Runestone: An Open-Source Platform for Interactive Ebooks

Barbara J. Ericson
University of Michigan, School of Information
Ann Arbor, Michigan, USA
barbarer@umich.edu

Bradley N. Miller
Luther college / Founder Runestone Interactive
Minneapolis, Minnesota, USA
brad@runestoneinteractive.com

ABSTRACT

The Runestone platform is an open-source platform for interactive ebooks. It served over 100,000 registered learners and an average of 800 thousand page views per week during the 2019 - 2020 academic year. There are ebooks for secondary computer science as well as for undergraduate computing courses: CS1, CS2, data science, and web programming. There is even an ebook for a course on Linear Algebra.

The platform supports executable and editable examples in Python, Java, C, C++, HTML, JavaScript, Processing, and SQL. It also includes code visualizers/steppers for Python, Java, and C++ code [4]. The ebooks contain typical instructional material: text, videos, and images. They also include practice problems with immediate feedback such as multiple-choice, fill-in-the-blank, and matching questions. Runestone also has unusual features such as audio tours of code, clickable areas, adaptive Parsons problems, and a unique practice tool. This paper highlights some of the unusual features, explains the log data that is collected and is available for analysis, and describes plans for future development.

Author Keywords

on-line learning, ebooks, adaptive learning, practice tools, intelligent ebooks, Parsons problems, ACOS, LTI

INTRODUCTION

Brad Miller started creating the Runestone ebook platform in 2011. He wanted to have ebooks with embedded code that the learner could run and edit. Brad also wanted ebooks to be free so that everyone has the chance to learn computer science, even if they can not afford a $200 textbook. Brad's goal was *"democratizing textbooks for the 21st century"*.

UNUSUAL INTERACTIVE FEATURES

Runestone has several unusual interactive features. These include audio tours of code, clickable area problems, adaptive Parsons problems, and a practice tool.

Sixth SPLICE Workshop at L@S 2020 "Building an Infrastructure for Computer Science Education Research and Practice at Scale", SPLICE'20, August 12, 2020, Virtual Event

In audio tours, lines of code are highlighted as audio plays that explains those lines. This feature takes advantage of the fact that humans can process both audio and visual information at the same time [6, 7]. Teachers have reported that they find audio tours useful since they model how to describe code [3]. However, most users do not listen to them [3].

A user answers a question by clicking on code or text in a clickable area problem. These questions can be used to click on parts of a program, such as the variable declarations. They can also be used to click on items at specific indices in a list or on words that indicate the required variable type. A log file analysis has shown that most users attempt to solve these problems.

In a Parsons problem, the learner places code blocks in the correct order to solve a problem [8]. Parsons problems can have extra blocks, called distractors, that are not needed in a correct solution. In an adaptive Parsons problem the difficulty of a problem is based on the learner's performance. Runestone supports two types of adaptation for Parsons problems: intra-problem and inter-problem. In intra-problem adaptation if the user is struggling to solve the current Parsons problem it can dynamically be made easier by removing distractors from the solution or by combining code blocks. In inter-problem adaptation the difficulty of the next problem is modified based on the learner's performance on the last problem. The goal is to keep the learner in Vygotsky's Zone of Proximal Development where the learner is challenged, but not frustrated [9]. Log file analysis has shown that most users attempt to solve Parsons problems [3] and that users are nearly twice as likely to correctly solve adaptive Parsons problems than non-adaptive ones [2].

The practice tool in Runestone can be setup to provide students points for correctly solving a specified number of problems from an ebook each day for up to a maximum number of days [10]. An algorithm selects a question on a topic when it predicts the student is about to forget that topic.

LOG DATA

Runestone logs all user interaction in the ebook. Each log entry includes the date and time (timestamp), user identifier (sid), event, data about the event (act), item identifier (div_id), course identifier (course_id), base course identifier (base_course), chapter, subchapter, and institution identifier (anon_institution). It logs page views, answers to any of the practice problems, video plays, audio tour plays, and every move of a block in a Parsons problem. Part of a log file is shown in Figure 1. All of the code that is executed or saved is also available in a separate log.

<table>
<tr><th>timestamp</th><th>sid</th><th>event</th><th>act</th><th>div_id</th></tr>
<tr><td>8/21/17 23:54</td><td>27</td><td>Audio</td><td>Line-by-line Tour</td><td>lcfc1</td></tr>
<tr><td>8/21/17 23:54</td><td>27</td><td>Audio</td><td>play</td><td>lcfc1</td></tr>
<tr><td>8/21/17 23:54</td><td>27</td><td>Audio</td><td>closeWindow</td><td>lcfc1</td></tr>
<tr><td>8/21/17 23:57</td><td>23</td><td>video</td><td>play</td><td>ants</td></tr>
<tr><td>8/21/17 23:57</td><td>25</td><td>mChoice</td><td>answer:3:correct</td><td>q2_2_1</td></tr>
<tr><td>8/21/17 23:57</td><td>25</td><td>mChoice</td><td>answer:1:correct</td><td>q2_2_2</td></tr>
<tr><td>8/21/17 23:58</td><td>25</td><td>mChoice</td><td>answer:3:no</td><td>q2_2_3</td></tr>
<tr><td>8/21/17 23:59</td><td>25</td><td>mChoice</td><td>answer:0:correct</td><td>q2_2_3</td></tr>
<tr><td>8/22/17 0:02</td><td>26</td><td>page</td><td>view</td><td>index.html</td></tr>
<tr><td>8/22/17 0:03</td><td>26</td><td>page</td><td>view</td><td>JavaBasics/\</td></tr>
<tr><td>8/22/17 0:03</td><td>25</td><td>activecode</td><td>edit</td><td>lcfc1</td></tr>
<tr><td>8/22/17 0:04</td><td>25</td><td>parsonsMove</td><td>start|0_1_0-5_0-6_0-2_3_0-4_0|-|c0</td><td>thirdClass</td></tr>
<tr><td>8/22/17 0:04</td><td>25</td><td>parsonsMove</td><td>move|5_0-6_0-2_3_0-4_0|0_1_0|c0</td><td>thirdClass</td></tr>
<tr><td>8/22/17 0:04</td><td>25</td><td>parsonsMove</td><td>move|5_0-6_0-4_0|0_1_0-2_3_0|c0</td><td>thirdClass</td></tr>
<tr><td>8/22/17 0:04</td><td>25</td><td>parsonsMove</td><td>move|5_0-6_0|0_1_0-2_3_0-4_0|c0</td><td>thirdClass</td></tr>
<tr><td>8/22/17 0:04</td><td>25</td><td>parsonsMove</td><td>move|6_0|0_1_0-2_3_0-4_0-5_0|c0</td><td>thirdClass</td></tr>
<tr><td>8/22/17 0:04</td><td>25</td><td>parsonsMove</td><td>move|-|0_1_0-2_3_0-4_0-5_0-6_0|c0</td><td>thirdClass</td></tr>
<tr><td>8/22/17 0:04</td><td>25</td><td>parsons</td><td>correct|-|0_1_0-2_3_0-4_0-5_0-6_0|c1-s</td><td>thirdClass</td></tr>
</table>

Figure 1. Part of a log file from Runestone that has been anonymized.

Researchers can request an anonymous log file from Brad Miller. In this file user and institution identifiers will be replaced with numbers. Instructors who create a custom course can download a log file directly from the instructor interface.

FUTURE WORK

Our plans for future work include LTI integration, ACOS support, randomized exams, templated questions, support for Peer Instruction, and new research on Parsons problems.

While Runestone has had some support for integration with Canvas, Miller is adding support for LTI/ACOS to allow other researchers access to the problems in Runestone ebooks.

We are also adding support for randomized exams where Runestone will pick questions at random from a set of equivalent questions in a question pool. This will make it harder for students to cheat on exams with other students.

We plan to integrate support for both in-person and remote Peer Instruction. Peer Instruction was originally developed by Eric Mazur to improve student understanding in physics [1]. In Mazur's Peer Instruction students read material before lecture and take an assessment based on the reading either before or at the beginning of lecture [1]. In lecture the instructor displays a difficult multiple-choice question that contains distractors (incorrect answers) based on common misconceptions. The students answer the question individually (vote), then discuss their answers with neighboring students (peers), and then answer (vote) individually again. Finally, the instructor shows the result of the two votes and leads a discussion of the question [1].

We plan to use log file analysis to determine the difficulty of multiple-choice questions in the free ebooks in order to create a pool of good questions for Peer Instruction. A good question for Peer Instruction is one that about 40-60% of the students get wrong on the first vote [5]. We will also modify Runestone to make it easy to find, serve, author, and modify Peer Instruction questions in an ebook. We also plan on adding questions from public question banks (Peer Instruction for Computer Science and the Canterbury Question Bank) to Runestone as well.

While most students enjoy solving Parsons problems, some would rather write the code themselves. We plan on providing an option on Parsons problems to allow students to choose to solve the equivalent write code problem with unit tests instead. In addition, we want to scaffold students who are having problems solving write code problems with unit tests by providing them with a similar Parsons problem to their current solution.

REFERENCES

[1] Catherine H Crouch and Eric Mazur. 2001. Peer instruction: Ten years of experience and results. *American journal of physics* 69, 9 (2001), 970–977.

[2] Barbara Ericson, Austin McCall, and Kathryn Cunningham. 2019. Investigating the Affect and Effect of Adaptive Parsons Problems. In *Proceedings of the 19th Koli Calling International Conference on Computing Education Research*. 1–10.

[3] Barbara J Ericson, Kantwon Rogers, Miranda Parker, Briana Morrison, and Mark Guzdial. 2016. Identifying design principles for CS teacher Ebooks through design-based research. In *Proceedings of the 2016 ACM Conference on International Computing Education Research*. 191–200.

[4] Philip J Guo. 2013. Online python tutor: embeddable web-based program visualization for cs education. In *Proceeding of the 44th ACM technical symposium on Computer science education*. 579–584.

[5] Nancy Kober. 2015. *Reaching students: What research says about effective instruction in undergraduate science and engineering.* National Academies Press.

[6] Richard E Mayer. 2008. Applying the science of learning: Evidence-based principles for the design of multimedia instruction. *American psychologist* 63, 8 (2008), 760.

[7] Richard E Mayer and Roxana Moreno. 1998. A split-attention effect in multimedia learning: Evidence for dual processing systems in working memory. *Journal of educational psychology* 90, 2 (1998), 312.

[8] Dale Parsons and Patricia Haden. 2006. Parson's programming puzzles: a fun and effective learning tool for first programming courses. In *Proceedings of the 8th Australasian Conference on Computing Education-Volume 52*. 157–163.

[9] Lev Semenovich Vygotsky. 1980. *Mind in society: The development of higher psychological processes*. Harvard university press.

[10] Iman YeckehZaare, Paul Resnick, and Barbara Ericson. 2019. A Spaced, Interleaved Retrieval Practice Tool that is Motivating and Effective. In *Proceedings of the 2019 ACM Conference on International Computing Education Research*. 71–79.