Engaging students in Learning Activities

Carlton McDonald, University of Derby, Kedleston Road, Derby, DE22 1GB, UK

Abstract

Teaching programming has been a major challenge for decades. It seems that student engagement is an additional problem that must be overcome. Many students only engage in what they know will be assessed. Other "directed study", they ignore, not recognising the foundation these other activities provide enable them to do well when assessed. What techniques might be used to enthuse students to participate in the activities required to learn to be effective programmers?

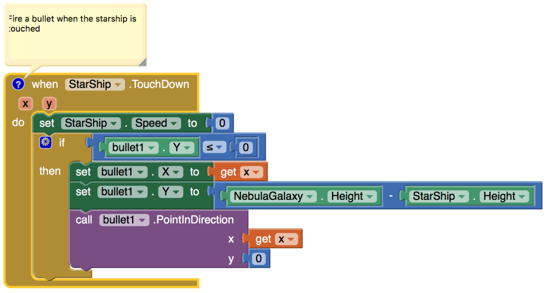
Keywords
Continuous assessment, formative feedback, POOT, MEQ, AppInventor, JavaScript, Java Applets, Java Applications, Student motivation

Teaching 1st year programming is a major challenge at all universities. It doesn't seem to matter what programming language is used, how much support is provided to the students, or how the students are assessed. learning to program is hard enough as it is.[1] This paper looks at the experience of teaching programming to IT students on two courses: BSc IT and BSc Information Technology Management for Business. We look at the teaching methods, assessment methods, student engagement patterns in terms of attendance at lectures (1 hour per week) and practicals (3 hours per week).

The module concerned is an introduction to programming for two separate courses of students. The BSc IT students do not learn any other programming languages on their course although they are expected to apply programming knowledge to SAS development in the final year of the course. The BSc Information Technology Management for Business is an introduction to programming that will be followed up at level 5 with a "Systems Development" module. The title of the module is Programming Principles for the IT students and Introduction to Systems Development. From this point on the module will be referred to as Programming Principles. The content is primarily:

- 5 weeks of AppInventor
- 3 weeks of JavaScript
- 3 weeks of Java Programming

AppInventor[2] is a graphical drag and drop, cloud-based programming environment, that enables complete beginners (including young children) to program Android phones. AppInventor has graphical representations for basic building blocks which are assembled in jigsaw puzzle style assembly such that the high level structure of the program can be designed directly from programming blocks:

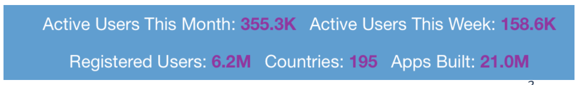


```
Fire a bullet when the starship is touched

when StarShip .TouchDown
  x  y
do  set StarShip . Speed to 0
    if   bullet1 . Y ≤ 0
    then set bullet1 . X to get x
         set bullet1 . Y to NebulaGalaxy . Height - StarShip . Height
         call bullet1 .PointInDirection
                                   x get x
                                   y 0
```

Figure 1 Fire Bullet code for Starship game

AppInventor usage statistics are pretty impressive:



Active Users This Month: 355.3K Active Users This Week: 158.6K
Registered Users: 6.2M Countries: 195 Apps Built: 21.0M

Figure 2 AppInventor User Statistics, 31st March 2017[3]

Now used all over the world, AppInventor has made programming available to children as well as adults. The reasons the author chose to utilise AppInventor are

1. To motivate the student to engage with programming.
2. To help them to get quick successes visible on their devices.

3. To present programming concepts in a graphical, colourful environment.
4. To integrate User Interface Design and Programming as essential companions.

Assessment Strategy
The assessment is 100% coursework, 5 portfolio demonstration exercises 10% each, and a small development project making up the remaining 50%. The portfolio exercises were to be demonstrated fortnightly from week 3 in order to give students an opportunity to benefit from formative assessment and early feedback on their progress in the module. The first 3 demonstrations were AppInventor Exercises:

1. Follow a flipped learning video tutorial and complete some exercises on creating a mobile music player App called GhettoBlaster.
2. Write an App which utilises a number of screens to calculate a user's longevity utilising different UI components and taking a user forward and backward through a number of screens. Calculations affecting longevity included amount of cigarettes smoked, units of alcohol consumed, exercise, geographical location of residence etc.
3. Given the skeleton of a space ship game where the ship is able to move left and right and a single enemy appears, modify the game to have multiple enemies fall, the space ship shoot bullets, detect collisions, modify scores and no. of lives left.
4. Demonstrate 5 exercises from 10 on using Java Script to: generate dice values; generate random numbers for a lottery; respond to events onLoad() and onUnload() on a webpage; load a document after pressing a button; and check a web form prior to submission to a server.
5. Modify a Java Application to draw Circles and Squares that are red, green or blue such that the application is able to draw triangles, that are also orange and yellow and the whole application is converted to an Applet for execution, as if it were in a browser environment.

Motivating Students
Both groups of students had a lecture at 9:00 am on Tuesday, one group had their 3 hour practical immediately afterward, the other had the 3 hour practical 3 days later on Friday morning. The average attendance over 10 of the 12 weeks is 28% of the students for the lecture; for the IT class the average attendance at the practicals was 43%, for ITMB the average attendance was 66%.

Session	**IT**	**ITMB**	**Total**
Lecture	5.4 (30%)	3.2 (32%)	8.1 (28%)
Practical	7.7 (43%)	6.6 (66%)	14.3 (51%)
Average	**6.6 (37%)**	**4.9 (49%)**	**11.2 (40%)**

Figure 3 Average Attendance by class

These figures are extremely low, however when the software demonstrations were required most of the students turned up. When tuition took place in computer laboratory practicals, during the weeks between the assessed software demonstrations, fewer of the students turned up. When they appeared in the practical during the assessed demonstrations many of the students started the work of the demonstration, rather than demonstrate immediately. This indicated that they were not working outside of class time. This is something that has been observed by staff for a number of years.

As can be seen from Figure 4, the average first year student is attending this programming principles module 59% (BSc IT) and 51% (BSc ITMB). A significant factor in the very low average student attendance is the 9:00 am start for the lecture on Tuesday mornings. The figures in the table don't capture the figures at 9:00 or even 9:15 when some students are still turning up at 9:30.

Session	**IT**	**ITMB**
Lecture (x10)	5.4 (30%)	3.18 (32%)

Demo due x 5	4.3 (86%)	3.3 (66%)
Practical x 5	3.5 (68%)	2.8 (56%)
Average	**13.2 (61%)**	**9.3 (51%)**

Figure 4 Average Attendance by Student

"Research conducted over the past few decades shows it's impossible for students to take in and process all the information presented during a typical lecture, and yet this is one of the primary ways college students are taught, particularly in introductory courses."[4] Impossible? That may be true, however, the lecture is an opportunity to ask questions, complete exercises in class, work with others at solving problems. The approach in the lectures is not solely to "lecture"[5] A variety of activities are provided so that all students are able to recognise a learning activity that matches their preferred learning style. However, when asked to write solutions on paper and discuss them in small groups only one or two students have paper or pens with them. This lack of preparedness wastes time. University students should not need to be told to bring paper and pens to a class.

Those students that attend the least, achieve the lowest grades, there is a positive correlation between attendance and achievement. "Attendance was shown to be a key indicator significantly correlated with high school graduation".[6] "lack of attendance not only affected individual students academically, but also affected the learning environment of the entire school".[7]

Poor Attendance is Due to Poor Teaching?
The lack of engagement is interpreted in some management circles as a commentary on the quality of the teaching rather than a measure of the commitment of the students. Those students that attend and engage in the learning activities achieve exceptionally good results. Some universities expend great energy in improving the quality of the teaching and this is indeed important but only if the students are required to participate in those learning activities. Indeed, at the institution where this programming principles module is delivered the Module Evaluation Questionnaire (MEQ) asks for students comments on the quality of the tutor's ability to explain, the organisation of the module etc. One particular student missed the second demonstration of software on the Tuesday (week 5), didn't come on the Friday and sought an extension. When not permitted, and told that he and four others in his position needed to "get their act together" emailed: "I find it quite insulting that you think that we haven't got our act together as a course". When the final demonstration was due in week 11, this student arrived two hours late and asks questions, the tutor pointed to the whiteboard and indicated that had he "got his act together" and arrived on time, he would have heard the explanation and been able to ask questions and seek clarification. It was also pointed out that the student had not reflected on the statement made at the time:

```
"Every week, the only thing the
whole class does is prepare for the
demo, you do NOT do the other
tutorial exercises designed to
help you understand – so tell me
how it is possible to understand if
you do not go through the exercises
planned to help you understand?

"Your statement that you feel
insulted reveals a lack of
responsibility for your selective
approach to learning.  If you
continue this throughout the
course you will not be a strong IT
professional.  If you only do
assessments and not tutorial
exercises you let yourself
down.  If someone tells you that
you haven't got your act together
and you think you have you will not
change anything – its up to you.

"One of the most important things
in life is to be self critical".
```

Figure 5 Excerpt of Email to a Student

On the day when he was two hours late, this same student said he could not arrive earlier that day but provided no reason as to why not. To be fair to this student, his attendance was better than most of his classmates, turning up to 14 out of 20 opportunities. Nevertheless, his attitude toward attendance and feeling quite insulted for having it pointed out that his attendance and attitude required improvement is a cause for concern. Learning is a relationship between tutor and student. Tutors should be able to improve following student feedback but students too need to respond to lecturer feedback.

One final comment on student engagement: "Engaging students throughout their educational process has become the greatest challenge facing educators today".[8]

At the University concerned, student attendance is monitored with little real consequences for the students on a module by module basis because students are not removed from modules but only from the entire programme.

MEQs
Module Evaluation Questionnaires (MEQ) are not effective measures of student feedback as they only give a single perspective: that of students on the tutor and module, and not of the student on themselves. One student turned up to a single session in week 6, when asked why, he replied that he was "too busy". This student would have been able to complete a MEQ, nevertheless, it is far more important that we seek to determine what is causing students to not engage in learning activities, not just those that are assessed. On the MEQ for this module, a number of students commented that the tutor was not good at explaining things. This feedback would have been quite disheartening to the tutor had there not have been an observation of a lecture by a colleague.

POOT
A colleague from the English department performed a peer observation of teaching (POOT) that all staff are to have each year. The lecturer despite never having any tuition in Computing, let alone programming, having observed only one lecture said, "the explanations were very clear" and she found it quite interesting. The topic happened to be the hardest of all and was an introduction to Object Oriented Programming in Java. Although the observer was selected by the tutor, the purpose was to share a way of interacting with students on challenging subject matter, so it was surprising, knowing how difficult OOP is, that an English lecturer found it clear and understandable. The expectation was that not having any computing background the English lecturer would not really be able to comment on the content.

This does however indicate that MEQs should capture, or provide, some of the students' data, e.g. what percentage of classes have been attended? How many of the learning activities have been engaged with? How many hours per week is the student working and earning money at vocations unrelated to the course?

Learning Activities
Students, that do not engage in directed learning activities are unlikely to establish a sound foundation of knowledge on which they are able to build through the course of a module. As a result, when they feedback that the tutor is not good at explaining things it is more of an indication of their engagement than the tutor's ability to explain things, unless the student has attended every lesson and engaged in all directed study activities. The reason for this is that if only one activity was not engaged with, a related subsequent topic relying on the foundation would be much more difficult to understand. Imagine learning

multiplication without first learning addition, it doesn't matter how good the teacher, the quality of the explanation of multiplication requires some earlier knowledge.

Continuous Assessment

A portfolio of assessment in programming has been the approach taken in introductory programming rather than a big programming assignment. However, the approach in this module is not to have all the portfolio exercises demonstrated at the end of the semester. This is because when demonstrations of the assignments were conducted at an overseas partner, some of the students had working solutions but didn't understand what they presented. Quirks in solutions would reappear in other students' solutions. When asked to explain these quirks only the originator could explain why it/they were there. It was clear that some students were passing the module having not really understood much as they went along having been assisted with a solution from their classmates. In this module students did assist each other to get their solution working. The tutor recognising that it was at the point of an hiatus that students were being assisted mostly permitted this learning, although at times the solution was within their grasp and they would be required to work it out for them selves based on what they had done so far.

The approach in this module was to have the portfolio work demonstrated on a fortnightly basis. Students would therefore have to attend in order to demonstrate their learning and participation in learning activities. Students must either do the work themselves in their own time, or in the laboratory sessions.

Although it was clear that the majority of the students were not working very much outside of class time, the tutor was complicit in permitting students to work on the demonstration a week in advance of the demonstration and during the session when the demonstration was required. The tutor's preference would have been for the students to work on practical exercises that would enhance their foundation knowledge and skills some of which were to be assessed the following week. It meant that at times 10 or 15 small exercises were provided but the students only did the those that were being assessed the following week. The tutor felt that a more rounded experience would have been achieved by doing all the exercises as there were some nuances captured by an exercise that might not be assessed, yet, at some point in the student's career, if not on the course, that required that particular programming nuance, the student would been prepared had the student engaged in all the activities.

The tutor resigned himself to the fact that students have to manage their time, and such selective learning has been the approach of students at all universities and all levels of education. Students learn what they care about and remember what they understand.[9] The students care about the assessments points not the learning opportunities.

Student Achievement

Each demonstration was worth 10 marks, all five being 50% of the module outcome. No student that engaged in all demonstrations achieved less than 56%, i.e. 28 out of 50. The other 50 marks coming from an assignment to be handed in at the end of the module.

No of Demos done	*n*	Average %
5	16	39 (78%)
4	4	28 (56%)
3	6	21 (42%)
2	1	15 (30%)
1	1	3.5 (7%)

Figure 6 Student Achievement from 5 Demonstrations

A really interesting statistic is that of the students completing all demonstrations, all but one achieved a mark of more than 70%.

Grade	*n*
1st (70 - 100)	15
2:1(60 – 69)	0
2:2(50 – 59)	1

Figure 7 Grades of those engaging in all demonstrations

The student that achieved 56% failed the first two demonstrations i.e. achieved marks less than 4 but worked hard and improved his engagement in the module. This feedback of marks early on, provides students reassurance that they are doing okay, or need to improve their performance. Students, with few exceptions, did better on later assignments than earlier assignments.

Demo No.	Average Mark
1	6.1
2	5.4
3	4.8
4	6.9
5	7.9

The formative feedback inspired students on a whole, and this is much better than taking in all the portfolio exercises at the end, in that the knowledge gained from engaging in activities, especially in computer programming, is built upon throughout the module.

Learning to Program

There are a few researchers that refer to "incremental learning" for Machine learning algorithms and Inductive Programming[10],[11] There isn't an easy accessible body of research demonstrating that learning to program is an incremental activity. This author always states this point, in the first class, by saying: "if you do not understand what happens in week 1 you have little chance in week 2. If you do not understand what happens in Week 2 you have no chance in week 3, and so on". Students learning to program cannot approach programming as they approach essay writing, often leaving it until the week or two before the deadline. After all, no computer software project manager would wait until a week or two before the deadline to ensure that the project is planned and that milestones are agreed. Computer projects as assignments are no different. The project must be planned. This required students, all of whom had to select a project using any of the technologies presented on the module: AppInventor for writing Android software, Javascript for writing browser software and Java for writing Applications or Applets. The order of presentation of topics was designed to enable all students to take on an AppInventor project early. Some students considered JavaScript and Java but as they were not going to be covered until later in the course, all students selected AppInventor projects. They were therefore required by the end of Week 5, to provide a proposed title and discuss what they intended to do in their project. Although it took an additional couple of weeks for some students to do this (because it wasn't assessed), a title and a few statements of what they planned to do on a weekly basis was provided and was reviewed by the tutor. This enabled their plans to be verified as achievable, at the appropriate level. Once they were agreed, these plans enabled progress to be monitored before the due date.

Portfolio Exercises

Having observed the way that students prepare their portfolios by preparing the exercises that are to be submitted at the end of a module, it was clear that some students were completing the portfolio exercises on the day they were to be assessed. Having learnt the topics two or 3 months earlier, they were, often quickly, trying to complete the program in order to receive the marks. The process of completing early portfolio exercises and receiving feedback that would inform later submissions and learning was lost. This was the main reason marks improved throughout the course of study: students were benefitting from feedback from the tutor.

Conclusion

An Introduction to learning computer programming has revealed some issues to do with student selective engagement because students are focussed almost solely on what will be assessed. This appears to be because they are managing their time due to having to juggle external commitments. Continuous assessment has the benefit of reinforcing learning at regular intervals throughout the module so that progress and feedback are achieved with plenty of opportunity to respond for subsequent assessments. Students that engage in all learning activities achieve consistently high grades, and their progress improves throughout the module. A portfolio of exercises taken in at the end of a module is not as effective as exercises performed throughout a programming course because have the benefit of early feedback and some students treat the portfolio as something that can be completed in the last couple of weeks of the course.

References

[1] Chris Pine (2009) Learn to Program (2nd Edition), The Pragmatic Programmers, **ISBN:** 978-1-93435-636-4

[2] http://appinventor.mit.edu/explore/

[3] http://appinventor.mit.edu/explore/ [Accessed March 31st, 2017]

[4] Emily Hanford, http://americanradioworks.publicradio.org/features/tomorrows-college/lectures/rethinking-teaching.html

[5] Richard M. Felder, Linda K. Silverman (1988) Learning and Teaching Styles In Engineering Education, North Carolina State University, Institute for the Study of Advanced Development, Engr. Education, 78(7), 674–681.

[6] Allensworth, E., and Easton, J.Q. (2005). *The On-Track Indicator as a Predictor of High School Graduation*. Chicago: Consortium on Chicago School Research.

[7] UK Essays. (2013). Correlation Between Students Attendance And Achievement Education Essay. [online]. Available from: https://www.ukessays.com/essays/education/correlation-between-students-attendance-and-achievement-education-essay.php?cref=1 [Accessed 2 April 2017].

[8] Rosario Hernández, (2016). Student Engagement in Assessment for Learning International Conference on Engaging Pedagogy, Maynooth University. [online]. Available from: http://icep.ie/wp-content/uploads/2010/01/Hernandez.pdf [accessed 31/03/2017]

[9] Ericksen, S. (1984). *The essence of good teaching*. San Francisco: Jossey-Bass, p51 cited in Charles C. Bonwell, Ph.D. Active Learning Workshops [online]. Available from: http://www.ydae.purdue.edu/lct/hbcu/documents/Active_Learning_Creating_Excitement_in_the_Classroom.pdf [accessed 31/03/2017]

[10] Robert Henderson, (2009), Incremental Learning in Inductive Programming, 3rd Workshop on Approaches and Applications of Inductive Programming, in conjunction with 14th ACM SIPLAN International Conference on Functional Programming (ICFP 2009), Edinburgh, Scotland [online]. Available from: http://www.cogsys.wiai.uni-bamberg.de/aaip09/aaip09_submissions/incremental.pdf [accessed 31/03/2017]

[11] Christophe Giraud-Carrier (2000) A Note on the Utility of Incremental Learning, AI Communications, Volume 13 Issue 4, Pages 215 – 223.