A case study on using iPads to encourage collaborative learning in an undergraduate web development class

Aekaterini Mavri, Fernando Loizides, Nicos Souleles
Cyprus University of Technology, Cyprus

Abstract
Since its release in 2010 the iPad has become the leading computer tablet in the market. Due to its popularity, the iPad is widely used in different higher education (HE) institutions around the world. However, the research on the use of this tablet in HE remains limited. This paper describes how the iPad was used in an undergraduate web development class to encourage collaboration through participatory exercises. Firstly, the students were provided with the tablet in order to contribute to the class exercise, and then the iPad display was streamed real-time to a projector. The device was passed from student to student, thus bypassing the need for them to individually walk up to the lecturer's computer. This instructional approach also eliminated the use of laptops or workstations, and encouraged collaborative and active learning. Data was gathered through surveys and interviews with participating students, and a mixed methods approach was applied to the analysis. This paper reports on the user experience and the perceived learning outcomes, as well as the advantages and disadvantages of using this instructional approach for the specific lesson.

Keywords
active learning, collaborative learning, programming, web development

1. Introduction

General pedagogy and educational psychology research shows that active learning can facilitate knowledge assimilation more effectively than traditional lecture-based teaching approaches (Felder, Stice & Rugarcia, 2000). Furthermore, active collaboration is an effective component for student engagement (Kuh, Cruce, Kinzie & Gonyea, 2008) associated with positive learning outcomes (Diemer, Fernandez & Streepey, 2012). Active learning implies that students are 'actively engaged in the learning process' (Prince, 2004). In short, it involves students in 'learning through doing' while 'thinking about what they are doing in a classroom' (Bonwell & Eison, 1991), as opposed to passively listening. It implies moving away from teaching transmission towards knowledge assimilation through participatory activity, and aims to enhance student motivation through 'higher order thinking' (analysis, synthesis and evaluation) (Prince, 2004).

Amongst the various methods that make use of the active learning model is mobile learning. Current studies show that mobile computing is found to increase student motivation and engagement in the learning activity (Swan, Kratcoski & Unger, 2005). They also highlight the collaborative potential of mobile devices, especially when producing or editing work (Traxler, 2005). Although the practice of mobile learning looks at the use of handheld devices, both inside and outside the classroom, this study focuses on a particular tablet device, the iPad, as the means for incorporating

an active learning method into the classroom. The reasons behind this choice lie, mostly, in the fact that most design education institutions rely and run on Apple Mac technology (Souleles, Savva, Watters, Bull & Annesley, 2014). Similarly, the Department of Multimedia and Graphic Arts at the Cyprus University of Technology provides a wide range of Apple Mac computers and peripherals for use by the students in its labs. As a result students are well acquainted with the Mac OS environment and user interface as well as other native or related software.

The researchers were interested in investigating the potential of learning using the affordances of a portable device (in this case the iPad) in a four-week long undergraduate web development unit. The study was implemented in collaboration between the Cyprus Interaction lab and the Networked Learning Technologies in Art and Design research lab in the same department. Its aims were to elicit information and report on the student experience in using the iPad as part of a collaborative active learning approach as well as the perceived impact of this method on their learning outcomes. Data were collected by means of empirical assessment (survey and focus groups) and peer observation and analysed using a mixed methods approach. Results indicate that the active learning method steered excitement, participation and productive collaborative activity during class. It was also perceived to enhance motivation and lead to better comprehension of theoretical concepts. Usage of the device also received mixed responses due to its lightweight and convenient use both in and out of the classroom. Nonetheless, serious problems were reported in producing code through its touchscreen interface and the process was thought to be more effectively facilitated through any other device with a keyboard and a mouse, i.e. a laptop.

The following sections report on the work that has been produced up to date (Related Work) and provide a detailed description of the experiment, the data collection as well as the analysis methods (Study Design). Both quantitative and qualitative results are presented and analysed (Results and Discussion), and finally the main outcomes are outlined (conclusion).

2. Related work

A wide number of studies denote that an active learning approach is deemed as beneficial to teaching in K-12 education (Felder et al., 2000). Active learning differs to the active 'lecture-and-practice-through-assignments' method in that it takes place predominantly in the classroom. It also places an emphasis on transforming students from 'passive recipients' to active learners, a condition that is usually induced by traditional lecture-led classes (Jenkins, 1998). This method of learning forms a constant 'dialogic process' between instructor and students, whereby the latter are encouraged to discover and experience principles by themselves through the constructive process of learning-by-doing (Bruner, 1966). Additionally, it is known to encourage a deep rather than a surface approach to learning, one that is adequate for all educational disciplines, but crucial in practical modules such as programming (Jenkins, 1998).

Programming is difficult due to the fact that it deals with understanding a lot of abstract concepts. Another major issue is also the fact that novice learners 'are very local in their comprehension of programs' (Wiedenbeck & Ramalingam, 1999). In other words they often know the syntax and semantics of the code line by line, however, they do not know how to combine these together in order to produce more compound code structures (Winslow, 1996). Combined with limited personal assistance in larger groups, as a result it is hard for students to cope and they frequently fail the class (Lahtinen, Ala-Mutka & Järvinen, 2005). This problem is further exacerbated in design education, where students are reluctant to accept that programming skills are necessary for their academic and professional careers.

There are many studies that attempted to investigate the reasons behind this and provide solutions to these issues. For example, Jenkins (Jenkins, 2001) argues that the traditional lecture-led method is the wrong environment setting for programming. Other studies (Krieckhaus & Eriksen, 1960) highlight that learners are bound to engage more and remember what they have learned, when they care about and understand the subject. Kukulska (Kukulska-Hulme & Shield, 2008) proposes that a technology-supported learning approach through mobile devices in the classroom can promote a more situated, contextualised and experiential way of learning. Evidently, sophisticated multi-touch mobile devices that work with intuitive gesture-based interactions can 'garner a lot of excitement'(Johnson, Adams & Cummins, 2012); combined with a challenge-based approach, they provide the kind of genuine student interest and necessary mental involvement that educators have long strived for in programming modules. Up to date there have been a few studies that examine the use of mobile devices and particularly the iPad (Cavanaugh, Hargis, Munns & Kamali, 2013; Diemer et al., 2012; Kinash, Brand, Mathew & Kordyban, 2011a; Manuguerra & Petocz, 2011; Souleles et al., 2014) both horizontally and vertically in academia.

Universities, schools and even kindergartens are following iPad-related initiatives for a number of reasons, a few of which are portability, multi-modality, cost savings and sustainability (replacement of textbooks) (Diemer et al., 2012). Additionally, one of the major advantages of the iPad as an instructional tool may lie in the ability to promote collaboration (Parker, Bianchi & Cheah, 2008), which is known to promote active learning involvement (Diemer et al., 2012). Results from other studies, however, indicate that the device has little to do with the positive impact, if any, on the learning process; they argue that mobility does not necessarily 'equate learning' (Kinash et al., 2011a). Additional evaluations attribute this to the fact that iPads are largely focused on media consumption rather than media creation and re-use (Norman, Nielsen & Group, 2010). However, there still is a noticeable shortage of research looking at the use of the iPad within the context of dedicated programs in HE (Souleles et al., 2013) such as programming or web development.

This study hypothesises that introducing the iPad as a platform for active, challenge-based learning and student collaboration in a web development module can aid in better comprehension of code and therefore lead to a deeper learning outcomes. It aims to investigate the students' perceptions in using the tablet to 'learn-by-doing'

ISBN: 978-9963-697-10-6

and it observes in-class behavioural phenomena deriving from the active collaborative learning approach.

3. Study design

This section describes the design of a four-week long study that incorporates the iPad in order to introduce an active collaborative learning approach in a web development course. Specifically, 'Web design and development 2' is a 13-week, three-hour long elective class and requires successful completion of 'Web design and development 1'. (For the purposes of this paper, these will be referred to as Web 1 and 2). Both courses' objectives mainly concentrate on the major components of front-end web technologies, such as HTML, XML, CSS and Javascript. Web 2 additionally introduces server-side technologies, such as PHP and MySQL. Although combining design, technical and user-centered knowledge is the primary evaluation criterion, Web 2 focuses more on code development. The course consists of three distinct units: a) Javascript and XML, b) PHP and MySQL, c) HTML5, CSS3 and content management systems (CMS).

The study spanned the first unit of the course. The participants of the study, ten fourth-year multimedia students, six females and four males, had also attended Web 1 in the previous semester. It is important to provide a brief insight into the teaching style used in Web 1 in order to understand the students' experience in this type of modules.

4. Web 1

Web 1 is also a 13-week, three-hour long lesson divided into two sessions: a) Lecture presentation session that aims to introduce theory, concepts and coding structures as well as to provide application paradigms, and b) Lab-based practical session during which students have to complete practical exercises – with help from the tutor, if needed. This process often requires longer than the remaining lesson time. Students therefore, have to finish their assignments at home, based on assistance from the lecture notes and other online resources.

5. Web 2

Web 2 follows a student-centred learning approach, in that the lecture is clustered into several theoretical units coupled with problem-solving exercises. In contrast to Web 1, coding applications are to be gradually constructed by students, as opposed to readily appearing on the whiteboard. These applications also require additional information that can either be retrieved by collective group suggestions or from online documentation resources. In this study, the lesson plan in Web 2 was as follows:

a) First, an informal communication setting was encouraged and students were assured that their performance on the iPads would not affect their marks.

b) The instructor ensured that both the problem and required solution were fully understood by the students.

c) Through constructive dialogue students identified the steps needed and how these were translated from natural to programming lingo.

d) Students performed the steps.

As explained in previous sections (see 'Introduction'), this model was based on both single and team exercises, described below.

6. Single-input exercises

The single-input exercises were carried out by a volunteer who was handed the iPad that had the exercise (in set-up mode) already loaded. This bypassed logistics such as the creation of a new file, initialisation and storage, and focused on the important tasks at hand. The user's input was projected on the whiteboard in real time by means of a remote desktop protocol (RDP)[5] client, Doceri™, which was connected to the instructor's workstation. The rest of the group was encouraged to provide verbal assistance in order to help their fellow student solve the coding exercise. If the student was unable to continue or wished to stop, the iPad was passed on to the next volunteer.

7. Groups-of-twos exercises

Unlike the single-input, team exercises were compulsory for students in the class, who were asked to form teams of two and were given an iPad per team to complete an exercise. These were also pre-loaded on the iPads, but this time in separate web-based IDE[6] accounts (per team); the teams worked independently from one another, and there was no real time projection. There was also a time constraint, after which the solution of the first team to finish was displayed on the whiteboard through the instructor's computer. Since the instructor was the administrator of the accounts, there was fast access to the coding documents.

8. Technical details

8.1 Doceri™

In the single-input exercises, the coding process was projected on the board in real time through RDP to the instructor's personal computer. Doceri™, an interactive whiteboard, screencast recorder and RDP tool was used for this. This step-by-step simulation (similar to Google Documents) to the level of basic input activity such as 'select', 'copy/paste', 'delete' and cursor insertion point- was important in that it illustrated precisely the cognitive processes that took place at the time. That was also the reason why an RDP client rather than an IDE was used. An IDE would undoubtedly provide a faster and better interface solution minus the poor visual screen reproduction. However, at the time of the study there were no adequate IDE tools on the iPad that facilitated the anticipated level of real-time projection needed. Installation of the Doceri™ application on both the instructor's computer and the iPad was required - the Doceri™ desktop and the Doceri™ app respectively.

[5] Remote Desktop Protocol (RDP) is a proprietary protocol developed by Microsoft, which provides a user with a graphical interface to connect to another computer over a network connection.

[6] An integrated development environment (IDE) or interactive development environment is a software application that provides comprehensive facilities to computer programmers for software development.

Doceri™ has a built-in interface keyboard that enables users, who are remotely logged in to a desktop computer, to type as if they were actually working on the actual computer. Additionally the screen area could be moved up-down and left-right by using two fingers to pan, and scaled by pinching and spreading two fingers again.

8.2 JS Bin

While there are a number of IDEs that support Javascript, HTML and CSS (i.e. Stypi, Floobits, Cloud9) most of these failed to fulfil the requirements of this study; these were a WIDE (web integrated development environment) tool providing concurrent views of code and output windows and supporting JavaScript, HTML and CSS development. It needed to also provide 'Save' and version control features. For this reason, JS bin was used in this study.

9. Post-study analysis

Individual information and participant bias prior, as well as empirical evidence following the study, was collected from students, who participated both as users and viewers. The information was elicited through two surveys (pre and post-study) and two focus groups (first and last class), and the resulting data was analysed using a mixed methods approach:

a. Quantitative analysis and reporting on the close-ended questions:

All close-ended answers were imported through Excel sheets as separate datasets into nVivo 10 (qualitative data analysis software), and were ordered as 'classification nodes'. As a result all pre-defined options were automatically classified as 'attributes' and the individual answers (choices) as respective 'values'.

b. Qualitative analysis:

Results from open-ended questions in surveys as well as focus group transcriptions were imported in nVivo and clustered into codes by means of thematic coding. Following an iterative coding approach and the resulting code saturation, a total of 46 thematic codes were identified. These were also classified by attitude type as 'positive', 'negative' and 'neutral' in order to examine generic trends. Following coding iterations, a fourth attribute was identified and added later on: 'prerequisite' since participants had made a lot of conditional expressions such as 'if the iPad had this…' or 'if that was like this…' and so on. Aside from survey and focus group data, observational information from the four-week classes was recorded and analysed by the researchers.

10. Results and discussion

10.1 Quantitative results (close-ended questions), pre-study

Question	totally disagree	disagree	undecided	agree	totally agree
I believe that I can learn quickly how to use the iPad	-	-	10%	60%	30%
I believe that is easy and pleasant to use the iPad	-	-	20%	50%	30%
I believe that the use of iPads during the teaching part of a class can enhance the learning	-	-	60%	40%	-
I believe that the use of iPads during the teaching part of a coding/programming class can enhance the learning process		10%	50%	30%	10%
Question	female	male			
Gender	60%	40%			
Question	none	minimal	average	above average	Excellent
Previous iPad experience	-	10%	30%	30%	30%

Table 1: pre-study survey close-ended questions

10.2 Post-study

Question	totally disagree	disagree	undecided	agree	totally agree
The area and size of the iPad screen was sufficient for writing, editing and displaying the code.	8%	38%	31%	23%	-
It was easy to write code using a touchscreen device	-	46%	46%	8%	-
It was easy to write code using the iPad's native keyboard	15%	39%	15%	23%	8%
The iPad was easy to use in terms of shape and weight	-	8%	-	54%	38%

I was able to understand and solve a coding exercise through the single projected approach, with possible help from others in class, better than solving it on my own	8%	15%	-	54%	23%
I prefer the active learning approach through the use of the iPad, than the traditional method used before in this type of lesson	-	8%	-	54%	38%
I prefer the single-projected exercise (with active contribution from the rest of the class) compared to the exercise in groups of two people	8%	8%	53%	23%	8%
The active learning approach could work equally well through the use of any other device	-	-	46%	23%	31%
Question	horizontal	vertical			
I used the iPad in the following orientation:	100%	-			
Question	very long	long	moderate	brief	very brief
The period of time required for learning to use the iPad interface for writing and editing code was:	15%	8%	31%	38%	8%

Table 2: post-study survey close-ended questions

10.3 Qualitative results

The following sections focus on the overall user experience and perceived learning outcomes elicited from the participants in the study. We report on the themes that have surfaced from the results of the follow up surveys and focus groups analysis next.

10.4 Active learning

The quantitative and qualitative results indicate a strong consensus towards a 'learning by doing' method. Students have based this on the fact that they were enrolled in a 'practice-based' course, and have so far become accustomed to 'digesting' knowledge better through application. More specifically, they explained that while coding statements appeared simple and comprehensible in traditional lectures, it was hard for them,to reflect on the tutor's instructions and 'how-tos'

when working on assignments from home later on. Additionally, it is worth noting that there usually are multiple concepts (e.g. variables, arrays and loops) included in a single lecture. Students commented that previously, while each unit 'made sense on its own', it was more difficult to put it into practice later on, after learning about the rest of the units. In agreement with Winslow (1996), they were also at a loss when combining these units to create more compound coding structures.

In order to address these issues each iPad exercise required students to use knowledge from previous exercises as well as performing additional research on the topic (Lim, 1998) to find a solution. This provided the kind of specific focus needed as well as an incremental knowledge build-up (scaffolding). The benefits from simultaneously 'applying-while-learning' were evident in remarks such as 'we were better learning through practising with the code at the same time…', 'it helped more with understanding…while practising at the time of teaching and also making mistakes'.

Regardless of the obvious advantages of the active learning approach, students confirmed that some degree of initial instruction was needed prior to the application stage: 'it will be impossible to start solving an exercise from scratch without being taught about it first or… how it must be syntaxed', ' …we need a little theory and a lot of practice', 'we just need the basic important things like functions and what each one does before we proceed…'. Existing literature also agrees that proper 'programming ability must rest' amongst others on ideally 'theory and formal methods' (Robins, Rountree & Rountree, 2003).

10.5 Problem solving and collaboration

Students deemed mistakes as important for their coding performance. Without exception, all responses with a 'positive attitude' attribute, classified under the 'active learning' and 'collaboration' thematic categories, pointed to the trial-and-error process.

In particular, students reported that during the projected method, observation of mistakes and attempts to solve a coding problem helped both participant and viewers arrive to a more 'effective understanding of programming methods' than by 'being served with the solution right away'. Research also denotes that 'people progress to the next level by solving problems' (Winslow, 1996). Perkins, Hancock, Hobbs, Martin, and Simmons (1986) highlight that the students' attitude to problems is essential when it comes to programming, and proceed to classify novice learners into 'stoppers' and 'movers'. As the words indicate, 'stoppers' are those who are bound to give up easily when they face a problem they cannot solve, and 'movers' are those who keep 'modifying their code' until they reach a solution. As expected, the Web 2 group, like any other class, consists of both types. However, the instructor observed persistent successive verbal contributions from the entire group towards resolving a task during the single-input exercises. Similar to previous work (Ioannou & Artino Jr, 2010), this study also shows that students were better able to understand through the group discussions on problem solving. As a result, the vast majority was eager to have this method incorporated into the module, with or

without the use of iPads, either in an open setting or in smaller groups. They commented that it was extremely helpful to 'solve the exercises collectively' and that 'the process helped them answer their own questions by actively participating in the resolution process'.

Undoubtedly this collective approach may have thrown in a safety net for 'stoppers', but all the same - based on the outcome - it is possible that the 'crowd' method has exposed that arriving to a solution through iterations between 'wrong and right' is the correct way to address a programming problem; it also illustrated that such tasks require patience, persistence and repetition. Schön (1987) focuses on the advantages of this method, and states that only in this way can programming novices become 'reflective practitioners basing learning on reflection of experience'. Likewise, students mentioned that they found it easier to remember the knowledge acquired from the lesson afterwards: 'it helps more in comprehension. Instead of seeing only theory during the class and being a 'zombie' during class… and then go home and remember nothing'.

A student has emphasised that active learning, especially through the projected method, provided a degree of 'equal comprehension amongst everyone'. Although the terms 'equality', 'similarity' and even 'unity' were not explicitly phrased by others, there was significant feedback in regards to 'setting off from common grounds of knowledge' or 'seeing what I had in mind, others did too as it appeared on the projector' and 'it was nice for all of us to be on the same level'.

10.6 Single-input projected exercises

One of the major problems instructors face is students' 'unwillingness to expose their own ignorance' (Jenkins, 2001) in front of their peers or tutor. This can have a considerable effect in the active learning progress, especially when this requires student involvement in an open setting. Likewise, although participants in the study favored the 'projected' approach, very few of them volunteered to work with the iPad themselves.

Although results from the close-ended questions place the 'projected' approach higher than the 'groups-of-twos', the disadvantages of this approach surfaced during the focus groups. Issues drew upon uncovering personal limitations such as learning disabilities i.e. dyslexia, the use of English as not first language, gaps in knowledge or delays during public task solving. Surprisingly, such attitudes are to be expected in larger groups where students are reportedly 'feeling rather lonely' (Race, 2001), and are unwilling to partake in the learning process. One would assume that this would not apply to smaller, more familiar groups; students in Web 2 are going through the fourth year together; they are well acquainted and also participate in common social activities. This was also the second consecutive module taught by the same tutor, thus a good relationship had already been established between the two parties, whereby students were referred to on a first-name basis.

Observation showed that while viewers would enthusiastically step in and help the 'operator' during a task, they were unwilling to take the part themselves. A student

commented that ‘it is as awkward for the person with the iPad as it is for a student watching a fellow student struggling to cope…’ Additionally he argued ‘help is good and will help solve the exercise in the end, but the person who’s handling the iPad might feel useless in the meantime’. Likewise, other students added that although they might know how to go about solving an exercise, they ‘can get confused and fail’ by being aware of the ‘public or time pressure’. The possible reward of publically accomplishing a task successfully was under-rated by students, compared with being ‘criticizsed’ by their peers. Gibbs (1992) suggest that the reasons for this are based on fear of being considered ‘stupid, attention seekers or creeps’. Results of the study prove that this fear is dominant, in that it transcends the feelings of familiarity and intimacy and finds its way even in smaller, friendlier groups.

10.7 Groups of twos

As explained this method followed a different approach; there was a time constraint, by which the solution of the first team to finish was displayed on the whiteboard through the instructor’s computer. The rest of the groups would then have to stop working and the class proceeded to the next unit. The downside to this was that the remaining groups were readily offered the completed exercise.

Conversely, one can argue that this may pose an incentive for groups to focus on the instructions in order to complete the tasks faster. Previous work with computer science students supports this by stating that a friendly competitive programming method was found to ‘increase student motivation’ (Lawrence, 2004). By comparing code against that of their peers, students were encouraged to put ‘more effort in its development’. Competition is known to promote interaction and involve students with different learning styles (Felder et al., 2000). However, female students repeatedly stressed that collaboration was crucial for their understanding of programming. ‘Peer-tutoring’ (Topping, 1996) was also found to be ‘rewarding’, as female students were keen to ‘fill in the gaps’ about something that they themselves felt confident about. This result came as no surprise since women are known to value teamwork over competition (Lawrence, 2004). On the contrary, many of the male students felt that heterogeneity in personal coding styles can be an obstacle in teamwork.

10.8 Virtual keyboard

The perceptual outcome of the collaborative active learning method was compromised, based on the users’ experience with the iPad keyboard. Previous studies concerned with usability in both education and industry domains indicate that users dislike typing on touch devices; they find it ‘uncomfortable and error-prone’ (Kinash, Brand, Mathew & Kordyban, 2011b; Norman et al., 2010). This study offers similar evidence. Two of the major issues recorded were a) the inability of the keyboard to cater for specific coding needs and b) the obscured screen view issue after keyboard activation; the native keyboard spans the entirety of the screen width and a good deal of the screen height. The same occurs in the Doceri™ interface. The keyboard-screen asymmetry - adopted from Nielsen’s (Budiu & Nielsen, 2011) read-tap asymmetry - seemed to frustrate the students while trying to type code: ‘basically the biggest problem is the keyboard because it covers up half of the screen

and we cannot see or do anything with the code…'. Programmers typically need to have simultaneous overview of the entire code as well as large-enough font sizes to work with specific code chunks. A clear view is of utmost importance when something as trivial as an extra dot or a comma may cause havoc in the output window. On the iPad, magnification offers detail at the expense of overview. Subsequently, one finds oneself constantly 'pinching and spreading' (zooming in and out), swiping fingers left and right (panning) and expanding - collapsing the virtual keyboard.

Likewise, students found that they had to go through 'too many movements' in order to achieve something that was literally effortless on a device with a 'tangible' keyboard; the problematic keyboard situation was, in fact, detrimental to students' overall perception of coding with the iPad. The instructor offered to lend the iPads out to students for practising their code exercises at home, in order to get used to them. The offer was immediately rejected by the claim that 'there was no time to waste'. Some students suggested that pairing the iPads with a wireless keyboard might solve the issue, an idea that generated reactions: that would defeat the point of using a tablet after all.

An additional context-specific issue was the absence of certain characters and symbols required from the keyboard. The native keyboard does not accept a customised selection of characters. At the time of the experiment, the only keyboard apps (such as a five-row keyboard tool) that allowed this required jailbreaking[7] the device; this was not an option. In order to access symbols such as **< > {} = + ( ) ‘‘** for javascript, students needed to tap once on the 'numbers' key to access a few, then once again on the 'symbols' key to display the rest. Unarguably, remembering when and how many times to tap combined with the complex mechanisms of reflecting the code syntax can significantly increase mental overhead, something that novice programming learners are better off without.

Other problems reported, especially by male participants, were the small size of keys compared to fingertip size (read-to-tap asymmetry) (Budiu & Nielsen, 2011), which resulted in repeated mistakes and caused considerable time delays.

10.9 Point to (click)

A much needed and second-most criticised feature was the 'point-to-click' functionality. This is equivalent to a mouse click on a normal computer; on a touch device it is subject to the fingertip diameter. As mentioned, programmers can spend hours on debugging, only to find that the error was a capital-cased character, a zero instead of an o, or even an extra space. Copy-paste is also an important function when programming because it guarantees exactness and re-usability; that is, the correct match of multiple instances of elements (such as functions, variables and objects) throughout the code. Selecting is an integral part of the copy/paste action. Smooth mobility and pointing precision are therefore vital. One of the biggest drawbacks experienced while using the iPad to code, was pointing to a specific

[7] iOS jailbreaking is the process of removing the limitations on Apple Inc. devices running the iOS operating system through the use of software and hardware exploits.

location in text so as to make such selects and edits. Students complained repeatedly and mentioned that they resorted to deleting and re-writing the whole line instead as it was 'easier to control and less time-consuming' than editing existing code; it was 'too hard to point to the exact spot', 'what was infuriating was that I wanted to take my mouse to a specific letter of a word but couldn't!' Evidently, students were aware of the built-in lens feature, from their previous experience with the iPad. Using it within this context though helped them to quickly reach a common verdict: 'confusing and misleading' in that it appeared above the actual position of the finger, as if corresponding to 'some other indivisible pointing device'. The ambiguity in actual versus virtual position is intensified in light of two important context-specific factors:

a. In text-heavy environments such as programming documents, where the default line-height is quite dense, this can trick users into thinking they are editing the line above: 'The touch or gesture is a little bit off'. Users intend to touch somewhere but end up touching something else instead since proximity is so dense between items due to the small screen size.

b. When experienced by expert users in graphic/animation/image editing software; combined with a mouse, these offer extremely high precision levels. Hence, the distinction between the two (touchscreen and mouse) is intensified in this case.

10.10 Code editing interface

Modern development editors and IDE interfaces allow for side-by-side coding and preview windows. These behave like two separate entities in that the preview pane is unaffected from changes that occur in the source code pane such as select, scrolling and sometimes magnification. Through the use of Doceri™, students could view both panes in the software; however, this feature was compromised, primarily during magnification, which affected the whole interface rather than just the coding panel.

On another note, scrollbars were too small and thin to handle, unless the screen was magnified, in which case the 'edges' of the code lines were cut-off from the screen while the outline view was lost in the meantime. Students, who were used to this standard functionality, found it hard to code through Doceri™, whereas they commented that the IDE environment 'was slightly better': it offered isolated manipulation of the code on the left and the results in a stable pane on the right. It nevertheless lacked two important features: code hinting and auto-completion; these are important for programmers; they can increase productivity by decreasing development time in avoiding common 'spelling and logic errors', eliminating unnecessary keystrokes and averting from browsing in documentation sources for code syntax (Omar, Yoon, LaToza & Myers, 2012). A few students enquired whether Adobe Dreamweaver could be installed on the iPads to facilitate a familiar, code-oriented environment. Even if there were an iOS-compatible app for Adobe Dreamweaver, this would not match the criterion of real-time edit simulation required for the purposes of this experiment.

11. General discussion

Evidence from this study agrees with the key concept in that 'all genuine education comes about through experience' (Dewey, 2007). In the case of practice-based courses such as programming or web/code development, it is necessary for educators to comply with the nature of the course and substitute ineffective one-way lecturing with interactive student-oriented self-derived learning through active participation and collaboration. However, one outcome is explicit: theory should not be entirely precluded in favour of practice; both are essential. Teachers and learners need iterative interaction on both levels (Laurillard, 2009). The responsibility lies with instructors; they need to carefully make the right decisions about what and how much content goes where as well as the order and treatment of that content in relation to its practical counterpart. Similarly, this study finds that the participants' requirements on this level were clear: a better refined delivery of basic concepts and ideas: 'we just need the basics like functions and syntax in order to continue on our own'. Jenkins (2001) contrasts this to the common obsession of faculties to push as much content as possible into a single session and states that this is no short of 'tyrannical'. The emphasis should be shifted instead towards encouraging deep learning and reflection (Robins et al., 2003), through shorter and more interesting theory-and-practice sessions. In doing so educators need to overcome the instilled tendency for controlled and organised lecture-led approaches that possibly derive from their own HE educational backgrounds (Jenkins, 2001). On a more practical level, this study also shows that the iterative 'theory-then-practice' approach can sometimes promote a form of disorder, generated by individual technical issues, large number of questions, delays and chatting; essentially, a state where learner attention and focus diminishes rapidly. This usually requires resources beyond a single instructor to help provide support and ensure that the lesson maintains a controlled rhythm and flow.

In this study the active learning process induced collaboration and through collaboration, especially in problem-solving, students admitted to better comprehension of coding problems and related theory, as opposed to performing exercises on their own. Interestingly, the help and support between students during exercises promoted an overall heightened sense of 'equality' that favoured sharing a 'synchronised' level of assimilated knowledge with peers. Prince (2004) states that such evidence may challenge the traditional assumptions that individual work and competition best promote achievement. Based on current results, we find that it is beneficial to maintain a balance; we quote Prince again, in that 'the entire course need not be team-based… nor must individual responsibility be absent'. This study suggests that a total lack of individual responsibility is not likely to occur in such settings; being a small group, students felt accountable towards their teams, the class and themselves. Receiving help and support to successfully complete a task in front of the entire group led to the realisation that this was not a personal achievement; this had both positive (i.e. effort for further improvement) and negative effects (i.e. embarrassment, low self-esteem). However, a comparison between the outcomes of collaborative learning against competitive or individualistic effort is quite ambitious and not the focus of this paper.

This work denotes that through active, real-time collective 'problem-solving' students perceived to have accomplished a higher level of comprehension and recall of coding concepts and structures. However, there is no evidence indicating that the use of the iPad was central in this process. Similar to other work, positive outcomes evidently are based upon two key factors: technology-supported student learning and the educational decisions made by the instructor (Kinash et al., 2011). The author states that 'the authentic independent variable is the collection of pedagogical decisions that the educator puts into play'.

Technology can indisputably facilitate a constructive learning-by-doing framework, something received with interest and keenness in this study; in fact, learners were so keen that they went the extra mile to contribute suggestions to ensure that the method would be effectively adopted into the lesson; none of these included the use of iPads; in reality, they all excluded the device in exchange for normal laptops or any other 'equipment' that offered a 'keyboard and a mouse'. The requirements were clear, they were rooted in a series of usability issues encountered in attempting to produce code on the device: lack of control and feedback in the gestural interface, accidental error-prone activation, keyboard-to-screen and read-to-tap asymmetry (Budiu & Nielsen, 2011), problematic typing on touchscreen and inability to point and select with precision. In some cases these were not perceived as 'irrecoverable; response was 'forgiving'; a number of students felt that these problems may have occurred due to lack of experience with the device. Regardless of the perceived causes, results show that the potential of the technology in learning code development is hindered by, unsurprisingly, bad usability. Evidence from this study agrees with Traxler's (2005) outcomes, that although the ability to 'synthesise data in real-time' is exciting since it restructures the learning dynamic, yet, in most cases equipment problems 'constrain the use of mobile technology in education'.

12. Conclusion

Similarly with Kinash's (2011) work, this study shows that the device had little to do with the positive attitude towards the active-mobile-learning approach. Unarguably, the iPad was favoured in that it was small, lightweight and portable and could conveniently rotate amongst the group of students. The process was found to be pleasant and entertaining, partially based on the iPad's intuitive and 'cool' interface, the gestures employed, and partially due to the 'inexperienced' manner in which participants strived to complete coding tasks in front of an audience. Students were keen on using the iPads, but only as a means to diverge from a traditionally led lecture-based class. Projection of the simulated coding activity on the board was favoured. However there was an explicit preference to do so through a different device, one that offered a normal keyboard and a mouse instead, i.e. a laptop. Collaboration between peers and equal levels of knowledge amongst the entire group were highly valued; the latter was preferred rather than smaller group exercises.

The disadvantages of the projected approach draw upon considerations in exposing weaknesses and gaps in knowledge, failure to complete a task, embarrassment, pressure of time and stress even in small and friendly teams. Although the mobile

ISBN: 978-9963-697-10-6

active learning method was perceived to increase knowledge assimilation and memorability through a collective problem-solving practice, there is no strong evidence that there was a significant positive learning impact relating to the use of iPads.

References

Bonwell, C., & Eison, J. (1991). Active learning: Creating excitement in the classroom (Vol. 80819). Retrieved December 30, 2013 from http://www.ydae.purdue.edu/lct/hbcu/documents/Active_Learning_Creating_Excitement_in_the_Classroom.pdf

Bruner, J. S. (1966). Towards a theory of instruction. Cambridge, MA: Harvard University Press.

Budiu, R., & Nielsen, J. (2011). iPad App and Website Usability. Retrieved December 30, 2013 from http://www.nngroup.com/reports/ipad-app-and-website-usability/

Cavanaugh, C., Hargis, J., Munns, S., & Kamali, T. (2013). iCelebrate Teaching and Learning: Sharing the iPad Experience. Journal of Teaching and Learning with Technology, 1(2), 1–12.

Dewey, J. (2007). Experience and education. New York: Simon and Schuster.

Diemer, T. T., Fernandez, E., & Streepey, J. W. (2012). Student perceptions of classroom engagement and learning using iPads. Journal of Teaching and Learning with Technology, 1(2), 13-25.

Felder, R. M., Stice, J. E., & Rugarcia, A. (2000). The future of engineering education II. Teaching methods that work. Chem. Engr. Education, 34(1), 26–39.

Gibbs, G. (1995). Improving the quality of student learning. Bristol, UK: Technical & Educational Services Ltd.

Ioannou, A., & Artino Jr, A. R. (2010). Learn more, stress less: Exploring the benefits of collaborative assessment. College Student Journal, 44(1), 189–199.

Jenkins, T. (2001). Teaching programming - A journey from teacher to motivator. In proceedings of the 2nd Annual Conference of the LSTN Center for Information and Computer Science. University of North London.

Johnson, L., Adams, S., & Cummins, M. (2012). NMC horizon report: 2012 Higher Education edition. Retrieved December 30, 2013 from http://www.editlib.org/p/48964/

Kinash, S., Brand, J., Mathew, T., & Kordyban, R. (2011). Uncoupling mobility and learning: When one does not guarantee the other. In R. Kwan (Ed.), Enhancing Learning Through Technology. Education Unplugged: Mobile Technologies and Web

2.0. Communications in Computer and Information Science, (pp. 342-350). Berlin: Springer.

Krieckhaus, E. E., & Eriksen, C. W. (1960). A study of awareness and its effects on learning and generalization. Journal of Personality, 28(4), 503–517.

Kuh, G. D., Cruce, T. M., Kinzie, J., & Gonyea, R. M. (2008). Unmasking the Effects of Student Engagement on First-Year College Grades and Persistence. The Journal of Higher Education, 79(5), 540–563.

Kukulska-Hulme, A., & Shield, L. (2008). An overview of mobile assisted language learning: From content delivery to supported collaboration and interaction. ReCALL, 20(3).

Lahtinen, E., Ala-Mutka, K., & Järvinen, H. M. (2005). A study of the difficulties of novice programmers. In proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education - ITiCSE ’05.

Laurillard, D. (2009). The pedagogical challenges to collaborative technologies. International Journal of Computer-Supported Collaborative Learning, 4(1), 5–20.

Lawrence, R. (2004). Teaching data structures using competitive games. Education, IEEE Transactions, 47(4), 459–466.

Lim, B. B. L. (1998). Teaching web deyelopment technologies in CS / IS curricula. ACM SIGCSE Bulletin, 30(1), 107-111.

Manuguerra, M., & Petocz, P. (2011). Promoting student engagement by integrating new technology into tertiary education: The role of the iPad. Asian Social Science, 7(11), 61–65.

Norman, D. A., Nielsen, J., & Group, N.N. (2010). Gestural interfaces: A step backward In usability. Retrieved December 30, 2013 from http://www.jnd.org/dn.mss/gestural_interfaces_a_step_backwards_in_usability_6.html

Omar, C., Yoon, Y., LaToza, T., & Myers, B. (2012). Active code completion. In Proceedings of ICSE '12. IEEE Press, Piscataway, NJ, USA, 859-869.

Parker, R., Bianchi, A., & Cheah, T. (2008). Perceptions of instructional technology: Factors of influence and anticipated consequences. Educational Technology & Society, 11, 274–293.

Perkins, D. N., Hancock, C., Hobbs, R., Martin, F., & Simmons, R. (1986). Conditions of learning in novice programmers. Journal of Educational Computing Research, 2(1), 37–55.

Prince, M. (2004). Does active learning work? A review of the research. Journal of engineering education, 93(3), 223–231.

Race, P. (2001). Using feedback to help students learn. Higher Education Academy: York.

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. Computer Science Education, 13(2), 137–172.

Schön, D. A. (1987). Educating the reflective practitioner. Jossey-Bass: San Francisco.

Souleles, N., Savva, S., Watters, H., Bull, B., & Annesley, A. (2014). A phenomenographic investigation on the use of iPads among undergraduate art and design students. British Journal of Educational Technology. Retrieved April, 8, 2014 from: http://onlinelibrary.wiley.com/journal/10.1111/%28ISSN%291467-8535/earlyview

Swan, K., van 't Hooft, M., Kratcoski, A., & Unger, D. (2005). Uses and effects of mobile computing devices in K-8 classrooms: a preliminary study. Journal of Research on Technology and Education, 38(1), 99-112.

Topping, K. (1996). The effectiveness of peer tutoring in further and higher education: A typology and review of the literature. Higher Education, 32(3), 321–345.

Traxler, J. (2005). Defining mobile learning. In proceedings of the IADIS International Conference Mobile Learning 2015, 261–266.

Wiedenbeck, S., & Ramalingam, V. (1999). Novice comprehension of small programs written in the procedural and object-oriented styles. International Journal of Human-Computer Studies, 51(1), 71–87.

Winslow, L. E. (1996). Programming pedagogy - a psychological overview. ACM SIGCSE Bulletin, 28(3), 17–22.

Acknowledgments
Special thanks to Zacharias Mavris, Antigoni Parmaxi, Dr Andri Ioannou, Web Design and Development 2 class 2013.