Γραφικά Πραγματικού Χρόνου: μια διδακτική πρόταση για την εισαγωγή στον Αντικειμενοστραφή Προγραμματισμό

Κ. Βασιλειάδης[1], Ε. Ρόμπολα[2]

[1] Μηχανικός Η/Υ και Πληροφορικής, μέλος HGDA
vasileiadi@ceid.upatras.gr
[2] Εκπαιδευτικός ΠΕ19 5ου ΓΕΛ Βύρωνα
eleni.rompola@gmail.com

Περίληψη

Η διδασκαλία του Αντικειμενοστραφούς Προγραμματισμού συνοδεύεται από ποικίλες δυσκολίες ιδίως εάν έχει προηγηθεί σε βάθος διδασκαλία Διαδικαστικού Προγραμματισμού. Απαιτείται σημαντικός χρόνος για την αλλαγή του τρόπου σκέψης, ώστε τελικά να μην δικαιολογείται η απόρριψη του Αντικειμενοστραφούς Προγραμματισμού ως εισαγωγικού διδακτικού αντικειμένου. Ο βασικός στόχος αυτής της εργασίας είναι να δείξει ότι η επιλογή ενός αρχικού παραδείγματος, αρκετά σύνθετου ώστε να αντανακλά όλες τις έννοιες του αντικειμενοστραφούς προγραμματισμού και ικανού να δημιουργήσει ένα πλαίσιο αυθεντικής μάθησης, σε συνδυασμό με μια αντικειμενοστραφή γλώσσα προγραμματισμού με αρκετά απλή σύνταξη ώστε να μην δυσκολεύονται οι μαθητές, μπορεί να λειτουργήσει αποτελεσματικά.

Λέξεις κλειδιά: Αντικειμενοστραφής Προγραμματισμός, Processing, Αυθεντική Μάθηση, Γραφικά Πραγματικού Χρόνου.

1.Εισαγωγή

Η διδασκαλία και εκμάθηση του αντικειμενοστραφούς προγραμματισμού (στο εξής ΑΠ) συνοδεύεται από ποικίλες δυσκολίες και παρανοήσεις, οι οποίες εξαρτώνται σε μεγάλο βαθμό από την προηγούμενη εμπειρία των μαθητών. Μαθητές εξοικειωμένοι με γλώσσες διαδικαστικού προγραμματισμού δυσκολεύονται περισσότερο. Συνεπώς, η διδακτική προσέγγιση που θα ακολουθήσει ο εκπαιδευτικός για να εισάγει τους μαθητές στον ΑΠ θα πρέπει να είναι ελκυστική, να στηρίζεται σε καθημερινές εμπειρίες των μαθητών οι οποίες θα λειτουργήσουν ως βάση πάνω στην οποία θα οικοδομηθεί η νέα γνώση, και να μπορεί να αποτρέψει την εμφάνιση των παραπάνω παρανοήσεων και δυσκολιών.

Στην παρούσα εργασία, παρουσιάζουμε μια διδακτική πρόταση, η οποία υιοθετεί την μεθοδολογία ανάπτυξης Γραφικών Πραγματικού Χρόνου, την προσαρμόζει στα δεδομένα (πλαίσιο και πρότερες γνώσεις) μαθητών Γενικού Λυκείου και τελικά

επιτυγχάνει την ομαλή εισαγωγή τους στον ΑΠ μέσα από δημιουργικές διαδικασίες. Επιδίωξή μας είναι η δημιουργία ενός περιβάλλοντος αυθεντικής μάθησης, μέσα στο οποίο οι μαθητές θα μάθουν επιλύοντας προβλήματα (problem based learning), συνεργατικά (Savery & Thomas, 2001) (Ryoo, Fonseca, & Janzen, 2008). Δεν στοχεύουμε στην εκμάθηση συγκεκριμένης αντικειμενοστραφούς γλώσσας προγραμματισμού, αλλά στην απόκτηση δεξιοτήτων μοντελοποίησης και αναπαράστασης των πραγματικών αντικειμένων ως προγραμματιστικά αντικείμενα. Ο προγραμματισμός γίνεται στο περιβάλλον της γλώσσας Processing.

2. Θεωρητικό Υπόβαθρο

2.1 Δυσκολίες και Παρανοήσεις των Μαθητών

Στην διεθνή βιβλιογραφία υπάρχουν καταγεγραμμένες χαρακτηριστικές παρανοήσεις των μαθητών, όπως (ΕΑΙΤΥ, 2008): (α) Συγχέουν τις έννοιες της κλάσης και του αντικειμένου και δεν μπορούν να τις διαχωρίσουν, (β) Πιστεύουν ότι μια κλάση είναι μια συλλογή αντικειμένων και όχι ένα πρότυπο δημιουργίας αντικειμένων, (γ) Δυσκολία κατανόησης της στατικής φύσης μιας κλάσης και της δυναμικής φύσης ενός αντικειμένου, (δ) Θεωρούν πως κάθε αντικείμενο αποτελεί απλά ένα «περιτύλιγμα» μιας μεταβλητής, (ε) Δυσκολία κατανόησης της επίδρασης που έχει η εκτέλεση μιας μεθόδου στην κατάσταση ενός αντικειμένου. Σε αρκετές περιπτώσεις δεν γίνεται αντιληπτό ότι η αρχική κατάσταση ενός αντικειμένου καθορίζεται όταν αυτό δημιουργείται και στη συνέχεια ενημερώνεται με την εκτέλεση μεθόδων από το αντικείμενο, (στ) Θεωρούν πως δύο αντικείμενα της ίδιας κλάσης δεν μπορούν να έχουν τις ίδιες τιμές πεδίων, (ζ) Σε αρκετές περιπτώσεις δεν γίνεται αντιληπτό ότι η συμπεριφορά ενός αντικειμένου μπορεί να αλλάξει ουσιαστικά ανάλογα με την κατάστασή του, (η) Δεν αξιοποιείται η λειτουργικότητα των μεθόδων και επαναλαμβάνεται ένα τμήμα κώδικα αντί να ορίζεται μια μέθοδος, (θ) Πιστεύουν πως κάθε μέθοδος μπορεί να κληθεί μόνο μία φορά, (ι) Δυσκολία κατανόησης του γεγονότος ότι μια μέθοδος μπορεί να κληθεί για οποιοδήποτε αντικείμενο μιας κλάσης, (ια) Θεωρούν πως οι μέθοδοι εκτελούνται σύμφωνα με τη σειρά που εμφανίζονται στον ορισμό της κλάσης.

Οι παραπάνω παρανοήσεις οφείλονται και στο γεγονός πως η εισαγωγή των μαθητών στον προγραμματισμό γίνεται συχνότατα μέσω του διαδικαστικού προγραμματισμού. Οι υποστηρικτές αυτής της μεθόδου την προτιμούν διότι τα προγράμματα μπορούν να είναι πολύ μικρά και απλά και η εισαγωγή στις σύνθετες προγραμματιστικές δομές να γίνεται σταδιακά με προσθήκη γραμμών κώδικα, όπως υποστηρίζουν. Ωστόσο, έχει παρατηρηθεί πως προγραμματιστές με σοβαρή εμπειρία στον διαδικαστικό προγραμματισμό χρειάζονται έως και 18 μήνες για να επιτύχουν την αλλαγή σκέψης που απαιτεί ο ΑΠ (Kölling, 1999).

Πιστεύουμε πως η εισαγωγή των μαθητών στον προγραμματισμό πρέπει να γίνεται μέσω του ΑΠ (Bergin, 2000). Η βασική διαφορά του ΑΠ από τον διαδικαστικό, βρίσκεται στην μοντελοποίηση του κόσμου και των αντικειμένων του προς επίλυση προβλήματος, ενώ ο κώδικας των μεθόδων είναι στην πραγματικότητα διαδικαστικός. Συνεπώς, εισάγοντας τους μαθητές στον ΑΠ, επιτυγχάνεται η ταυτόχρονη εισαγωγή τους στις προγραμματιστικές δομές του διαδικαστικού προγραμματισμού. Η πραγματική πρόκληση είναι η εύρεση ενός πρώτου παραδείγματος, το οποίο θα είναι αρκετά σύνθετο ώστε να αντανακλά όλες τις έννοιες του ΑΠ αλλά ταυτόχρονα αρκετά απλό ώστε η εισαγωγή των μαθητών στις προγραμματιστικές δομές να γίνεται εύκολα (Goldwasser & Letscher, 2008).

Στην διδακτική πρόταση που παρουσιάζουμε αποφεύγεται η στείρα θεωρητική παρουσίαση των εννοιών του ΑΠ και η κατανόησή τους προκύπτει αβίαστα μέσα από δραστηριότητες σχεδίασης (Moritz, et al., 2005) και μοντελοποίησης. Ωστόσο, ακόμη και ο σχεδιασμός και η μοντελοποίηση απλών αντικειμένων παρουσιάζει δυσκολίες για τους μαθητές, διότι τα τελευταία χρόνια η διδασκαλία της πληροφορικής έχει περιοριστεί στην εκμάθηση εργαλείων και όχι στην απόκτηση αναλυτικής και συνθετικής σκέψης. Για να μπορέσουν οι μαθητές να “μπουν στο πνεύμα” είναι απαραίτητη, πιστεύουμε, η δημιουργία ενός περιβάλλοντος αυθεντικής μάθησης στην τάξη.

2.2 Αυθεντική Μάθηση

Αυθεντική μάθηση είναι η πρόσκληση του πραγματικού κόσμου μέσα στην τάξη, ώστε οι μαθητές να μαθαίνουν μέσα σε πραγματικές συνθήκες, επιλύοντας πραγματικά προβλήματα (ή προβλήματα που θα μπορούσαν να είναι πραγματικά) και έχοντας την βεβαιότητα πως η εργασία τους δεν είναι θεωρητική αλλά οδηγεί σε ένα προϊόν το οποίο έχει πραγματική υπόσταση και χρησιμότητα. Ένα πολύ σημαντικό στοιχείο της αυθεντικής μάθησης είναι το γεγονός πως οι μαθητές παράγουν οι ίδιοι αντί να αναπαράγουν γνώση.

Συστατικά τα οποία συνθέτουν ένα περιβάλλον αυθεντικής μάθησης είναι μεταξύ άλλων και τα ακόλουθα (Herrington, Oliver, & Reeves, 2002): (α) αυθεντικό πλαίσιο (authentic context) το οποίο να υποδεικνύει με ποιον τρόπο η γνώση θα χρησιμεύσει και χρησιμοποιηθεί στην πραγματική ζωή, (β) δραστηριότητες οι οποίες να γίνονται και στην πραγματική ζωή, (γ) επαφή με ειδικούς ώστε οι μαθητές να μπορούν να παρακολουθήσουν την επιτέλεση ίδιου ή παρόμοιου έργου, (δ) ύπαρξη πολλών ρόλων και επεξεργασία ποικίλων απόψεων, (ε) συνεργασία μεταξύ των μαθητών για την επίλυση των προβλημάτων και την σύνθεση της νέας γνώσης, (στ) συμβουλευτική παρουσία του εκπαιδευτικού.

Στο πλαίσιο της διδακτικής μας πρότασης, η δημιουργία ενός περιβάλλοντος αυθεντικής μάθησης προτείνουμε να γίνει σε συνεργασία με τον HGDA (Hellenic Game Developers Association). Ο HGDA διοργανώνει κάθε χρόνο το Hellenic Game

Jam και συμμετέχει στο Global Game Jam. Οι διαγωνισμοί αυτοί, οι οποίοι γίνονται σε πανελλαδικό επίπεδο ο πρώτος και σε παγκόσμιο επίπεδο ο δεύτερος, διαρκούν 3 ημέρες. Η συμμετοχή είναι εθελοντική και ελεύθερη, όπως και η σύνθεση των ομάδων. Την πρώτη ημέρα ανακοινώνεται το θέμα πάνω στο οποίο κάθε ομάδα πρέπει να εμπνευστεί και να υλοποιήσει ένα videogame. Τα χαρακτηριστικά του παιχνιδιού (αρχιτεκτονική, αισθητική, πλατφόρμα, κλπ.) είναι επιλογή της κάθε ομάδας. Οι ομάδες πρέπει να έχουν ολοκληρώσει την εργασία τους σε 48 ώρες. Ακολουθεί η δημόσια παρουσίαση των videogames και ψηφίζεται το καλύτερο.

Προτείνουμε την πραγματοποίηση μιας εκπαιδευτικής επίσκεψης σε κάποιον από τους χώρους όπου πραγματοποιείται το Hellenic ή το Global Game Jam, ιδίως κατά την $2^{η}$ ημέρα της διοργάνωσης. Οι μαθητές θα δουν από κοντά πως σχεδιάζεται και πως προγραμματίζεται ένα παιχνίδι. Θα δουν ότι το τελικό καταπληκτικό αποτέλεσμα περνάει μέσα από διάφορες φάσεις κατά τις οποίες ούτε καν προσεγγίζει την τελική του μορφή. Είναι οι φάσεις της σχεδίασης και της τμηματικής ανάπτυξης του παιχνιδιού. Συζητώντας με τα μέλη των ομάδων θα δουν πως γεννιούνται και πως απορρίπτονται ιδέες. Τελικά, θα δουν πως ο "μάγος" που συνθέτει όλα τα επιμέρους κομμάτια ώστε να παραχθεί το τελικό προϊόν δεν είναι άλλος από τον προγραμματιστή. Είναι αξιοσημείωτη η προθυμία των ομάδων να μιλήσουν στους μαθητές, η ελευθερία με την οποία οι μαθητές επικοινωνούν με τους ανθρώπους αυτούς και τελικά η "απομυθοποίηση" των videogames καθώς οι μαθητές συνειδητοποιούν ότι μπορείς να φτιάξεις τα πάντα αρκεί να έχεις τις απαραίτητες γνώσεις και όρεξη για δουλειά. Επιστρέφοντας στην σχολική τάξη, το έδαφος είναι ήδη πρόσφορο και ο εκπαιδευτικός μπορεί πολύ εύκολα να πείσει τους μαθητές ότι "δεν είναι και τόσο δύσκολο να φτιάξουμε κι εμείς ένα παιχνίδι. Θα το κάνουμε σταδιακά όπως τα παιδιά στο game jam". Οι μαθητές πιθανότατα αμφιβάλλουν για τις προγραμματιστικές τους ικανότητες, ωστόσο η ιδέα τους ενθουσιάζει. Χαρακτηριστική ήταν η αντίδραση μαθητή μας ο οποίος πρότεινε "να κάνουμε ένα School Game Jam"! Ήδη έχει δημιουργηθεί περιβάλλον αυθεντικής μάθησης με όλα τα χαρακτηριστικά που αναφέραμε νωρίτερα.

Κατά την εξέλιξη της διδασκαλίας, είναι σημαντικό οι μαθητές να νιώθουν ελεύθεροι στις επιλογές τους, ώστε να διατηρηθεί η αίσθηση που αποκόμισαν από το Game Jam. Αυτό απαιτεί ευελιξία από την πλευρά του εκπαιδευτικού και ίσως λίγη παραπάνω προεργασία, ώστε να είναι σε θέση να φτάσει τους μαθητές στον επιθυμητό στόχο χωρίς να χαθεί η αίσθηση της ελεύθερης δημιουργικότητας.

2.3 Γραφικά Πραγματικού Χρόνου – Processing

Τα Γραφικά Πραγματικού Χρόνου (real time graphics) αποτελούν έναν σημαντικό κλάδο της Πληροφορικής, όχι ιδιαίτερα ανεπτυγμένο στην Ελλάδα. Χαρακτηριστικοί εκπρόσωποι των Γραφικών Πραγματικού Χρόνου είναι τα Videogames και η Demoscene. Τα Videogames μπορούν να θεωρηθούν οικείο περιβάλλον για την

πλειοψηφία των μαθητών, τα αντικείμενα που συνθέτουν ένα τέτοιο περιβάλλον είναι για τους μαθητές εξίσου πραγματικά με τα αντικείμενα που χρησιμοποιούν στην καθημερινή τους ζωή και επομένως μπορούν να αποτελέσουν βάση για την ανάπτυξη ικανοτήτων μοντελοποίησης των αντικειμένων αυτών ώστε μέσα από την διαδικασία αυτή να προκύψουν οι αφηρημένοι ορισμοί των κλάσεων (Overmars, 2004). Η Demoscene διαφέρει από τα Videogames ως προς τα τελικά προϊόντα στα οποία αποσκοπεί. Τα Demos είναι καλλιτεχνικές δημιουργίες που στοχεύουν στο να αναδείξουν τις ικανότητες του προγραμματιστή, την φαντασία και δημιουργικότητά του. Προτείνουμε την δημιουργία Demo ως εναλλακτική δραστηριότητα, στην περίπτωση που υπάρξουν μαθητές χωρίς ενδιαφέρον για την δημιουργία Videogame.

Το περιβάλλον προγραμματισμού Processing δημιουργήθηκε για να παρέχει μια πλούσια σε χαρακτηριστικά γλώσσα, η οποία να υποστηρίζει την δημιουργία γραφικών πραγματικού χρόνου. Σύμφωνα με τους δημιουργούς της (Reas & Fry, 2007) “Processing is an open source programming language and environment for people who want to program images, animation, and interactions. It is used by students, artists, designers, researchers, and hobbyists for learning, prototyping, and production. It is created to teach fundamentals of computer programming within a visual context and to serve as a software sketchbook and professional production tool. Processing is an alternative to proprietary software tools in the same domain.”

Η γλώσσα προγραμματισμού Processing είναι ιδιαιτέρως κατάλληλη για αρχάριους προγραμματιστές. Διαθέτει όλα τα χαρακτηριστικά και τις δυνατότητες γλωσσών ΑΠ, αλλά απλούστατη σύνταξη, γεγονός που διευκολύνει τους μαθητές. Η επαναληπτική δομή της ίδιας της γλώσσας καθώς και η απλούστατη διαχείριση γεγονότων (events) που παρέχει, επιτρέπουν στους μαθητές να δημιουργήσουν πολύ γρήγορα αλληλεπιδραστικές εφαρμογές, χωρίς για παράδειγμα να χρειαστεί να χρησιμοποιήσουν δομή επανάληψης. Δεν αποκρύπτει από τον μαθητή την διαδικασία μεταγλώττισης και εκσφαλμάτωσης, γεγονός ιδιαίτερα σημαντικό. Ο μαθητής είναι σε θέση να διορθώσει τα λογικά λάθη του προγράμματος και να δει άμεσα το αποτέλεσμα. Στην Processing δεν υπάρχει κάτι που να γίνεται “μαγικά”, όπως επίσης δεν υπάρχει κάτι που να γίνεται δύσκολα.

Συνοπτικά, τα πλεονεκτήματα της Processing τα οποία την καθιστούν κατά την γνώμη μας ιδανική επιλογή ως εισαγωγική γλώσσα προγραμματισμού για μαθητές Λυκείου, είναι τα εξής (Meyseburg, 2011): (α) open-source και multi-platform (Linux, OSX, Windows), (β) απλούστατο προγραμματιστικό περιβάλλον (IDE), το οποίο παρέχει βοηθητικές λειτουργίες κατά την σύνταξη των προγραμμάτων (syntax coloring, auto format, κλπ.), (γ) εξαιρετική υποστήριξη γραφικών και ήχου (απλές στη χρήση τους rendering methods για δισδιάστατα και τρισδιάστατα γραφικά), (δ) event-driven (built-in μέθοδοι και μεταβλητές που διευκολύνουν εξαιρετικά την δημιουργία διαδραστικών προγραμμάτων και βοηθούν τους μαθητές να κατανοήσουν τις αρχές του event-driven προγραμματισμού πριν ακόμη ασχοληθούν με δομές

επιλογής και επανάληψης), (ε) java-based (σε αντίθεση με εργαλεία τύπου Scratch ή Alice, οι μαθητές μέσω της Processing υιοθετούν την βασική σύνταξη της Java και μπορούν πολύ εύκολα να μεταβούν σε αυτήν αργότερα), (στ) πληθώρα βοηθητικών μεθόδων και βιβλιοθηκών (για την διευκόλυνση αρχάριων ιδίως προγραμματιστών).

Στην διδακτική πρόταση που παρουσιάζουμε επιλέξαμε τον συνδυασμό Γραφικών Πραγματικού Χρόνου και Processing διότι (α) η δημιουργία ενός videogame/demo είναι ένα παράδειγμα ικανοποιητικά σύνθετο, ώστε να αντανακλά όλες τις έννοιες του ΑΠ και (β) τα ιδιαίτερα χαρακτηριστικά της Processing καθιστούν εύκολη την συγγραφή κώδικα για αρχάριους προγραμματιστές.

2.4 Μεθοδολογία Σχεδίασης και Υλοποίησης Videogames

Η μεθοδολογία ανάπτυξης Videogames, η οποία μπορεί φυσικά να εφαρμοστεί σε κάθε έργο γραφικών πραγματικού χρόνου το οποίο αναπτύσσεται συνεργατικά, συνοψίζεται στα εξής βήματα (Schell, 2008) (Varnado, 2011):

Καταιγισμός Ιδεών (brainstorming). Η ομάδα συσκέπτεται και αποφασίζει για ένα πλήθος στοιχείων: ποιος είναι ο ήρωας του παιχνιδιού, είδος παιχνιδιού, υπόθεση παιχνιδιού (story), χώρος και χρόνος (παρελθόν, μέλλον, παρόν) όπου διαδραματίζεται η υπόθεση του παιχνιδιού, σκοπός παιχνιδιού, μέσα (actions), looks (χαρακτήρες του παιχνιδιού, υπόλοιπα αντικείμενα, κλπ.), ηχητικό κομμάτι (εφέ, μουσική), σκοπός δημιουργίας του παιχνιδιού (διασκέδαση, εκπαίδευση, "να αλλάξει τον κόσμο", κέρδος, κλπ.), τι είναι αυτό που το κάνει διαφορετικό από άλλα παιχνίδια του είδους του.

Δημιουργία του Design Document. Οι ιδέες και οι τελικές αποφάσεις της ομάδας καταγράφονται. Το έγγραφο που παράγεται (Design Document) αποτελεί την "βίβλο" της ομάδας και το σημείο αναφοράς κατά την τμηματική ανάπτυξη του παιχνιδιού. Το Design Document αποτελείται από τέσσερα βασικά κεφάλαια: Αισθητική (μουσική, ηχητικά εφέ, looks), Story (υπόθεση, χώρος, χρόνος), Mechanics (είδος παιχνιδιού, σκοπός, μέσα-actions), Γιατί το φτιάχνουμε και Γιατί θα το παίξουν.

Καθορισμός Απαιτήσεων. Η ομάδα αποφασίζει το προφίλ των ατόμων που θα χρειαστεί (π.χ. γραφίστας, μουσικός, προγραμματιστής) και κατά πόσο υπάρχουν τέτοια άτομα στην ομάδα ή πρέπει να αναζητηθούν, τα εργαλεία που θα απαιτηθούν για την ανάπτυξη του παιχνιδιού και κατά πόσο αυτά είναι διαθέσιμα, κλπ.

Ανάπτυξη (Development). Η ομάδα χωρίζεται σε υποομάδες κάθε μια από τις οποίες αναλαμβάνει την παράλληλη ανάπτυξη ενός τμήματος του παιχνιδιού (γραφικά, μουσική, κώδικας), σύμφωνα πάντα με το Design Document.

Ενοποίηση. Τελικά οι προγραμματιστές της ομάδας ενοποιούν την μουσική, τα γραφικά και τον κώδικα που έχει γραφεί τμηματικά, σε ένα ενιαίο παιχνίδι ώστε να παραχθεί το τελικό προϊόν και να ακολουθήσουν οι δοκιμές.

3. Η Διδακτική Πρόταση

3.1 Στόχοι και Οργάνωση της Διδασκαλίας

Στο πλαίσιο της γενικότερης υποβάθμισης της διδασκαλίας της Πληροφορικής στο Γενικό Λύκειο, δεν προβλέπεται μάθημα εισαγωγής στον προγραμματισμό ή έστω στην αλγοριθμική. Συνεπώς, οι μαθητές δεν έχουν ανεπτυγμένες σημαντικές δεξιότητες, όπως η αναλυτική και η συνθετική σκέψη καθώς και η ικανότητα αφαιρετικής αναπαράστασης ενός προβλήματος που το απαλλάσσει από στοιχεία που δεν επηρεάζουν την λύση. Επομένως, η απόπειρα εισαγωγής των μαθητών στον προγραμματισμό με τρόπο που να μην εκφυλίζεται σε ένα ακόμη βαρετό μάθημα, όπου *πρέπει να μάθουν και να θυμούνται* και στην χειρότερη περίπτωση *να αποστηθίζουν* προγραμματιστικές δομές ή αλγορίθμους, αποτελεί μια πραγματική πρόκληση για τον καθηγητή πληροφορικής. Κυρίως, απαιτεί αρκετό χρόνο χωρίς ασφυκτική πίεση προκαθορισμένης ύλης, όπως συμβαίνει στο μάθημα επιλογής της Α' τάξης.

Πιστεύουμε ότι στο Γενικό Λύκειο το καταλληλότερο μάθημα στο οποίο μπορεί να ενταχθεί ο ΑΠ είναι η Ερευνητική Εργασία (project). Στο 5ο ΓΕΛ Βύρωνα έχουμε υλοποιήσει επανειλημμένα την Ερευνητική Εργασία «Γραφικά Πραγματικού Χρόνου: δημιουργώντας από το μηδέν», όπου παράλληλα με το θεωρητικό μέρος (videogames, demoscene, εικονική και επαυξημένη πραγματικότητα, κλπ.) το οποίο διερευνούν οι μαθητές σύμφωνα με την μέθοδο Project, υλοποιούν στο σχολικό εργαστήριο πληροφορικής τα δικά τους παιχνίδια ή demos.

Κατά την διάρκεια του 1^{ου} τετραμήνου γίνεται η εισαγωγή στις έννοιες του ΑΠ, όπως θα περιγραφεί στη συνέχεια. Κατά την διάρκεια του 2^{ου} τετραμήνου οι ομάδες βελτιώνουν και επεκτείνουν τον κώδικά τους ή ξεκινούν κάτι καινούργιο, χωρίς να υπάρχει προκαθορισμένος στόχος από την πλευρά του εκπαιδευτικού. Σε αυτό το διάστημα ο εκπαιδευτικός εργάζεται ξεχωριστά με κάθε ομάδα, διορθώνει προβλήματα και επιλύει δυσκολίες, οι οποίες προκύπτουν καθώς η κάθε ομάδα έχει διαφορετικές ιδέες και κινείται προς διαφορετική κατεύθυνση. Κατά το σχολικό έτος 2013-2014 οι μαθητές μας διοργάνωσαν αυθόρμητα ένα School Game Jam. Επέλεξαν το κοινό θέμα "Candies" πάνω στο οποίο οι ομάδες δημιούργησαν διαφορετικά παιχνίδια τα οποία παρουσίασαν στο τέλος του project.

Τα προσδοκώμενα μαθησιακά αποτελέσματα κατά το 1^{ο} τετράμηνο, όπου και γίνεται η εισαγωγή στις βασικές έννοιες του ΑΠ, είναι τα εξής:

- Αναγνώριση των οντοτήτων που υπάρχουν στο παιχνίδι/demo και αναπαράστασή τους ως αντικείμενα.
- Διάκριση ιδιοτήτων ("ποιος είμαι") και ενεργειών ("τι κάνω") ως συστατικά στοιχεία των αντικειμένων.
- Διάκριση κλάσης και αντικειμένου.

- Υλοποίηση και κλήση των μεθόδων σύμφωνα με τις αρχές του ΑΠ.
- Υλοποίηση της αλληλεπίδρασης αντικειμένων και πρόβλεψη του αποτελέσματος που θα έχει στα συμμετέχοντα αντικείμενα.
- Πρόβλεψη της αλλαγής στην συμπεριφορά των αντικειμένων εξαιτίας αλλαγών στην κατάστασή τους, οι οποίες προκαλούνται από την κλήση άλλων μεθόδων.

Η έννοια της κληρονομικότητας θεωρήσαμε πως ξεπερνά το πλαίσιο ενός εισαγωγικού μαθήματος στον ΑΠ, γι' αυτό και δεν τέθηκε ως αρχικός στόχος. Ωστόσο, είναι μια έννοια με την οποία μπορούν οι μαθητές να ασχοληθούν στο 2ο τετράμηνο, όπου και επεκτείνουν/βελτιώνουν τον κώδικά τους.

Άλλα προσδοκώμενα μαθησιακά αποτελέσματα τα οποία θεωρούμε εξίσου σημαντικά, είναι τα εξής:

- Βελτίωση των δεξιοτήτων επίλυσης προβλημάτων με συστηματικό και δομημένο τρόπο.
- Συνεργατική οργάνωση ενός έργου, μέσω της διάσπασής του σε υπο-έργα.
- Σύνθεση μιας πολύπλοκης λύσης μέσω της αναλυτικής καταγραφής και αυστηρής τήρησης των προδιαγραφών (Design Document).
- Αποτελεσματική χρήση βασικών προγραμματιστικών δομών (if, for).
- Επιλεκτική χρήση global variables και αιτιολόγηση.
- Εφαρμογή διαδικασίας debugging και αποδοχή της ως αναγκαίας πρακτικής κατά την ανάπτυξη προγραμμάτων.
- Εισαγωγή στην βασική χρήση πινάκων (σε πολύ απλό επίπεδο).
- Εισαγωγή στις αρχές της τεχνητής νοημοσύνης (σε πολύ απλό επίπεδο).
- Εφαρμογή γνώσεων από διάφορα γνωστικά πεδία (π.χ. Γεωμετρία, Φυσική).

Υπάρχουν πολλά είδη παιχνιδιών τα οποία μπορούν να υλοποιήσουν οι μαθητές εφαρμόζοντας τις αρχές του ΑΠ. Προτείνουμε ως καταλληλότερο είδος παιχνιδιού μέσω του οποίου επιτυγχάνονται ευκολότερα οι παραπάνω στόχοι, ένα παιχνίδι τύπου Mario.

3.2 Φάσεις Διδασκαλίας

Οι φάσεις διδασκαλίας που περιγράφονται στην συνέχεια καλύπτουν τις φάσεις *Ανάπτυξης* και *Ενοποίησης* της μεθοδολογίας και δεν αντιστοιχούν μονοσήμαντα σε διδακτικές ώρες. Κατανέμονται κατά την κρίση του εκπαιδευτικού σε ολόκληρο το 1ο τετράμηνο, ανάλογα με την διαθεσιμότητα του σχολικού εργαστηρίου.

Φάση 1η – Αναγνώριση Οντοτήτων
Η φάση αυτή διεξάγεται με συζήτηση στην τάξη. Οι μαθητές έχουν ήδη καταγράψει στο Design Document το story του παιχνιδιού. Οι βασικές οντότητες που αναγνωρίζουν αρχικά και των οποίων τη συμπεριφορά μπορούν εύκολα να περιγράψουν είναι ο παίκτης, τα φιλικά αντικείμενα (π.χ. νομίσματα), οι εχθροί

(αρχικά στατικοί). Για κάθε οντότητα καλό είναι να καταγραφούν και όχι απλά να συζητηθούν, τα χαρακτηριστικά της (εμφάνιση-γραφικό, μέγεθος, θέση, κλπ.) και οι βασικές ενέργειες που θα μπορεί να κάνει (κίνηση δεξιά με ποιό πλήκτρο κλπ.).

Φάση 2η – Κατασκευή Πίστας

Ο χώρος της πίστας χωρίζεται σε ισομεγέθη τετράγωνα. Κάθε τετράγωνο μπορεί να θεωρηθεί ως ένα αντικείμενο, το οποίο χαρακτηρίζεται από δικό του γραφικό και δική του θέση. Συνεπώς όλα τα τετράγωνα μοιάζουν μεταξύ τους, παράγονται κατά κάποιο τρόπο από το "ίδιο καλούπι". Αυτό το "καλούπι" είναι η κλάση. Η ύπαρξη και μόνο του καλουπιού δεν συνεπάγεται την ύπαρξη αντικειμένων. Πρέπει να "χύσουμε υλικό μέσα στο καλούπι" για να δημιουργηθεί ένα αντικείμενο. Προγραμματιστικά, η περιγραφή του "καλουπιού" γίνεται ως εξής:

```
class Pista
{
  int x, y;                                  // attributes
  PImage fileName;


  Pista(int p_x, int p_y, PImage p_f)        // constructor
  {
    x = p_x;
    y = p_y;
    fileName = p_f;
  }


  void display()                            // method to display (draw) object
  {
      image(fileName, x, y+70, 100, 30);
  }

}
```

Εικόνα 1. *Δήλωση Κλάσης*

Η εκτέλεση του παραπάνω κώδικα εμφανίζει ένα κενό παράθυρο. Προγραμματιστικά "χύνουμε υλικό στο καλούπι" για να παραχθούν πραγματικά αντικείμενα ως εξής:

```
Pista P1, P2, P3;

void setup() {

  PImage f_pic1, f_pic2;

  size (800, 600);                          // set window size
  f_pic1 = loadImage("pista1.png");         // load image files
  f_pic2 = loadImage("pista2.png");

  P1 = new Pista(0, 400, f_pic1);             //create objects (stage)
  P2 = new Pista(100, 400, f_pic1);
  P3 = new Pista(200, 400, f_pic2);

}

void draw() {

  P1.display();                       //display objects (stage)
  P2.display();
  P3.display();

}
```

Εικόνα 2. *Δημιουργία τριών αντικειμένων*

Η κατασκευή της πίστας απαιτεί χρόνο και φαντασία ώστε να επιλεγούν τα κατάλληλα γραφικά που θα σχηματίσουν μια αισθητικά και λειτουργικά καλή πίστα. Με την ολοκλήρωση της πίστας οι μαθητές έχουν πλήρως κατανοήσει την διαφορά κλάσης–αντικειμένου, καθώς έχουν γράψει μία φορά τον κώδικα της κλάσης αλλά πολλές φορές τον κώδικα δημιουργίας αντικειμένου. Αν αποφασιστεί η δημιουργία μεγάλης πίστας, μπορεί ο εκπαιδευτικός να κάνει μια μικρή εισαγωγή στην χρηστικότητα των πινάκων σε συνδυασμό με την εντολή επανάληψης for.

Φάση 3η – Υλοποίηση Παίκτη

Το κεντρικό σημείο της υλοποίησης του παίκτη είναι η προσθήκη των μεθόδων ως δυνατότητες και όχι ως ενέργειες που εκτελούνται με τη σειρά. Ως χρήστες παιχνιδιών, οι μαθητές γνωρίζουν εμπειρικά την διαφορά του “μπορώ” (δήλωση μεθόδου) από το “κάνω” (κλήση μεθόδου). Γνωρίζουν δηλ. ότι ο παίκτης έχει ένα σύνολο δυνατοτήτων αλλά κάθε φορά επιλέγει ποια θα χρησιμοποιήσει και φυσικά δεν τις εκτελεί όλες με την ίδια σειρά.

```
class myPlayer {

  float xPos, yPos;                                    // attributes
  PImage fileName;


  myPlayer(float p_xPos, float p_yPos, PImage p_file)    // constructor
  {
    xPos = p_xPos;
    yPos = p_yPos;
    fileName = p_file;
  };


  void display()                                       // method to display (draw) object
  {
    image(fileName, xPos, yPos, 80, 80);
  };

  void moveR()                                         // method to move right one step
  {
    xPos = xPos + 2.5;
  }


  void moveD()                                        // method to move down (when falling)
  {
    yPos = yPos + 5;
  }

}
```

Εικόνα 3. *Απλή κλάση παίκτη με δυνατότητες κίνησης δεξιά και κάτω*

Το πότε θα κάνει τι ο παίκτης εξαρτάται από την επιλογή του χρήστη και τον σχεδιασμό του προγράμματος. Εδώ διαφαίνεται η ανάγκη διαχείρισης της εισόδου του χρήστη και γενικότερα των events. Η Processing διαθέτει μηχανισμό αναγνώρισης των events, αλλά η διαχείρισή τους (event handler) γίνεται από τον προγραμματιστή. Η εμπειρία των μαθητών συνάδει με αυτή την δομή του προγράμματος, διότι ως χρήστες παιχνιδιών γνωρίζουν πως όταν πατηθεί ένα πλήκτρο ή ένα κουμπί του ποντικιού συμβαίνει κάτι (συνήθως μικρό σε έκταση και συγκεκριμένο) και πως η συνολική εξέλιξη του παιχνιδιού συντίθεται από πάμπολλες τέτοιες “μικρές” ενέργειες.

Η παράθεση κώδικα Processing μέχρι στιγμής αποσκοπεί στο να καταδείξει την απλότητα της σύνταξης των προγραμμάτων. Από την εμπειρία μας στην τάξη, πιστεύουμε ότι οι μαθητές Λυκείου είναι σαφώς σε θέση να γράψουν κώδικα αυτής της μορφής και πως η εισαγωγή στον ΑΠ με εργαλεία αυτόματης παραγωγής κώδικα κατά κάποιο τρόπο τους υποτιμά.

```
Pista P1, P2, P3;
myPlayer pl;

void setup()
{
  PImage f_pic1, f_pic2, f_paixtaras;

  size(800, 600);                                  // set window size

  f_pic1 = loadImage("pista1.png");                // load image files
  f_pic2 = loadImage("pista2.png");
  f_paixtaras = loadImage("paixtaras.gif");

  P1 = new Pista(0, 400, f_pic1);                  // create objects (stage)
  P2 = new Pista(100, 400, f_pic1);
  P3 = new Pista(200, 400, f_pic2);

  pl = new myPlayer(0, 400, f_paixtaras);          // create player
}


void draw()
{
  P1.display();                                    // display objects (stage)
  P2.display();
  P3.display();

  pl.display();                                    // display player
}


void keyPressed()                                 // event handler
{
  if ((key == CODED) && (keyCode == RIGHT))       // if right arrow pressed
    pl.moveR();                                   // move player right
}
```

Εικόνα 4. *Δημιουργία πίστας και παίκτη με δυνατότητα κίνησης δεξιά*

Φάση 4η – Αλληλεπίδραση Αντικειμένων

Ως αποτέλεσμα των προηγούμενων φάσεων, οι μαθητές έχουν έτοιμη την πίστα του παιχνιδιού καθώς κι έναν παίκτη που μπορεί να εκτελεί απλή κίνηση. Το επόμενο στάδιο είναι η προσθήκη νέων αντικειμένων με τα οποία ο παίκτης θα αλληλεπιδρά. Τέτοια αντικείμενα είναι για παράδειγμα τα νομίσματα τα οποία ο παίκτης πρέπει να συλλέξει για να κερδίσει πόντους. Προφανώς χρειαζόμαστε ένα νέο "καλούπι" (κλάση) διότι ούτε το "καλούπι" της πίστας ούτε το "καλούπι" του παίκτη είναι κατάλληλα για την δημιουργία νομισμάτων.

Τα ερωτήματα που ανακύπτουν κατά την φάση αυτή είναι: (α) πως γνωρίζουμε ότι ο παίκτης ακούμπησε ένα νόμισμα, (β) πως εξαφανίζουμε το νόμισμα όταν το ακουμπήσει ο παίκτης, (γ) πως διαχειριζόμαστε τους πόντους που κερδίζει ο παίκτης. Η επίλυση όλων αυτών των σημείων αποτελεί ιδανική αφετηρία για την εξοικείωση των μαθητών με απλές τεχνικές debugging. Το περιβάλλον της Processing διαθέτει

κονσόλα κειμένου, όπου με χρήση εντολών τύπου println() εμφανίζονται μηνύματα κατά την εκτέλεση του προγράμματος. Ο εκπαιδευτικός καθοδηγεί τους μαθητές στη συγγραφή κώδικα έτσι ώστε κάθε νέα λειτουργία που προστίθεται να επιβεβαιώνεται μέσω debug μηνυμάτων στην κονσόλα κειμένου.

Στο τέλος αυτής της φάσης οι μαθητές έχουν εξοικειωθεί με την χρήση δομών ελέγχου (if) και global μεταβλητών. Επίσης έχουν υλοποιήσει αλληλεπίδραση αντικειμένων όπου η δράση ενός αντικειμένου (ο παίκτης ακουμπάει νόμισμα) προκαλεί την κλήση μεθόδου άλλου αντικειμένου (το νόμισμα εξαφανίζεται), χωρίς ακόμη να αναφερόμαστε σε αλλαγές κατάστασης.

Φάση 5η – Στοιχειώδες AI

Στην φάση αυτή καλούμε τους μαθητές να δημιουργήσουν στατικούς εχθρούς (νέα κλάση και αντικείμενα). Πρόκειται για δραστηριότητα παρόμοια της προηγούμενης φάσης, με την διαφορά ότι αν ο παίκτης ακουμπήσει έναν εχθρό χάνει ζωές. Ας σημειωθεί πως ο κώδικας κάθε κλάσης μπορεί να γραφεί σε διαφορετική καρτέλα στο περιβάλλον της Processing ώστε ο συνολικός κώδικας να είναι πιο ευανάγνωστος και διαχειρίσιμος.

Μόλις οι μαθητές δημιουργήσουν και εμφανίσουν στα διάφορα τετράγωνα της πίστας τους εχθρούς, μπορούμε να εισάγουμε μια πρώτη έννοια τεχνητής νοημοσύνης. Για παράδειγμα, οι εχθροί να μην είναι στατικοί αλλά να κινούνται εναλλάξ δεξιά – αριστερά για πάντα. Για να το επιτύχουν αυτό οι μαθητές θα πρέπει να προσθέσουν κατάλληλες μεθόδους στην κλάση και να τις καλέσουν στο κυρίως πρόγραμμα. Είναι πολύ πιθανό να παρατηρήσουν πως ο κώδικας αυτών των μεθόδων είναι πανομοιότυπος με τον κώδικα των μεθόδων κίνησης του παίκτη. Η παρατήρηση αυτή, μπορεί να αποτελέσει αργότερα την αφετηρία για την εισαγωγή της έννοιας της κληρονομικότητας.

Φάση 6η – Αλλαγή Καταστάσεων Παίκτη

Στα παιχνίδια τύπου Mario μια βασική δυνατότητα του παίκτη είναι το jump το οποίο ενεργοποιείται συνήθως με πάτημα του spacebar. Προγραμματιστικά η υλοποίηση του jump είναι πιο σύνθετη διότι βασίζεται σε αλλαγές κατάστασης του παίκτη. Η κίνηση αναλύεται σε ανεξάρτητες κινήσεις στους δύο άξονες. Χτίζοντας σταδιακά τον κώδικα για το jump, γίνεται φανερό πως αλλάζει η συμπεριφορά ενός αντικειμένου όταν αλλάζει η κατάστασή του.

Φάση 7η – Γενικές Βελτιώσεις

Γενικές βελτιώσεις στον κώδικα του παιχνιδιού μπορεί να είναι η κύλιση της πίστας όταν ο παίκτης φτάνει στο άκρο της, η επανεκκίνηση (reset) του παιχνιδιού όταν ο παίκτης χάσει όλες τις ζωές του, η μέτρηση του χρόνου, η προσθήκη μουσικής. Πρόκειται για βελτιώσεις που δεν σχετίζονται με την αντικειμενοστραφή φύση του προγράμματος και οι οποίες γίνονται πολύ εύκολα με την βοήθεια των βιβλιοθηκών της Processing.

4. Συμπεράσματα

Θεωρώντας ότι ευθύνη των εκπαιδευτικών Πληροφορικής είναι η προετοιμασία των μαθητών για τον πραγματικό κόσμο στον οποίο θα δράσουν, διεξάγαμε επανειλημμένα στο 5ο ΓΕΛ Βύρωνα την Ερευνητική Εργασία «Γραφικά Πραγματικού Χρόνου: δημιουργώντας από το μηδέν» με στόχο την εισαγωγή των μαθητών στον Αντικειμενοστραφή Προγραμματισμό. Η εργασία αυτή μπορεί να χαρακτηριστεί ως διαθεματική, διότι άπτεται κι άλλων γνωστικών αντικειμένων, ιδίως των Μαθηματικών (χρήση γεωμετρικών σχημάτων για την σύνθεση των γραφικών, υπολογισμοί συντεταγμένων) και της Φυσικής (στοιχειώδης ανάλυση κινήσεων). Χρησιμοποιήθηκε η γλώσσα προγραμματισμού Processing, η οποία ακολουθεί τη δομή της Java, αλλά με εξαιρετικά απλοποιημένη σύνταξη κατάλληλη για αρχάριους προγραμματιστές.

Επιχειρήθηκε η δημιουργία ενός περιβάλλοντος αυθεντικής μάθησης, τόσο ως κίνητρο για τη συμμετοχή των μαθητών στην εργασία αλλά και ως επιβεβαίωση της πραγματικής χρησιμότητας της νέας γνώσης. Οι μαθητές ήρθαν σε επαφή με μέλη του HGDA, παρακολούθησαν από κοντά την ανάπτυξη παιχνιδιών και στη συνέχεια σχεδίασαν και υλοποίησαν ένα παιχνίδι ακολουθώντας την ίδια μεθοδολογία. Επιλέξαμε την δημιουργία ενός παιχνιδιού τύπου Mario, διότι (α) ως πρόβλημα είναι αρκετά σύνθετο ώστε να αντανακλά όλες τις βασικές έννοιες του αντικειμενοστραφούς προγραμματισμού και (β) ως κώδικας Processing είναι αρκετά απλός για μαθητές Λυκείου, και αναλύσαμε σε λεπτομερή βήματα τις φάσεις *Ανάπτυξης* και *Ενοποίησης* της μεθοδολογίας.

Επόμενοι στόχοι μας είναι επεξεργασία περαιτέρω διδακτικών προτάσεων που με χρήση της Processing θα οδηγούν στην δημιουργία εφαρμογών για Android καθώς και εφαρμογών επικοινωνίας με Arduino. Ευελπιστούμε οι μαθητές μας να αποκτήσουν τελικά ικανοποιητική σχεδιαστική και προγραμματιστική εμπειρία.

Αναφορές

Bergin, J. (2000). *Why Procedural is the Wrong First Paradigm if OOP is the Goal.* Retrieved August 1, 2014, from PACE University Seidenberg School of Computer Science and Informations Systems: http://csis.pace.edu/~bergin/papers/Whynotproceduralfirst.html

Goldwasser, M. H., & Letscher, D. (2008). Teaching an Object-Oriented CS1 - with Python. *ITiCSE '08 Proceedings of the 13th annual conference on Innovation and technology in computer science education,* (pp. 42-46). NY, USA.

Herrington, J., Oliver, R., & Reeves, T. (2002). *ASCILITE 2002.* Retrieved Αύγουστος 1, 2014, from Patterns of Engagement in Authentic Online Learning Environments: http://www.ascilite.org/conferences/auckland02/proceedings/papers/085.pdf

Kölling, M. (1999). *The design of an object-oriented environment and language for teaching.* Doctoral dissertation, Department of Computer Science, University of Sydney.

Meyseburg, M. (2011). Introductory Programming using Processing. *Midwest Instruction and Computing Symposium.* Duluth, MN.

Moritz, S. H., Wei, F., Parvez, S. M., & Blank, G. D. (2005). From Objects-First to Design-First with Multimedia and Intelligent Tutoring. *ITiCSE '05 Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education*, (pp. 99-103). NY, USA.

Overmars, M. (2004, April). Teaching Computer Science through Game Design. *IEEE Computer* , pp. 81-83.

Reas, C., & Fry, B. (2007). *Processing, a Programming Handbook for Visual Designers and Artists.* MIT Press.

Ryoo, J., Fonseca, F., & Janzen, D. S. (2008). Teaching Object-Oriented Software Engineering through Problem-Based Learning in the Context of Game Design. *IEEE 21st Conference on Software Engineering Education and Training, 2008. CSEET '08.*, (pp. 137-144). Charleston, SC.

Savery, J. R., & Thomas, M. D. (2001). *Problem Based Learning: an instructional model and its constructivist framework.* Indiana University. The Center for Research on Learning and Technology.

Schell, J. (2008). *The Art of Game Design.* Burlington, USA: Morgan Kaufmann.

Varnado, C. (2011, October). *World Building (How to Get Beyond Cool in Video Games).* Retrieved Αύγουστος 1, 2014, from Game Developers Conference Online: http://www.gdcvault.com/play/1015169/%28301%29-Game-Writing

ΕΑΙΤΥ Τομέας Επιμόφωσης και Κατάρτισης. (2008). *Επιμορφωτικό υλικό για την επιμόρφωση των εκπαιδευτικών στα Κέντρα Στήριξης Επιμόρφωσης.* Πάτρα.

Abstract

Teaching Object-Oriented Programming may be difficult to bear fruit, especially if Procedural Programming has been thoroughly taught beforehand. Considerable amount of time is needed to shift from Procedural to Object-Oriented way of thinking, so as not to reject teaching it as an introductory course. The main objective of this paper is to illustrate that the choice of an initial example, quite complicated so that it reflects the spectrum of OOP concepts and able to create an authentic learning context, in combination with an OOP language with quite simple syntax so as to facilitate students, can have effective learning outcomes.

Keywords: Object-Oriented Programming, Processing, Authentic Learning, Real Time Graphics.