ΑΝΩΤΑΤΟ ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΠΕΙΡΑΙΑ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Ο ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΗΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΜΕΣΩ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΟΥ ΠΕΡΙΒΑΛΛΟΝΤΟΣ "ALICE" ΚΑΙ ΑΝΑΠΤΥΞΗ ΔΙΔΑΚΤΙΚΟΥ ΥΛΙΚΟΥ

"ALICE IN JAVALAND"

ΑΓΓΕΛΙΚΗ ΣΙΔΕΡΗ (ΑΜ:37254) – ΒΑΣΙΛΕΙΑ ΦΤΩΧΟΓΙΑΝΝΗ (ΑΜ:36875)

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: ΨΑΡΟΜΗΛΙΓΚΟΣ ΙΩΑΝΝΗΣ

ΑΘΗΝΑ, 2014

Περιεχόμενα

Πρόλογος

Η εργασία αυτή αποτελεί μια προσπάθεια για την ανάπτυξη εκπαιδευτικού υλικού για την κατανόηση βασικών εννοιών του αντικειμενοστραφή προγραμματισμού μέσω του προγραμματιστικού περιβάλλοντος "ALICE".

Ο στόχος της πτυχιακής εργασίας ήταν **η δημιουργία και η παρουσίαση μιας σειράς σεναρίων για την κατανόηση των θεμελιωδών εννοιών και βασικών δομών του αντικειμενοστραφή προγραμματισμού**, χρησιμοποιώντας ως βασικό εργαλείο το ALICE (πχ σενάριο που μέσα από τις κινήσεις των ηρώων γίνεται η κατανόηση της έννοιας του πολυμορφισμού).

Το ALICE είναι ένα προγραμματιστικό περιβάλλον σχεδιασμένο για να αποτελέσει την πρώτη επαφή ενός εκπαιδευόμενου με τον αντικειμενοστραφή προγραμματισμό. Το ALICE στοχεύει, μέσα από την εύχρηστη διεπαφή και τα ελκυστικά τρισδιάστατα γραφικά του, να προσελκύσει το ενδιαφέρον των μαθητών για τον προγραμματισμό.

Το πρόγραμμα "ALICE" έχει δημιουργηθεί από μια ομάδα ερευνητών του πανεπιστήμιου Carnegie Mellon το 1999 μεταξύ των οποίων είναι και οι Wanda Dann, Stephen Cooper, Randy Pausch.

Πιο συγκεκριμένα στο *πρώτο μέρος* παρουσιάζεται **η διδασκαλία του προγραμματισμού** και τι μπορεί να προσκομίσει κάποιος που θα ασχοληθεί, καθώς και εκπαιδευτικά εργαλεία και προγραμματιστικά περιβάλλοντα για τη διδακτική του αντικειμενοστραφούς προγραμματισμού.

Στο *δεύτερο μέρος* παρουσιάζεται **ο αντικειμενοστραφής προγραμματισμός και το προγραμματιστικό περιβάλλον ALICE**. Αναφέρεται ποιές από τις θεμελιώδεις έννοιες του αντικειμενοστρεφούς προγραμματισμού μπορούν να διδαχτούν με το ALICE. Επίσης περιλαμβάνει έναν οδηγό χρήσης (tutorial) για το ALICE, με οδηγίες εγκατάστασης του λογισμικού, βοήθεια σε περιπτώσεις προβλημάτων και τη διαδικασία δημιουργίας μιας ολοκληρωμένης εφαρμογής. Σε αυτή τη διαδικασία ο χρήστης ακολουθεί μια σειρά βημάτων και δημιουργεί μια ολοκληρωμένη εφαρμογή χρησιμοποιώντας τις βασικότερες δομές που υποστηρίζει το περιβάλλον του ALICE.

Στο *τρίτο μέρος* παρουσιάζεται **η ανάπτυξη εκπαιδευτικών σεναρίων** που είναι σχεδιασμένα για τη διδασκαλία και την εκμάθηση βασικών αρχών του προγραμματισμού, που περιλαμβάνονται στο ALICE. Πιο συγκεκριμένα τα μαθήματα είναι for, if, polymorphism, private-public, and creating classes/inheritance. Οι προγραμματιστικές έννοιες διδάσκονται μέσα από τη διαδικασία δημιουργίας ενός "κόσμου" (εφαρμογή). Με την χρήση της ιστορίας η Αλίκη στη χώρα των θαυμάτων και την μεταφορά της στο δικό μας ALICE IN JAVALAND γίνεται η προσπάθεια για την κατανόηση βασικών εννοιών και αρχών προγραμματισμού. Μέσα από περιπέτειες μαθαίνει τη σημασία σημαντικών εννοιών του αντικειμενοστραφούς

προγραμματισμού. Το σχέδιο μαθήματος παραθέτει τους εκπαιδευτικούς στόχους που επιδιώκονται να επιτευχθούν κατά την ολοκλήρωση της διδασκαλίας και καθοδηγεί τον καθηγητή. Τα παραδείγματα είναι από απλές κινήσεις χαρακτήρων σ' ένα κόσμο όπου ο κάθε χαρακτήρας αλληλοεπιδρά με τον χρήστη. Ο χρήστης έχει τη δυνατότητα με το πάτημα ενός κουμπιού, με την εισαγωγή χαρακτήρων ή με το κλικ του ποντικιού να προκαλέσει την κίνηση ενός χαρακτήρα ή ενός αντικειμένου.

Στο *τέταρτο μέρος* γίνεται μια σύνοψη της παρούσας διπλωματικής εργασίας και παρατίθενται ορισμένα **συμπεράσματα** που προέκυψαν από την μελέτη του προγραμματιστικού εκπαιδευτικού υλικού ALICE.

Τέλος επιδιώκεται με τα μαθήματα οι μαθητές να μάθουν να δημιουργούν μεθόδους και συναρτήσεις, να χρησιμοποιούν λίστες και πίνακες, να κατανοούν τη Boolean λογική, να κάνουν σωστή χρήση των τοπικών και καθολικών μεταβλητών και να μάθουν τις δομές επιλογής και επανάληψης. Επίσης είναι πολύ βασικό ότι μαθαίνουν να προγραμματίζουν με αντικείμενα και διδάσκονται τις έννοιες των κλάσεων και της κληρονομικότητας που αποτελούν βασικές έννοιες του αντικειμενοστραφή προγραμματισμού.

Εισαγωγή

Ο προγραμματισμός είναι ένα αναπόσπαστο κομμάτι στο πρόγραμμα σπουδών της πληροφορικής σε όλες τις βαθμίδες εκπαίδευσης στην Ελλάδα και στο εξωτερικό. Η μεγάλη σημασία εκμάθησης του προγραμματισμού και επίλυσης προβλημάτων σε αυτό το πλαίσιο έχει προβληματίσει αρκετές φορές. Η **γνώση προγραμματισμού** μπορεί να αποτελέσει εκπαιδευτικό εργαλείο για την καλλιέργεια και ανάπτυξη νοητικών δεξιοτήτων και να δώσει κίνητρα για ένα δομημένο τρόπο σκέψης και αντιμετώπισης προβλημάτων σε ποικίλα γνωστικά αντικείμενα.

Παρόλα αυτά όμως, οι δυσκολίες στην παραδοσιακή διδακτική προσέγγιση που εφαρμόζεται κατά την διδασκαλία δεν βοηθούν στην ανάπτυξη θετικής στάσης ως προς τα μαθήματα του προγραμματισμού.

Για την αντιμετώπιση διαφαίνονται **δύο κύριες τάσεις** η πρώτη αφορά την ανάπτυξη ειδικών προγραμματιστικών περιβαλλόντων και εργαλείων και η δεύτερη σχεδίαση διδακτικών προσεγγίσεων.

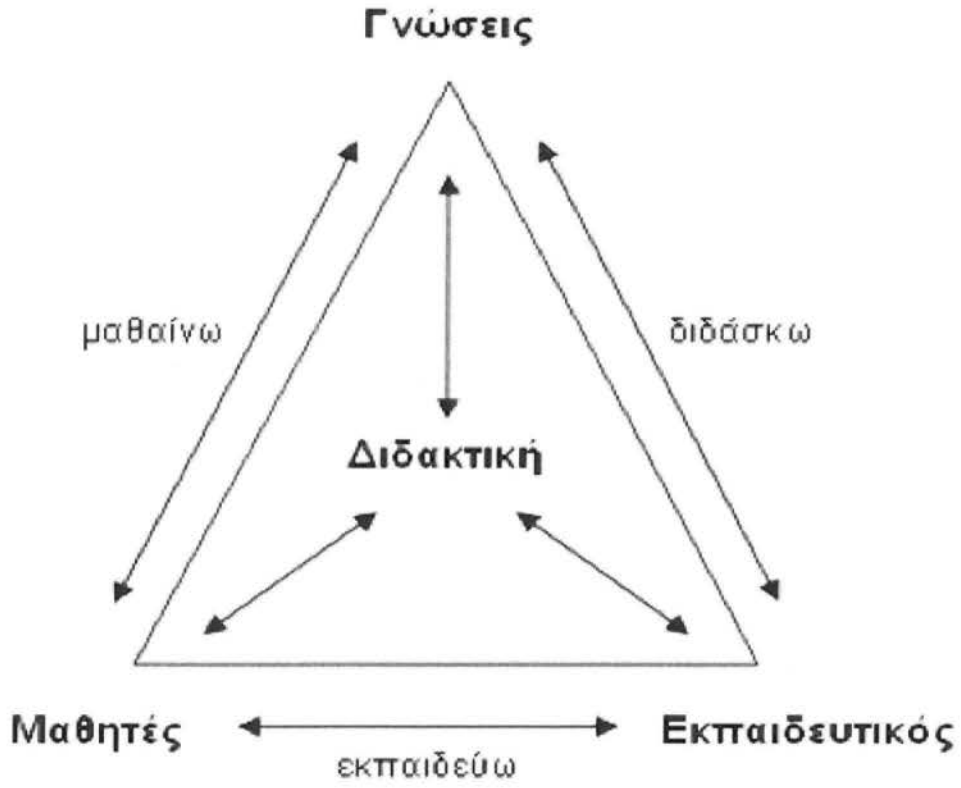
ΜΕΡΟΣ 1ο

Προγραμματιστικά Περιβάλλοντα και η Διδασκαλία του Προγραμματισμού

1.1 - Διδακτική της Πληροφορικής

Η διδακτική μελετά τις διαδικασίες μετάδοσης και πρόσκτησης των γνώσεων με απώτερο στόχο τη βελτίωση αυτών των διαδικασιών. Μελετά τις συνθήκες μέσα στις οποίες τα υποκείμενα μαθαίνουν, εστιάζοντας την προσοχή της στα ιδιαίτερα προβλήματα που ανακινούν τόσο το περιεχόμενο των γνώσεων όσο και των δεξιοτήτων που πρέπει να αποκτηθούν. Άρα ενδιαφέρεται για τους τρόπους με τους οποίους ευνοείται η οικοδόμηση των γνώσεων στο πλαίσιο καταστάσεων διδασκαλίας (ατομικών ή συλλογικών).

Η διδακτική σύμφωνα με τον **Houssaye** αναπαρίσταται από ένα τρίγωνο το οποίο συνδέει τις γνώσεις , το μαθητή και τον εκπαιδευτικό.



Τομείς που συνδέουν τις κορυφές :

- ο τομέας ανάπτυξης των περιεχομένων (διδακτικός μετασχηματισμός, κοινωνικές πρακτικές αναφοράς κτλ.)
- ο τομέας των στρατηγικών της οικοδόμησης και της μάθησης (αναπαραστάσεις, διδακτικά εμπόδια, επίλυση προβλήματος κτλ.)
- ο τομέας της οικοδόμησης διδακτικών καταστάσεων (διδακτικό συμβόλαιο, κατάσταση – πρόβλημα κτλ.)
- ο τομέας των αλληλεπιδράσεων (διδακτική βοήθεια κτλ.) και των χρησιμοποιούμενων μέσων που παίζουν διαμεσολαβητικό ρόλο στις αλληλεπιδράσεις (η ύπαρξη υπολογιστή δρα καταλυτικά στην διδασκαλία – η αλληλεπίδραση του μαθητή με το τεχνολογικό μέσο).

1.2 - Τα πλεονεκτήματα του μαθήματος

i. Αυστηρότητα στη σκέψη, ακρίβεια στην έκφραση, συνειδητή ανάγκη για αποσαφήνιση των ενεργειών.
ii. Πρόσκτηση και κατανόηση γενικών εννοιών, όπως διαδικασία, μεταβλητή, συνάρτηση, μετασχηματισμός (που σχετίζονται άμεσα και με τη μαθηματική παιδεία).
iii. Πρόσκτηση ευρετικών ικανοτήτων και μεθοδολογίας: σχεδιασμός, αναζήτηση παρόμοιων περιπτώσεων, επίλυση με ανάλυση σε μέρη.
iv. Μάθηση τεχνικών αναζήτησης λαθών που μπορούν να μεταφερθούν και σε άλλους, εκτός προγραμματισμού, χώρους.
v. Πρόσκτηση της γενικής ιδέας οικοδόμησης της λύσης με τη μορφή μικρών διαδικασιών ή στοιχειωδών τμημάτων, τα οποία μπορούν να συνδεθούν και να χρησιμοποιηθούν για την οικοδόμηση της λύσης σύνθετων προβλημάτων.
vi. Επέκταση της συνειδητοποίησης και της γνώσης πάνω σε τεχνικές επίλυσης προβλημάτων.
vii. Επέκταση και ανάπτυξη της χρήσης συγκριτικών μεθόδων που αφορούν στην πολλαπλότητα των τρόπων ώστε να επιτευχθεί ένας δεδομένος στόχος.

1.3 - Βασικές γνώσεις και δεξιότητες

Γνώσεις των προγραμματιστικών εννοιών και δομών: Προκειμένου οι μαθητές να μπορούν να αναπτύσσουν ένα πρόγραμμα, θα πρέπει να γνωρίζουν τις έννοιες και τις δομές του προγραμματισμού. Η γνώση, όμως, αυτή δεν επαρκεί για την αποτελεσματική σύνθεση και χρησιμοποίηση των δομών στο πλαίσιο της ανάπτυξης ενός προγράμματος.

Ικανότητες στη σχεδίαση: αφορούν κυρίως ικανότητες που πρέπει να έχει ένας μαθητής ώστε να μπορεί να αξιοποιεί και να συνδυάζει αποτελεσματικά τις προγραμματιστικές δομές για την ανάπτυξη ενός προγράμματος. Οι ικανότητες αυτές σχετίζονται κυρίως με:

i. την ανάλυση ενός προβλήματος σε επιμέρους προβλήματα και τη σύνθεση των επιμέρους τμημάτων για την ολοκληρωμένη λύση του προβλήματος
ii. την υλοποίηση και τη δοκιμή της λύσης τμηματικά
iii. την επαναχρησιμοποίηση γνωστών λύσεων
iv. την επέκταση/ τροποποίηση υπαρχουσών λύσεων

Προκειμένου οι μαθητές να αναπτύξουν τέτοιες ικανότητες, θα πρέπει, μεταξύ άλλων, να γνωρίζουν **τεχνικές επίλυσης συγκεκριμένων προβλημάτων** (π.χ., εύρεση ελάχιστου αριθμού από μια ακολουθία αριθμών, ταξινόμηση μίας ακολουθίας αριθμών) και βασικούς κανόνες που αφορούν την ονοματολογία των μεταβλητών, τη χρησιμοποίηση των επαναληπτικών δομών κτλ.

Ικανότητες στην επίλυση προβλημάτων: Οι ικανότητες στην επίλυση προβλημάτων έχουν άμεση σχέση με τις ικανότητες στη σχεδίαση και αφορούν την επίλυση προβλημάτων με ή χωρίς χρήση υπολογιστή και τη χρησιμοποίηση διαφόρων γλωσσών προγραμματισμού και υπολογιστικών εργαλείων για την επίλυση προβλημάτων. Η απόκτηση τέτοιων ικανοτήτων απαιτεί ενασχόληση με ποικιλία προβλημάτων διαφορετικού βαθμού δυσκολίας και εμπειρία στη χρησιμοποίηση διαφόρων εργαλείων και στην ανάπτυξη προγραμμάτων.

Βασικό στόχο των εισαγωγικών μαθημάτων προγραμματισμού αποτελεί η κατανόηση της λειτουργίας των βασικών προγραμματιστικών εννοιών/δομών και η απόκτηση γνώσεων και δεξιοτήτων στη σχεδίαση και στην υλοποίηση λύσεων απλών προβλημάτων σε ένα συγκεκριμένο προγραμματιστικό περιβάλλον. Η διδασκαλία των προγραμματιστικών εννοιών/δομών και οι εργασίες που πραγματοποιούν οι μαθητές πραγματοποιούνται χρησιμοποιώντας μία συγκεκριμένη γλώσσα προγραμματισμού.

1.4 - Ταξινόμηση διδακτικών στόχων για τον προγραμματισμό

Σε επίπεδο ωριαίας διδασκαλίας, οι διδακτικοί στόχοι αφορούν τα προσδοκώμενα μαθησιακά αποτελέσματα που αναμένεται να επιτευχθούν μετά το πέρας της διδασκαλίας. Οι διδακτικοί στόχοι θα πρέπει να καθορίζονται σε αντιστοιχία με ακρίβεια και σαφήνεια και θα πρέπει να ανταποκρίνονται στις πραγματικές ανάγκες των μαθητών. Έχουν αναπτυχθεί διάφορες ταξινομήσεις για τον καθορισμό διδακτικών (μαθησιακών) στόχων και για την αξιολόγηση των μαθητών. Η πιο γνωστή και ευρέως χρησιμοποιημένη ταξινομία είναι **του Bloom**, ο οποίος, μαζί με τους συνεργάτες του, όρισε δύο ταξινομίες, τη γνωστική, που αναφέρεται στη μνήμη, στη γνώση, στην αντίληψη και στη σκέψη, και τη συναισθηματική, που αναφέρεται στις στάσεις, στις αξίες και στις διαθέσεις. Η γνωστική ταξινομία ορίζει έξι επίπεδα μάθησης:

i. Γνώση,
ii. Κατανόηση,
iii. Εφαρμογή,
iv. Ανάλυση,
v. Σύνθεση και
vi. Αξιολόγηση.

1.5 - Δυσκολίες στην εκμάθηση του προγραμματισμού

Ο Du Boulay διαχωρίζει τις μαθησιακές δυσκολίες, που αντιμετωπίζουν οι μαθητές κατά την εκμάθηση του προγραμματισμού, σε πέντε κατηγορίες, όπου σε κάποιο βαθμό επικαλύπτονται:

Η πρώτη κατηγορία καλείται **Προσανατολισμός (Orientation)** και σχετίζεται με τα ερωτήματα: «τι είναι ο προγραμματισμός;» και «σε τι μας είναι χρήσιμος;». Οι μαθητές δυσκολεύονται να κατανοήσουν τι είναι ο προγραμματισμός και ποια είναι τα οφέλη που αποκομίζουν από την απόκτηση δεξιοτήτων στην επίλυση προβλημάτων μέσω του υπολογιστή (πώς δηλαδή μπορούν να αξιοποιήσουν τις δεξιότητες που αποκτούν) καθώς και τα είδη προβλημάτων που μπορούν να επιλυθούν μέσω του υπολογιστή.

Η δεύτερη κατηγορία μαθησιακών δυσκολιών ονομάζεται **Νοητή Μηχανή (Notional Machine)** και επικεντρώνεται στο ερώτημα: «πώς λειτουργεί ο υπολογιστής;». Ο μαθητής καλείται να «ελέγξει» μια νοητή μηχανή. «Τι μορφή παίρνει η εικονική μηχανή και ποιο είδος εντολών καταλαβαίνει;» «Πώς γίνεται η επικοινωνία, η έκδοση και αναγνώριση εντολών;» Για παράδειγμα, δε γνωρίζουν πώς ο υπολογιστής διαχειρίζεται τις μεταβλητές και δεν μπορούν να διακρίνουν αν τα μηνύματα που εμφανίζονται στην οθόνη αποτελούν αποτελέσματα των εσωτερικών λειτουργιών του ή της εκτέλεσης του προγράμματος.

Η τρίτη κατηγορία **(Notation)** αναφέρεται στις δυσκολίες που προκύπτουν από την ίδια τη γλώσσα προγραμματισμού που χρησιμοποιούν στην ανάπτυξη προγραμμάτων, συμπεριλαμβανομένων των συντακτικών και τη σημασιολογία των

προγραμματιστικών δομών. Οι μαθητές δυσκολεύονται να εξηγήσουν τη λειτουργία των εντολών και τον τρόπο που τις εκτελεί ο υπολογιστής (όπως αναφέρθηκε παραπάνω).

Η τέταρτη κατηγορία **(Structures)** αναφέρεται στη δυσκολία των μαθητών να ανακαλούν και να επαναχρησιμοποιούν τις προγραμματιστικές δομές για την επίλυση κάποιου προβλήματος. Παραδείγματα τέτοιων δομών είναι ο υπολογισμός ενός αθροίσματος με χρήση βρόχου, ένας αλγόριθμος αναζήτησης, ένας αλγόριθμος ταξινόμησης, κώδικας για την αντιμετάθεση τιμών κ.ά. Οι έμπειροι προγραμματιστές έχουν στο μυαλό τους έτοιμη τη λύση προς χρήση ανά πάσα στιγμή για τέτοιες δομές, αφού έχουν αντιμετωπίσει ανάλογες καταστάσεις στο παρελθόν. Οι αρχάριοι,λόγω απειρίας, στερούνται τέτοιας δυνατότητας με αποτέλεσμα να βρίσκει εμπόδια η προσπάθειά τους να λύσουν ανάλογα προβλήματα.

Η πέμπτη κατηγορία δυσκολίας **(Pragmatics)** αναφέρεται στις βοηθητικές δεξιότητες που είναι απαραίτητες για τον προγραμματισμό και αφορούν την ικανότητα προσαρμογής και ελέγχου ενός περιβάλλοντος στον υπολογιστή που θα χρησιμοποιηθεί για τη συγγραφή κώδικα, τη μεταγλώττιση και αποσφαλμάτωση των λαθών ενός προγράμματος. Οι μαθητές κάποιες φορές δυσκολεύονται να προσαρμοστούν στο περιβάλλον ανάπτυξης προγραμμάτων και να μάθουν να χρησιμοποιούν τα διαθέσιμα εργαλεία, πριν ακόμα αρχίσουν να μαθαίνουν την ίδια τη γλώσσα προγραμματισμού.

Ο Du Boulay παρατηρεί επίσης ότι οι μαθητές συχνά θεωρούν ότι ο υπολογιστής (ή το πρόγραμμα) θα επιστρέψει ως αποτέλεσμα αυτό που οι ίδιοι πιστεύουν (και θέλουν) και όχι αυτό που προκύπτει από την εκτέλεση των εντολών του προγράμματος που έχουν γράψει. Ακόμη θεωρεί ότι η χρησιμοποίηση λέξεων σε μια γλώσσα προγραμματισμού μπορεί να παραπλανήσει τους μαθητές κάνοντάς τους να σκεφτούν ότι ο υπολογιστής διαθέτει (όπως και ο άνθρωπος) την ικανότητα να συμπεραίνει τι εννοεί κάποιος με τα λεγόμενά του.

Επίσης ο Du Boulay σημειώνει ότι οι μαθητές αρκετές φορές αγνοούν ή είναι απληροφόρητοι για την κατάλληλη αντιμετώπιση ενός λάθους με αποτέλεσμα να καταφεύγουν σε λάθος ενέργειες, ακόμα και σε υπερπροσπάθεια για ένα ασήμαντο συντακτικό λάθος. Για παράδειγμα μπορεί να καταφύγουν σε διαγραφή όλου του προγράμματος ή ακόμα και σε επανεκκίνηση του υπολογιστή.

Ο Perkins μελέτησε τις στρατηγικές των μαθητών που μαθαίνουν να προγραμματίζουν. Όπως και ο Du Boulay, διαπίστωσε ότι οι αρνητικές εμπειρίες αναγκάζουν μερικούς μαθητές να σταματήσουν την προσπάθεια τους. Αυτοί οι μαθητές (Stoppers) εγκαταλείπουν την προσπάθειά τους μόλις βρουν ένα πρόβλημα. Διαπιστώθηκε ότι αυτή η τάση θα μπορούσε να αντιμετωπιστεί αν δοθεί στους μαθητές μία μικρή θετική εμπειρία για να τους ενθαρρύνει. Το άμεσο συμπέρασμα είναι ότι η πρώτη επαφή και εμπειρία στον προγραμματισμό, στο στάδιο όπου οι μαθητές καταστρώνουν την τεχνική του προγραμματισμού, είναι ιδιαίτερα σημαντική.

Η άλλη κλάση των μαθητών (Movers) ακολουθεί διαφορετικές στρατηγικές προκειμένου να συντάξει ένα σωστό πρόγραμμα, αν και όχι πάντα με επιτυχία. Οι μαθητές αυτοί πολλές φορές αλλάζουν τυχαία τον κώδικα ενός προγράμματος (προφανώς πειραματίζονται) χωρίς να κατανοούν που ακριβώς είναι το πρόβλημα.

Μια μέθοδος που χρησιμοποιείται συχνά για τη στήριξη των μαθητών στην αντιμετώπιση των παραπάνω δυσκολιών είναι η σχεδίαση και ανάπτυξη εκπαιδευτικών προγραμματιστικών περιβαλλόντων, χωρίς βέβαια αυτό να σημαίνει πως λύνουν αυτόματα όλα τα προβλήματα. Ωστόσο διευκολύνουν σημαντικά την μάθηση του εκπαιδευτικού αντικειμένου. Τα προγραμματιστικά περιβάλλοντα που χρησιμοποιούνται συνήθως στα μαθήματα προγραμματισμού:

i. Δεν διευκολύνουν και δεν παρέχουν πρόσβαση σε πληροφορίες, βοήθεια και παραδείγματα ώστε να μην αναγκάζονται οι μαθητές να θυμούνται λεπτομέρειες που αφορούν το συντακτικό και τη σημασιολογία της γλώσσας.
ii. Αναγκάζουν τους μαθητές να «μεταφράζουν» τη λύση που σκέφτονται (σε γενικό και αφηρημένο επίπεδο) σε χαμηλού επιπέδου δομές (προγραμματιστικές δομές που υποστηρίζει η γλώσσα).
iii. Χρησιμοποιούν λεξιλόγιο και συντακτικό των γλωσσών προγραμματισμού που δεν είναι οικεία στους μαθητές. Επιπλέον, επειδή η εκτέλεση του προγράμματος δε είναι διαφανής (η αλληλεπίδραση με τον υπολογιστή κατά την εκτέλεση ενός προγράμματος γίνεται μέσω μηνυμάτων), οι μαθητές, συχνά, θεωρούν ότι η εκτέλεση είναι μια διαδικασία ενεργειών εισόδου- εξόδου. Είναι δύσκολο να διαμορφώσουν μια ορθή εικόνα για τον τρόπο και τη σειρά εκτέλεσης των εντολών

1.6 - Διδακτικές προσεγγίσεις

1.6.1 - Κλασική Διδακτική προσέγγιση

Για τη διδασκαλία του προγραμματισμού, υπάρχουν πολλές μέθοδοι και προσεγγίσεις. Κάποιες από αυτές έχουν δοκιμαστεί αρκετά κατά το παρελθόν, στο πλαίσιο της διδακτικής πράξης, ενώ άλλες λειτουργούν ακόμη σε ερευνητικό επίπεδο. 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[9]. Ωστόσο, η προσέγγιση αυτή κρίνεται αναποτελεσματική, κυρίως για μαθητές μικρής ηλικίας, καθώς θέτει μια σειρά από εμπόδια στους αρχάριους προγραμματιστές[9], όπως παρακάτω αυτά ενδεικτικά εκτίθενται:

i. 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,
ii. η διδασκαλία και χρήση μιας πλήρους γλώσσας προγραμματισμού στο σχολείο μπορεί να αποβεί πολύ χρονοβόρα,
iii. παρέχεται συνήθως περιορισμένη υποστήριξη όσον αφορά την κατανόηση των βασικών εντολών και δομών ελέγχου της γλώσσας αφού η διαδικασία της εκτέλεσης του προγράμματος παραμένει κρυμμένη από τον μαθητή, ενώ η έλλειψη οπτικής ανάδρασης εμποδίζει την κατανόηση της σημασιολογίας της γλώσσας,
iv. οι μαθητές δύσκολα μπορούν να εντοπίσουν και να διορθώσουν τα λάθη στα προγράμματά τους
v. 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

1.6.2 - Η μέθοδος των προγραμματιστικών περιβαλλόντων

Μία διαφορετική μέθοδος προσέγγισης στα εισαγωγικά μαθήματα προγραμματισμού που έχει προταθεί είναι η χρήση εκπαιδευτικών περιβαλλόντων (environments) που **βασίζονται σε μίνι-γλώσσες** (mini-languages) **και μικρόκοσμους** (microworlds)[1]. Οι μίνι-γλώσσες είναι μικρές γλώσσες προγραμματισμού ειδικά σχεδιασμένες για τη διδασκαλία του προγραμματισμού. Οι μαθητές μαθαίνουν να προγραμματίζουν δίνοντας εντολές για να κινηθεί μια ψηφιακή οντότητα (π.χ. χελώνα, ρομπότ) που ζει σε έναν εικονικό κόσμο (μικρόκοσμο)[1]. Η οντότητα μπορεί να εκτελεί ένα μικρό σύνολο εντολών και να επιστρέψει τιμές σε ορισμένες ερωτήσεις. 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[1]. Τα προγραμματιστικά μινι-περιβάλλοντα συνιστούν ένα απλό αλλά ισχυρό μέσο για την εισαγωγή των μαθημάτων 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.

Τα *πλεονεκτήματα* της προσέγγισης αυτής είναι τα ακόλουθα[1]:

- Μια μίνι 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 αναπαριστάνονται 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 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 τα παραπάνω γίνεται σαφές ότι τα προγραμματιστικά μίνι-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.

Οι περισσότερες από τις μικρο-γλώσσες που έχουν κατασκευαστεί και έχουν εφαρμοστεί μέχρι σήμερα εστιάζουν στο Διαδικαστικό ή Συναρτησιακό προγραμματισμό. Ορισμένες από αυτές, στις νέες τους εκδόσεις, υποστηρίζουν χαρακτηριστικά και δομές του αντικειμενοστραφούς προγραμματισμού όπως η StarLogo, Karel++ και το MicroWorlds Pro, χωρίς όμως να είναι αμιγώς αντικειμενοστρεφή[5].

Μια νέα γενιά, αμιγώς αντικειμενοστρεφών περιβαλλόντων, έχουν αναπτυχθεί τα τελευταία χρόνια και βρίσκονται στο στάδιο της ερευνητικής τους εφαρμογής και χρήσης, ως εργαλεία ανάπτυξης της προγραμματιστικής σκέψης και της εισαγωγής στον προγραμματισμό.Χαρακτηριστικά παραδείγματα τέτοιων περιβαλλόντων είναι το Objectkarel, το AgentSheets, και το ALICE.

Το **ALICE** χρησιμοποιεί μια πολύ πλούσια σε δομές και εντολές αντικειμενοστραφούς γλώσσας προγραμματισμού σε ένα τρισδιάστατο εικονικό κόσμο, το οποίο απευθύνεται σε αρχάριους προγραμματιστές όπως είναι οι μαθητές της Τριτοβάθμιας και της Δευτεροβάθμιας εκπαίδευσης. Το ALICE είναι ένα από τα σημαντικότερα και πλέον αποδοτικά προγραμματιστικά περιβάλλοντα εκμάθησης προγραμματισμού, στο οποίο και θα επικεντρωθεί η παρούσα εργασία.

1.6.3 - Η "object-first" προσέγγιση

Ο τρόπος διδασκαλίας του αντικειμενοστρεφή προγραμματισμού, τα τελευταία χρόνια έχει αλλάξει υιοθετώντας αυτή την προσέγγιση. Χαρακτηριστικό της προσέγγισης, είναι η **έμφαση στις αντικειμενοστραφείς έννοιες** της κλάσης (class) και του αντικειμένου (object) και η παρουσίασή τους από τα αρχικά μαθήματα. Εδώ, η σημασία της βασικής για την αντικειμενοστρεφή φιλοσοφία προγραμματισμού έννοιας του αντικειμένου (object) και αποτελεί το βασικό στοιχείο διδασκαλίας από την αρχή, όπως αναφέρεται στο IEEE/ACM Computing Curricula[13].

Η έμφαση δίνεται στις έννοιες που συνιστούν τη φιλοσοφία του προγραμματισμού και αναδεικνύει τη σημασία τους ως δομικά συστατικά των προγραμμάτων. Οι δομές της γλώσσας προγραμματισμού που χρησιμοποιούνται μαζί με τις λεπτομέρειες των εντολών και της σύνταξης τους, παρουσιάζονται όταν απαιτούνται για την εφαρμογή των εννοιών και την ανάδειξη του ρόλου τους στην επίλυση προβλημάτων. Ο τρόπος αυτός *βοηθά στην εμπέδωση της αντικειμενοστρέφειας* ως παραδείγματος προγραμματισμού σε πρόωρη φάση κατά τη διάρκεια της διδασκαλίας και συνεπώς της φιλοσοφίας της.

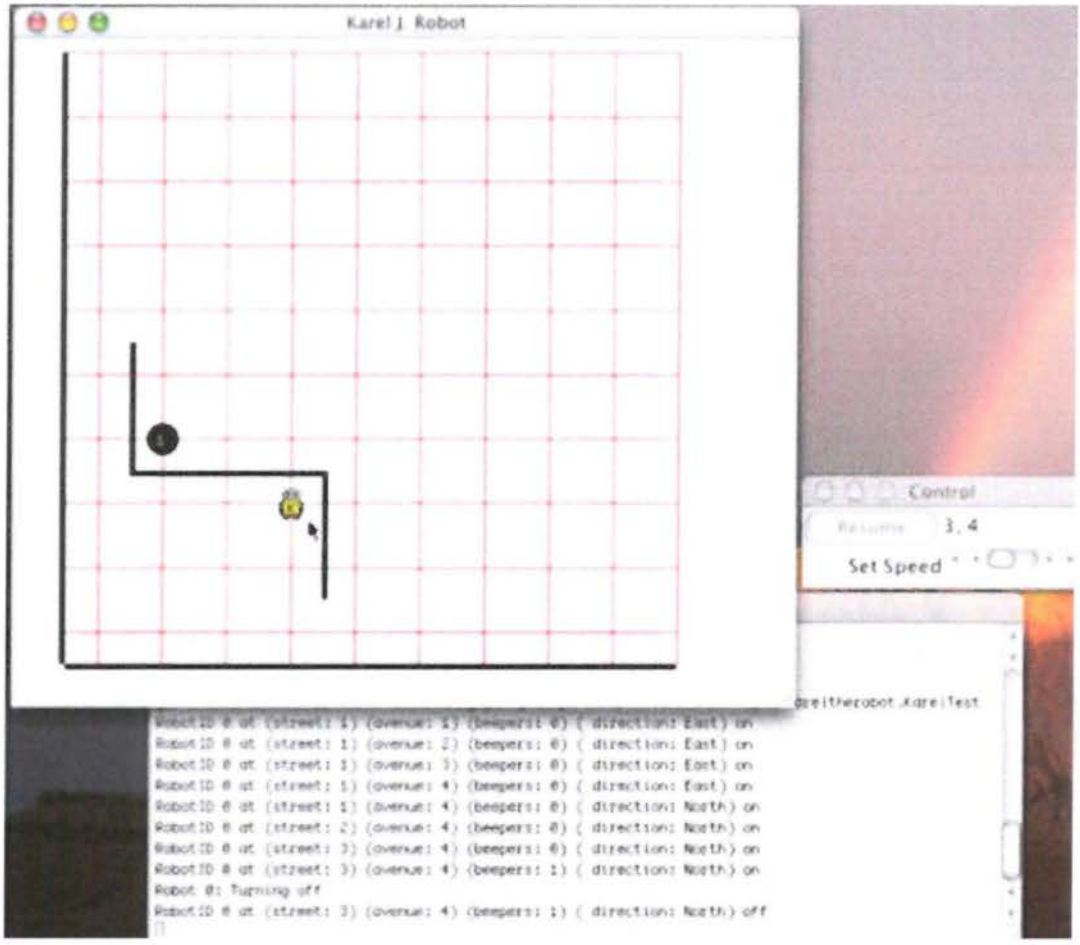
Ένα πλήθος εκπαιδευτικών εργαλείων και προγραμματιστικών περιβαλλόντων έχει σχεδιαστεί για την υποστήριξη αυτής της διδακτικής προσέγγισης στη διδασκαλία του Αντικειμενοστραφούς Προγραμματισμού. Αυτά, αποτελούν προγραμματιστικούς μικρόκοσμους, όπως είναι οι KarelJ Robot, ALICE, objectKarel, Jeroo, ολοκληρωμένα προγραμματιστικά περιβάλλοντα με εκπαιδευτικά χαρακτηριστικά όπως το BlueJ και το greenfoot (για μαθητές σχολικής ηλικίας), εργαλεία που υποστηρίζουν την προσέγγιση μέσω γραφικού περιβάλλοντος διασύνδεσης χρήστη και εργαλεία που βρίσκονται σε γραφικές βιβλιοθήκες (Grafic Libraries).

1.7 - Προγραμματιστικά Εργαλεία που χρησιμοποιούνται στην εκπαιδευτική διαδικασία- Εκπαιδευτικά εργαλεία και προγραμματιστικά περιβάλλοντα για τη διδακτική του αντικειμενοστραφούς προγραμματισμού

1.7.1 - KarelJ.Robot

Το **KarelJ.Robot** αποτελεί τη μετάφραση που ο Bergin έκανε στο μικρόκοσμο Karel++ σε αμιγώς γλώσσα Java δηλαδή ένα αντικειμενοστρεφές Java περιβάλλον.

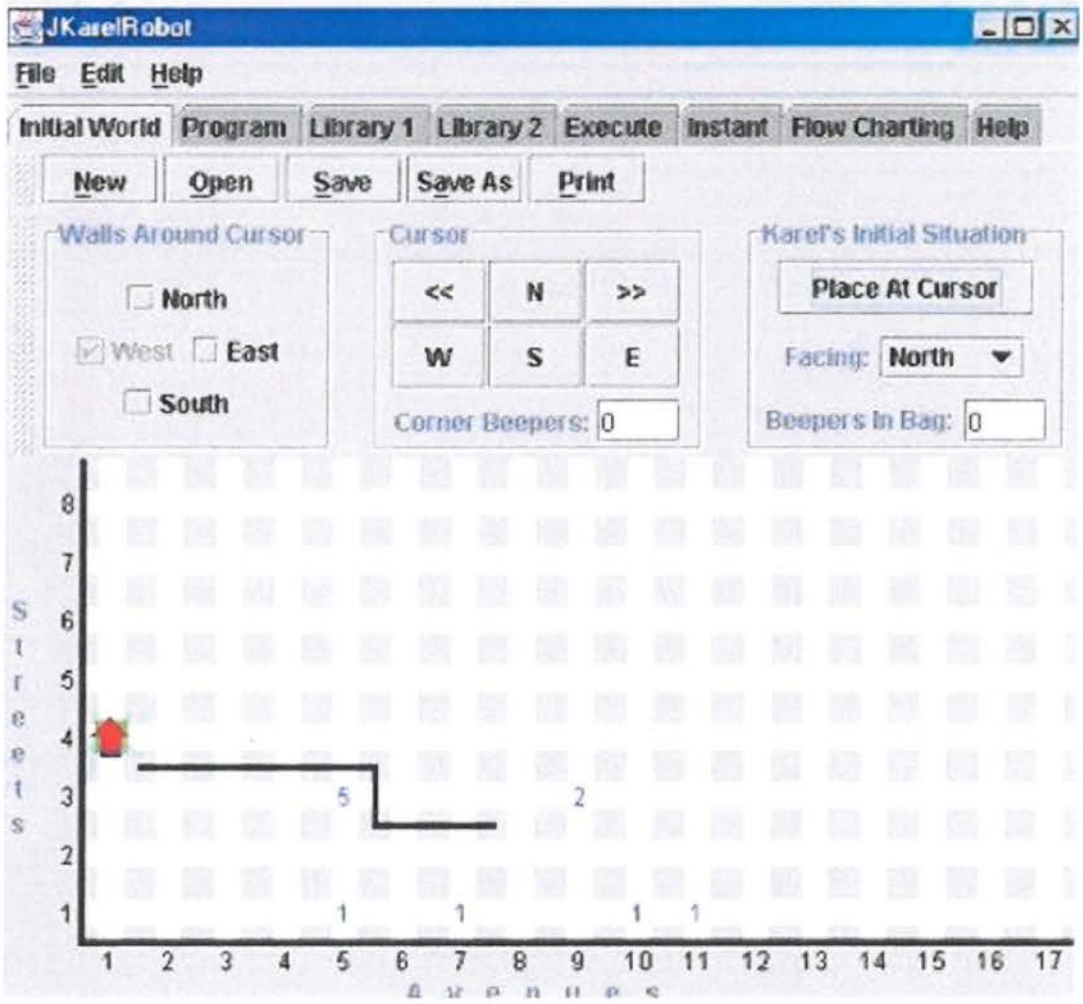
http://csis.pace.edu/~bergin/KarelJava2ed/karelexperimental.html



1.7.2 – JKarelRobot

Το **JKarelRobot** είναι ένας μικρόκοσμος ανεξαρτήτως πλατφόρμας (γραμμένος σε Java) και παραδείγματος προγραμματισμού, ο οποίος υποστηρίζει τύπου Pascal, Java και Lisp περιβάλλοντα και επίσης διαγράμματα ροής (flow charts) (με αντίστροφη κατεύθυνση από την παραδοσιακή, δηλαδή για τη μετατροπή έτοιμων προγραμμάτων σε διαγράμματα ροής) και τη δυνατότητα διαγραμματικής παρουσίασης του προγράμματος με τη χρήση διαγραμμάτων δομής (control structure diagrams,CSD). Τα τελευταία έχουν στόχο τη βελτίωση της καταληπτότητας του πηγαίου κώδικα και είναι διαγράμματα που απεικονίζουν γενικά τη συνολική δομή κάθε τμήματος του προγράμματος. Αναπτύχθηκε από τους Buck και Stucki και χρησιμοποιήθηκε στο Otterbein College.

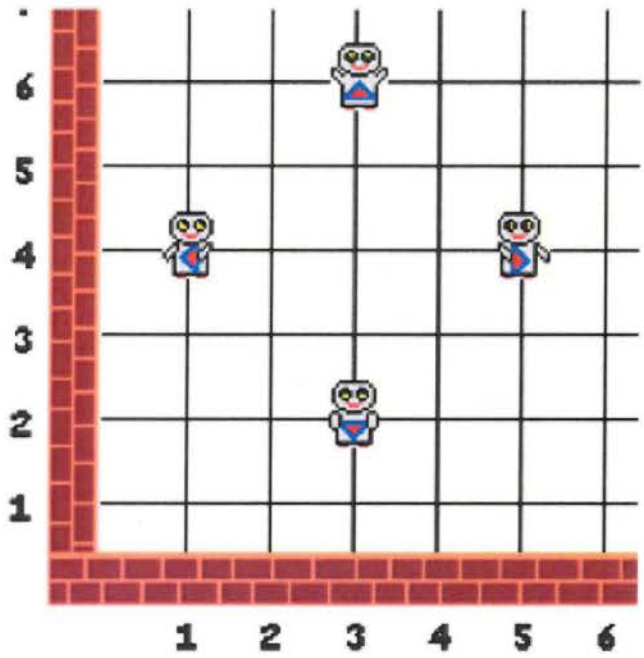
http://math.otterbein.edu/home/Class/Csc120/WebPages/KarelStart.html



1.7.3 – objectKarel

Ο προγραμματιστικός μικρόκοσμος **objectkarel**, αποτελεί ένα αξιόλογο εκπαιδευτικό περιβάλλον το οποίο βασίζεται στον Karel++ και αναπτύχθηκε στο τμήμα Εφαρμοσμένης Πληροφορικής του Πανεπιστημίου Μακεδονίας. Σημαντικές πρωτοτυπίες αποτελούν η ύπαρξη του εκδότη δομής (structure editor) για τη δημιουργία των προγραμμάτων, η δυνατότητα της επεξηγηματικής οπτικοποίησης (δηλαδή της εμφάνισης επεξηγήσεων για την τρέχουσα εντολή), η σειρά μαθημάτων (e-lessons) που διαθέτει στα ελληνικά και στα αγγλικά και των δραστηριοτήτων για την εξοικείωση των σπουδαστών με τις βασικές αντικειμενοστρεφείς έννοιες και τις δομές ελέγχου και η δυνατότητα καταγραφής των ενεργειών των σπουδαστών καθώς δημιουργούν προγράμματα (καταγραφή των προγραμμάτων και όλων των μεταγλωττίσεών τους) για τον προσδιορισμό δυσκολιών και παρανοήσεών τους.[18]

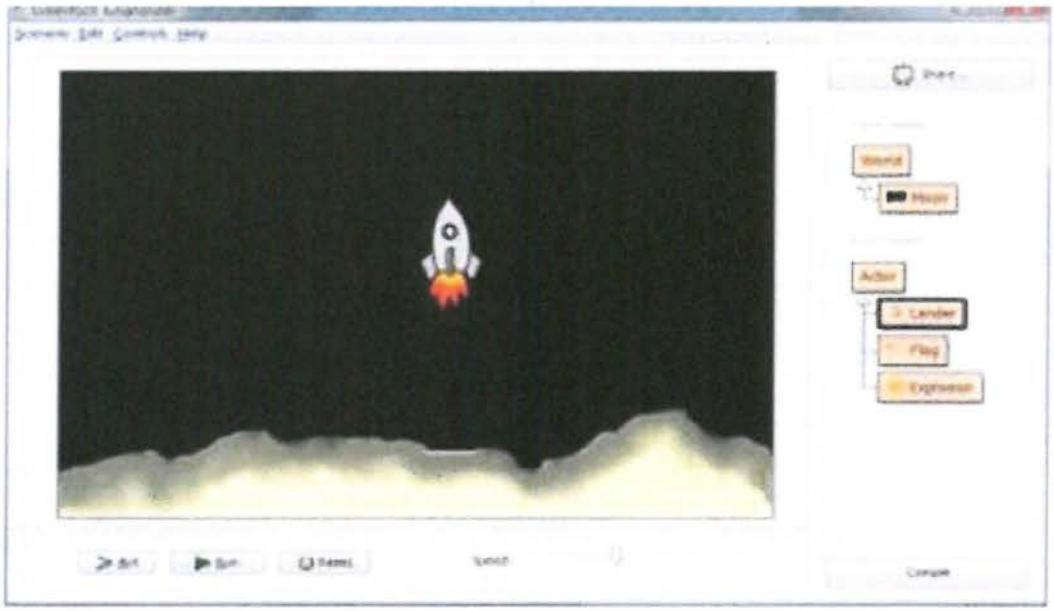
http://csis.pace.edu/~bergin/temp/findkarel.html



1.7.4 – Greenfoot

Το **Greenfoot** αποτελεί ένα εργαλείο που στοχεύει στην υποστήριξη της διδασκαλίας του Αντικειμενοστραφούς Προγραμματισμού σε εκπαιδευόμενους σχολικής ηλικίας δηλαδή του Λυκείου ή χαμηλότερης βαθμίδας. Σχεδιάστηκε συνδυάζοντας τα πλέον χρήσιμα χαρακτηριστικά μερικών ήδη υπαρχόντων εργαλείων. Κυρίως ο σχεδιασμός του εμπνεύσθηκε από την απλότητα και την πολύ καλή οπτικοποίηση των αντικειμένων, της κατάστασής τους και της συμπεριφοράς τους (object visualization), που διαθέτει το περιβάλλον του Karel the Robot (microworld) και από την ευελιξία και τη δυνατότητα άμεσης αλληλεπίδρασης (direct interaction) με τα αντικείμενα και τις κλάσεις που διαθέτει το περιβάλλον BlueJ. Αποτελεί συνδυασμό ενός πλαισίου για τη δημιουργία προγραμμάτων σε Java που μπορούν να οπτικοποιηθούν σε δισδιάστατο πλέγμα (two-dimensional grid) και ενός ολοκληρωμένου περιβάλλοντος ανάπτυξης (με class browser, editor, compiler, execution control, debugger κλπ.) κατάλληλο για αρχάριους προγραμματιστές.

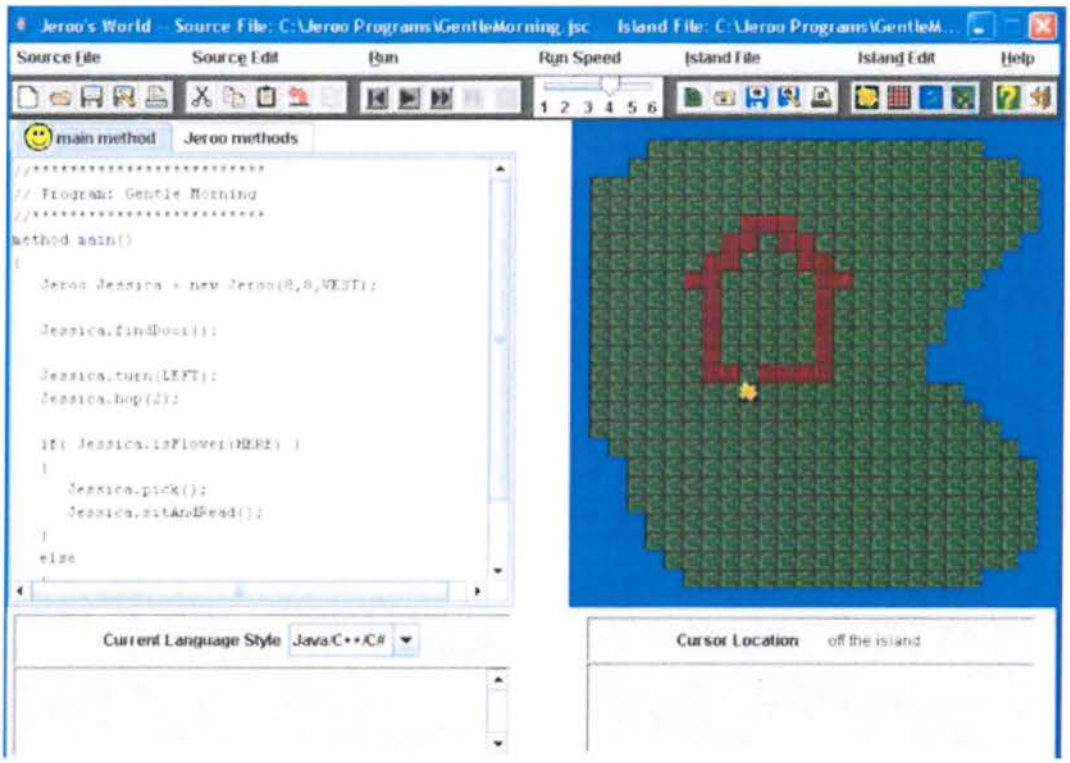
http://www.greenfoot.org



1.7.5 – Jeroo

Ο προγραμματιστικός μικρόκοσμος **Jeroo** χρησιμοποιεί ως μεταφορέα ένα σπάνιο είδος θηλαστικού παρόμοιο με τα μικρόσωμα καγκουρό της Αυστραλίας. Από κατασκευής άποψης, το περιβάλλον εστιάζει στις έννοιες «αντικείμενο» και «μέθοδος» και στις δομές ελέγχου (control structures). Υποστηρίζει δύο οικογένειες γλωσσών προγραμματισμού, Java/C++/C# και VB.NET. Το δεύτερο είναι ένα πραγματικό υποσύνολο της VB.NET. Εκτός από τη συνάφεια της γλώσσας του περιβάλλοντος με τη Java και την ευκολία σύνταξης της, πλεονέκτημά του αποτελεί και το ότι όλα είναι ορατά κάθε στιγμή σε ένα και μοναδικό παράθυρο. Ο κώδικας, το περιβάλλον του μεταφορέα, η περιοχή μηνυμάτων μεταγλώττισης και εκτέλεσης, η περιοχή ένδειξης της θέσης του αντικειμένου στο περιβάλλον, η περιοχή ένδειξης της κατάστασης του αντικειμένου κατά την εκτέλεση του προγράμματος επίσης διαθέτει οπτικοποίηση της κίνησης των αντικειμένων, οπτικοποίηση της εκτέλεσης του κώδικα (τονίζεται η γραμμή κώδικα η οποία εκτελείται με αλλαγή του χρώματος των χαρακτήρων της και του υποβάθρου της), παρέχει τη δυνατότητα εκτέλεσης βήμα προς βήμα και συνεχόμενα και επιλογής ταχύτητας εκτέλεσης.

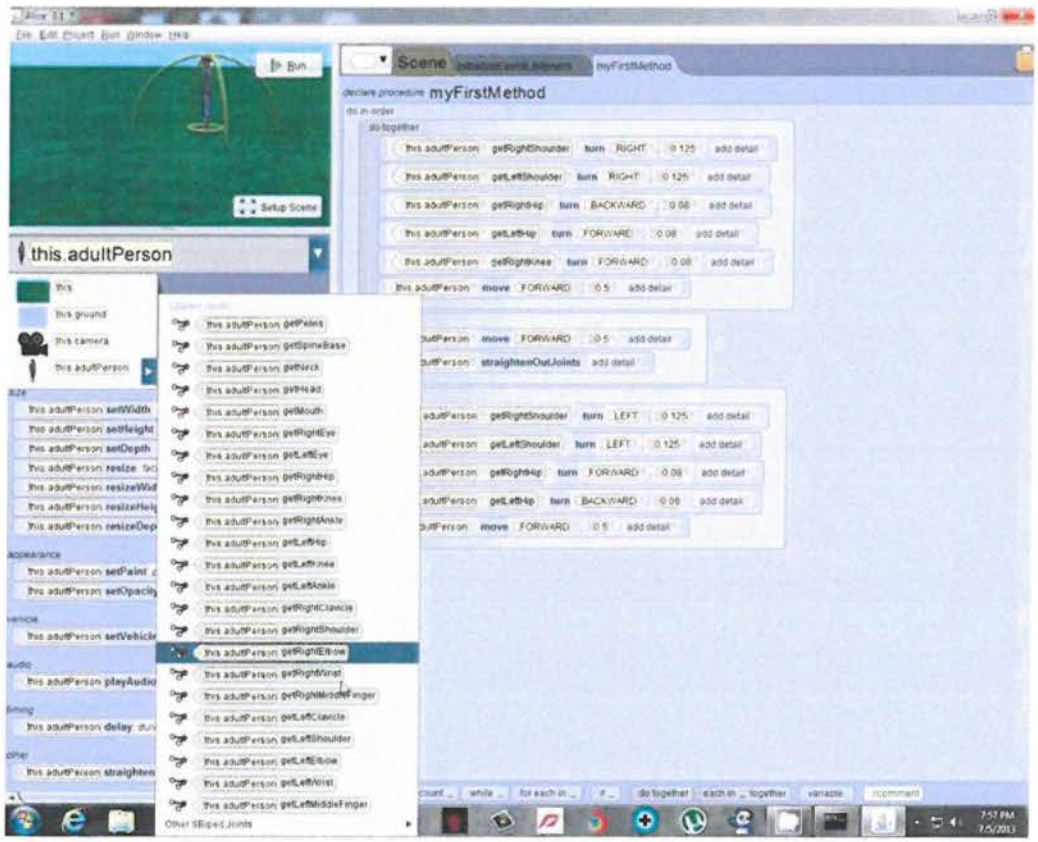
http://www.jeroo.org/



1.7.6 – ALICE

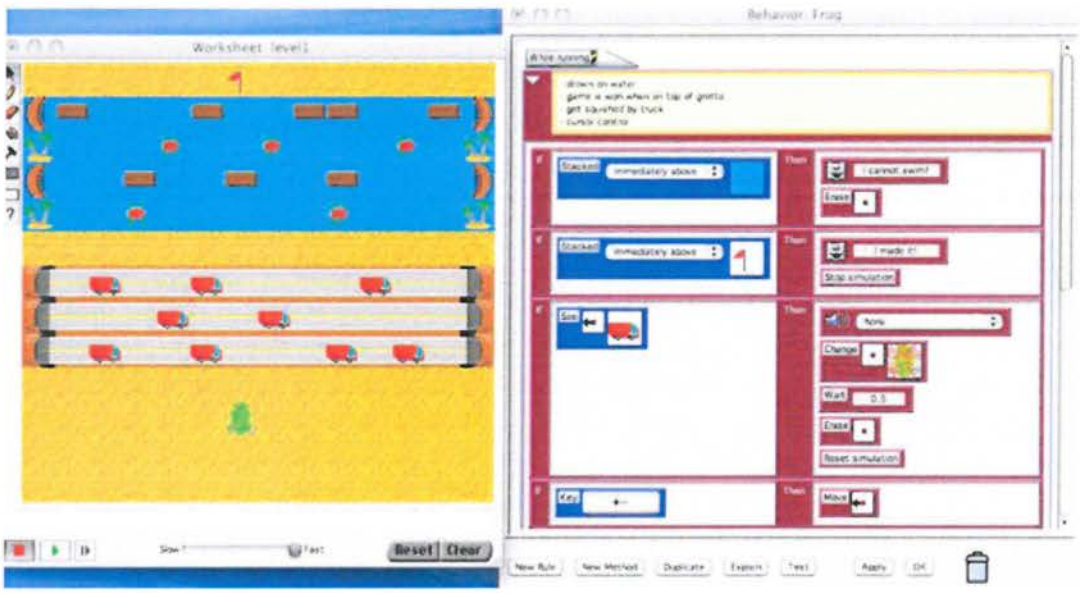
Το εργαλείο λογισμικού **ALICE** είναι ένα τρισδιάστατο διαδραστικό περιβάλλον απεικόνισης (animation) στο οποίο οπτικοποιούνται τα αντικείμενα και οι συμπεριφορές τους και το οποίο επιχειρεί να υλοποιήσει την object-first προσέγγιση στη διδασκαλία εισαγωγικών μαθημάτων προγραμματισμού. Η τριών διαστάσεων απεικόνιση «ενισχύει» την οπτικοποίηση των αντικειμένων παρέχοντας μια αίσθηση πραγματικότητας γι' αυτά και παρέχει ένα ευέλικτο και εποικοδομητικό πλαίσιο για την κατανόηση των αντικειμενοστρεφών εννοιών. Ουσιαστικά ακολουθεί την παράδοση των μικρόκοσμων Karel παρέχοντας μια επιπλέον διάσταση. Οι τριών διαστάσεων κόσμοι είναι περισσότερο ρεαλιστικοί από τους αντίστοιχους δισδιάστατους. Στο περιβάλλον αυτό οι μαθητές μπορούν να δημιουργήσουν τους δικούς τους εικονικούς κόσμους. Το εργαλείο αναπτύχθηκε από τον Randy Pausch και την ερευνητική ομάδα Stage 3 research στο πανεπιστήμιο Carnegie Mellon (Carnegie Mellon University, CMU).[12]

http://www.ALICE.org



1.7.7 – AgentSheets

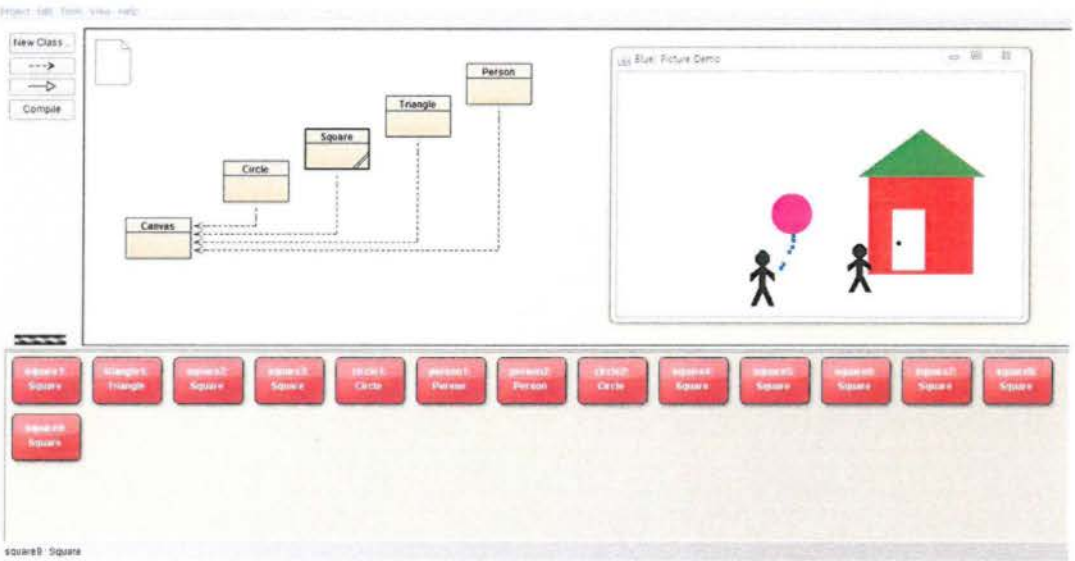
Το **AgentSheets** είναι ένα εκπαιδευτικό διαδικτυακό εργαλείο που χρησιμοποιείται για τη δημιουργία προσομοιώσεων παιχνιδιών. Χρησιμοποιείται από εκπαιδευτικούς πολλών χωρών, προκειμένου να διδάξουν προγραμματισμό στους μαθητές τους μέσα από τη σχεδίαση παιχνιδιών. Το AgentSheets παρέχει μια drag-and-drop γλώσσα που δίνει τη δυνατότητα στους μαθητές που δεν γνωρίζουν προγραμματισμό, να δημιουργήσουν τα δικά τους παιχνίδια και να τα δημοσιεύσουν στο διαδίκτυο.



1.7.8 – BlueJ

Το **BlueJ** είναι ένα εκπαιδευτικό προγραμματιστικό περιβάλλον για τη διδασκαλία αντικειμενοστραφούς προγραμματισμού. Χρησιμοποιεί την «object- first» προσέγγιση. Εστιάζει στην αντικειμενοστρέφεια και αποτελεί ένα πλήρες περιβάλλον ανάπτυξης εφαρμογών σε Java. Παρέχει οπτικό περιβάλλον και εύκολη μεταγλώττιση με επισήμανση των λαθών κατευθείαν στο συντάκτη κώδικα, με τονισμό της γραμμής του λάθους και επίδειξη του κειμένου του μηνύματος λάθους.

http://www.bluej.org/



1.8 - Πορίσματα που έχει αναδείξει η έρευνα σχετικά με την αξιοποίηση του ALICE και άλλων προγραμματιστικών εργαλείων σε τυπικά εκπαιδευτικά περιβάλλοντα

Το πρόγραμμα ALICE επιλέχθηκε για τους εξής λόγους:

- Η εργασία με την βοήθεια ενός 3D περιβάλλοντος είναι ελκυστική και υποκινεί την σημερινή γενιά
- Η οπτική φύση και η άμεση ανάδραση του προγράμματος κάνουν εύκολο στους μαθητές να δουν την επίδραση μιας εντολής που έχουν δώσει. Επιπλέον κάνει τον έλεγχο του προγράμματος ευκολότερο.
- Ο συντάκτης σκηνών, που λειτουργεί με σύρσιμο και επιλογή, εμποδίζει τους μαθητές από το να κάνουν συντακτικά λάθη στα οποία οι αρχάριοι είναι ευάλωτοι.
- Οι κλάσεις και τα 3D αντικείμενα στο ALICE παρέχουν μια εμφανή ιδέα και περιγραφή του αντικειμένου.

Χρησιμοποιώντας το ALICE, δεν παραλήφθηκε καμία από τις βασικές έννοιες του αντικειμενοσταρφούς προγραμματισμού – συναρτήσεις, μέθοδοι, κλάσεις, κληρονομικότητα.

Για τους μαθητές του "The Imaginary Worlds Camps", η δημιουργία των προσωπικών τους ταινιών ήταν ένα κίνητρο και μια άκρως ευχάριστη ενασχόληση, λόγω:

- Το πρόγραμμα ALICE απαγορεύει την ύπαρξη συντακτικών λαθών, τα οποία είναι και ο κύριος παράγοντας για τον οποίο αποφεύγουν οι αρχάριοι τον προγραμματισμό.
- Επειδή τα λάθη δεν είναι συντακτικά, αλλά λογικά μπορούν εύκολα να διορθωθούν και να προκαλέσουν γέλιο αντί για απογοήτευση.

ΜΕΡΟΣ 2^{Ο}

Αντικειμενοστραφής Προγραμματισμός και παρούσιαση

ALICE

2.1 - Αντικειμενοστραφής προγραμματισμός και Έννοιες

Ο αντικειμενοστραφής προγραμματισμός έγινε πολύ γνωστός στην επιστήμη των υπολογιστών. Ένα αντικειμενοστραφές πρόγραμμα θεωρείται ως μια συλλογή από αντικείμενα με συμπεριφορά, χαρακτηριστικά και ιδιότητες παρά μια λίστα εντολών. Έτσι η επαναχρησιμοποίηση ορισμένων τμημάτων κώδικα και οι φόρμες των αντικειμένων έχουν κάνει ευκολότερη την συντήρηση και την αναβάθμιση των προγραμμάτων.

Παρόλα αυτά παρουσιάστηκε ένα *πρόβλημα*. Οι μαθητές της μέσης εκπαίδευσης, ακόμη και οι πανεπιστημιακοί φοιτητές δεν επέλεγαν εύκολα ένα μάθημα προγραμματισμού. Οι περισσότεροι φοιτητές δεν προσελκύονταν από ένα μάθημα προγραμματισμού και αυτοί που απλά παρακολουθούσαν, για δοκιμή, ένα τέτοιο μάθημα δεν συγκινούνταν ώστε να το επιλέξουν. Οι κυριότεροι λόγοι ήταν απογοήτευση των μαθητών λόγω του επιπέδου δυσκολίας και η αμφιβολία για το αν θα καταφέρουν να ανταπεξέλθουν στις απαιτήσεις ενός τέτοιου μαθήματος.

Με αφορμή το παραπάνω γεγονός έγινε μια έρευνα από το Πανεπιστήμιο Carnegie Mellon και τελικά **το 1999 δημιουργήθηκε ένα εκπαιδευτικό προγραμματιστικό περιβάλλον με το όνομα "Alice"**. Η αρχική ιδέα του προγράμματος ALICE εμφανίστηκε στις αρχές του 1991, όταν τα πρώτα συστήματα εικονικών κόσμων άρχισαν να δημιουργούνται στο Πανεπιστήμιο της Virginia. Ξεκινώντας με την επιθυμία δημιουργίας νέων τεχνικών αλληλεπίδρασης για την προσομοίωση προγραμμάτων, ανακαλύφθηκε ότι τα εργαλεία που ήταν διαθέσιμα για αυτό το σκοπό ήταν δύσκολα στη χρήση. Έτσι ξεκίνησε η ιδέα για δημιουργία νέων προγραμματιστικών εργαλείων που ήταν γρηγορότερα και ευκολότερα στη χρήση για οποιονδήποτε είχε την επιθυμία

να μάθει να προγραμματίζει, ακόμη και για τους νέους μαθητές. Αυτή η προσπάθεια ξεκίνησε με πολλά διστακτικά προσχέδια με διάφορα ονόματα και τελικά κατέληξε στο σημερινό πρόγραμμα ALICE.

Το ALICE όπως αναφέρθηκε παραπάνω δημιουργήθηκε ουσιαστικά το 1999 από το Πανεπιστήμιο Carnegie Mellon με κύριους ερευνητές τους Wanda Dann, Stephen Cooper και Randy Pausch. **Το ALICE είναι ένα προγραμματιστικό περιβάλλον που χρησιμοποιείται ευρέως για εκπαιδευτικό σκοπό**. Είναι ένα 3D περιβάλλον για δημιουργία εικονικών κόσμων που εμπεριέχει δυναμικές κινήσεις των χαρακτήρων και αλληλεπίδραση τους με το χρήστη. Αυτό είναι και το κύριο πλεονέκτημα του ALICE. Χρησιμοποιεί 3D γραφικά ώστε να προσελκύσει τους μαθητές. Είναι πολύ πιο εύκολο και ευχάριστο, για παράδειγμα, να χρησιμοποιήσει κανείς μια κλάση Ανθρώπου – να της δώσει χαρακτηριστικά όπως φύλλο, χρώμα δέρματος, ύψος, χρώμα μαλλιών και ματιών κ.ά. - σαν αντικείμενο στον κόσμο τους προγράμματος που δημιουργεί, παρά να γράψει αμέτρητες γραμμές κώδικα χωρίς να έχει άμεσο αποτέλεσμα στην οθόνη του υπολογιστή.

Για τον παραπάνω λόγο το πρόγραμμα ALICE έγινε ιδιαίτερα γνωστό και αποδεκτό σε ευρύ προγραμματιστικό κοινό. Κυρίως όμως είχε απήχηση σε φοιτητές και παιδιά σχολείου που πλέον έβρισκαν ευχάριστο τον προγραμματισμό με αυτό το εργαλείο. Ακόμη και οι γυναίκες σπουδαστές ξεκίνησαν να επιλέγουν μαθήματα προγραμματισμού που γίνονταν με την βοήθεια του προγράμματος ALICE γιατί αισθάνονταν εξοικειωμένες με χαρακτήρες που υπάρχουν στην βιβλιοθήκη της ALICE. Μερικοί ισχυρίζονται ότι το όνομα του προγράμματος ALICE έχει δοθεί επίτηδες σε γένος θηλυκού έτσι ώστε να δημιουργηθεί ένα κλίμα «οικειότητας» και να προσεγγίσει περισσότερες γυναίκες. Στην πραγματικότητα **το σύστημα ονομάστηκε ALICE προς τιμήν του Charles Lutwidge Dodson ο οποίος είχε ψευδώνυμο Lewis Caroll**. Ο Lewis Caroll ήταν ο μαθηματικός που έγραψε το "ALICE's Adventures in Wonderland" και το "Through the Looking Glass". Ο Caroll γνώριζε ότι το σημαντικότερο θέμα είναι να γίνουν τα πράγματα απλά και ενδιαφέροντα προς τον μαθητή.

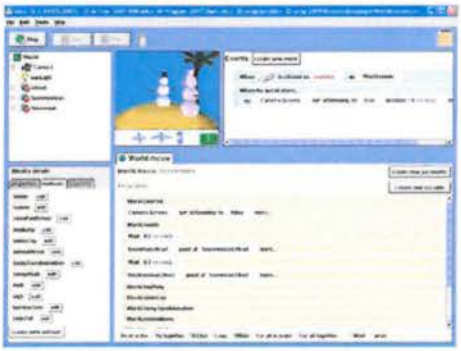
Αναλυτικότερα, στο προγραμματιστικό περιβάλλον ALICE χρησιμοποιείται ένας συντάκτης μέσα στον οποίο **ο χρήστης απλά επιλέγει, σύρει και αφήνει τα αντικείμενα και τις μεθόδους τους**. Με αυτό τον τρόπο αποφεύγονται τα συντακτικά λάθη. Μπορεί να συμβούν λογικά λάθη αλλά εφόσον ο χρήστης μπορεί να εκτελέσει το πρόγραμμα, θα εντοπίσει οπτικά το λάθος και θα το διορθώσει εύκολα και κατόπιν δοκιμών. Έτσι προσαρμόζεται εύκολα ο κώδικας και εντοπίζονται τα αποτελέσματα των αλλαγών. Στην εικόνα παρακάτω φαίνεται ο συντάκτης σκηνών.

Στο ALICE οι κλάσεις και τα αντικείμενα είναι κάτι που μπορεί κανείς να δει και να επιλέξει πριν τα προσθέσει στο πρόγραμμα του. **Η ευκολία χρήσης του προγράμματος, κατέστησε το ALICE ένα κοινώς αποδεκτό πρόγραμμα το οποίο πλέον χρησιμοποιείται σε πολλά πανεπιστήμια και σχολεία.** Ενδεικτικά μερικά από αυτά είναι:

California State University at Humboldt
Camden County College
Cornell University
Duke University
Mississippi Valley State University
Plymouth State University
Saint Edward's University
San Diego State University
Sierra Nevada College
University of Colorado
University of Illinois
κ.ά.

Η χρήση του προγράμματος ALICE δεν παρέμεινε μόνο στα σχολεία, προχώρησε και σε άλλες εφαρμογές. Ένα τέτοιο παράδειγμα είναι η συνεργασία του Carnegie Mellon με την EA (Electronic Arts), εταιρία δημιουργίας και παραγωγής videogames. Έτσι το ALICE συνεργάζεται με χαρακτήρες του παιχνιδιού Sims 2 της EA. Το παιχνίδι Sims 2 είναι ένα παιχνίδι που είναι γνωστό παγκοσμίως. Στις 10 Μαρτίου, λοιπόν, του 2006 στο Pittsburg, το Πανεπιστήμιο Carnegie Mellon συμφώνησε μια συνεργασία με την EA, που είχε σκοπό την αναζωογόνηση της επιστήμης των υπολογιστών στην Αμερική σε όλες τις βαθμίδες εκπαίδευσης. Η EA συμφώνησε να βοηθήσει στην ανάπτυξη του ALICE 3.0 παρέχοντας χαρακτήρες του παιχνιδιού The Sims – του παιχνιδιού για ηλεκτρονικούς υπολογιστές που έχει πουλήσει περισσότερο από κάθε άλλο παιχνίδι της εταιρίας. Το περιεχόμενο του Sims θα μετατρέψει το ALICE από ένα άτεχνο και άκομψο 3D προγραμματιστικό εργαλείο σε ένα όμορφο και φιλικό προς το χρήστη προγραμματιστικό περιβάλλον.

Με αυτόν τον τρόπο οι μαθητές που θα χρησιμοποιούν το ALICE 3.0 θα δουλεύουν ουσιαστικά σε ένα περιβάλλον όμοιο με αυτό του Sims, εφόσον οι χαρακτήρες θα μοιάζουν και θα κινούνται όπως αυτοί του Sims και η βιβλιοθήκη του Sims θα ενσωματωθεί στο πρόγραμμα. Με αυτό τον τρόπο ο προγραμματισμός θα γίνει ακόμη πιο διασκεδαστικός. Παρακάτω φαίνεται η διαφορά ανάμεσα στους χαρακτήρες των προγραμμάτων ALICE v.2 και ALICE v.3:



2.2 - Το περιβάλλον του ALICE και η χρήση του στην Εκπαιδευτική Πράξη

Είναι ένα εκπαιδευτικό προγραμματιστικό περιβάλλον με δυνατότητα τρισδιάστατης απεικόνισης του προγράμματος. **Σκοπός της χρήσης του είναι να εισάγει τους μαθητές στις βασικές έννοιες του αντικειμενοστραφούς προγραμματισμού** (π.χ. κλάση, αντικείμενο) **και στις προγραμματιστικές δομές** (π.χ. δομή επιλογής, δομή επανάληψης). Το πλεονέκτημα κατά τον Ράντυ Πάους, ο οποίος ήταν ένας από τους δημιουργούς, είναι ότι διδάσκει προγραμματισμό **με τη χρήση της «προσποίησης»**, δηλαδή όταν διδάσκεις σε κάποιον κάτι αφήνοντάς τον να πιστεύει ότι μαθαίνει κάτι άλλο. Έτσι οι μαθητές νομίζουν ότι χρησιμοποιούν το «ALICE» για να φτιάξουν ταινίες ή ηλεκτρονικά παιχνίδια. Η «προσποίηση» έγκειται στο ότι στην πραγματικότητα μαθαίνουν πώς να γίνουν προγραμματιστές.

Με τη χρήση του ALICE οι μαθητές δημιουργούν εικονικούς κόσμους, στους οποίους μπορούν να παρουσιάζονται οι δυναμικές κινήσεις των αντικειμένων (π.χ. κίνηση αυτοκινήτου, κίνηση ενός ζώου) και οι αλληλεπιδράσεις τους με το χρήστη (π.χ. οδήγηση αυτοκινήτου), χαρακτηριστικά τα οποία αποτελούν και το κύριο πλεονέκτημα του ALICE. Χρησιμοποιώντας ένα πρωτότυπο περιβάλλον γραφής κώδικα, επιτρέπει στο χρήστη να δημιουργήσει, όπως αναφέραμε πιο πάνω εικονικούς κόσμους, με τρισδιάστατα αντικείμενα (π.χ. άνθρωποι, ζώα, οχήματα) που κατοικούν σε αυτούς και μέσω της συγγραφής του κώδικα του προγράμματος και των παρεχομένων εντολών τα αντικείμενα αυτά να κινούνται. Ο χρήστης δημιουργεί ένα πρόγραμμα με τη μέθοδο «σύρε και άφησε» τις εντολές/εκφράσεις (οι οποίες δίνονται έτοιμες) με το ποντίκι στο χώρο που δημιουργείται ο κώδικας, αντί να γράφει τον κώδικα λέξη προς λέξη. Ακόμη,

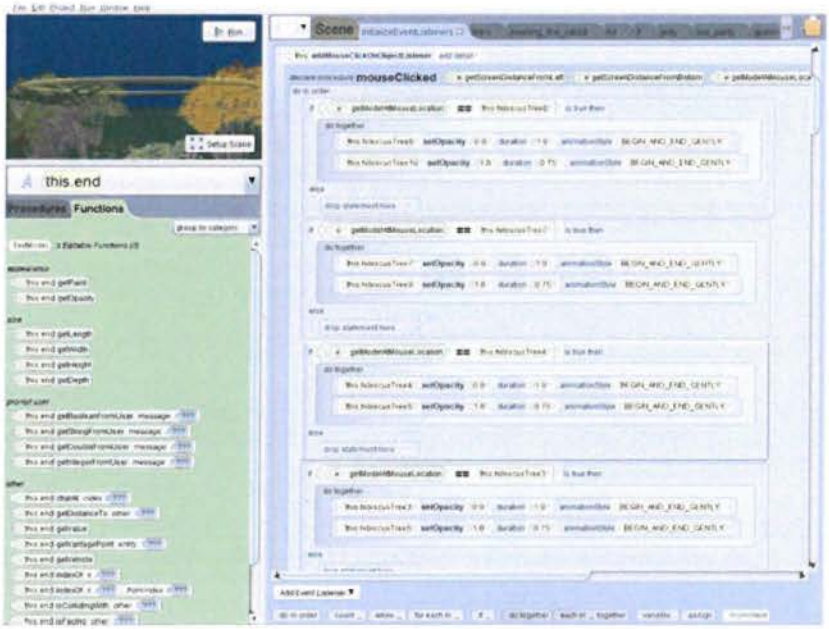
όποτε επιθυμεί ο χρήστης, μπορεί να δει να εκτελείται το πρόγραμμα, που έχει γράψει, σε ένα παράθυρο σε κινούμενη τρισδιάστατη απεικόνιση του μικρόκοσμου.

Το εργαλείο Alice χρησιμοποιεί τρισδιάστατα γραφικά και αυτό έχει ως αποτέλεσμα να προσελκύει τους μαθητές. Για παράδειγμα, είναι πολύ πιο εύκολο και ευχάριστο να δημιουργήσει κάποιος μια κλάση Ανθρώπου (να της δώσει χαρακτηριστικά όπως φύλλο, χρώμα μαλλιών, ύψος και άλλα) σαν αντικείμενο στον κόσμο τους προγράμματος, παρά να γράψει αμέτρητες γραμμές κώδικα χωρίς να έχει άμεσο αποτέλεσμα στην οθόνη του υπολογιστή. Για τον παραπάνω λόγο το πρόγραμμα Alice έγινε ιδιαίτερα γνωστό και αποδεκτό σε ευρύ προγραμματιστικό κοινό. Κυρίως όμως είχε απήχηση σε φοιτητές και παιδιά σχολείου που πλέον έβρισκαν ευχάριστο τον προγραμματισμό και οι μαθητές είχαν την ευκαιρία να κατανοήσουν θεμελιώδεις έννοιες προγραμματισμού στο πλαίσιο δημιουργίας ταινιών κινουμένων σχεδίων και απλών βιντεοπαιχνιδιών.

Η δενδροειδής διάταξη περιέχει μια λίστα αντικειμένων που περιέχονται στον τρέχοντα κόσμο του Alice και επιτρέπει σους μαθητές να επιλέξουν τα αντικείμενα που θέλουν να κινήσουν.

Το καρέ επιτρέπει στους μαθητές να βλέπουν τα αποτελέσματα του προγράμματος που έχουν γράψει για τα αντικείμενα στο τρισδιάστατο κόσμο.

Οι μαθητές χρησιμοποιούν τα γεγονότα (events) για να συνδυάσουν μεθόδους, επιλέγοντας με το ποντίκι. Η βασική περιοχή περιέχει λεπτομέρειες όπως μεθόδους /εντολές, λειτουργίες, στοιχεία (ιδιότητες) για το επιλεγμένο αντικείμενο. Ο συντάκτης του κώδικα είναι η περιοχή όπου δημιουργούνται τα προγράμματα με τη μέθοδο «σύρε και άφησε».



Στο προγραμματιστικό περιβάλλον του ALICE, ένα πρόγραμμα δημιουργείται στον συντάκτη κώδικα (editor), επιλέγοντας ο χρήστης, αρχικά το αντικείμενο που θέλει να προγραμματίσει και στη συνέχεια «σύρει και αφήνει» τις μεθόδους στον συντάκτη. Με αυτό τον τρόπο αποφεύγονται τα συντακτικά λάθη. Μπορεί να συμβούν λογικά λάθη αλλά εφόσον ο χρήστης μπορεί να εκτελέσει το πρόγραμμα, θα εντοπίσει οπτικά το λάθος και θα το διορθώσει εύκολα μετά από δοκιμές. Έτσι προσαρμόζεται εύκολα ο κώδικας και εντοπίζονται τα αποτελέσματα των αλλαγών. Επιπλέον η τρισδιάστατη απεικόνιση παρέχει στο χρήστη μια πραγματική αίσθηση των «αντικειμένων», των μεθόδων και των προγραμματιστικών δομών, όπως π.χ. τη δομή επιλογής. Κατά συνέπεια προσφέρει μια εύκολη πρόσβαση του μαθητή σε κάποια γλώσσα αντικειμενοστραφούς προγραμματισμού.

Το ALICE είναι ένα εργαλείο με προσέγγιση **"object-first"**, δηλαδή ο μαθητής διδάσκεται και κατανοεί την έννοια του αντικειμένου από την αρχή των μαθημάτων. Η χρήση του εργαλείου επιτρέπει στους μαθητές να δημιουργούν ταινίες και παιχνίδια, όπου η έννοια του «αντικειμένου», γίνεται αντιληπτή μέσω της οπτικοποιησής του στην οθόνη ευρισκόμενο στον τρισδιάστατο μικρόκοσμο.

2.3 - Συνοπτική Περιγραφή του εργαλείου «ALICE»

2.3.1 - Εκδόσεις του ALICE

Αυτή τη στιγμή, υπάρχουν δύο εκδόσεις του ALICE που είναι ενεργές: η παλιά 2.4.1 και η νεότερη 3.1. Ο λόγος είναι ότι η παλιά έκδοση έχει αρκετές διαφορές και δεν υπάρχει συμβατότητα των παλαιών αρχείων με τη νέα έκδοση. Οι δύο εκδόσεις παρουσιάζουν μια ελαφρώς διαφορετική προσέγγιση στον αντικειμενοστραφή προγραμματισμό και κρίθηκε ότι αξίζει να συντηρούνται και οι δύο. Κατά κάποιο τρόπο η έκδοση 2.4.1 λόγω της απλότητάς της προορίζεται για τις μικρότερες ηλικίες. Στην έκδοση 3.1 θα βρείτε και τους χαρακτήρες από το Sims2 της Electronic Arts. Μάλιστα, στις πρόσφατες εκδόσεις (2.4 και 3.1) πρόσφατα προστέθηκαν και νέοι φίλοι: ο γνωστός Garfield.

2.3.2 - Γιατί καλείται ALICE;

Πρώτα απ ' όλα, το ALICE δεν είναι αρκτικόλεξο. Δεν είναι A.L.I.C.E.. Η ομάδα ονόμασε το σύστημα ALICE στην μνήμη του Charles Lutwidge Dodson, έναν Άγγλο μαθηματικό που υπέγραφε με το όνομα Lewis Carroll. Ο Carroll έγραψε το έργο "Οι περιπέτειες της ALICE στην χώρα των θαυμάτων". Όπως οι άνθρωποι που δημιούργησαν το ALICE έτσι και ο Lewis Carroll ήταν ικανός να επιλύσει πολύπλοκα μαθηματικά, αλλά γνώριζε ότι το σημαντικότερο ήταν να κάνει τα πράγματα απλά και συναρπαστικά σε ένα μαθητή.

Με τον ίδιο τρόπο με τον οποίο, η ALICE ήταν διστακτική όταν αρχικά πέρασε μέσα από το καθρέφτη, έτσι και εσείς μπορεί να έχετε κάποιες αμφιβολίες σχετικά με την εκμάθηση ενός προγράμματος. Παρακαλώ προχωρήστε και σας υποσχόμαστε ότι η εκμάθηση του προγραμματισμού θα είναι ευκολότερη απ' ότι νομίζετε.

2.3.3 - Το Alice

Ο προγραμματισμός, με το σύστημα Alice, σημαίνει ότι θα δημιουργήσετε πραγματικούς κόσμους με διάφορα αντικείμενα στον υπολογιστή σας. Μετά θα γράψετε προγράμματα για να κατευθύνετε τα δικά σας κινούμενα σχέδια. Θα ξεκινήσουμε με μια περίληψη του λογισμικού Alice και με μια επισκόπηση του περιβάλλοντος εργασίας για να σας βοηθήσουμε να ξεκινήσετε.

Αρχικά: Ένας εικονικός κόσμος

Τα video games και οι προσομοιώσεις μπορούν να είναι είτε δυο, είτε τριών διαστάσεων (2D, 3D). Μπορεί να έχετε χρησιμοποιήσει ένα 2D προσομοιωτή γραφικών σε κάποιο μάθημα οδήγησης. Ένα μέρος της εκπαίδευσης των πιλότων είναι η χρήση προσομοιωτών πτήσης. Το πλεονέκτημα των προσομοιώσεων είναι εμφανές - όταν ένας αρχάριος πιλότος καταστρέψει ένα πολεμικό αεροπλάνο, ούτε ο πιλότος άλλα ούτε και το αεροπλάνο δεν βρίσκεται σε κίνδυνο. Ένα video game ή μια προσομοίωση υλοποιημένα σε 3D καλείται εικονικός κόσμος. **Χρησιμοποιώντας έναν εικονικό κόσμο, ο προσομοιωτής γίνεται ρεαλιστικότερος και αποτελεσματικότερος.** Τα αντικείμενα βρίσκονται σε ένα 3D εικονικό κόσμο που έχει πλάτος, μήκος και βάθος, οπότε η κάμερα μπορεί να κινηματογραφήσει από διάφορες οπτικές γωνίες, πράγμα το οποίο δίνει μια αίσθηση πραγματικότητας στα αντικείμενα.

Τρεις και έξι κατευθύνσεις

Τα αντικείμενα στο κόσμο του Alice είναι τριών διαστάσεων. Κάθε αντικείμενο έχει μήκος(ή ύψος), πλάτος και βάθος. Το ύψος μετριέται κατά μήκος μιας νοητής γραμμής που είναι κατακόρυφη από πάνω προς τα κάτω, το πλάτος κατά μήκος μιας νοητής γραμμής που είναι οριζόντια από αριστερά προς τα δεξιά και το βάθος κατά μήκος μιας νοητής γραμμής από μπροστά προς τα πίσω.

Όσον αφορά αυτές τις τρεις διαστάσεις, το αντικείμενο γνωρίζει ποιος δρόμος είναι πάνω ή κάτω σε σχέση με τον εαυτό του. Επίσης το αντικείμενο γνωρίζει το αριστερά και το δεξιά, το μπροστά και το πίσω. Αυτό ισοδυναμεί με **έξι πιθανές κατευθύνσεις στις οποίες μπορεί να κινηθεί ένα αντικείμενο.** Το αντικείμενο έχει έξι βαθμούς ελευθερίας για να κινηθεί σε ένα κόσμο. Είναι σημαντικό να προσέξετε ότι οι κατευθύνσεις είναι προς τα δεξιά και αριστερά σε σχέση με το αντικείμενο, όχι με την θέση της κάμερας. **Υπάρχουν έξι βαθμοί ελευθερίας** (πιθανές κατευθύνσεις κίνησης) **όσον αφορά τον προσανατολισμό του αντικειμένου.** Όταν κάνετε κλικ με το ποντίκι σε ένα αντικείμενο, εμφανίζεται ένα κίτρινο περίβλημα. Το κίτρινο αυτό περίβλημα τονίζει το αντικείμενο.

2.3.4 - Οδηγός χρήσης /εγκατάστασης

Η δημιουργία ενός έργου στο ALICE περιλαμβάνει δύο στάδια: το στήσιμο της σκηνής και τη δημιουργία του κώδικα. Στο πρώτο στάδιο οι μαθητές μπορούν να επιλέξουν από μία ευρεία γκάμα τρισδιάστατων χαρακτήρων αυτούς που θα απαρτίζουν τον εικονικό τους κόσμο και να τους τοποθετήσουν στον χώρο (ουσιαστικά δημιουργούν στιγμιότυπα κλάσεων). Μην ξεχνάτε ότι μιλάμε για τρισδιάστατους κόσμους και η τοποθέτηση των χαρακτήρων/αντικειμένων μπορεί να σας δυσκολέψει λίγο (μπορείτε να μετακινείτε, να περιστρέφετε και να αλλάζετε το μέγεθος των χαρακτήρων-αντικειμένων). Στη συνέχεια ο προγραμματισμός της συμπεριφοράς των αντικειμένων-χαρακτήρων γίνεται ορίζοντας μεθόδους και συναρτήσεις και καλώντας τις μεθόδους αυτές στους κατάλληλους event listeners. Μην σας τρομάζουν όλα αυτά: οι μαθητές απλώς πρέπει να σύρουν πλακίδια τύπου do in order, do together, count, while, if else, each in, variable, assign κ.λ.π. και πλακίδια των έτοιμων μεθόδων των αντικειμένων (move, turn, roll, resize κ.λ.π.) και να ορίσουν τις τιμές των πλακιδίων, όλα μέσω του γραφικού περιβάλλοντος, χωρίς να χρειάζεται να θυμούνται τίποτα από μνήμης. Τέλος, οι μαθητές μπορούν να τρέξουν το έργο τους και να καταγράψουν την εκτέλεσή του σε βίντεο έτσι ώστε να το ανεβάσουν στο internet (στο ALICE 3 υπάρχει ενσωματωμένη επιλογή για εξαγωγή σε μορφή βίντεο και upload στο youtube).

Για να κατεβάσετε το ALICE :

http://www.ALICE.org/index.php?page=downloads/download_ALICE3.1

Για το ALICE 3 θα χρειαστείτε και το Java Development Kit. Για την εγκατάσταση και τα άλλα βασικά μπορείτε να βρείτε βοήθεια στο wiki του ALICE. Χρήσιμος είναι και ο οδηγός εκκίνησης με το ALICE 3.1. Αν χρειάζεστε έναν εκτενέστερο οδηγό, με βίντεο, διαφάνειες και παραδείγματα μπορείτε να δείτε το υλικό εισαγωγής στο ALICE3. Δημιουργήθηκε, επίσης, ένα μάθημα γνωριμίας με το ALICE 2.4. Και μη φοβάστε τον προγραμματισμό: δεν υπάρχει ντάμα κούπα να σας πάρει το κεφάλι.

Συνήθως, όλα τα λογισμικά συνοδεύονται με ένα εγχειρίδιο χρήσης (tutorial), στο οποίο παρουσιάζεται ο τρόπος χειρισμού του λογισμικού και οι βασικές λειτουργίες του. Τα περισσότερα εγχειρίδια χρήσης είναι σε μορφή κειμένου και ακολουθούν μια σειρά από βήματα που ο χρήστης πρέπει να ακολουθήσει.

Το εκπαιδευτικό λογισμικό ALICE χρησιμοποιεί ένα διαφορετικό είδους παρουσίασης tutorial/εγχειρίδιο χρήσης από τα συνηθισμένα, παραδοσιακά εγχειρίδια χρήσης που είναι σε μορφή κειμένου. Χρησιμοποιεί την τεχνική των Stencils, μια εναλλακτική μέθοδος παρουσίασης tutorial με αλληλεπίδραση με το χρήστη. Η τεχνική Stencils χρησιμοποιεί ένα Tutorial που αναφέρεται σε πληροφορίες που παρουσιάζονται σε διδακτική μορφή μέσα από παραδείγματα και με αλληλεπίδραση με το χρήστη, χρησιμοποιώντας χρωματιστά σημειώματα, τα οποία κατευθύνουν την προσοχή του χρήστη και υποδεικνύουν προς το σωστό χειρισμό του λογισμικού.

ΜΕΡΟΣ 3ο

Ανάπτυξη εκπαιδευτικών σεναρίων

3.1 - Εφαρμογές – μαθήματα αντικειμενοστρεφή εννοιών

Μέσα από την δημιουργία εφαρμογών σεναρίων μαθημάτων στο τρισδιάστατο περιβάλλον ALICE μπορούν να γίνουν άμεσα κατανοητές έννοιες του αντικειμενοστραφή προγραμματισμού με ευχάριστο τρόπο. Μέσα από τις κινήσεις των ηρώων, στο συγκεκριμένο εκπαιδευτικό υλικό βασισμένο στην πλοκή της Αλίκης στην χώρα των θαυμάτων (Alice in wonderland) και στην μεταφορά του στο δικό μας **ALICE IN JAVALAND**, μπορούν να γίνουν κατανοητές βασικές έννοιες της JAVA όπως η κληρονομικότητα, ο πολυμορφισμός κτλπ. Ύστερα ο χρήστης μπορεί ακολουθεί μια σειρά βημάτων και δημιουργεί μια ολοκληρωμένη εφαρμογή χρησιμοποιώντας τις βασικότερες δομές που υποστηρίζει το περιβάλλον του ALICE.

Σε ένα πλάνο διδασκαλίας, θα μπορούσε να χωριστεί και σε αυτοτελή κομμάτια αλλά και να διδαχτεί ολόκληρο και αυτούσιο σαν ένα πλάνο μαθήματος που σε καθοδηγεί μέσα από τις σκηνές και τις αναδράσεις, αφήνοντας μετά τον χρήστη να δημιουργήσει το δικό του κόσμο και να κάνει χρήση των δικών του εντολών με σκοπό την δημιουργία ενός κόσμου και δυναμικών κινήσεων των αντικειμένων.

Δεν είναι απαραίτητο να δοθεί στους χρήστες/μαθητές κάποιο διδακτικό πλάνο διότι μέσα από τις διαδράσεις με τον κόσμο που δημιουργήσαμε στο ALICE IN JAVALAND δίνονται σε όλα τα σημεία ευκαιρίες αλληλεπιδράσεις και επεξηγήσεις με έμμεσο ή άμεσο τρόπο ώστε να επιτευχθεί η συνέχεια της δράσης στον εικονικό διαμορφωμένο κόσμο.

Το σχέδιο μαθήματος αφορά τους εκπαιδευτικούς στόχους που επιδιώκονται να επιτευχθούν κατά την ολοκλήρωση της διδασκαλίας και καθοδηγεί τον καθηγητή ώστε να είναι εντός των στόχων. Οι στόχοι επικεντρώνονται στο ότι μετά το πέρας του ALICE IN JAVALAND και την πρώτη επαφή με έννοιες προγραμματισμού, ο χρήστης/μαθητής να μπορεί να δημιουργήσει κάτι δικό του, κάνοντας εφαρμογή βασικές έννοιες αλλά κυρίως να θέλει να ασχοληθεί μ΄αυτό και να παραμείνει το ενδιαφέρον του ζωντανό.

Στόχος είναι, με το ALICE IN JAVALAND, να γίνεται μια εισαγωγή στους μαθητές/χρήστες για το τι μπορούν να κάνουν χρησιμοποιώντας αυτό το περιβάλλον ανάπτυξης αντικειμενοστραφή προγραμματισμού. Και να δίνεται η ευκαιρία για παρακολούθηση και μάθηση βασικών εννοιών με απότερο τελικό στόχο, μετά το πέρας της παρουσίασης να γίνεται δημιουργία και συνέχεια της χρήσης αντικειμενοστραφή εννοιών από τους χρήστες/μαθητές μέσα από το ALICE.

3.2 - Σχεδιασμός και υλοποίηση προγράμματος

Το Alice in Javaland είναι σε έκδοση Alice 3.1 που είναι η τελευταία μέχρι στιγμής. **Η δημιουργία ενός έργου στο Alice περιλαμβάνει δύο στάδια: το στήσιμο της σκηνής και τη δημιουργία του κώδικα.** Στο πρώτο στάδιο γίνεται η επιλογή από μία ευρεία γκάμα τρισδιάστατων χαρακτήρων αυτούς που θα απαρτίζουν τον εικονικό τους κόσμο και να τους τοποθετήσουν στον χώρο (ουσιαστικά δημιουργούν στιγμιότυπα κλάσεων).

Στη συνέχεια, ο προγραμματισμός της συμπεριφοράς των αντικειμένων-χαρακτήρων γίνεται ορίζοντας μεθόδους και συναρτήσεις και καλώντας τις μεθόδους αυτές στους κατάλληλους event listeners. Σέρνοντας πλακίδια τύπου do in order, do together, count, while, if else, each in, variable, assign κ.λ.π. και πλακίδια των έτοιμων μεθόδων των αντικειμένων (move, turn, roll, resize κ.λ.π.) και ορίζοντας τις τιμές των πλακιδίων.

Στο προγραμματιστικό περιβάλλον του Alice, ένα πρόγραμμα δημιουργείται στον συντάκτη κώδικα (editor), επιλέγοντας ο χρήστης αρχικά το αντικείμενο που θέλει να προγραμματίσει και στη συνέχεια «σύρει και αφήνει» τις μεθόδους στον συντάκτη. Με αυτό τον τρόπο αποφεύγονται τα συντακτικά λάθη. Μπορεί να συμβούν λογικά λάθη αλλά εφόσον ο χρήστης μπορεί να εκτελέσει το πρόγραμμα, θα εντοπίσει οπτικά το λάθος και θα το διορθώσει εύκολα μετά από δοκιμές. Έτσι προσαρμόζεται εύκολα ο κώδικας και εντοπίζονται τα αποτελέσματα αλλαγών. Επιπλέον **η τρισδιάστατη απεικόνιση παρέχει στο χρήστη μια πραγματική αίσθηση των «αντικειμένων», των μεθόδων και των προγραμματιστικών δομών**, όπως π.χ. τη δομή επιλογής. Κατά συνέπεια προσφέρει μια εύκολη πρόσβαση του μαθητή σε κάποια γλώσσα αντικειμενοστραφούς προγραμματισμού .

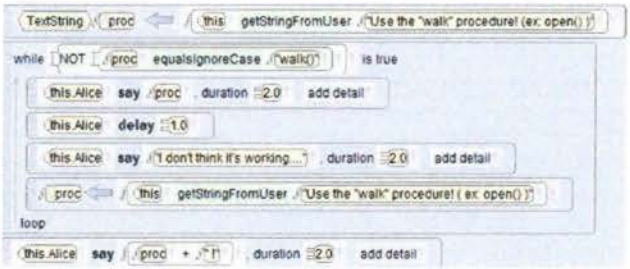
Ο χρήστης - μαθητής μπορεί να δημιουργεί ένα πρόγραμμα με τη μέθοδο «σύρε και άφησε» τις εντολές/εκφράσεις (οι οποίες δίνονται έτοιμες) με το ποντίκι στο χώρο που δημιουργείται ο κώδικας, αντί να γράφει τον κώδικα λέξη προς λέξη. Ακόμη, όποτε επιθυμεί ο χρήστης, μπορεί να δει να εκτελείται το πρόγραμμα, που έχει γράψει, σε ένα παράθυρο σε κινούμενη τρισδιάστατη απεικόνιση του μικρόκοσμου.

3.3 - Σενάρια μαθημάτων και σκηνές έργου

Στο πρώτο σενάριο του μαθήματος στόχος είναι ο μαθητής/χρήστης να μάθει να εκτελεί εντολές καθώς και τη for loop. Πρώτα δημιουργήσαμε την εισαγωγή με διάφορες κινήσεις της κάμερας και των χαρακτήρων με τις εντολές this.camera **move** *direction* ??? , *amount* ???, this.camera **turn** *direction* ??? , *amount* ??? this.camera **pointAt** *target* ??? , κτλ.

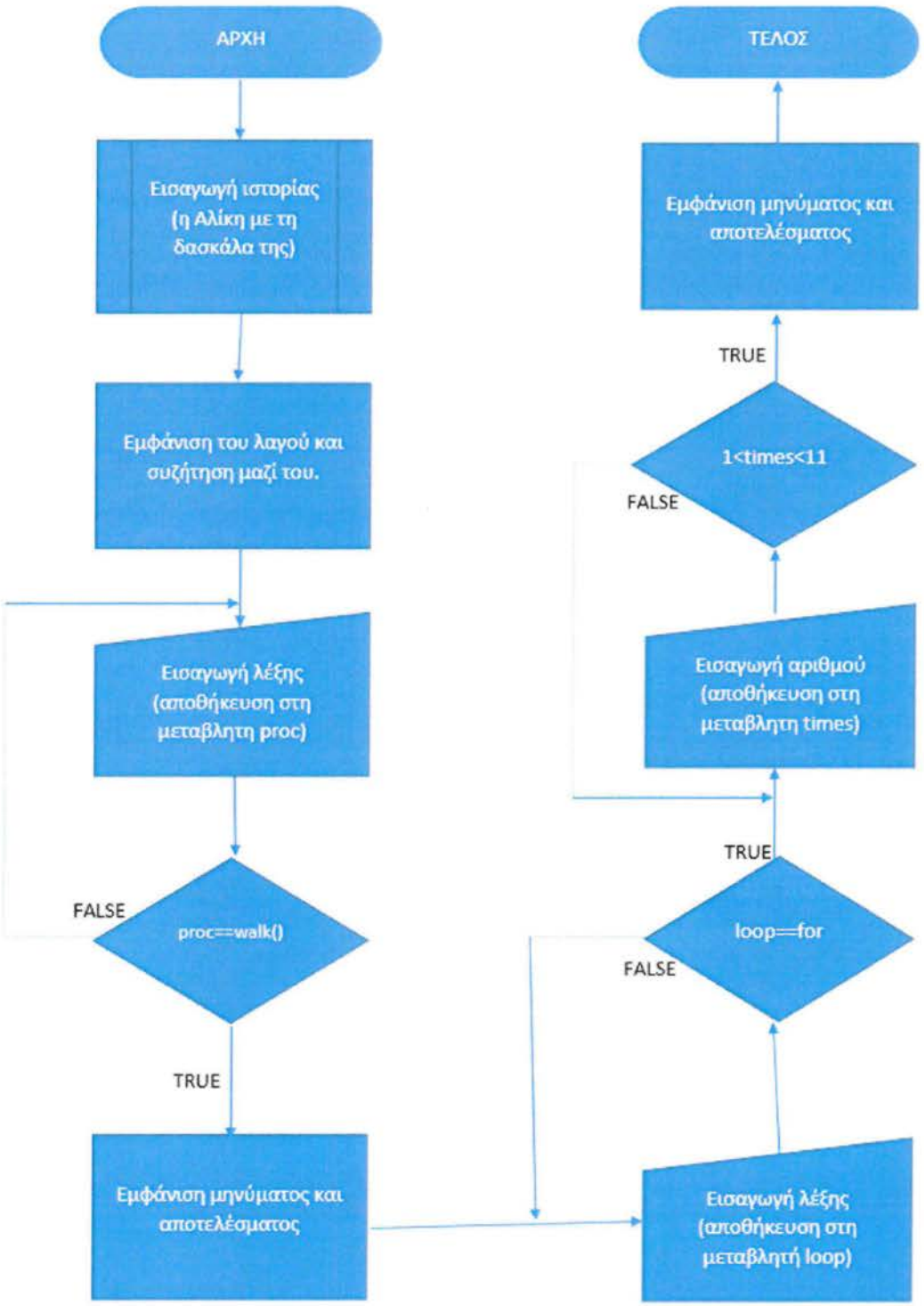
Έπειτα μετακινώντας την πρωταγωνίστρια και την κάμερα μεταφερόμαστε στο επόμενο σκηνικό. Εμφανίζοντας το λαγό και δημιουργώντας έναν διάλογο - με εντολές όπως this.Alice **say** *text* ???, this.Alice **turnToFace** *target* ??? κτλ. – οδηγούμε το χρήστη στην πρώτη του ανάδραση με το πρόγραμμα, δηλαδή εισαγωγή λέξης απο το πληκτρολόγιο. Για να ζητήσουμε να εισάγει ο χρήστης μια λέξη χρησιμοποιούμε την εντολή this getStringFromUser "Use the "walk" procedure! (ex: open())".

Με το παρακάτω block εντολών ελέγχεται το string που εισάγει ο χρήστης. Χρησιμοποιώντας την while loop επιτυγχάνουμε να ελέγχεται κάθε προσπάθεια που κάνει ο χρήστης και να εμφανίζει αντίστοιχο μήνυμα λάθους.

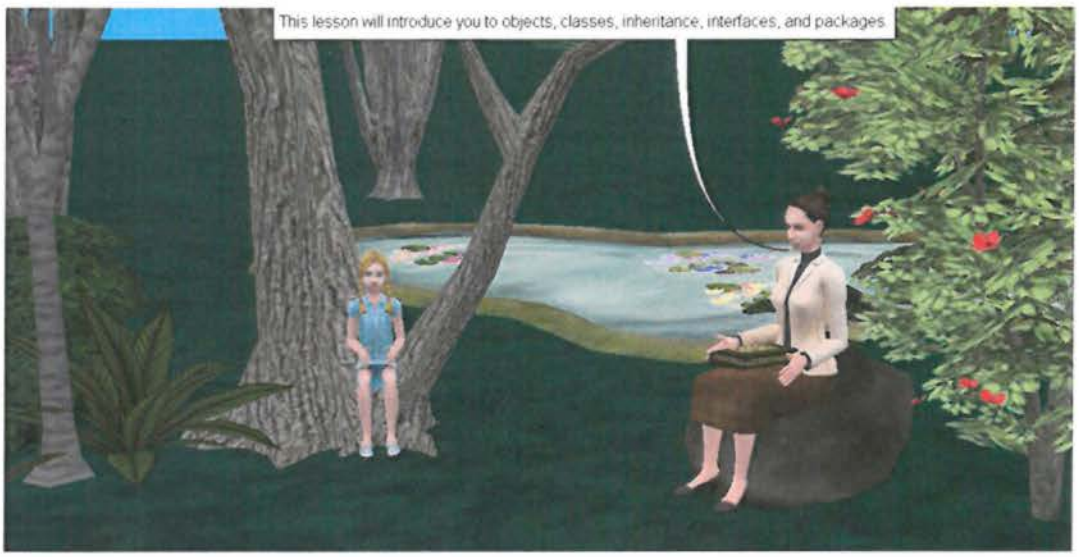


Με τον ίδιο τρόπο (με while loop) ελέγχουμε την επόμενη εισαγωγή λέξης και αριθμού.

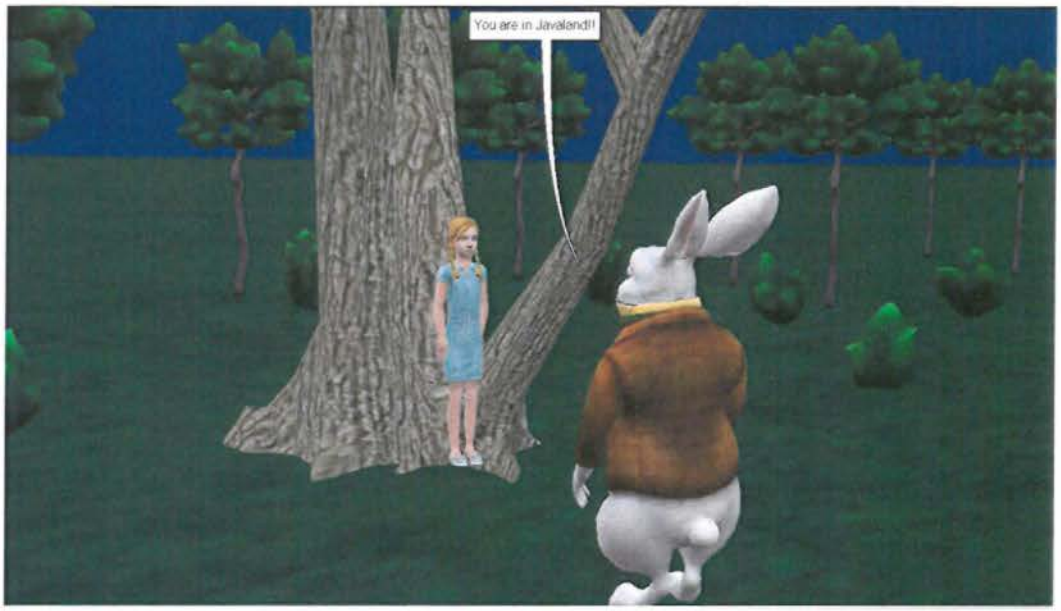
Παρακάτω εμφανίζεται το λογικό διάγραμμα καθώς και το αποτέλεσμα του πρώτου προγράμματος.

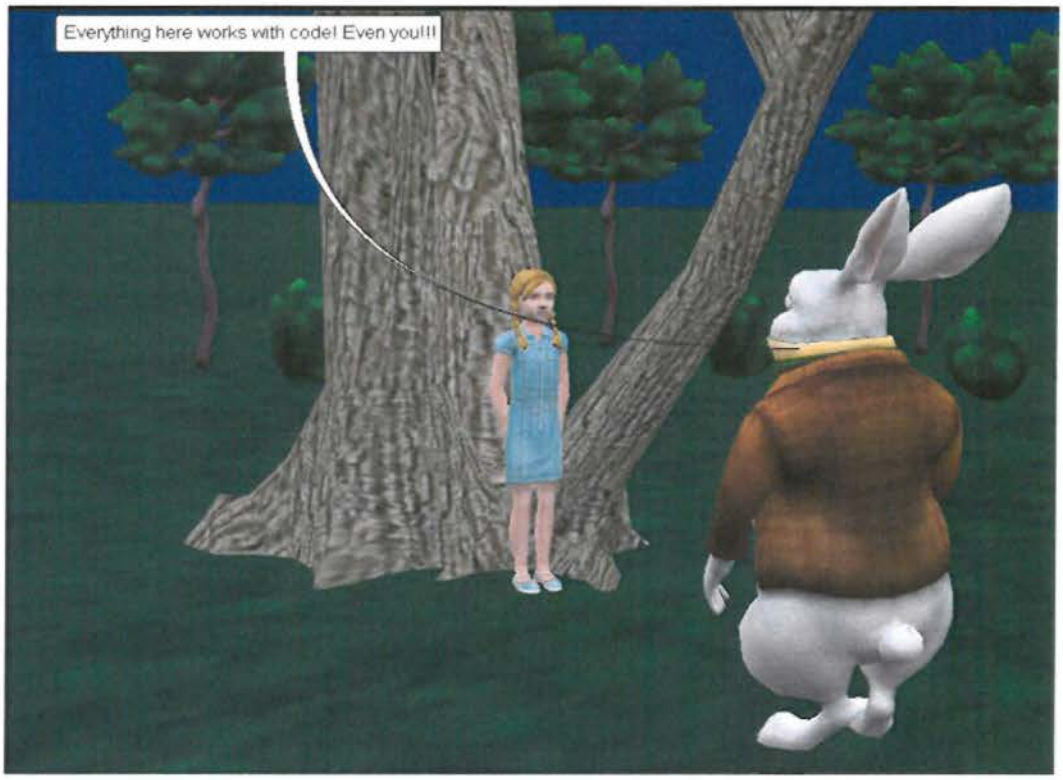

ΑΡΧΗ
Εισαγωγή ιστορίας (η Αλίκη με τη δασκάλα της)
Εμφάνιση του λαγού και συζήτηση μαζί του.
Εισαγωγή λέξης (αποθήκευση στη μεταβλητη proc)
proc==walk()
FALSE
TRUE
Εμφάνιση μηνύματος και αποτελέσματος
Εισαγωγή λέξης (αποθήκευση στη μεταβλητή loop)
loop==for
FALSE
TRUE
Εισαγωγή αριθμού (αποθήκευση στη μεταβλητη times)
1<times<11
FALSE
TRUE
Εμφάνιση μηνύματος και αποτελέσματος
ΤΕΛΟΣ

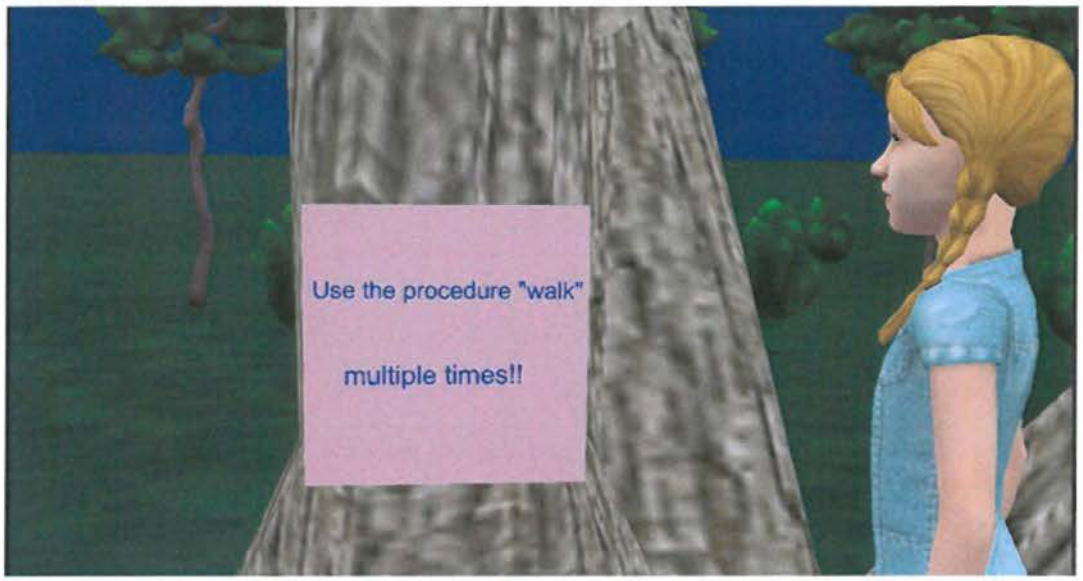
«Στην αρχή εμφανίζεται η Alice στον εικονικό μας κόσμο σαν μαθήτρια που της διατυπώνει η δασκάλα της πως θα αρχίσουν μάθημα με τον κλασσικό τρόπο διδασκαλίας. Η δυσαρέσκια της είναι έντονη και τότε εκείνη μπαίνει στην διαδικασία να ονειρευτεί τον ιδανικό τρόπο να μάθει όλες αυτές τις έννοιες. Σ' ένα ενδιαφέρον περιβάλλον με διαφορετικό τρόπο και λίγο δια μαγείας κόσμο, όχι θαυμάτων, αλλά που θα συνεχίσει μόνο με την χρήση JAVA και αντικειμενοστραφή εννοιών.»



«Συνάντηση με το λαγό, που την προτρέπει να κάνει χρήση λίγου κώδικα και όλα θα πάνε καλά! Πρέπει να κινηθεί κάνοντας χρήση μεθόδων αλλά πως;; Η απάντηση δεν αργεί και σύντομα μπορεί και συμπληρώνει το πρώτο της διαδραστικό παράθυρο και ύστερα κάνοντας ***χρήση βρόγχου*** *ώστε να περπατήσει περισότερες φορές.»*



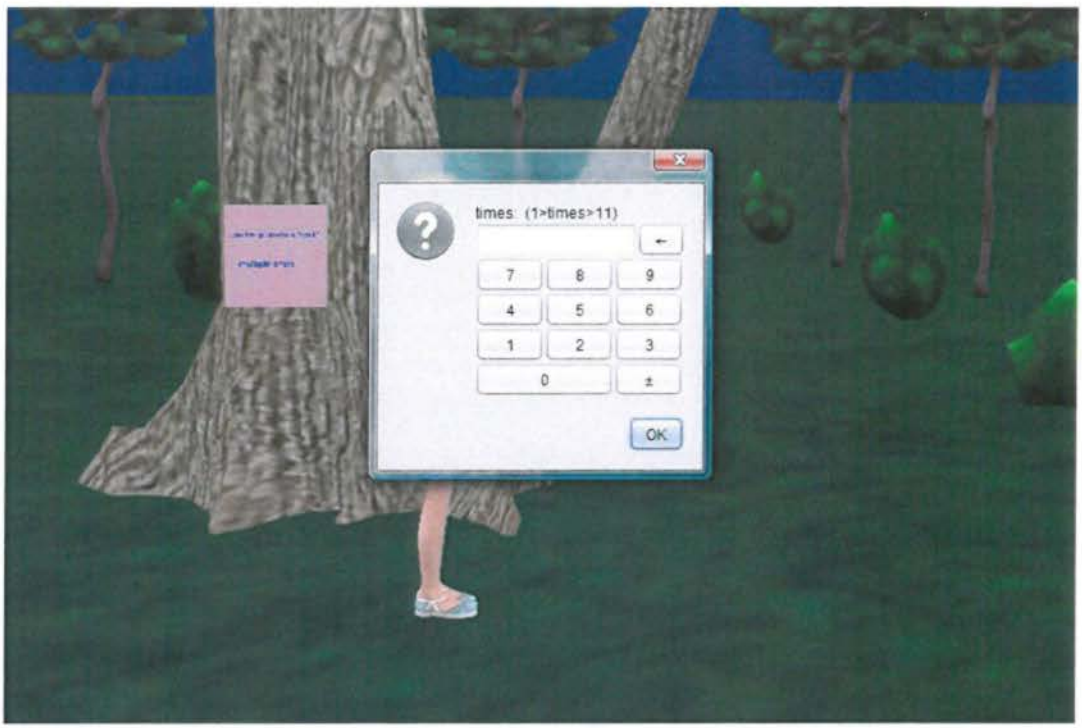



Use the procedure "walk"
multiple times!!


Use the "walk" procedure! (ex: open())
walk()
OK


Now I have to do it many times...


Loop type:
for
OK

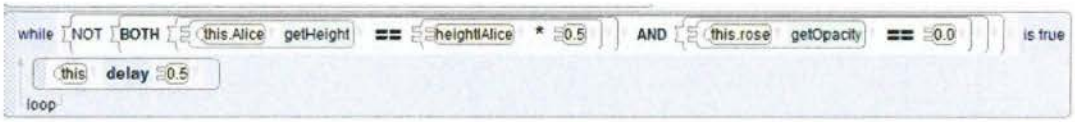

times: (1>times>11)
←
7
8
9
4
5
6
1
2
3
0
±
OK


for(i=0;i<5;i++) { walk(); }

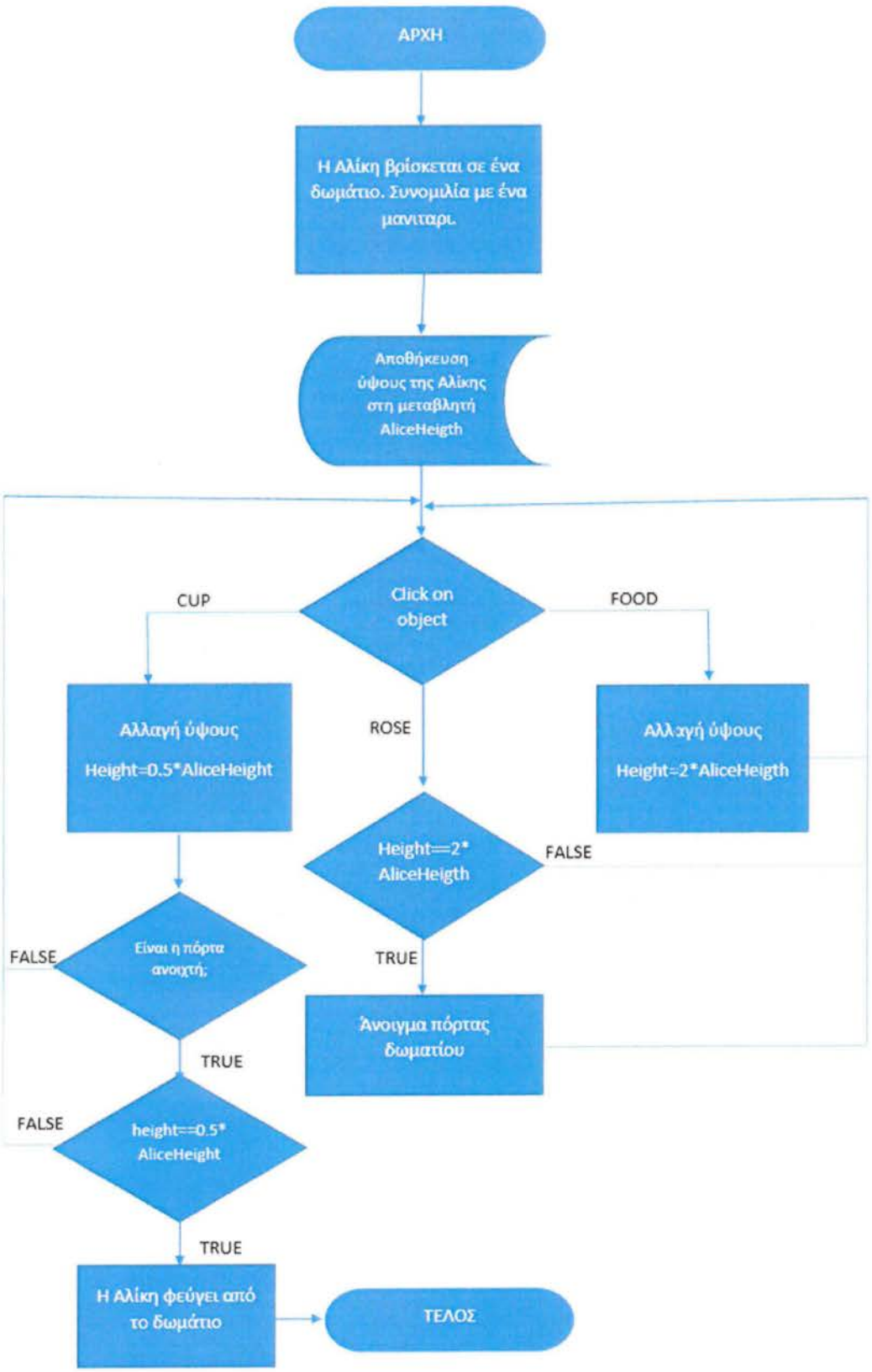
Σε αυτό το σενάριο μαθήματος **ο στόχος είναι ο χρήστης να μάθει τη σημασία της συνθήκης IF**. Μεταφέρουμε την Alice στο δωμάτιο, σύμφωνα με το παραμύθι, με εντολές όπως αναφέραμε πριν. Αφού δημιουργήσαμε ένα διάλογο μεταξύ Alice και μανιταριού η επόμενη ανάδραση με το χρήστη είναι με το click του ποντικιού. Αυτό επιτεύχθηκε με το παρακάτω block κώδικα.

Ο κώδικας αναδράσεων του χρήστη με το ποντίκι γράφεται στο *initializeEventListener* του προγράμματος. Οπότε παρομοίως και τα υπόλοιπα click αναδράσεων δημιουργούνται με τον ίδιο τρόπο στο initializeEventListener, ανάλογα κάθε φορά ποιό θέλουμε να 'ναι το αποτέλεσμα του click.

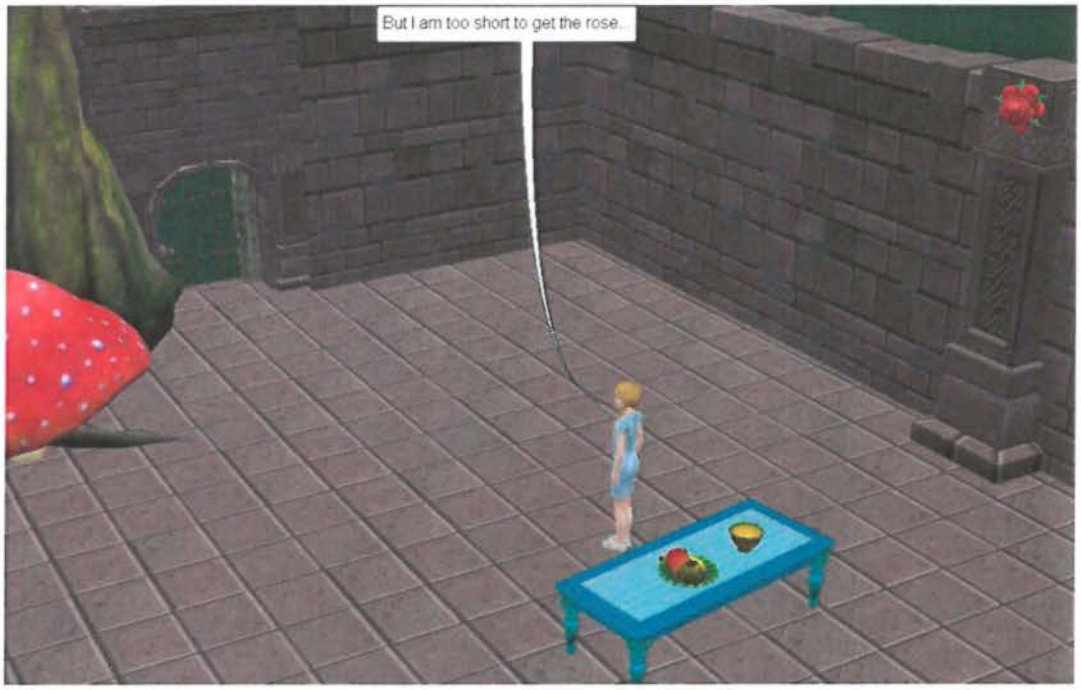
Στο παρακάτω block κώδικα ελέγχονται όλες οι προυποθέσεις ώστε η Alice να αποχωρήσει από το δωμάτιο.

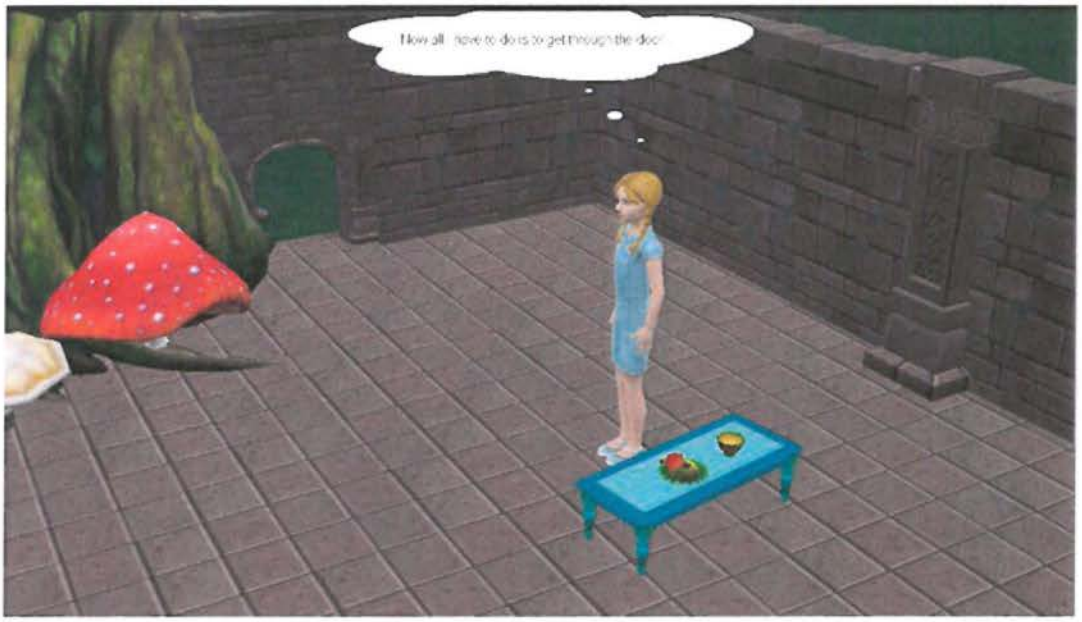


Παρακάτω έχουμε το λογικό διάγραμμα και το αποτέλεσμα αυτού του προγράμματος.

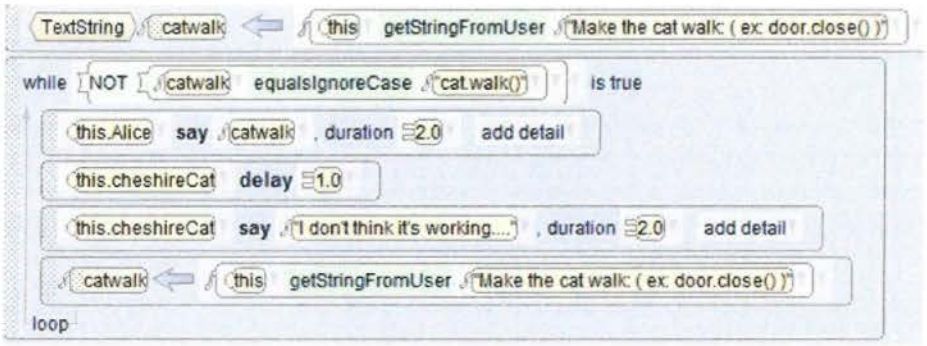

ΑΡΧΗ
Η Αλίκη βρίσκεται σε ένα δωμάτιο. Συνομιλία με ένα μανιταρι.
Αποθήκευση ύψους της Αλίκης στη μεταβλητή AliceHeigth
Click on object
CUP
FOOD
ROSE
Αλλαγή ύψους Height=0.5*AliceHeight
Αλλαγή ύψους Height=2*AliceHeigth
Height==2* AliceHeigth
FALSE
FALSE
Είναι η πόρτα ανοιχτή;
TRUE
Άνοιγμα πόρτας δωματίου
TRUE
FALSE
height==0.5* AliceHeight
TRUE
Η Αλίκη φεύγει από το δωμάτιο
ΤΕΛΟΣ

«Έπειτα θα μάθει μια από τις σημαντικότερες έννοιες στον προγραμματισμό την ***χρήση της IF logic*** *αφού θα βρεθεί στο δωμάτιο οπού θα πρέπει να φάει κάτι αν θέλει να ψηλώσει για να φτάσει το τριαντάφυλλο και να πιει κάτι αν θέλει να μικρύνει για να περάσει από την πορτούλα και να συνεχίσει το ταξίδι της σ'αυτόν το εικονικό κόσμο που έχει να της μάθει πολλά!»*

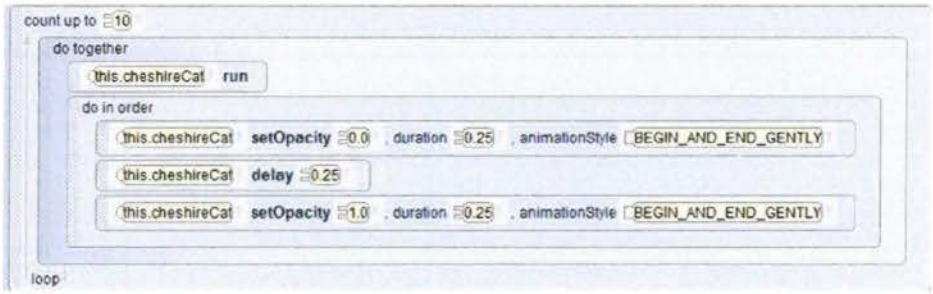



Now all I have to do is to get through the door

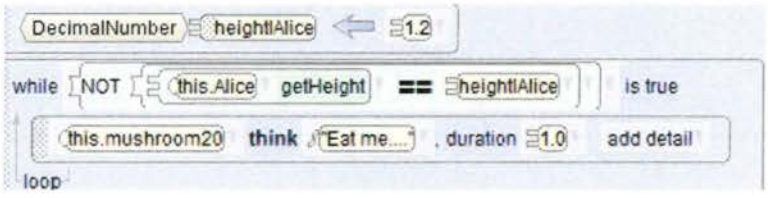
Στο τρίτο σενάριο μαθήματος μεταφέρεται η πρωταγωνίστρια στο τρίτο σκηνικό με εντολές όπως αναφέραμε παραπάνω και το πρόγραμμα ξεκινάει με διάλογο με το γάτο. Έχουμε πάλι σ' αυτό το πρόγραμμα μια εισαγωγή λέξης από το πληκτρολόγιο και ελέγχεται όπως δείχνουμε παρακάτω:



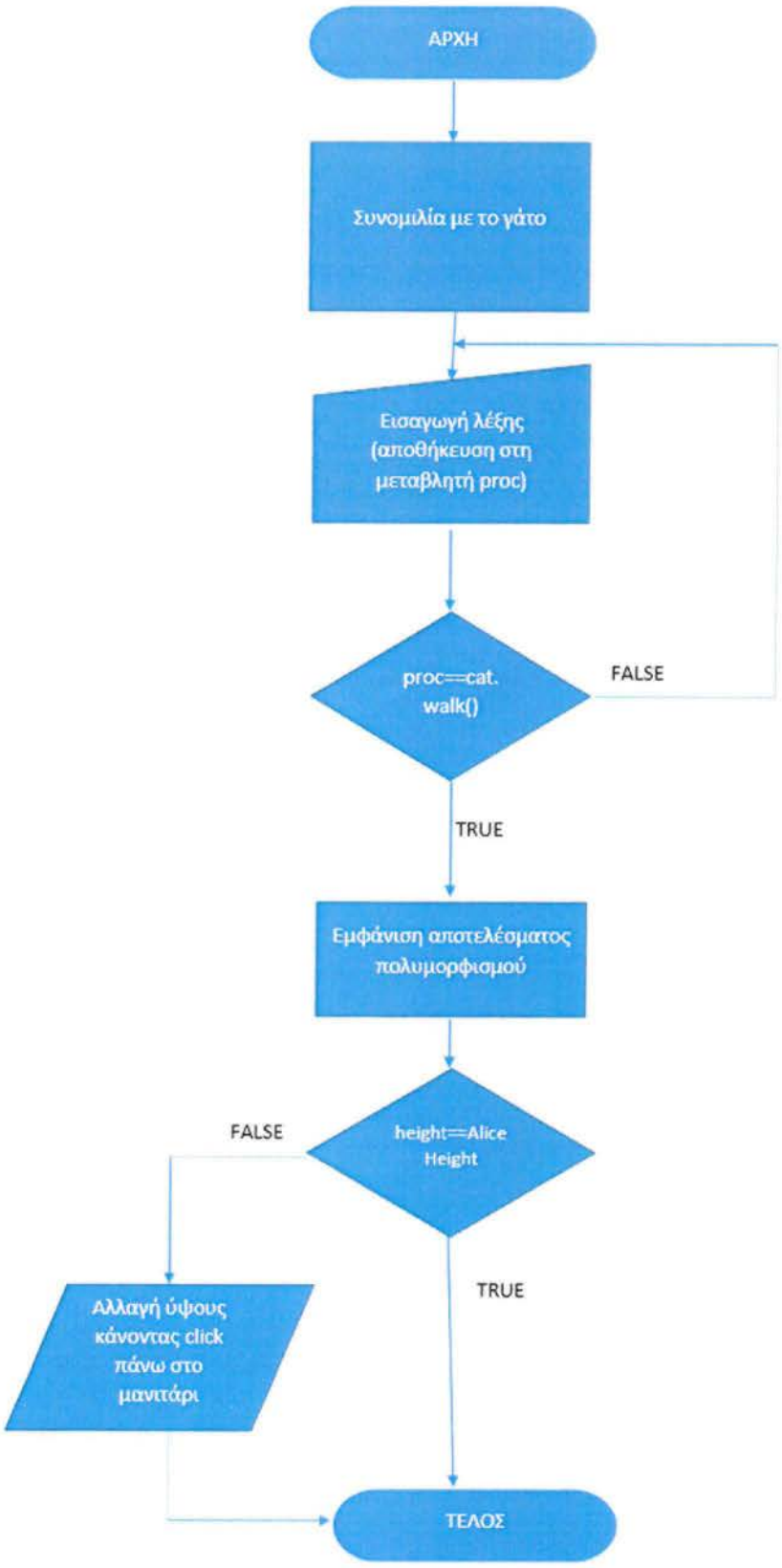
Αφού δοθεί η σωστή λέξη, φαίνεται **το αποτέλεσμα του πολυμορφισμού** δημιουργώντας διαφορετικό τρόπο κίνησης αλλάζοντα το opacity του γάτου σε κάθε του βήμα.



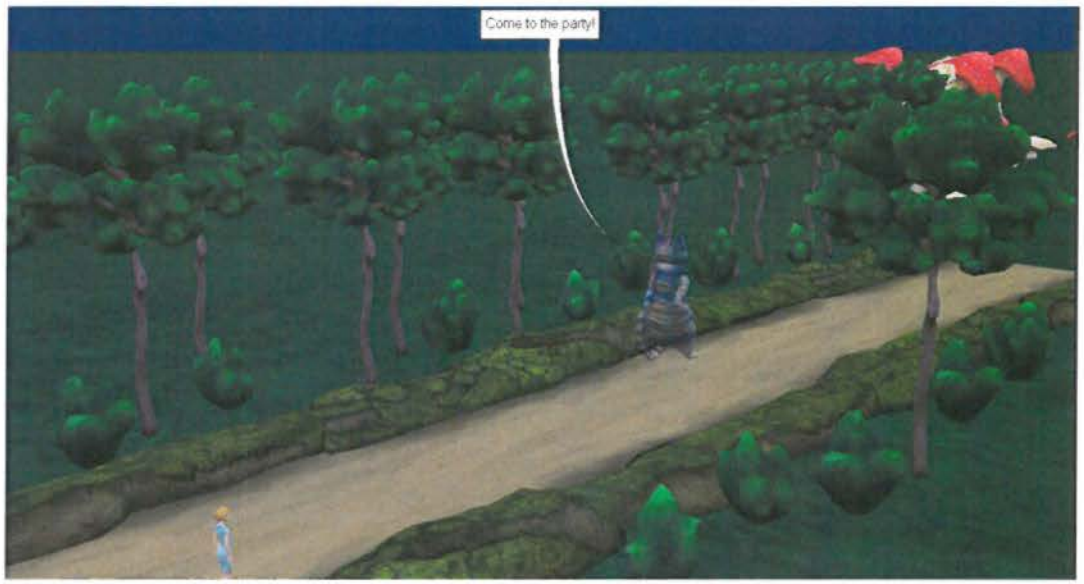
Τέλος , ελέγχεται το ύψος της Alice και με την κατάλληλη ανάδραση από τον χρήστη, δηλαδή click πάνω στο μανιτάρι, έχουμε την αυξομείωση του ύψους της.

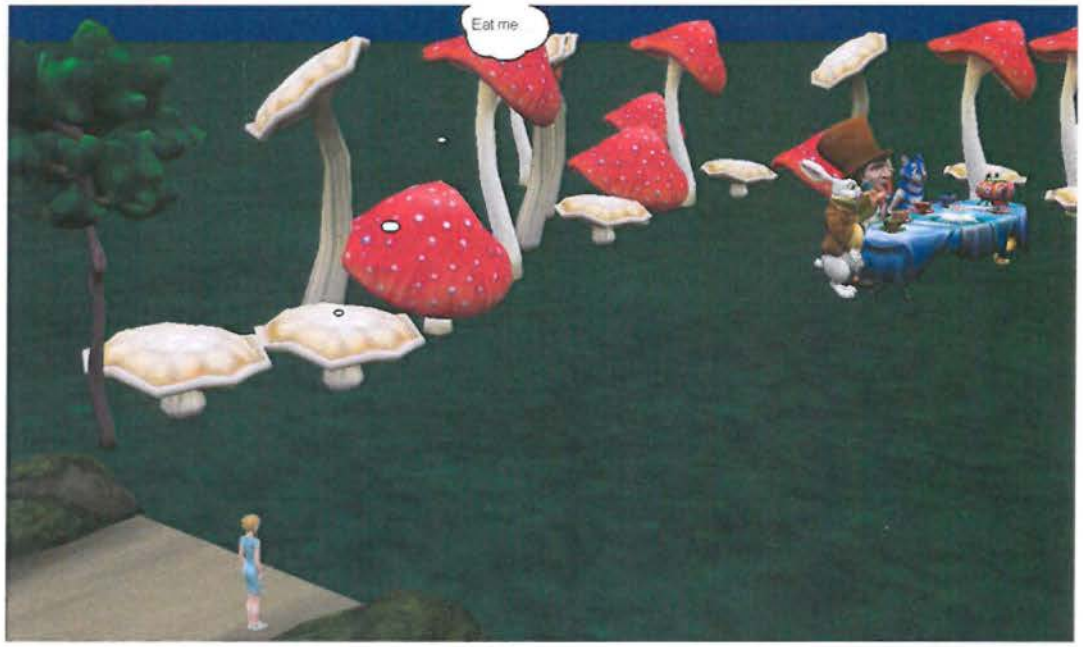


Παρακάτω φαίνεται το λογικό διάγραμμα και το αποτέλεσμα του προγράμματος.

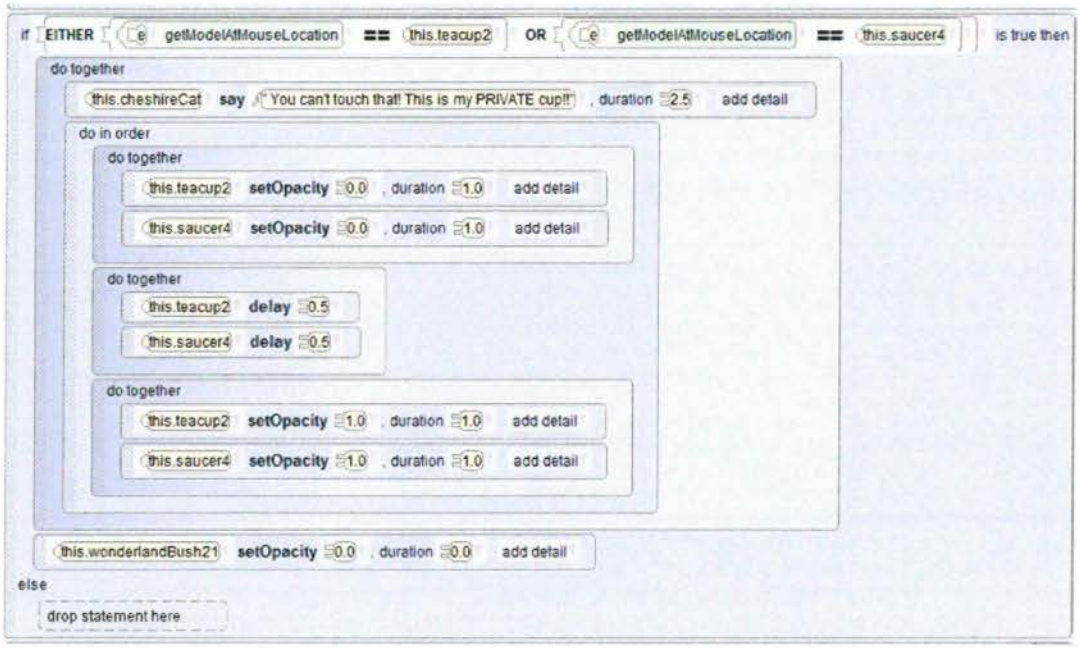

ΑΡΧΗ
Συνομιλία με το γάτο
Εισαγωγή λέξης (αποθήκευση στη μεταβλητή proc)
proc==cat. walk()
FALSE
TRUE
Εμφάνιση αποτελέσματος πολυμορφισμού
height==Alice Height
FALSE
TRUE
Αλλαγή ύψους κάνοντας click πάνω στο μανιτάρι
ΤΕΛΟΣ

*«Σ' αυτήν την διαδρομή θα συναντήσει το γάτο που θα της δώσει να καταλάβει **την έννοια του πολυμορφισμού** και αυτή θα τον βοηθήσει να περπατήσει και αυτός, όπως έμαθε και αυτή στην αρχή του ταξιδιού της στην γνώση των εννοιών!»*


Come to the party!

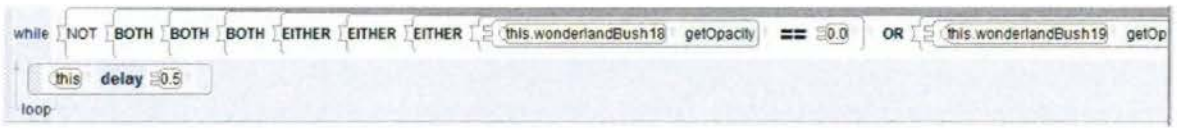

Eat me

Σε αυτό το σενάριο στόχος είναι να δείξουμε την **έννοια της public και private μεταβλητής.** Ξεκινάμε μ' ένα διάλογο μεταξύ των χαρακτήρων στο τραπέζι και της Alice με εντολές όπως προαναφέραμε. Η επόμενη ανάδραση που προγραμματίσαμε είναι click πάνω σ' αντικείμενα που βρίσκονται πάνω στο τραπέζι. Ο κώδικας φαίνεται παρακάτω:

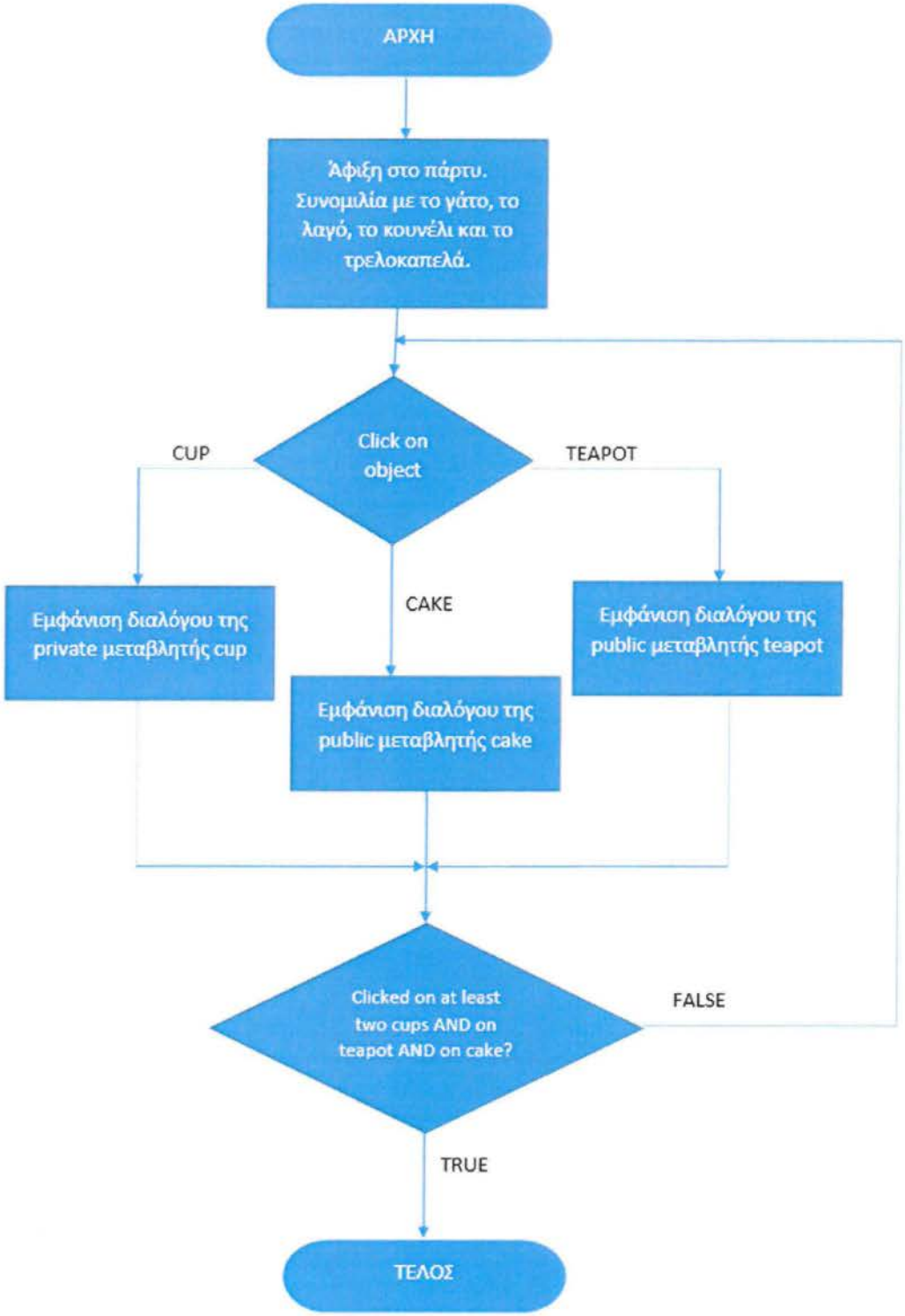


Παρομοίως είναι και η σύνταξη του κώδικα για τα υπόλοιπα αντικείμενα του τραπεζιού.

Τέλος δημιουργήσαμε μια συνθήκη σύμφωνα με το opacity αντικειμένων που επηρεάστηκαν από την ανάδραση του χρήστη, ώστε να μπορέσει ο χρήστης να πάει στο επόμενο σενάριο.



Παρακάτω είναι το λογικό διάγραμμα και το αποτέλεσμα του προγράμματος.


ΑΡΧΗ
Άφιξη στο πάρτυ. Συνομιλία με το γάτο, το λαγό, το κουνέλι και το τρελοκαπελά.
Click on object
CUP
CAKE
TEAPOT
Εμφάνιση διαλόγου της private μεταβλητής cup
Εμφάνιση διαλόγου της public μεταβλητής cake
Εμφάνιση διαλόγου της public μεταβλητής teapot
Clicked on at least two cups AND on teapot AND on cake?
FALSE
TRUE
ΤΕΛΟΣ

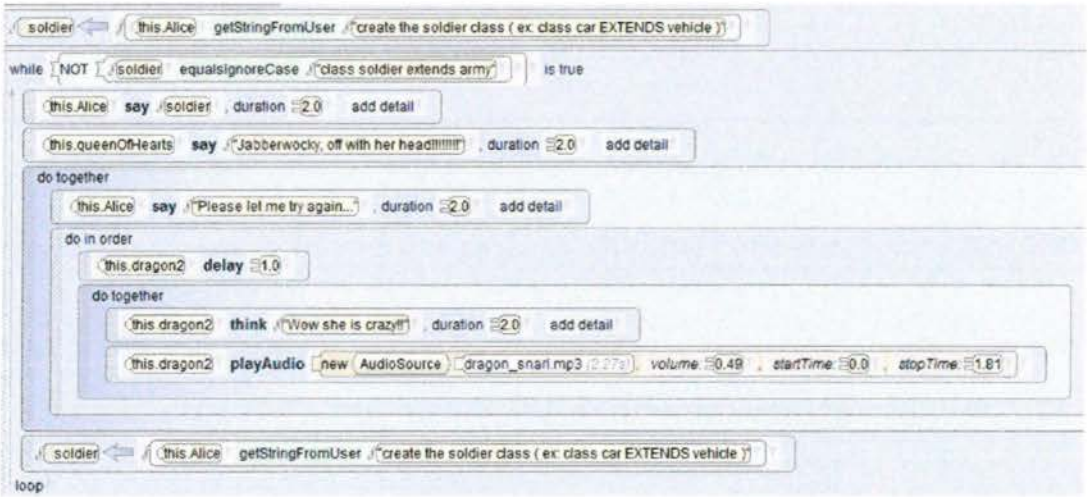
«Όταν βοηθάς ανταμείβεσαι, γι' αυτό και θα πάει σ΄ ένα tea party ώστε να μάθει κάτι πολύ ενδιαφέρον για ***public και private μεταβλητές*** *με την βοήθεια των καλεσμένων.»*

Στο τελευταίο σενάριο στόχος είναι να δείξουμε την **δημιουργία των κλάσεων , την κληρονομικότητα, την δημιουργία αντικειμένων, και την δημιουργία πινάκων.** Αυτά τα μαθαίνει ο χρήστης με την εισαγωγή strings.

Αρχικά ο κώδικας αποτελείται από εντολές που δημιουργούν κινήσεις και διαλόγους.

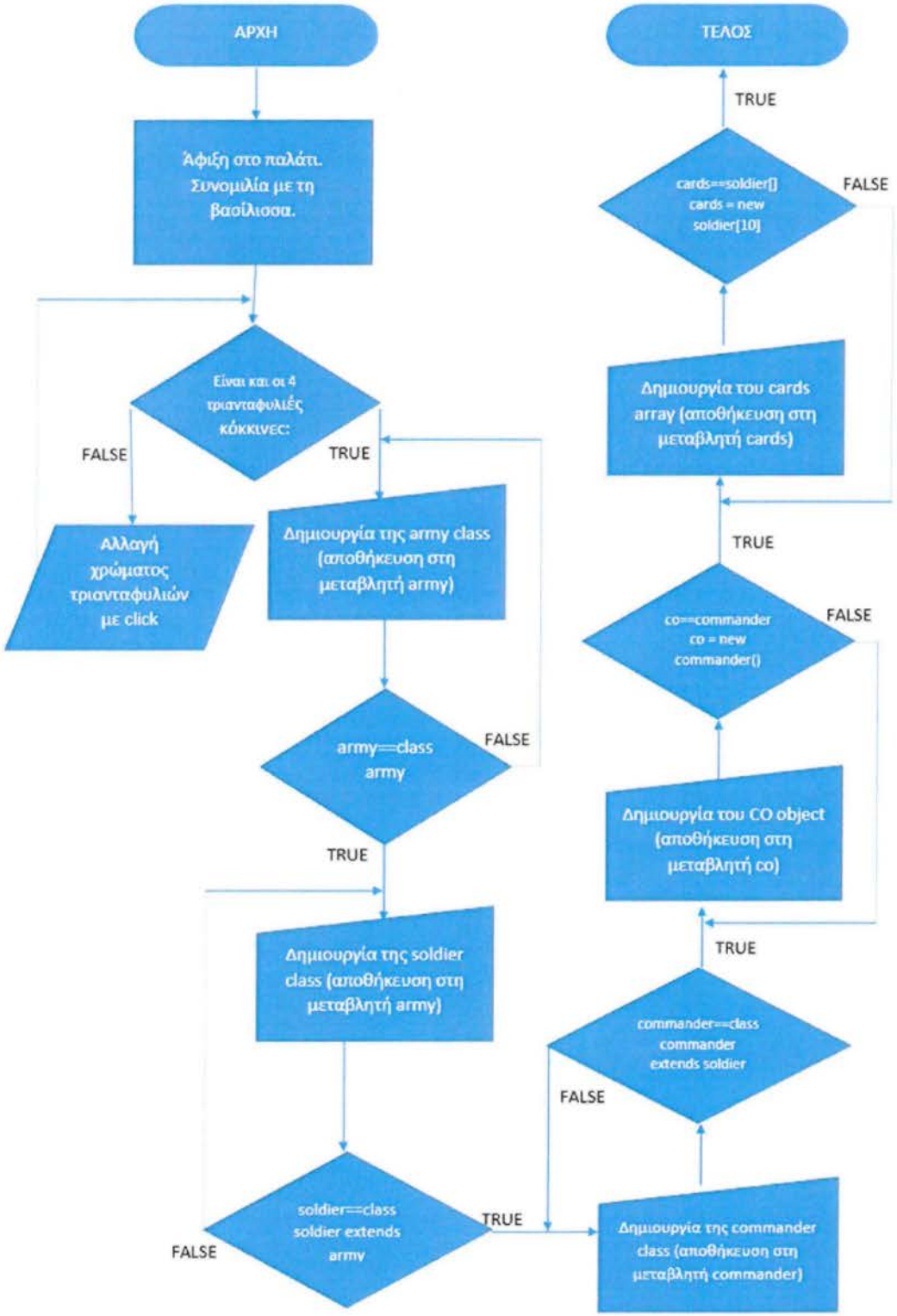
Κάνοντας click πάνω στα λουλούδια γίνεται η αλλαγή χρώματος. Ο έλεγχος του και η αλλαγή τους φαίνεται παρακάτω:

Στην συνέχεια έχουμε τον έλεγχο των strings που εισάγει ο χρήστης.



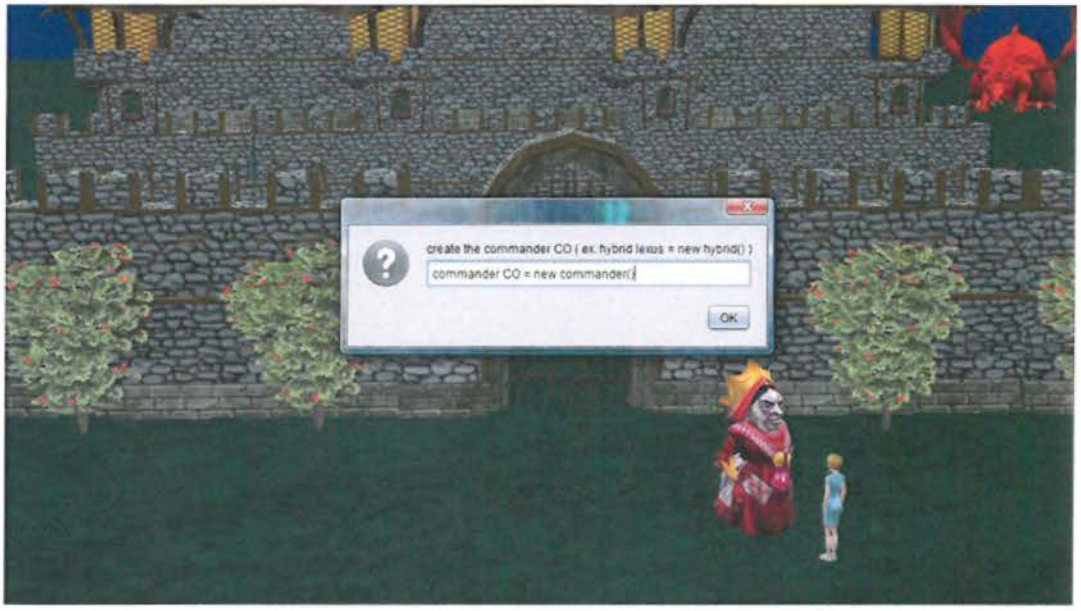
Παρομοίως και για τα υπόλοιπα strings.

Παρακάτω φαίνεται το λογικό διάγραμμα και το αποτέλεσμα του προγράμματος:



«Η συνάντηση της με την βασίλισσα θα την κάνει σοφότερη και θα της προσκομίσει ***κατανόηση νέων εννοιών όπως αυτής της κλάσης και της κληρονομικότητας των κλάσεων****, αλλά θα πρέπει να τα εφαρμόσει αν θέλει να επιβιώσει. Έτσι για χάρη της γνώσης θα δημιουργήσει στρατιώτη τραπουλόχαρτο για την βασίλισσα έναν ύστερα πολλούς με* ***χρήση πινάκων****.»*


create the commander class (ex. class hybrid EXTENDS car)
class commander EXTENDS soldier
OK


create the commander CO (ex. hybrid lexus = new hybrid())
commander CO = new commander()
OK


create the soldier array CARDS (ex: car[] opel = new car[5])
soldier[] CARDS = new soldier[10]
OK


THE END

3.4 - ALICE IN JAVALAND

Οι προγραμματιστικές έννοιες διδάσκονται μέσα από τη διαδικασία δημιουργίας ενός «κόσμου» (εφαρμογή). **Με την χρήση της ιστορίας η Αλίκη στη χώρα των θαυμάτων και την μεταφορά της στο δικό μας ALICE IN JAVALAND γίνεται η προσπάθεια για την κατανόηση βασικών εννοιών και αρχών του προγραμματισμού.** Μέσα από περιπέτειες μαθαίνει τη σημασία εννοιών του αντικειμενοστραφούς προγρμματισμού.

Η ανάπτυξη εκπαιδευτικών σεναρίων είναι σχεδιασμένα για τη διδασκαλία και την εκμάθηση βασικών αρχών του προγραμματισμού, που περιλαμβάνονται στο ALICE. Πιο συγκεκριμένα τα μαθήματα είναι for, if, polymorphism, private-public, και creating classes- κληρονομικότητα.

Τα παραδείγματα απαρτίζονται από απλές κινήσεις χαρακτήρων. Σε ένα κόσμο όπου ο κάθε χαρακτήρας αλληλεπιδρά με τον χρήστη, ο χρήστης έχει τη δυνατότητα με το πάτημα ενός κουμπιού, με την εισαγωγή χαρακτήρων ή με το click του ποντικιού να προκαλέσει την κίνηση ενός χαρακτήρα ή ενός αντικειμένου.

3.5 - Εκπαιδευτικοί στόχοι

Το ALICE είναι ένα προγραμματιστικό περιβάλλον σχεδιασμένο για να αποτελέσει την πρώτη επαφή ενός εκπαιδευόμενου με τον αντικειμενοστρεφή προγραμματισμό. Το ALICE στοχεύει μέσα από την εύχρηστη διεπαφή και τα ελκυστικά τρισδιάστατα γραφικά του να προσελκύσει το ενδιαφέρον των μαθητών για τον προγραμματισμό.

Βήμα-βήμα θα μπορούσε ο δάσκαλος να δείξει στους μαθητές/χρήστες πρώτα, στο θεωτητικό μέρος, το αρχικό έτοιμο κομμάτι σαν παράδειγμα προσομοίωσης, τι δυνάτοτητες έχουν και παράλληλα εξηγώντας τους κάποιες βασικές έννοιες και έπειτα να τους βάλει στην διαδικασία να κάνουν οι ίδιοι μια εφαρμογή ώστε να γίνει και το πρακτικό μέρος της διδασκαλίας.

Αυτό θα μπορούσε να γίνει **χρησιμοποιώντας τα αυτοτελή μαθήματα ως κεντρικό άξονα διδασκαλίας στο θεωρητικό μέρος**, γιατί στο πρακτικό θεωρείται απαραίτητο να υπάρξει επαφή του χρήστη με τη συγγραφή σεναρίου και τη χρήση κώδικα.

Πρωταρχικός στόχος είναι, με το ALICE IN JAVALAND, να γίνεται μια εισαγωγή στους μαθητές/χρήστες για το τι μπορούν να κάνουν χρησιμοποιώντας αυτό το περιβάλλον ανάπτυξης αντικειμενοστραφή προγραμματισμού. Επίσης να δίνεται η ευκαιρία παρακολούθησης και μάθησης βασικών εννοιών με απώτερο τελικό στόχο. Μετά το πέρας της παρουσίασης να γίνεται η δημιουργία και η συνέχεια της χρήσης των αντικειμενοστραφή εννοιών από τους χρήστες/μαθητές μέσα από το ALICE.

Στο σχέδιο του μαθήματος θα μπορούσε σαν πλάνο διδασκαλίας να χωριστεί και σε αυτοτελή κομμάτια αλλά και να διδαχτεί ολόκληρο και αυτούσιο σαν ένα πλάνο

μαθήματος που σε καθοδηγεί μέσα από τις σκηνές και τις αναδράσεις, αφήνοντας ύστερα τον χρήστη να δημιουργήσει το δικό του κόσμο και να κάνει χρήση των δικών του εντολών με σκοπό την δημιουργία ενός κόσμου και δυναμικών κινήσεων των αντικειμένων.

Δεν είναι απαραίτητο να δοθεί στους χρήστες ή μαθητές κάποιο διδακτικό πλάνο διότι μέσα από την διάδραση με τον κόσμο που δημιουργήσαμε, το Alice In Javaland, δίνονται σε όλα τα σημεία ευκαιρίες αλληλεπιδράσεις και επεξηγήσεις με έμμεσο ή άμεσο τρόπο ώστε να επιτευχθεί η συνέχεια της δράσης στον εικονικό διαμορφωμένο κόσμο.

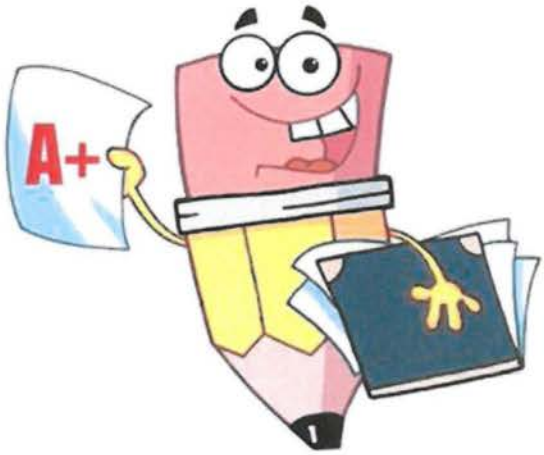
Το σχέδιο μαθήματος αφορά τους εκπαιδευτικούς στόχους που επιδιώκονται να επιτευχθούν κατά την ολοκλήρωση της διδασκαλίας και **καθοδηγεί τον καθηγητή ώστε να είναι εντός των στόχων.** Οι στόχοι επικεντρώνονται στο ότι μετά το πέρας του Alice In Javaland και την πρώτη επαφή με έννοιες προγραμματισμού, ο χρήστης/μαθητής να μπορεί να δημιουργήσει κάτι δικό του εφαρμόζοντας βασικές έννοιες αλλά κυρίως να θέλει να ασχοληθεί μ΄αυτό και να παραμείνει το ενδιαφέρον του ζωντανό.

Στόχος είναι, με το ALICE IN JAVALAND, να γίνεται μια εισαγωγή στους μαθητές/χρήστες στο τι μπορούν να κάνουν χρησιμοποιώντας αυτό το περιβάλλον ανάπτυξης αντικειμενοστραφή προγραμματισμού. Επίσης, να δίνεται η ευκαιρία παρακολούθησης και μάθησης βασικών εννοιών με απώτερο τελικό στόχο. Μετά το πέρας της παρουσίασης να γίνεται η δημιουργία και η συνέχεια της χρήσης αντικειμενοστραφών εννοιών από τους χρήστες/μαθητές μέσα από το Alice.

Τέλος επιδιώκεται με τα μαθήματα, οι μαθητές να μάθουν να δημιουργούν μεθόδους και συναρτήσεις, να χρησιμοποιούν λίστες και πίνακες, να κατανοούν τη Boolean λογική, να κάνουν σωστή χρήση των τοπικών και καθολικών μεταβλητών και να μάθουν τις δομές επιλογής και επανάληψης. Είναι πολύ βασικό ότι μαθαίνουν να προγραμματίζουν με αντικείμενα και να διδάσκονται τις έννοιες των κλάσεων και της κληρονομικότητας που αποτελούν βασικές έννοιες του αντικειμενοστρεφή προγραμματισμού.

ΜΕΡΟΣ 4ο

Αξιολόγηση και Συμπεράσματα



Το ALICE υποστηρίζει την δημιουργία ιστοριών, σκηνών και παιχνιδιών παρέχοντας:

i. ένα σύνολο υψηλού επιπέδου κινήσεων, που υποστηρίζουν οι χαρακτήρες και με αυτό τον τρόπο μπορούν να αλληλεπιδράσουν μεταξύ τους,
ii. μια συλλογή 3D χαρακτήρων και σκηνικών που ανταποκρίνονται σε κάθε είδους ιδέα του προγραμματιστή,
iii. ένα εγχειρίδιο που εισάγει τους χρήστες στις έννοιες του προγραμματισμού και τους βοηθάει να δημιουργούν δικά τους παραδείγματα.

Οι χρήστες του ALICE αποδείχτηκε ότι έμαθαν πολύ εύκολα και διασκεδαστικά τις βασικές δομές του προγραμματισμού με την χρήση του συγκεκριμένου προγράμματος. Παρακινούνταν περισσότερο από χρήστες που δούλευαν με άλλα προγράμματα γιατί:

- Μπορούν να σκεφτούν σε λίγο χρόνο μια ιστορία που θα θέλουν να δημιουργήσουν.
- Οι ιστορίες έχουν φυσική ροή και είναι απίθανο να απαιτήσουν εξεζητημένες προγραμματιστικές τεχνικές, έτσι είναι κατάλληλο και για τους αρχάριους μαθητές.
- Οι ιστορίες είναι ένα είδος προσωπικής έκφρασης και παρέχουν μια ευκαιρία να πειραματιστούν σε διαφορετικούς ρόλους, μια σημαντική δραστηριότητα για την εφηβεία.
- Οι φίλοι που δεν ασχολούνται με τον προγραμματισμό μπορούν άμεσα να καταλάβουν και να εκτιμήσουν ένα σχέδιο με κίνηση, το οποίο μπορεί να αποτελέσει θετική επίδραση.

Τα τρία κίνητρα που παρέχει το ALICE είναι:

- Αποτελεί εργαλείο που επιτρέπει στα παιδιά την διερεύνηση ιδεών
- Αποτελεί μέσο προσωπικής έκφρασης
- Αποτελεί κύριο λίθο για την μετέπειτα καριέρα ενός προγραμματιστή

Το πρόγραμμα ALICE επιλέχθηκε για τους εξής λόγους:

- Η εργασία με την βοήθεια ενός 3D περιβάλλοντος είναι ελκυστική και υποκινεί την σημερινή γενιά.
- Η οπτική φύση και η άμεση ανάδραση του προγράμματος, κάνει εύκολο στους μαθητές να δουν την επίδραση μιας εντολής που έχουν δώσει. Επιπλέον κάνει τον έλεγχο του προγράμματος ευκολότερο.
- Ο συντάκτης σκηνών που λειτουργεί με σύρσιμο και επιλογή, εμποδίζει τους μαθητές από το να κάνουν συντακτικά λάθη, στα οποία οι αρχάριοι είναι ευάλωτοι.
- Οι κλάσεις και τα 3D αντικείμενα στο ALICE παρέχουν μια εμφανή ιδέα και περιγραφή του αντικειμένου.

Χρησιμοποιώντας το ALICE, δεν παραλήφθηκε καμία από τις βασικές έννοιες του προγραμματισμού - μέθοδοι, κλάσεις, κληρονομικότητα - που χρησιμοποιούνται στην Java και την C++.

Η τεχνική με βάση τα αντικείμενα είναι η πιο ενδιαφέρουσα και πιο αποτελεσματική τεχνική εκμάθησης εισαγωγικών αρχών προγραμματισμού. Αυτή η τεχνική δίνει **έμφαση στις αρχές του αντικειμενοστραφούς προγραμματισμού** και σχεδιασμού από το πρώτο κιόλας μάθημα. Η τεχνική ξεκινάει με τις έννοιες των αντικειμένων και της κληρονομικότητας, συνεχίζει με τις πιο παραδοσιακές δομές ελέγχου αλλά πάντα μέσα στα πλαίσια του αντικειμενοστραφούς σχεδιασμού.

Για να γίνει μια ενδιαφέρουσα προσέγγιση αυτής της τεχνικής, δημιουργήθηκαν αρκετά προγραμματιστικά εργαλεία που προωθούν την τεχνική που έχει ως βάση τα αντικείμενα. Σε αυτή τη μελέτη γίνεται λόγος για το προγραμματιστικό εργαλείο ALICE που έχει 3D περιβάλλον εργασίας. Το 3D περιβάλλον εργασίας βοηθάει στην νοερή απεικόνιση των αντικειμένων και τους μαθητές να καταλάβουν και να "δουν" τις βασικές αντικειμενοστραφείς αρχές. Έτσι αν κάποιος μαθητής δημιουργήσει ένα αντικείμενο βάτραχο δεν χρειάζεται να γράψει κώδικα, παρά να προσθέσει το αντικείμενο στο πρόγραμμα που υλοποιεί τον κόσμο του.

Μετά από 3 χρόνια χρήσης του ALICE *παρατηρήθηκε ότι* οι μαθητές:

- Ανέπτυξαν ικανότητα σχεδιασμού προσεγμένων και αξιόλογου περιεχομένου προγραμμάτων.
- Ανέπτυξαν ικανότητα δημιουργίας αξιόλογων εικονικών κόσμων. Στους κόσμους που δημιουργούσαν οι μαθητές τα πάντα ήταν αντικείμενα που εκτελούσαν κινήσεις όπως σε ταινίες και σε videogames.
- Έμαθαν να ελέγχουν και να επιδιορθώνουν κώδικα. Επειδή οι μαθητές γνώριζαν ποιό κομμάτι κώδικα αναφέρεται σε κάποιο αντικείμενο, με ξεχωριστές αλλαγές, και αφού έβλεπαν τι επίδραση έχει στο αντικείμενο έκαναν τις κατάλληλες διορθώσεις.
- Έμαθαν να δημιουργούν το πρόγραμμα τους κομμάτι κομμάτι και να κάνουν παράλληλους ελέγχους σωστής λειτουργίας.
- Ασχολήθηκαν με την κληρονομικότητα. Οι μαθητές έγραφαν κώδικα ώστε να δημιουργήσουν μια πιο ισχυρή κλάση.
- Συνεργάστηκαν. Οι μαθητές δημιουργούσαν τους δικούς τους χαρακτήρες και μετά τους συνδύαζαν ώστε να υλοποιήσουν ένα κοινό κόσμο.
- Απέφυγαν τα λάθη. Οι μαθητές δεν μπορούν να κάνουν συντακτικά λάθη αφού ο συντάκτης τους απαγορεύει να δώσουν λανθασμένη τιμή σε μια μεταβλητή η σε ένα επαναληπτικό βρόγχο.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Adelman, C., Kemmis, S. and Jenkins, D. (1980) "Rethinking case study: notes from the Second Cmbridge Conference". H. Simons Towards a Science of the Singular. Centre for Applied Research in Education, University of East Anglia, 45-61.

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

[3] Bergin, J., Stehlik, M., Roberts, J., and Pattis, R. (2003). Karel J. Robot: A Gentle Intoduction to the Art of Object-Oriented Programmating in Java. Published manuscript. Διαθέσιμο στην ηλεκτρονική διεύθυνση http://csis.pace.edu/~bergin/KarelJava2ed/Karel++JavaEdition.html (τελευταία ανάκτηση 30/09/2012)

[4] Bergin, J. (2000b). Introducing Objects with Karel J. Robot. Position paper presented in Workshop 8 (Tools and Envoriments for Understanding Object-Oriented Concepts) at ECOOP 2000. June 8, Sophia antipolis, France. Διαθέσιμο στην ηλεκτρονική διεύθυνση http://csis.pace.edu/~bergin/karel/ecoop2000JBKarel.html (τελευταία ανάκτηση 30/09/12)

[5] Bergin, J., Stehlik, M., Roberts, J., and Pattis, R. (1997). Karel++ A Gentle Introduction to the Art of Object Oriented Programming, Wiley.

[6] Beyer, B., Rynes, K., Perrault, J., Hay, K., and Haller, S. Gender. Differences in Computer Science Students. Proceedings of the Thirty- Fourth SIGCSE Techincal Symposium, Vol 35, No. 1, February 2003, pp.49-53.

[7] Bishop-Clark, C., Courte, J., & Howard, E.. V., "Programming in pairs with ALICE to improve confidence, enjoyment, and achievement", Journal of Educational Computing Research, Vol.34, No2, 2006, pp.213-228

[8] Bruce, K., Danyluk, A., and Murtagh, T. (2001). A library to support a graphics-based object-first approach to CS 1. ACM SIGCSE Bulletin 33(1), 6-10.

[9] 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

[10] 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).

[11] Buck, D., and Stucki, D.J. (2001). JKarelRobotQ A case Study in Supporting Levels of Cognitive Development in the Computer Science Curriculum. ACM SIGSE Bulletin 33(1), 6-10

[12] CC2001 Computing Curricula 2001 (final report) (2001). The Joint Task Force on Computing Curricula (IEEE Computer Society and Association for Computing Machinery), December 15.

[13] Cooper et al., 2003; Kolling, 1999a and b; CC2001, 2001; Proulx et al., 2002; Bruce et al., 2001

[14] Ξυνόγαλος, 2002; Xinogalos et al., 2006; Xinogalos and Satratzemi, 2002