HZ BOOKS
华章教育

计 算 机 科 学 丛 书

ELSEVIER
爱思唯尔

原书第2版

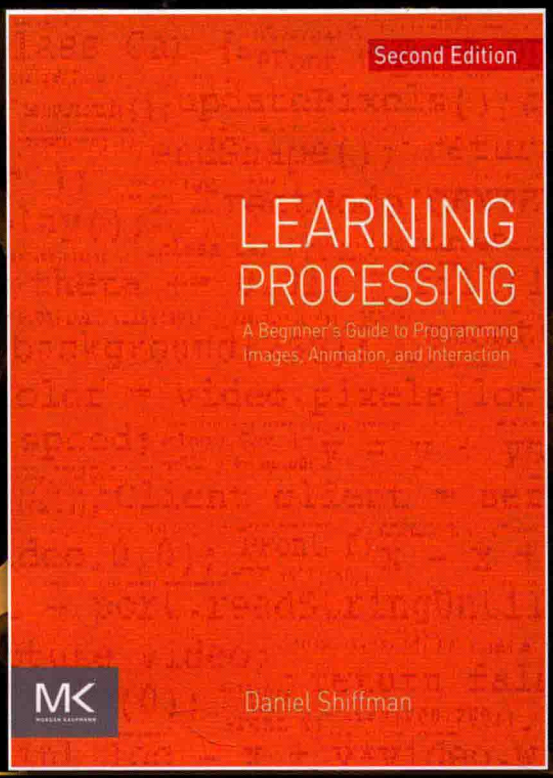
Processing编程学习指南

[美] 丹尼尔·希夫曼（Daniel Shiffman） 著
李存 译

Learning Processing

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

Second Edition



机械工业出版社
China Machine Press

计 算 机 科 学 丛 书

原书第2版

Processing编程学习指南

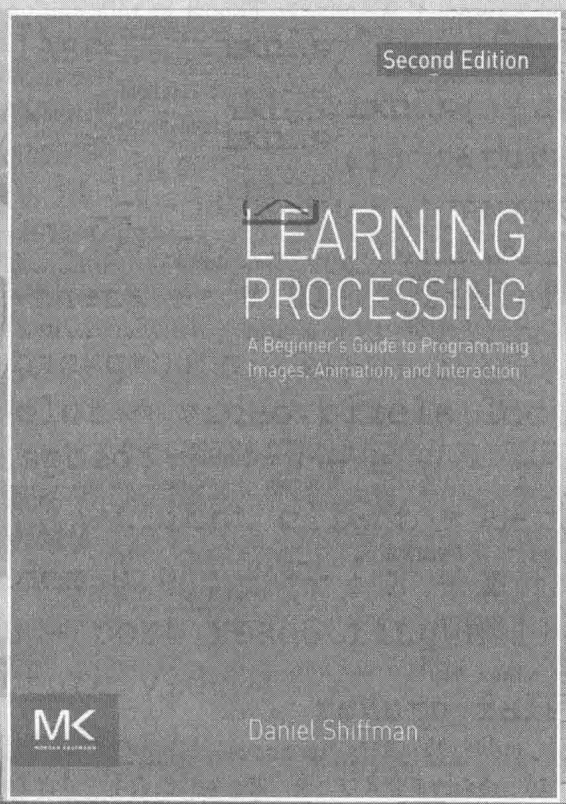
[美] 丹尼尔·希夫曼（Daniel Shiffman） 著

李存 译

Learning Processing

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

Second Edition



机械工业出版社
China Machine Press

图书在版编目（CIP）数据

Processing 编程学习指南（原书第 2 版）/（美）丹尼尔·希夫曼（Daniel Shiffman）著；李存译 . —北京：机械工业出版社，2017.3
（计算机科学丛书）
书名原文：Learning Processing：A Beginner's Guide to Programming Images, Animation, and Interaction, Second Edition

ISBN 978-7-111-55867-5

I. P… II. ① 丹… ② 李… III. 程序设计—指南 IV. TP311.1-62

中国版本图书馆 CIP 数据核字（2017）第 015310 号

本书版权登记号：图字：01-2016-0706

本书详细介绍了 Processing 编程的基本原理，全书分为十节课共 23 章，涵盖了创建最前沿的图形应用程序例如互动艺术、实时视频处理和数据可视化所需要的基础知识。此外，作为一本实验风格的手册，书中精心挑选了部分高级技术进行详尽解释。可以让图形和网页设计师、艺术家及平面设计师快速熟悉 Processing 编程环境。

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号　邮政编码：100037）
责任编辑：缪　杰　　责任校对：董纪丽
印　　刷：中国电影出版社印刷厂　　版　　次：2017 年 3 月第 1 版第 1 次印刷
开　　本：185mm × 260mm　1/16　　印　　张：27
书　　号：ISBN 978-7-111-55867-5　　定　　价：99.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88378991　88361066　　投稿热线：（010）88379604
购书热线：（010）68326294　88379649　68995259　　读者信箱：hzjsj@hzbook.com

文艺复兴以来，源远流长的科学精神和逐步形成的学术规范，使西方国家在自然科学的各个领域取得了垄断性的优势；也正是这样的优势，使美国在信息技术发展的六十多年间名家辈出、独领风骚。在商业化的进程中，美国的产业界与教育界越来越紧密地结合，计算机学科中的许多泰山北斗同时身处科研和教学的最前线，由此而产生的经典科学著作，不仅擘划了研究的范畴，还揭示了学术的源变，既遵循学术规范，又自有学者个性，其价值并不会因年月的流逝而减退。

近年，在全球信息化大潮的推动下，我国的计算机产业发展迅猛，对专业人才的需求日益迫切。这对计算机教育界和出版界都既是机遇，也是挑战；而专业教材的建设在教育战略上显得举足轻重。在我国信息技术发展时间较短的现状下，美国等发达国家在其计算机科学发展的几十年间积淀和发展的经典教材仍有许多值得借鉴之处。因此，引进一批国外优秀计算机教材将对我国计算机教育事业的发展起到积极的推动作用，也是与世界接轨、建设真正的世界一流大学的必由之路。

机械工业出版社华章公司较早意识到“出版要为教育服务”。自 1998 年开始，我们就将工作重点放在了遴选、移译国外优秀教材上。经过多年的不懈努力，我们与 Pearson，McGraw-Hill，Elsevier，MIT，John Wiley & Sons，Cengage 等世界著名出版公司建立了良好的合作关系，从他们现有的数百种教材中甄选出 Andrew S. Tanenbaum，Bjarne Stroustrup，Brian W. Kernighan，Dennis Ritchie，Jim Gray，Afred V. Aho，John E. Hopcroft，Jeffrey D. Ullman，Abraham Silberschatz，William Stallings，Donald E. Knuth，John L. Hennessy，Larry L. Peterson 等大师名家的一批经典作品，以“计算机科学丛书”为总称出版，供读者学习、研究及珍藏。大理石纹理的封面，也正体现了这套丛书的品位和格调。

“计算机科学丛书”的出版工作得到了国内外学者的鼎力相助，国内的专家不仅提供了中肯的选题指导，还不辞劳苦地担任了翻译和审校的工作；而原书的作者也相当关注其作品在中国的传播，有的还专门为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近两百个品种，这些书籍在读者中树立了良好的口碑，并被许多高校采用为正式教材和参考书籍。其影印版“经典原版书库”作为姊妹篇也被越来越多实施双语教学的学校所采用。

权威的作者、经典的教材、一流的译者、严格的审校、精细的编辑，这些因素使我们的图书有了质量的保证。随着计算机科学与技术专业学科建设的不断完善和教材改革的逐渐深化，教育界对国外计算机教材的需求和应用都将步入一个新的阶段，我们的目标是尽善尽美，而反馈的意见正是我们达到这一终极目标的重要帮助。华章公司欢迎老师和读者对我们的工作提出建议或给予指正，我们的联系方法如下：

华章网站：www.hzbook.com
电子邮件：hzjsj@hzbook.com
联系电话：（010）88379604
联系地址：北京市西城区百万庄南街 1 号
邮政编码：100037

华章科技图书出版中心

译者序

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

Processing 诞生于美国麻省理工学院媒体实验室（MIT Media Lab）的美学与计算研究小组。麻省理工学院媒体实验室一直致力于将科技、媒体、科学、艺术以及设计融合在一起。自然 Processing 也具有融合艺术性和科学性的基因：它以数字艺术为背景，通过可视化的方式进行编程，在 Java 语言的基础上简化语法，并具备跨平台的特性。

Processing 起初是专门为视觉交互和媒体艺术设计而创建的，它是面向艺术家和设计师所开发的语言。在创意性行业中，工具会影响创作过程进而影响创作结果。当下的诸多设计软件，比如 Photoshop、Illustrator、Flash 和 3ds Max 等，固然可以便捷地满足艺术家和设计师的基本需求，但是 Processing 的出现让我们再次意识到：编程并不仅仅是工程师的工作，编程可以为广大艺术家和设计师以及所有想以编程方式实现绘画、动画和交互的人提供一个有效的途径。我们不必再囿于生产工具和软件的限制，借助于 Processing，我们可以充分发挥自己的创意和想法。

Processing 的另一大特点在于充分利用网络论坛，将其开源的优势发挥至最大。网站 www.processing.org 将大量的开发者、教师和艺术家聚集到一起，通过公开交流创意和作品来实现代码的共享，进而大大拓展了 Processing。这是一个你可以参与其中的、充满勃勃生机的社区。

本书的作者 Daniel Shiffman 是美国纽约大学帝势艺术学院（Tisch School of the Arts, New York University）的助理教授，本书的内容主要来自于其主讲的“计算媒体导论”（Computational Media）课程。本书的特色体现在以下几个方面：

- 本书适合于零基础编程的读者。作者循序渐进地讲解了 Processing 编程的基础知识，尤其是其中的高级编程知识并不仅仅局限于 Processing，对学习其他编程语言也有巨大帮助。掌握 Processing 之后，可以方便地拓展至其他语言。而对于具有一定编程基础的读者，学习 Processing 会更加容易。
- 本书的语言风格平易近人，通俗易懂。阅读本书就像是一位和蔼的老师亲自向你传授知识。本书使用了大量的比喻：“一个变量就像一个桶。你可以在木桶里放一些东西，提着桶走来走去，这比直接拿着那些东西更加轻便，你也可以随时将桶里的东西取回。”生动的比喻使得编程知识不再那么晦涩难懂。
- 本书采用增量开发的理念（philosophy of incremental development），也就是循序渐进的讲述方式：从整体到局部再到整体，将复杂的程序分解为尽可能小的部分，各个击破。这是本书的核心原则。而本书贯穿始终的卡通形象 Zoog 践行了这一理念。Zoog 从功能单一的童年开始，作者逐渐为 Zoog 拓展新的功能，这也是 Zoog 长大的过程。
- 本书的配套网站（www.learningprocessing.com）不仅提供了所有案例和练习题的源代码，还有教学视频和其他教程资料。

最后，感谢机械工业出版社华章公司的编辑在本书翻译过程中提供的帮助和支持。由于译者水平有限，书中难免存在错误和疏漏，欢迎广大读者批评指正。希望在 Processing 领域结识更多的朋友。

李　存

纪念 Red Burns

Red Burns，1925 年出生于加拿大渥太华。在经历了人生的各种风风雨雨之后，她于 1971 年在纽约大学创建了备用媒体中心（Alternate Media Center）。这个媒体中心后来发展成为互动电信项目（Interactive Telecommunications Program，ITP），而她于 1982～2010 年间一直在此担任教授。我和 Red Burns 相识于 2001 年，当时她在第 20 个新生入学见面会上向我介绍了上述项目。一开始我有些怕她，但是这并没有持续多久。因为，我很快发现了她的友善之处。在接下来的 12 年中，我非常幸运能够和她共事，并且逐渐体会到她的聪颖过人之处，她坚持认为人性比技术更加重要。人一直是她思考的核心，她认为技术（如本书所传授的内容）不过是表达和沟通的工具。多年来，她对我的指导和友谊最终成就了本书的出版。正如 ITP 人一直所说的："Red Burns 改变了我的人生。"

http://itp.nyu.edu/redburns/

致 谢

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

2001 年秋天，我不经意地参与了纽约大学帝势艺术学院的互动电信项目（ITP）。20 世纪 80 年代早期，我曾在 AppleII+ 苹果电脑上用 BASIC 语言做过一些小尝试，但我从未编写过一行代码。在 ITP 第一学期的“计算媒体导论”课程中，我才第一次接触到计算机编程。没有该学院的启发和支持，我根本无法完成本书的编写。

Red Burns，也就是 ITP 的创建者，一直鼓励并且支持我在 ITP 的前 10 年工作。令人难过的是，她于 2013 年 8 月去世了，本书一方面也是为了纪念她。Dan O'Sullivan 是第一个建议我尝试教授 Processing 的人，并说服我将各种编程教程进行融合。在写本书第 1 版的绝大部分时间里，Shawn Van Every 一直是坐在我旁边的办公室同事，并在整个过程中提供了许多宝贵的建议、代码，以及大量的精神支持。Tom Igoe 从事的物理计算工作则为本书提供了灵感来源，尤其是网络以及串行通信的各种示例整合时，他给予了非常大的帮助。出版本书的缘由是 Clay Shirky 某一天在走廊里遇到我，建议我写一本书，他同样为本书第 1 版的早期草稿提供了许多反馈意见。

在 ITP 计算媒体的所有教师，一直以来也提供了宝贵的意见和反馈：Danny Rozin（为第 15 章和第 16 章提供了灵感来源），Mimi Yin，Lauren McCarthy（其创新的工作成果 p5.js 开阔了我的眼界，让我看到了 JavaScript 和 Web 的世界），Amit Pitaru（为第 1 版中关于声音的章节提供了巨大的帮助），Nancy Lewis，James Tu，Mark Napier，Chris Kairalla，Luke Dubois，Roopa Vasudevan，Matt Parker，Heather Dewey-Hagborg 和 Jim Moore（我第一学期课程的老师）。向一直以来在本书写作过程中提供支持与帮助的以下 ITP 职员表示感谢：Marianne Petit，Nancy Hechinger，Marina Zurkow，Katherine Dillon，Eric Rosenthal，Gabe Barcia-Colombo 和 Benedetta Piantella Simeonidis。同样，也要感谢 ITP 的其他员工：George Agudow，Edward Gordon，Midori Yasuda，Rob Ryan，John Duane，Marlon Evans，Tony Tseng，Matthew Berger，Karl Ward 和 Megan Demarest，没有他们的帮助，本书不可能完成。

ITP 的学生们也为本书提供了大量的反馈意见，太多了，以至于在这里无法逐一提及他们的名字，他们在许多课程中使用本书的示例进行了大量的试验。我有一摞在空白处写满潦草笔记的稿纸，以及关于更正、评论和大量鼓励话语的往来邮件存档，这所有的一切都成为完成本书不可或缺的一部分。

我同样非常感谢 Processing 程序员和艺术家社区源源不断的创作和支持。毫不夸张地说，如果不是因为 Casey Reas 和 Ben Fry 创造了 Processing，我现在很可能已经失业了。通过阅读 Processing 源代码，我掌握了 Processing 一半的知识；Processing 语言、网站和集成开发环境的优雅与简洁使得我和我所有的学生都容易理解和学习。我也收到了来自许多 Processing 程序员的意见、启发和评论。包括 Andres Colubri，Scott Murray，Florian Jennet，Elie Zananiri，Scott Garner，Manindra Mohanara，Jer Thorp，Marius Watz，Robert Hodgin，Golan Levin，Tom Carden，Karsten Schmidt，Ariel Malka，Burak Arikan 和 Ira Greenberg。以下这些老师在他们的课程中采用本书第 1 版的早期内容进行测试，为本书的完成提供了巨大帮助：Hector Rodriguez，Keith Lam，Liubo Borissov，Rick Giles，Amit Pitaru，David Maccarella，Jeff Gray

和 Toshitaka Amaoka。

在本书第 1 版的技术审校过程中，Peter Kirn 和 Douglas Edric Stanley 给我提供了非常详尽的意见和反馈，他们的努力也使本书变得更加完善。Demetrie Tyler 为本书初始的封面设计和正文版式设计做出了巨大的贡献。在此，也要向 David Hindman 表示感谢，他帮助我整理初始的屏幕截图和图表。感谢 Rich Hauck，他帮助我开发了本书第 1 版的网站。

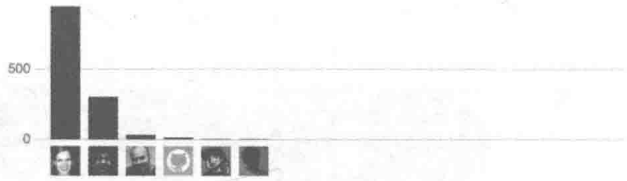
我同样要感谢 Morgan Kaufmann/Elsevier 出版社里为本书第 1 版提供帮助的每一个人：Gregory Chalson，Tiffany Gasbarrini，Jeff Freeland，Danielle Monroe，Matthew Cater，Michele Cronin，Denise Penrose 和 Mary James。

对于本书第 2 版，我要感谢 Morgan Kaufmann/Elsevier 和 O'Reilly 出版社的每一个人，他们一直支持我使用 Atlas 出版平台（https://atlas.oreilly.com/）写作本书。

使用 Atlas 平台使我的工作流程更加顺畅，同时可以采纳许多反馈和意见。Wilm Thoben、Seth Kranzler 和 Jason Sigal 都为第 20 章提供了中肯的反馈和编注。Mark Sawula、Yong Bakos 和 Kasper Kasperman 阅读了本书的 PDF 文档，也提供了有益的评论和反馈。J. David Eisenberg 扮演了实际技术编辑的角色，为完善本书的示例提供了许多很棒的建议。特别要感谢 Johanna Hedva，她在版式修改过程中几乎梳理了整本书的文字。除此之外，正是在她的建议下，本书几处关键的修改内容才得以保留。

来自 Elsevier 出版集团的 Todd Green 忙前忙后地处理与 O'Reilly 和 Atlas 之间合作的具体细则。同样要感谢 Charlie Kent 和 Debbie Clark，他们协助解决了书籍出版过程中的细节问题。总而言之，Atlas 和 O'Reilly 的团队合作让我感到愉快：尽管这本书遇到过各种各样排版上的小问题，但让人感到惊奇的是，使用 CSS 和 XSLT 为版式生成的 HTML 文件解决了这些问题。感谢 Andrew Odewahn，Rune Madsen，Sanders Kleinfeld，Dan Fauxsmith 和 Adam Zaremba 帮助我了解 Atlas 并领教了它的神奇之处。感谢 Rebecca Demarest 提出的关于插图的意见，以及 Ron Bilodeau 关于 CSS 的技术指导。最后一点也非常重要，我要感谢 Kristen Brown，她耐心细致地倾听了我询问的关于我知识短板的每一个细节，让我了解了如何设定并管理日程安排的优先顺序，才能最终确保本书按时完成。在本书 GitHub 库的 pulse 工具中，你可以看到她的贡献大小。

排除合并后，6 位作者分别为 git 的主分支和全部分支提交了 1299 处修改。在主分支中，有 476 个文件进行了修订，总共有 213 930 处补充项和 7 处删除项。



最重要的是，我要感谢我的妻子 Aliki Caloyeras，我的孩子 Elias 和 Olympia，我的父母 Doris 和 Bernard Shiffman，以及我的兄弟 Jonathan Shiffman。他们不仅为本书第 2 版提供了大量帮助，还给予了我精神上的支持和鼓励。

前　言

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

本书讲的是什么

本书讲了一个故事。一个关于解放与自由的故事，一个关于逐步了解计算机基础知识的故事。通过编写代码，可以创造属于你自己的多媒体设计，而不必拘泥于已有的软件工具。这个故事不仅仅是为科学家和工程师准备的，同时也是为你准备的。

本书是为谁准备的

本书是为初学者准备的。如果你到目前从未编写过一行代码，那么本书对你来说再合适不过了。本书的前 9 章会由浅入深地讲授编程的基础知识。你并不需要任何编程的背景知识，只需要有操作电脑的基础知识——打开电脑、浏览网页、运行程序之类的知识就足够了。

由于本书使用 Processing 进行学习，因此对于那些在视觉领域学习或工作的人来说，它就更加适用了，例如图形设计、绘画、雕塑、建筑、电影、视频、插图、网页设计等。如果你从属于上述领域（在上述领域使用电脑），你很可能精通某个特定的设计软件（很可能不止一个软件），例如 Photoshop、Illustrator、AutoCAD、Maya、After Effects 等。而本书的意义在于使你摆脱（至少是部分摆脱）现有软件工具的束缚。如果可以创造自己的工具，而不是使用他人的软件，那你能创造出什么？

如果你已经具有一定的编程经验，并且对 Processing 非常感兴趣，那么本书同样非常有用。本书的前面几章会为你提供一个速成的编程复习资料（和坚实的基础知识），本书的后面则是关于 Processing 编程的高级话题。

什么是 Processing

假设你正在学习 CS 101（Computer Science 101）课程，其中可能讲到了 Java 编程语言的内容。下面是课程中第一个示例程序的输出结果：

一直以来，教授给程序员的基本命令行输出是：

1. 文本输入（TEXT IN）→以文本的形式编写代码。
2. 文本输出（TEXT OUT）→在命令行显示文本输出。
3. 文本交互（TEXT INTERACTION）→用户可以在命令行输入文本，实现和程序的交互。

这个示例程序中的输出“Hello, World!”是一个经典段子，按照惯例，在各种编程语言教学中，“Hello, World”总是作为第一个程序的文本输出。这个示例程序最早出现在1974贝尔实验室的备忘录中，它是由Brian Kernighan撰写的，题名为《Programming in C: A Tutorial》。

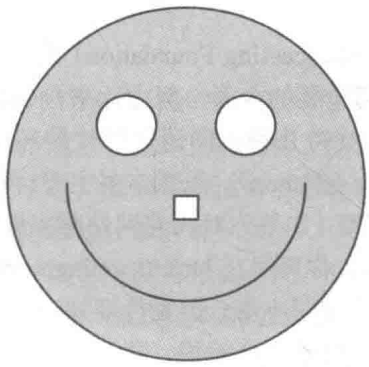
学习Processing的优势在于：它自身强调一种更直观并且基于视觉反馈的编程环境，因而它更有助于艺术家和设计师学习编程。

1. 文本输入（TEXT IN）→以文本的形式编写代码。

2. 视觉输出（VISUALS OUT）→在窗口显示视觉输出。

3. 鼠标交互（MOUSE INTERACTION）→用户通过鼠标和程序进行交互（在本书中你会看到更多例子）。

在Processing中，“Hello，World！”很可能如下图所示：



你好，图形！

虽然看上去是相当友好的设计，但是它并没有引人注目的感觉（在这里暂且忽略掉第3步：交互），“Hello, World!”也是这样。然而，这种方式所聚焦的理念（通过即时的视觉反馈来学习）却是截然不同的。

Processing并不是第一个遵循这种范式的语言。1967年，Daniel G. Bobrow、Wally Feurzeig和Seymour Papert创造了Logo编程语言。一名程序员使用Logo语言编写了一个指令：在屏幕上用龟标生成图形和设计。尔后John Maeda在1999年设计了名为Design By Numbers的语言，该语言使视觉设计师和艺术家以简单、易用的句法来进行编程。

尽管这些语言具有令人惊叹的简洁性和创新性，但它们的功能非常有限。

Processing作为Logo和Design by Numbers的直系后代，于2001年诞生于麻省理工学院媒体实验室的美学与计算研究小组。它是由Casey Reas和Benjamin Fry设计的开源语言，当时他们是著名的计算机艺术家John Maeda的研究生。

Processing是一门开源编程语言，提供了对图片、动画和声音进行编程的环境。学生、艺术家、设计师、建筑师、研究人员和业余爱好者可以使用Processing进行学习、制作原型以及作为生产工具。你可以通过视觉化界面学习计算机编程的基础知识，或者作为软件速写本以及专业化的生产工具。除了该领域的其他相关专有软件之外，Processing为艺术家和设计师提供了一个新的选择。

——www.processing.org

总之，Processing是非常令人惊叹的。首先，它是免费的，你不用花一分钱。其次，由于

Processing 基于 Java 编程语言（本书后面的章节将会对此做进一步探讨），因此它是一门十分实用的功能性语言，没有 Logo 或者 Design by Numbers 语言的限制，使用 Processing 几乎可以实现各种功能。最后，Processing 是开源的。虽然在大多数情况下这并不会是本书内容的关键细节，可是，随着深入学习 Processing，你就会意识到这种开源的理念是非常宝贵的。正是源于此，大量的开发者、教师和艺术家才会聚集到一起分享作品，贡献想法，进而大大拓展了 Processing。

快速浏览一下 processing.org 网站，你就会发现这是一个充满勃勃生机、具有创造力的社区。在这里，初学者和专家通过公开交流创意和作品共享代码。尽管网站上有完整的参考文档，以及数量庞大的示例帮助你快速上手，但是并没有给真正的初学者提供一个系统的详尽教程。本书通过详尽地介绍编程基础知识和探索高级编程话题，可以帮助你参与到 Processing 这个社区网站，并做出你的贡献。

2012 年，Processing 基金会（Processing Foundation）成立，它旨在规范 Processing 软件的目标和理念："在编程知识越来越重要的今天，努力让来自各行各业的人都能轻松学习编程。"为了实现这个目标，基金会大力支持几种不同语言的软件环境，其中包括 Processing（Java）、p5.js（JavaScript）和 Processing.py（Python）。虽然本书主要讨论 Java 框架的知识，但是我也极力向你推荐其他几个编程语言框架（如果你对构建网站感兴趣的话，尤其推荐 p5.js）。我同样保留了本书所有示例的 p5.js 版本，你可以在 http://learningprocessing.com 上找到。

虽然没有 Processing 就不可能编写本书，但是你要知道，本书实质上并不仅仅是一本关于 Processing 的书。编写本书的初衷是教会你编程。我只是选择了使用 Processing 作为编程的学习环境，但本书所关注的是核心计算编程概念，这些概念将会在你以后学习其他编程语言和环境时，继续带领你前行。

难道我不应该学习_____

在空白处填上你想学习的编程语言。你可能曾经听说某门编程语言"Flibideeflobidee"将会是未来前景最广阔的语言。你肯定听出来这是瞎编的，但是我敢肯定你的某个朋友曾经不断跟你讲某门语言是多么功能强大。它是如何使得编程变得如此容易。使用这门语言，5 分钟之内你就能实现以前需要花费一天时间才能完成的程序。而且，它在 Mac 上、个人电脑上，甚至烤面包机上都能运行！你还可以用它编写一个陪你聊天的宠物！而且是用日语聊天！

事情是这样的。那个可以解决你所有问题的神奇编程语言根本不会存在。没有一门语言是完美的，Processing 也有与生俱来的缺陷和不足。可是 Processing 是一个学习编程的很棒的起点。本书传授计算机编程的基本原理，不论是使用 Processing、Java、JavaScript、C、Python 还是其他语言，它们都会使你受益终身。

当然，对于某些项目来说，其他语言和环境可能更加适合。但 Processing 对于大部分的项目来说都是相当不错的选择，尤其是媒体相关和基于屏幕的任务。对 Processing 一个普遍的误解是它只适合于小打小闹，其实并不是这样的：许多人（包括我在内）都在项目自始至终使用 Processing。Processing 可以用来制作网络应用、博物馆和美术馆的艺术装置、公共空间的展览互动装置。比如，我曾经使用 Processing 在纽约市军队总部的大厅里制作一个实时的图像视频墙，它展示在 120 英尺[⊖] ×12 英尺（没有错，是英尺！）大的屏幕上。

⊖ 1 英尺＝0.3048 米。

Processing 不仅适合于项目制作，它还非常容易上手，它真的很棒。它是免费的、完全开源的软件；它的界面简洁；它是基于视觉的工具；它还非常有趣；它是面向对象的语言（后面会讲解）。此外，它能够在 Mac 端、PC 端以及 Linux 机器上运行。

但是 Processing 的一个短板是对于网页的兼容性不足。2001 年，在 Processing 刚诞生的时候，Java applet 是将实时图形项目发布到网页的主要方法。可是到了 2015 年，Java applet 已经不复存在。由 Lauren McCarthy 倡导的 Processing 基金会的 p5.js 项目（http://p5js.org）现在成为一个新的选择。关于这点，本书第 21 章将会具体探讨。

说了这么多，我就是想告诉你不要再去纠结应该选择哪门编程语言了，应该把精力集中到学习 Processing 编程的基本原理上。这方面的知识将有助于你超越本书的有限内容，帮助你学习其他任何编程语言。

把想法直接写在这本书上

假如你是一名小说家或者剧本作家，你的写作时间仅仅是坐在计算机前打字的时间吗？大多数情况下并不是这样。或许是晚上躺在床上的时候，脑海中突然出现了一些想法；或许是坐在公园的长椅上惬意地喂着鸽子时，脑海中上演着一幕幕的对话；又或许是有一天晚上在酒吧，你在纸巾上快速勾勒出一个精彩的故事情节。

好吧，其实编写软件、程序、代码并没有什么不同。只是由于编程工作本质上和计算机紧紧捆绑在一起，因此你常常会忘记这一点。但有机会的话，你一定要让你的头脑发散、畅想。在远离办公桌、电脑的时候去头脑风暴一些奇思妙想。就我个人而言，我常常在慢跑的时候完成了编程工作最棒的构思。

当然，使用电脑实际输入代码的部分也是非常重要的。我的意思是，虽然不可能仅仅通过舒舒服服地躺在游泳池里就能完成一个复杂的任务，但如果只是每天伏案工作，面对着刺眼的显示器，这是远远不够的。

所以，随时在书上做笔记就是一个好的方法，这样能锻炼你离开键盘后思考代码的能力。我已在本书中包含了许多填空形式的练习题。（这些练习题的所有答案都可以在本书配套网站 http://learningprocessing.com 上找到，方便你检查自己的答案。）充分利用本书的空白处吧！每当你有灵感和想法的时候，就迅速把它们写到书上。把本书当成一个练习册或者速写本。（当然，你也可以使用自己的速写本。）

最后我建议，你要花一半的时间在不用电脑的时候阅读本书，另一半时间则是坐在计算机旁，实践本书中的示例。

我应该如何阅读本书

最好是按照章节顺序阅读。第 9 章之后，你就可以轻松地随便翻看本书了，但是前面几章，建议你按照顺序来读。

本书按照先后顺序教你编程。更高级的阅读方式则是：根据自己的需要跳读，将本书作为一个参考文档来使用。本书的前一半内容都是首先讲解一个示例，然后一步一步分解这个示例中所涵盖的知识点。除此以外，计算机编程的基本原理是按照一个特定的顺序来逐步呈现的，这个顺序是多年来在纽约大学帝势艺术学院的互动电信项目中许多同学反复摸索之后的结果（http://itp.nyu.edu）。

我将本书所有 23 章内容分为十节课。前面 9 章介绍了计算机图形学，涵盖了计算机

编程的基本原理。第 10～12 章则暂停讲授新知识，转向讨论如何用增量方法（incremental approach）构建更加大型的项目。第 13～23 章继续拓展基础知识，并且展示一系列更加高级的话题，涉及 3D、直播视频和数据可视化等。

这些内容分为容易理解的几个部分。每节课的末尾，我都准备了一个项目，建议你从单纯阅读本书的过程中转换下思维，尝试将该节课中的全部内容整合为一个完整的项目。我也为这个项目提供了一些建议，但它们真的仅仅是建议而已。

这是一本教科书吗

本书既可用作编程课程导论的教科书，也可以用来自学。

这里，我要再次提及：本书的基本结构直接来自于 ITP 的“计算媒体导论”课程。如果没有同事和这门课数百名学生的帮助（我多么希望我可以把他们所有人的名字都写在这里），本书是不可能完成的。

坦白讲，本书的内容要比针对初学者的一学期课程要多一些。本书共计 23 章，我曾经在课堂上详细讲过其中 18 章内容。可是，不论你是否将本书作为课程教材或是自学读物，你完全可以在几个月里消化本书的知识。当然，你也可以读得更快，但是如果你要在 Processing 中测试本书的代码，并且完成课后项目，确实是需要一段时间的。那些所谓“10 天上 10 节课就能学会编程”的书看似非常吸引人，但实际并不现实。

下面是一个用 14 周时间学完本书内容的参考计划。

第 1 周	第一节课：第 1～3 章
第 2 周	第二节课：第 4～6 章
第 3 周	第三节课：第 7～8 章
第 4 周	第四节课：第 9 章
第 5 周	第五节课：第 10～11 章
第 6 周	期中！（继续第五节课：第 12 章）
第 7 周	第六节课：第 13～14 章
第 8 周	第七节课：第 15～16 章
第 9 周	第八节课：第 17～19 章
第 10 周	第九节课：第 20～21 章
第 11 周	第十节课：第 22～23 章
第 12 周	最终项目研讨会
第 13 周	最终项目研讨会
第 14 周	最终项目展示

本书有测验题目吗

师傅领进门，修行在个人。真正的窍门在于练习，练习，练习。假设你现在是 10 岁的孩子，在学习小提琴课程，老师肯定会跟你讲每天都要练习。同理，要完成本书提供的练习，如果可能的话每天都要练习。

作为初学者，在学习的过程中，一开始可能并不会提出自己的想法，而这也就是那些练习

存在的目的。不过，如果你有自己开发项目的想法，那就跳过练习，尝试去实现它。

大多数练习都是小的演练，几分钟就能完成。有一些则会稍难，可能需要一小时才能完成。在整个学习过程中，有时候可能要暂停学习新知识，花上几小时、一天甚至一周时间来完成一个项目，这也会让你收益颇丰。正如我之前提到的，这就是我这样安排课程结构的用意。我建议你在每节课结束后，暂停阅读，自己用 Processing 做一些小项目、小练习。本书每节课后面都提供给你项目的建议。

本书所有练习的答案都可以在其配套网站上找到。

本书有配套网站吗

本书的网址是：http://learningprocessing.com。

网站上提供了以下内容：

- 本书所有练习的答案
- 本书所有代码的可下载版本
- 本书所有内容的配套教学视频
- 书中示例的在线版本（通过 p5.js 运行）
- 本书额外的提示和教程
- 问题以及评论

本书中的许多示例原本是彩色的，并且具有动态特效，因此书中的黑白和静态屏幕截图无法描绘示例的全部效果。当你阅读时，你可以通过浏览器（使用 p5.js）查看在线示例，或者下载到你的电脑上在本地运行。

本书中示例的源代码同样可以在 Learning Processing github 库（https://github.com/shiffman/LearningProcessing/）上找到。我还使用了 github issues（https://github.com/shiffman/Learning-Processing/issues）作为系统工具来发现本书中错误，所以如果你发现本书中有任何错误，请在那里给予反馈。[⊖]你有可能会在本书示例和在线示例之间发现些许差别，但是它们的核心概念是相同的。（例如，为了适合本书的排版布局，书中的示例是以 200×200 的像素大小呈现的，而在线示例的尺寸可能会相对大一些。）

本书的配套网站并不能取代 Processing 官方网站 http://processing.org。官方网站提供了 Processing 参考文档和更多的示例，此外，还有一个活跃的论坛。

一步一个脚印

增量开发的理念

在你开始学习编程之前，还有一个方法需要和你讨论一下。它是我学习编程的一个重要驱动力，并且也对本书的编写风格产生了巨大影响。这个方法是由我之前的一个教授提出来的，叫做“增量开发的理念”（philosophy of incremental development）。更通俗地讲，就是“一步一个脚印的方法”（one-step-at-a-time approach）。

无论你是一个新手还是具有几年编程经验的程序员，当你面对任何一个编程项目时，千万不要落入毕其功于一役的陷阱中去。比如，你的目标可能是创建一个这样的 Processing 程序：使用 Perlin 噪声按照顺序为 3D 顶点图形生成贴图。这个图形可以通过神经网络中的人工智能

⊖ 中文版读者请在 https://github.com/hzbooks/learning-processing/issues 上给予反馈。

技术实现自行演化。然后通过基于 Web 的数据挖掘抓取每天的新闻和故事，而这些故事内容的字体颜色用户可以语音控制。

拥有一个宏大的愿景并没有错，但是对你来说很重要的一点是：学会如何将这个愿景分解为几个小的部分然后各个击破。前面的例子有一点荒谬，如果你尝试一次性将那个愿景实现出来，我敢保证你最终不得不用冷敷法治疗你的头痛。

为了进行演示，让我们简化一下，假设你希望编写游戏《 Space Invaders 》（详见 http://en.wikipedia.org/wiki/Space_Invaders）。注意，尽管本书并不是一本游戏编程书，但本书可以传授给你游戏编程的基本方法。遵循前面介绍的理念，你应该已经知道每一次只编写一部分内容，这就要把编写《 Space Invaders 》分解为几个部分。下面是我的快速尝试：

1. 编写宇宙飞船的程序。
2. 编写入侵者的程序。
3. 编写记分系统的程序。

这里我将程序分为 3 步！我并没有尝试一次性地解决这个问题。关键就在于将问题分解为尽可能小的部分，有必要的话，甚至可以到极端的地步。毕竟，如果要编写一个诸如《 Space Invaders 》这样复杂的游戏程序，将其分解为几个相对容易上手的部分，就不会觉得那么不知所措。

请牢记以上内容，接下来我开始着手稍微复杂一点的工作，将第 1 步分解为更小的部分。在这里我打算编写 6 个子程序，第一个也是最简单的：显示一个三角形。后面每一步我都会进行小小的改进：移动三角形。随着程序变得愈加复杂，最终我就能够完成整个程序的编写。

1. 编写宇宙飞船的程序。
   a. 在屏幕上绘制一个三角形。这个三角形将会成为宇宙飞船。
   b. 将三角形置于屏幕的底端。
   c. 将三角形置于它之前位置的偏右一侧。
   d. 将三角形变为动态图形，使得它从左侧移动到右侧。
   e. 仅当按向右的方向键时，三角形才会从左侧移动到右侧。
   f. 当点击向左的方向键时，三角形从右侧移动到左侧。

当然，这些仅仅是整个《 Space Invaders 》游戏需要的所有步骤当中的一小部分，但是这阐述了一种重要的思维方式。这种思维方式不但使编程更加容易，而且使得调试（debugging）更加方便。

排除故障[⊖]指的是发现电脑程序中的故障问题并且修复它们，使得程序正常运转的过程。我想你之前曾经听说过故障（bug）这个名词，比如 Windows 操作系统在代码深处偶尔会出现微小而奇怪的错误。对于我们来说，故障是一个更加简单的概念：错误。当你尝试编写某个东西时，很有可能程序并不完全如我们所愿地执行，甚至完全不会。所以说，如果你一开始就把所有内容一次性编写完毕，就很难找到这些故障所在。一步一个脚印的方法则允许你每次处理一个故障。

除此以外，增量开发（incremental development）也非常适合面向对象编程（object-oriented programming）方法，这是本书的核心原则。对象（object）这一概念将会在本书第 8 章进行介

⊖ 术语“调试”来自于一个虚构的故事：故障的原因是一只蛾子被困在了计算机科学家 Grace Murray Hopper 的电脑的继电器电路中。

绍，它将有助于你以模块化的方式开发项目，同时也为构建和分享代码提供了一个很好的方法。可复用性（reusability）也非常关键。例如，如果你已经为《Space Invaders》游戏编写了一个宇宙飞船的程序，现在想让该程序在行星游戏上运行，你可以抓取你需要的部分（也就是宇宙飞船部分的代码），在新的程序中围绕该部分继续进行开发。

算法

归根结底，计算机编程实际是编写算法。算法（algorithm）是用于解决一个特定问题的一系列有序指令。增量开发（对于你来说，本质上是一种算法）的理念就使得编写一个用于实现你想法的算法更加容易。

开始学习第 1 章之前，作为一个练习，尝试去写一个关于你每天都会做的事情的算法，比如刷牙。尽量使这些指令看上去非常直白、简单。（比如：将牙刷往左边移动 1 厘米。）

设想下你需要为一个完全不熟悉牙刷、牙膏以及牙齿的人提供一个完整的指令帮助他完成刷牙任务。这其实就是编写程序了。计算机只不过是一个善于遵循精确指令的智能机器，它并不能理解这个世界。接下来，你就要以一个程序员的身份开始你的旅程、你的故事和你的新生活。你将学习如何与你的计算机朋友交谈。

练习 I-1 入门练习：编写刷牙指令。

一些建议：

- 你是否会基于不同的条件做不同的事情？在你的指令中是否会使用词语“如果”（if）和“否则”（otherwise）？（例如：如果水非常凉，那么加入温水；否则，加入凉水。）
- 在你的指令中使用单词“重复”。例如：上下移动牙刷。重复五次。

同样，注意我是从第 0 步开始的。在编程过程中，通常是从 0 开始算起的，因此最好从现在就养成习惯！

如何刷牙 _____

第 0 步：_____

第 1 步：_____

第 2 步：_____

第 3 步：_____

第 4 步：_____

第 5 步：_____

第 6 步：_____

第 7 步：_____

第 8 步：_____

第 9 步：_____

目 录

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