Supporting Creativity and User Interaction in CS 1 Homework Assignments

Tammy VanDeGrift
University of Portland
5000 N Willamette Blvd.
Portland, OR 97203
503-943-7256
vandegri@up.edu

ABSTRACT
In this paper, we describe CS 1 programming assignments that encourage design creativity and that utilize user testing. All course assignments allowed for some student-defined specifications; some assignments required user tests to encourage interaction with other people. The open-ended nature of the assignments supported students' creativity and motivation to learn. The user tests provided a platform for students to share their creations and knowledge about computing with others. Both the creative aspect and sharing aspect of the assignments led to students taking ownership of their work. Overall, 41 of 44 (93.2%) students enjoyed the open-ended nature of the assignments and sharing programs with users also enhanced their understanding of their programs and computing concepts.

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

General Terms
Documentation, Human Factors

Keywords
Novice programmers, motivation, user interaction, creativity

1. INTRODUCTION
"I hear and I forget. I see and I remember. I do and I understand." *Confucius*. Like most university computing courses, our CS 1 students understand the most about computing and programming when constructing their own programs, generally outside of formal instructional time. Like most universities, our students are expected to spend two hours on coursework outside of class for every hour inside class [1]. This poses an interesting challenge for computing faculty: out-of-class time is where the learning unfolds and develops, so homework assignments should be intentionally designed to engage students in the learning process.

SIGCSE '15, March 04 – 07, 2015, Kansas City, MO, USA
Copyright 2015 ACM 978-1-4503-2966-8/15/03…$15.00
http://dx.doi.org/10.1145/2676723.2677250

There are several models for designing programming assignments for introductory computing students. Each model has advantages and disadvantages for learning and assessment. For example, one homework model uses precise program specifications, so grading can be automated through test case comparisons between the model solution and the student's solution [10, 13]. This model may be necessary to assess large sets of programming assignments. Another example model uses lab-like problems where students complete functions and methods either from scratch or by extending existing code. This has the advantage that students can get automatic and immediate feedback on a shorter piece of the solution, but may not showcase the creative and design aspects of software development [5, 6].

In our approach, we used three design principles in developing the programming assignments: 1) every assignment includes open-ended elements to encourage students to decide how to define part of the specification and provide latitude for students to be creative in their design and implementation, 2) over half of the assignments have a required user interaction session to encourage the idea that programs are written for other users, and 3) a written summary accompanies each assignment; some written summaries ask students to write a user manual, other summaries ask students to supply their test cases, and other summaries ask students to supply class diagrams. The purpose of the summaries is to challenge students to describe their programs in English and reflect on the design, implementation, and testing process, while scaffolding good documentation habits.

The media computation approach for CS 1 has been successful in providing context and outlets for creativity for students [4]. One of the assignments used in our CS 1 course has elements of media computation (image transformations). Like [4], there were several motivating factors in developing assignments with elements of creativity: provide an opportunity for students to take ownership and pride in their work and to support learning. Another approach described in [8] is to use creative thinking exercises to encourage students to learn about computation; these exercises were not part of the programming itself but focused more on novelty, challenging existing patterns, broadening knowledge and using new environments for stimuli.

The learning theory supporting this programming assignment approach is constructivism [3, 9]. Constructivism is the idea that students build their own interpretations of the material based on their own models and experiences. It also relates to Vygotsky's zones on proximal development [14]. Challenging students to exercise their creativity allows each student to explore at a level that is closest to their zone of proximal development. Also, explaining their code and program to a potential user scaffolds the expansion of their zone of proximal development.

Another consideration in the development of this homework assignment approach is the target audience. Students who were born in the early 1980's to the late 1990's comprise the millennial generation [2]. While it is dangerous practice to stereotype an entire generation, some general trends describe this population of students. Traits of Generation Y include *customization* (students want to personalize their things and experiences), *creativity* (students are more right-brained than left-brained), *authority* (students want to have ownership and authority), and *connected* (students are tightly connected to peers and family). The design of the homework assignments in our CS 1 course takes these generational qualities into consideration. By allowing students to *customize* their solutions, they can exercise *creativity* and take *authority* over the design and development of their programs. Secondly, students can exercise their *connectedness* with others through the user interaction and user tests of their solutions. Note that *connected* is part of the SIGCSE 2015 conference theme.

2. STUDY

This study expands on the work in [12] that describes the injection of creativity in CS 1 programming assignments. The study reported here seeks to examine if the use of creativity-supported assignments motivates students to learn and the impact on learning when students share their programs with users. The hypotheses for this study are two-fold: 1) allowing students to exercise creative freedom motivates students to learn and complete homework, 2) asking students to share their programs with non-programmers engages their learning.

In general, we sought to understand what motivates students to complete programming assignments. Second, will students be motivated to go beyond what is asked in the basic homework requirements? These questions relate to studies regarding student motivation (intrinsic versus extrinsic) and grades. Lin, McKeachie and Kim found that college students with medium extrinsic and high intrinsic motivation have higher course grades than students with low or high extrinsic motivation [7]. Students may be better served when emphasis is on learning and reflection on learning instead of exam and homework grades.

2.1 Context

The University of Portland is a comprehensive, private Catholic university serving ~3700 undergraduate students. The course is CS 203, Introduction to Computer Science, taught in two sections in spring 2012 by the same professor. The two sections had common assignments, lectures, exams, and in-class activities. Java is used to emphasize problem-solving and computer science concepts. In total, 44 students gave consent to be in this study.

The course is required for Computer Science, Electrical Engineering, Math, and Physics majors. Other students can take CS 203 to fulfill elective credits. Of the 44 students, 31 (70.5%) were male and 13 (29.5%) were female. Most students were majoring in computer science, but there was some diversity among the majors: 19 CS, 8 EE, 4 Math, 4 General Engineering, 2 Biology, 2 Mechanical Engineering, 1 Sociology, 1 Psychology, 1 Economics, 1 Education, and 1 Undeclared.

2.2 Homework Assignments

The course included ten assignments. Eight were one-week assignments and two were two-week assignments, split into two parts. In general, students submitted an assignment or part of an assignment every week of the semester. As mentioned previously, every homework assignment had a creative entity and most encouraged students to extend the program beyond the basic requirements. HW 2, 3, 4, 5, 8, and 9 required user interaction and a summary of the user test in the written report. Table 1 briefly describes each homework assignment. See https://sites.up.edu/sigcse2015/ for the full text of each assignment.

Table 1: Descriptions of homework assignments

Assignment	Objective	Creative Bit	Primary CS Concepts Addressed
1: Fortune Teller	Get first Java program written/compiled; use of variables, input and output	Students chose their own fortunes to present to the user, along with the calculation for their 'lucky number'.	Variables I/O Compilation and Execution
2: Madlib	Create a Madlib story, replacing user-input into story with some user input modified using String methods.	Students chose their own Madlib story and missing words. The assignment required that they perform certain operations, such as capitalize just the first letter of the word, switch the order of a two-word phrase, etc., but students chose where to place these operations in their stories.	I/O String manipulation and Java-defined String methods
3: Graphical Greeting Card	Create a graphical greeting card	Students designed the graphics, wrote the code to generate the image, and processed user input (such as the card recipient's name, the sender's name, the desired color, etc) for the custom card.	I/O with dialog boxes Java Graphics (JApplet, Graphics class) Selection statements
4: Golf Simulation	Create a 2D golf landscape and draw a ball flying through the air, given the user's launch angle and velocity.	Students chose how the background of the golf hole was graphically designed. They could include wind direction and speed (optional input).	Java Graphics Selection statements Loops (to draw the ball path)
5: Wheel of Fortune	Create a textual Wheel of Fortune game. Users guess a letter and the locations of the guessed letters are presented to the user.	Students chose the list of potential phrases that could be used as the puzzle. Students could also enhance the game with 'spinning the wheel', earning money, vowels costing money, and supporting multiple players.	Random numbers Selection statements Loops Arrays Methods

6: Library Design	Create UML diagrams for two different classes that would be included in the software to support library users, borrowing books, and/or searching the library's collection. (The University Library was going through a physical renovation at the time, so this system was relevant to the students.)	Students chose the two classes they designed.	UML Class design Methods Instance variables Types
7: Game Player class implementation	Create and implement a Java class to represent a game character for a game of their choice. For example, the class could represent a player in a car racing game, a board game, etc.	Students chose what game they wanted to model for the player in that game.	Class implementation Instance variables Methods Constructors
8: Image Transformations	Implement six different image filters, such as convert color to grayscale, shift right, create a border, create high contrast, brighten, etc.	Students chose their own filters to implement. Some did edge detection, blurring, brightening, pixilation, sepia, etc.	Arrays of objects (the picture was a 2D array of Pixel objects) Java Interfaces Nested loops
9: Inheritance (Choose your own project)	Design and create a program of the student's choice, as long as the program includes inheritance. The inheritance relationship should make sense and not be contrived to satisfy the assignment.	The entire assignment.	Inheritance (class extension) Class relationships Polymorphism
10: Palindromes and Files	Create a program to search for palindromes in the English language (provided as a text document)	Students had to choose some other property to search for (such as words containing the substring 'cs', the number of English words longer than 15 characters).	Recursive methods File I/O Exceptions

2.3 Data Collection and Analysis

The primary data for this study were responses to an IRB-approved end-of-semester paper survey. The survey was administered on paper to encourage a high response rate and to distinguish this survey as separate from the regular on-line course evaluation survey. See Appendix 1 for the full list of survey questions; note that the actual survey instrument spanned multiple pages. This paper focuses on the questions related to creativity in assignments, student motivation, and connectedness regarding the user tests. The open-ended survey responses were coded using content analysis with emergent categories [11]. An individual response may be coded into more than one theme. For example, one student wrote in response to what they liked best about the homework assignments: "They helped me learn the material and gave me practice coding. Also, the room to be creative with assignments was nice." This response was coded into the categories "skill-development" and "creativity". The researcher then went back and re-coded all responses to confirm that the emergent themes were complete and consistent.

3. RESULTS

Results related to creativity, student motivation, user interaction, and overall impact are reported in the subsections below.

3.1 Impact of Creativity

The feature of assignments having creative, open-ended, and flexible components came up as the most popular answer to what students liked best about the homework assignments. Note that students were not told the reasons for this study and this question came before any mention of creativity in the survey questions. Figure 1 shows the top five answers for what students liked best about the homework assignments. 38.6% of students responded that the flexibility, creativity, and open-endedness is what they liked best.

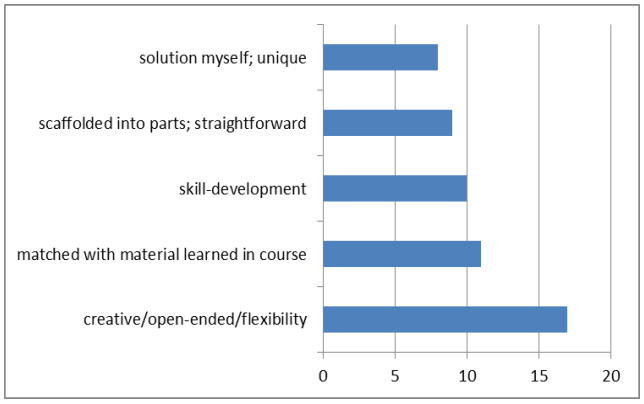

Figure 1: Top 5 Answers: What students liked best about homework assignments

Other reasons that one to two students gave included: wanted to see what CS is like, feeling of accomplishment, additional enrichment in the assignments, they are fun, they have user tests, instructor gives good feedback, can get partial credit, they are challenging, and not too hard.

When asked at the end of the survey if they liked the open-ended nature of the assignments, 41 of the 44 (93.2%) responded that they enjoyed it.

3.2 Impact of Student Motivation

Students were asked via an open-ended question what factors led them to complete homework assignments. Not surprisingly, grades and impact to final grade was the most popular answer. 33 of 44 students said they wanted to get a good grade or the homework scores impacted the final grade. However, only 10 students reported just grades as the only motivator to complete assignments. Other reasons included: interested in how computers work and interest in material (11), wanted to learn concepts (9), wanted to practice programming and improve skills (9), they were fun and satisfying (9), they related to my career (3), and I was capable in completing them (1). In summary, 34 of 44 (77.3%) students had motivations other than grades or in addition to good grades for completing the homework. Students saw value in the task itself (intrinsic) beyond just earning points (extrinsic).

3.3 Impact of User Interaction

User tests were required for HW 2, 3, 4, 5, 8, and 9. Summaries of the user tests were part of the written documentation that accompanied the code submission for each student. Students reported the number of different people with whom they shared their programs over the entire semester. The average number of unique people per student was 3.4, with a minimum of 1 to a maximum of 10. Of those with whom they shared programs, 1.81 people showed an interest in computing. Of the 44 respondents, most shared the programs with friends. The data is as follows: 43 shared with friends, 13 shared with parents, 1 shared with another professor, 10 shared with siblings, and 5 shared with others (girlfriends, housemates, resident assistant, family friend). When they shared their programs, many computing concepts came up during the user tests. Figure 2 shows the self-reported topics (listed in reverse order of presentation in lecture) and number of students who explained that topic when sharing their code.

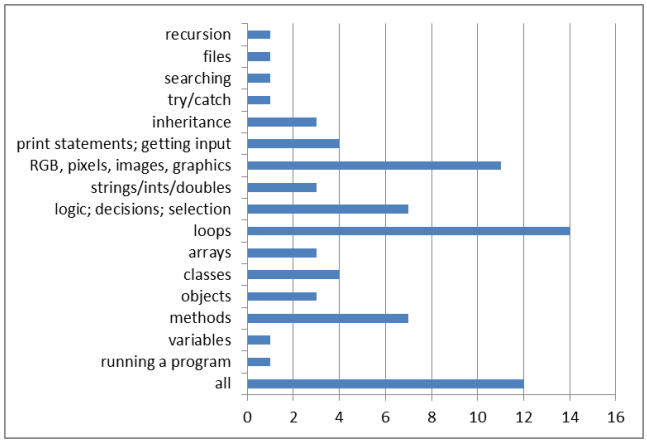

Figure 2: Summary of topics and number of students who described the topics during user tests

Students were also asked if the requirement of sharing a program with a user altered when they started the assignment. The instructor's hypothesis was that students would start assignments that required user tests earlier than assignments that did not require user tests. 34 of 44 (77.3%) responded that sharing the program with another person had no impact on when they started the assignment. Nine (20.5%) responded that they started sooner for those that had user tests and one (2.3%) responded that they started later for those that had user tests. In addition to impact of start time, students were asked if showing their program to another person influenced their effort. 27 of 44 (61.4%) students said it did not influence their effort.

Finally, the students were asked what surprised them when they showed their programs to users. Figure 3 shows the summary of what surprised the CS 1 students during user tests. In some cases, the user helped the CS 1 student make improvements to the program. One student stated, "They seemed to care about things I didn't think they would or they always had suggestions most of which I used." This statement was coded as "improvements". Overall, having users run their programs showcased that people try to input invalid data, people do not read instructions, and testing with other people helped the programmer find bugs.

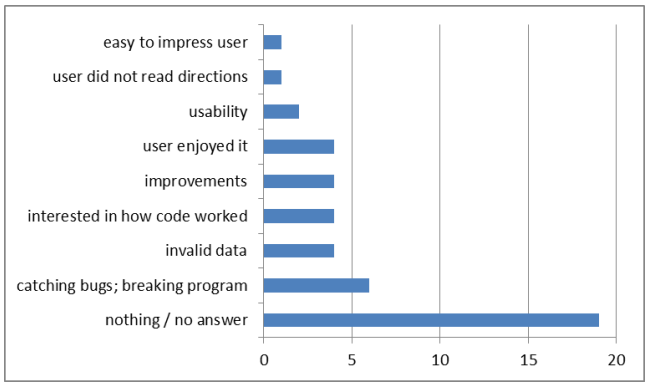

Figure 3: Surprising user test experiences

3.4 Overall Impact in the Course

Overall, students were satisfied with the homework assignments in the course. When asked what could be improved, 15 (34.1%) respondents said nothing – they were already well-designed. No other category of responses had more than four respondents.

Students' favorite assignments were HW 8 (12 respondents), HW 4 (11 respondents), and HW 9 (10 respondents). Creativity in the form of graphical art seems motivating for students. The least favorite assignment was more scattered in terms of responses, but the assignment getting the most votes was HW 6 (11 respondents). HW 6 was a class design-only assignment, and students likely missed the chance to program on this assignment.

In order to assess if students went beyond the explicit homework requirements, students were asked if they completed additional enrichment (optional extensions for no formal credit) during the semester and if they wrote programs that were not assigned as part of the class. 27 (61.3%) completed additional enrichment and 15 (34.1%) wrote programs that were not part of the course (examples include games, sorting numbers, animations, poker game, puzzles, vector experiments, calculators, image transformations, an audio player, and printing random messages.)

Finally, the turn-in rate for homework was high, indicating that students took the course and the assignments seriously. Of 510 possible homework submissions (51 students x 10 assignments), 505 assignments were submitted for grading. The average grades per assignment ranged from 87.2% (HW 10) to 96.0% (HW 1). Even though each homework assignment had open-ended parts, each specification included a grading rubric: 7 points code completion, 7 points code design, and 6 points reflection paper.

4. DISCUSSION

Bias and threats to validity: As with most studies, this study had potential biases in the results. In total, 51 students took CS 203 in spring 2012 and 44 of the 51 gave consent to use their data in the study. It could be that the 44 were not the most representative sample of the students taking the course. Students may have responded more positively in the survey since the instructor had access to the survey responses; however, students were assured that the surveys would be sealed in an envelope until after final grades were submitted. Also, the instructor left the classroom while students completed the survey.

The results regarding creativity indicate that this millennial generation of students may be inspired by the chance to customize and personalize. Over 25% of students stated, without prompting, that the creativity/open-ended nature of the assignments was one of the features they liked best about the homework in the course. Sharing the programs through user tests gave students the opportunity to have share and get feedback about their programs. While the construct of authority or ownership was not explicitly studied or measured, it appears as "I get to do the solution myself and get to do something unique" as the reason eight students offered as to what they liked best about homework.

Although this study presents many of the strengths in giving creative, open-ended programming assignments, this practice does pose challenges. Using creative, open-ended assignments in large classes reduces the amount of automation one can use when grading. At University of Portland, the CS 203 course section size is generally 35 or fewer students, so individual execution of student programs for grading is feasible. If an instructor wants to use automation, perhaps the basic set of features could be graded automatically with scripts and the unique features graded by hand. If done modularly, this approach could work. Another potential drawback of giving students creative freedom is that later in the curriculum, students design and implement software systems to a given specification. However, we decided that getting CS 1 students excited about programming by letting them be creative outweighs the risk of students not conforming to set requirements in later courses. More than half of CS 203 students are not CS majors, so giving them practice in defining programs they want to create and then creating them will serve them well as they use computing as a tool in their careers.

5. CONCLUSION

This paper describes the design aspects of homework assignments and study for a CS 1 course. The design elements include: 1) required and open-ended program specifications, 2) user tests for more than half the assignments, and 3) written documentation that accompanies the code. The results indicate that, not surprisingly, this millennial generation is motivated to complete work for a grade; but, more than that, they appreciate the opportunity to *connect* with users and *connect* their creativity with coding.

6. ACKNOWLEDGMENTS

We thank the introductory computer science students who participated in this study and the course. The author is appreciative of Dr. Karen Eifler, Professor of Education, for fruitful discussions about this work.

7. REFERENCES

[1] Berrett, Dan. 2012. Carnegie, the Founder of the Credit-Hour, Seeks Its Makeover. *The Chronicle of Higher Education.* December 5, 2012.

[2] Elmore, T. 2010. *Generation iY: Our Last Chance to Save Their Future*. Poet Gardener Publishing.

[3] Gonzalez, G. 2004. Constructivism in an introduction to programming course. *Journal of Computing Sciences in Colleges.* 19(4). 299 – 305.

[4] Guzdial, M. 2013. Exploring Hypotheses about Media Computation. In Proceedings of the *International Computing Education Research Conference (ICER).* 19 – 26.

[5] Hovemeyer, D. *et al.* 2014. Using and sharing programming exercises to improve introductory courses. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*. 737.

[6] Kumar, A. Problets – The Home Page. URL: http://problets.org. Last accessed September 4, 2014.

[7] Lin, Y., McKeachie, W., and Kim, Y. 2003. College student intrinsic and/or extrinsic motivation and learning. *Learning and Individual Differences.* 13. 251–258.

[8] Miller, L.D. *et al.* 2014. Integrating Computational and Creative Thinking to Improve Learning and Performance in CS1. In Proceedings of *ACM Special Interest Group for Computer Science Education.* 475 – 480.

[9] Piaget, J. 1950. *The Psychology of Intelligence.* New York: Routledge.

[10] Sherman, M., Bassil, S., Lipman, D., Tuck, N., and Martin, F. 2013. Impact of auto-grading on an introductory computing course. *Journal of Computing Sciences in College.* 28(6). 69 – 75.

[11] Stemler, S. 2001. An Overview of Content Analysis. *Practical Assessment, Research & Evaluation.* 7(17). http://pareonline.net/getvn.asp?v=7&n=17 Last accessed September 4, 2014.

[12] VanDeGrift, T. 2007. Encouraging creativity in introductory computer science programming assignments. In *Proceedings of the American Society for Engineering Education Conference (ASEE).*

[13] Vander Zanden, B., and Berry, M. 2013. Improving automatic code assessment. *Journal of Computing Sciences in College.* 29(2). 162 – 168.

[14] Vygotsky, L. S. 1978. *Mind in society: The development of higher psychological processes.* Cambridge: Harvard University Press. 79 – 91.

Appendix 1: Survey Questions

Overall Feedback

1. Overall, what did you like best about CS203 homework assignments?
2. Overall, what would you suggest to improve the CS203 homework assignments?
3. Here is a list of assignments that you completed for CS203. Put an "F" next to the assignment that was your favorite. Put an "L" next to the assignment that was your least favorite. Include any comments/feedback about your choices below.

Assignment	Topics
Homework 1: Fortune Generator	input, output, arithmetic
Homework 2: Madlib	Strings
Homework 3: Greeting Card	Graphics, conditionals
Homework 4: Golf Simulation	Graphics, loops, conditionals
Homework 5: Wheel of Fortune	Arrays, loops, conditionals
Homework 6: Library Design	Class design
Homework 7: GamePlayer	Class implementation (instance variables, methods, constructors)
Homework 8: Image Transformations	2D Arrays, arrays of objects
Homework 9: Choose Your Own	Inheritance
Homework 10: Palindromes	File I/O, recursion

Comments on choices (if any):

4. What factors motivated you to complete the CS203 assignments?

User Tests

5. Several homework assignments asked you to have a friend execute your program and to solicit his/her feedback.
a. With how many *different* people did you share one of your homework programs? _____
b. Of those with whom you shared your programs, how many showed an interest in computer science or programming? _____

c. For each person below, check the box if you shared at least one of your homework programs with them:
□ a friend (peer), □ a parent/guardian, □ a professor, □ a sibling, □ other _____

d. Did you explain any computer science or programming concepts when you shared your programs with other users? No/Yes
If so, which concept(s) did you explain?

e. Recall that some homework assignments required you to share your program with a friend and some did not. Select the statement that best fits your experience by checking the box.
□ In general, I finished the programs that I shared with a friend earlier (with respect to the due date) than those that I did not share with a friend.
□ In general, I finished the programs that I shared with a friend later (with respect to the due date) than those that I did not share with a friend.
□ Having a friend run my program had no influence on how early I finished my programs with respect to the due dates.

f. Did having a friend run your programs influence your effort or affect the quality of your programs? Explain why or why not.

g. Describe any results from your user tests (friends running your programs) that surprised you.

Open-Ended Assignments

6. Most of the CS203 assignments had an open-ended component that you got to choose to implement. For example, in the image processing homework (HW 8), you designed and implemented an image filter of your choice. In the greeting card homework (HW 3), you designed the greeting card picture and text. In the madlib homework (HW 2), you designed your own story.

a. Did you enjoy the open-ended parts of the homework assignments? No/Yes/Not applicable
b. Why or why not?
c. Most of the assignments had additional enrichment opportunities (not graded for points) to further your knowledge. Did you complete at least one additional enrichment feature during the semester? No/Yes

7. Did you write any Java programs for your own use (not for graded HW in CS203) during the semester? No/Yes
If so, what did the program(s) do?

8. Do you have any other comments about the homework assignments in CS203? If so, put them here: