

Work in Progress: A Seamless, Customizable e-Book for Introductory Programming Courses

Prof. Petra Bonfert-Taylor, Dartmouth College

Petra Bonfert-Taylor is the Associate Dean for Diversity and Inclusion and a Professor and Instructional Designer at the Thayer School of Engineering at Dartmouth College. She received her Ph.D. in Mathematics from Technical University of Berlin (Germany) in 1996 and subsequently spent three years as a postdoctoral fellow at the University of Michigan before accepting a tenure-track position in the Mathematics Department at Wesleyan University. She left Wesleyan as a tenured full professor in 2015 for her current position at Dartmouth College. Petra has published extensively and lectured widely to national and international audiences. Her work has been recognized by the National Science Foundation with numerous research grants. She is equally passionate about her teaching and has recently designed and created a seven-MOOC Professional Certificate on C-programming for edX for which her team won the "2019 edX Prize for Exceptional Contributions in Online Teaching and Learning". Previously she designed a MOOC "Analysis of a Complex Kind" on Coursera. The recipient of the New Hampshire High Tech Council 2018 Tech Teacher of the Year Award, the Binswanger Prize for Excellence in Teaching at Wesleyan University and the Excellence in Teaching Award at the Thayer School of Engineering, Petra has a strong interest in broadening access to high-quality higher education and pedagogical innovations that aid in providing equal opportunities to students from all backgrounds.

Mr. Simon Ethan Oster, Dartmouth College

Work in Progress: A seamless customizable e-book for an introductory programming course

Abstract

Introductory computer programming courses are essential in providing students with marketable skills, but in a fast-paced course, the transition into them can be daunting to new coders. This project describes the creation of a live e-textbook for use in Thayer School of Engineering's ENGS 20: Introduction to Scientific Computing course at Dartmouth College, an introductory computer programming course in C and MATLAB taken by all engineering students. Historically, students must access a number of sites to find various course materials, such as coding environments, course notes, lecture videos, and assignments; but without a central repository for all course materials, students often waste time searching through them and become frustrated while trying to complete their assignments. This live e-textbook combines all these course materials, including live coding windows, into a unified source where students can move through the chapters in a front-to-back-cover manner that mimics traditional textbooks and access all the necessary resources in one place. The live e-textbook can be instantly exported into a PDF version, which still compiles all the course materials into one place. While interactive content, such as videos and coding windows, appears in an alternate static version, the textbook provides a QR code for each such instance, with which students can easily access the live content if desired. The platform used to create the e-textbook is PreTeXt, which is an open-source XML-based markup language that combines the best of LaTeX and HTML, and that can output, amongst others, both web and PDF content. The e-textbook is hosted on an open-source Runestone Interactive server, which enables LTI integration to the course's existing Learning Management System (Canvas) and provides additional interactive functionality to the students and instructors.

Introduction

The value of computer programming skills continues to surge in a world that relies so heavily on technology. Recognizing the marketability of coding skills in the workforce, students of many disciplines now desire such knowledge. Introductory computer programming courses are thus becoming increasingly essential, but due to the fast-paced nature of college courses, the learning curve can be daunting to new coders. As a result, mastering one's first programming language can seem like an insurmountable task. Add in the need to install necessary coding environments and to purchase expensive textbooks, and students often start out already feeling overwhelmed.

Course-specific materials, such as online coding environments and explanatory notes, can help reduce the stress on students when entering an introductory coding course. With pedagogical innovation, a wider variety of course materials have come into fruition, each aiming to support students in a specific way. However, the combination of several such aids can become overwhelming for students, as it becomes increasingly difficult to find and keep track of all these individual resources. In just one course, for example, students may need to access online coding environments, course notes, lecture videos, in-class responses systems and homework assignments in different platforms all at the same time. This requirement may lead students to become lost navigating between various websites and handouts, thus, despite the availability of improved individual resources, the daunting work of learning to code becomes marred by the lack of cohesiveness amongst the course resources.

By providing students with a single source for all course materials, introductory computer programming courses will pose fewer barriers-to-entry. The problem is that no computer programming textbook exists that cohesively provides students with all their course materials in one central location.

Motivation

This paper describes the creation and use of a live online textbook which combines a variety of interactive course materials into a unified source for students of *ENGS 20: Introduction to Scientific Computing*, a prerequisite course for all engineering students at Dartmouth's Thayer School of Engineering. Traditional course textbooks contain explanatory notes and practice problems; this live textbook additionally incorporates coding environments, course notes, lecture videos, in-class activities, in-class responses and homework assignments. While students enjoy the maximal benefit from using the fully online version of the book, another objective of this project is to allow for the live textbook to be instantly converted into a complete PDF textbook. Key to this idea is the fact that the textbook's source code (written in XML) can be translated into both HTML for the online version or (via LaTeX) into PDF for the static version. The existence of a PDF option is crucial both to accommodate students who prefer a physical textbook as well as those who do not have consistent internet access. Since online-only content such as coding environments cannot be printed to PDF, innovations were needed to automatically include alternate versions of these features in the offline version of the textbook.

To engage with students through the live textbook, another key goal of the project is to enable easy access for the instructor to view student submissions in real-time. For instance, if students are working on in-class activities, it is beneficial for the instructor to instantly access their submissions to gauge overall class progress.

With a live textbook, students will be able to focus their attention on one source of material that provides them with all the necessary information for the course. This saves time and reduces frustration for students as well as reducing the number of course material-based questions posed to the instructors and teaching assistants, freeing up their time for deeper coding questions. This more efficient use of time and resources will ultimately reduce the unnecessary obstacles in the way of students seeking to gain introductory computer programming skills.

Methodology

We chose PreTeXt as the platform upon which to build our live e-textbook. PreTeXt is open-source software that turns user-created XML source code into various formats, including HTML and PDF. The ability to easily produce both an online and offline textbook provides flexibility to students in choosing how to consume the course content. While the source of the book is written in XML, the way it is displayed is controlled by XSL style sheets (similarly to how the appearance of an HTML document is controlled by CSS style sheets). This mechanism allows the author to build and style new environments for the textbook with differing instructions for the interactive versus the static version. The live textbook is hosted on a Runestone Interactive server, which provides some of the interactive in-class response features.

Coding Environments

For the purpose of this introductory coding course we have developed a server-based online coding environment in which students practice coding and complete activities as well as homework problems. To access these code windows, students authenticate using their single sign-on and can then code directly in a web browser without having to navigate the underlying server architecture. This online environment saves students from having to struggle through installing compilers and IDEs, or worse, having to log into a server in order to use command-line tools there. These coding windows are directly embedded into the live textbook via special XSL styling instructions. Each coding assignment in the book comes with its own unique coding window which often provides initial code to help students get started. Furthermore, each such window retains the student's typing history so that a student can leave and return to a coding window at any time. Moreover, the instructor can add automatic grading features to a coding window in order to provide instant feedback to students. Figure 1 shows a snapshot of one page of the live textbook, which includes an embedded code window.

The screenshot shows a web interface for a textbook. At the top, it says "ENGS 20: Introduction to C-Programming" by Petra Bonfert-Taylor. On the left is a navigation menu with sections like "A First C-Program: Structure, Printing, Repetition" and "Variables in C". The main content area is titled "3.1 Hello, World!". It contains a paragraph explaining that a Hello World program displays "Hello, World!" to the user. Below the text is a code editor window with the following C code:

```

1 #include <stdio.h>
2
3 int main(void) {
4
5     printf("Hello, World!");
6
7     return(0);
8 }

```

There are "Start Fresh" and "Run" buttons at the bottom right of the code editor. A link "admin open in new window" is also visible above the code.

Figure 1: Live textbook page with embedded code window

Alternatively, Figure 2 depicts what the same content shown in Figure 1 looks like in the static PDF version of the textbook. In this version, any pre-populated code in the coding windows is automatically printed in a special text environment with syntax highlighting. Additionally, a QR code appears next to each static code environment, and scanning the QR code will bring students to that specific coding window online. In this case, for example, students consuming the static content can decide at any time to practice in the live version of the book.

The screenshot shows a static PDF-like layout for the "3.1 Hello, World!" section. It includes the same explanatory paragraph as Figure 1. Below the text, the C code is displayed in a monospaced font with syntax highlighting. To the right of the code is a square QR code.

```

#include <stdio.h>

int main(void) {

    printf("Hello, World!");

    return(0);
}

```

Figure 2: Static textbook section with pre-populated code and QR code

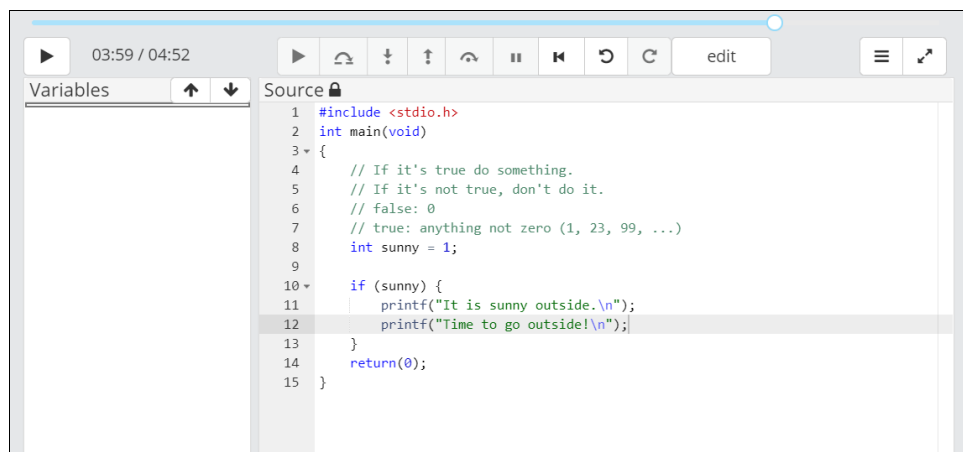
The code windows collect keystroke-by-keystroke data in order to provide valuable information to the instructor and teaching assistants. For example, the instructor can easily access a status report during class to find out how many students have solved a problem, how many are still encountering syntax errors, and so on. The instructor interface further enables the teaching staff to access student code in cases where students need assistance with their code. From this administrative side, the teaching staff can directly edit the code in each student's code window,

which makes providing virtual assistance much smoother. Prior to the existence of a unified e-textbook, these code windows were scattered throughout different pages of the course's learning management system and the in-class response system, which made it cumbersome for students to find any particular previous coding work. By embedding all code windows into a cohesive textbook with a clear table of contents, accessing the different code windows is straightforward for all.

In order to transition to the 'real world', halfway through the course students are introduced to directly working on Thayer School of Engineering's Linux servers, which entails learning some rudimentary Linux commands. Just as in the case of coding environments, there are embedded server windows in the e-textbook which allow students to use their single sign-on to practice logging into the course server directly from the relevant page of the textbook. This simple method enables students to efficiently access the Linux course server without the hassle of installing a terminal program and yet again having to click between different resources.

Interactive Coding Videos

In *ENGS 20: Introduction to Scientific Computing*, students watch and interact with "Codecasts", which are interactive coding videos. At first glance, they appear to be regular videos with pause and play buttons, with a scroll bar to scrub through the video. However, upon pausing the video, students can edit the code they are seeing in the video and test out modifications for themselves. Figure 3 shows a sample Codecast with which students can interact.



The screenshot shows a video player interface with a progress bar at the top indicating 03:59 / 04:52. Below the progress bar is a control bar with play, pause, and other video controls, and an 'edit' button. The main content area is split into two panes: 'Variables' on the left and 'Source' on the right. The 'Source' pane displays C code with line numbers 1 through 15. The code is as follows:

```
1 #include <stdio.h>
2 int main(void)
3 {
4     // If it's true do something.
5     // If it's not true, don't do it.
6     // false: 0
7     // true: anything not zero (1, 23, 99, ...)
8     int sunny = 1;
9
10    if (sunny) {
11        printf("It is sunny outside.\n");
12        printf("Time to go outside!\n");
13    }
14    return(0);
15 }
```

Figure 3: Codecast, an interactive coding video

Again, these Codecasts are embedded directly into the live textbook with the help of a special XSL style guide. They often appear in the first few sections of each chapter, where new topics are introduced to students. Similar to the live coding environments, the interactive Codecasts have alternate static versions for the PDF book. Figure 4 shows the appearance of the same Codecast shown in Figure 3, but in static form.

9.1 if Statements

We are ready for a really important and powerful new concept: the ability to create computer instructions that are executed only if certain conditions are met. Let's take a look. The code below is from a Codecast video. Access the video by scanning the QR code.

```
#include <stdio.h>
int main(void)
{
    // If it's true do something.
    // If it's not true, don't do it.
    // false: 0
    // true: anything not zero (1, 23, 99, ...)
    int sunny = 1;

    if (sunny) {
        printf("It_is_sunny_outside.\n");
        printf("Time_to_go_outside!\n");
    }
    return(0);
}
```



Figure 4: Static Codecast in PDF version

The static version of a Codecast shows code from the final seconds of the video so that students only viewing the PDF version still can see and learn from the static code. Moreover, QR codes appear next to each static Codecast, so that students can scan the QR code and instantly watch the Codecast online, such as on a mobile device.

Activity and Homework Submissions

Beyond the ability for students to interact with the textbook material, instructors and teaching assistants can also interact with students through the textbook. Mainly, this interaction takes place via submission windows, where students can enter their code or text answers and submit them to the teaching staff. These live submissions allow the instructors to see student answers in real-time during class and use this information to gauge the class's progress. Two options exist for student submission in the live textbook: code submission and text submission. The code submission windows appear below any coding task where students are asked to submit their answer to the instructors. Figure 5 shows a code submission window, where some input has been entered and saved. For assignments where the sought answer is not code, there exist text submission boxes, as shown in Figure 6.

Activity 1.1.

How do you print "Hello, World!" in C?

admin open in new window

```
1 #include <stdio.h>
2
3 int main(void) {
4     printf("Hello, World!");
5     return(0);
6 }
```

Start Fresh Run

When you are done, please paste your code into the submission box below:

Submit

```
1 #include <stdio.h>
2
3 int main(void) {
4     printf("Hello, world!");
5     return(0);
6 }
```

Figure 5: Code submission box

What are the two different types of loops that we have learned about so far?

Save

Instructor's Feedback

You have not answered this question yet.

Figure 6: Text submission box

In both instances, the student answers submitted to these submission boxes can be viewed by the instructors and TAs in the same interface on Runestone. There, the teaching staff can provide comments and grades to the student submissions, which, in turn, directly appear in the student's live textbook.

Discussion

Introductory computer programming courses can be daunting to students who have not previously learned a coding language. Without consistency in course materials, students may find themselves struggling to find needed information. This cohesive textbook seeks to increase the efficiency with which students access course materials, as the textbook is a one-stop-shop for all such materials. With course notes, lecture videos, coding windows, homework assignments, and more all in one place, students can focus their efforts on learning the material instead of searching for the information they need. With both a live and an offline version of the textbook, students have the opportunity to access the same information in different ways, which gives them flexibility in how they approach their learning.

Since the textbook contains both in-class and out-of-class material, students benefit from the consistency in the medium through which course materials are delivered. For example, homework assignments—whether they are questions in response to a recently presented Codecast or coding tasks to practice newly acquired skills—are completely embedded in the textbook pages through interactive submission boxes. This layout removes the need for students to click through tabs or flip through handouts to find related notes. Additionally, student answers are saved in their e-textbook, allowing review of responses while studying as well as easy access to instructor feedback. Together, this cohesion streamlines the students' learning experience.

Instructors similarly benefit from the book's cohesiveness. Through the Runestone Interactive server, student responses from each submission box are collated and presented in an easy-to-use interface. In class, the instructor can watch live as students work through activities and submit their answers. Furthermore, it is easy for the instructor to access a particular student's code in order to present it to the class as a sample solution.

To better measure the effects of this live textbook on students, a comprehensive survey will be created and distributed to the students enrolled in the prior year's course offering (in which students did not yet benefit from the newly created e-textbook). Before completing the survey, the respondents will be asked to view the live textbook and explore its features. Some of the questions included in the survey will ask students to rate statements, such as:

1. *"It is easy to navigate through this textbook to find course material."*
2. *"The embedded code windows encourage me to try out code."*
3. *"This textbook would have improved my overall course experience."*

From the responses to the survey, it will be determined whether previous students expect this live textbook to be useful to future students in the course. Metrics derived from the survey results will signal what the general student view of this textbook is, thereby informing the need for any changes ahead of its implementation. Further surveys will be deployed to students using the actual e-textbook in class.

Further Research

Based on the results of 225 studies on active learning analyzed by Freeman and others, active learning in the classroom has been shown to increase overall student performance in examinations, leading to higher average grades compared to traditional lecture classes [1]. Therefore, one could expect similar improvements in overall student performance in a coding course through the integration of a live textbook, where students can actively practice their coding, both during and outside of lecture. It can be hypothesized that, if this live textbook is implemented in the relevant introductory coding course, then one could expect to see improvements in overall student performance. Following the next course offering when this live textbook is implemented, analysis can be conducted to compare student performance with prior course offerings.

Conclusion

This work-in-progress paper introduces a seamless, customizable online textbook for use in an introductory programming course. The textbook is created on PreTeXt's open source platform.

The innovative aspects, including embedded SSO-controlled coding windows and interactive Codecast coding videos, are presented and demonstrated. Additional features of the book, such as its ability to be instantiated in both a live HTML version and a static PDF version, are highlighted. By hosting the HTML version on an open-source Runestone Interactive server, instructors enjoy increased functionality, including the ability to collect student responses in-book and provide integrated feedback through the platform. Data will be collected on the efficacy of the live textbook on student experience in the course. It is expected that this innovative textbook will increase student efficiency and improve overall experience in the introductory coding course.

References

- [1] S. Freeman et al., “Active Learning Increases Student Performance in Science, Engineering, and Mathematics.” *Proceedings of the National Academy of Sciences*, vol. 111, no. 23, 2014, pp. 8410–8415., doi:10.1073/pnas.1319030111.