Συνέδρια της Ελληνικής Επιστημονικής Ένωσης Τεχνολογιών Πληροφορίας & Επικοινωνιών στην Εκπαίδευση

Τόμ. 1 (2014)

7ο Πανελλήνιο Συνέδριο Διδακτικής της Πληροφορικής

Περιβάλλοντα προγραμματισμού για αρχάριους. Scratch & App Inventor: μια πρώτη σύγκριση.

Στ. Παπαδάκης, Β. Ορφανάκης, Μ. Καλογιαννάκης, Ν. Ζαράνης

Βιβλιογραφική αναφορά:

Παπαδάκης Σ., Ορφανάκης Β., Καλογιαννάκης Μ., & Ζαράνης Ν. (2022). Περιβάλλοντα προγραμματισμού για αρχάριους. Scratch & App Inventor: μια πρώτη σύγκριση. *Συνέδρια της Ελληνικής Επιστημονικής Ένωσης Τεχνολογιών Πληροφορίας & Επικοινωνιών στην Εκπαίδευση*, *1*, 020–029. ανακτήθηκε από https://eproceedings.epublishing.ekt.gr/index.php/cetpe/article/view/4370

Περιβάλλοντα προγραμματισμού για αρχάριους. Scratch & App Inventor: μια πρώτη σύγκριση.

Στ. Παπαδάκης[1], Β. Ορφανάκης[2], Μ. Καλογιαννάκης[3], Ν. Ζαράνης[4]

[1] Υποψήφιος Διδάκτορας, καθηγητής Πληροφορικής, επιμορφωτής Β' επιπέδου
stpapadakis@gmail.com
[2] Υπεύθυνος ΚΕ.ΠΛΗ.ΝΕ.Τ. Λασιθίου, επιμορφωτής Β' επιπέδου
vorfan@gmail.com
[3] Λέκτορας, Πανεπιστήμιο Κρήτης, Παιδαγωγικό Τμήμα Προσχολικής Εκπαίδευσης
mkalogian@edc.uoc.gr
[4] Επίκουρος Καθηγητής, Πανεπιστήμιο Κρήτης, Παιδαγωγικό Τμήμα Προσχολικής Εκπαίδευσης
nzaranis@edc.uoc.gr

Περίληψη

Από την πρώτη είσοδο του προγραμματισμού στις σχολικές μονάδες, η διδασκαλία του παρουσιάζει μεγάλες δυσκολίες. Τα μαθησιακά αποτελέσματα είναι φτωχά, εγείροντας ερωτήματα σχετικά με την αποτελεσματικότητα της μεθόδου που χρησιμοποιείται. Επιπλέον, οι μαθητές συχνά θεωρούν τον προγραμματισμό ως μια επίπονη και βαρετή διαδικασία. Το πρόβλημα ενδεχόμενα θα μπορούσε να λυθεί κάνοντας τον εισαγωγικό προγραμματισμό εύκολο και διασκεδαστικό για τους μαθητές. Ένας τρόπος να επιτευχθεί αυτό είναι αμβλύνοντας τα εμπόδια για την εισαγωγή στον προγραμματισμό μέσω της χρήσης ειδικών περιβαλλόντων προγραμματισμού για αρχαρίους. Τέτοια περιβάλλοντα είναι το Scratch και το App Inventor, τα οποία, αμφότερα, έχουν δημιουργηθεί από το MIT. Στην παρούσα μελέτη, θα επικεντρωθούμε στις ομοιότητες και στις διαφορές των δύο περιβαλλόντων προγραμματισμού προσπαθώντας να αναζητήσουμε το καταλληλότερο για χρήση σε κάθε εκπαιδευτική βαθμίδα.

Λέξεις κλειδιά: Περιβάλλοντα Προγραμματισμού για Αρχάριους (NPE), Scratch, App Inventor.

Εισαγωγή

Οι μαθητές, για να επιτύχουν στη σημερινή κοινωνία της καινοτομίας, πρέπει να διαθέτουν μια σειρά από δεξιότητες, όπως αυτή της δημιουργικής σκέψης και της κριτικής ανάλυσης, οι οποίες είναι γνωστές ως μαθησιακές δεξιότητες του 21ου αιώνα. Δυστυχώς, οι περισσότερες χρήσεις των ΤΠΕ (Τεχνολογιών της Πληροφορίας και Επικοινωνίας) στα σχολεία σήμερα φαίνεται να μην υποστηρίζουν αυτές τις δεξιότητες μάθησης. Σε πολλές περιπτώσεις οι ΤΠΕ απλώς ενισχύουν παλιούς τρόπους διδασκαλίας και μάθησης (Resnick, 2008). Στον αντίποδα, δραστηριότητες όπως ο προγραμματισμός υπολογιστών, θεωρούνται ιδιαίτερα σημαντικές για το μαθητή του 21ου αιώνα. Μαθαίνοντας να προγραμματίζουν, οι μαθητές αποκτούν ποικίλα οφέλη, όπως ικανότητα πληρέστερης και δημιουργικής έκφρασης των απόψεών τους, ανάπτυξη λογικού τρόπου σκέψης και κατανόηση του τρόπου λειτουργίας των ΤΠΕ, οι οποίες βρίσκονται διάσπαρτες στη καθημερινή τους ζωή (Gans, 2010).

Με βάση τα παραπάνω, θα περίμενε κάποιος την αύξηση των μαθητών που μαθαίνουν προγραμματισμό. Ωστόσο, ο αριθμός αυτός μειώνεται συνεχώς τα τελευταία χρόνια. Για παράδειγμα, στη Μεγάλη Βρετανία οι μαθητές που μαθαίνουν προγραμματισμό έχουν μειωθεί στο ένα τρίτο κατά την τελευταία πενταετία, δημιουργώντας αντίστοιχη τάση και

Π. Αναστασιάδης, Ν. Ζαράνης, Β. Οικονομίδης & Μ. Καλογιαννάκης, (Επιμ.), *Πρακτικά 7ου Πανελλήνιου Συνεδρίου «Διδακτική της Πληροφορικής»*. Πανεπιστήμιο Κρήτης, Ρέθυμνο, 3-5 Οκτωβρίου 2014.

στα πανεπιστήμια (Wilson & Moffat, 2010). Οι Forte και Guzdial (2004) υποστηρίζουν ότι η «παραδοσιακή» διδακτική προσέγγιση στον προγραμματισμό είναι πιθανότερο να αποτρέψει, παρά να προσελκύσει τους μαθητές. Ιδιαίτερα, η απουσία κινήτρου αποτελεί ένα από τους κύριους λόγους για τους οποίους οι μαθητές εγκαταλείπουν πρόωρα μαθήματα προγραμματισμού (Siegle, 2009).

Προκειμένου να κάνουν τη διαδικασία του προγραμματισμού πιο φιλική και ευχάριστη στους αρχάριους μαθητές, έχουν αναπτυχθεί τα τελευταία χρόνια περιβάλλοντα προγραμματισμού για αρχάριους (NPEs, Novice Programming Environments) (Utting et al., 2010). Δημοφιλή NPE, όπως το Scratch και η Alice, έχουν χαμηλώσει με επιτυχία το εμπόδιο της εισόδου στον προγραμματισμό (Roy et al., 2012), προσφέροντας τη δυνατότητα στους μαθητές να εξασκήσουν τη δημιουργική τους φαντασία σε δραστηριότητες που είναι συναφείς με τα ενδιαφέροντα τους (Federici, 2011). Τα τελευταία χρόνια οι νέοι ζουν σ' ένα κόσμο που κατακλύζεται από έξυπνες κινητές συσκευές (ΕΚΣ) και είναι φανατικοί χρήστες, τόσο των συσκευών αυτών όσο και των εφαρμογών τους (Wagner et al., 2013; Zaranis et al., 2013), με αρκετούς να εκδηλώνουν το ενδιαφέρον να αναπτύξουν τις δικές τους φορητές εφαρμογές (Roy, 2012). Εφόσον τα NPE επιθυμούν να παραμείνουν στο επίκεντρο του μαθητικού και νεανικού ενδιαφέροντος, θα πρέπει να προσαρμοστούν και να στοχεύσουν σε ένα φορητό-κεντρικό κοινό. Το MIT σε συνεργασία με τη Google, προσφέρουν ένα από τα πρώτα NPE, με την ονομασία App Inventor (AI) (Roy et al., 2012).

Στην παρούσα μελέτη θα επιχειρήσουμε μια παρουσίαση και σύγκριση δύο NPE, του δημοφιλούς Scratch και του ανερχόμενου AI. Η σύγκριση μας, μέσω της προβολής των ισχυρών και αδύνατων στοιχείων τους, θα προσπαθήσει να αξιολογήσει την καταλληλότητά τους ως εργαλείων εκμάθησης προγραμματισμού. Η μελέτη μας μπορεί να αποτελέσει ένα χρήσιμο οδηγό για οποιονδήποτε προσανατολίζεται να τα εντάξει (και ιδιαίτερα το νέο-εμφανιζόμενο AI) ως εκπαιδευτικά εργαλεία για στην εκμάθηση του προγραμματισμού.

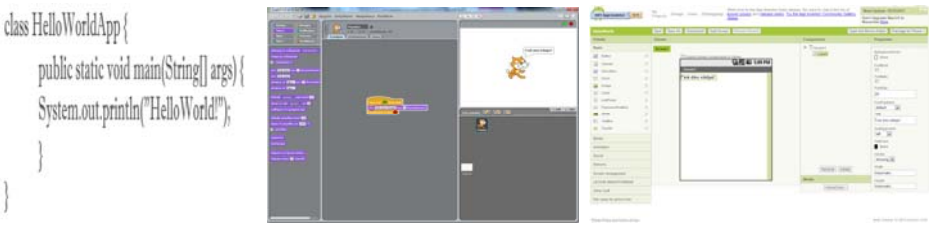
Προβληματική για τη διδασκαλία του προγραμματισμού

Από την εισαγωγή του προγραμματισμού στις σχολικές μονάδες, η διδασκαλία του έχει συναντήσει μεγάλες δυσκολίες, ενώ τα φτωχά μαθησιακά αποτελέσματα εγείρουν αμφιβολίες για τις μεθόδους που χρησιμοποιήθηκαν για αυτή την εισαγωγή (Forte & Guzdial, 2004). Οι μαθητές συχνά θεωρούν ότι η εκμάθηση του προγραμματισμού είναι δύσκολη και επίπονη διαδικασία (Krul, 2012). Οι γνωστικές θεωρίες, για παράδειγμα, προβάλουν την αδυναμία των μαθητών ως προς την επίλυση, καθώς και τη δυσκολία τους στην κατανόηση της σύνταξης και της σημειολογίας των προγραμματιστικών εντολών (Robins, Rountree & Rountree, 2003). Βασικές δομές ελέγχου, όπως οι συνθήκες (if, if-else) και οι βρόχοι (while, for), είναι δύσκολο να κατανοηθούν και να εφαρμοστούν από τους αρχάριους προγραμματιστές (Krul, 2012). Αρκετοί μαθητές θεωρούν τον προγραμματισμό ως μια μυστηριώδη και πολύπλοκη διαδικασία, η οποία απαιτεί εξειδικευμένη τεχνική κατάρτιση και εκπαίδευση (Ford, 2008). Οι μαθητές περιγράφουν τα μαθήματα προγραμματισμού ως υπερβολικά τεχνικά, αποκομμένα από τον πραγματικό κόσμο και στερούμενα δημιουργικότητας (Khuloud & Gestwicki, 2013). Οι Forte και Guzdial (2004) θεωρούν ότι η κύρια αιτία αποτυχίας ή πρόωρης εγκατάλειψης των μαθημάτων προγραμματισμού αποτελεί η αντίληψη των μαθητών ότι δεν είναι ενδιαφέροντα ή χρήσιμα. Σύμφωνα με τους Freudenthal et al. (2010), η διδασκαλία του προγραμματισμού θα πρέπει να γίνεται με τρόπο, ώστε να ελαχιστοποιείται το γνωστικό φορτίο, ενώ ταυτόχρονα να μεγιστοποιείται η παιδαγωγική αξία. Η εμπλοκή των μαθητών είναι συχνά επιτυχής, όταν το πλαίσιο διδασκαλίας τροφοδοτείται από θέματα που έχουν άμεσο ενδιαφέρον για τους μαθητές (Gray, Abelson, Wolber, & Friend, 2012).

Οι Margulieux et al. (2012) επισημαίνουν ότι το πρόβλημα μπορεί να αντιμετωπιστεί, μετατρέποντας τον εισαγωγικό προγραμματισμό σε μια εύκολη και διασκεδαστική εμπειρία, και υπάρχουν διάφοροι τρόποι προκειμένου να ευοδωθεί η προσπάθεια αυτή. Ένας τρόπος είναι μέσω της μείωσης του ενδογενούς γνωστικού φορτίου που απαιτείται από τους αρχάριους για την εκμάθηση του προγραμματισμού, με αντίστοιχη μείωση της ποσότητας των πληροφοριών που χρησιμοποιούνται για την επίλυση ενός προβλήματος (Robins et al., 2003). Για να μειωθεί ο όγκος των πληροφοριών, τα συστατικά του προγραμματισμού δύνανται να απομονωθούν, έτσι ώστε οι μαθητές να μην προσπαθούν να μάθουν ταυτόχρονα πολλαπλά θέματα. Οι μαθητές θα μπορούσαν πρώτα να διδαχθούν την επίλυση ενός προβλήματος σε θεωρητικό επίπεδο, δημιουργώντας τα νοητικά μοντέλα, τα οποία επικεντρώνονται στην κατασκευή των λύσεων, δίχως οι ίδιοι να ασχοληθούν ιδιαίτερα με τη σύνταξη δυσνόητων εντολών (Resnick et al., 2009).

Ο Papert υποστήριζε ότι οι γλώσσες προγραμματισμού θα πρέπει να έχουν ένα χαμηλό δάπεδο (ώστε να είναι εύκολο κάποιος να ξεκινήσει) και ένα υψηλό ταβάνι (προκειμένου να προσφέρει δυνατότητες για όλο και πιο πολύπλοκα έργα σε βάθος χρόνου) (Harvey & Monig, 2010). Επιπλέον, οι διάφορες γλώσσες θα πρέπει να υποστηρίζουν πολλαπλούς τύπους έργων (projects), προκειμένου να παρακινούν ανθρώπους διαφορετικών ενδιαφερόντων και μαθησιακών στυλ (Guzdial, 2004). Οι προγραμματιστικές γλώσσες τύπου «σύρε και άσε» (drag-and-drop) αντικαθιστούν τον προς σύνταξη κώδικα με συρόμενα συστατικά μέρη (components), προσέγγιση η οποία μειώνει το γνωστικό φορτίο, το σχετιζόμενο με τη σύνταξη των εντολών, επιτρέποντας στους χρήστες να επικεντρωθούν στην εννοιολογική επίλυση ενός προβλήματος. Επίσης, οι γλώσσες αυτού του τύπου θεωρείται ότι είναι εύκολες για τους χρήστες όλων των ηλικιών, γνωστικών υποβάθρων και ενδιαφερόντων, επιτρέποντάς τους να πειραματιστούν με τα διάφορα συστατικά τους μέρη, απλά ενώνοντας κομμάτια κώδικα μαζί, όπως ακριβώς ενώνουν τουβλάκια Lego (Resnick et al., 2009).

Τα περιβάλλοντα που συγκεντρώνουν τα παραπάνω χαρακτηριστικά ονομάζονται Περιβάλλοντα Προγραμματισμού για Αρχάριους (NPE - Novice Programming Environments) (Krul, 2012). Τα NPE, όπως το Scratch, η Alice και τα Lego Mindstorms NXT, έχουν γνωρίσει μεγάλη αποδοχή και δημοσιότητα τα τελευταία έτη, καθώς η έρευνα έχει δείξει ότι τα NPE διαδραματίζουν σημαντικό ρόλο στην προσέλκυση και διατήρηση νέων προγραμματιστών στα σχολικά και μη περιβάλλοντα (Federici, 2011). Ένα NPE χρησιμοποιεί οπτικά στοιχεία αντί για προγραμματιστικές εντολές αποκρύπτοντας τη συντακτική πολυπλοκότητα των γλωσσών προγραμματισμού, καθιστώντας ευκολότερη την κατανόηση και χρήση των βασικών αλγοριθμικών δομών από τους αρχάριους προγραμματιστές (Roy et al., 2012). Τα NPE διευκολύνουν την ανάπτυξη του λογισμικού σε ένα πλαίσιο που είναι διασκεδαστικό και μη «απειλητικό». Η ελπίδα είναι ότι οι μαθητές δεν θα αισθάνονται πλέον άγχος και χαμηλή αυτοεκτίμηση, όταν έρχονται αντιμέτωποι με τους υπολογιστές, ώστε να είναι πιο δεκτικοί για περαιτέρω εμβάθυνση στον προγραμματισμό (Olabe, Olabe, Basogain, & Castaño, 2011). Ωστόσο, δεν πρέπει να μας διαφεύγει ότι η χρήση ενός NPE δε λύνει το πρόβλημα της σύνταξης των εντολών, δεδομένου ότι ο μαθητής θα πρέπει να το αντιμετωπίσει αργότερα, κατά την εκμάθηση μιας δεύτερης, «παραδοσιακής» γλώσσας, αλλά το αναβάλλει έως ότου ο μαθητής έχει κατανοήσει τις βασικές προγραμματιστικές αρχές (Wilson & Moffat, 2010). Στο σχήμα 1 παρουσιάζεται η διαφορά στη σύνταξη του τυπικού προγράμματος (Hello World!) σε μια παραδοσιακή γλώσσα προγραμματισμού (Java) και σε 2 NPE (Scratch και AppInventor).



Σχήμα 1. Διαφορά σύνταξης προγράμματος σε Java, Scratch & AppInventor

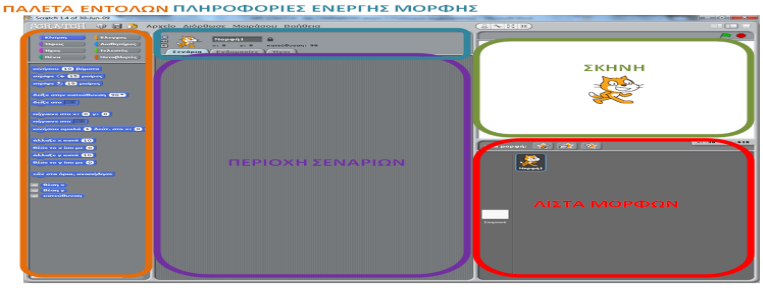
Scratch

Το Scratch αποτελεί ένα ΝΡΕ, το οποίο δημιουργήθηκε από την ερευνητική ομάδα Lifelong Kindergarten του MIT Media Lab (http://scratch.mit.edu) και θεωρείται ως το κορυφαίο περιβάλλον για την εισαγωγή των παιδιών στον προγραμματισμό (Wilson & Moffat, 2010). Όπως αναφέρουν οι Utting et al., (2010), οι Maloney και Resnick, μέλη της ομάδας ανάπτυξης του Scratch, είχαν δηλώσει ότι κατά την ανάπτυξη του Scratch, ήθελαν να χαμηλώσουν την προγραμματιστική οροφή, προκειμένου τα παιδιά να ξεκινήσουν νωρίτερα τον προγραμματισμό. Για τους ίδιους ερευνητές η εκμάθηση του προγραμματισμού είναι παρόμοια της γραφής. Είναι σκόπιμο για τα παιδιά να ξεκινήσουν με απλές μορφές έκφρασης και σταδιακά να μάθουν πιο εκλεπτυσμένους τρόπους, για να εκφράσουν τον εαυτό τους σε βάθος χρόνου. Το σλόγκαν του Scratch είναι «φαντάσου - προγραμμάτισε - μοιράσου!» (Resnick et al., 2009). Με το Scratch οι μαθητές μετατρέπονται από καταναλωτές των μέσων σε παραγωγούς, δημιουργώντας τις δικές τους διαδραστικές ιστορίες, παιχνίδια και διαμοιράζοντάς τα ελεύθερα, μέσω του διαδικτύου (Resnick, 2008).

Το Scratch ενθαρρύνει και διευκολύνει την ανάπτυξη των προγραμμάτων, χρησιμοποιώντας ένα μίγμα πολυμεσικών στοιχείων, όπως γραφικά, ήχους, βίντεο κ.α., ώστε να δημιουργηθεί ένα νέο έργο (Ford, 2008). Άλλωστε το όνομα Scratch υποδηλώνει την ιδέα της μείξης και της τροποποίησης, καθώς μιμείται την αντίστοιχη τεχνική των dj's, οι οποίοι παίζουν με τους δίσκους βινυλίου, προκειμένου να δημιουργήσουν νέους ήχους (Resnick et al., 2009). Το Scratch έχει ονομαστεί ως το YouTube των διαδραστικών μέσων, καθώς καθημερινά οι «Scratchers» από όλο τον κόσμο ανεβάζουν έως και 1.000 νέα έργα στο δικτυακό τόπο του έργου, με τον πηγαίο κώδικα ελεύθερα διαθέσιμο για κοινή χρήση και τροποποίηση. Μέσα σε 3 χρόνια ο αριθμός των έργων που κατέβηκαν από το δικτυακό τόπο του Scratch αυξήθηκε από 70.000 σε περισσότερα από 1.8 εκατομμύρια (Federici, 2011). Μέχρι τα μέσα του 2012 είχε μεταφραστεί σε περισσότερες από 50 γλώσσες, ενώ κάθε χρόνο, παγκοσμίως, διοργανώνεται μια μέρα Scratch (day.scratch.mit.edu).

Για τις ανάγκες του προγραμματισμού το Scratch παρέχει έτοιμους χαρακτήρες, σκηνικά και πολυμεσικά στοιχεία. Πρόσθετα στοιχεία μπορούν να δημιουργηθούν από το χρήστη με τη βοήθεια του ενσωματωμένου προγράμματος ζωγραφικής, ενώ υπάρχει δυνατότητα χρήσης και εξωτερικών πηγών (Ford, 2008). Στο Scratch κάθε αντικείμενο (sprite) μπορεί να έχει ένα ή περισσότερα σενάρια (scripts) συνδεδεμένα με αυτό. Τα σενάρια προσθέτουν ιδιότητες στα αντικείμενα, επιτρέποντάς τους να ενεργούν με όποιο τρόπο επιθυμεί ο χρήστης στα πλαίσια του έργου του. Τα σενάρια δημιουργούνται ενώνοντας πλακίδια, (blocks) τα οποία είναι οργανωμένα ανά κατηγορία, όπως π.χ. έλεγχος, κίνηση κ.ά. Απαιτείται ελάχιστη χρήση του πληκτρολογίου από τον χρήστη, καθώς τα πλακίδια μετακινούνται στη σκηνή με τη βοήθεια του ποντικιού και ενώνονται μεταξύ τους σαν τουβλάκια Lego. Τα πλακίδια συνδέονται μόνο όταν υπάρχει συντακτική λογική, εξαλείφοντας πλήρως τα λάθη σύνταξης, απλοποιώντας σημαντικά την ανάπτυξη

εφαρμογών κάνοντας, ωστόσο, χρήση του ίδιου ρεπερτορίου εντολών και δομών που απαντώνται σε άλλες γλώσσες προγραμματισμού (Olabe et al., 2011). Για παράδειγμα, το Scratch υποστηρίζει τη χρήση μεταβλητών, δομών επιλογής και επανάληψης καθώς και, υπό όρους, τον αντικειμενοστραφή και παράλληλο προγραμματισμό (Ford, 2008).



Σχήμα 2. Διεπαφή χρήστη Scratch

Το Scratch παρουσιάζει και κάποιες αδυναμίες, αφού όπως επισημαίνουν οι Harvey & Monig (2010), δε διαθέτει διαδικασίες, και κατά συνέπεια δεν μπορεί να εφαρμοστεί η έννοια της αναδρομής, μία από τις κεντρικές ιδέες της Logo. Επίσης η υποστήριξή του στις δομές δεδομένων είναι αδύναμη. Ωστόσο, οι αδυναμίες αυτές είναι «σκόπιμες», καθώς οι δημιουργοί του Scratch εσκεμμένα απέφυγαν να «φορτώσουν» τη γλώσσα με χαρακτηριστικά τα οποία θα λειτουργούσαν αποτρεπτικά για τους αρχάριους προγραμματιστές και ιδίως για τα μικρά παιδιά. Επίσης, το Scratch δεν υποστηρίζει κλάσεις και κληρονομικότητα. Το Πανεπιστήμιο του Berkeley εξάλλου, έχει αναπτύξει μια επέκταση (extension) με την ονομασία BYOB (Build Your Own Blocks), η οποία αντιμετωπίζει αρκετές από τις ελλείψεις του Scratch.

Σχεδόν 7 χρόνια από την πρώτη παρουσίασή του, το MIT προχώρησε, μετά από μια μακρά περίοδο δοκιμών, στην ανάπτυξη μιας νέας έκδοσης με την ονομασία Scratch 2.0. (http://scratch.mit.edu/overview/). Η νέα έκδοση φέρνει αρκετές βελτιώσεις, κυριότερη εκ των οποίων είναι η μετάβαση από την υποχρεωτική τοπική χρήση στη χρήση μέσω ενός προγράμματος περιήγησης (browser-based).

App Inventor

Το App Inventor (AI) είναι ένα δωρεάν δικτυακό περιβάλλον προγραμματισμού με πλακίδια για τη δημιουργία εφαρμογών για έξυπνες κινητές συσκευές (ΕΚΣ) με λειτουργικό σύστημα Android (Krul, 2012; Wolber, 2010). Το AI ανακοινώθηκε ως ένα πειραματικό έργο των εργαστηρίων της Google στα τέλη του 2009, ενώ συνέχισε την ανάπτυξή του μέχρι τα τέλη του 2011. Επικεφαλής της ομάδας ανάπτυξης υπήρξε ο καθηγητής του MIT Harold Abelson (Abelson, 2009). Στις αρχές του 2012 ο ίδιος καθηγητής μετέφερε το έργο στο κέντρο για την εκμάθηση της φορητής μάθησης του MIT (Mobile Learning Center) για δημόσια χρήση ως λογισμικό ανοικτού κώδικα. Μέσα σε 18 μήνες από την «υιοθέτησή» του από το MIT, το AI έχει προσελκύσει περισσότερους από 2 εκατομμύρια εγγεγραμμένους χρήστες, 40.000 από τους οποίους είναι ενεργοί σε εβδομαδιαία βάση.

Πριν την έλευση του AI, η δημιουργία μιας Android εφαρμογής ήταν μια αρκετά δύσκολη και εξειδικευμένη διαδικασία, λόγω των πολλών απαιτήσεων, όπως η καλή γνώση της Java καθώς και η ανάγκη εξοικείωσης του χρήστη με επαγγελματικά εργαλεία

ανάπτυξης λογισμικού (π.χ. Eclipse, Android SDK κ.ά.). Το AI, αντίθετα, υιοθετεί το επιτυχημένο παράδειγμα της χρήσης του οπτικού προγραμματισμού με πλακίδια (π.χ. Scratch), προσαρμόζοντάς το στον προγραμματισμό έξυπνων κινητών συσκευών (π.χ. smartphones) (Gray et al., 2012). Η εφημερίδα New York Times αποκάλεσε το AI ως «Do-it-yourself App Creation Software» (Wolber, 2010). Το περιβάλλον ανάπτυξης του AI υποστηρίζει και τα τρία δημοφιλή λειτουργικά συστήματα. Στο σχήμα 3 παρουσιάζεται η διαφορά στη δημιουργία μιας φορητής εφαρμογής αναπαραγωγής ήχου σε Java και στο AI.

Σχήμα 3. Η διαφορά στη δημιουργία μιας εφαρμογής σε Java & eclipse και σε AI

Συνδυάζοντας το αυξημένο κίνητρο των χρηστών για τη δημιουργία φορητών εφαρμογών με τα πλεονεκτήματα της χρήσης ενός ΝΡΕ, το AI θα μπορούσε να χρησιμοποιηθεί ως εισαγωγικό περιβάλλον στον προγραμματισμό και τη διδασκαλία των βασικών δομών ελέγχου (Roy, 2012; Hsu, Rice & Dawley, 2012). Οι Liu et al., (2013) αναφέρουν ότι, εδώ και 4 έτη, το AI χρησιμοποιείται στην πρωτοβάθμια και δευτεροβάθμια εκπαίδευση, ενώ αρκετά πανεπιστήμια στις ΗΠΑ αναπροσαρμόζουν το πρόγραμμα σπουδών τους εισάγοντας το AI για τη διδασκαλία του προγραμματισμού (http://mobile-csp.org/). Όπως επισημαίνει ο Loukides (2010), σκοπός του AI είναι να επιτρέψει σε ανθρώπους, οι οποίοι κανονικά δε θα προγραμμάτιζαν ποτέ, να δημιουργήσουν τις εφαρμογές που επιθυμούν, δίχως κατ' ανάγκη να είναι προγραμματιστές. Ο Hal Abelson (Abelson, 2010) αναφέρει ότι ο γενικότερος σκοπός ανάπτυξης του AI είναι να επιτρέψει στους χρήστες συσκευών με εγκατεστημένο λογισμικό Android να μετατραπούν από καταναλωτές σε δημιουργούς.

Όπως περιγράφεται στο δικτυακό τόπο του προγράμματος (MIT, 2013), το AI αποτελείται από δυο βασικά συστατικά μέρη, τα οποία επιτρέπουν στους χρήστες να χτίσουν τις εφαρμογές τους:

- Σχεδιαστής (Designer): πρόκειται για μια ιστοσελίδα στην οποία ο χρήστης επιλέγει τα συστατικά μέρη για την εφαρμογή του και προσαρμόζει τις ιδιότητες του κάθε συστατικού.
- Συντάκτης (Blocks Editor): πρόκειται για ένα παράθυρο υλοποιήσιμο σε java, στο οποίο ο χρήστης τοποθετεί τα κομμάτια κώδικα (πλακίδια), προκειμένου να «μεταφέρει» στα συστατικά μέρη του προγράμματος το πώς να «συμπεριφερθούν».

Το AI παρέχει σχεδόν σε πραγματικό χρόνο τη δυνατότητα προσαρμογής στις ενέργειες του χρήστη, με αποτέλεσμα ο χρήστης να μπορεί άμεσα να παρατηρήσει την τροποποίηση της εφαρμογής στη συσκευή του Android. Εναλλακτικά, υπάρχει η δυνατότητα χρήσης του ενσωματωμένου προσομοιωτή. Πρόκειται για μια πλήρη εικονική συσκευή, με μοναδικό αρνητικό στοιχείο ότι είναι σχετικά αργός σε σύγκριση με μια φορητή συσκευή. Το τελικό προϊόν μπορεί ή να συσκευαστεί σε μορφή .apk (Android application package), είτε να διανεμηθεί στο δικτυακό κατάστημα της Google. Η δομή του AI, καθώς και οι δυο τύποι εξόδων, παρουσιάζονται στο σχήμα 4.

Σχήμα 4. Δομή και τύποι εξόδων του AI (προσομοιωτής & smartphone)

Ήδη η σχετική ομάδα του MIT δουλεύει στη δημιουργία μιας νέας έκδοσης του AI, η οποία αναμένεται να κυκλοφορήσει στα τέλη του 2013 (ai2.appinventor.mit.edu). Η μεγαλύτερη διαφορά ανάμεσα στις δυο εκδόσεις είναι ότι το AI2 θα τρέχει εξ ολοκλήρου δικτυακά, ενώ θα ενσωματώνει νέες δυνατότητες και χαρακτηριστικά.

Ομοιότητες - διαφορές Scratch & AI

Μετά την παρουσίαση των δυο ΝΡΕ, κάποιος θα κατέληγε στο συμπέρασμα ότι το AI θα μπορούσε να θεωρηθεί ως το Scratch για τις φορητές συσκευές (Wolber, 2011). Εκτίμηση η οποία επιβεβαιώνεται από τον Mitch Resnick (Resnick, 2010), που δήλωνε τον Ιούλιο του 2010 ότι: «*τα μέλη της ομάδας ανάπτυξης του Scratch μοιράζονται ιδέες με τα μέλη της ομάδας ανάπτυξης του AI. Τα 2 έργα έχουν παρόμοιους στόχους, αν και το AI είναι προσανατολισμένο στα χαρακτηριστικά και στις δυνατότητες των ΕΚΣ. Κάποιες από τις συζητήσεις μας έχουν επηρεάσει σχετικά με τι δυνατότητες του Scratch, ιδίως στον τρόπο με τον οποίο το Scratch οργανώνει, προσπελάζει και διαχειρίζεται τα δεδομένα στο διαδίκτυο*». Αλλά και ο Harold Abelson (Abelson, 2010) σε απάντησή του σε δικτυακό φόρουμ χρηστών, δήλωνε χαρακτηριστικά: «*Οι ρίζες μεταξύ του AI και του Scratch είναι μακριές και βαθιές, οι οποίες πηγάζουν από μια κοινή θεώρηση για τη φιλοσοφία της εκπαίδευσης που μας πηγαίνει πίσω στο Seymour Papert και στη Logo, στα τέλη της δεκαετίας του 1960*». Σε άλλη ερώτηση στο ίδιο φόρουμ, σχετικά με το διαμοιρασμό κώδικα, ανέφερε: «*δεν υπάρχει στην πραγματικότητα κοινόχρηστος κώδικας μεταξύ των δυο εφαρμογών. Το AI κάνει χρήση του «OpenBlocks», ένα έργο του MIT το οποίο χρησιμοποιεί εξίσου και το Scratch*».

Λαμβάνοντας υπόψη διεθνείς έρευνες (Liu et al., 2013; Gray et al., 2012; Roy et al., 2012; Olabe et al., 2011; Utting et al., 2010), θα λέγαμε ότι τα δυο ΝΡΕ παρουσιάζουν τις ακόλουθες ομοιότητες:

- Το AI μοιράζεται την εμφάνιση και την αίσθηση του Scratch, από το οποίο αντλεί και μια αξιοσημείωτη δεξαμενή χρηστών. Ως εκ τούτου, η μετάβαση ενός μαθητή από το Scratch στο AI γίνεται απρόσκοπτα και δίχως ιδιαίτερα προβλήματα. Στο AI, όπως και στο Scratch, οι μαθητές μετακινούν πλακίδια εντολών σαν να ενώνουν κομμάτια παζλ προκειμένου να δημιουργήσουν τα προγράμματά τους.
- Ο προγραμματισμός με το AI και το Scratch βοηθά τους αρχάριους μαθητές να επικεντρώνονται περισσότερο στην επίλυση προβλημάτων και λιγότερο στο συντακτικό της γλώσσας. Και τα δυο ΝΡΕ εμπλέκουν ενεργά το χρήστη, επιτρέποντάς του να γράψει προγράμματα για καταστάσεις που συνδέονται άμεσα με τα ενδιαφέροντά του, σε αντίθεση με τη συμβατική διδασκαλία του προγραμματισμού.

Στον αντίποδα παρουσιάζουν και διαφορές, οι οποίες είναι:

- Στο AI η διεπαφή του λογισμικού αποτελείται από τρία διακριτά μέρη (σχεδίαση, συντάκτη, προσομοιωτής ή φορητή συσκευή). Όταν ο χρήστης δημιουργεί μια εφαρμογή, χρειάζεται συνεχώς να κινείται μεταξύ των τριών αυτών μερών. Αντίθετα,

στο Scratch ο χρήστης χρειάζεται να έχει μπροστά του μια μόνο οθόνη. Οι ενέργειες που απαιτούνται για την προετοιμασία του συστήματος για την εκτέλεση του AI είναι σαφώς δυσκολότερες από τις αντίστοιχες του Scratch

- Ένα από τα μεγαλύτερα πλεονεκτήματα του Scratch είναι η δημοφιλής συλλογή έργων και πηγαίου κώδικα, επιτρέποντας τη λήψη ενός μεμονωμένου μπλοκ εντολών από μια εφαρμογή και την ενσωμάτωσή του σε μια άλλη. Το AI δε διαθέτει κάποια επίσημη υποστήριξη για μια αντίστοιχη συλλογή. Μόλις πρόσφατα η ομάδα έργου του AI έχει ξεκινήσει πειραματικά τη δημιουργία μιας αντίστοιχης συλλογής (beta στάδιο).
- Το Scratch δε διαθέτει διαδικασίες. Στο AI ο χρήστης μπορεί να ορίσει τις δικές του διαδικασίες με παραμέτρους, ενσωματώνοντας και απλές δομές δεδομένων (λίστες).
- Το AI παρέχει ένα πιο ολοκληρωμένο περιβάλλον για ανάπτυξη αντικειμενοστραφών προγραμμάτων, διευκολύνοντας τη μελλοντική μεταπήδηση του χρήστη σε πιο εξειδικευμένες γλώσσες προγραμματισμού, όπως την Python ή την Java, μέσω και του χαρακτηριστικού Java Bridge.
- Το AI προσφέρει πλουσιότερες μαθησιακές εμπειρίες συγκριτικά με το Scratch, καθώς με το AI ένας μαθητής μπορεί να δημιουργήσει οτιδήποτε θα δημιουργούσε και με το Scratch, αλλά επιπρόσθετα και εφαρμογές που περιλαμβάνουν φυσικές αλληλεπιδράσεις μέσω λειτουργιών drag and drop.
- Η εκτέλεση των εντολών στο AI δεν είναι ορατή, σε σχέση με το Scratch, στο οποίο, για παράδειγμα, τα προς εκτέλεση μπλοκ κώδικα έχουν λευκό περίγραμμα, για να υποδηλώσουν ότι είναι ενεργά.
- Το Scratch παρέχει πλήρη υποστήριξη της ελληνικής γλώσσας, ενώ το AI υπάρχει μόνο στην Αγγλική.

Στον πίνακα 1 που ακολουθεί συνοψίζονται ορισμένες από τις διαφορές ως προς τα χαρακτηριστικά των περιβαλλόντων που μόλις περιγράψαμε.

Πίνακας 1. Διαφορές App Inventor & Scratch

Περιγραφή	App Inventor	Scratch
Εγκατάσταση	Σύνθετη	Απλή
Διεπαφή Χρήστη	3 Μέρη	1 Οθόνη
Συλλογή Εφαρμογών	Εν μέρει διαθέσιμη	Διαθέσιμη
Διαδικασίες	Ναι	Όχι
Οπτικός Προγραμματισμός	Ναι	Όχι
Εφαρμογές για Έξυπνες Συσκευές	Ναι	Όχι
Εφαρμογές φυσικής αλληλεπίδρασης	Ναι	Όχι

Συμπεράσματα - Συζήτηση

Κατά την άποψή μας, δεν υπάρχει ένας ξεκάθαρος νικητής. Θεωρούμε ότι το εκάστοτε πλαίσιο ή εκπαιδευτικό περιβάλλον το οποίο ορίζει τη χρήση του Scratch ή του AI. Και τα δυο ΝΡΕ έχουν τα δυνατά και αδύναμα σημεία τους. Το Scratch ενδεχομένως να είναι πιο κατάλληλο για διδασκαλία σε μικρούς μαθητές ή σε προγράμματα σπουδών, των οποίων ο πρωταρχικός στόχος είναι η ευχάριστη ενασχόληση και όχι μια πιο σοβαρή γνωριμία με προγραμματισμό. Το AI ενδεχόμενα να είναι πιο κατάλληλο για μια πιο επίσημη εισαγωγή στον προγραμματισμό, στην οποία ο τελικός στόχος θα είναι η ανάπτυξη της ικανότητας του προγραμματισμού και η μετάβαση σε μια συμβατική γλώσσα (Roy et al., 2012).

Για τη διδασκαλία του προγραμματισμού στο Δημοτικό Σχολείο ή στις πρώτες τάξεις του Γυμνασίου, θεωρούμε ότι είναι προτιμότερο κάποιος να ξεκινήσει με το Scratch και να

συνεχίσει με το AI στις επόμενες τάξεις ή εκπαιδευτικές βαθμίδες. Με τον τρόπο αυτό, η μετάβαση ανάμεσα στα δυο περιβάλλοντα γίνεται πολύ ομαλά λόγω της ίδιας αίσθησης (look & feel) που αποπνέουν. Το AI σε σχέση με το Scratch φαίνεται να αναδύεται ως ισχυρότερο προς χρήση προγραμματιστικό περιβάλλον στο Γενικό Λύκειο και στα ΕΠΑΛ, καθώς θεωρείται ιδανικό εργαλείο για την κινητοποίηση των ενδιαφερόντων των νέων μαθητών μέσω της σύνδεσης με τις κινητές τους συσκευές (Roy 2012). Άλλωστε, και στο Πρόγραμμα Σπουδών του μαθήματος «Εφαρμογές Πληροφορικής» της Α΄ τάξης του νέου Γενικού Λυκείου (ΦΕΚ 932, 14.04.2014), στους στόχους του μαθήματος αναφέρεται για πρώτη φορά ότι ο μαθητής θα πρέπει να μπορεί να σχεδιάζει και να αναπτύσσει μικροεφαρμογές των «έξυπνων» κινητών συσκευών. Αναμφίβολα, οι παραπάνω υποθέσεις κινούνται σε θεωρητικό επίπεδο, μιας και απουσιάζει το αντίστοιχο ερευνητικό υπόβαθρο, κύρια λόγω του σύντομου χρονικού διαστήματος εμφάνισης του AI. Ωστόσο, τα πρώτα αποτελέσματα από την εισαγωγή του AI στο σχολικό περιβάλλον δείχνουν να επιβεβαιώνουν τους παραπάνω ισχυρισμούς (Ορφανάκης & Παπαδάκης, 2014). Ο Wolber (2010) θεωρεί το AI ως το σωστό εργαλείο στο σωστό χρόνο. Ενώ το Scratch με την ευρεία διάδοσή του έχει αποδείξει την αξία και την ευχρηστία των ΝΡΕ, το AI έρχεται να συμπληρώσει τη δίψα των χρηστών για φορητές συσκευές. Η παρούσα μελέτη ευελπιστούμε να αποτελέσει ένα χρήσιμο οδηγό για τους εκπαιδευτικούς, οι οποίοι σχεδιάζουν εισαγωγικά μαθήματα και δραστηριότητες προγραμματισμού, προσελκύοντας περισσότερους μαθητές στον κόσμο της πληροφορικής.

Αναφορές

Abelson, H. (2010). *Is App Inventor based on Scratch?* [Online forum comment]. Retrieved 25 November 2013 from https://groups.google.com/forum/#!topic/appinventor/LvhZz8NZZ5g

Abelson, H. (2009). *App Inventor for Android. Google Research Blog*. Retrieved 25 November 2013 from http://googleresearch.blogspot.com/2009/07/appinventor-for-android.html

Khuloud, A., & Gestwicki, P. (2013). Studio-based learning and app inventor for android in an introductory CS course for non-majors. In *Proceeding of the 44th ACM technical symposium on Computer science education (SIGCSE '13)* (pp. 287-292). New York: ACM.

Federici, S. (2011). A minimal, extensible, drag-and-drop implementation of the C programming language. In *Proceedings of the 2011 conference on Information technology education* (SIGITE '11) (pp. 191-196). New York: ACM.

Ford, J. L. (2008). *Scratch Programming for Teens*. Boston, MA: Course Technology Press.

Forte, A., & Guzdial, M. (2004). Computers for Communication, Not Calculation: Media as a Motivation and Context for Learning. *Proceedings of the 37th Annual Hawaii International Conference on System Sciences*, (pp. 1- 10).

Freudenthal, E., Roy, M., Ogrey, A., Magoc, T., & Siegel, A. (2010). Media Propelled Computational Thinking. *Proceedings of the 41st ACM technical symposium on Computer science education*, (pp. 37-41).

Gans, P. (2010). The benefits of using scratch to introduce basic programming concepts in the elementary classroom: poster session. *Journal of Computing Sciences in Colleges*, 25(6), 235-236.

Gray, J., Abelson, H., Wolber, D., & Friend, M. (2012). Teaching CS principles with app inventor. In *Proceedings of the 50th Annual Southeast Regional Conference* (ACM-SE '12). ACM, NY, USA, 405-406.

Guzdial, M. (2004). Programming environments for novices. In S. Fincher and M. Petre (Eds.), *Computer Science Education Research (pp.* , 127-154). London: Taylor & Francis.

Harvey, B., & Monig, J. (2010). Bringing 'No Ceiling' to Scratch: Can One Language Serve Kids and Computer Scientists? In *proceedings of Constructionism 2010 (Paris)*, 2010.

Hsu, Y.-C., Rice, K., & Dawley, L. (2012). Empowering educators with Google's Android App Inventor: An online workshop in mobile app design. *British Journal of Educational Technology*, 43(1), E1-E5.

Krul, K. Y. (2012). *Teaching Control Structures Using App Inventor*. Master thesis. Retrieved 26 November 2013 from http://igitur-archive.library.uu.nl/student-theses/2012-0905-200808/UUindex.html

Liu, J., Lin, C-H., Potter, P., Philip, E., Zebulun, H., Barnett, D-B., & Singleton, M. (2013). Going mobile with app inventor for android: a one-week computing workshop for K-12 teachers. In *Proceeding of the 44th ACM technical symposium on Computer science education* (SIGCSE '13), (pp. 433-438). NY, USA.

Loukides, M. (2010). *App Inventor and the culture wars*. Retrieved 26 November 2013 from http://radar.oreilly.com/2010/07/culture-wars.html

Margulieux, L. E., Guzdial, M., & Catrambone, R. (2012). Subgoal-labeled instructional material improves performance and transfer in learning to develop mobile applications. *Proceedings of the ninth annual international conference on International computing education research - ICER '12*, 71.

Massachusetts Institute of Technology (2013). What is App Inventor? Retrieved 01 December 2013 from http://appinventor.mit.edu/explore/content/what-app-inventor.html

Olabe, J.C., Olabe, M.A., Basogain, X., & Castaño, C. (2011). Programming and robotics with Scratch in primary education. In A. Mendez-Vilas (Ed.). *Education in a Technological World: Communicating current and Emerging Research and Technological Efforts*, (pp. 356–363). Badajoz - Spain: Formatex.

Papert, S. (1993). *Mindstorms: Children, Computers, and Powerful Ideas*. New York: Basic Books.

Resnick, M., Maloney, J, Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., & Kafai, Y. (2009). Scratch: programming for all, *Communications of the ACM*, 52(11), 60-67.

Resnick, M. (2008). Sowing the Seeds for a More Creative Society. *Learning & Leading with Technology*, 35(4),18-22.

Resnick, M. (2010). *Google's App Inventor*. [Online forum comment]. Retrieved 01 December 2013 from http://scratched.media.mit.edu/discussions/news-and-announcements/googles-app-inventor

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

Roy, K., Rousse, W.C., & DeMeritt, D.B. (2012). Comparing the mobile novice programming environments: App Inventor for Android vs. GameSalad. In *Proceedings of the 2012 IEEE Frontiers in Education Conference (FIE)* (FIE '12) (pp. 1-6). Washington, DC: IEEE Computer Society.

Roy, K. (2012). App inventor for android: report from a summer camp. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education* (SIGCSE '12). ACM, New York, NY, USA, 283-288.

Siegle, D. (2009). Developing Student Programming and Problem-Solving Skills with Visual Basic. *Gifted Child Today*, 32(4), 24-29.(ERIC Document Reproduction Service No. EJ860950).

Utting, I., Cooper, S., Kolling, M., Maloney, J., & Resnick, M. (2010). Alice, greenfoot and scratch –A discussion. *ACM Transactions on Computing Education*. 10, 4.

Wagner, A., Gray, J., Corley, J., & Wolber, D. (2013). Using app inventor in a K-12 summer camp. In *Proceeding of the 44th ACM technical symposium on Computer science education* (SIGCSE '13) (pp. 621-626). New York: ACM.

Wilson, A., & Moffat, D. (2010). Evaluating scratch to introduce younger schoolchildren to programming. J. Lawrance, R. Bellamy (Eds.), *Proceedings of the 22nd annual workshop of the psychology of programming interest group – PPIG2010* (pp. 64-74). September 19–22, 2010, Universidad Carlos III de Madrid, Leganés, Spain (2010).

Wolber, D. (2010). *A blocks language for mobile phones: App Inventor for Android*. In E. Canessa & M. Zennaro (Eds.), mScience: Sensing, computing and dissemination. Trieste, Italy: ICTP—The Abdus Salam International Centre for Theoretical Physics.

Wolber, D. (2011). App inventor and real-world motivation. In *Proceedings of the 42nd ACM technical symposium on Computer science education* (SIGCSE '11) (pp. 601-606). ACM, New York, NY, USA.

Zaranis, N., Kalogiannakis, M., & Papadakis, S. (2013). Using Mobile Devices for Teaching Realistic Mathematics in Kindergarten Education. *Creative Education (Special Issue in Preschool Education)*, 4(7A1), 1-10.

Ορφανάκης, Β., & Παπαδάκης, Στ. (2014). Προγραμματίζοντας τα Lego Mindstorms NXT με τη χρήση του App Inventor. Μια πρόταση για τη διδασκαλία των μαθημάτων Πληροφορικής του Γενικού Λυκείου. *Πρακτικά 8ου Πανελλήνιου Συνεδρίου Καθηγητών Πληροφορικής «Η Πληροφορική στην Πρωτοβάθμια και Δευτεροβάθμια Εκπαίδευση – Διδασκαλία και Διδακτική»*, Πανεπιστήμιο Θεσσαλίας, Βόλος, 28-30 Μαρτίου 2014.