Why Do Fools Fall Into Infinite Loops: Singing To Your Computer Science Class

Eric V. Siegel
Department of Computer Science
Columbia University
New York, NY 10027
evs@cs.columbia.edu

Abstract

One effective way to introduce a dose of humanity, acknowledge the needs and struggles of CS1 students, and appeal to a broader range of learning styles is to present a computer science topic in an entertaining manner, e.g., with some form of artistic performance. In this paper, I describe three songs for CS1 that are designed to help students surmount three of the most difficult hurdles of that course. Empirical and anecdotal results demonstrate that the songs help students learn, and help them enjoy learning. These songs are a case study in entertainment; all instructors can find some way to entertain their class, and recordings of the songs themselves are available on the web for any instructor or student.

1 Introduction

I have been singing to my computer science students for three semesters, accompanied by my Yamaha PSR-520 keyboard. This is my story.

An undergraduate's introduction to computer science is all too often a tremendously intimidating experience. The student is faced with a slew of completely new vocabulary and concepts, some of which are particularly difficult. The range and magnitude of material she will have to face that semester is particularly unpredictable, potentially of forboding magnitude. What's more, many of the classmates surrounding her have had years of programming experience. As a result, students in this course experience more stress than any other introductory undergraduate course [3].

On top of this intimidation, several factors can make the learning and doing of computer science a dehumanizing experience, i.e., revealing a lack of concern with human needs and values:

- There are various sources of pressure today to become computer-literate, since there is a general message, pushed by the media and popular culture, that we are in the midst of a revolution of technology and information. This pressure can serve as an obstacle to students experiencing their personal desire to learn about computer science. The concept of what they "should" be clouds their subjective experience of what they want to be. Without a love for her or his work, a student is operating out of selfless submission or servitude [1].
- Since course lectures are generally only long enough to cover a portion of the details of any given programming language and computer system, students are faced with long, solitary hours working through many technical details. To a novice, such details can appear to be senseless, since there is not yet a framework from which to understand their necessity. They are expected to sink or swim by finding many solutions on their own, e.g., with elaborately technical Unix man pages. This attitude is exemplified by the frequent message, "RTFM."
- Like many high school and college courses in mathematics, computer science courses run the risk of underemphasizing the motivation behind technical content. This is exacerbated by the fact that in CS1, there is often a large amount of technical overhead relative to the volume of underlying principles and concepts, especially when using a large-scale programming language such as C.
- Since introduction courses tend to have the largest enrollments, especially at large universities, CS1 students can suffer from the feeling that the course is impersonal.

These factors contribute to making the learning environment more intimidating and non-constructive. They can serve to cripple a student's ability to work effectively, or lead to student burnout. As a result, many prospective computer science students with worthy potential are filtered (or "weeded") out. Alternatively, some students find the need to "disown themselves," sacrificing or repressing their needs and feelings, i.e., their humanity, for work [1].

Moreover, these factors can contribute to the struggle by students of all success levels to feel interested. Since their personal human needs are effectively being ignored, students can become increasingly disattached, begin to care less about the course material, and therefore experience difficulty paying attention. That is to say, many college students would prefer to be at a rock concert than in class.

One effective way to introduce a dose of humanity, acknowledge the needs and struggles of computer science students, and appeal to a broader range of learning styles is to present a computer science topic in an entertaining manner, e.g., with some form of artistic expression such as music, poetry (e.g., beat poetry) or a dramatic dialogue (e.g., between puppets). In this paper, I describe three songs for CS1 that are designed to help students surmount three of the most difficult hurdles of that course. These songs are a case study in entertainment; all instructors can find some way to entertain their class, and recordings of the songs themselves are available on the web for any instructor or student. Empirical and anecdotal results demonstrate that the songs help students learn, and help them enjoy learning.

2 Approach: Artistic Expression in Class

There are several benefits to artistically performing a CS1 lecture. First, senses of humor and irony facilitate a release of tension on many levels, since they provide extremely powerful methods to both acknowledge and cope with the struggles of learning and applying, be they technical struggles (e.g., the difficulty of using pointers), social struggles (e.g., the factors that cause feelings of intimidation and dehumanization mentioned above), or personal struggles (e.g., the angst of a long debugging process). Humor and irony are accomplished by such a performance in several ways:

- Directly, with the lyrics or prose of a piece.
- The incongruity of this type of endeavor in the context of a serious, academic course on a scientific subject.
- The incongruity of approaching an esoteric topic within a mainstream musical style; for example, singing about the angst of debugging in a rock ballad, or about the triumph of modular decomposition in a musical theater number.
- The potential for the instructor to compromise her or his personal integrity (on a harmless level); the scientifically proficient teacher is putting her or his artistic self "on the line."

Introducing an element of entertainment to classroom activities, e.g., with music, humor or clever lyrics, has several advantages. First, it conveys the message that it is OK to enjoy work in the area of computer science. Furthermore, if the students are being entertained, they are focused and paying attention. If they are charmed by an artistic piece, they will be intrigued and motivated to understand the lyrics on "artistic" levels, such as clever handling of the material, or other humorous aspects, as listed above. That is, to understand the aesthetic appeal, they must understand the content. This harnesses the power of group learning, since the responses of fellow classmates such as applause or laughter can stimulate an individual student to keep up with the group ("Hmmm, why were they laughing at *that*?").

An artistic endeavor in your computer science course also provides the following contributions:

- Adds variation to lecturing style or classroom activities.
- Helps destigmatize elusive concepts and big technical jargon by placing them in an informal context.
- Provides a conducive venue for metaphorically relating the material to everyday life in extreme terms, thus broadening context, improving abstract comprehension and increasing motivation. Examples include comparing program modules to hamster Habitrail (tm) modules and to sub-tasks in life, and, "if you follow the pointer, you'll get to the point."
- Instructor demonstrates her investment in the student as a whole person, beyond mainstream academic performance.
- Instructor enjoys work more, which is clearly conducive to creating a constructive learning environment. Many CS teachers are frustrated musicians and, honestly, there are moments when I'd prefer to be a rock star than a computer instructor.

Among artistic endeavors, music in particular has advantages. For one, music helps with memorization. For example, many U.S. citizens in their late 20's and early 30's have the Preamble to the Constitution memorized due to an educational 1970's musical television show on ABC called "Schoolhouse Rock," which used song to teach various topics in history, science and grammar (the preamble song is now available on a CD, "Schoolhouse Rock: America Rock," by Kid Rhino Records). Furthermore, previous work illustrates the potential to convey irony with song, e.g., alternative computer lyrics that parody well-known songs (see `http://www.poppyfields.net/filks/` and `http://www.netspace.org/users/dmacks/pub/parodies/internet-songbook/`).

3 The Songs

Three topics that arguably require some of the most dramatic mental adjustments by CS1 students are 1) the attention to detail required to carefully design and debug a program into working order, 2) the elusive concepts of encapsulation, abstraction and modular decomposition and 3) the highly technical domain of pointers. This section describes three original songs that help smooth over these adjustments. `www.cs.columbia.edu/~evs` has the complete lyrics and live recordings of these three songs, an orchestrated studio rendition by a professional Hollywood musician of the second song, "Modular Decomposition," as well as two newer songs.

The chorus (i.e., refrain) of each song describes the main concept of the song humorously, metaphorically, or both. Then, each verse is densely packed with details, such as technical error messages, so they rhyme. It is important to set up a context for the song before performing it, so that students have already been introduced to the main concepts. However, it is most effective if students are only beginning to learn the concepts, and even if portions of the song introduce some details that are new. After each performance, the lyrics are then handed out and explained in greater detail.

3.1 "The Angst of Debugging"

"[Students] must come to grips with the difficulties and pitfalls [programming] entails." [2] This song is a rock ballad with something of a descending baseline. Currently, some parts are specific to programming in C.

"syntax error", "parsing error"; did you forget your semicolon? / Programming is 99% debugging, so you better keep it scrollin'. / "unterminated string", did you match your double-quote? / Comment your code, label your node, write yourself a note!

Chorus:
It's the pain of finally figuring out what went wrong. / Who's to blame when you confuse two equal signs with one? / Compiler warnings should not be ignored; you'll dereference

a NULL pointer. / If you screw up and find your butt in an infinite loop, "`control-c`*" will control C.*

Don't disrespect your teaching assistant – she's your biggest hero./ "floating point exception" is a run-time error when you divide by 0. / "segmentation fault" can bring you to tears – in your throat there'll be a lump, / 'cause when it's time to submit your homework all you'll get is a "core dump"

Chorus

Bridge:
Ultimately what you've got to do is narrow down the problem. / It's like finding a needle in a haystack; you're never gonna solve them, / unless you put in some printf()'s, comment half that haystack./ "stack overflow", you've no place to go, your program is a lame hack!

Chorus

3.2 "Modular Decomposition"

This song is a musical theater ballad with a tone of triumph and dramaticism. There is a piano solo in the middle; before the performance, the song is deconstructed down to a trill between the notes G and A in the piano solo, to illustrate the hierarchical decomposition of the song itself.

How do I program each line, and finish it on time, / when this project's so darn big and overwhelming me? / With stepwise refinement, you're breakin' down the problem, / then build it back up systematically.

Chorus:
Modular decomposition! / It's the way things are and ought to be. / If your life is hierarchical, it's really quite remarkably / easy to do each sub-sub-sub-sub-sub-routine.

All the functions I service must serve a general purpose, / yet operate alone with one-track minds. / Conceptual abstraction is maintained with a module interface, / to build up libraries of every kind.

Chorus
Piano Solo
Chorus

Bridge:
You and and your abstraction, like Batman and Robin, / an inventive co-dependent relationship of epic proportions. / Who will invent a multi-purpose metaphor and meta-metaphor? / Who will architect your hamster's Habitrail (tm) with multi-purpose tubes? / Who will protect your private micro-thoughts from hacking invaders? / Who will save your code from infinite tedious intricacy? / Who knows how to use Legos really well? / You and your modular abstraction.

Chorus

3.3 "Pointers Rap"

The chorus is sung to a funky tune accompanied by clavichord, and the verses are spoken rhythmically with an attitude, accompanied by a percussive beat. Currently, some parts are specific to programming in C.

Chorus *(sing):*
Show me your data, point it out to me. / If I know where it is then I can access easily. / With the asterisk I'll dereference; I can print it out, / and I can even change it, without a doubt.

The ampersand can awesomely tell you the location. / When your program's running smoothly you can go on a vacation. / So a program with no pointer's like an elbow with no joint; / if you follow the pointer, you will get to the point.

Verse *(rap):*
"Memory's like real estate" – I'll state and make my metaphor. / You want to get some property, a lot, a plot, a farm, a store. / So allocate your memory and claim it, 'til you use it up. / Otherwise, "memory fault", "core dump", all screwed up!

Variable values are stored at locations... / in memory that is; that's where the space is. / A pointer is some data like an int, char or double, / but you better treat it special or you're gonna be in trouble.

Each location in memory has a memory address. / That address is the number used to find where data rests. / An address is a value, just like any other, / so it is stored in memory like its sisters and brothers.

"All memory's created equally," said William Gates. / (But Microsoft consumers rank as low as big Bill states.) / Values are content or pointers; I prefer the latter. / Pointers are meta-data tellin'-me where is what that matters.

Chorus

Verse *(rap):*
An array's just a pointer, it turns out, so whatta ya know? / It's a pointer to yer data, all set out in a row. / That's why when ya pass it, it's always by reference; / the function that gets it can index and dereference it.

It's not polite to point; it's totally rude. / But the data that you point to might be a pointer, dude, / and that can point me elsewhere, ultimately, / A pointer to a pointer and I'm meta-meta-C.

Don't dereference a NULL pointer; you should not abuse it. / Value memory's values; use it or lose it. / In Java, everything is a pointer, consistently. / But "C" is for "cookie" and that's good enough for me!

So, don't forget to dereference a valid pointer, / and don't forget to double-dereference a pointer to a pointer, / and don't forget to triple-dereference a pointer to a pointer to a pointer, / and don't forget to quadruple-dereference a pointer to a pointer to a pointer to a pointer,

Chorus

Rap:
Linked lists will be your very first abstract data type. / We saved it 'til the end because now your brain is ripe. / Each element is stored in an allocated structure called a "node"; / Each node points to the next one, and "node" rhymes with "toad".

So now you know how to manage memory's massive space. / My first computer had 48k and no lowercase! / The past is gonna haunt us though your computer's better than mine, / 'cause next year we're gonna party like it's 1899.

Chorus

4 Results

Results from a poll indicate that students believe the songs helped them learn. As part of an anonymous, "initial reaction" survey, midway through the semester, after the first two of the songs above had been performed in class, students were asked to respond to the statement, "The songs help me learn," with an integer rating between 5, "completely agree," and 1, "completely disagree." The results were 2 10's (these students did not adhere to the constraints of the question), 35 5's, 15 4's, 35 3's, 10 2's and 9 1's. Of the 19 2's and 1's, 8 students included spontaneous, unsolicited written statements in support of the songs, which clarified that their response was not a rating of the songs and indicated that they felt the songs were appropriate for the course. Since a portion of the students come to CS1 with programming experience, they presumably already knew too much to learn as much from the songs as the true beginners. Despite this, it appears that many such students found the entertainment of the songs valuable for the course.

Written comments by students on the anonymous course evaluation at the end of the semester showed a disproportionately large amount of attention to the songs. Of 30 students who provided positive written comments, 5 included positive comments regarding the songs. This is a large proportion, considering the songs only took about 45 minutes (15 minutes each, including post-discussion of lyrics) of a total of 2,025 minutes of lectures (27 lectures, not including exams) plus 6 recitation sessions. Comments included, "All Columbia instructors should sing in class. It is very energizing and arousing. Viva Siegel!" and, "The songs really were a great way to remember the topics." One of seven non-anonymous unsolicited emails with positive comments about the songs said, "Just the idea of incorporating song into actual pertinent class information is brilliant... It is truly inspiring to see a teacher so very committed to and involved with the class he teaches." No negative or critical student communications concerning the songs have been made on polls or elsewhere.

The songs were received enthusiastically by the students. Both "Modular Decomposition" and the "Pointers Rap" concluded with rounds of applause reaching a healthy 16 seconds, as measured from the live recordings available from the URL above. Several performances had partial standing ovations. This enthusiasm demonstrates that students were in the least paying clear attention. Note that the on-line recording of "The Angst of Debugging" was a reprise, so the audience response is not as vitalized.

The benefit of introducing the songs to CS1 is reflected in the course evaluation ratings. In the fall semester of 1997 when I taught CS1 with no songs, responses to the final question of the course evaluation, "Course: Overall Quality," averaged 3.83 (out of a possible 5, over 90 responses). In the spring semester of 1998, I taught CS1 with the same course material, primarily the same in-class examples, and the same programming and reading assignments, but with the addition of the three songs, and six informal recitation sessions given by undergraduate TAs. This second semester, the average rating was 4.13 (over 107 responses across two sections). Although there is no true control experiment to keep this comparison objective, the addition of the songs was certainly part of a systematic improvement of the course's content and structure.

5 Conclusions and Additional Work

An entertainment-oriented presentation of CS1 subject matter is engaging, effective, and affirming. Our results show that it is an alternate teaching method with a highly beneficial payoff, and indicate that it is felicitous for virtually all students. Teaching and learning are human experiences, and one straightforward way to humanize computation is to package it within activities that are designed to be aesthetically pleasing. Moreover, computer science emphasizes that "multi-media" is an important tool for communication to a computer user, and this can be generalized to communication to a computer student.

A few cs-educator colleagues have initially responded with mild skepticism to this unorthodox paradigm, asking what kind of student response was elicited. Perhaps some are concerned that singing in class is distracting from the content of the course. I hope that the reasoning and results in this paper show the clear advantages of performing for your class. For non-singing instructors, consider these songs a case study in entertainment; all instructors can find some way to entertain their students. Furthermore, recordings of the songs themselves are available for instructors and students at `www.cs.columbia.edu/~evs`.

I have also developed a patter, "Digital Love," on logic circuits and computer organization (for a service course), and a rap, "Learn This," on machine induction (for graduate AI courses), both of which are also available at the web site above. Recursion is another outstanding hurdle for many CS1 students, and I have developed a written homework assignment in which a series of co-dependent recursive functions' outputs map to a three-part harmony rendition of a catchy circle-of-fifths based melody, available from the author on request. Further songs and a CD compilation of studio recordings are forthcoming.

Acknowledgements

The idea of educational computer science songs came when Leana Golubchik mentioned songs Mark Rubin sang for computer science students at Johns Hopkins' Center for Talented Youth summer program. I have received invaluable support and input in developing the songs from many of my friends and family, as well as enthusiastic students. Thanks to Adam Cohen for orchestrating the recording of "Modular Decomposition," and to Lisa A. Schamberg, Judith Klavans, Alex Chaffee, and Eleazar Eskin for helpful comments on an earlier draft of this paper.

References

[1] E.H. Fromm. *Escape from Freedom.* Henry Holt and Company, Inc., New York, 1941.

[2] J. Gal-Ezer and D. Harel. What (else) should cs educators know? *Communications of the ACM*, 41(9), 1998.

[3] L. Sproull, S. Kiesler, and D. Zubrow. Cultural socialization to computing in college. *Computers in Human Behavior*, 2:257–275, 1989.