Document generated on 04/17/2024 3:17 p.m.

International Review of Research in Open and Distributed Learning

Book Review: Exploratory Programming in the Arts and Humanities

Kelly Hammond

Volume 23, Number 2, May 2022

URI: https://id.erudit.org/iderudit/1089168ar
DOI: https://doi.org/10.19173/irrodl.v23i2.5997

See table of contents

Publisher(s)

Athabasca University Press (AU Press)

ISSN

1492-3831 (digital)

Explore this journal

Cite this review

Hammond, K. (2022). Review of [Book Review: Exploratory Programming in the Arts and Humanities]. *International Review of Research in Open and Distributed Learning*, *23*(2), 264–266. https://doi.org/10.19173/irrodl.v23i2.5997

Copyright (c) Kelly Hammond, 2022

This document is protected by copyright law. Use of the services of Érudit (including reproduction) is subject to its terms and conditions, which can be viewed online.

https://apropos.erudit.org/en/users/policy-on-use/

This article is disseminated and preserved by Érudit.

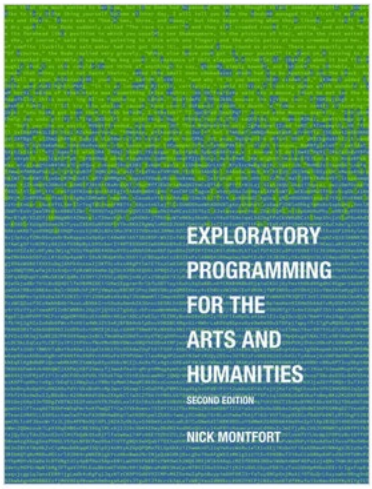
Érudit is a non-profit inter-university consortium of the Université de Montréal, Université Laval, and the Université du Québec à Montréal. Its mission is to promote and disseminate research.

https://www.erudit.org/en/

International **R**eview of **R**esearch in **O**pen and **D**istributed **L**earning
Volume 23, Number 2

May – 2022

Book Review: Exploratory Programming in the Arts and Humanities



Author: Nick Montfort (The MIT Press, 2021, 2nd ed., 363 pages)

ISBN: 9780262044608

Reviewed by: Kelly Hammond, *CUNY Graduate Center*

For humanists interested in incorporating programming into their practice or research, open-access (OA) resources abound. The Internet teems with how-to videos and textbooks—even full-blown OA courses—in open languages such as Python and R. These tutorials can be useful for faculty and students who can extrapolate with ease how a Python program, scraping jobs in the Houston area from monster.com, might be adapted to uncover new questions about women's literature in the late 1800s. For those seeking guidance rooted more directly in the humanities, there is a growing number of humanities-specific OA sites and journals, such as the Programming Historian, that provide lessons on skills useful for humanistic inquiry, such as how to geocode historical data or use stylometry to determine authorship. These resources can be great for faculty and students already comfortable with programming basics who have specific tasks they need to accomplish, but they do not lend themselves to much tinkering for the programming novice.

Nick Montfort's *Exploratory Programming in the Arts and Humanities*, whose second edition was released both in print and on MIT Press Open Access (available with a Creative Commons Attribution-NonCommercial-ShareAlike license), offers humanists something far more important than the programming he promises (and delivers) in the title: a mindset about how computational thinking and play can lead to discovery. He proves that programming, even the simple altering of someone else's code, can be fun, generative, socially transformative, and analytically powerful in humanities work. His book is useful as a complete textbook, as a course in and of itself, or as a dabbler's companion intended to open up possibilities. He writes both for those who have never programmed before as well as those who have. Further, he guides the reader to experiment with programming to explore text, images, and sound, and provides practical programming skills as well as a sense of structure that work across and between disciplines.

Structure and Content

Montfort begins with a grounding, philosophical introduction followed by a pragmatic chapter on installing the free software to be used throughout the book. He then presents fourteen instructional chapters—some devoted to general programming concepts as well as three specifically devoted to working with text, three to working with images, and one on working with sound. In each, he (a) introduces a concept, (b) asks readers to program along as he explains how the concept works, (c) presents free projects prompting readers to practice what they have learned through optional exercises, and (d) offers an essential concepts review for readers to solidify their understanding or spot areas that need a second look. Throughout it all, he provides tips on good practice, instilling habits that will save neophytes time and headaches as they increase their skill. He uses illustrations sparsely and wisely, generally to provide concrete visualization of complex ideas such as swapping pixels. These illustrations look hand drawn, like sketches on a napkin, lending a personal, tactile feel that many humanists crave.

Montfort also includes two not-to-be-missed appendices. The first, titled "Why Program?" shares provocative ideas about how programming can make us better thinkers and how it can shape a better world. Then, "Contexts for Learning" outlines ways the book can be used to teach ourselves and others, from what to do in a one-day workshop to how to sequence chapters for full semester- or quarter-long courses.

What's new in the second edition? Montfort has updated both his instructions and the setup chapter to reflect newer versions of the free and open software packages used for the exercises. He has incorporated the use of free and open Jupyter Notebooks to demystify code interpretation and to allow readers to check their programs more easily as they build them. He has also reorganized the book, frontloading abstract concepts like statistics, probability, visualization, and classification before diving into higher-level chapters such as "Image III: Visual Design and Interactivity" and "Text III: Advanced Text Programming." Importantly, has increased the frequency and visibility of his conviction that the best way to learn is to do.

Overall Impression and Relevance to the Field of Distance Education and E-learning

Exploratory Programming is a testament to what open-access can mean, especially in an e-learning environment. Used in full, it is a free course (that relies on free and open software) from a gifted MIT professor whose pedagogy is clear in structure and tone. He scaffolds, promotes predictive thinking, lauds collaborative learning, and urges readers to *do* not just to *read*. Used in part, it can be equally powerful. Thanks to Appendix B ("Contexts for Learning"), sections of the book can be easily adapted for a range of academic classes. Appendix A ("Why Program?") could serve as a stand-alone philosophical introduction to any course with a required or optional programming component or unit. As online distance education assumes Internet connectivity, and not all students have Internet access at all times, Montfort provides guidance in Chapter 2 on how to program offline.

The book is also significant in that it broadens our scope as humanists. Montfort makes compelling arguments both within and across chapters that text, image, and sound speak to each other, especially

through the language and structures of programming. A scholar of literature or history, hunting for open-access programming resources, may look only for tutorials on working with text, missing perhaps original opportunities to investigate visually or auditorily. The same is true for the scholars of art or archaeology who might limit themselves to working with images. Montfort helps the humanist see possibilities—analytical or creative—and then actually explore them.

The book's only real weakness is its limited cultural scope. References to Shakespeare, baseball, and *Spinal Tap* and the use of terms such as hose (to mean destroy) suggest an imagined reader who is perhaps white, American, and male. At times, Montfort is aware of these cultural assumptions, taking advantage of them to introduce some profound truths about cultural bias in computing. When introducing the conventional Hello, world! exercise, for example, he encourages users to alter the program to return phrases in other languages, not as a superficial nod to multilingual users, but rather as inquiry into the cultural assumptions built into the systems we use. Similarly, when exploring functions with if-else statements, he daringly asks the reader to imagine a world called Binaria where name endings indicate one of two genders. After leading the reader through this binary program, he makes the important point:

> Computers have been used to formalize and maintain traditional categories, including binary gender distinctions, ever since computers were invented. … Understanding how computer programs categorize, and how to build new and different systems, can have positive social potential when a classification issue like this one is involved. (Montfort, 2021, pp. 107–108)

Perhaps in the third edition, a wider audience can be assumed, and, even better, the book could be offered in fully open format, ripe for remixing by teachers and students who can insert their own cultural references, slang, and free project prompts.