Computers & Education 82 (2015) 409–420

Contents lists available at ScienceDirect

Computers & Education

journal homepage: www.elsevier.com/locate/compedu

Closing the gender gap in an introductory programming course

Miguel Angel Rubio [a, *], Rocio Romero-Zaliz [a], Carolina Mañoso [b], Angel P. de Madrid [b]

[a] Departamento de Ciencias de la Computación e IA, ETSIIT, University of Granada, C/Periodista Daniel Saucedo Aranda, s/n, E-18071, Granada, Spain
[b] Departamento de Sistemas de Comunicación y Control, ETSI Informática, UNED, C/ Juan del Rosal, 16, E-28040, Madrid, Spain

ARTICLE INFO

Article history:
Received 29 July 2014
Received in revised form
1 December 2014
Accepted 2 December 2014
Available online 10 December 2014

Keywords:
Teaching/learning strategies
Programming and programming languages
Gender studies
Improving classroom teaching

ABSTRACT

Although there is a growing interest in learning to program, the number of women involved in programming remains surprisingly low. We don't understand completely the causes but it has become clear that men and women have different perceptions of programming. The pedagogy of introductory programming courses should take these differences into account. In this study we analyze gender differences in an introductory programming course at the university level. Our results indicate that male and female students have different perceptions and learning outcomes: male students find programming easier, have a higher intention to program in the future and show higher learning outcomes than female students. To reduce these differences we have designed and implemented several learning modules using the principles of physical computing. The physical computing approach aims to take computational concepts out of the screen and into the real world so that students can interact with them. We have applied these modules in a MATLAB introductory programming course in a biology degree. When using these modules both male and female students showed similar results in perceptions and learning outcomes. The use of physical computing principles in combination with the traditional methodology reduced −actually closed− this gender gap.

1. Introduction

In the last years there has been a growing interest in learning and teaching to program (Wortham, 2012). Initiatives like The Hour of Code ("The Hour of Code," n.d.) or Codecademy ("Codecademy: Learn to code," n.d.) have taught programming to thousands of students. Computer science topics have been introduced in the primary-school curriculum in the UK (Brown, Sentance, Crick, & Humphreys, 2014) and New Zealand (Bell, Andreae, & Robins, 2014). New methodological approaches have been developed to help students in these courses. Successful examples are Scratch (Resnick et al., 2009), App Inventor (Wolber, Abelson, Spertus, & Looney, 2011), and Light-Bot ("Light-Bot," n.d.). These learning resources have been shown to improve students outcomes (Goadrich, 2014; Gouws, Bradshaw, & Wentworth, 2013; Guzdial, Ericson, Mcklin, & Engelman, 2014).

There remains one dark spot: the number of women involved in computer science is surprisingly low. In the United States only 0.4 percent of girls entering college intended to major in computer science in 2013 and they made up 14 percent of all computer science graduates, down from 37% in the mid-80s (Alvarado, Dodds, & Libeskind-Hadas, 2012; Patitsas, Craig, & Easterbrook, 2014; Tiku, 2014). Other studies show similarly disproportionate ratios of participation between male and female students in computer science programs (Stoilescu & Egodawatte, 2010). This problem is global: a study conducted on the use of computers and the Internet among fifteen-year olds showed that boys report using computers more often than girls in the vast majority of the 40 countries under investigation (Drabowicz, 2014).

Although we don't understand completely the causes of the differences in participation it has become clear that men and women have different perceptions of programming. Werner, Hanks, and McDowell (2004) analyzed a survey of over 400,000 entering freshman across the US. They found that the gender gap in computer use was almost non-existent but there was a very big confidence gender gap in computer skills. Several authors (Alvarado, Lee, & Gillespie, 2014; Carter & Jenkins, 1999) have found that female students are much less confident in their programming abilities than male students.

* Corresponding author. Tel.: +34 958240466.
E-mail address: marubio@ugr.es (M.A. Rubio).

http://dx.doi.org/10.1016/j.compedu.2014.12.003
0360-1315/

If we want to involve more women in computing, the pedagogy of introductory programming courses needs to change. This is a complex problem and there are no magic bullets but new approaches might help. In 2005 Harvey Mudd College started a three pronged approach: a breadth-first CS1 course with separate tracks according to previous experience, computing research experiences for first-year women and female community building activities. They observed a marked increase of women majoring in computer science (Alvarado et al., 2012). Google has launched a $50 million initiative to teach programming to young girls ("Made with Code," 2014). This initiative includes coding projects, female community building activities and video profiles of women that use programming to solve all kind of problems. In Europe, the European ACM Committee on Women in Computing (ACM-WE) has launched several initiatives to facilitate women participation in computing (Hanson, Ayfer, & Bachmayer, 2014).

One approach that might be effective is contextualized computing. Contextualized computing education is defined as the use of a consistent application or domain area, which effectively covers the core areas of a computer science course (Guzdial, 2010). Examples of contexts for introductory computer science include Media Computation (Guzdial, 2003), traditional manipulatives ("Computer Science Unplugged," n.d.), and robotics (Cuéllar & Pegalajar, 2014).

Students find contextualized approaches to programming very attractive. Instead of writing an abstract program, students can learn about basic programs by programming a robot to exit a maze, animating a story, or creating light symphonies. Rich, Perry, and Guzdial (2004) explored the possibilities of using context teaching to specifically address female students and obtained good results.

One contextualized approach that has attracted increased attention is the physical computing approach. This approach takes the computational concepts "out of the screen" and into the real world so that student can interact with them (Richard, 2008). Several studies have analyzed the feasibility of using physical computing principles in the teaching of computer programming, see for example Ruthmann, Heines, Greher, Laidler, and Saulters (2010).

Male and female students might react differently to physical computing activities. McGill (2012) studied the use of robots in an introductory programming course and found that female students were slightly more intimidated than males by the robots. She also found that female students believed more strongly that using the robots helped them to learn and that it was a pleasure to work with robots.

One alternative approach, used in this study, is to use electronic boards to develop small and simple systems capable of display interesting behaviors (Grasel, Vonnegut, & Dodds, 2010). This approach presents several advantages: the systems are simpler and easier to understand, they are more reliable, show more reproducible behaviors and the overall cost is lower (Hill & Ciccarelli, 2013).

Our aim in this study is to answer the following research questions: is there a gender gap in the traditional introductory programming course? And, if the answer is affirmative, can we use physical computing principles to reduce it? To answer these research questions the following research hypotheses are examined first:

- H_1: Using traditional teaching methods there is no gender difference in the perception on programming.
- H_2: Using traditional teaching methods there is no difference between male and female failure rates.
- H_3: Using physical computing modules there is no gender difference in the perception on programming.
- H_4: Using physical computing modules there is no difference between male and female failure rates.

With this goal in mind we have developed several learning modules based on the physical computing approach. We have used them in an introductory programming course and analyzed the perceptions and learning outcomes of male and female students. As a control we performed the same analysis in another introductory programming course taught with traditional methods.

There have been several studies about gender differences in introductory programming (Murphy et al., 2006; Stoilescu & Egodawatte, 2010) but —to our knowledge— only one study compared the effect of the intervention with a similar group used as control (Sabitzer & Pasterk, 2014).

2. Methods

2.1. Materials

In this study we have developed several learning modules for an introductory programming course at the university level. These modules can be used to teach C/C++, Python or MATLAB covering both compiled languages and interpreted ones. Different course approaches and teaching methodologies might benefit from their use.

One of the first design decisions we had to make was whether to use an electronic board or a robotic platform. Both present several advantages and disadvantages. We decided to work with an electronic board because robotic platforms are more complex and, therefore, their behavior is less reproducible in an educational laboratory (Cuéllar & Pegalajar, 2014). Alvarez and Larranaga (2013) found that factors related to surface friction, battery load or light conditions affected significantly the behavior of the robot and hindered students' work. We heeded McGill's warning that "*potential technical problems should be seriously considered since they can easily negate any potential positive motivational effect*" (McGill, 2012).

We have selected the Arduino microcontroller board (Banzi, 2009) as the development platform. Arduino is an open hardware board that is becoming increasingly common within the teaching community (Grasel et al., 2010; Mellodge & Russell, 2013). One important advantage for our project is that Arduino is a very easy to use board: its first users were artists and designers. Also, thanks to its open-source nature, a wide variety of developers have selected it as a development platform for all kinds of computational systems (Hill & Ciccarelli, 2013).

We designed specific modules for lecture demonstrations and for laboratory sessions (Fig. 1). The contents of the lecture demonstrations and the laboratory sessions are directly related. It is our experience that lecture demonstrations create a desire to learn more about the inner workings of the system shown. We can take advantage of this interest if students find similar activities during the laboratory sessions.

Lecture demonstrations are designed to enhance the traditional teaching methodology, not to replace it. Lecturers will explain a computational concept using the traditional methodology and afterward will reinforce the explanation doing a physical computing demonstration.

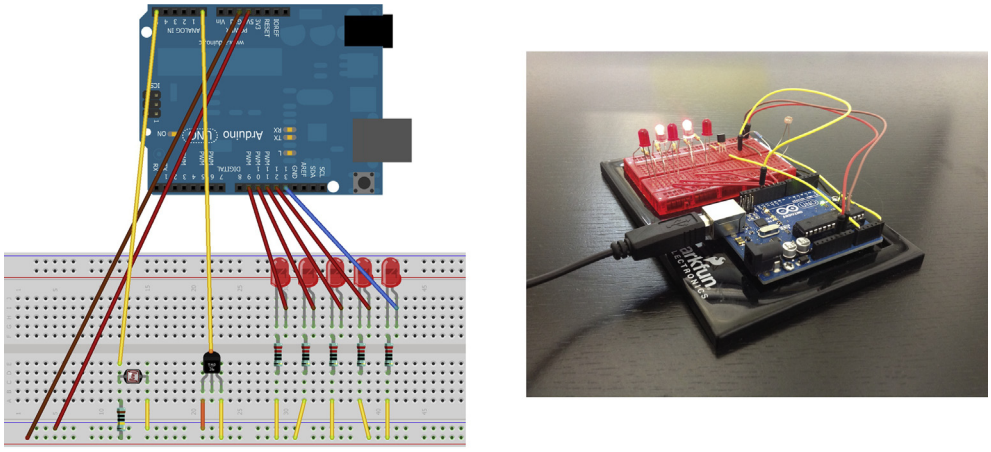

Fig. 1. Electronic circuit created for the laboratory sessions: design (left) and implementation (right). The design shows a photocell (left), a temperature sensor (center) and several LEDs (right).

The lecture modules show different physical examples of computational concepts. Lecture demonstrations use different perceptive elements —light, sound and movement— to reach a broader audience. It's been shown that the use of diverse perceptive paths enhances the student understanding (Ainsworth, 1999). A brief description of selected demonstrations follows:[1]

- We use musical melodies to teach arrays. Following previous studies (Misra, Blank, & Kumar, 2009) we associate different arrays to different melodies. Using this approach, we can explore concepts as arrays concatenation or the difference between the position and the value of an array element.
- Conditional structures are illustrated using a photocell and LEDs. We write during the lecture a small program that will turn on a variable number of LEDs taking as input the ambient light.
- Loop concepts are reinforced using an ultrasonic sensor and a servo motor. During the lecture we implement a program that will continuously read from the proximity sensor. When the value drops below a certain threshold the servo motor and the associated LEDs are activated.

The laboratory sessions allow students to have additional hands-on time with the physical computing system shown in lectures. Several studies show that these activities can greatly improve students' learning (Kay, 2011; Wu, Hsu, Lee, Wang, & Sun, 2014).

After completing the design process we piloted the first version of the learning modules in an introductory programming course. Preliminary results of our experiences were positive and showed that students had a good experience. Students completed a questionnaire and we used their answers to improve the learning material before conducting the study.

2.2. Student demographic

In our study the sample contained 76 university students: 47 women and 29 men. We selected only students that were in their first year, had no previous programming experience and were 18 years old.

By restricting our analysis to freshmen without previous programming knowledge we expect to obtain a clearer picture of the impact of the learning modules on our target population. In a study of students' attitudes towards programming Gomes, Santos, and Mendes (2012) found that freshmen and repeaters differed on personal perceptions and learning approaches.

2.3. Study design

One of the goals of our study was to assess whether men and women have different perceptions and learning outcomes when they learn to program. Another goal was to analyze if the physical computing modules we have designed were effective in reducing these differences.

To this end we used two sections of an introductory programming course in a Biology degree. Students in this course learn basic computing skills and devote ten weeks to learn to program using MATLAB. The course comprises two weekly lectures of 1 h and a 2 h lab session. Students from different sections had different lectures and lab sessions. Students were assigned into one of the two groups either by the university administrative staff or by online registration based on student schedule availability only.

The learning modules were used in one section —the experimental group— and the other section was designated as the control group. In the experimental group the number of students included in the study was 38: 21 women and 17 men. In the control group there were 38 students: 26 women and 12 men. The gender ratios were close to the Biology degree average. In this degree around 60% of students are female.

In the control group the instructor used traditional methods: PowerPoint slides and multimedia material were used to introduce theoretical concepts. Students would also discuss some generic examples using peer instruction techniques (Crouch & Mazur, 2001). In lab sessions students would work individually. In the experimental group the instructor used the physical computing modules in the lectures.

[1] A more detailed description of the modules can be found at http://wdb.ugr.es/~marubio/?page_id=532.

Lecture demonstrations were conducted following Crouch's suggestions to increase student's engagement (Crouch, Fagen, Callan, & Mazur, 2004). In lab sessions students worked in pairs only when completing the physical computing modules.

The same instructor taught both sections back-to-back. He used lesson plans and a course diary to guarantee both courses comparability. The time devoted by the experimental group to work on the learning modules was used by the control group to work on additional examples and exercises.

2.4. Measurements

We used two different measurements to compare the control and experimental group: students' perceptions and learning outcomes. Several studies have established the importance of students' perceptions in their future performances in the STEM disciplines (Valentine, DuBois, & Cooper, 2004).

2.4.1. Students perceptions

There are several instruments capable of measuring students' attitudes towards introductory programming. Each instrument is aimed to a different group of students and aims to measure a different construct. We will describe some of the instruments available and justify our choice.

The Computing Attitudes Survey (CAS) is a newly designed instrument developed by Tew, Dorn, and Schneider (2012). It focuses in the differences in perceptions between novices and experts programmers. The preliminary results obtained by the authors are quite encouraging. We decided not to use this survey in our study because we are teaching to non-majors and we do not expect them to become experts. Additionally, the survey has not finished the validation process and it is not publicly available (Dorn & Tew, 2013).

Another option is the survey developed by Hoegh and Moskal (2009). This survey focuses on high level perceptions about computer science and has been validated with non-major students. In our study we decided not to use it because we were interested only in attitudes specific towards programming.

We chose the TAM model (Davis, 1993) to evaluate students' perception on programming. The TAM model is a powerful tool commonly used to predict the acceptance, adoption and real use of new technologies in production environments. In our case we were interested in assessing whether students had the intention to program in the future.

Pejcinovic, Holtzman, Chrzanowska-Jeske, and Wong (2013) have shown that a significant percentage of students that learn to program in their first university year do not use this knowledge during their studies. Using the TAM model we can estimate the future use of the technology —computer programming in our case— from the user perceptions.

In the TAM model (Fig. 2) the main constructs are the perception of usefulness, the perception of ease of use and the behavioral intention to use. These are defined by Davis (1993) as:

1. Perceived usefulness: the degree to which an individual believes that using a particular system would enhance his or her job performance.
2. Perceived ease of use: the degree to which an individual believes that using a particular system would be free of physical and mental effort.
3. Behavioral intention to use: the degree to which an individual has formulated plans to use a certain system in the future.

The TAM model has been used to estimate the future use of a wide variety of new innovations in information technology. Several studies have applied this model in educational environments (Liu, Chen, Sun, Wible, & Kuo, 2010; Padilla-Meléndez, Garrido-Moreno, & Aguila-Obra, 2008; Selim, 2003). The TAM model has received extensive empirical support (Venkatesh, Morris, Davis, & Davis, 2003) and has been validated in meta-analyses that involved dozens of studies (King & He, 2006; Turner, Kitchenham, Brereton, Charters, & Budgen, 2010).

We designed a questionnaire based on the TAM model to measure students' perceptions. A translation of the survey —the survey was conducted in Spanish-can be found in Appendix A. The surveys used a Likert scale to collect students' opinion using several items for each TAM construct. Students could grade each item with a score ranging from 1 to 5. Students' perceptions were calculated using the mean score of the students' answers to each construct in the TAM model. The surveys were anonymous to reduce any bias in students' answers that may occur if they believed their answers would affect their course grade.

We conducted three surveys model during the course. One survey was conducted at the beginning of the course, the second at the midterm exam and the third at the final exam. The second and third surveys were conducted just after the exams to increase student participation. The third survey of the experimental group contained several additional questions regarding the use of the Arduino board. These questions are listed in Appendix B.

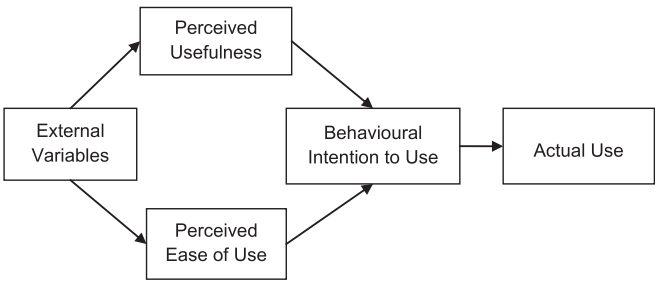

Fig. 2. Original TAM model proposed by Davis (1993).

2.4.2. Learning outcomes

There is a notable lack of easily accessible and validated assessment tools in introductory programming (Tew, 2010). A small set of tools are in development or have been recently developed but they were not applicable to our study.

For example, Tew and Guzdial (2011) have validated a standardized exam —the FCS1— that can be used with different programming languages and methodologies. We could not use the FCS1 exam in our study because the validation results indicate that it is not applicable to courses using contextualized computing methods.

Another promising line of work is the development of concept inventories for introductory programming (Goldman et al., 2010). These concept inventories have been very successful in other scientific fields but no concept inventory aimed to introductory programming has been completed. It remains an open research question if it is possible to identify a set of misconceptions common to the different programming languages used in teaching.

In our study we measured students' learning achievements by means of an exam testing their programming skills. Our exam was designed to assess students' writing and reading skills. It contained three questions that asked to trace and explain code —we summarize them as reading questions— and two questions that asked to write code. Both reading and writing questions contained a combination of conditional and loops. Several studies (Lopez, Whalley, Robbins, & Lister, 2008; Venables, Tan, & Lister, 2009) have found that there is a strong correlation among tracing, explaining and writing code. Students were given 2 h to complete the exam.

The assessment was performed following the guidelines proposed by the BRACElet group (Lister et al., 2010). These guidelines are based on a widely used cognitive taxonomy, the SOLO taxonomy (Lister, Simon, Thompson, Whalley, & Prasad, 2006). This taxonomy describes the type of responses a student may give to a task and it has been validated as a reliable tool to assess introductory programming exams (Clear et al., 2008).

2.5. Analysis performed

2.5.1. Students perceptions

We analyzed students' perceptions in the control and experimental groups. In both groups we compared the men and women perceptions at the end of the course. The analysis was performed applying Student's *t*-test to the different constructs present in the TAM model.

There has been some controversy on the use of parametric methods like the *t*-test with data obtained from Likert scales. It is generally accepted that parametric statistical tests can be applied if we first sum all the Likert items associated to a construct. The sums obtained can be treated as interval data measuring a latent variable (Carifio & Perla, 2008).

2.5.2. Learning outcomes

We used the final exam to measure students' learning outcomes. We analyzed the exam results using clustering techniques: a class of computational methods that has been proved effective in analyzing complex datasets (Brooks, Erickson, Greer, & Gutwin, 2014). Several studies have successfully applied these techniques in the introductory programming context (Bumbacher, Sandes, Deutsch, & Blikstein, 2013; Worsley & Blikstein, 2013).

We clustered the data using the K-Medoids technique, a variation of K-Means clustering where centroids are represented by the median. We have used the Partitioning Around Medoids (PAM) algorithm (Reynolds, Richards, Iglesia, & Rayward-Smith, 2006) implemented in R (R Core Team, 2014).

The first step in this method is to choose the correct number of clusters. The quality of the results depends heavily on this choice: choosing a very large number of clusters reduces the model representative power; choosing a very small number of clusters reduces the accuracy of any given cluster. In our study we have used a subset of the classification scheme proposed by Lahtinen (2007):

- Competent students: Students that have learned all aspects of programming fairly.
- Theoretical students: Students that have learnt to read program code but have difficulties in producing programs on their own.
- Practical students: This group of students had succeeded in writing code and has average reading code skills.
- Unprepared students: Students that lacked reading or writing skills.

We defined the failure rate in a group as the ratio of students in that group that belong to the unprepared students cluster to the total number of students in the group. We used Fisher's exact test to compare the failures rate because the sample size was too small for Student's *t*-test.

In our study we created one data set using the experimental and control measurements and did the clustering on this data set. We were interested in comparing the number of students that belong to each of the clusters and we needed the clusters to be comparable.

3. Results

3.1. Student perception

In our study we conducted three surveys to measure students' perception on programming. We measured the values of the TAM model parameters —perceived usefulness, perceived ease of programming and intention to program— in the control and the experimental group. One survey was conducted at the beginning of the course, the second at the midterm exam and the third at the final exam. We conducted the second and third surveys just after the exams to increase student participation.

We analyzed the survey reliability calculating Cronbach's alpha on the proposed questions within each construct. We obtained in all cases values over 0.7, a quality threshold widely accepted (Santos, 1999).

In Table 1 we show student' initial attitudes toward programming. The experimental and control groups showed small differences but none was statistically significant. From these results we can conclude that both groups were equivalent at the beginning of the course.

Table 1
Student perceptions at the beginning of the programming course.

	Control				Experimental			
	Mean scores		|Δ|	*p*-value	Mean scores		|Δ|	*p*-value
	Male	Female			Male	Female		
Perceived ease of programming	2.47	2.29	0.18	0.409	2.22	2.07	0.15	0.625
Perceived usefulness	4.23	3.96	0.27	0.264	4.00	3.83	0.17	0.435
Intention to program	3.60	3.46	0.14	0.663	3.14	3.32	0.18	0.535

Table 2
Student perceptions at the end of the programming course.

	Control				Experimental			
	Mean scores		|Δ|	*p*-value	Mean scores		|Δ|	*p*-value
	Male	Female			Male	Female		
Perceived ease of programming	3.31	2.48	0.83	0.054[a]	2.87	2.85	0.02	0.961
Perceived usefulness	4.29	3.84	0.45	0.111	3.73	3.70	0.03	0.914
Intention to program	3.89	3.18	0.71	0.054[a]	3.31	3.10	0.21	0.557

[a] Result significant at the 10% level.

To establish the presence of a gender gap in the control group we need to compare males and females' attitudes at the end of the course. These results are collected in Table 2. Male students find programming significantly easier than female students: their perceived ease of programming score is 30% higher, a difference that is statistically significant at the 10% level (p-value = 0.054). They also show a higher intention to program in the future: their future intention to program score is 21% higher (p-value = 0.054). In the case of perceived usefulness the difference is also noticeable (11%), but not statistically significant.

The experimental group shows a different pattern: gender differences in attitudes are much smaller and none is statistically significant. The differences in the perceived ease and usefulness of programming are in both cases less than 1%. The differences in the future intention to program are close to 7%.

We also analyzed the change in students' attitudes during the course. Fig. 3 shows the evolution of the perceived ease of programming. There is a marked difference between male and female attitudes in the control group but these differences disappear in the experimental group. If we focus our attention on the evolution of perceived usefulness (Fig. 4) we don't see any significant change during the course. There is a small systematic difference between males and females in the control group but it is not statistically significant. Fig. 5 shows the evolution of student's intention to program. Males in the control group show a small increase and females show a small decrease, but neither of them is statistically significant. In the experimental group the scores for men and women are similar.

We also measured the perceptions of the students regarding the Arduino board. The results obtained can be found in Table 3. Both male and female students show similar perceptions with no significant differences.

3.2. Learning outcomes

We assessed students learning outcomes analyzing the final exam results. Both control and experimental group completed the same exam at the same time. This eased the analysis as we were able to compare the results directly.

The experimental and control group had equivalent knowledge levels at the beginning of the course as we restricted our analysis to students without any previous programming knowledge.

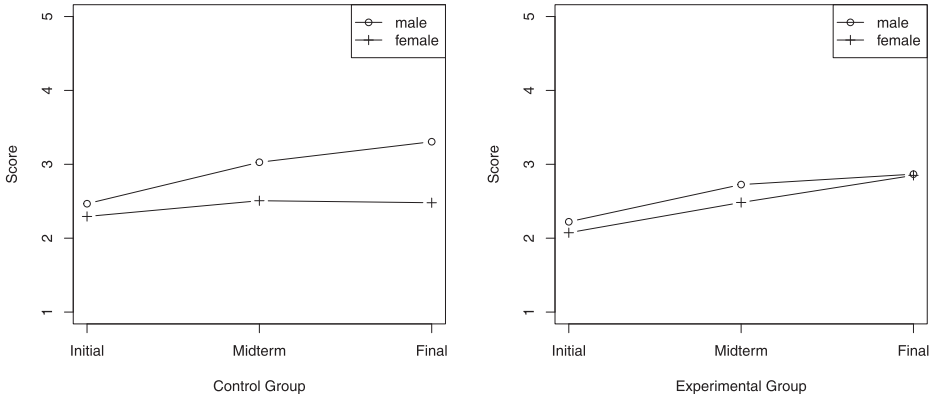

Fig. 3. Changes in perceived ease of programming during the course. At the beginning of the course males and females had similar scores. At the end of the course in the control group they had significant different but in the experimental group males and females showed similar scores.

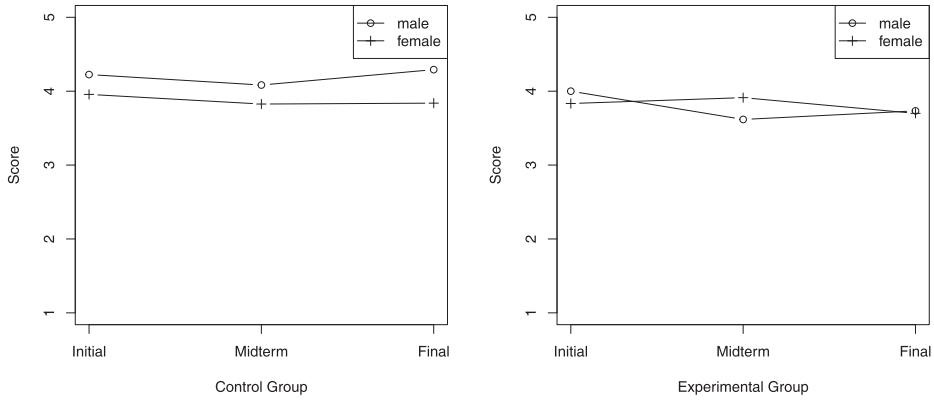

Fig. 4. Changes in perceived usefulness during the course. We can observe a systematic difference between males and females in the control group but it is not statistically significant. The experimental group shows no differences.

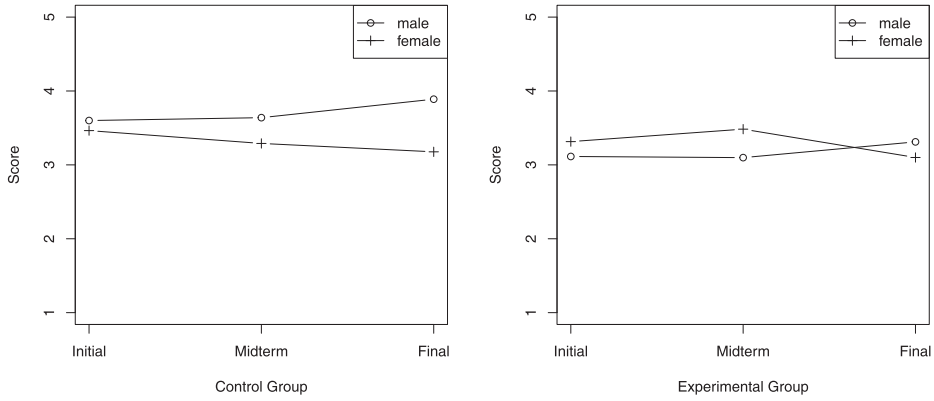

Fig. 5. Changes in students' intention to program during the course. At the beginning of the course males and females had similar scores. At the end of the course the control group showed a significant difference but the experimental group did not.

Cluster analysis of the reading code and writing code scores classified students in four different groups: competent students, theoretical students, practical students and unprepared students. The location of these clusters along the writing-reading dimensions is shown in Fig. 6.

Each group presents different characteristics: unprepared students have low scores for writing and reading code, theoretical students have high reading scores but low writing scores, practical students have average reading and writing scores and competent students have high reading and writing scores.

All the clusters had similar sizes. The most numerous group was the competent students with 23 members. There were similar numbers of practical and unprepared students, 19 for the former and 18 for the later. The smaller group was the one comprised by theoretical students with only 16 members.

One noticeable fact is that the presence of an empty area at the top left corner of the graph. That indicates an absence of students with high writing scores and low reading scores. This makes sense as it would be hard for a student to be able to write meaningful code without the ability to read it.

In our study we define the failure rate as the ratio of students classified as unprepared to the total number of students in each group. Failure rate results are shown in Fig. 7. If we compare failure rates for men and women in the control and experimental group we observe an interesting difference. In the control group females' failure rate, 35%, double males' failure rate, 17%. This difference disappears in the experimental group, there both rates are similar: 19% for women versus 18% for men. The difference in the control group is very suggestive but is not statistically significant, probably due to the small sample size.

Table 3
Student perceptions of the Arduino board.

	Experimental			
	Mean scores		$\|\Delta\|$	p-value
	Male	Female		
Perceived ease of use of the Arduino board	4.10	4.17	0.07	0.823
Perceived usefulness of the Arduino board	3.79	3.48	0.31	0.271
Perceived enjoyment when using the Arduino board	4.69	4.61	0.08	0.779

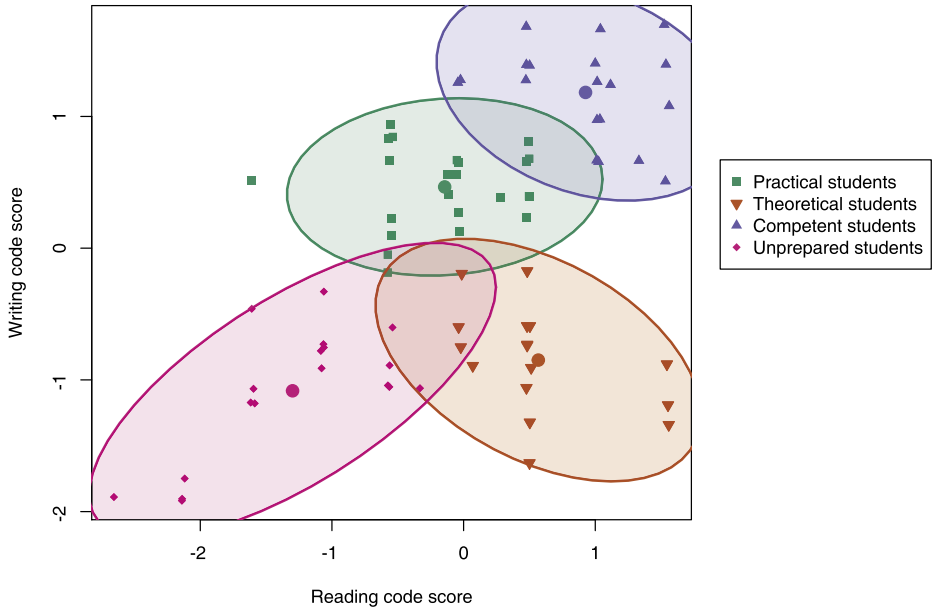

Fig. 6. Clustering of the final exam results. Ellipses represent the normal probability contours at the 90% confidence level. A small amount of jitter has been added to reduce the points overlap.

4. Discussion

We have found differences in perception between men and women in the traditional introductory programming course. The perceived ease of programming and the intention to program in the future were significantly higher in males than in females. These differences in perception allow us to reject the hypothesis H_1. We also found suggestive differences in learning outcomes but these differences were not statistically significant so we cannot reject the hypothesis H_2.

To reduce these differences we have designed and implemented several modules to teach introductory programming using the physical computing approach. These modules comprise lecture demonstrations and laboratory sessions. They aim to enhance the traditional teaching methodology without replacing it.

We evaluated the modules in an introductory programming course and found that they were highly effective. Using these modules the differences in perception between males and females became negligible and no difference in learning outcomes was found. These results strongly support hypotheses H_3 and H_4 and we cannot reject them.

Using these results we can now answer our original research questions. Our first question was: is there a gender gap in the traditional introductory programming course? Our study indicates that the answer is affirmative; there is a significant difference in students'

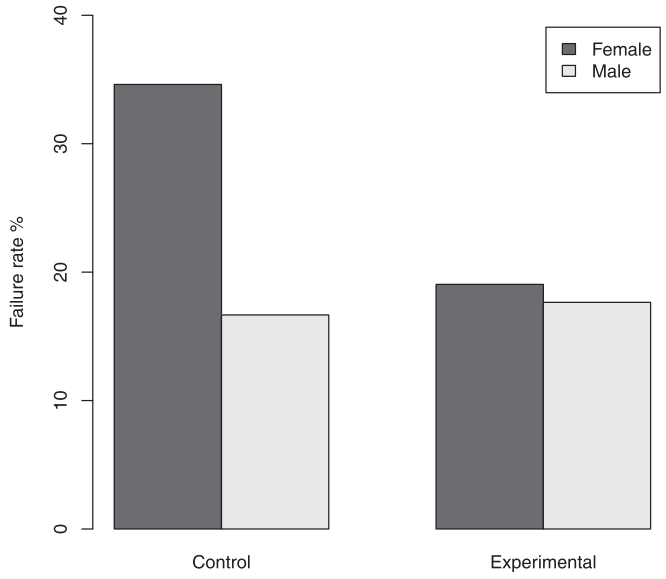

Fig. 7. Programming failure rates. The failure rate for females shows a marked reduction in the experimental group. Males' failure rate shows no such variation.

perceptions. The difference in learning outcomes is also noticeable although it is no statistically significant. Our second question was: can we use physical computing principles to reduce it? Our results indicate that using the physical computing approach reduces this gender gap.

To our knowledge there are no other studies assessing gender differences that use contextualized computing techniques and compare them with a control group. McGill (2012) studied how female and male university students differed in their perceptions of robots but she didn't use a control group.

In our study we also found that all the students able to write code knew how to read code. Similar results have been obtained by other authors (Lister, Fidge, & Teague, 2009; Venables et al., 2009). This suggests that acquiring a certain level of tracing and reading code skills is a first step in the path of learning to write code (Tan & Venables, 2010).

Students found the learning modules useful and highly enjoyable. They perceived them as a valuable learning experience. Students stated that the effort necessary to complete the lab sessions was reasonable and that more laboratory sessions should be devoted to this kind of experiences.

One mechanism that could explain the effectiveness of these modules is that women might find easier to overcome their lack of confidence thanks to the learning modules novelty. McGill (2012) found some interesting differences in the perceptions of boys and girls related to the use of robots and suggested that using robots may give a sense of empowerment to women making them more confident.

Other authors have successfully improved women attitudes using other educational approaches. Sabitzer and Pasterk (2014) developed an introductory programming course based on educational neuroscience principles and found that the gender based differences present at the traditional course disappeared in the new one. Freeman et al. (2014) developed a music-based introductory programming course for high school students and found that it was particularly effective in increasing female motivations to persist on computing problems.

Media computing (Guzdial, 2013) has also been studied extensively with good results. Porter and Simon (2013) found that women in media computing courses expressed greater enjoyment and interest that those attended a traditional introductory programming course.

The differences in perceptions between males and females that we have reported are coherent with other authors' findings. Beyer, Rynes, Perrault, Hay, and Haller (2003) found that women's computer confidence was lower than men's, even when controlling for quantitative ability. Other studies have found gender differences in perceived usefulness, perceived ease of use and intention to use in e-learning environments (Ong & Lai, 2006; Padilla-Meléndez, del Aguila-Obra, & Garrido-Moreno, 2013).

We have found some differences in learning between males and females in the traditional programming course. Other authors have obtained similar results: Sabitzer and Pasterk (2014) obtained significant differences between male and female learning outcomes in a traditional introductory programming course. But these results should be taken with care: in our study they were not statistically significant and it is possible that they are a statistical outlier. Without further studies the differences found in learning outcomes should be taken as suggestive but inconclusive.

Our study has several limitations. The sample size is small as the study comprised only one course and two sections. We think that the results are relevant because the differences were statistically significant. Additionally there is no reason to believe that the results will be different in other scientific and engineering disciplines.

Another possible source of bias is that students in the control group worked always individually and the ones in the experimental group worked by pairs in certain moments. Although the advantages of pair programming have been well established (Williams, Wiebe, Yang, Ferzli, & Miller, 2002), in our case students only worked in pairs in a small number of lab sessions.

The fact that the teacher in charge of both courses was involved in the developing of the learning modules is another limitation. These modules might be less effective when used by teachers that are less familiar with the material. We think this would be unlikely as no specific knowledge —apart from a basic understanding of electric circuits— is needed.

This study needs to be confirmed using data from other degrees in different institutions. We plan to extend this study to see if we obtain similar results. Additionally we will use clustering techniques to analyze the learning outcomes of other courses using the traditional approach to see if the differences we have found are reproducible.

5. Conclusions

We have found differences between male and female students in a traditional introductory programming course. In the traditional setting we found that men and women differ significantly in their perception of the ease of programming and their intention to program in the future. We also found suggestive differences in learning outcomes, although these differences were not statistically significant.

To reduce these differences we have designed and implemented several learning modules using the principles of physical computing. These modules can be easily integrated within the existing methodologies and they can be used in lectures and laboratory sessions.

We evaluated the modules in an introductory programming course and found that they were highly effective. Using these modules the differences in perception and learning outcomes between men and women disappeared.

Learning to program is incredibly useful and will become even more important in the future. As a consequence there is a growing interest in learning to program and programming courses are being introduced in schools. The fact that computer literacy rates for women are dropping is a worrisome fact. Active strategies should be conducted to try to reverse this trend.

In our study the use of the physical computing approach has reduced —actually closed— this gender gap. If these results can be reproduced in other institutions it could increase the number of women interested in computing.

Acknowledgment

This work is supported by the University of Granada (PID/13-54). The authors wish to thank the Biology degree students at UGR that participated in the experience. The authors would also like to thank the reviewer number 1 for his/her insightful comments.

Appendix A

Questionnaire items used in this study by construct.

Do you agree or disagree with the following statements?

	Totally agree	Neither agree nor disagree		Totally disagree	
Perceived Usefulness of programming					
Knowing how to program will help me find a job.	5	4	3	2	1
It will be easier to finish my studies if I know how to program.	5	4	3	2	1
During my studies it will be useful for me to know how to program.	5	4	3	2	1
Knowing how to program will be useful for my work.	5	4	3	2	1
Perceived Ease of programming					
It is easy for me to learn how to program.	5	4	3	2	1
It is easy to make the program do what I want to.	5	4	3	2	1
Programing is easy.	5	4	3	2	1
Intention to program					
I intend to use programing during my studies.	5	4	3	2	1
In my job I will code programs that will be helpful for me.	5	4	3	2	1
Programing will form part of my profession.	5	4	3	2	1

Appendix B

Questionnaire items related to the use of the Arduino board by construct.
Do you agree or disagree with the following statements?

	Totally agree	Neither agree nor disagree		Totally disagree	
Perceived Usefulness of the Arduino board					
When programing with Arduino I learn more.	5	4	3	2	1
With Arduino I learn better how to program.	5	4	3	2	1
Arduino helps me understand programming.	5	4	3	2	1
I learn more quickly when working with Arduino.	5	4	3	2	1
Perceived Ease of Use of the Arduino board					
It is easy to work with Arduino.	5	4	3	2	1
Arduino is easy to use.	5	4	3	2	1
It is easy to program the Arduino board.	5	4	3	2	1
Perceived Enjoyment when using the Arduino board					
Lab sessions with Arduino are more entertaining.	5	4	3	2	1
I have fun working with Arduino.	5	4	3	2	1
Working with Arduino is more enjoyable.	5	4	3	2	1

References

Ainsworth, S. (1999). The functions of multiple representations. *Computers & Education, 33*(2–3), 131–152. http://dx.doi.org/10.1016/S0360-1315(99)00029-9.

Alvarado, C., Dodds, Z., & Libeskind-Hadas, R. (2012). Increasing women's participation in computing at Harvey Mudd College. *ACM Inroads, 3*(4), 55–64. http://dx.doi.org/10.1145/2381083.2381100.

Alvarado, C., Lee, C. B., & Gillespie, G. (2014). New CS1 pedagogies and curriculum, the same success factors?. In *Proceedings of the 45th ACM technical symposium on computer science education* (pp. 379–384) New York, NY, USA: ACM. http://dx.doi.org/10.1145/2538862.2538897.

Alvarez, A., & Larranaga, M. (2013). Using LEGO mindstorms to engage students on algorithm design. In *Frontiers in education conference, 2013 IEEE* (pp. 1346–1351). http://dx.doi.org/10.1109/FIE.2013.6685052.

Banzi, M. (2009). *Getting started with arduino*. O'Reilly Media, Inc.

Bell, T., Andreae, P., & Robins, A. (2014). A case study of the introduction of computer science in NZ schools. *ACM Transactions on Computing Education, 14*(2), 10:1–10:31. http://dx.doi.org/10.1145/2602485.

Beyer, S., Rynes, K., Perrault, J., Hay, K., & Haller, S. (2003). Gender differences in computer science students. *ACM SIGCSE Bulletin, 35*, 49–53.

Brooks, C., Erickson, G., Greer, J., & Gutwin, C. (2014). Modelling and quantifying the behaviours of students in lecture capture environments. *Computers & Education, 75*(0), 282–292. http://dx.doi.org/10.1016/j.compedu.2014.03.002.

Brown, N. C. C., Sentance, S., Crick, T., & Humphreys, S. (2014). Restart: the resurgence of computer science in UK schools. *ACM Transactions on Computing Education, 14*(2), 9:1–9:22. http://dx.doi.org/10.1145/2602484.

Bumbacher, E., Sandes, A., Deutsch, A., & Blikstein, P. (2013). Student coding styles as predictors of help-seeking behavior. In H. C. Lane, K. Yacef, J. Mostow, & P. Pavlik (Eds.), *Artificial intelligence in education* (Vol. 7926, pp. 856–859). Berlin Heidelberg: Springer. http://dx.doi.org/10.1007/978-3-642-39112-5_130.

Carifio, J., & Perla, R. (2008). Resolving the 50-year debate around using and misusing Likert scales. *Medical Education, 42*(12), 1150–1152. http://dx.doi.org/10.1111/j.1365-2923.2008.03172.x.

Carter, J., & Jenkins, T. (1999). Gender and programming: what's going on? *SIGCSE Bulletin, 31*(3), 1–4. http://dx.doi.org/10.1145/384267.305824.

Clear, T., Whalley, J., Lister, R., Carbone, A., Hu, M., Sheard, J., et al. (2008). Reliably classifying novice programmer exam response using the SOLO taxonomy. In *21st annual NACCQ conference, NACCQ, Auckland, New Zealand* (pp. 23–30).

Codecademy: Learn to code. (n.d.). Retrieved 7–25, 2014, from http://www.codecademy.com/.

Computer Science Unplugged. (n.d.). Retrieved April 2014, from http://csunplugged.org.

Crouch, C., Fagen, A. P., Callan, J. P., & Mazur, E. (2004). Classroom demonstrations: learning tools or entertainment? *American Journal of Physics, 72*(6), 835–838. http://dx.doi.org/10.1119/1.1707018.

Crouch, C., & Mazur, E. (2001). Peer Instruction: ten years of experience and results. *American Journal of Physics, 69*(9), 970–977. http://dx.doi.org/10.1119/1.1374249.

Cuéllar, M. P., & Pegalajar, M. C. (2014). Design and implementation of intelligent systems with LEGO Mindstorms for undergraduate computer engineers. *Computer Applications in Engineering Education, 22*(1), 153–166. http://dx.doi.org/10.1002/cae.20541.

Davis, F. D. (1993). User acceptance of information technology: system characteristics, user perceptions and behavioral impacts. *International Journal of Man-Machine Studies, 38*(3), 475–487. http://dx.doi.org/10.1006/imms.1993.1022.

Dorn, B., & Tew, A. E. (2013). Becoming experts: measuring attitude development in introductory computer science. In *Proceeding of the 44th ACM technical symposium on computer science education* (pp. 183–188). New York, NY, USA: ACM. http://dx.doi.org/10.1145/2445196.2445252.

Drabowicz, T. (2014). Gender and digital usage inequality among adolescents: a comparative study of 39 countries. *Computers & Education, 74*(0), 98–111. http://dx.doi.org/10.1016/j.compedu.2014.01.016.

Freeman, J., Magerko, B., McKlin, T., Reilly, M., Permar, J., Summers, C., et al. (2014). Engaging underrepresented groups in high school introductory computing through computational remixing with EarSketch. In *Proceedings of the 45th ACM technical symposium on computer science education* (pp. 85–90).
Goadrich, M. (2014). Incorporating tangible computing devices into CS1. *Journal of Computing Sciences in Colleges, 29*(5), 23–31. Retrieved from http://dl.acm.org/citation.cfm?id=2600623.2600627.
Goldman, K., Gross, P., Heeren, C., Herman, G. L., Kaczmarczyk, L., Loui, M. C., et al. (2010). Setting the scope of concept inventories for introductory computing subjects. *ACM Transactions on Computing Education, 10*(2), 5:1–5:29. http://dx.doi.org/10.1145/1789934.1789935.
Gomes, A. J., Santos, A. N., & Mendes, A. J. (2012). A study on students' behaviours and attitudes towards learning to program. In *Proceedings of the 17th ACM annual conference on innovation and technology in computer science education* (pp. 132–137). New York, NY, USA: ACM. http://dx.doi.org/10.1145/2325296.2325331.
Gouws, L. A., Bradshaw, K., & Wentworth, P. (2013). Computational thinking in educational activities: an evaluation of the educational game light-bot. In *Proceedings of the 18th ACM conference on innovation and technology in computer science education* (pp. 10–15). New York, NY, USA: ACM. http://dx.doi.org/10.1145/2462476.2466518.
Grasel, J., Vonnegut, W., & Dodds, Z. (2010). Bitwise biology: crossdisciplinary physical computing atop the arduino. In *2010 AAAI spring symposium series*.
Guzdial, M. (2003). A Media computation course for non-majors. *SIGCSE Bulletin, 35*(3), 104–108. http://dx.doi.org/10.1145/961290.961542.
Guzdial, M. (2010). Does contextualized computing education help? *ACM Inroads, 1*(4), 4–6. http://dx.doi.org/10.1145/1869746.1869747.
Guzdial, M. (2013). Exploring hypotheses about media computation. In *Proceedings of the Ninth annual international ACM conference on international computing education research* (pp. 19–26). New York, NY, USA: ACM. http://dx.doi.org/10.1145/2493394.2493397.
Guzdial, M., Ericson, B., Mcklin, T., & Engelman, S. (2014). Georgia computes! An intervention in a US State, with formal and informal education in a policy context. *ACM Transactions on Computing Education, 14*(2), 13:1–13:29. http://dx.doi.org/10.1145/2602488.
Hanson, V., Ayfer, R., & Bachmayer, B. (2014). European women in computing. *Communications of the ACM, 57*(7). http://dx.doi.org/10.1145/2631183, 5–5.
Hill, L., & Ciccarelli, S. (2013). Using a low-cost open source hardware development platform in teaching young students programming skills. In *Proceedings of the 14th annual ACM SIGITE conference on information technology education* (pp. 63–68). New York, NY, USA: ACM. http://dx.doi.org/10.1145/2512276.2512289.
Hoegh, A., & Moskal, B. M. (2009). Examining science and engineering students' attitudes toward computer science. In *Frontiers in education conference, 2009. FIE'09. 39th IEEE* (pp. 1–6). http://dx.doi.org/10.1109/FIE.2009.5350836.
Kay, J. S. (2011). Contextualized approaches to introductory computer science: the key to making computer science relevant or simply bait and switch?. In *Proceedings of the 42Nd ACM technical symposium on computer science education* (pp. 177–182) New York, NY, USA: ACM. http://dx.doi.org/10.1145/1953163.1953219.
King, W. R., & He, J. (2006). A meta-analysis of the technology acceptance model. *Information & Management, 43*(6), 740–755. http://dx.doi.org/10.1016/j.im.2006.05.003.
Lahtinen, E. (2007). A categorization of Novice Programmers: a cluster analysis study. In *Proceedings of the 19th annual workshop of the psychology of programming interest group, Joensuu, Finnland* (pp. 32–41).
Light-Bot. (n.d.). Retrieved 18–11, 2014, from http://lightbot.com/.
Lister, R., Clear, T., Simon, Bouvier, D. J., Carter, P., Eckerdal, A., et al. (2010). Naturally occurring data as research instrument: analyzing examination responses to study the novice programmer. *SIGCSE Bulletin, 41*(4), 156–173. http://dx.doi.org/10.1145/1709424.1709460.
Lister, R., Fidge, C., & Teague, D. (2009). Further evidence of a relationship between explaining, tracing and writing skills in introductory programming. *SIGCSE Bulletin, 41*(3), 161–165. http://dx.doi.org/10.1145/1595496.1562930.
Lister, R., Simon, B., Thompson, E., Whalley, J. L., & Prasad, C. (2006). Not seeing the forest for the trees: novice programmers and the SOLO taxonomy. *SIGCSE Bulletin, 38*(3), 118–122. http://dx.doi.org/10.1145/1140123.1140157.
Liu, I.-F., Chen, M. C., Sun, Y. S., Wible, D., & Kuo, C.-H. (2010). Extending the TAM model to explore the factors that affect intention to use an online learning community. *Computers & Education, 54*(2), 600–610. http://dx.doi.org/10.1016/j.compedu.2009.09.009.
Lopez, M., Whalley, J., Robbins, P., & Lister, R. (2008). Relationships between reading, tracing and writing skills in introductory programming. In *Proceedings of the Fourth international workshop on computing education research* (pp. 101–112). New York, NY, USA: ACM. http://dx.doi.org/10.1145/1404520.1404531.
Made with Code. (2014). Retrieved 7–25, 2014, from https://www.madewithcode.com/.
McGill, M. M. (2012). Learning to program with personal robots: influences on student motivation. *ACM Transactions on Computing Education, 12*(1), 4:1–4:32. http://dx.doi.org/10.1145/2133797.2133801.
Mellodge, P., & Russell, I. (2013). Using the arduino platform to enhance student learning experiences. In *Proceedings of the 18th ACM conference on innovation and technology in computer science education*. New York, NY, USA: ACM. http://dx.doi.org/10.1145/2462476.2466530 (pp. 338–338).
Misra, A., Blank, D., & Kumar, D. (2009). A music context for teaching introductory computing. *SIGCSE Bulletin, 41*(3), 248–252. http://dx.doi.org/10.1145/1595496.1562955.
Murphy, L., Richards, B., McCauley, R., Morrison, B. B., Westbrook, S., & Fossum, T. (2006). Women catch up: gender differences in learning programming concepts. *SIGCSE Bulletin, 38*(1), 17–21. http://dx.doi.org/10.1145/1124706.1121350.
Ong, C.-S., & Lai, J.-Y. (2006). Gender differences in perceptions and relationships among dominants of e-learning acceptance. *Computers in Human Behavior, 22*(5), 816–829. http://dx.doi.org/10.1016/j.chb.2004.03.006.
Padilla-Meléndez, A., del Aguila-Obra, A. R., & Garrido-Moreno, A. (2013). Perceived playfulness, gender differences and technology acceptance model in a blended learning scenario. *Computers & Education, 63*(0), 306–317. http://dx.doi.org/10.1016/j.compedu.2012.12.014.
Padilla-Meléndez, A., Garrido-Moreno, A., & Aguila-Obra, A. R. D. (2008). Factors affecting e-collaboration technology use among management students. *Computers & Education, 51*(2), 609–623. http://dx.doi.org/10.1016/j.compedu.2007.06.013.
Patitsas, E., Craig, M., & Easterbrook, S. (2014). A historical examination of the social factors affecting female participation in computing. In *Proceedings of the 2014 conference on innovation & technology in computer science education* (pp. 111–116). New York, NY, USA: ACM. http://dx.doi.org/10.1145/2591708.2591731.
Pejcinovic, B., Holtzman, M., Chrzanowska-Jeske, M., & Wong, P. K. (2013). Just because we teach it does not mean they use it: case of programming skills. In *Frontiers in education conference, 2013 IEEE* (pp. 1287–1289). http://dx.doi.org/10.1109/FIE.2013.6685038.
Porter, L., & Simon, B. (2013). Retaining nearly one-third more majors with a trio of instructional best practices in CS1. In *Proceeding of the 44th ACM technical symposium on computer science education* (pp. 165–170). New York, NY, USA: ACM. http://dx.doi.org/10.1145/2445196.2445248.
R Core Team. (2014). R: a language and environment for statistical computing. Retrieved from http://www.R-project.org/.
Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., et al. (2009). Scratch: programming for all. *Communications of the ACM, 52*(11), 60–67. http://dx.doi.org/10.1145/1592761.1592779.
Reynolds, A. P., Richards, G., Iglesia, B., & Rayward-Smith, V. J. (2006). Clustering rules: a comparison of partitioning and hierarchical clustering algorithms. *Journal of Mathematical Modelling and Algorithms, 5*(4), 475–504. http://dx.doi.org/10.1007/s10852-005-9022-1.
Rich, L., Perry, H., & Guzdial, M. (2004). A CS1 course designed to address interests of women. *SIGCSE Bulletin, 36*(1), 190–194. http://dx.doi.org/10.1145/1028174.971370.
Richard, G. T. (2008). Employing physical computing in education: how teachers and students utilized physical computing to develop embodied and tangible learning objects. *The International Journal of Technology, Knowledge and Society, 4*(3), 93–102.
Ruthmann, A., Heines, J. M., Greher, G. R., Laidler, P., & Saulters, I. C. (2010). Teaching computational thinking through musical live coding in scratch. In *Proceedings of the 41st ACM technical symposium on computer science education* (pp. 351–355). New York, NY, USA: ACM. http://dx.doi.org/10.1145/1734263.1734384.
Sabitzer, B., & Pasterk, S. (2014). Brain-based programming continued. In *Frontiers in education conference, 2014 IEEE* (pp. 1495–1500).
Santos, J. R. A. (1999). Cronbach's alpha: a tool for assessing the reliability of scales. *Journal of Extension, 37*(2), 1–5.
Selim, H. M. (2003). An empirical investigation of student acceptance of course websites. *Computers & Education, 40*(4), 343–360. http://dx.doi.org/10.1016/S0360-1315(02. http://dx.doi.org/10.1016/S0360-1315(02)00142-2.
Stoilescu, D., & Egodawatte, G. (2010). Gender differences in the use of computers, programming, and peer interactions in computer science classrooms. *Computer Science Education, 20*(4), 283–300. http://dx.doi.org/10.1080/08993408.2010.527691.
Tan, G., & Venables, A. (2010). Wearing the assessment "BRACElet". *Journal of Information Technology Education: Innovations in Practice, 9*(1), 25–34. Retrieved from http://www.editlib.org/p/111692.
Tew, A. E. (2010). *Assessing fundamental introductory computing concept knowledge in a language independent manner*. Atlanta, GA, USA: Georgia Institute of Technology.
Tew, A. E., Dorn, B., & Schneider, O. (2012). Toward a validated computing attitudes survey. In *Proceedings of the Ninth annual international conference on international computing education research* (pp. 135–142). New York, NY, USA: ACM. http://dx.doi.org/10.1145/2361276.2361303.
Tew, A. E., & Guzdial, M. (2011). The FCS1: a language independent assessment of CS1 knowledge. In *Proceedings of the 42Nd ACM technical symposium on computer science education* (pp. 111–116). New York, NY, USA: ACM. http://dx.doi.org/10.1145/1953163.1953200.
The Hour of Code. (n.d.). Retrieved 7–17, 2014, from http://csedweek.org/.
Tiku, N. (2014). How to get girls into coding. *New York Times*. Retrieved from http://www.nytimes.com/2014/06/01/opinion/sunday/how-to-get-girls-into-coding.html.
Turner, M., Kitchenham, B., Brereton, P., Charters, S., & Budgen, D. (2010). Does the technology acceptance model predict actual use? A systematic literature review. *Information and Software Technology, 52*(5), 463–479. http://dx.doi.org/10.1016/j.infsof.2009.11.005.

Valentine, J. C., DuBois, D. L., & Cooper, H. (2004). The relation between self-beliefs and academic achievement: a meta-analytic review. *Educational Psychologist, 39*(2), 111–133. http://dx.doi.org/10.1207/s15326985ep3902_3.

Venables, A., Tan, G., & Lister, R. (2009). A closer look at tracing, explaining and code writing skills in the novice programmer. In *Proceedings of the Fifth international workshop on computing education research workshop* (pp. 117–128). New York, NY, USA: ACM. http://dx.doi.org/10.1145/1584322.1584336.

Venkatesh, V., Morris, M. G., Davis, G. B., & Davis, F. D. (2003). User acceptance of information technology: toward a unified view. *MIS Quarterly, 27*(3), 425–478. Retrieved from http://dl.acm.org/citation.cfm?id=2017197.2017202.

Werner, L. L., Hanks, B., & McDowell, C. (2004). Pair-programming helps female computer science students. *Journal on Educational Resources in Computing, 4*(1). http://dx.doi.org/10.1145/1060071.1060075.

Williams, L., Wiebe, E., Yang, K., Ferzli, M., & Miller, C. (2002). In support of pair programming in the introductory computer science course. *Computer Science Education, 12*(3), 197–212.

Wolber, D., Abelson, H., Spertus, E., & Looney, L. (2011). *App inventor.* O'Reilly Media, Inc.

Worsley, M., & Blikstein, P. (2013). Programming pathways: a technique for analyzing novice programmers' learning trajectories. In H. C. Lane, K. Yacef, J. Mostow, & P. Pavlik (Eds.), *Artificial intelligence in education* (Vol. 7926, pp. 844–847). Berlin Heidelberg: Springer. http://dx.doi.org/10.1007/978-3-642-39112-5_127.

Wortham, J. (2012). A surge in learning the language of the internet. *New York Times*. Retrieved from http://www.nytimes.com/2012/03/28/technology/for-an-edge-on-the-internet-computer-code-gains-a-following.html.

Wu, H.-T., Hsu, P.-C., Lee, C.-Y., Wang, H.-J., & Sun, C.-K. (2014). The impact of supplementary hands-on practice on learning in introductory computer science course for freshmen. *Computers & Education, 70*(0), 1–8. http://dx.doi.org/10.1016/j.compedu.2013.08.002.