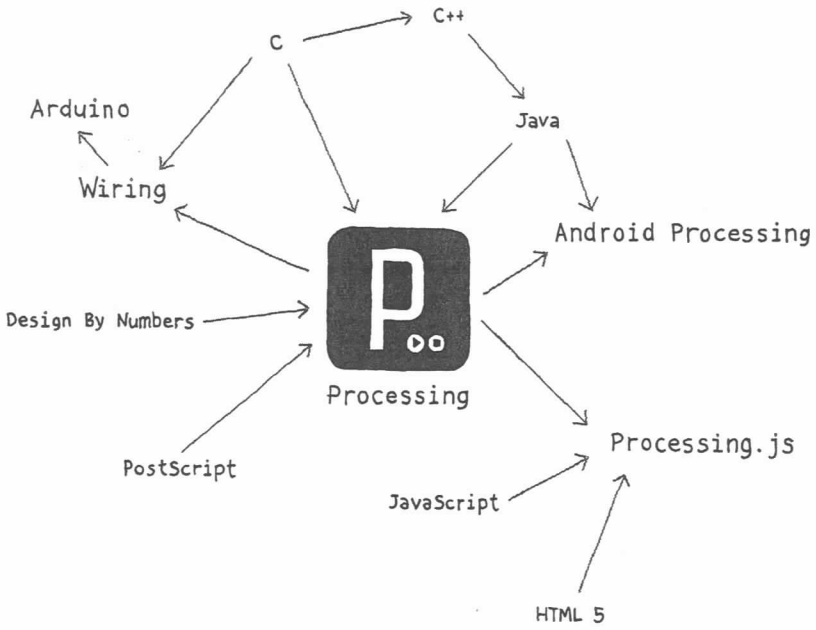


図1-4 Processingは多くのプログラミング言語や開発環境と親戚関係にあります。

Join In

コミュニティ

日々たくさんの人がProcessingを使っています。誰であってもProcessingをダウンロードするときにお金を払う必要はありません。コードに手を加えて、自分の用途に合うProcessingを作ることも許されています。ProcessingはFLOSSプロジェクト*1です。コミュニティの精神に則って、知識や作品をシェアすることが奨励されており、その成果はProcessing.orgや多くのソーシャルネットワークサービスで見つけることができるでしょう。Processing.orgには、そうした情報源に対するリンクも用意されています。

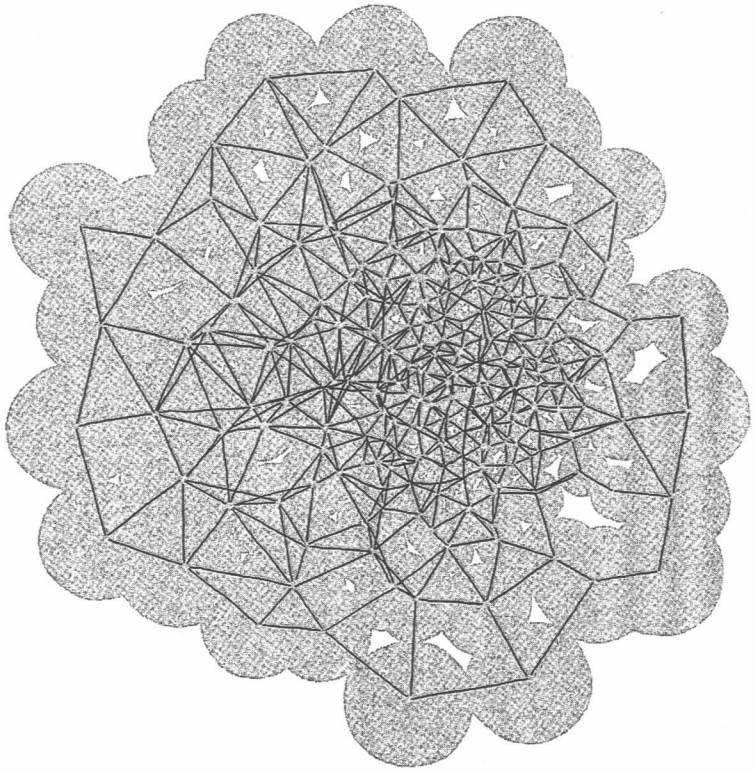


図1-5 Processingはインターネットを通じて寄せられるたくさんの人の貢献によって活気づけられてきました。この図はそうした人々のつながりを表現したものです。

*1 原注：FLOSSは無料（Free）で自由（Libre）なオープンソースソフトウェア（OSS）の略。

2

コードを書いてみよう

Starting to Code

この本を活用するには、読むだけでなく実際にやってみることが大事です。コードを入力し、動かしてみてください。目だけでなく、手も使うと理解が深まります。Processingをダウンロードして、1つ目のスケッチを動かすことから始めましょう。

まず、http://processing.org/downloadを開いて、Mac OS X、Windows、Linuxのなかから自分が使っているマシンと同じものを選んでください。どのマシンでもインストールは簡単です。

- Windowsの場合、ダウンロードが終わると.zipファイルが手に入ります。それをダブルクリックして、フォルダを取り出してください。現れたフォルダをハードディスク上の好きな位置へ移動します。「Program Files」のなかでもいいですし、デスクトップに置いてもかまいません。そのフォルダのなかにあるprocessing.exeをダブルクリックすると、Processingがスタートします。

- Mac OS Xでは、ダウンロードすると現れるProcessing.appアイコンを「アプリケーション」フォルダへドラッグしてください。アクセス権の制約などで書き込めないときは、デスクトップに置いてもかまいません。Processingアイコンをダブルクリックするとスタートします[*1]。

- Linuxバージョンは.tgzファイルです。ホームディレクトリにダウンロードしたら、ターミナルを開き、次のように入力してください（xxxxの部分はバージョン番号です）。

```
tar xvfz processing-xxxx.tgz
```

processing-1.xといった名前のフォルダが現れたら、そのなかに移動して、実行します。

```
cd processing-xxxx
./processing
```

ここまで問題がなければ、Processingのメインウィンドウが見えているはずです（図2-1）。セットアップの環境は人それぞれなので、どこかでつまずいてしまい、プログラムをうまく実行できないこともあるかもしれません。そういうときは、次のページを参照してください。
http://wiki.processing.org/w/Troubleshooting

*1 訳注：Mac OS X Lionでは、最初に起動するとき、Javaランタイムが必要であることを告げるダイアログが表示されます。インストールボタンを押すと、必要なプログラムが自動的にインストールされ、Processingを実行する準備が整います。

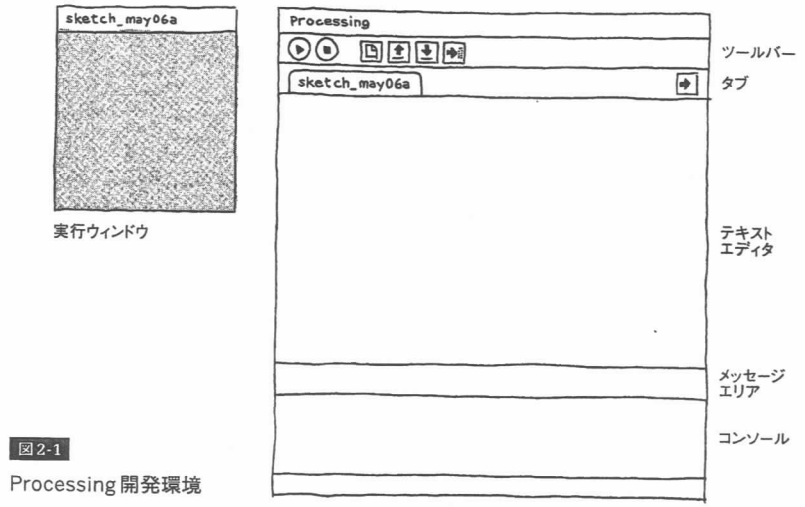


図2-1
Processing開発環境

最初のプログラム

Processing開発環境（PDE）の準備ができました。見ると、そんなに複雑なものではないことがわかるでしょう。一番広いエリアはテキストエディタです。その上に一列のボタンがありますが、ここはツールバーと呼ばれます。エディタの下はメッセージエリア、さらにその下はコンソールです。メッセージエリアは1行単位のメッセージの表示に使われます。コンソールにはもっと細かい技術的な情報が表示されます。

Example 2-1：円を描く

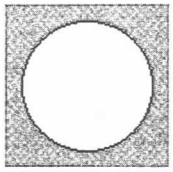
エディタを使って、次のように打ち込んでください。

```
ellipse(50, 50, 80, 80);
```

このコードは「中心が左から50ピクセル、上から50ピクセルの位置にある、幅80ピクセル、高さ80ピクセルの円を描け」という意味です。

こんな形をしているRunボタンを押してみましょう。

すべて間違いなく入力できていれば、次のような円が表示されます。



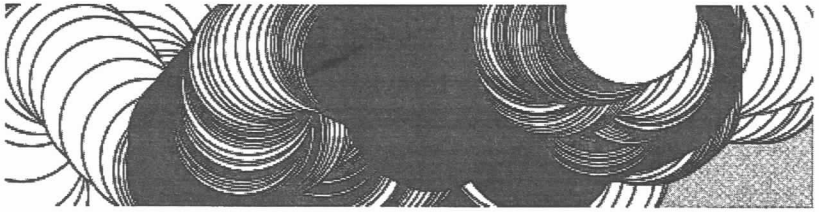
打ち間違いがあると、メッセージエリアが赤くなって、エラーがあることを訴えてきます。もしそうなったら、自分が打ち込んだコードが先の例と完全に同じか確認してください。数値はすべてカッコのなかにあって、カンマで区切られているでしょうか。行の最後のセミコロンは忘れていないでしょうか。

プログラミング初心者にとって難しいことのひとつは、文法をとても厳密に守らなくてはならない点です。Processingは、あなたがしたいことを察してくれるほど賢くありませんし、おかしな句読点の位置を都合よく解釈してくれたりもしません。あなたのほうが少しずつ慣れていく必要があります。

次はもうちょっとエキサイティングなスケッチです。

Example 2-2：円の生成

先ほどのコードは消して、こんどは次のように入力してください。



```
void setup() {
  size(480, 120);
}

void draw() {
  if (mousePressed) {
   fill(0);
  } else {
   fill(255);
  }
  ellipse(mouseX, mouseY, 80, 80);
}
```

このプログラムは幅480ピクセル、高さ120ピクセルのウィンドウを開いたあと、マウスカーソルの位置に白い円を描きます。マウスのボタンを押している間は、円の色が黒に変わります。プログラムの詳しい説明は後ほどしますので、とりあえず今はコードを走らせて、マウスを動かし、クリックして、どうなるか見てみましょう。

実行と停止

Runボタンの隣にあるStopボタンの意味は説明しなくても分かりますね。

ボタンを使いたくない人は、Sketchメニューからも同じことができます。Ctrl-R（MacではCmd-R）というショートカットも、Runボタンを押すのと同じ意味です。

Sketchメニューには「Present」という項目があります。これは画面全体をクリアして、スケッチだけを表示したいときに使います。

スケッチの管理

下向き矢印のSaveボタンはとても重要です。プログラムを保存したいときに使います。

この機能はFileメニューからも実行できます。

通常、あなたが作ったプログラムは、「sketchbook」というフォルダにまとめて保存されます。

上向き矢印のOpenボタンをクリックすると、あなたのsketchbookに保存されている全スケッチのリストが現れます。このリストには、あらかじめProcessingにインストールされているプログラム例（example）も含まれます。

書きかけのスケッチはこまめに保存しましょう。新しいことを試す前に名前をつけて保存しておけば、すぐに元の状態へ戻せます。そうすることで、失敗への備えにもなります。スケッチがディスク上のどこに保存されているかを知りたいときは、Sketchメニューの「Show Sketch Folder」を実行してください。

新しいスケッチを作るときは、Newボタンを押します。

現在のウィンドウは、確認後、空になります。空にしたくない場合は、シフトキーと同時にNewボタンを押すと、別のウィンドウを開いて新しいスケッチを作ることができます。この機能は、Fileメニューから「New」を実行するときにも有効です。

シェアしよう

作品の共有はProcessingの重要なテーマのひとつです。Exportボタンはこんな形をしています。

このボタンを押すと、コードは「applet」という名前のフォルダにまとめられ、ウェブサーバにアップロードできる状態になります（図2-2）。Exportの実行後、appletフォルダの内容がデスクトップに表示されます。PDEファイルはソースコード、JARファイルはプログラム、HTMLファイルはウェブページです。GIFファイルはプログラムをロードしている間、ブラウザに表示されます。index.htmlファイルをダブルクリックするとブラウザが立ち上がって、あなたのスケッチがウェブページとして表示されるでしょう。

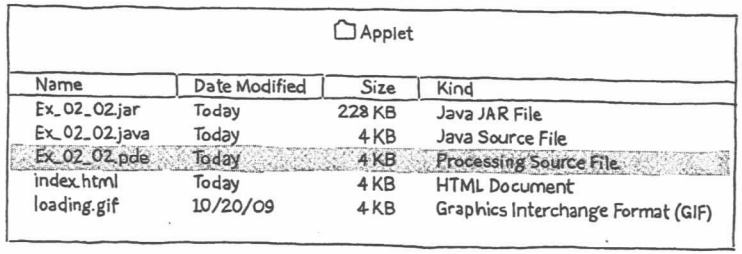


図2-2 appletフォルダには書き出されたスケッチが保存されています。

[NOTE] appletフォルダは、Exportコマンドを実行するといったん消去され、改めて作成されます。フォルダ内のファイル（たとえばHTMLファイル）を修正したら別の場所にコピーしておかないと、次のExportで消えてしまいます。

Fileメニューには「Export Applet」と「Export Application」というコマンドがあります。Export Appletは先ほどのExportボタンと同じ働きですが、Export ApplicationはMac、Windows、Linuxから実行環境を選択した上で、アプリケーションを出力するコマンドです。必要なファイルがすべてまとめられた、ダブルクリックで起動できる作品を簡単に作ることができます（図2-3）。

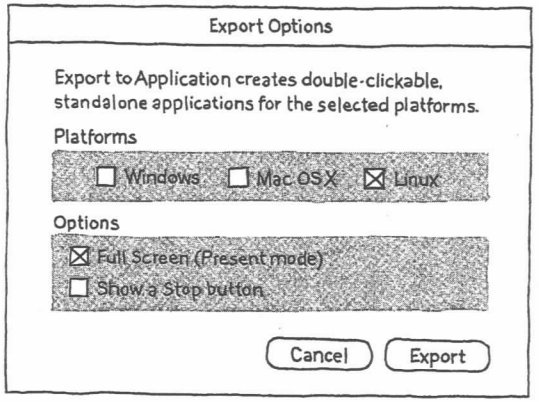


図2-3 Export Applicationの実行時に現れるオプション設定ウィンドウ

ツールバーのExportボタンをシフトキーと同時に押すと、「Export Application」を実行したことになります。

例とリファレンス

Processingを学ぶ過程で、たくさんのコードを探検することになるでしょう。いろんなコードを走らせ、書き換え、ときには壊し、新しいものに生まれ変わるまで拡張してください。そのために、ダウンロードしたProcessingには個々の機能をデモするたくさんの例（example）があらかじめ入っています。例を見たいときは、Fileメニューの「Examples」を選択するか、ツールバーのOpenボタンをクリックしてください。Form、Motion、Imageといった機能ごとに分類されています。リストを眺めて面白そうなトピックを見つけたら、すぐに試してみましょう。

プログラムを見ると、オレンジ色で表示されているコードがあるはずです。オレンジはProcessing固有の言葉であることを示していて、それをマウスで選択してからHelpメニューの「Find in Reference」を実行すると、ブラウザ上にリファレンスが表示されます。Helpメニューの代わりに、右クリック（MacではCtrl＋クリック）で開くこともできます。このリファレンスは次のURLでも参照可能です。

http://www.processing.org/reference/

Processingリファレンスは個々の機能に関する解説と例文の集まりです。リファレンスにある例文は、Examplesのコードよりもずっと短い４～５行程度のものが主なので、理解しやすいでしょう。プログラミングをしている間はずっと開いておくことをおすすめします。項目はトピックごと、あるいはアルファベット順に整理されていますが、量が多いので、捜し物があるときはブラウザ上でテキスト検索を使ったほうが見つけやすいかもしれません。

初心者が使うことを念頭に、明解で理解しやすいリファレンスとなるよう心がけています。ありがたいことに、これまで多くの利用者から間違いを指摘する連絡をもらいました。あなたがもしリファレンスの誤りや改良点を見つけたら、各ページの先頭にある「please let us know」というリンクを使って報告してください。

3

描く
Draw

コンピュータの画面に図形を描くことは、方眼紙の上でそうするのに似ています。最初は技術的手続きばかりですが、新しいアイデアをいくつか導入することで単純な図形の描画からアニメーションやインタラクションへと発展させます。いきなり難しいものに取り組むのではなく、基礎からはじめましょう。