Introductory Programming in a Web Context

Michael de Raadt
Department of Mathematics and Computing, Centre for Research in Transitional Pedagogies
University of Southern Queensland
Toowoomba, Queensland, 4350

deraadt@usq.edu.au

Abstract

A number of studies have recognised the benefits of using a context or theme consistently throughout an introductory programming course. Examples of contexts in which programming is related and taught include micro-worlds, robotics, games and media computation. Such contexts bring relevance to the content of programming courses. In this paper, a Web context is proposed and described. This context has been successfully used in an introductory programming course and received a positive student response.

Keywords: Introductory programming, context, Web.

1 Introduction

Traditionally, introductory programming has been taught independently of any context; removed from the real world to distil programming to its purest, simplest form. Relevance has been achieved through assignments and practical examples, but on the whole, programming has been presented as an independent practice. A study of assignments in "top" US computer science institutions found, "Only 34% of the CS1 projects had a practical or socially-relevant context, 41% had no context at all..." (Layman et al., 2007, p. 459)

A context can be used consistently through an entire course of programming, in illustrations of programming concepts, practical exercises and assignments. Students can become familiar with the context and see how programming is relevant there. Often students may already be familiar with the context before they begin.

Contexts can be stimulating and exciting and can be relevant to novice programmers' lives, outside of their academic careers, thus providing an incentive to learn about programming with a deep approach. "Engaging students is critical for them to learn something well enough to use it again in a new situation." (Guzdial & Soloway, 2002, p. 18).

At the same time, contexts can be detrimental to students learning. If students learn in a particular domain, it can be difficult for them to transfer their learning to another domain. Within programming, if novices have learned in one context they may see programming only in the related domain. It is therefore important to choose a domain that is relevant to students (Guzdial, 2005) and to demonstrate how concepts learned in the chosen context are relevant in other domains (Guzdial, 2009).

This paper begins with review of contexts that have been used in introductory programming courses, and the effects of using these contexts. A description of a Web context and its use in an introductory programming course is then described. This is followed by possible impact results and measured student attitudes towards this context. Finally, conclusions are made.

1.1 Contexts in Introductory Programming

A number of contexts have been used in introductory programming courses. In this section, some of these contexts are reviewed, and their effects reported. This brief overview is far from comprehensive (as many papers have been written on contexts), but provides a number of examples from each context.

1.1.1 Micro-worlds

The earliest context proposed for teaching programming was used in Logo (Papert, 1970), which focussed on "physical examples" of geometric principles, in a programmable graphical environment with Lisp-like syntax. Seymour Papert believed physical analogies involve students in their learning. "Without this benefit, seeking to "motivate" a scientific idea by drawing an analogy with a physical activity could easily denigrate into another example of "teacher's double talk"" (Papert, 1980, p. 96).

Another example of a micro-world used for programming is that of Karel the Robot (Bergin et al., 2005), who embodies the notion of an object with behaviours in a virtual world. A similar notion is used with Jeroo, which attempts to engage novice programmers on a virtual island (Sanders & Dorn, 2003).

Lister (Lister, 2004) used a micro-world called "Pig's World" to emphasise object-oriented concepts such as message passing and containers, using fun-loving pigs as objects.

A micro-worlds context is useful for teaching, but students may not be able to transfer the relevance of this context to the real world.

1.1.2 Robotics

Using robotics as a context for introductory programming has been successful at a number of institutions. Imberman & Klibaner (2005) report on the use of Lego robots in an introductory programming course. "The drudgery of traditional text based programming assignments was replaced with a "real life" application" (p. 136). They claim a positive student response.

Summet et al. (2009) reported that students studying programming in a robotics context were more successful than students in non-robotics contexts (including media computation and Matlab). Yet, when searching for quantitatively improved student motivation, McWhorter & O'Connor (2009) found little statistical evidence to suggest Lego robots motivated students to learn. Follow-up interviews in this study discovered that students did, however, enjoy working with robots.

The downside to a robotics context is the cost of robots and their availability to students. This context is not convenient when students are studying via distance education.

1.1.3 Games Programming

A number of papers have reported the use of games programming as a context.

Bayliss & Strout (2006) used games programming in an alternate CS1 course and compared student attitudes with those in their traditional course. They found students felt less intimidated by their peers in a games context, Students reported bonding with other students through the development of games.

Haden (2006) reported on the use of a games programming context in a follow-on programming course. Students created simple 2D games, applying object-oriented techniques, physics and recursion. Students were positive about their outcomes in the course. A number of games were exhibited in a public showing and were received with enthusiasm.

The success of a games programming context relies on students having a familiarity with computer games and an interest in producing them, which is not true for all students. There may also be barriers created by the cost of purchasing environments and suitable hardware for games programming. Again this may be a limitation when students are studying via distance education.

1.1.4 Media Computation

A media computation context for programming, sometimes referred to as "media-comp", was originally considered for students from non-computer science backgrounds (Guzdial, 2003). Initial studies of the use of a media-comp context showed improved retention and enthusiasm among students. Media computation has also been shown to encourage greater participation of females (Rich et al., 2004). Media computation involves the manipulation of media such as images and sound files, stimulating creative expression while still covering programming concepts such as iteration and data handling. The success of this context has encouraged wider adoption (Yarosh & Guzdial, 2008).

2 A Web Context

A Web context, put simply, is students writing code which is used in Web pages (JavaScript in HTML pages). The description of a Web context given here relates to Web pages as viewed in a Web browser. It does not attempt to include a client-server model, merely files on the local machine.

This context arose after a change of language in an introductory programming course. Previously this course had used the C programming language and was targeted towards computer science students. After an amalgamation of programs, this course became the single introductory programming course for the university. The mix of students changed also; currently, the greater majority of students in the course will not go on to study further programming, so a strong computer science focus is unnecessary for these students.

A number of languages were proposed to replace C, including Python. JavaScript was chosen as a compromise as it had been used previously in a now defunct course. Despite initial reservations over the limitations of JavaScript, it soon became apparent that this language could be used in a Web context, which has advantages (and some disadvantages) when compared to other contexts used in introductory programming.

2.1 Relevance to Students

It is hardly necessary to define "the Web" here in this paper. Most people in developed countries have a familiarity with the Web. Although students may be naive about how the Web works, they do have an understanding of what Web pages are and the interaction that can take place in them. They are familiar with the purpose of JavaScript, even if they have not heard this name before.

What is most important is for students to see that what they are learning has relevance to themselves and to the real-world. In that sense, a Web context is more potent than any of the contexts mentioned earlier in this paper.

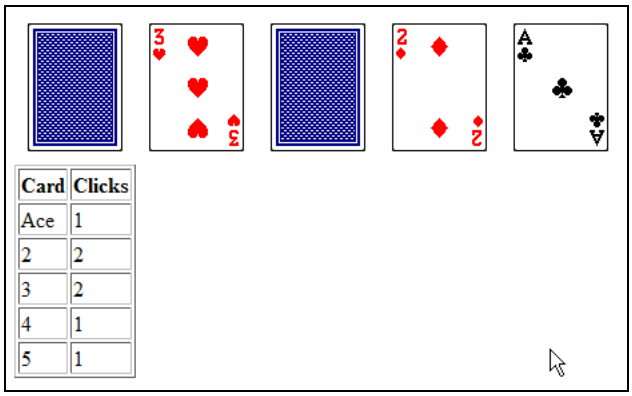

Figure 1. A simple card game in JavaScript

The Web context can be stimulating, reaching beyond text based interaction to graphical user interfaces and interactive programs. Figure 1 shows a simple card game which students created in their final assignment (cards must be revealed in the correct order and will reset on a failed attempt).

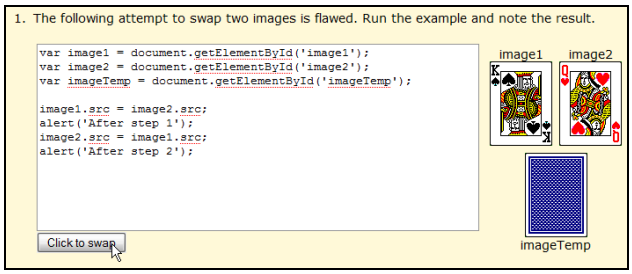

Figure 2. Demonstrating a swap concretely

Even before students can achieve this level of interaction, it is possible to provide examples in materials that use these ideas, without daunting students with complexity. Figure 2 shows an illustration of a triangular swap plan.

One disadvantage of using JavaScript as a programming language is that it must be used in, or with, an HTML document. This means that HTML tag syntax and document structure must be introduced, which creates additional teaching. Persuading students that HTML and the presentation of documents, is not the focus of the course, is also somewhat problematic. Styles and CSS were avoided in the course described here, but could be studied in a course with a different focus.

2.2 Ability to Embed Malleable Examples in Materials

The course described in this paper involved on-campus students (28%) and external (distance) students (72%), which is typical at this university. There is therefore an emphasis on creating materials suitable for both modes.

In the course, all materials were presented online with no paper alternative. The materials were also made available to download and view offline, and recorded onto CD for distribution to external students. Written materials were complemented with short (~5min) recorded video snippets of on-campus didactic teaching. Exercises and examples were intermixed with teaching materials so students would experience concrete examples of code throughout each lesson. This mix of teaching and practice was referred to as a workshop.

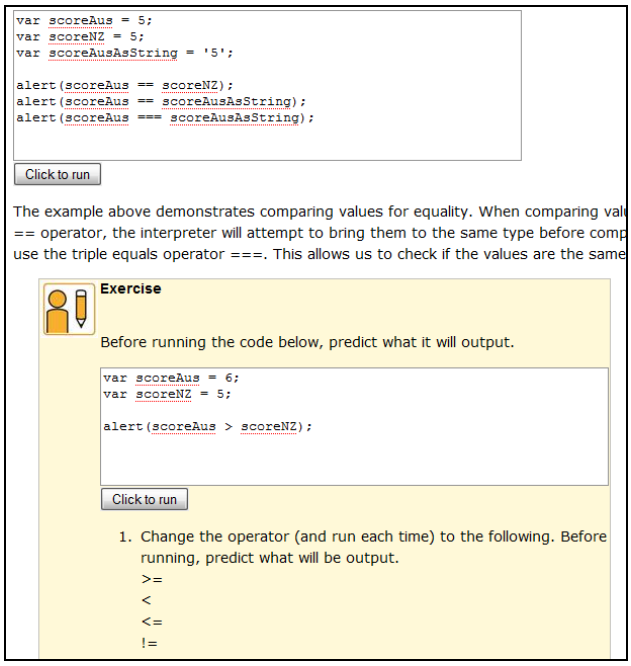

Figure 3. Embedded code examples in course materials (note textareas with code and run button)

The key advantage of this approach was the potential to provide examples of code, embedded in a page, which could be edited and executed. An example of this is shown in Figure 3. All code examples in the course were presented in this manner. Students could manipulate and test the examples immediately, without leaving the learning environment.

These embedded examples are simple to create, making use of the `eval()` function in JavaScript. The text content of a pre-filled textarea can be passed to this function and executed as code, with the same results as normal code. Students can modify the code, on their own or as directed in an exercise, and experience the results of such changes immediately. HTML examples can be achieved in a similar manner. HTML source can be written in a text area and rendered to a section of the document by assigning its `innerHTML` property.

The downside of such embedded examples is that it creates a second way of entering code. Students are also expected to create source code documents in a text editor. The distinction between the two methods of entering code must be explicit when giving students tasks.

2.3 Ability to Teach the Majority of Basic Programming Concepts

JavaScript is not a general purpose language. It is primarily a scripting language used to enhance Web pages. Despite this, most concepts taught in a traditional programming course can still be covered in this context. The following topics were covered in the course described here.

- Programming process, HTML and JavaScript
- Sequence
- Values, Objects, Arrays, Operations, Dynamic typing, Roles of variables
- Expressions, Using functions
- User I/O, String handling
- Programming Strategies (Initialisation, Averaging, Divisibility, Cycle position, Number decomposition, Triangular swap)
- Testing, Debugging, Programming style
- Selection, Iteration
- Programming Strategies (Summing and Counting, Guarded exceptions, Counter controlled loops, Primed sentinel-controlled loops, Validation)
- Writing functions, Recursion
- Programming Strategies (Tallying, Searching, Min/Max, Sorting)
- Interacting with HTML objects, Forms, Events

Topics that are not covered in this course, but can be covered using JavaScript, also include exceptions, creation of objects (paradigm issues are discussed in section 2.4) and possibly more advanced Web interaction through technologies such as Ajax. It is even possible, with perhaps some effort, to achieve media-comp, games and micro-worlds contexts within JavaScript, although going this far may confuse students. In a limited fashion, a games context was used for some later assignments in the course described here.

JavaScript, like other scripting languages, offers a simple typing model. There are three primitive types: numbers, strings and Booleans. Typing is not strict and variables can change their type dynamically.

JavaScript offers a simple I/O model. The `prompt()` function delivers a string input, which is easily converted to the number type as either an integer or floating point. Output can be in the form of simple `alert()` calls,

which pop up a message box, or written to the document body using `document.write()`. Output written to the document body can include HTML tags, so students can create formatted output such as tables and lists, however writing to the document body from a script in the head section can cause confusion for some students. Input and output to a script can be extended to include form elements and images (event driven programming will be discussed in section 2.4).

Because JavaScript is limited to working in a Web browser, it cannot be used to cover the following topics.

- Compilers and libraries
- File I/O
- ADTs and Information hiding

For the majority of students who take this course alone as a brief exposure to programming, these limitations are acceptable. However, transitioning the smaller number of continuing programming students to a general purpose language does require more time and effort than using a single general purpose language through a series of initial programming courses.

2.4 Potential to explore multiple paradigms

JavaScript is a scripting language, however there is also potential to explore other paradigms in the Web context, to a degree that suits the instructor and the course. One benefit of this flexibility is the possibility to start with very simple scripts, then move to more complex programs as the course progresses.

An imperative paradigm can be examined through the creation and use of functions. Functions can be written in a script and called as needed. One downside of writing functions in JavaScript is that they are automatically overloaded. For example, a function that has two specified arguments can be called and supplied zero, one, two or more arguments. It is the responsibility of the programmer to check that sufficient arguments have been provided and to ensure the function reacts accordingly.

Objects can be explored in a simple manner. A number of built in "global" objects are provided in the language, which are used for I/O, arrays, date and time, and mathematical functions. Object-oriented programming can be investigated to a greater depth, however the object model presented in JavaScript is not as clear as in other OO languages. Firstly, there are no classes, only objects, some which can be copied and some which cannot. It is possible to use global objects without copying them, which can cause problems. Functions are objects and primitive types can also be treated as objects.

Graphical user interfaces and event driven programming can be explored through the use of HTML forms and images. This was explored in the last part of the course described here. Some students had trouble understanding a second paradigm. Interacting with HTML elements is not trivial as each has its own set of properties. Care must also be taken when dealing with timeouts as this can result in unwanted parallel sub-processes in a program.

2.5 Consistent environment across platforms

Consistency between browsers is not a great issue with JavaScript. The ECMA standard is followed in almost all browsers. Incompatibilities tend to arise in the use of styles and formatting. The only JavaScript incompatibility that arose in the course was the use of the `const` modifier, which is not supported by Internet Explorer and was therefore avoided (unfortunately). Any script written by students should have worked equally well in all browsers. Students were encouraged to use the Firefox browser as it is available for and consistent across multiple platforms. It can also be extended to support JavaScript development as described in section 2.6.

2.6 Access to error messages and debugger

The Firebug add-on for Firefox includes an error console, debugger and stack tracer. These were particularly useful, right from the start of the course. JavaScript error messages are not perfect, but they are relatively informative and accurate, especially considering the interpreter is relaxed about syntax. Testing and debugging were introduced into the course, which was not possible with previous languages without forcing students to use a specific platform. The stack tracer worked remarkably well and assisted in illustrating recursive function calls without great effort.

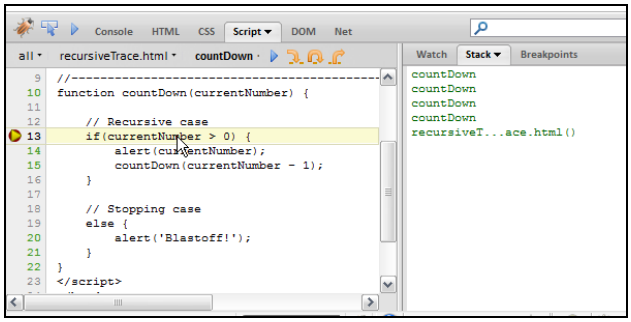

Figure 4. Debugger and Stack Trace

An example of the debugger and a stack trace is shown in Figure 4.

3 Evaluation

To evaluate the Web context after it was used in an introductory programming course, impact on student retention and student attitudes were measured.

Student results were not comparable with previous instances of the course as the student cohort had changed after an amalgamation of programs. Student retention is a major problem in the course being examined here; more students drop out of the course than those who complete the course and fail. Student retention can be seen as removed from student potential so it is possible to examine impact in that regard. Impact was measured by comparing participation in the course with previous instances of the course. Participation was judged by submissions of assignments and the completion of the examination. Results of this comparison are shown in section 3.1.

To measure student attitudes a survey was conducted after the final assignment deadline and before the exam. The anonymous survey contained six questions related to the Web context, including five five-point Likert scale statements and the potential to add a free-form comment. The survey was delivered using a feedback facility of the learning management system used in the course. Students

were encouraged by email to participate, but participation was voluntary. Results of this survey are shown in section 3.2.

3.1 Impact

Participation in the course was measured by counts of assignment and exam submissions. All assignments are submitted electronically in the course, so a count was easily obtained.

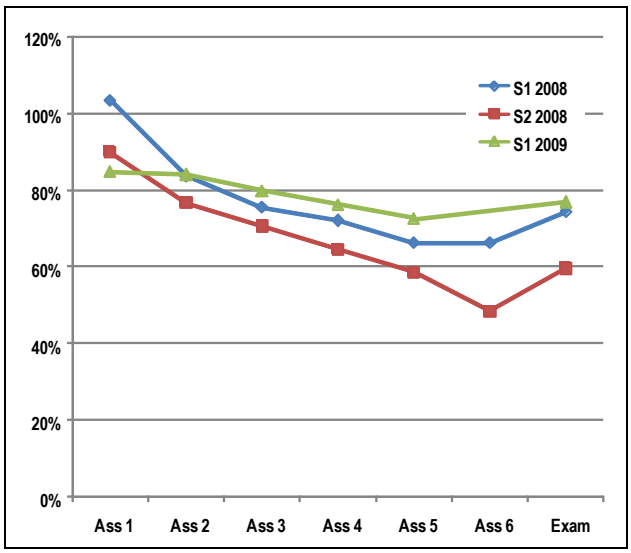

Figure 5. Student retention

Student participation in the exam rose from 74% and 60% in the previous two offerings, to 77% as shown in Figure 5. This is not significant, particularly in light of the irregularity of the previous two semesters. A number of other factors beyond the introduction of a Web context may have affected this result, including the reduction of the number of assignments from six to five, the introduction of weekly quizzes with incentive marks and the use of a new time-management tool for students.

What is interesting to note in Figure 5 is the consistency of participation through the course. It could be argued that students were more engaged.

3.2 Student Attitudes

This section reports on a survey of student attitudes towards a Web context. Seventy-six survey responses were recorded, corresponding to a response rate of 55% when measured against initial enrolments, and to 75% when measured against the total number of active students at the time of the fifth and final assignment. Students were asked what mode they were enrolled in. The responses were 28% from on-campus students and 72% from external students, which was consistent with enrolments in the course.

Instead of using the phrase "Web context" the term "workshop" was used to describe the context through the course and this was continued in the survey.

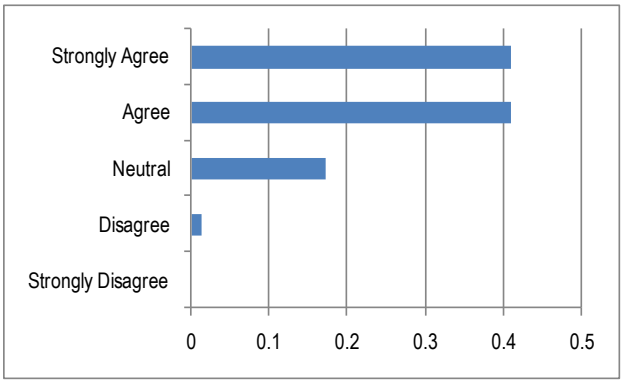

Figure 6. I appreciated the mix of learning and practice in the workshops.

Students were asked about the mix of learning and practice in the course (Figure 6). It is clear that students appreciated the practice they could achieve through the embedded examples that were presented in this context.

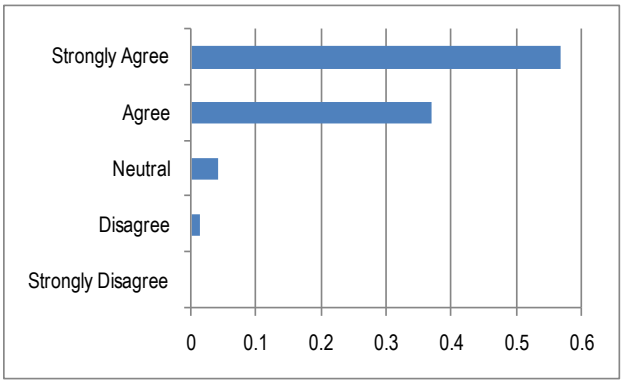

Figure 7. Being able to interact with embedded examples was helpful to my understanding.

When asked if the embedded examples helped their understanding (Figure 7), an even stronger majority agreed that it was helpful. This is a clear indicator that the potential that can be achieved in a Web context is valued by the students.

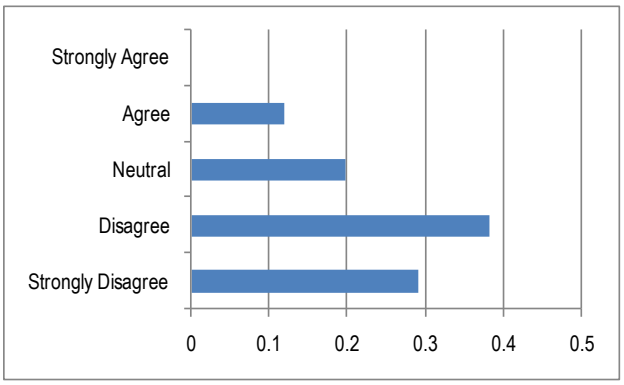

Figure 8. I was confused between when I should be working in an embedded example and when I should be working in my editor.

The majority of initial exercises made use of embedded examples. As the course progressed students were transitioned to writing code in a text editor. One concern with this transition was that asking students to write code in both these forms would cause confusion. The question

(reported in Figure 8) asked students if they had been confused by this change. This was a negatively phrased statement. A majority of students disagreed, thus stating they were not confused. A small proportion (12%) said that they were confused and a number were neutral about this statement. There is still a need to clearly distinguish these two coding activities.

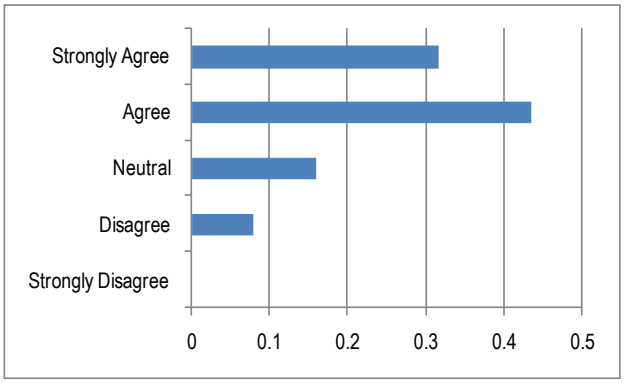

Figure 9. The workshops presented materials that catered for my learning style.

The notion of learning styles had been introduced during the introduction to the course and students had measured their learning style using the VARK questionnaire (Fleming, 2009). The use of the Web context allowed the materials to be presented in visual, aural, read/write and especially kinesthetic modalities, within the same, interactive documents. As shown in Figure 9, 75% of participating students agreed that this mix catered for their learning style.

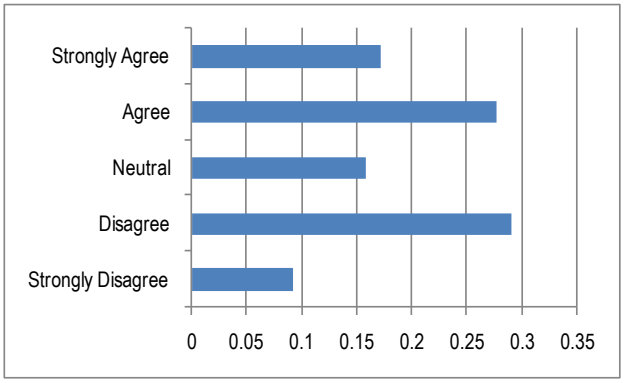

Figure 10. Completing workshops took more time than I normally put into studying the materials of other courses.

The final statement (in Figure 10) asked students if this course required more time than other courses. This received a mixed response (SA+A≈45%, N≈16%, D+SD≈38%). Students in the course come from a variety of disciplines including IT and other sciences, business, engineering and the arts. There is a large part of the cohort that sees this course as requiring more time and work than their other courses (which is probably quite true).

3.2.1 Comments

Students were asked to provide open comments using the prompt "Please feel free to provide comments on the workshops." One of the reasons for asking for comments was to discover if students saw a Web context as relevant. No student specifically stated that it was relevant or irrelevant; it seems they were familiar with this context and merely accepted it. Some students expressed surprised enjoyment in the course, and perhaps this can be attributed to the context. *The course material is well-detailed and so I have had no trouble understanding what is required of me. It is possibly even enjoyable! Wow.*

The majority of comments were positive. A number of students commented on the "format" of the workshops. *The workshop format is how courses of this nature should be laid out.*

Embedded examples were appreciated by students. The most frequently repeated comment related to these examples. *The embedded examples are a great idea as you have the ability to see the code working and also make small modifications to see what the results will be. Enhances learning* (sic).

A number of negative comments provided by students related to workload. *I found the workshops took considerably more time than other subjects, but that I also had a much more thorough understanding of the subject afterwards.* This was consistent with Figure 10 and with student feedback on the course from previous offerings conducted before the introduction of the Web context.

Using a Web context was done, in part, to achieve real-world relevance. Some students, it seems, cannot be distracted from their own discipline. One student commented, *I couldn't relate how I'd need to know so much about computer programming to be a surveyor.*

Perhaps the best perspective was provided by a student who had failed the course in its previous incarnation. *The teaching team has clearly put a lot of work into the preparation of this course. It is truly appreciated... I have previously undertaken this course with the C content and found that to be difficult to follow and understand. Please, please make all IT cou[r]ses like this one.* It should be noted that most of the concepts covered in the course were repeated in the new version of the course; some of the delivery methods changed slightly, but the most significant change was the use of the Web context.

4 Conclusions

Use of a Web context has many advantages over a traditional context-free introductory programming course. Instructors intending to use a context in their introductory programming teaching should consider the Web context, particularly to provide relevance to a cohort from various disciplines. The Web context is well suited for a blended learning environment, providing more immediate kinesthetic interaction than other contexts.

A Web context can be used across platforms with no more than a Web browser and a text editor. It can be used to teach multiple paradigms to varying degrees.

Students are familiar with the Web context. Students appreciate the features made possible by a Web context, particularly embedded examples which allow students to experiment with example code.

It is clear that the use of a context doesn't magically make student's workload disappear, but it may engage and encourage them to participate longer.

5 References

Bayliss, J. D., & Strout, S. (2006): Games as a "flavor" of CS1. *Proceedings of the 37th SIGCSE technical symposium on Computer science education (SIGCSE2006)*, Houston, USA 1-5 March, 2006. 500 - 504.

Bergin, J., Stehlik, M., Roberts, J., & Pattis, R. E. (2005): *Karel J Robot: A Gentle Introduction to the Art of Object-Oriented Programming in Java*. Redwood City, USA, Dream Songs Press.

Fleming, N. D. The VARK Questionnaire, http://www.vark-learn.com/english/page.asp?p=questionnaire. Accessed 17 August 2009.

Guzdial, M. (2003): A media computation course for non-majors. *Proceedings of the 8th annual conference on Innovation and technology in computer science education*, Thessaloniki, Greece. 104 - 108, ACM Press, New York, NY, USA.

Guzdial, M. (2005): Design process for a non-majors computing course. *Proceedings of the 36th SIGCSE technical symposium on Computer science education (SIGCSE2006)*, St. Louis, USA 1-5 March, 2006. 361 - 365.

Guzdial, M. (2009): Contextualized Computing Education of Programming. *Proceedings of the Eleventh Australasian Computing Education Conference (ACE 2009)*, Wellington, New Zealand, 20 - 23, 2009. 3.

Guzdial, M., & Soloway, E. (2002): Teaching the Nintendo generation to Program. *Communications of the ACM*, **45**(4):17 - 21.

Haden, P. (2006): The Incredible Rainbow Spitting Chicken: Teaching Traditional Programming Skills Through Games Programming. *Proceedings of the Eighth Australasian Computing Education Conference (ACE2006)*, Hobart, Australia, January 2006. 81 - 89.

Imberman, S. P., & Klibaner, R. (2005): A robotics lab for CS1. *Journal of Computing Sciences in Colleges*, **21**(2):131 - 137.

Layman, L., Williams, L., & Slaten, K. (2007): Note to self: make assignments meaningful. *Proceedings of the 38th SIGCSE technical symposium on Computer science education (SIGCSE2007)*, Covington, Kentucky, USA 7 - 9 March, 2007. 459 - 463.

Lister, R. (2004): Teaching Java First: Experiments with a Pigs-Early Pedagogy. *Proceedings of the Sixth Australasian Computing Education Conference (ACE2004)*, Dunedin, New Zealand, 18 - 22 January, 2004. 193 - 199.

McWhorter, W. I., & O'Connor, B. C. (2009): Do LEGO® Mindstorms® motivate students in CS1? *Proceedings of the 40th ACM technical symposium on Computer science education (SIGCSE2009)*, Chattanooga, USA 4-7 March, 2009. 438 - 442.

Papert, S. (1970): *Teaching Children Thinking (LOGO Memo)*, Massachusetts Institute of Technology, A.I. Laboratory

Papert, S. (1980): *Mindstorms: Children, Computers, and Powerful Ideas*, Basic Books, Inc., USA.

Rich, L., Perry, H., & Guzdial, M. (2004): A CS1 course designed to address interests of women. *Proceedings of the 35th SIGCSE technical symposium on Computer science education (SIGCSE2004)*, Norfolk, USA March, 2004. 190 - 194.

Sanders, D., & Dorn, B. (2003): Jeroo: a tool for introducing object-oriented programming. *Proceedings of the 34th SIGCSE technical symposium on Computer science education (SIGCSE2003)*, Reno, Navada, USA 19-22 Februrary. 201 - 204.

Summet, J., Kumar, D., O'Hara, K., Walker, D., Ni, L., Blank, D., et al. (2009): Personalizing CS1 with robots. *Proceedings of the 40th ACM technical symposium on Computer science education (SIGCSE2009)*, Chattanooga, USA 4-7 March, 2009. 433 - 437.

Yarosh, S., & Guzdial, M. (2008): Narrating data structures: The role of context in CS2. *Journal of Educational Resources in Computing*, **7**(4):1 - 20.