Interactive Web Activities for Online STEM Learning Materials

Alex Edgcomb and Frank Vahid
Department of Computer Science and Engineering
University of California, Riverside

Abstract

We are developing a repository of parameterized interactive web activities to aid in learning STEM (Science, Technology, Engineering, and Math) concepts. Much web-based material today, including online textbooks, online tutorials, and MOOCs (massive open online courses), include quiz-like activities to support interaction with the user. Varied customizable interactive activities, such as drag-and-drop definition matching, or shooting or navigation games driven by quiz-like questions, are provided for free on various sites or for a fee by commercial firms. Quiz-like activities merely scratch the surface of the power of web-based learning. We have found that learning STEM concepts requires more-specific activities that allow for exploration and tinkering-with a concept to support bottom-up learning, such as a tool that allows tinkering with a binary-to-decimal converter or an equation plotter. Such tools can be developed for HTML5 via custom Javascript and CSS programs. Our goal is to create a repository of parameterized customizable activities that authors can use without requiring Javascript/CSS expertise. We have developed several activities, all in HTML5, originally for introductory computer programming concepts. We discuss how those activities can be generalized and thus be made applicable to a wider variety of STEM topics, such as math, physics, or chemistry. Our goal is to create a repository of approximately 100 STEM-focused activities that web-based material authors can instantiate to create powerful web-based learning materials.

Introduction

STEM (Science, Technology, Engineering, and Mathematics) disciplines tend to involve challenging concepts, the learning of which can be enhanced by students performing activities related to the concepts – learning by doing. Lacking the ability for interactivity, traditional textbooks resort to longer explanations and series of drawings. Porting existing textbooks to electronic formats lowers costs and may increase access modes but does not capitalize on the web's potential for interactivity. Supplemental interactive activities has been done but may increase the burden on the student if not accompanied by decreases in excessively-large textbooks, class notes or Powerpoints, and other materials. Carefully planned interactive web activities can potentially decrease the need for lengthy written materials and thus improve learning.

This paper describes several types of interactive web activities developed for an introduction to programming course, namely binary-to-decimal converter, interactive inheritance tree, equation plotter, swap sorter, and quick sorter. The eventual goal is to create many tens of such activities, parameterized so that they can be reused across STEM disciplines.

Background

STEM students prefer to experience specific cases of a concept, such as sorting a list of random numbers, then work up to a general principle[1,4,5,6], such as an algorithm for sorting any list of numbers. Interactive activities provide a student with specific cases of a concept that can be explored and tinkered-with to both learn a concept for the first time and refresh the understanding of a previously learned concept.

Interactive software has been shown by researchers to improve student outcomes in engineering[25], math[8], and science[2]. Wood[25] built an interactive program for engineering education that allowed a student to manipulate basic engineering math equations representing signal filters and simultaneously see the affects on many perspectives of the equation. ALEKS[8] is a web-based interactive software for enhancing college algebra education that significantly improved the student exam performance in college algebra courses. Broschat[2] developed interactive software that allowed a student to manipulate electromagnetism equations and visualize in 3-D the shape of the electromagnetic forces, such as electric potential and the magnitude of the electric potential. Interactive software has also been shown to improve education with adults. Shaw[23] evaluated patient colonoscopy education by the overall comprehension and satisfaction of the colonoscopy patients, comparing traditional education and traditional education plus interactive software. Shaw found significant improvements in the overall comprehension and satisfaction when using interactive software.

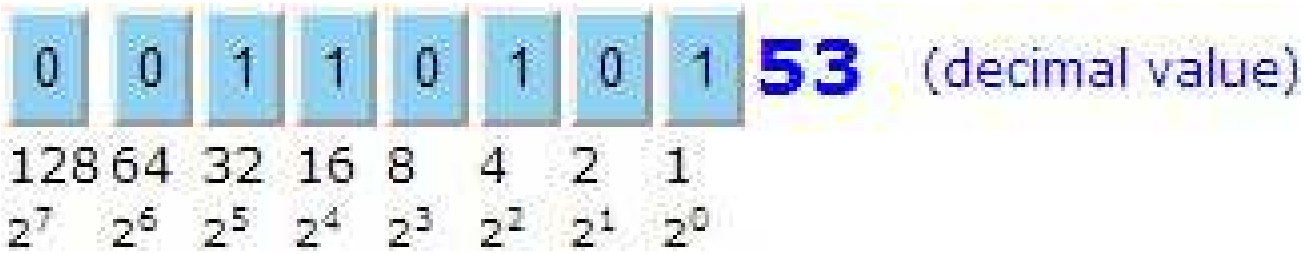
Interactive web activities have been built to explain STEM-specific concepts, including data structures[7,10,13] and algorithms[17] for Computer Science undergraduates. Kloss[13] built a tool that displays a binary tree of numbers from which a student can insert a number, delete a number, and search for a number. Mukundan[17] built an activity that graphically and textually shows the difference between searching a graph via depth-first and breadth-first. The student builds the graph by clicking on any two nodes to create an edge between the two nodes.

Literally hundreds of tools support development of learning content, such as Kenexa[12], LearnerWeb Cap[14], Microsoft Learning Development System (LCDS)[16], ReadyGo[22], and ToolBook[24], to name just a very few. Many such tools are highly-polished and target creation of corporate training materials. Extensive support is provided for creating quizzes, tree-structured content, and animations.

Web-based authoring tools have also been used to create educational content. For example, Inkling Habitat[9] and Lectora Online[15] support web-based integration of existing content like text, images, and videos. Some allow creation of games and other activities, e.g., ClassTools[3] and Raptivity[21]. Various open-source tools exist to assist[18].

Raptivity[21] is software to build interactive web activities from customizable, general-purpose template activities. An example activity is a graphical book that can be loaded with pictures and text, in which the text can be automatically converted to audio or a voice recording can be played. Another activity is a graphical pyramid with a variable number of levels that describes a particular level with text and audio when that level is clicked. Since Raptivity's template

Figure 1: Binary-to-decimal converter activity allows students to toggle each bit of a binary number with a click and instantly show the decimal value.



activities are general enough for any discipline, the benefit to STEM-specific disciplines is limited, particularly in the way of feedback to the student.

Interactive Web Activities

Interactive web activities teach a concept via exploring and tinkering. The interactive elements are visually easy to identify using standard web interactive elements, such as shadowed buttons. The activities give instant feedback to the student allowing for rapid building and refinement of understanding. Web-based learning materials can consist of fewer words and more activities.

The following subsections describe initial interactive web activities that we originally developed for introductory computer programming material in C[19] and C++[20]. Also discussed are possible expansions to other STEM disciplines.

Binary-to-decimal converter

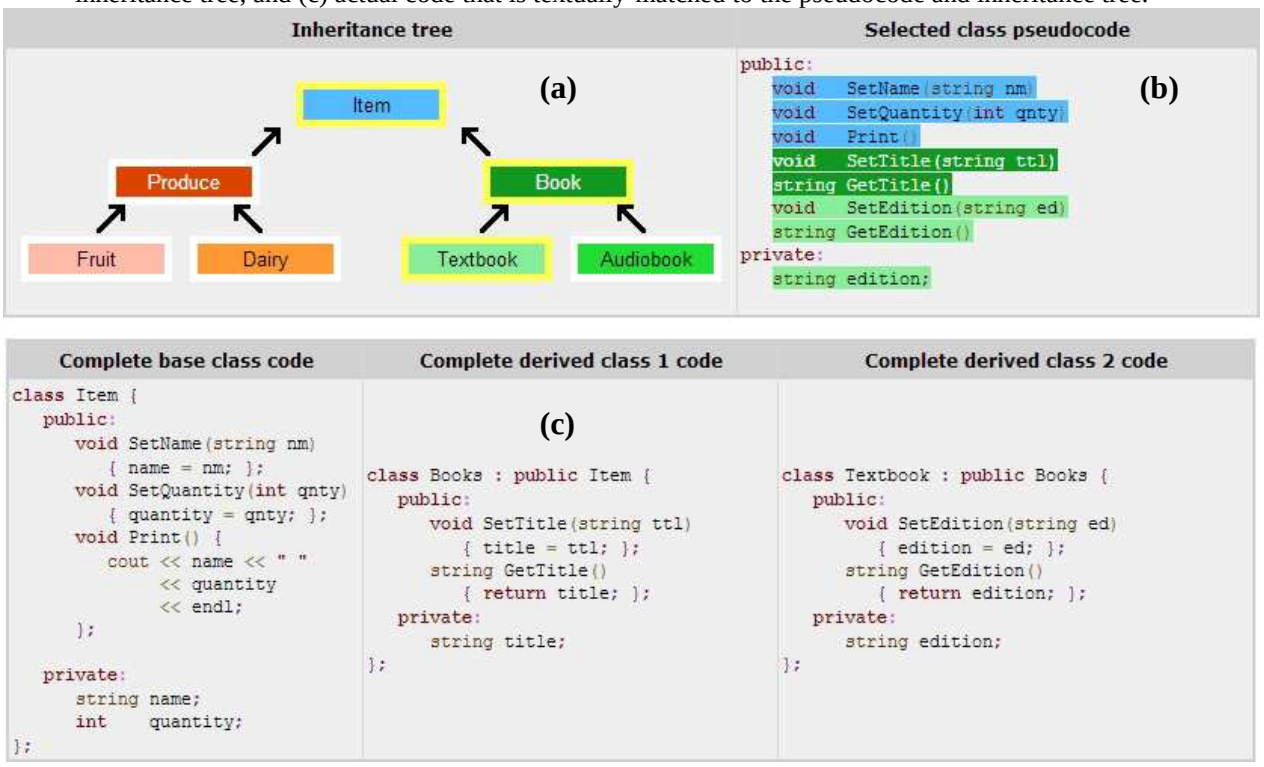
The binary-to-decimal converter activity teaches a fundamental concept in Computer Science that can also be applied to teach any number system. As shown in Figure 1, the activity contains 8 bits, which interact as buttons. The sum of the bits is a decimal value shown on the right. Below each bit is that bit's value as a decimal number. Below the bit's value as a decimal number is the bit's value as a power of 2.

When a student clicks a bit, the bit instantly toggles (from "0" to "1", or "1" to "0"), and the decimal value is instantly updated to the appropriate value. This activity quickly teaches a student how binary works and is a useful quick reference converting a binary number to decimal.

We have described binary-to-decimal numbers in a traditional textbook, requiring several pages. The web-based material instead consists of a single paragraph and the tool. Our initial experience is that the tool enables students to construct their own understanding of how each digit contributes to the decimal amount. Anecdotally, we have had students as young as 14 years old experiment with the tool and within minutes they understand the concept.

A similar tool, wherein buttons are pressed to include or exclude an item, could potentially assist with other STEM topics. The tool could be parameterized to allow each button to toggle between include and exclude states (rather than 0 or 1), or could allow selection of any number (e.g., 0-99). Each digit can be associated with different values, rather than 1, 2, 4, 8, etc. Buttons could be inserted anywhere within a string of text. One example usage would be in understanding how

Figure 2: Interactive inheritance tree activity contains (a) an inheritance tree, (b) pseudocode that is color-matched to the inheritance tree, and (c) actual code that is textually-matched to the pseudocode and inheritance tree.



a polynomial is impacted by constants for a given x value; the user could click on constants, e.g., y = _x^2 + _x + _, where each _ is a button whose value is selectable. For a given x (which could be modified by the user), the user can vary the values and see the difference in output value. Alternatively, buttons could represent the resistance of a parallel circuit, and the result could represent the current for a given voltage.

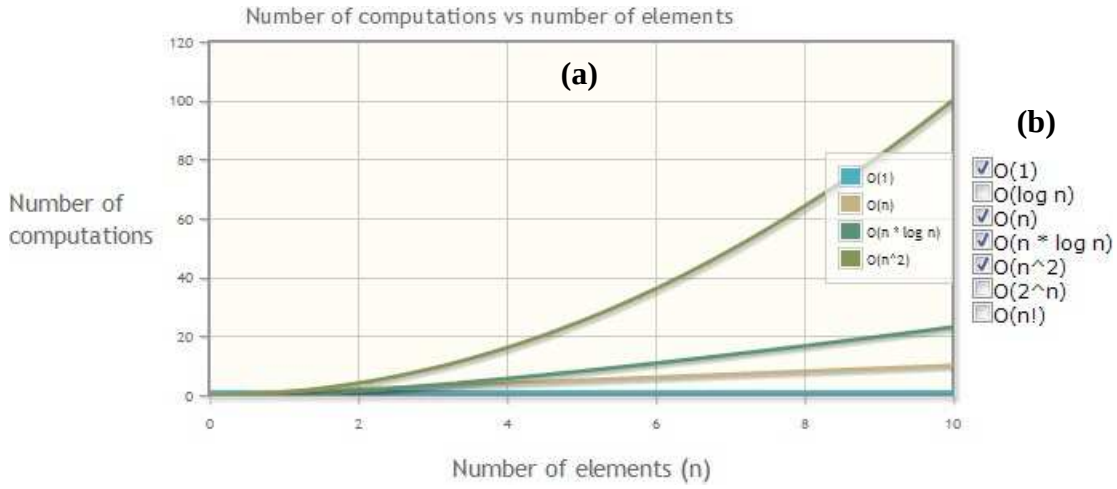
Interactive inheritance tree

Inheritance trees are used across many fields to show the relationship of objects. The interactive inheritance tree activity teaches the concept of inheritance in three ways: entirely visually, partially visually and partially textually, and entirely textually. The three ways show the same idea in different contexts. In Computer Science, a class, which consists of data and functions, may inherit data and/or functions from another class.

The activity includes an inheritance tree of classes, shown in Figure 2(a), that is entirely visual. The inheritance tree has arrows pointing from one class to the inherited class. When a user clicks on a class in the tree, that class and all the classes which are inherited become highlighted, thus showing the lineage of inheritance. Each class' background color is distinct yet similar to the inherited class' background color, e.g., the class "Textbook" is light green while "Book" is dark green.

The activity also includes pseudocode, shown in Figure 2(b), that is partially visual and partially textual. To the right of the tree, the clicked class' data and functions are shown as pseudocode, including the inherited data and functions. Each data and function is colored with the distinct

Figure 3: Equation plotter activity contains (a) a graph and (b) check-boxes for equations.



background color of the originating class, e.g., the function "SetEdition" is colored the same as the class "Textbook", while the function "SetTitle" is colored the same "Book". This coloring makes associating data and functions with the original class fast and easy.

Lastly, the activity includes actual code, shown in Figure 2(c), that is entirely textual. Below the tree, the actual code is shown that describes the clicked class and the classes inherited by the clicked class. Each class' code is contained in a different column and each column has a header name describing the relationship of the classes.

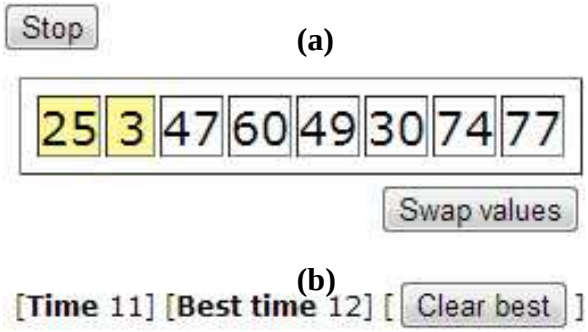
The parameterized version of the tool could allow for an arbitrary number of objects in the tree and with an arbitrarily number of connections, including multiple inheritance. Also, each object could be represented by a picture, text, or other chosen media. The color-coded information to the right of the tree would be optional and, if chosen, automatically generated based on the selected object and that object's inheritance. A pre-set list of rules for automatically generating the color-coded information would be built based on needs from other disciplines. The entirely textual area would be optional and be capable of a chosen media.

Equation plotter

Equation plotting and comparing are common activities in engineering education. The equation plotter activity visually shows the relative difference between equations plotting the equations on the same graph. The equations shown can be instantly changed by a student. The equations in Figure 3 are common computational complexities in Computer Science.

The equation plotter activity, shown in Figure 3, contains a graph (generated using jqPlot[11]) and check-boxes for equations. The graph shows an equation when the respective check-box is checked. Otherwise, the equation is not shown. The y-axis of the graph is updated to optimize the viewing of the equations, which makes seeing the difference between equations clear. In particular, all of the equations are entirely shown and the equations dictate the range of the y-axis. The y-axis minimum and maximum values are updated based on the equations that are checked, e.g., if only O(1) and O(n^2) were checked and the x-axis maximum is 10, then O(n^2)

Figure 4: Swap sorter activity intuitively teaches structured sorting of numbers, from smallest on the left to largest on the right. (a) A student can swap two numbers by highlighting the two numbers then clicking "Swap values". (b) The current playtime and best sorting time are shown in seconds.



would produce the largest y-value at 100 so the y-axis range maximum would be at least 100. The y-axis minimum and maximum range also stay relatively close to the smallest and largest equation values, respectively. For example, in Figure 3, even though O(n^2) has the largest y-axis value of 100, the graph's maximum y-axis value is 120.

The equation plotter activity scales well with the number of equations since the student can control which equations are displayed. Also, when an equation with particularly large y-axis value causes the remaining equations to appear identical, such as the affect of O(n!) on the other equations in Figure 3, the student can remove the overbearing equation.

The activity can be parameterized to accept instructor and/or student-defined equations. Controls for zooming in and out of the graph are a potential parameter, as well as the option for alternative graph types including 3D and bar graphs. Further analytics of the equations could be included.

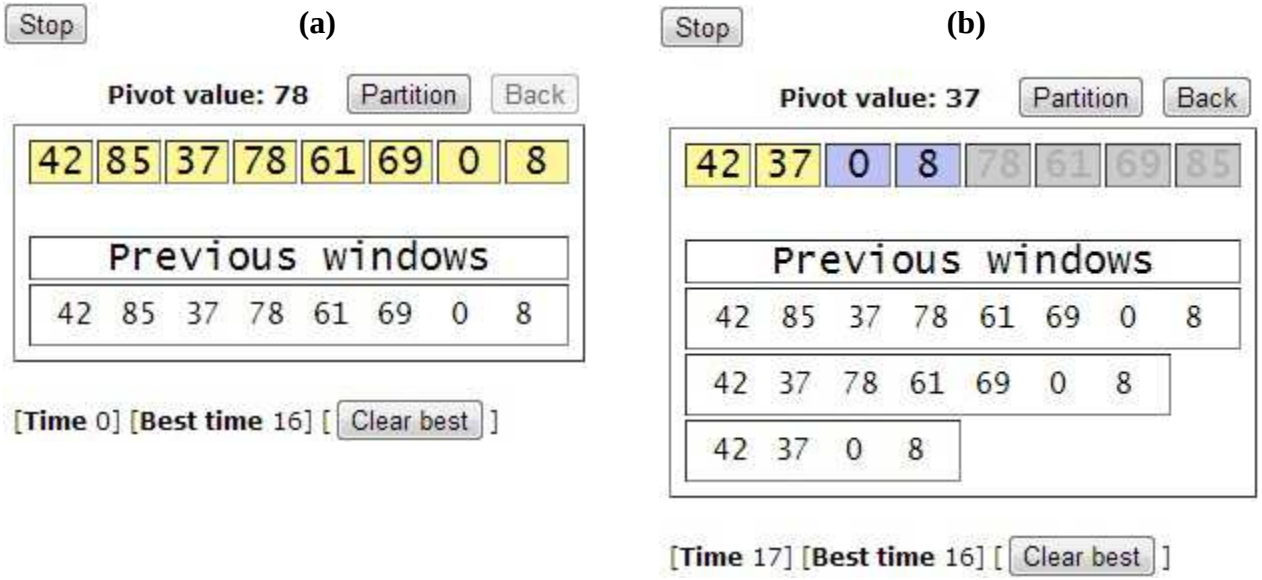
Swap sorter

The swap sorter activity introduces structured sorting of numbers, in this case from smallest to largest. The activity limits the student's controls to be analogous to instructions given to a computer via code. Thereby, intuitively teaching a student how to program a computer to sort numbers, which is a common task for novice and advanced programmers.

The list of numbers and sorting controls are shown in Figure 4(a). When a student clicks the "Start" button, which changes to the "Stop" button as shown in Figure 4(a), a list of randomly generated numbers appear. The student is instructed to sort the numbers by swapping values. Two values are swapped by highlighting the two values then clicking "Swap values". The swap sorter activity ends when the numbers are sorted from smallest on the left to largest on the right, or if the student clicks the "Stop" button. When the activity ends, the student can restart the swap sorter with a new list of randomly generated numbers.

The student is encouraged to play again to beat his or her best sorting time. The current playtime and best sorting time are shown in seconds, as shown in Figure 4(b). The best time is updated when the swap sorter activity ends if the current playtime is smaller than the best time. The best time can be cleared with the "Clear best" button.

Figure 5: Quick sorter activity intuitively teaches the quick sort algorithm. (a) The student must select all values in the current window of values that are smaller than the pivot value, then click the "Partition" button. Highlighted values are blue, while unhighlighted are yellow. (b) The previous windows are displayed to visualize the recursion.



A similar tool, in which a list can be interacted with by selecting elements in the list and manipulating the selected element with a button, could assist in other STEM disciplines. The tool could be parameterized to allow elements to be represented by any media, and could allow for the button affects on the list to be selected from a set of pre-loaded behaviors, including swap. One example usage is a parallel resistor simulator. Each element in the list represents a resistance. The simulator has a given input voltage and computes an output voltage based on the selected resistances when the student clicks the button.

Quick sorter

The quick sorter activity intuitively teaches the quick sort algorithm by constraining the student controls to be analogous to computer instructions via code. Quick sort partitions a window of values into two parts: (1) values smaller than the pivot value and (2) values greater than the pivot value. Then, repeats the partitioning on each part until each part has only one value. The pivot value is the value located at the middle of the current window.

When the student clicks the "Start" button, which changes to the "Stop" button shown in Figure 5(a), a list of randomly generated numbers appears, and the pivot value is automatically calculated and displayed. The student must highlight the values smaller than the pivot value, then click the "Partition" button. Highlighted values are blue. If a value is mistakenly highlighted or unhighlighted when "Partition is clicked, then a pop-up appears indicating the specific mistake. Otherwise, the partition proceeds with smaller values on the left and larger values the right. The larger values are grey and disabled, as shown in Figure 5(b), while the smaller values are further partitioned. Partitioning stops when the current window contains only one value.

The previous windows are displayed automatically and updated after each partition. The "Previous windows" hierarchically shows the recursive partitioning, which connects the idea of

partitioning a specific window with the recursive nature of quick sort. The "Back" button steps back one partition per click, which strengthens student's understanding of the specific window and recursive nature connection.

Due to the quick sorter activity having more complexity than the swap sorter, the amount of interactivity was extended to include the ability to step backwards, the ability to disable elements in the list, and a history of selectable elements in the list. These inclusions could be added as additional parameters to the parameterized swap sorter tool. One example usage is a limited-resistor allocation game in which the user must allocated a limited number of resistors to three circuits. Each circuit has a given input voltage and a required range of output voltages. The student must determine which combination of resistors to put in parallel for each circuit. The student solves the game by iteratively selecting which resistors will be applied to each circuit. The back button would be used to step-back to a previously applied circuit in the case that a needed resistor is being used by a previous circuit. The previous windows should show the resistor grouping for each circuit.

Conclusion

We described in detail 5 interactive activities, namely binary-to-decimal converter, interactive inheritance tree, equation plotter, swap sorter, and quick sorter. The activities have been implemented for the web and are already available for Computer Science education. Furthermore, we discussed the reuse of the activities for other engineering disciplines. Interactive activities benefit education by supplementing textual explanations with activities to explore and tinker-with, and allow for shorter worded explanations. STEM educators, who are not necessarily expert programmers, need to be enabled to develop interactive activities. Our approach is to build a repository of parameterized tools that allow for high-levels of customization for STEM topics. We plan to refine the repository by applying the parameterized tools across STEM education.

Bibliography

1. Bransford, J. How people learn: Brain, mind, experience, and school. National Academies Press, 2000.
2. Broschat, S.L. Interactive software for undergraduate electromagnetics. Education, IEEE Transactions, Volume 36, No. 1, pgs. 123-126, 1993.
3. ClassTools. http://www.classtools.net/. January 2013.
4. Felder, R.M. "Reaching the second tier." Journal of College Science Teaching, Volume 23, No. 5, pgs 286-290, 1993.
5. Felder, R.M., D.R. Woods, J.E. Stice and A. Rugarcia. "The future of engineering education II. Teaching methods that work." Chemical Engineering Education, Volume 34, No. 1, pgs 26-39, 2000.
6. Felder, R.M. and L.K. Silverman. "Learning and teaching styles in engineering education." Engineering education, Volume, 78, No. 7, Pgs 674-681, 1988.
7. Franco, J. http://www.ece.uc.edu/~franco/C321/html/RedBlack/redblack.html. January 2013.
8. Hagerty, G. and S. Smith. Using the web-based interactive software ALEKS to enhance college algebra. Mathematics and Computer Education, Volume 39, No. 3, pgs 183-194, 2005.
9. Inkling Habitat. https://www.inkling.com/habitat/. January 2013.
10. Jarc, D. J. http://nova.umuc.edu/~jarc/idsv/lesson1.html. January 2013.
11. jqPlot. http://www.jqplot.com/. January 2013.
12. Kenexa. http://www.outstart.com/outstart_lcms.htm. January 2013.

13. Kloss, J. http://www.cs.jhu.edu/~goodrich/dsa/trees/btree.html. January 2013.
14. LearnerWeb Cap. http://www.maxit.com/learn/. January 2013.
15. Lectora. http://lectora.com/online-e-learning/. January 2013.
16. Microsoft Learning Development System (LCDS). http://www.microsoft.com/learning/en/us/lcds-tool.aspx. January 2013.
17. Mukundan, R. http://www.cosc.canterbury.ac.nz/mukundan/dsal/GraphAppl.html. January 2013.
18. OpensourceLearning. http://blog.efrontlearning.net/2010/10/open-source-authoring-tools-for-e.html. January 2013.
19. Programming in C. http://progc.zyante.com/. January 2013.
20. Programming in C++. http://pcpp.zyante.com/. January 2013.
21. Raptivity. http://www.raptivity.com/. January 2013.
22. ReadyGo. http://readygo.com/. January 2013.
23. Shaw, M.J., T.J. Beebe, P.A. Tomshine, S.A. Adlis, O.W. Cass. A randomized, controlled trial of interactive, multimedia software for patient colonoscopy education. Journal of clinical gastroenterology, Volume 32, No. 2, pgs 142-147, 2001.
24. ToolBook. http://www.sumtotalsystems.com/products/toolbook-elearning-content.html. January 2013.
25. Wood, S.L. A new approach to interactive tutorial software for engineering education. Education, IEEE Transactions, Volume 39, No. 3, pgs 399-408, 1996.