Computer Science Illustrated: Engaging Visual Aids For Computer Science Education

Ketrina Yim
EECS Department
University of California, Berkeley
kyim@berkeley.edu

Daniel D. Garcia
EECS Department
University of California, Berkeley
777 Soda Hall
Berkeley, CA 94720-1776
+1 510-642-9595
ddgarcia@cs.berkeley.edu

Sally Ahn
CS Department
University of California, Berkeley
sallyahn@berkeley.edu

ABSTRACT
Computer Science Illustrated[1] is an endeavor to help visual learners comprehend computer science topics through a series of resolution-independent illustrations, which are made available online for use as handouts in class and posters in the computer labs. These illustrations are designed to present concepts as engaging and memorable visual metaphors combined with concise explanations or short narratives, intended to maintain the students' interest and facilitate retention. An additional goal of the project is to make learning the concepts an entertaining experience through the use of colorful and whimsical characters in the illustrations. In producing our twenty-seven illustrations, we determined which topics were most difficult for students to understand in our university's introductory computer science courses and followed a step-by-step process of design, redesign, and revision to generate our illustrations. We also assessed the effectiveness of our creations, using both subjective and objective measures.

Categories and Subject Descriptors
K.3.2 [**Computers and Education**]: Computer and Information Science Education – *Computer Science Education.*

General Terms
Algorithms, Measurement, Human Factors, Standardization, Performance, Languages.

Keywords
Introductory curriculum, illustration, visual learning, cartoon, learning style.

[1] **http://csillustrated.berkeley.edu/**

SIGCSE'10, March 10–13, 2010, Milwaukee, Wisconsin, USA.

1. INTRODUCTION
The relationship between a student's learning style and performance in computer science courses is a topic of considerable interest in computer science education research. Felder has shown that students have preferred modes of learning that can be categorized using the Index of Learning Styles, forty-four questions designed to determine a student's learning preferences, and Chamillard's work revealed the correlation between learning styles and course performance, regardless of the instructor teaching the course [4, 2]. More specifically, a study conducted by Thomas et al. showed that verbal learners perform significantly better than visual learners in introductory programming courses [8]. The implication is that current methods of teaching computer science present an advantage to students who learn verbally rather than visually. Students who learn visually frequently have difficulty understanding computer science topics because the lectures often lack images, charts, diagrams, and animations due to the time and skill required to produce them and the abstract nature of most concepts. Thus, the primary goal of this project is to assist visual learners by providing illustrations that convey the concepts. However, all students can benefit from this work, because it allows them to see the material presented in a fun and different way.

Constructivism also plays a role in a student's understanding of computer science subjects. According to the constructivist theory, students create knowledge structures by mentally selecting and organizing visual and verbal material and subsequently integrating it with prior knowledge they have on the subject [7]. These structures can be fallible, due to gaps in understanding. If the gaps go unnoticed, some students create mental models based on incorrect prior knowledge and require guidance to reconstruct them. At the same time, other students may enter a computer science course with no prior model of the concepts and may require guidance to prevent faulty constructions. Thus, this work also serves a purpose in constructivist learning by helping students with their mental models.

2. RELATED WORK
Incorporating visuals into computer science education is not novel. Diagrams, graphs, and data tables are often found sandwiched between text blocks in computer science books, but some authors let images play a larger role. One notable example is Larry Gonick's wonderful *Cartoon Guide to Computer Science*, a book describing the basics and history of computer science through hand-drawn diagrams and an assortment of characters [5].

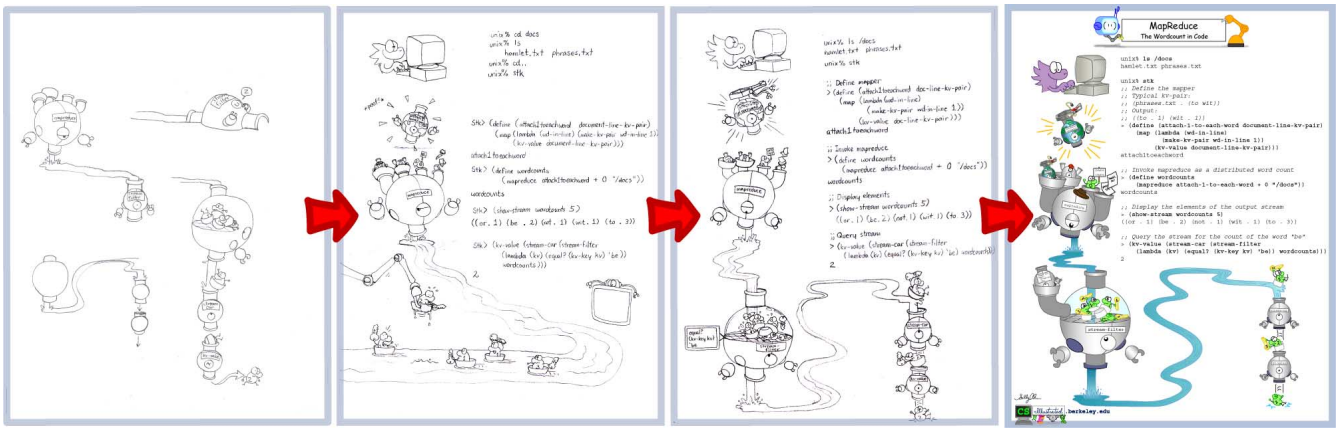


Figure 1. The production process (left to right): initial sketch, first and revised prototypes, and final digital illustration

These comical pictures help readers remember ideas by associating concrete objects with the abstract concepts, while at the same time offering a friendly introduction to basic computer science concepts. Another example is the *Teach Yourself Visually* series of texts by IDG Books. In addition to a variety of other subjects, *Teach Yourself Visually* conveys computer-related topics through step-by-step illustrations to reach out to visual learners.

In the space of computer science education research, attempts have been made to develop learning tools benefiting visual learners. Some incorporate visual elements into programming languages, such as Wally Feurzeig's LOGO. Through the use of turtle graphics, LOGO helps students learn programming as they procedurally generate line drawings. Scratch is another language that relies on visuals, allowing the creation of applications with graphical code blocks.

Most other approaches fall into the category of algorithm visualizations. Attempts have also been made to present visualizations of algorithms at various levels of abstraction, such as Müldner's Algorithm Explanation [6]. Animation is another option considered in algorithm visualization, since seeing the algorithm in action often helps with the understanding of its inner workings. Sorting algorithms are commonly explained through animated visualizations, as seen in Baecker's *Sorting Out Sorting*, a video explaining nine sorting methods [1]. While these visualizations have proven to be effective education tools, they are limited by three ways. First, their main purpose is to teach how algorithms work, so the methods used do not necessarily extend to all computer science concepts. Second, many algorithm visualizations, particularly animations, are presented through the computer and thus cannot be used as offline resources. Third, the visualizations commonly rely on abstract lines and shapes, which can convey actions without visual distraction but make it difficult to form memorable mental pictures of concepts.

Our approach can be considered a complement to algorithm visualization. It covers essential concepts as well as algorithms, and the illustrations are printable, which makes them accessible away from the computer. We also make use of visual metaphors to connect computer science concepts with concrete characters and objects, which students can more easily relate to than lines, shapes, and numbered nodes. This can help students when they attempt to formulate mental images.

3. PRODUCTION PROCESS

3.1 Development

The process of producing one of our illustrations follows the standard design process between artists and clients, beginning with determining which concept should be visualized. Topics are typically chosen from the set of subjects students most often have difficulty grasping within the courses. Occasionally, faculty will directly suggest a topic to illustrate.

Once a suitable topic has been selected, we plan out the illustration, discussing which aspects should be represented by images and which should be included as accompanying text. While the pictures are designed to convey as much of the information as possible, the inclusion of text is sometimes unavoidable, such as when code fragments or narrative segments form an integral part of the illustration. The product of these meetings is a set of sketches plotting out the basic structure of the illustration and rough ideas for the characters and objects that will be used as visual metaphors of the concept.

In the next step, we design the visual metaphors that will illustrate the concept and begin to consider factors such as how understandable, memorable, and aesthetically appealing the metaphors are. The illustrations often contain multiple metaphors, new and old, interacting with each other, so consistency with prior metaphors is essential in preventing confusion among students. Developing the visual metaphors is both the most important and the most difficult aspect of production.

After prototyping the illustration, setting the layout, and refining the visual metaphors' appearance, we start the revision process. In addition to inspecting the illustration ourselves, we present the prototypes to the teaching faculty and other members of the course staff and ask for feedback regarding accuracy, consistency, and clarity. Aside from corrections and suggestions for improvement, responses from the previewing audience can also include different opinions on how to present the concept and advice involving possible alternative visual metaphors. These responses can sometimes significantly alter the appearance of the final product, as well as increase the iterations of revision needed. An illustration may undergo numerous revisions before being considered ready for distribution, but at some point it must be considered "done" so that the next topic can be tackled. We have developed a checklist to know when we've finished:

1. All typographical and visual errors have been corrected.
2. The meaning of the text is clear.
3. The visual metaphors used are understandable.
4. For code segments, the code is free of syntax errors and the output is consistent with the input.
5. If the illustration contains aspects common to multiple illustrations, the visual metaphors used for those aspects are consistent.

On occasion, a revision in one aspect necessitates revision in the rest of the illustration. Thus the guidelines must be consulted at every revision step. Once the illustration satisfies all of the guidelines, it is added to the collection for distribution.

The final step is to draw the illustrations digitally [Fig. 1], allowing for online distribution of the illustrations and flexibility in printing. We convert the paper prototype into vector graphics and text in Adobe Illustrator, the industry standard for creating and editing vector illustrations. We use vectors instead of rasterized images for the digital illustration because vector graphics are infinitely scalable without loss of quality, which allows the illustration to be printed in arbitrary sizes. The conversion phase is also the point at which we add color and further detail to the illustrations. Details are added to the illustrations to make them professional, and include gradients for shading, grounding shadows, and variations in character poses to prevent a "cut-and-paste" appearance. All of the work is done on drawing tablets, the standard tool of graphic designers and illustrators in industry. To make the most efficient use of time, this conversion can take place in parallel with the revision step. The final output is a Portable Document Format (PDF) file.

The production process for one illustration requires two weeks, working six to ten hours per week, on average. Planning and revision occupies the bulk of the two-week span. However, revision of one illustration often occurs concurrently with the planning and drawing of another illustration, though one artist will usually work on no more than two at any given time.

3.2 Challenges

Through several iterations of the production process, we have identified four main challenges that we must address: what to illustrate, what metaphors to use, level of detail, and keeping consistent.

For the first challenge, we typically choose to illustrate topics students consider to be difficult. We gather sets of topics from the course staff, because instructors and teaching assistants can often identify problematic areas from interactions with students during office hours and discussion, as well as from the portions of assignments and tests where students are weakest. In addition, we must determine the reasons why students find a topic difficult to understand. Identifying the problem areas allows us to decide what aspects of the topic to visualize and guides the creation of visual metaphors.

The second challenge is ensuring the effectiveness of the metaphors used to convey the concepts. To allow students to quickly and easily make the connection between what occurs in the illustrations and the ideas they represent, the imagery must be simple, clear, and memorable. The additional factor of aesthetic appeal enhances retention and recall of the metaphor and its corresponding idea. However, many aesthetically appealing designs are not simple or easy to memorize, while the simplest or clearest representation may not be the most attractive. In designing a metaphor, we seek to achieve a balance among the four competing factors, a task depending heavily upon the knowledge, skill, and creativity of the illustrators.

Often metaphor generation involves producing a design that encapsulates the most important aspects of the represented concept or element. For instance, in Scheme lists, the characteristics to emphasize are the groupings established by the lists' parentheses and the fact that list elements are ordered. Thus lists are symbolized by ordered rows of values occupying rectangular buckets marked with parentheses [Fig. 2].

Figure 2. Visual metaphors for a simple Scheme list (left) and a nested Scheme list (right). The nesting of buckets makes the nesting of parentheses more apparent.

Embodying the key aspects of the concept or element allows students who already have a working knowledge of what the illustrations represent to follow along when the metaphors are used to teach something they are less familiar with.

Compromises must be also made regarding the level of detail provided by each illustration. Insufficient detail can contribute to confusion and reduce the illustration's usefulness, but covering too many points at once can overwhelm the viewer.

Consistency among visual metaphors in illustrations, and among the illustrations themselves, is essential to the project. The visual metaphors are designed to help students establish a mental image of computer science concepts. Sudden changes in representations can cause confusion and misunderstanding, so a metaphor must remain an effective representation of an aspect, whether it is alone or combined with another metaphor. There are two main cases where consistency must be ensured: between subtopic illustrations and among different topics within the course. For example, representations of different Scheme functions share a common appearance, regardless of the number or type of arguments they take [Fig. 3]. Unnecessary visual variations here could lead students to false conclusions, such as one saying functions are different datatypes.

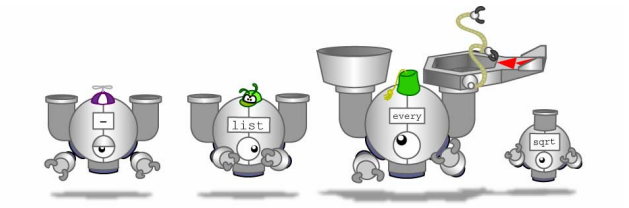


Figure 3. All Scheme functions are given the appearance of an input-output machine.

Since illustrations can use multiple metaphors interacting with each other to convey information, old and new metaphors often interact. For example, the docking station found on higher-order function machines was developed to maintain the consistency of list and sentence input as boats. If the characters' or objects'

modes of interaction are not compatible, inconsistencies will emerge during the narratives.

4. DISTRIBUTION

Once an illustration is produced, there are three main methods that it can be put to use in a computer science course. Online distribution is one avenue we have established. The illustrations that we have created so far are available in PDF on a publicly accessible website, along with brief descriptions of the project, the illustrators, and recurring characters in the illustrations.

The second use we intend for the illustrations is as course handouts, supplementing the lecture or discussion sections. We foresaw the possibility of printing illustrations as handouts, so all those currently produced are designed to fit the standard 8.5”x11” paper size. Finally, the resolution-independence of our graphics also allows larger scale printing, allowing faculty to print them as computer lab posters [Fig 4].

Figure 4. Illustrations proudly on display in one of our labs.

With these distribution options, particularly online availability, the matter of intellectual property becomes a concern. We chose to release our work under the *Creative Commons Attribution-Noncommercial-Share Alike 3.0* license to enable non-commercial use of the illustrations and the creation of derivative works based on them [3].

5. RESULTS

To quantitatively measure the effectiveness of Computer Science Illustrated, we conducted experimental assessments and a survey on the students of the introductory computer science courses. In particular, we selected the introduction to programming course and the machine structures course for our assessments. We chose these two courses because their lab sections were presented through UCWISE, a platform for displaying lessons, activities, and student assessments online [9].

For the study conducted on the introduction to programming course, the lab sections of the course were randomly divided into two groups: the control group and experimental group. The students were not informed of this division, and all received the same instruction in lecture. During the lab sessions, both groups received a seven-question Scheme quiz. The experimental group was allowed to view three illustrations relevant to the quiz topics beforehand. To ensure fairness, in case the illustrations offered a significant advantage, every student received the illustrations at some point; the control group was allowed to view them after completing the quiz. The quiz scores collected were then analyzed to gauge whether the illustrations were helpful for the students' understanding of the concepts.

This first study suggested that student understanding of concepts improved when the lessons were supplemented with the illustrations. On average, the group of students who received the illustrations scored higher on the quiz than the control group. The average score experimental group was 6.053, while the control group had an average score of 4.771. Both scores were out of seven points.

Despite the outcome, a couple of concerns emerged. The first was the possibility that the experimental group had an obvious advantage, since the control group had no supplements. Another issue was that providing the illustrations to the experimental group minutes before the quizzes may not allow time for students to fully comprehend the metaphors and remember the ideas. Therefore, we structured the study conducted on students of the machine structures course differently. This time the illustrations, covering caching, were presented a week prior to the quiz. This gave the students of the experimental group more time to study the illustrations and us the chance to determine if any deep learning was taking place. Additionally, the control group received text-only equivalents of the illustrations, so that the assessment would be a direct comparison between illustrated lessons and text-based lessons. The quiz consisted of three questions about caching and cache misses in general. Again, to maintain fairness, both groups received the text supplements and illustrations after the quiz.

Despite considering these additional factors, the study conducted on the machine structures course produced inconclusive results. There was no significant difference between the average quiz scores of the control and experimental groups, which were 2.429 and 2.433 respectively. These quiz scores were out of three points, so it may be that the quiz was too short or the questions too general to determine the students’ understanding of the concepts. Perhaps the supplements improved comprehension, regardless of whether they were purely text or illustrations. It is also possible that by giving both groups a week to absorb the information, the students consulted other resources and achieved nearly equal levels of understanding. The lessons learned from both experiments will serve to improve future assessments and create better gauges of the illustrations’ effectiveness.

The responses from the subjective survey revealed that the students’ reception of the illustrations was very positive overall [Fig. 5]. The average rating, a score between 1 (poor) and 7 (excellent), students gave the illustrations was 5.12, with a mode of 6. When asked if they wanted to see illustrations offered in other computer science courses, more than half the students agreed. As for how they used the illustrations, most students used them as introductions and to create mental models of the concepts. In the open-ended responses, most of the students who used the illustrations stated that they were “helpful in visualizing what some of the functions did,” “easy to read and understand,” “a cool way to teach CS to beginners,” “amusing”, and “cute.” Among the small minority of students who did not

find the illustrations helpful, the most common comment was that the textbook and other course material were adequate for their understanding, suggesting these students already had a good grasp of the concepts prior to viewing the illustrations.

6. SUMMARY

We have presented an approach to facilitate computer science education through the use of visually engaging and informative illustrations made available to students as course handouts, as an online resource, and as large posters to view in the computer labs. The detailed process we go through and the challenges we must address for every illustration ensures the visual metaphors encapsulating the concepts are understandable, memorable, and consistent within the groups they are used.

The assessments we conducted to evaluate the effectiveness of the illustrations seem to suggest that they are useful as supplements to the lessons taught in class, though further studies will be necessary to arrive at a more solid conclusion. Surveys have also shown that students are highly in favor of using the illustrations as entertaining introductions to the concepts, as well as guides to forming mental models.

7. ACKNOWLEDGMENTS

We wish to thank Michael Clancy for his encouragement, and providing assistance during the assessments. We would also like to thank Nate Titterton and Colleen Lewis for helping out with the UCWISE assessments. This work was funded by the Weiner Foundation and the UC Berkeley Office of Educational Development.

8. REFERENCES

[1] Baecker, R., *Sorting Out Sorting*. Videotape, 1981.

[2] Chamillard, A. & Karolick, D. Using Learning Style Data in an Introductory Computer Science Course. *ACM SIGCSE Bulletin*, Vol. 31 No. 1 (Mar 1999) 291-295.

[3] Creative Commons. *Attribution-Noncommercial-Share Alike 3.0*. http://creativecommons.org/licenses/by-nc-sa/3.0

[4] Felder, R. & Spurlin. J., Applications, Reliability, and Validity of the Index of Learning Styles. *International Journal of Engineering Education*, Vol. 21 No. 1 (2005) 103-112.

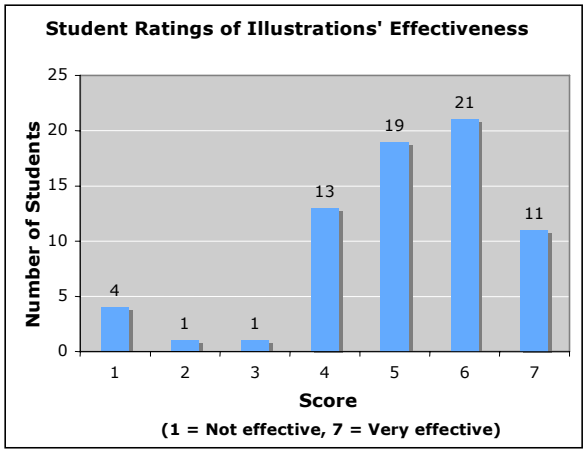
[5] Gonick, L., *The Cartoon Guide to Computer Science*. New York: Harper & Row, Publishers, Inc., (1983).

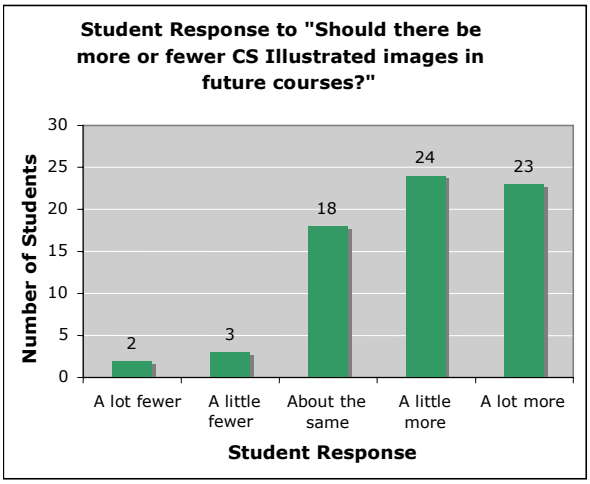
[6] Müldner, T. & Shakshuki, E., A New Approach to Learning Algorithms. Proceedings of the International Conference on Information Technology: Coding and Computing; 2004 Apr 5-7; Las Vegas, NV. Los Alamitos: IEEE Computer Society; (c2004).

[7] Otero, J. et al., *The Psychology of Science Text Comprehension*. Mahwah: L.Erlbaum Associates, (2002).

[8] Thomas, L. et al. Learning Styles and Performance in the Introductory Programming Sequence. Inroads, Vol. 34 No. 1, (Mar 2002) 33-37.

[9] UCWISE. *Home page*. http://www.ucwise.org/





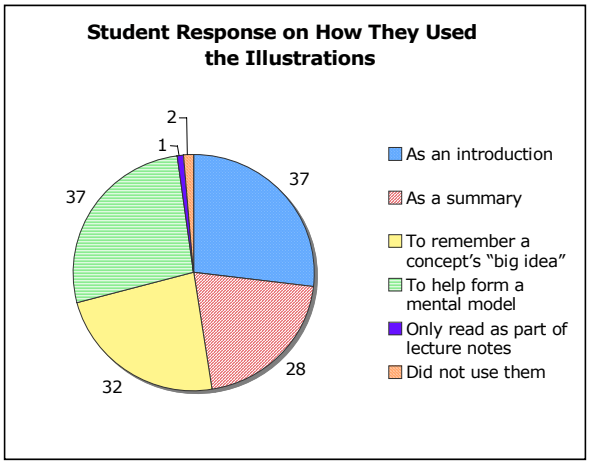


Figure 5. Results from student surveys regarding the effectiveness, future, and uses of CS Illustrated.