Εισάγοντας αρχάριους στον προγραμματισμό με τα περιβάλλοντα Kara: Μια προσέγγιση βασισμένη στη θεωρία υπολογισμού

Α. Μάργαρης[1], Μ. Παπαστεργίου[2]

[1]Τμήμα Πληροφορικής, ΑΤΕΙ Θεσσαλονίκης
amarg@uom.gr
[2]Τ.Ε.Φ.Α.Α., Πανεπιστήμιο Θεσσαλίας
mpapas@uth.gr

Περίληψη

Στην εργασία αυτή αρχικά παρουσιάζονται τα εμπόδια που θέτει στους αρχάριους προγραμματιστές η κλασσική μέθοδος εισαγωγής στον προγραμματισμό μέσω μιας γλώσσας προγραμματισμού γενικού σκοπού. Κατόπιν, παρουσιάζεται η εναλλακτική προσέγγιση των προγραμματιστικών μινι-περιβαλλόντων και τα πλεονεκτήματά της. Στη συνέχεια, παρουσιάζονται τα μινι-περιβάλλοντα Kara, τα οποία βασίζονται στη θεωρία υπολογισμού και αποσκοπούν στην εισαγωγή ατόμων χωρίς προηγούμενη προγραμματιστική εμπειρία σε βασικές αρχές του διαδικαστικού προγραμματισμού. Τέλος, συζητούνται οι δυνατότητες αξιοποίησής τους στο πλαίσιο της σχολικής εκπαίδευσης στην πληροφορική.
Λέξεις κλειδιά: *διδακτική πληροφορικής, προγραμματισμός, μινι-περιβάλλοντα.*

Abstract

In this paper the obstacles that the traditional method of teaching programming through a general purpose programming language poses to beginners as well as the advantages of the alternative programming mini-environments approach are initially presented. Afterwards, the Kara programming mini-environments, which are based on the theory of computation and are aimed at the introduction of beginners into procedural programming are presented. Finally, the possibilities of their didactical exploitation within scholastic education are discussed.
Keywords: *didactics of informatics, programming, mini-environments.*

1. Εισαγωγή

Σε μια κοινωνία που αναθέτει όλο και περισσότερες καθημερινές εργασίες στους ηλεκτρονικούς υπολογιστές και όπου οι τεχνολογίες της πληροφορίας και της επικοινωνίας αναπτύσσονται ραγδαία, η γενική σχολική εκπαίδευση θα πρέπει να προσφέρει στους μαθητές τη δυνατότητα να αποκτούν μια βαθύτερη κατανόηση της τεχνολογίας των υπολογιστών, των διαφόρων εφαρμογών της, καθώς και των περιορισμών της (Reichert, 2003). Κλειδί για την κατανόηση αυτή αποτελεί η εξοικείωση των μαθητών με βασικές αρχές και έννοιες του προγραμματισμού υπολογιστών, οι οποίες αποτελούν εξάλλου τη βάση για τη λογική και την

αφηρημένη συλλογιστική (Brusilovsky et al., 1997). Για το λόγο αυτό αποτελεί κοινή παραδοχή το ότι ο προγραμματισμός θα πρέπει να κατέχει κεντρικό ρόλο στα σχολικά προγράμματα σπουδών Πληροφορικής (Tucker, 1991). Ωστόσο, παρά τα αναμφισβήτητα οφέλη του προγραμματισμού για τους μαθητές, στην πράξη έχει αποδειχθεί ότι η εκμάθησή του είναι μια δύσκολη υπόθεση για αρχάριους όλων των ηλικιών (Deek & McHugh, 1998; Kelleher & Pausch, 2005).

2. Η κλασσική μέθοδος εισαγωγής στον προγραμματισμό

Η πιο διαδεδομένη μέθοδος εισαγωγής στον προγραμματισμό είναι η σταδιακή παρουσίαση των δομών μιας γλώσσας προγραμματισμού γενικού σκοπού και η επίλυση προβλημάτων αυξανόμενης δυσκολίας με τη χρήση αυτών των δομών (Brusilovsky et al., 1997). Ωστόσο, η προσέγγιση αυτή κρίνεται ως αναποτελεσματική, ιδιαίτερα για μαθητές μικρής ηλικίας, καθώς θέτει μια σειρά από εμπόδια στους αρχάριους προγραμματιστές (Brusilovsky et al., 1997; Reichert, 2003): α) απαιτείται από το μαθητή να εξοικειωθεί ταυτόχρονα τόσο με την αυστηρή σύνταξη και τη σημασιολογία της ίδιας της γλώσσας όσο και με τις βασικές αρχές του προγραμματισμού, β) η διδασκαλία και χρήση μιας πλήρους γλώσσας προγραμματισμού στο σχολείο μπορεί να αποβεί πολύ χρονοβόρα, γ) παρέχεται συνήθως περιορισμένη υποστήριξη όσον αφορά στην κατανόηση των βασικών εντολών και δομών ελέγχου της γλώσσας αφού η διαδικασία της εκτέλεσης του προγράμματος παραμένει κρυμμένη από το μαθητή, ενώ η έλλειψη οπτικής ανάδρασης εμποδίζει την κατανόηση της σημασιολογίας της γλώσσας, δ) οι μαθητές δύσκολα μπορούν να εντοπίσουν και να διορθώσουν λάθη στα προγράμματά τους, ε) τα πρώτα προβλήματα που τίθενται στους μαθητές αφορούν κατά κανόνα στην επεξεργασία αριθμών ή συμβόλων και αποτυγχάνουν να κινήσουν το ενδιαφέρον των μαθητών, ενώ η ανάπτυξη πιο ελκυστικών εφαρμογών απαιτεί την εκμάθηση ενός μεγάλου υποσυνόλου της γλώσσας. Ειδικότερα, όσον αφορά στην παρακίνηση των μαθητών, αν η συγγραφή ενός προγράμματος που εμφάνιζε στην οθόνη τη φράση "Hello world" κινούσε το ενδιαφέρον των μαθητών παλαιότερα, δεν συμβαίνει το ίδιο με τη σημερινή 'γενιά του Nintendo' που έλκεται από πολυμεσικά μαθησιακά περιβάλλοντα που θυμίζουν ηλεκτρονικά παιχνίδια (Guzdial & Soloway, 2002).

3. Η μέθοδος των προγραμματιστικών μινι-περιβαλλόντων

Ως εναλλακτική, πιο αποτελεσματική προσέγγιση για την εισαγωγή μαθητών αλλά και φοιτητών στον προγραμματισμό έχουν προταθεί μαθησιακά μινι-περιβάλλοντα (mini-environments) που βασίζονται σε μινι-γλώσσες (mini-languages) και μικρόκοσμους (microworlds) (Brusilovsky et al., 1997). Οι μινι-γλώσσες είναι μικρές γλώσσες προγραμματισμού ειδικά σχεδιασμένες για τη διδασκαλία του προγραμματισμού. Οι μαθητές μαθαίνουν να προγραμματίζουν καθοδηγώντας τις ενέργειες μιας ψηφιακής οντότητας (π.χ. χελώνας, ρομπότ) που ζει σε έναν εικονικό κόσμο (μικρόκοσμο) (Brusilovsky et al., 1994; Kelleher & Pausch, 2005). Η

οντότητα μπορεί να εκτελέσει ένα μικρό σύνολο εντολών και να απαντήσει σε ορισμένες ερωτήσεις που επιστρέφουν τιμές. Συνήθως, ο μαθητής ελέγχει την οντότητα αρχικά δίνοντας μεμονωμένες εντολές και κατόπιν γράφοντας μικρά προγράμματα στη μινι-γλώσσα, που συνήθως περιλαμβάνει όλες τις βασικές δομές ελέγχου (π.χ. εκτέλεση υπό συνθήκη, βρόγχους), καθώς και μηχανισμούς για τη δημιουργία νέων εντολών και υποπρογραμμάτων (Brusilovsky et al., 1997).

Τα πλεονεκτήματα της προσέγγισης αυτής είναι τα ακόλουθα (Brusilovsky et al., 1997; Alessi & Trollip, 2001; Costelloe, 2004; Kelleher & Pausch, 2005):

- Μια μινι-γλώσσα έχει μικρό συντακτικό και απλή σημασιολογία. Επομένως, οι μαθητές μπορούν γρήγορα να τη μάθουν και να τη χρησιμοποιήσουν με ενδιαφέροντα αποτελέσματα, επενδύοντας το χρόνο τους σε σημαντικότερα ζητήματα, όπως η κατανόηση προγραμματιστικών δομών και αρχών, η ανάπτυξη αλγορίθμων και η σχεδίαση προγραμμάτων.
- Το όλο προγραμματιστικό περιβάλλον είναι κτισμένο πάνω σε κάποια οπτικά ελκυστική και παρακινητική για τους μαθητές μεταφορά (metaphor) και επιτρέπει στον εκπαιδευτικό να δημιουργήσει ενδιαφέροντα προβλήματα που σχετίζονται με τις καθημερινές εμπειρίες των μαθητών.
- Οι διάφορες ενέργειες που εκτελεί η οντότητα προκαλούν ορατές αλλαγές στο μικρόκοσμο που αναπαρίσταται στην οθόνη, πράγμα που βοηθά τον αρχάριο προγραμματιστή να αντιληφθεί τι κάνει το πρόγραμμά του και να κατανοήσει τη σημασιολογία των διαφόρων δομών της γλώσσας.
- Τα προβλήματα που συνοδεύουν το περιβάλλον μοιάζουν περισσότερο με σπαζοκεφαλιές παρά με «σοβαρά» προβλήματα και η δραστηριότητα της επίλυσής τους γίνεται ένα είδος παιχνιδιού για τους μαθητές.
- Προάγεται η δημιουργικότητα των μαθητών καθώς και η εποικοδομητική (constructivist) μάθηση μέσα από τον ενεργό πειραματισμό.
- Παρέχεται στους μαθητές η δυνατότητα να οικοδομούν νοητικά μοντέλα και να αναπτύσσουν στρατηγικές επίλυσης προβλημάτων που είναι πιθανό να μεταφερθούν αργότερα σε άλλα πλαίσια.

Από την παραπάνω παρουσίαση καθίσταται προφανές ότι τα προγραμματιστικά μινι-περιβάλλοντα συνιστούν ένα απλό αλλά ισχυρό μέσο για την εισαγωγή των μαθητών στις βασικές αρχές του προγραμματισμού, στην αλγοριθμική σκέψη και στη συστηματική επίλυση προβλημάτων, ενώ παράλληλα παρέχουν τα θεμέλια για τη μετέπειτα εκμάθηση μιας γλώσσας προγραμματισμού γενικού σκοπού.

Η ιδέα των προγραμματιστικών μινι-περιβαλλόντων δεν είναι νέα. Δύο ευρέως διαδεδομένα μινι-περιβάλλοντα είναι η Logo (Papert, 1980) και ο Karel the Robot (Pattis, 1995). Με τη Logo, που αναπτύχθηκε στη δεκαετία του 1970, οι μαθητές μπορούν να διατυπώνουν αλγορίθμους και να προγραμματίζουν μια χελώνα ώστε η κίνησή της να δημιουργεί συγκεκριμένα γεωμετρικά σχήματα. Ένας περιορισμός της Logo (τουλάχιστον στις αρχικές της υλοποιήσεις) είναι ότι η χελώνα είναι «τυφλή»,

δηλαδή δεν μπορεί να λαμβάνει πληροφορίες σχετικά με την τρέχουσα κατάσταση του κόσμου της (Reichert, 2003). Ο περιορισμός αυτός δεν υφίσταται στο περιβάλλον Karel the Robot που αναπτύχθηκε το 1981, δεδομένου ότι αυτό περιλαμβάνει ένα ρομπότ με «αισθητήρες». Το ρομπότ προγραμματίζεται σε μια μινι-γλώσσα που βασίζεται στην Pascal και περιέχει τις βασικές δομές ελέγχου, καθώς και τη δυνατότητα ορισμού διαδικασιών (procedures). Στο περιβάλλον Karel the Robot, ο αρχάριος προγραμματιστής καλείται να επιλύσει προβλήματα με δομημένο τρόπο και σε διακριτά στάδια (π.χ. σχεδίαση της λύσης, υλοποίηση της λύσης, εκσφαλμάτωση). Ωστόσο, κατά τον Reichert (2003), το περιβάλλον παρουσιάζει τα εξής μειονεκτήματα: α) απαιτείται η επιτυχής εκσφαλμάτωση ενός προγράμματος πριν την εκτέλεσή του γεγονός που συνεπάγεται επιπρόσθετες δυσκολίες για έναν αρχάριο, β) η εγγενής πολυπλοκότητα των διαδικασιών και της σειράς εκτέλεσής τους θέτει δυσκολίες στον αρχάριο προγραμματιστή. Θα πρέπει, βέβαια, να σημειωθεί ότι τα μειονεκτήματα αυτά έχουν σε μεγάλο βαθμό ξεπεραστεί σε περιβάλλοντα που αναπτύχθηκαν στη συνέχεια και τα οποία βασίστηκαν στον Karel the Robot (π.χ. Karel Genie, Knobby).

4. Τα περιβάλλοντα Kara

Στην εργασία αυτή παρουσιάζεται μια εναλλακτική προσέγγιση εισαγωγής στον προγραμματισμό η οποία βασίζεται στην ιδέα των μινι-γλωσσών και των προγραμματιστικών μικρόκοσμων και η οποία επινοήθηκε με στόχο να παρακάμψει τα παραπάνω προβλήματα και περιορισμούς. Πρόκειται για τα περιβάλλοντα Kara (http://www.swisseduc.ch/informatik/karatojava/kara/), στα οποία χρησιμοποιείται η θεωρία υπολογισμού ως όχημα για τη διδασκαλία βασικών εννοιών του προγραμματισμού και της Πληροφορικής.

Τα περιβάλλοντα Kara προσφέρουν (Reichert, 2003):

- μια εισαγωγή στον προγραμματισμό βασισμένη στις μηχανές πεπερασμένων καταστάσεων για άτομα χωρίς προηγούμενη προγραμματιστική εμπειρία (*περιβάλλον Kara - Programming with Finite State Machines (Kara-PFSM)*)
- μια εισαγωγή στο προγραμματισμό σε Java για άτομα χωρίς προηγούμενη προγραμματιστική εμπειρία (*περιβάλλον JavaKara*)
- μια εισαγωγή σε βασικές έννοιες ταυτόχρονου προγραμματισμού (concurrent programming) (*περιβάλλον MultiKara*)
- μια εισαγωγή στη θεωρία υπολογισμού βασισμένη στις δισδιάστατες μηχανές Turing (*περιβάλλον TuringKara*)

Από τα τέσσερα παραπάνω περιβάλλοντα, στην εργασία αυτή έμφαση δίνεται στα περιβάλλοντα Kara-PFSM και JavaKara λόγω του ιδιαίτερου ενδιαφέροντος που παρουσιάζουν για τη σχολική εκπαίδευση.

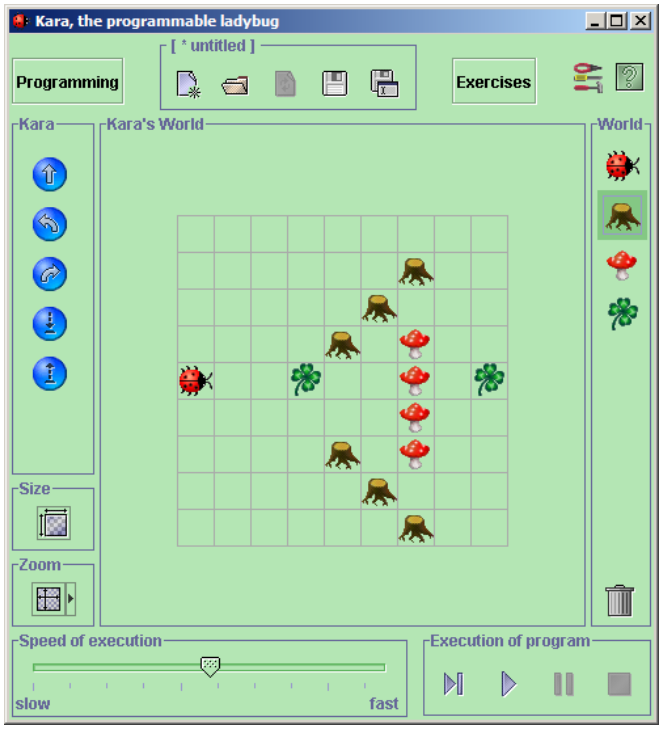
5. Το περιβάλλον Kara-PFSM

Το προγραμματιστικό μοντέλο στο οποίο στηρίζεται το περιβάλλον Kara-PFSM είναι οι μηχανές πεπερασμένων καταστάσεων (finite state machines). Η προσέγγιση αυτή έχει τα εξής βασικά πλεονεκτήματα (Reichert, 2003):

α) Χρήση παραστάσεων από την καθημερινή ζωή: οι μηχανές πεπερασμένων καταστάσεων επιτρέπουν τη διδασκαλία με αναφορές σε αντικείμενα και συσκευές καθημερινής χρήσης (π.χ. φωτεινούς σηματοδότες).
β) Εύκολη δυνατότητα ελέγχου της λειτουργίας κάποιας οντότητας (π.χ. ενός ρομπότ): οι μηχανές πεπερασμένων καταστάσεων αντιδρούν στην τρέχουσα είσοδο με τρόπο που εξαρτάται από την κατάσταση στην οποία βρίσκονται. Το γεγονός αυτό επιτρέπει τον εύκολο έλεγχο της λειτουργίας κάποιας οντότητας με τη βοήθεια μόνον τοπικών πληροφοριών. Αν η οντότητα ζει μέσα σε έναν κόσμο που εξελίσσεται σε συνάρτηση με το χρόνο και μπορεί να επηρεάσει την κατάστασή του, είναι σε θέση να πραγματοποιήσει πάρα πολλές λειτουργίες, γεγονός που μπορεί να αξιοποιηθεί για την υλοποίηση πολύπλοκων διαδικασιών.
γ) Οπτικός προγραμματισμός: οι μηχανές πεπερασμένων καταστάσεων περιγράφονται από μία τυποποιημένη διαγραμματική αναπαράσταση με τις καταστάσεις του συστήματος να αναπαρίστανται από κύκλους, οι οποίοι συνδέονται με προσανατολισμένα ζεύγη που εκφράζουν τις μεταβάσεις που υφίστανται ανάμεσά τους. Αυτές οι δομές μπορούν να κατασκευαστούν και να χρησιμοποιηθούν πάρα πολύ εύκολα δια της χρήσεως εφαρμογών οπτικού προγραμματισμού, χαρακτηριστικό που είναι ιδιαίτερα χρήσιμο σε περιπτώσεις αρχάριων προγραμματιστών, που γενικά δυσκολεύονται να χρησιμοποιήσουν κάποια σύνταξη για να περιγράψουν τεχνικές επίλυσης προβλημάτων.

Ο βασικός πρωταγωνιστής στον εικονικό κόσμο του Kara-PFSM είναι ένα έντομο (μια μικρή πασχαλιά) που κινείται επί ενός δισδιάστατου πλέγματος που έχει τη μορφή σκακιέρας. Εκτός από το έντομο, υπάρχουν άλλες τρεις κατηγορίες αντικειμένων: α) κομμένοι κορμοί δέντρου, οι οποίοι δεν μπορούν να μετακινηθούν (όταν το έντομο βρεθεί μπροστά σε έναν κορμό η κίνησή του σταματά και η εφαρμογή ενημερώνει το χρήστη με κατάλληλο μήνυμα), β) μανιτάρια, τα οποία η πασχαλιά μετακινεί ένα τετράγωνο κάθε φορά προς την κατεύθυνση επί της οποίας κινείται, και γ) φύλλα δέντρου, τα οποία το έντομο αποθέτει στο έδαφος, ή εναλλακτικά, τα απομακρύνει από αυτό. Οι κινήσεις που μπορεί να επιτελέσει το έντομο είναι: α) να προχωρήσει ένα τετράγωνο προς τα εμπρός κατά μήκος της τρέχουσας διεύθυνσης (δηλ. η διαγώνια κίνηση δεν είναι επιτρεπτή), β) να στρίψει δεξιά κατά 90^o επί του τετραγώνου στο οποίο βρίσκεται, γ) να στρίψει αριστερά κατά 90^o επί του τετραγώνου στο οποίο βρίσκεται, δ) να σπρώξει προς τα εμπρός κατά ένα τετράγωνο ένα μανιτάρι εφόσον αυτό βρεθεί στην πορεία του, ε) να ρίξει ένα φύλλο στο έδαφος ή να μαζέψει ένα φύλλο από το έδαφος επί του τετραγώνου στο οποίο βρίσκεται. Χρησιμοποιώντας τα κατάλληλα πλαίσια ελέγχου, ο μαθητής έχει τη

δυνατότητα με τεχνικές drag and drop να τοποθετήσει στο πλέγμα όσους κορμούς, μανιτάρια και φύλλα επιθυμεί, με εξαίρεση το ίδιο το έντομο για το οποίο υπάρχει μόνον ένα στιγμιότυπο. Τυπικό παράδειγμα οθόνης του περιβάλλοντος Kara-PFSM παρουσιάζεται στο Σχήμα 1.



Σχήμα 1: *Η κεντρική οθόνη του περιβάλλοντος Kara-PFSM*

Η κίνηση του εντόμου εντός του πλέγματος ελέγχεται με τη βοήθεια μιας μηχανής πεπερασμένων καταστάσεων. Στο παράδειγμα που παρουσιάζεται στο Σχήμα 2, η μηχανή καθοδηγεί το έντομο έτσι ώστε να μπορέσει να μετακινηθεί μέσα σε έναν κλειστό χώρο που περιβάλλεται από κομμένους κορμούς δέντρων. Για κάθε μετάβαση (transition) που πραγματοποιείται στη μηχανή πεπερασμένων καταστάσεων, μία λογική έκφραση, οι μεταβλητές της οποίας αρχικοποιούνται με τη βοήθεια των πέντε αισθητήρων του εντόμου και η οποία αποτιμάται ως αληθής ή ψευδής, ορίζει τις συνθήκες κάτω από τις οποίες λαμβάνει χώρα η συγκεκριμένη μετάβαση. Σημειώνεται ότι οι πέντε αισθητήρες του εντόμου επιστρέφουν αληθή ή ψευδή τιμή για καθένα από τα εξής ερωτήματα: α) υπάρχει δέντρο στο μπροστινό τετράγωνο, β) υπάρχει δέντρο στο αριστερό τετράγωνο, γ) υπάρχει δέντρο στο δεξί τετράγωνο, δ) υπάρχει μανιτάρι στο μπροστινό τετράγωνο, ε) υπάρχει φύλλο στο τρέχον τετράγωνο. Θεωρώντας για παράδειγμα την κατάσταση ‘walk’, όπως φαίνεται στο Σχήμα 2, αν το έντομο διαπιστώσει πως υπάρχει δέντρο μπροστά του και δέντρο

στα αριστερά του αλλά όχι δέντρο στα δεξιά του, θα επιλεγεί η τρίτη μετάβαση της μηχανής, γεγονός που θα οδηγήσει το έντομο στο να στρίψει δεξιά και να συνεχίσει να κινείται σε ευθεία γραμμή.

Σχήμα 2: *Έλεγχος της κίνησης του εντόμου στο περιβάλλον Kara-PFSM*

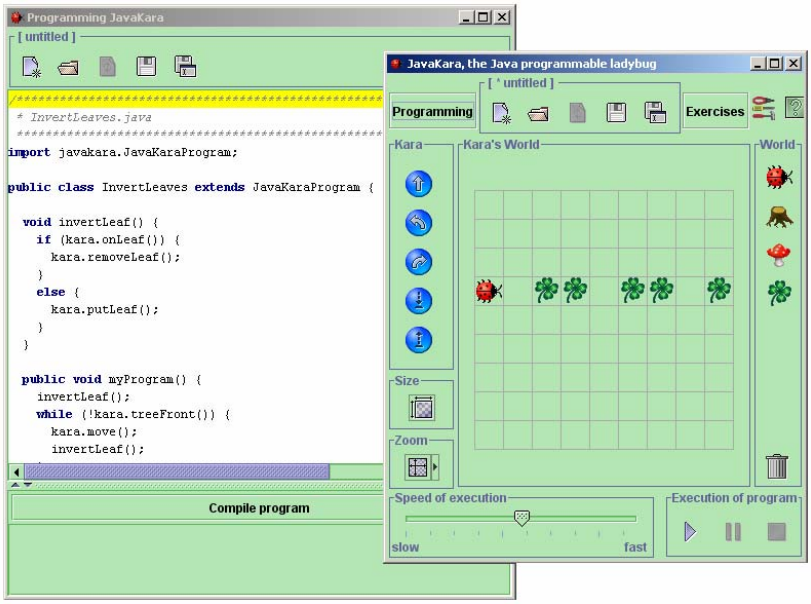
6. Το περιβάλλον JavaKara

Το περιβάλλον JavaKara (http://www.swisseduc.ch/informatik/karatojava/javakara/) επιτρέπει την ομαλή μετάβαση από την προσέγγιση των πεπερασμένων μηχανών στην προσέγγιση του διαδικαστικού προγραμματισμού χρησιμοποιώντας ως εργαλείο για αυτό το σκοπό τη γλώσσα προγραμματισμού Java. Ο κόσμος του JavaKara είναι ο ίδιος με τον κόσμο του Kara-PFSM με τη διαφορά ότι ο finite state machine editor έχει αντικατασταθεί από έναν text editor που επιτρέπει τη συγγραφή και τη μεταγλώττιση κώδικα που καθορίζει την κίνηση του εντόμου επί του τετραγωνικού πλαισίου. Στο στάδιο αυτό, υποτίθεται ότι ο μαθητής έχει ήδη εξοικειωθεί με το περιβάλλον της εφαρμογής έτσι ώστε να μπορεί άμεσα να αναπτύξει τις δεξιότητες επίλυσης προβλημάτων που του ανατίθενται από τον εκπαιδευτικό.

Χρησιμοποιώντας το περιβάλλον JavaKara, ο μαθητής έρχεται σε επαφή με τις κλασσικές δομές του παραδοσιακού διαδικαστικού προγραμματισμού που περιλαμβάνουν, μεταξύ άλλων, τη διαδικασία αποσύνθεσης ενός προβλήματος σε μεθόδους, τη σειρά εκτέλεσης των συναρτήσεων ενός προγράμματος και τις κλασσικές δομές ελέγχου και επανάληψης. Το περιβάλλον μπορεί επίσης να χρησιμοποιηθεί για μια εισαγωγή στον αντικειμενοστραφή προγραμματισμό.

Στο Σχήμα 3 παρουσιάζεται ένα παράδειγμα προγραμματιστικής άσκησης με το JavaKara. Η άσκηση αυτή, καθώς και άλλες υποδειγματικά λυμένες ασκήσεις (π.χ. υλοποίηση του αλγορίθμου ταξινόμησης bubblesort), συνοδεύουν τη διανομή της εφαρμογής. Στο Σχήμα 3, το έντομο κινείται κατά μήκος μιας γραμμής και αντιστρέφει το περιεχόμενο του κάθε τετραγώνου, τοποθετώντας ένα φύλλο σε κάθε άδειο τετράγωνο και απομακρύνοντας ένα φύλλο από κάθε τετράγωνο που περιέχει φύλλο. Εάν κάνουμε την αντιστοίχηση «άδειο τετράγωνο»→0 και «τετράγωνο με

φύλλο»→1, το παραπάνω απλό πρόγραμμα μπορεί να βοηθήσει τους μαθητές να κατανοήσουν την έννοια του συμπληρώματος ως προς 1 (1's complement) ενός δυαδικού αριθμού, το οποίο προκύπτει από τον αριθμό με αντιστροφή των ψηφίων του. Εάν δε αφήσουμε το έντομο να περάσει για δεύτερη φορά πάνω από την εν λόγω γραμμή, θα λάβει χώρα προφανώς επαναφορά της αρχικής γραμμής. Με τον τρόπο αυτό, οι μαθητές θα διαπιστώσουν στην πράξη ότι το συμπλήρωμα του συμπληρώματος ενός δυαδικού αριθμού είναι ο ίδιος ο αριθμός.



Σχήμα 3: *Οικοδόμηση της έννοιας του συμπληρώματος ως προς 1*

Προκειμένου ο χρήστης να διευκολυνθεί στην ανάπτυξη του κώδικα του προγράμματός του, το περιβάλλον JavaKara προσφέρει ένα προσχέδιο έτοιμου κώδικα (template), το οποίο ο χρήστης θα πρέπει να τροποποιήσει αναλόγως για να υλοποιήσει τη διαδικασία που επιθυμεί. Σε μια πιο τεχνική περιγραφή, η κλάση του προγράμματος που θα ελέγχει την κίνηση του εντόμου θα πρέπει υποχρεωτικά να είναι παράγωγη κλάση της κλάσης JavaKaraProgram, η οποία επιτρέπει στο χρήστη να προσπελάσει άμεσα τις συναρτήσεις των αντικειμένων της εφαρμογής. Στο περιβάλλον JavaKara υπάρχουν τρία θεμελιώδη αντικείμενα που μπορούν να χρησιμοποιηθούν από το χρήστη: α) το αντικείμενο kara που επιτρέπει τον έλεγχο της κίνησης του εντόμου, β) το αντικείμενο world που επιτρέπει τον έλεγχο του κόσμου μέσα στον οποίον αυτό ζει και γ) το αντικείμενο tools που προσφέρει στο χρήστη χρήσιμες και βοηθητικές λειτουργίες, όπως η εισαγωγή και η εμφάνιση κειμένου. Ενδεικτικές μέθοδοι του αντικειμένου kara δίνονται στον Πίνακα 1.

Πίνακας 1: *Ενδεικτικές μέθοδοι του αντικειμένου kara*

Μέθοδος	Λειτουργία
void move()	Μετακινεί το έντομο ένα βήμα προς τα εμπρός
void turnLeft()	Στρέφει το έντομο κατά 90^o προς τα αριστερά
void putLeaf()	Εξαναγκάζει το έντομο να πετάξει ένα φύλλο
boolean treeFront()	Επιστρέφει true αν μπροστά από το έντομο υπάρχει κορμός

7. Δυνατότητες εφαρμογής στην εκπαίδευση στην πληροφορική

Το ολοκληρωτικά γραφικό περιβάλλον Kara-PFSM, που όπως προαναφέρθηκε δεν απαιτεί τη χρήση κάποιας γλώσσας προγραμματισμού ούτε την πληκτρολόγηση κώδικα από μέρους του χρήστη, θα μπορούσε να χρησιμοποιηθεί για μια ευχάριστη και παιγνιώδη εισαγωγή στον προγραμματισμό στις τελευταίες τάξεις του Δημοτικού σχολείου και στο Γυμνάσιο. Το JavaKara θα μπορούσε να αξιοποιηθεί στο Λύκειο για την εισαγωγή των μαθητών στον αλγοριθμικό τρόπο σκέψης μέσα από δραστηριότητες επίλυσης προβλημάτων με όχημα μια ευρέως χρησιμοποιούμενη γλώσσα προγραμματισμού, την Java, την οποία μαθητές της βαθμίδας αυτής μπορούν να οικειοποιηθούν αν προσεγγίσουν μέσα από κατάλληλες εκπαιδευτικές δραστηριότητες (π.χ. βλ. Bennett et al., 2006).

Τα περιβάλλοντα Kara έχουν χρησιμοποιηθεί με επιτυχία σε μαθήματα Πληροφορικής σε διάφορα επίπεδα της εκπαίδευσης κυρίως στη Γερμανία, Αυστρία και Ελβετία. Για παράδειγμα, σε επίπεδο ανώτατης εκπαίδευσης, τα MultiKara και TuringKara έχουν αξιοποιηθεί σε μαθήματα θεωρητικής Πληροφορικής σε Τμήματα Πληροφορικής, ενώ τα Kara-PFSM και JavaKara έχουν χρησιμοποιηθεί σε παιδαγωγικά τμήματα με στόχο την εισαγωγή μελλοντικών εκπαιδευτικών σε βασικές προγραμματιστικές αρχές και στη διδασκαλία τους. Όσον αφορά στη δευτεροβάθμια εκπαίδευση, έρευνες (Reichert, 2003) που έγιναν σε μαθητές Γυμνασίου που είχαν χρησιμοποίησαν το περιβάλλον Kara-PFSM στα σχολεία τους, και οι οποίοι είχαν μηδενική ή περιορισμένη προηγούμενη εμπειρία στον προγραμματισμό, έδειξαν ότι: α) οι αντιδράσεις των μαθητών ήταν πολύ θετικές, β) το περιβάλλον κέντρισε το ενδιαφέρον των μαθητών και έδρασε παρακινητικά, ιδιαίτερα για τους μαθητές χωρίς προηγούμενη προγραμματιστική εμπειρία, γ) οι μαθητές βρήκαν τη διεπαφή της εφαρμογής πολύ απλή και εύχρηστη, δ) οι μαθητές εκτίμησαν ιδιαίτερα το ότι μετά από μια πολύ σύντομη φάση αρχικής εξοικείωσης με το περιβάλλον άρχισαν πολύ γρήγορα να επιλύουν ενδιαφέροντα προβλήματα, ε) το περιβάλλον επέτρεψε στους μαθητές να εστιάσουν στην επίλυση προβλημάτων και στον έλεγχο της ορθότητας των προγραμμάτων τους, χωρίς να αναλώνονται στη συγγραφή κώδικα σε κάποια γλώσσα προγραμματισμού. Επίσης, έρευνα σε εκπαιδευτικούς που είχαν χρησιμοποίησαν το JavaKara στα σχολεία τους (Reichert, 2003) έδειξε ότι το περιβάλλον ενδείκνυται για τη διδασκαλία Java σε αρχάριους.

Τα παραπάνω ευρήματα καταδεικνύουν ότι το εννοιολογικά απλό προγραμματιστικό μοντέλο των μηχανών πεπερασμένων καταστάσεων αποτελεί ένα αποτελεσματικό μέσο για την εισαγωγή αρχάριων στον προγραμματισμό. Επίσης, η απλή και διαισθητική διεπαφή χρήστη των περιβαλλόντων Kara ελαχιστοποιεί το χρόνο εκμάθησης της χρήσης των περιβαλλόντων αυτών, πράγμα που έχει ιδιαίτερη σημασία για τη σχολική εκπαίδευση όπου οι ώρες διδασκαλίας της Πληροφορικής είναι συνήθως περιορισμένες.

Τα περιβάλλοντα Kara αναπτύχθηκαν από το Τμήμα Πληροφορικής του Πολυτεχνείου της Ζυρίχης και διατίθενται ελεύθερα. Οι εκπαιδευτικοί που ενδιαφέρονται να τα χρησιμοποιήσουν στα μαθήματά τους μπορούν να κατεβάσουν τη σχετική διανομή από τη διεύθυνση: http://www.swisseduc.ch/informatik/karatojava/download.html.

Βιβλιογραφία

Alessi, S., & Trollip, S. (2001). *Multimedia for learning: Methods and. development*. Boston, MA: Allyn & Bacon.

Bennett, A., Briggs, J., & Clark, M. (2006). High school computing clubs: A pilot study. *ACM SIGCSE Bulletin*, 38(3), 38-42.

Brusilovsky, P., Calabrese, E., Hvorecky, J., Kouchnirenko, A., & Miller, P. (1997). Mini-languages: A way to learn programming principles. *Education and Information Technologies*, 2(1), 65-83.

Brusilovsky, P., Kouchnirenko, A., Miller, P., & Tomek, I. (1994). Teaching programming to novices: A review of approaches and tools. In *Proceedings of ED-MEDIA'94* (pp. 103–110).

Costelloe, E. (2004). Teaching programming: The state of the art. Center for Research in IT in Education (CRITE) Technical Report, Dublin.

Deek, F., & McHugh J. (1998). A survey and critical analysis of tools for learning programming. *Computer Science Education*, 8(2), 130-178.

Guzdial, M., & Soloway, E. (2002). Teaching the Nintendo generation to program. *Communications of the ACM*, 45(4), 17–21.

Kelleher, C., & Pausch, R. (2005). Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Computing Surveys*, 37(2), 83-137.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. New York, NY: Basic Books.

Pattis, R. (1995). *Karel the Robot: A gentle introduction to the art of programming*. New York, NY: Wiley.

Reichert, R. (2003). Theory of computation as a vehicle for teaching fundamental concepts of computer science. Dissertation No. 15035, ETH Zürich.

Tucker, A. (1991). Computing curricula 1991. *Communications of the ACM*, 34(6), 68–84.