Τμήμα
Μηχανικών
Πληροφορικής τ.ε.
Τεχνολογικό Εκπαιδευτικό Ίδρυμα
Δυτικής Ελλάδας

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«Συστήματα Εκπαίδευσης για Προγραμματισμό Εφαρμογών για Παιδιά Δημοτικού»

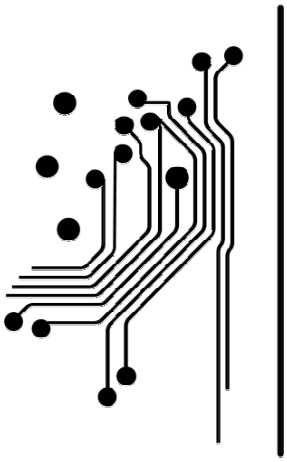
ΤΣΑΛΙΚΗ ΜΑΡΙΑ

Α.Μ.: 201

Επιβλέπων:

Σπύρος Συρμακέσης, Καθηγητής

ΑΝΤΙΡΡΙΟ
ΣΕΠΤΕΜΒΡΙΟΣ 2015



Τμήμα Μηχανικών Πληροφορικής τ.ε.

Τεχνολογικό Εκπαιδευτικό Ίδρυμα Δυτικής Ελλάδας

www.ebusiness-lab.gr

Τμήμα Μηχανικών Πληροφορικής ΤΕ

Σχολή Τεχνολογικών Εφαρμογών

Περίληψη

Θέμα της πτυχιακής εργασίας είναι μία μελέτη για τους τρόπους εκμάθησης προγραμματισμού σε νεαρές ηλικίες, που αναπτύχθηκε ως πτυχιακή εργασία για το Τεχνολογικό Εκπαιδευτικό Ίδρυμα Δυτικής Ελλάδας, για το τμήμα Μηχανικών Πληροφορικής.

Τα πρότυπα του Δυτικού κόσμου και του καταναλωτισμού έχουν επιφέρει ραγδαία ανάπτυξη της τεχνολογίας, με αυτή να εφαρμόζεται σε όλα τα πεδία της καθημερινής μας ζωής. Από σχολεία μέχρι νοσοκομεία, μέσα μετακίνησης και ενημέρωσης, τρόπους επιχειρηματικότητας ακόμα και τρόπους αγοράς προϊόντων, οι πολίτες του Δυτικού κόσμου είναι σχεδόν “υποχρεωμένοι” να είναι ενήμεροι για τις τελευταίες εξελίξεις τις τεχνολογίας και να είναι εξοικειωμένοι με αυτές και τους τρόπους χρήσης αυτών.

Μια τέτοια πραγματική επανάσταση στον τρόπο ζωής των ανθρώπων έχει ως συνέπεια και την ανάγκη αλλαγής του συστήματος εκπαίδευσης.

Ζούμε σε μία χώρα, στην οποία συνεχώς το σύστημα παιδείας μεταβάλλεται προσπαθώντας να ακολουθήσει τις εξελίξεις της επιστήμης της τεχνολογίας. Σίγουρα, λοιπόν, από τα πιο σημαντικά κεφάλαια της παιδείας πρέπει πλέον να είναι η απόκτηση οικειότητας των παιδιών με την τεχνολογία και όχι μόνο. Ακόμα πιο σημαντικό ίσως είναι η προσπάθεια διδασκαλίας μέσων δημιουργίας νέας τεχνολογίας.

Τα τελευταία χρόνια, οι οικονομικές κρίσεις που γεννάει το σύστημα του καπιταλισμού, έχουν δημιουργήσει την εντύπωση πως ίσως ένας δρόμος καταπολέμησης αυτού του προβλήματος είναι η στροφή στην εξαγωγή λογισμικού. Θα ήταν λοιπόν επιθυμητό, τα παιδιά μας, το μέλλον της χώρας μας, να είναι σε θέση να εξάγουν ικανό λογισμικό. Για το σκοπό αυτό, πρέπει

να βρούμε τους τρόπους διδασκαλίας γλωσσών προγραμματισμού, και επειδή αυτές δεν είναι εύκολες, και ευνόητες για ένα μικρό παιδί, ίσως πρέπει να βρεθούν εργαλεία για να καταστίσουν πιο εύκολη την εκμάθηση προγραμματισμού σε νεαρές ηλικίες. Η παρούσα έρευνα στόχο έχει να παρουσιάσει τα πιο διαδεδομένα και εύχρηστα εργαλεία που η χρήση τους θα δώσει τη δυνατότητα εκμάθησης προγραμματισμού σε νεαρές ηλικίες.

Δομή Εργασίας

Η εργασία ακολουθεί της εξής δόμηση:

Το 1^{ο} κεφάλαιο παρέχει μια εισαγωγή στο επιστημονικό πεδίο Εκμάθησης Προγραμματισμού σε νεαρές ηλικίες.

Το 2^{ο} κεφάλαιο αναλύει τον ορισμό της εκπαιδευτικής ψυχαγωγίας.

Το 3^{ο} κεφάλαιο αναλύει τα πιο βασικά εργαλεία εκμάθησης προγραμματισμού σε νεαρές ηλικίες.

Το 4^{ο} κεφάλαιο αναλύει ασκήσεις πάνω στις εφαρμογές που αναλύονται στο 3^{ο} κεφάλαιο.

Το 5^{ο} κεφάλαιο αναφέρεται σε συμπεράσματα της παρούσας έρευνας.

Το 6^{ο} κεφάλαιο αναφέρει τη βιβλιογραφία που χρησιμοποιήθηκε για την εκπόνηση της παρούσας έρευνας.

Περιεχόμενα

1. Πρόλογος

1.1 Εισαγωγή

Εάν κοιτάξουμε σήμερα γύρω μας, καθώς περπατάμε ή στεκόμαστε στο λεωφορείο θα δούμε παντού χιλιάδες μικρά παιδιά να κρατούν στα χέρια τους κινητά τηλέφωνα και να μπορούν να κάνουν κλήσεις ή να βγάζουν φωτογραφίες με αυτά, να παίζουν με tablet και να κατεβάζουν εφαρμογές, να

κάθονται με απόλυτη άνεση σε έναν Η/Υ και να μπορούν άνετα να σερφάρουν στον Ίντερνετ.

Είναι φανταστικό το πόσο γρήγορα και χωρίς ιδιαίτερη προσπάθεια τα μικρά παιδιά μπορούν να απορροφήσουν μία πληροφορία και να την κατανοήσουν. Συχνά ακούμε τους μεγαλύτερους να λένε πως “το μυαλό ενός παιδιού είναι 'σφουγγάρι'”. Και είναι αλήθεια.

Έρευνες έχουν δείξει, πώς ένα παιδί μπορεί να μάθει πιο γρήγορα και πιο εύκολα από κάποιον μεγάλο. Βασικό όπλο του παιδιού σε αυτό είναι φυσικά η φαντασία του.

Επειδή λοιπόν τα παιδιά έχουν αυτή τη μοναδική ικανότητα, είναι πολύ σημαντικό να βρεθούν τα σωστά εργαλεία από τους μεγαλύτερους που θα είναι σε θέση να τους μεταδώσουν με τον πιο σωστό και ολοκληρωμένο τρόπο τη γνώση.

Στην έρευνα που πραγματοποιήθηκε, βρέθηκαν διάφορα εργαλεία διδασκαλίας προγραμματισμού για παιδιά, αλλά θα αναλυθούν τα πιο διαδεδομένα, που είναι τα εξής:

1. Scratch
2. KidsRuby
3. Alice
4. Kodu
5. Greenfoot

2. Διδακτική του Προγραμματισμού

2.1 Η Διδακτική του Προγραμματισμού

Παλαιότερα, η διδακτική του προγραμματισμού ήταν συνδεδεμένη με την διδασκαλία μιας γλώσσας προγραμματισμού. Πλέον όμως, το ενδιαφέρον εστιάζεται στις μορφές συλλογισμού που χρησιμοποιούν οι έμπειροι και αρχάριοι προγραμματιστές καθώς και στις μεθόδους για την επίλυση προγραμματιστικών προβλημάτων.

Υπάρχουν διαφορετικές αντιλήψεις για το τι ακριβώς είναι ένα πρόγραμμα και κατά συνέπεια διαφορετικές προσεγγίσεις για τη διδασκαλία του προγραμματισμού.

- Η προσέγγιση που χρησιμοποιείται ακόμα στη δευτεροβάθμια εκπαίδευση: το πρόγραμμα περιγράφει ένα σύνολο από δυνατούς υπολογισμούς.
- Τα προγράμματα ως συναρτήσεις που δέχονται εισόδους (inputs) και έναν κανόνα με τον οποίο συνδυάζονται οι είσοδοι για να παράγουν μια τιμή ή έξοδο (output).
- Η τρίτη και πιο πρόσφατη αντίληψη του προγράμματος, συνίσταται στον ορισμό από τον προγραμματιστή αντικειμένων και σχέσεων μεταξύ τους.

Ωστόσο όποια προσέγγιση και αν ακολουθεί ο προγραμματιστής, πάντα βρίσκεται μπροστά σε έναν προς επίτευξη στόχο. Άρα, ο προγραμματισμός συνίσταται στην οργάνωση των διαδικασιών που θα επιτρέψουν την επίτευξη του στόχου με τη χρήση μιας γλώσσας προγραμματισμού πάνω σε μια συγκεκριμένη μηχανή.

Βασικός στόχος της διδασκαλίας του προγραμματισμού πρέπει να είναι η καλλιέργεια δεξιοτήτων επίλυσης προβλημάτων, δηλαδή η ανάπτυξη της ικανότητας των μαθητών να εφαρμόζουν τις γνώσεις τους για την επίλυση προβλημάτων που δεν έχουν διδαχθεί πιο πριν.

Οι δεξιότητες επίλυσης περιλαμβάνουν:

Α) την κατανόηση και αναπαράσταση της αρχικής κατάστασης του προβλήματος, συμπεριλαμβανομένου και του προσδιορισμού των ειδών της πληροφορίας που απαιτούνται για τη λύση του,

Β) τη συλλογή και οργάνωση της κατάλληλης και σημαντικής πληροφορίας

Γ) την κατασκευή και διαχείριση ενός σχεδίου δράσης ή μιας στρατηγικής και η αναζήτηση ευρετικών τεχνικών.

Δ) Τέλος το διαχωρισμό ενός σύνθετου προβλήματος σε απλούστερα-η λύση των οποίων είναι ήδη γνωστή.

Ε) ο έλεγχος υποθέσεων και λήψη απόφασης.

2.2 Διακρίσεις για τη γνώση του προγραμματισμού

Υπάρχει διάκριση ανάμεσα στη:

- θεωρητική γνώση του προγραμματισμού (δηλωτικής φύσεως, για παράδειγμα η δήλωση του πως λειτουργεί ένας βρόγχος for) και
- στις προγραμματιστικές στρατηγικές (το πως η γνώση χρησιμοποιείται, για παράδειγμα η σωστή χρησιμοποίηση ενός βρόγχου for μέσα στο πρόγραμμα).

Τα περισσότερα εισαγωγικά εγχειρίδια προγραμματισμού αφιερώνουν το μεγαλύτερο μέρος του περιεχομένου τους στην παρουσίαση της θεωρητικής γνώσης για μια ιδιαίτερη γλώσσα (που διαμορφώνεται με τα παραδείγματα και τις ασκήσεις).

Πολλές φορές ΔΕΝ υπάρχει αναφορά και ανάλυση των στρατηγικών που υιοθετούνται από τους προγραμματιστές στην παραγωγή και κατανόηση των προβλημάτων Π.χ. Ανάγνωση και κατανόηση προγραμμάτων: – Οι έμπειροι τείνουν να διαβάζουν τα συμβατικά προγράμματα υιοθετώντας μια από επάνω προς τα κάτω εννοιολογικά καθοδηγούμενη στρατηγική ενώ διάβαζαν τα ασυνήθιστα προγράμματα χρησιμοποιώντας μια από κάτω προς τα επάνω ευρετική στρατηγική – Οι αρχάριοι τείνουν να διαβάζουν και τα δυο είδη προγραμμάτων με τον ίδιο τρόπο.

Μια άλλη σημαντική διάκριση είναι αυτή μεταξύ των μελετών που ερευνούν την κατανόηση προγράμματος (όπου δίνεται ο κώδικας ενός προγράμματος και οι σπουδαστές πρέπει να κατανοήσουν το πως αυτό λειτουργεί), και εκείνων που επικεντρώνονται στην παραγωγή προγραμμάτων (όπου οι σπουδαστές πρέπει να δημιουργήσουν ένα μέρος ή ένα ολόκληρο πρόγραμμα για να εκτελεσθεί κάποιος στόχος ή να λυθεί κάποιο πρόβλημα).

Οι γλώσσες προγραμματισμού που ακολουθούν το διαδικαστικό παράδειγμα επιτρέπουν μετασχηματισμούς που οδηγούν από μια κατάσταση σε μια άλλη: – Το πρόγραμμα είναι η έκφραση μιας διαδικασίας που οδηγεί από μια αρχική κατάσταση (δεδομένα) σε μια τελική κατάσταση (τα αποτελέσματα) χρησιμοποιώντας τις επιτρεπτές από τη γλώσσα πράξεις. Στο

αντικειμενοστραφές παράδειγμα, η δραστηριότητα του προγραμματιστή συνίσταται σε μια δραστηριότητα σχεδιασμού αντικειμένων. – Μελέτες δείχνουν ότι ο προσδιορισμός των αντικειμένων δεν είναι εύκολη διαδικασία, τα αντικείμενα που αναγνωρίζονται στο πρόβλημα δεν είναι πάντα απαραίτητα στο πρόγραμμα και οι αρχάριοι πρέπει να κατασκευάσουν ένα πρότυπο των διαδικαστικών πτυχών μιας λύσης προκειμένου να σχεδιάσουν κατάλληλα τα αντικείμενα και κλάσεις.

2.3 Η γνώση του προγραμματισμού

- Οι φάσεις απόκτησης δεξιοτήτων
- Οι επιπτώσεις
- Οι γνωστικές προϋποθέσεις

Η δομή των γνώσεων συνίσταται

- σε ένα σύστημα εννοιών,
- πράξεων πάνω στις έννοιες,
- σχέσεων ανάμεσά τους και
- συμβολισμών για την αναπαράσταση των εννοιών
- Έννοιες, πράξεις στις έννοιες, σχέσεις
- Συμβολισμοί

Κάθε πρόβλημα προγραμματισμού αναφέρεται σε ένα σύνολο εννοιών σε αλληλεπίδραση, – π.χ., η επεξεργασία λιστών συνεπάγεται τις έννοιες ο της μεταβλητής, ο της επανάληψης, ο της αναδρομικότητας, ο των συναρτήσεων που επιδρούν πάνω στις λίστες, ο στις εισόδους και στις εξόδους δεδομένων – και αυτά σε οποιαδήποτε γλώσσα προγραμματισμού και με οποιεσδήποτε συντακτικές και σημασιολογικές ιδιαιτερότητες.

2.4 Οι Φάσεις της εκμάθησης

1η φάση: Προηγούμενες γνώσεις

Στην αρχή της μαθησιακής διαδικασίας, ο μαθητής κάνει αναφορές στην προηγούμενή του εμπειρία. Οι γνωστικές δραστηριότητες εξαρτώνται από την πρότερη εμπειρία και τα ατομικά γνωστικά εργαλεία του μαθητή. Προσπαθεί να συνδέσει το εξεταζόμενο υλικό με την προηγούμενή του κατανόηση.

Τα νοητικά σχήματα που δημιουργούνται από πρότερες γνώσεις μπορούν να προκαλέσουν παραγωγικά αποτελέσματα, δίνοντας, για παράδειγμα, νόημα σε νέες έννοιες και πράξεις – Τα νοητικά σχήματα που δημιουργούνται από πρότερες γνώσεις μπορούν να προκαλέσουν και μειωτικά αποτελέσματα (γνωστικά εμπόδια), λόγω των διαφορών των δομών των γνώσεων μέσα στο χώρο προέλευσης και το στόχο πρόσκτησης της γνώσης. – Μια οικοδομημένη γνώση προγραμματισμού μπορεί να αποδειχθεί εμπόδιο και να απαιτήσει μια διαδικασία «αποδόμησης».

2η φάση: Εξέλιξη τοπικών γνώσεων

Οι διάφοροι «χώροι» γνώσης εξελίσσονται. Νέες γνώσεις οικοδομούνται μέσω νέων αλληλεπιδράσεων. Για παράδειγμα, – το πέρασμα των παραμέτρων μέσα σε διαδικασίες και συναρτήσεις συνεπάγεται όχι μόνο την πρόσκτηση της έννοιας της διαδικασίας, αλλά επίσης έναν εκ νέου ορισμό των μεταβλητών ως τοπικών και καθολικών. Οικοδομείται συνεπώς ένα νέο επίπεδο κατανόησης των σχέσεων ανάμεσα σε μεταβλητή, όνομα και τιμή.

3η φάση: Γνωστική ευελιξία

Στη συνέχεια της μαθησιακής διαδικασίας, το σύνολο των ήδη επιλυμένων προβλημάτων αυξάνεται και μια κατηγοριοποίηση, έστω ασυνείδητη, γίνεται πλέον εφικτή. Διαφορετικές στρατηγικές γίνονται αυτόματα διαθέσιμες, συνδέοντας τύπους προβλημάτων με τρόπους επίλυσης, όπως, για παράδειγμα, τα προβλήματα που λύνονται μεαναδρομή (γνώση βασισμένη σε περιπτώσεις). Αυξάνει η πολυπλοκότητα των προβλημάτων τα οποία ο μαθητής μπορεί να προσεγγίσει. Πολύπλοκες δομές δεδομένων κατά την

αρχική φάση της διδασκαλίας γίνονται σταδιακά οικεία αντικείμενα (π.χ. οι πίνακες στις διαδικαστικές γλώσσες). Η δημιουργία ενός προγράμματος περνά από πολλά στάδια και ο προγραμματιστής πρέπει: – Να είναι σε θέση να κατανοήσει το πρόβλημα που τέθηκε (ποια είναι τα δεδομένα και ποια τα ζητούμενα – αρχική και τελική κατάσταση). – Να είναι σε θέση να καθορίσει και στη συνέχεια να σχεδιάσει τη μέθοδο επίλυσης με την περιγραφή του αλγόριθμου (να περιγράψει δηλαδή με σαφήνεια το πέρασμα από την κατάσταση εκκίνησης στην κατάσταση – στόχο). Η δημιουργία ενός προγράμματος περνά από πολλά στάδια και ο προγραμματιστής πρέπει: – Να είναι σε θέση (από τη στιγμή που ο αλγόριθμος έχει αναπτυχθεί) να τον μεταφράσει σε γραπτό κώδικα στο πλαίσιο ενός προγραμματιστικού περιβάλλοντος. – Να είναι σε θέση να βρίσκει τα λάθη μέσα σε ένα πρόγραμμα και να καθορίζει τις αντίστοιχες λύσεις. Η διαδικασία της αποσφαλμάτωσης, είναι εγγενές χαρακτηριστικό της προγραμματιστικής δραστηριότητας.

2.5 Επιπτώσεις

Η μάθηση του προγραμματισμού μπορεί να προκαλέσει επτά αλλαγές στο γνωστικό σύστημα των μαθητών:

1. Αυστηρότητα στη σκέψη, ακρίβεια έκφρασης, συνειδητή ανάγκη αποσαφήνισης των ενεργειών.
2. Πρόσκτηση και κατανόηση γενικών εννοιών, όπως διαδικασία, μεταβλητή,συνάρτηση, μετασχηματισμός (σχετίζονται άμεσα με τη μαθηματική παιδεία).
3. Πρόσκτηση ευρετικών ικανοτήτων και μεθοδολογίας: σχεδιασμός, αναζήτηση παρόμοιων περιπτώσεων, επίλυση με ανάλυση σε μέρη.
4. Μάθηση τεχνικών αναζήτησης λαθών, που μπορεί να μεταφερθεί και σε άλλους εκτός προγραμματισμού χώρους. Επιπτώσεις προγραμματισμού η μάθηση του προγραμματισμού μπορεί να προκαλέσει επτά αλλαγές στο γνωστικό σύστημα των μαθητών
5. Πρόσκτηση της γενικής ιδέας οικοδόμησης της λύσης με τη μορφή μικρών διαδικασιών ή στοιχειωδών τμημάτων, τα οποία μπορούν να

χρησιμοποιηθούν συνδεόμενα για την οικοδόμηση της λύσης σύνθετων προβλημάτων.

6. Επέκταση της συνειδητοποίησης και της γνώσης πάνω σε τεχνικές επίλυσης προβλημάτων.
7. Επέκταση και ανάπτυξη της χρήσης συγκριτικών μεθόδων που αφορούν την πολλαπλότητα των τρόπων, ώστε να επιτευχθεί ένας δεδομένος στόχος.

2.6 Γνωστικές προϋποθέσεις

Για να είναι δυνατή η μάθηση του προγραμματισμού, οι μαθητές πρέπει να διαθέτουν γνωστικές δομές οι οποίες τους επιτρέπουν:

1.1. Να οικοδομεί κανόνες προγραμματισμού.

1.2. Να οικοδομεί αναλυτικές νοητικές αναπαραστάσεις τού τι συμβαίνει στη μηχανή όταν το πρόγραμμα εκτελείται (π.χ. στη μνήμη του υπολογιστή).

1.3. Να προσομοιώνει τμήματα πράξεων του υπολογιστή ώστε να μπορεί να τα προβλέπει καλύτερα (όπως, π.χ., τις κλήσεις μιας αναδρομικής διαδικασίας).

1.4. Να συγκρατήσει νοητικά ικανές ποσότητες πληροφορίας.

2.7 Η διδακτική του προγραμματισμού

- Μελέτη περίπτωσης
- Προσέγγιση μαύρο κουτί
- Διερευνήσεις
- Συνεργατικές μέθοδοι
- Βάσει παραδειγμάτων

Χρήση μελετών περίπτωσης

Οι μελέτες περίπτωσης μπορούν να χρησιμοποιηθούν είτε στο πλαίσιο της διδασκαλίας είτε στο πλαίσιο εργασιών που ανατίθενται στους μαθητές. Τα βασικά συστατικά μιας μελέτης περίπτωσης (case study) είναι

1. η διατύπωση ενός προβλήματος
2. μία περιγραφή της διαδικασίας για την επίλυση του προβλήματος που ακολουθείται από έναν έμπειρο προγραμματιστή ο το πρόγραμμα (κώδικας) που επιλύει το πρόβλημα
3. ερωτήσεις που αξιολογούν κατά πόσο ο μαθητής έχει κατανοήσει τη λύση ο ερωτήσεις που επιδιώκουν την αντιμετώπιση μαθησιακών δυσκολιών ο ερωτήσεις που στοχεύουν στην εξάσκηση του μαθητή στην ανάλυση προβλημάτων και στη σχεδίαση και υλοποίηση της λύσης τους.

Μέσα από τις μελέτες περίπτωσης, οι μαθητές καλούνται – να σχολιάσουν τη λύση που προτείνεται, – να διερευνήσουν αν η λύση μπορεί να εφαρμοστεί σε παρόμοια προβλήματα, – να επεκτείνουν ή να τροποποιήσουν τη λύση στο πλαίσιο άλλων προβλημάτων, – να κάνουν προβλέψεις για τα αποτελέσματα της προτεινόμενης λύσης, – να διορθώσουν τυχόν λάθη ή ακόμη να κατασκευάσουν μελέτες περίπτωσης για συγκεκριμένα προβλήματα.

Παράδειγμα εφαρμογής:

Δίνεται στους μαθητές ένα πρόγραμμα το οποίο έχει ως αποτέλεσμα την εμφάνιση του ημερολόγιου (ανά μήνα) για συγκεκριμένο έτος που εισάγει ο χρήστης. Στα πλαίσια της μελέτης περίπτωσης, μπορεί να ζητηθεί από τους μαθητές – (i) να μελετήσουν και να σχολιάσουν με ποιο τρόπο το συγκεκριμένο πρόγραμμα χειρίζεται αρνητικές τιμές εισόδου, αν λαμβάνει υπόψη και με ποιο τρόπο τα δίσεκτα έτη, κ.λπ. – (ii) να τροποποιήσουν το πρόγραμμα ώστε να εμφανίζει με διαφορετικό τρόπο τις ημερομηνίες που ανήκουν σε Σαββατοκύριακα, επίσημες αργίες κ.λπ. Χρήση μελετών περίπτωσης Παράδειγμα εφαρμογής Εναλλακτικά, μπορεί να μη δοθεί στους μαθητές η ολοκληρωμένη λύση, αλλά τμήμα του κώδικα, η διατύπωση του προβλήματος, μια περιγραφή της λύσης από τον ειδικό και να ζητηθεί από τους μαθητές να ολοκληρώσουν τη λύση.

Η Προσέγγιση "Μαύρο Κουτί"

Στην προσέγγιση «Μαύρο-Κουτί», προτείνεται οι μαθητές να εξοικειωθούν αρχικά με τις νέες έννοιες κατά την εκπόνηση δραστηριοτήτων στο εργαστήριο και στη συνέχεια να συμμετάσχουν σε μία διάλεξη-συζήτηση Οι δραστηριότητες περιλαμβάνουν δύο βασικά τμήματα: – (i) αρχικά οι μαθητές καλούνται να εκτελέσουν απλά προγράμματα (των οποίων δε γνωρίζουν τον κώδικα και τη λειτουργία – "μαύρα κουτιά"), να συνδιαλλαγούν με τον υπολογιστή, και να απαντήσουν σε μία σειρά από ερωτήσεις που αφορούν κυρίως στο διάλογο με τον υπολογιστή, – (ii) στη συνέχεια, οι μαθητές μελετούν τον κώδικα του προγράμματος και απαντούν σε ερωτήσεις σχετικά με τις εντολές που χρησιμοποιούνται. – (iii)Τέλος, οι μαθητές συζητούν τις απαντήσεις και τους προβληματισμούς τους και αποσαφηνίζουν τυχόν απορίες τους με τον εκπαιδευτικό. Μέσω αυτής της διδακτικής προσέγγισης, οι μαθητές εισάγονται στις βασικές έννοιες και δομές του προγραμματισμού ενεργητικά, διερευνώντας οι ίδιοι αρχικά τα χαρακτηριστικά των προγραμματιστικών εννοιών και δομών.

Παράδειγμα εφαρμογής:

Όταν οι μαθητές πρόκειται να διδαχθούν την έννοια της μεταβλητής και τη λειτουργία των εντολών εισόδου-εξόδου και ανάθεσης τιμής, τους δίνεται μία σειρά από απλά προγράμματα που περιλαμβάνουν σχετικές εντολές, διαφέρουν ελάχιστα μεταξύ τους και καλύπτουν τα χαρακτηριστικά των εννοιών που διδάσκονται. Οι μαθητές, στο πρώτο μέρος της δραστηριότητας, καλούνται να απαντήσουν σε ερωτήσεις όπως «τι πληκτρολογήσατε ως είσοδο στο τάδε μήνυμα», «τι είδους μηνύματα εμφανίστηκαν στην οθόνη που ήταν σχετικά με την είσοδο που δώσατε», κ.λ.π. Στο δεύτερο μέρος, οι μαθητές μελετούν τον κώδικα και απαντούν σε ερωτήσεις όπως «που αποθηκεύτηκε η τιμή που πληκτρολογήσατε», «ποια πρόταση/εντολή προκάλεσε την είσοδο της τιμής», κ.λ.π.

Προσέγγιση βασισμένη στις διερευνήσεις

Οι διερευνήσεις είναι δομημένες διδακτικο-μαθησιακές δραστηριότητες οι οποίες επιτρέπουν/υποστηρίζουν την ανάδειξη της υπάρχουσας αντίληψης και δίνουν τη δυνατότητα στους μαθητές να ενεργοποιηθούν και μέσα από ένα πλαίσιο καθοδήγησης να οδηγηθούν στην επιθυμητή εννοιολογική αλλαγή.

Ο μαθητής καλείται αρχικά να διαβάσει ένα μικρό πρόγραμμα Στη συνέχεια απαντά σε ερωτήσεις σχετικές με τη λειτουργία και τα αποτελέσματα της εκτέλεσης των προγραμματιστικών δομών που χρησιμοποιούνται και προβλέπει τη «συμπεριφορά» του προγράμματος για προκαθορισμένες ή μη προκαθορισμένες τιμές εισόδου Ελέγχει τις απαντήσεις του, εκτελώντας το πρόγραμμα στον υπολογιστή. Σε περίπτωση που οι προβλέψεις του δεν ανταποκρίνονται στα πραγματικά αποτελέσματα, ο μαθητής καθοδηγούμενος από ειδικά σχεδιασμένες ερωτήσεις καλείται να δώσει εξηγήσεις. Οι ερωτήσεις μπορεί να έχουν τη μορφή προτεινόμενων ενεργειών που διευκολύνουν το μαθητή να εντοπίσει το λάθος του ώστε να μπορέσει να το διορθώσει.

Προκειμένου οι «διερευνήσεις» να έχουν τα επιθυμητά αποτελέσματα πρέπει να (i) επικεντρώνονται σε ένα συγκεκριμένο θέμα/έννοια, (ii) περιλαμβάνουν ερωτήσεις διαφορετικού βαθμού δυσκολίας (εύκολες ώστε να μην απογοητεύουν το μαθητή αλλά και δύσκολες ώστε να μην προβλέπονται εύκολα και να παροτρύνουν το μαθητή για περαιτέρω πειραματισμό), (iii) προσαρμόζονται στο επίπεδο γνώσεων των μαθητών, ώστε να μπορούν σταδιακά οι μαθητές να μαθαίνουν από τα λάθη τους, (iv) ενθαρρύνουν τους μαθητές να προσέχουν σημαντικά χαρακτηριστικά και ιδιότητες των εννοιών, και (v) έχουν μετρήσιμα αποτελέσματα.

Παράδειγμα εφαρμογής:

Οι μαθητές αντιμετωπίζουν δυσκολίες στην έννοια της μεταβλητής και στη λειτουργία των εντολών ανάθεσης τιμής. Συχνά θεωρούν ότι η αρχικοποίηση της τιμής των μεταβλητών, έχει ως αποτέλεσμα οι τιμές αυτές να μην αλλάζουν μέσω άλλων εντολών ανάθεσης. Προκειμένου οι μαθητές να καταλάβουν πώς οι μεταβλητές αλλάζουν τιμή χρησιμοποιώντας εντολές ανάθεσης τιμής, μπορεί να τους δοθεί ένα πρόγραμμα (αρχικά ο κώδικας του προγράμματος σε έντυπη μορφή και στη συνέχεια το εκτελέσιμο αρχείο σε ηλεκτρονική) το οποίο περιέχει σχετικές εντολές και συνδυάζει μεταβλητές με

εντολές ανάθεσης τιμής. Αρχικά, ζητείται από τους μαθητές να μελετήσουν τον κώδικα του προγράμματος, να προβλέψουν τις αρχικές τιμές των μεταβλητών, να περιγράψουν το αποτέλεσμα της εκτέλεσης κάθε εντολής και να προβλέψουν τις τελικές τιμές των μεταβλητών. Στη συνέχεια, καλούνται να εκτελέσουν το πρόγραμμα και να εξηγήσουν τις τυχόν διαφορές που υπάρχουν ανάμεσα στις προβλέψεις τους και στα πραγματικά αποτελέσματα.

Προσέγγιση βασισμένη στη συνεργασία

Η προσέγγιση ανάπτυξης προγραμμάτων από ομάδες των δύο ατόμων – προγραμματίζοντας σε ζευγάρια (pair-programming) - προτείνεται και εφαρμόζεται τόσο στη διαδικασία της διδασκαλίας όσο και στην εκπόνηση εργασιών. Δύο άτομα συνεργάζονται στη σχεδίαση και ανάπτυξη ενός προγράμματος. Το ένα μέλος της ομάδας, παίζει το ρόλο του «οδηγού» (driver) και έχει τον έλεγχο του μολυβιού/ποντικιού/πληκτρολογίου στη σχεδίαση και ανάπτυξη του προγράμματος, ενώ το δεύτερο μέλος είναι ο «παρατηρητής» (observer) που διαρκώς ελέγχει το έργο του «οδηγού» θέτοντας ερωτήσεις, διερευνώντας εναλλακτικές λύσεις, παρατηρώντας ελλείψεις, κ.λπ. Οι ρόλοι του «οδηγού» και του «παρατηρητή» εναλλάσσονται μεταξύ των δύο ατόμων και τα δύο μέλη συμμετέχουν ενεργά στη διαδικασία και είναι εξίσου υπεύθυνα για την επίτευξη του στόχου.

Πλεονεκτήματα της μεθόδου: την ικανοποίηση των μαθητών: οι μαθητές αισθάνονται μεγαλύτερη αυτοπεποίθηση για το αποτέλεσμα της εργασίας επειδή έχουν κάποιον να τους βοηθήσει και να τους υποδείξει ελλείψεις, αισθάνονται λιγότερη απογοήτευση και μεγαλύτερη ικανοποίηση επειδή η συνεργασία οδηγεί σε πιο αποτελεσματικές λύσεις αφιερώνοντας λιγότερο χρόνο. την ανάπτυξη δεξιοτήτων στην επίλυση προβλημάτων : οι μαθητές ανταλλάσσοντας ιδέες για την επίλυση ενός προβλήματος (είτε στη φάση της σχεδίασης είτε της υλοποίησης) και συζητώντας για τα πλεονεκτήματα/μειονεκτήματα της προτεινόμενης λύσης, μαθαίνουν να διερευνούν εναλλακτικές λύσεις και να επιλύουν προβλήματα που ενδεχομένως από μόνοι τους δε θα μπορούσαν.

Πλεονεκτήματα της μεθόδου: την ενίσχυση του μαθησιακού αποτελέσματος οι μαθητές, επειδή αναγκάζονται να στοχάζονται (reflection), να εξηγούν τις ενέργειες τους (self-explanations) και να αναπτύσσουν μηχανισμούς παρακολούθησης/ελέγχου (monitoring) της διαδικασίας και της προόδου της εργασίας, μαθαίνουν αποτελεσματικότερα και αναπτύσσουν δεξιότητες αυτο-αξιολόγησης και παρακολούθησης της προόδου τους την ανάπτυξη ικανοτήτων συνεργασίας οι μαθητές μαθαίνουν να επικοινωνούν και να συνεργάζονται αποτελεσματικά με άλλα άτομα αναπτύσσοντας την κοινωνικότητά τους.

Διαδικασία:

1. το ένα μέλος της ομάδας, παίζει το ρόλο του «οδηγού» (driver) και έχει τον έλεγχο του μολυβιού/ποντικιού/ πληκτρολογίου στην ανάπτυξη του προγράμματος
2. το δεύτερο μέλος είναι ο «παρατηρητής» (observer) που διαρκώς ελέγχει το έργο του «οδηγού» θέτοντας ερωτήσεις, διερευνώντας εναλλακτικές λύσεις, παρατηρώντας ελλείψεις, κ.λπ.
3. οι ρόλοι του «οδηγού» και του «παρατηρητή» εναλλάσσονται μεταξύ των δύο ατόμων.

Προσέγγιση βασισμένη στα παραδείγματα

Τόσο οι έμπειροι όσο και οι αρχάριοι προγραμματιστές χρησιμοποιούν για την επίλυση προβλημάτων σχετικά παραδείγματα που είτε τα έχουν διδαχθεί/διαβάσει είτε τα έχουν αναπτύξει οι ίδιοι. Τα παραδείγματα, συνοδευόμενα από τις επεξηγήσεις τους, χρησιμοποιούνται ιδιαίτερα από τους διδάσκοντες στη διδασκαλία των μαθημάτων προγραμματισμού Μία διδακτική προσέγγιση που βασίζεται σε παραδείγματα αφορά στη χρήση «ελάχιστων» παραδειγμάτων (minimal examples).

Προσέγγιση βασισμένη στα ελάχιστα παραδείγματα

«Ελάχιστα παραδείγματα»: Καθένα από τα παραδείγματα επικεντρώνεται σε μία συγκεκριμένη έννοια του προγραμματισμού ή

προγραμματιστική δομή και είναι μικρό σε έκταση. Τα βασικά συστατικά των «ελάχιστων» παραδειγμάτων είναι – μία ερώτηση/περιγραφή που χαρακτηρίζει το παράδειγμα – ο κώδικας του παραδείγματος που περιέχει τις σχετικές εντολές – το αποτέλεσμα της εκτέλεσης του κώδικα για κάποια είσοδο τιμών – σχόλια, επεξηγήσεις και επισημάνσεις σχετικά με την έννοια που παρουσιάζεται και τη λειτουργία των εντολών.

«Ελάχιστα παραδείγματα»: Το πλεονέκτημα των «ελάχιστων» παραδειγμάτων, σε σχέση με τα τυπικά παραδείγματα που υπάρχουν στα σχετικά συγγράμματα και ενδεχομένως χρησιμοποιούν οι διδάσκοντες στη διδασκαλία τους, είναι ότι δίνουν τη δυνατότητα στο μαθητή να επικεντρωθεί σε μία συγκεκριμένη έννοια, να εφαρμόσει γρήγορα και εύκολα το παράδειγμα και να πειραματιστεί με αυτό .

3. Εισαγωγή στα Εργαλεία Εκμάθησης

Αφού αναλύθηκαν οι μέθοδοι διδασκαλίας προγραμματισμού, μπορούμε να περάσουμε στα εργαλεία που έχει ο δάσκαλος στα χέρια του. Όπως αναφέρθηκε παραπάνω, τα εργαλεία ανάπτυξης που επιλέχθηκαν να

αναλυθούν είναι πέντε και είναι το KidsRuby, το Alice, το Scratch, το Kodu και τέλος το Greenfoot.

Στο παρόν κεφάλαιο θα αποκτήσουμε μία επαφή με το καθένα.

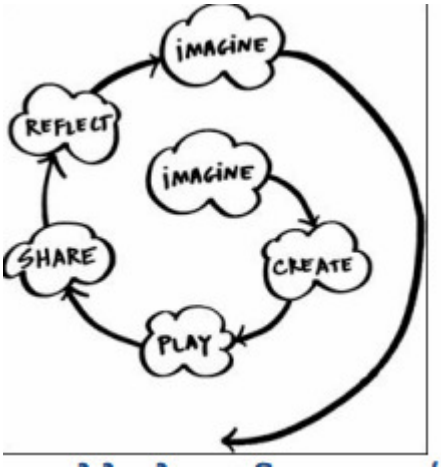
3.1 Scratch

Τι είναι η Scratch;

- Γλώσσα Προγραμματισμού
- Σχεδιασμένη για εκπαίδευση
    - Εύκολη στη χρήση της
    - Για όλες τις ηλικίες
    - Δεν απαιτείται προηγούμενη εξοικείωση με τον προγραμματισμό
- Οπτική, με χρώματα και γραφική
- Συνδέει μπλοκς για να φτιάξει προγράμματα
- Διασκεδαστική

Πού θα βρω τη Scratch;

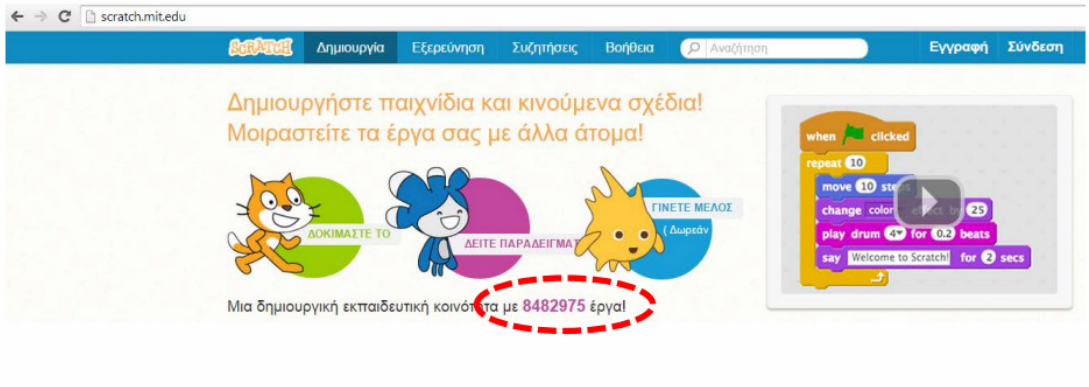
- Η Scratch αναπτύχθηκε από το MIT Media Lab.
- Είναι ανοιχτή για κατέβασμα από το Ίντερνετ. (https://**scratch**.mit.edu/)
- Είναι συμβατή με πολλά λειτουργικά συστήματα.
    - Windows (Windows 2000, XP, Vista)
    - Apple (MAC OSX 10.4 or later)
    - Linux (Ubuntu)



Το Scratch βασίστηκε στις σχεδιαστικές αρχές που παρουσιάζονται στο διπλανό σχήμα και χρησιμοποιείται για την εισαγωγή στον προγραμματισμό μαθητών ηλικίας από 10 έως 18 ετών (Resnick, 2007).
Οι μαθητές μπορούν να δημιουργήσουν προγράμματα, αλληλεπιδραστικές ιστορίες και παιχνίδια σέρνοντας και αφήνοντας εντολές που έχουν τη μορφή κομματιών πάζλ. Η αλληλεπίδραση και τα σενάρια που χρησιμοποιούνται στα παιχνίδια που δημιουργούνται παρέχουν κίνητρο στους μαθητές. Η πληθώρα διαθέσιμων χαρακτήρων (sprites), σκηνικών, ήχων και εικόνων παρέχουν κίνητρο στους μαθητές να φανταστούν (imagine), να δημιουργήσουν (create) τα δικά τους παιχνίδια και animations, να παίξουν (play), να μοιραστούν (share) τις δημιουργίες τους, να βασιστούν (reflect) στις δημιουργίες άλλων.

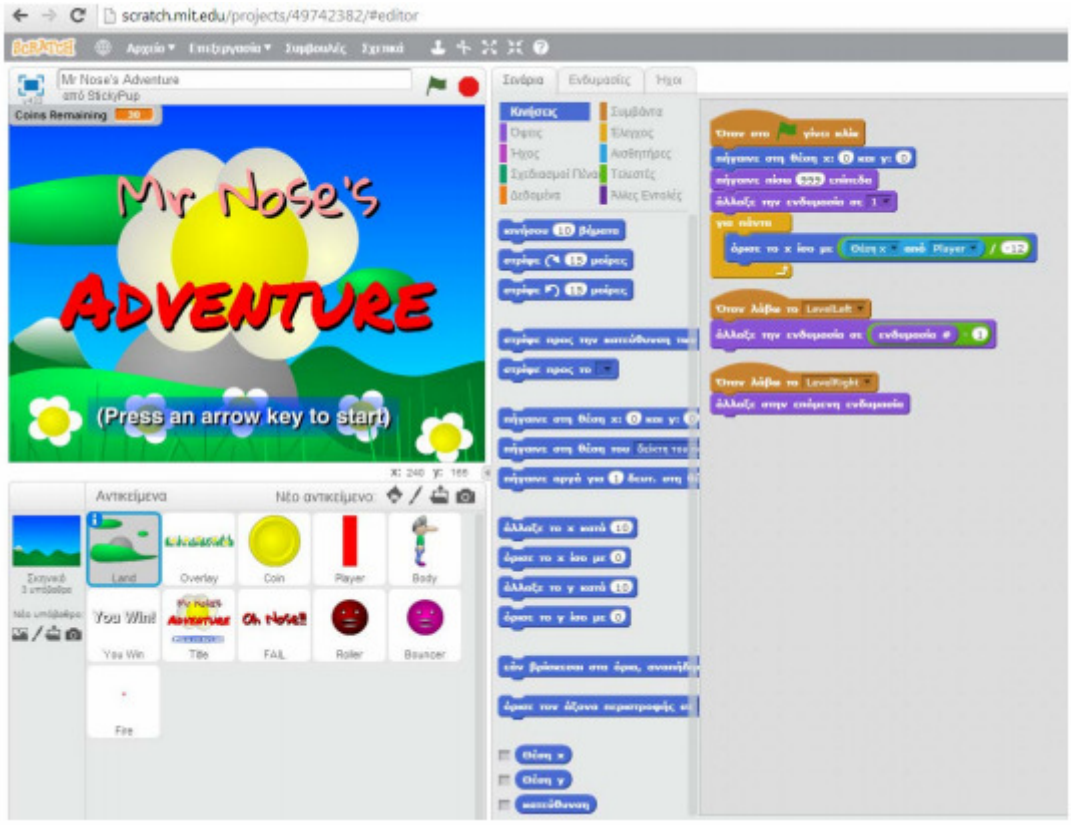
Εικόνα 1.1: Το σύμβολο της Scratch

Το Scratch έχει μια τεράστια κοινότητα χρηστών σε όλο τον κόσμο που μοιράζονται τα έργα τους μέσα από την επίσημη ιστοσελίδα του.



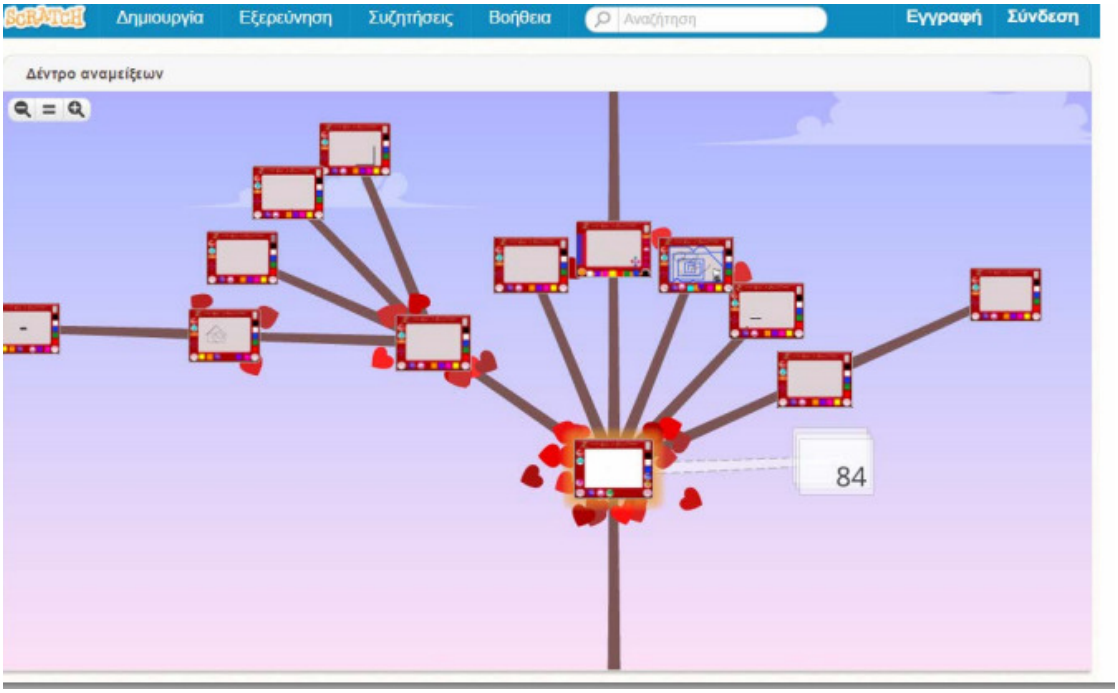
Εικόνα 1.2: Αρχική σελίδα Scratch

Το Scratch μπορεί να χρησιμοποιηθεί on-line χωρίς εγκατάσταση, αλλά μπορεί επίσης να εγκατασταθεί σε Mac, Windows, και μερικές διανομές των Linux (32 bit) για χρήση χωρίς σύνδεση στο Διαδίκτυο.



Εικόνα 1.3: Παράδειγμα Scratch

Στην on-line κοινότητα του Scratch υπάρχει διαθέσιμος ο κώδικας ενός τεράστιου πλήθους έργων. Τα μέλη της κοινότητας μπορούν να τροποποιήσουν υπάρχοντα έργα δημιουργώντας τη δική τους εκδοχή, διαδικασία γνωστή ως ανάμειξη (remixing). Για κάθε έργο υπάρχει δυνατότητα παρουσίασης του δέντρου αναμείξεων:



Εικόνα 1.4: Δένδρο αναμείξεων Scratch

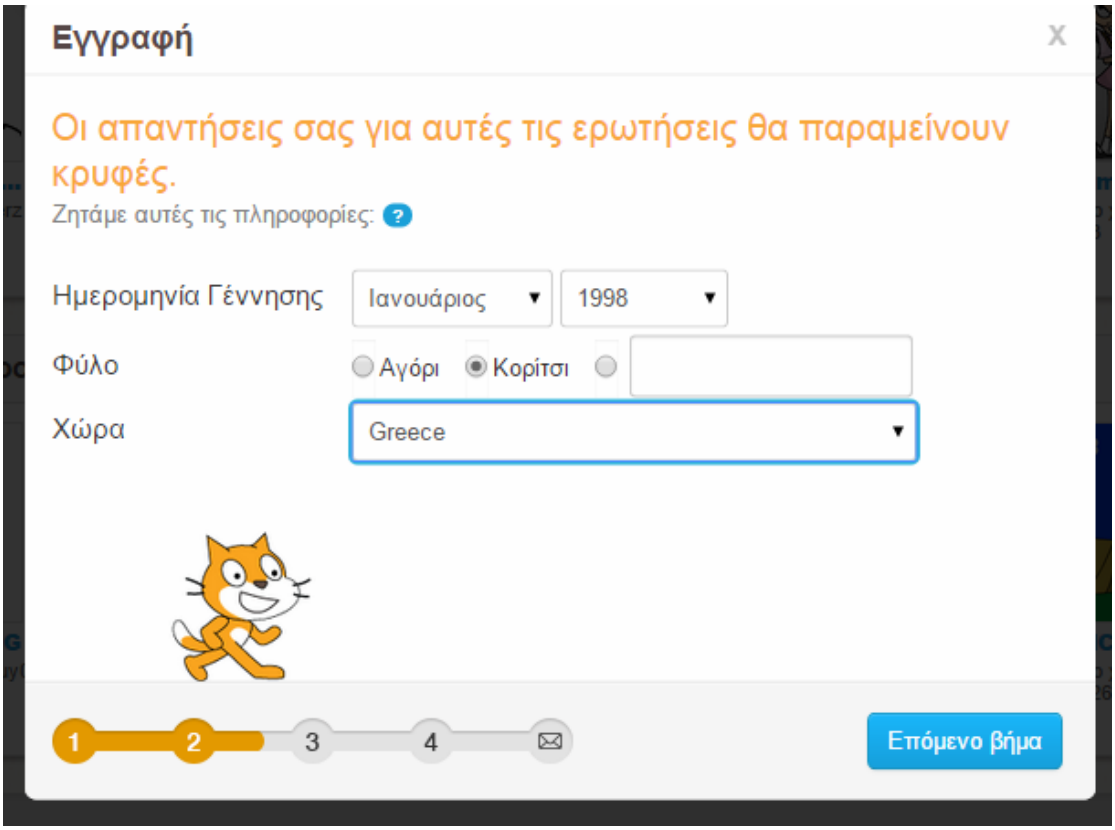
Εγχειρίδιο Χρήσης Scratch (Online)

Το Scratch μπορεί να το χρησιμοποιήσει κάποιος χρήστης είτε Online είτε ως εφαρμογή (offline). Θα δούμε παρακάτω τα βήματα χρήσης του Scratch online.

1. Εγγραφή χρήστη

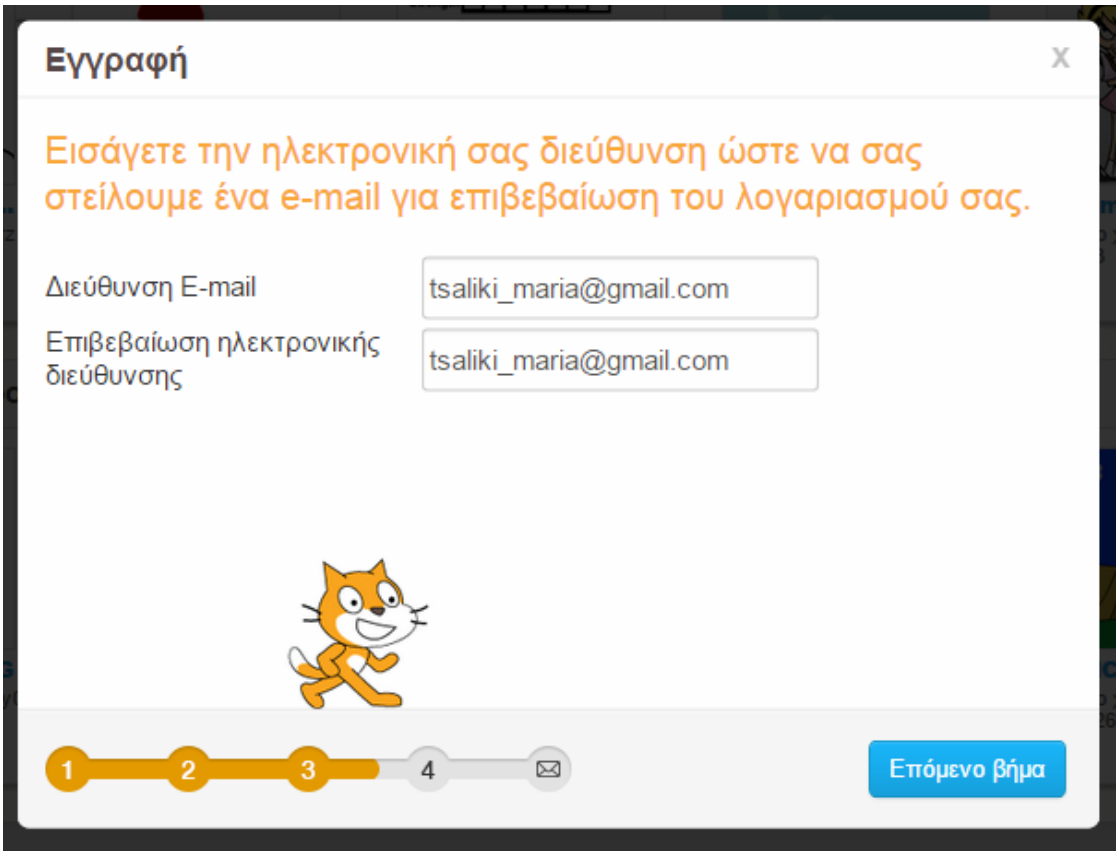
Εικόνα 1.6: Εγγραφή στο Scratch

Με μια πολύ απλή πλατφόρμα εγγραφής ο χρήστης καλείται να εισάγει το όνομα χρήστη που επιθυμεί και τον κωδικό πρόσβασης του.
Στη συνέχεια ο χρήστης καλείται να εισάγει κάποιες βασικές προσωπικές πληροφορίες, οι οποίες όμως δεν θα φαίνονται στους άλλους χρήστες.



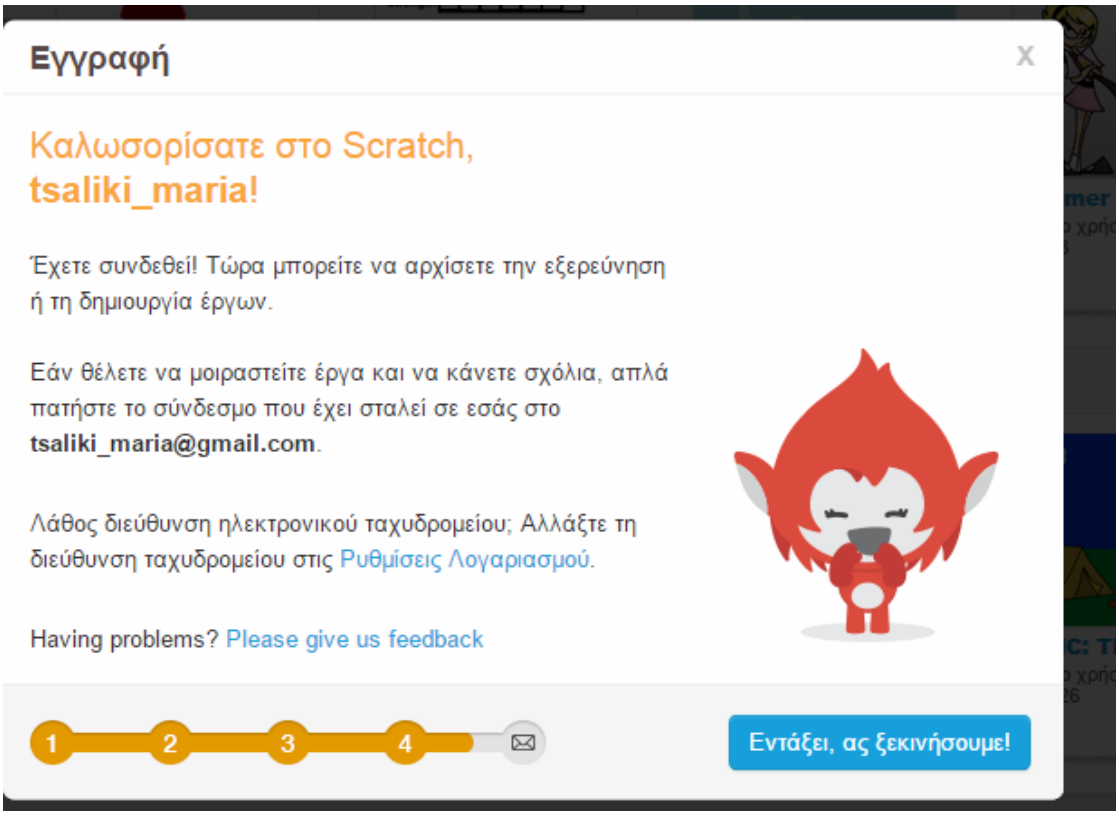
Εικόνα 1.7: Εγγραφή στο Scratch

Απαραίτητη είναι η εισαγωγή ηλεκτρονικής διεύθυνσης ταχυδρομείου για τη χρήση του online scratch.



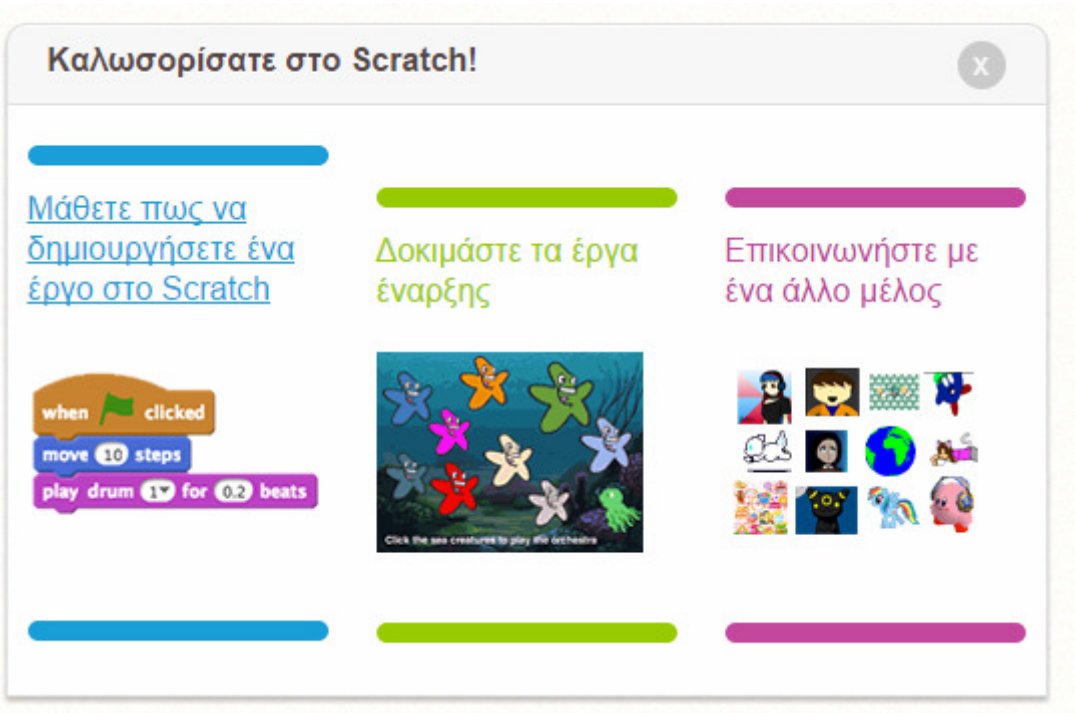
Εικόνα 1.8: Εγγραφή στο Scratch

Ο χρήστης είναι πλέον έτοιμος να ξεκινήσει!



Εικόνα 1.9: Εγγραφή στο Scratch

Στο σημείο αυτό ο χρήστης έχει ένα μενού για να επιλέξει το τι θέλει να κάνει όπως φαίνεται παρακάτω στην εικόνα:

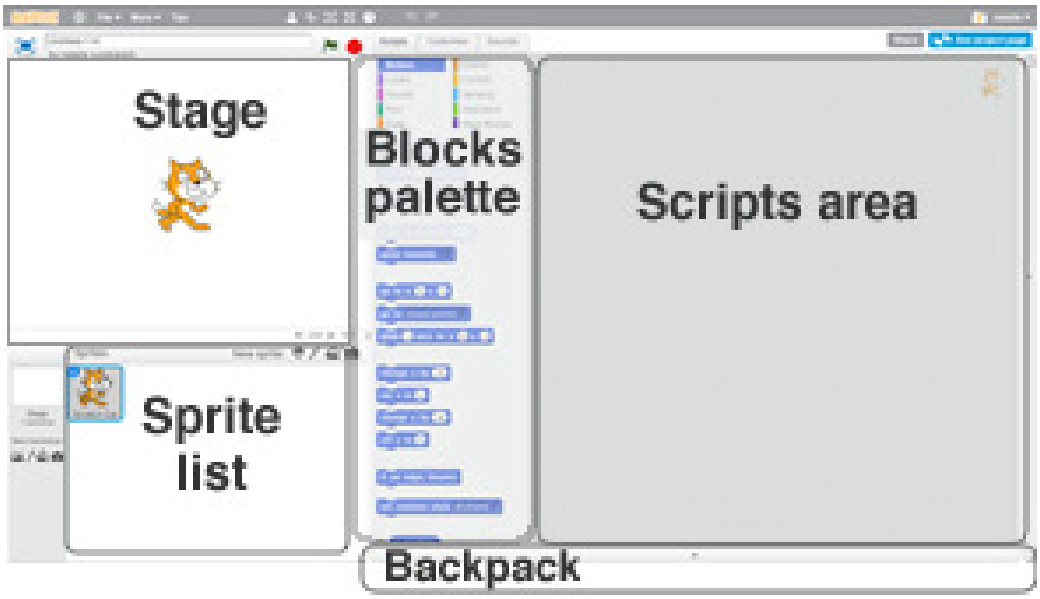


Εικόνα 1.10: Μενού Επιλογών Scratch

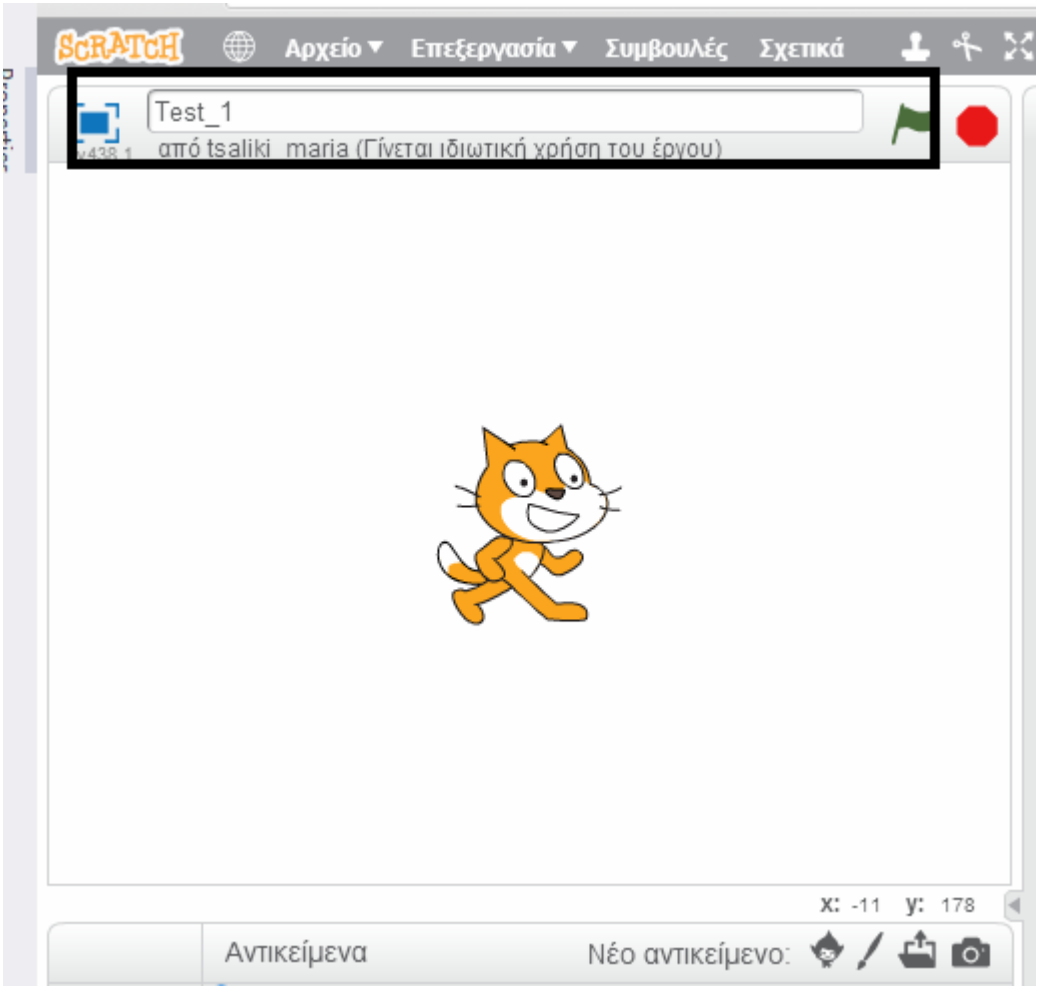
Εκτός από πολύ οργανωμένο, το Scratch έχει το πλεονέκτημα ότι είναι στα Ελληνικά. Είναι από τα λίγα εκπαιδευτικά περιβάλλοντα που είναι διαθέσιμα στην Ελληνική γλώσσα.

Εμείς σε αυτό το σημείο θα επιλέξουμε να “Μάθουμε πώς να δημιουργήσουμε ένα έργο στο Scratch”.

Αρχικά καλό είναι να δούμε το χάρτη της σελίδας που θα χρησιμοποιήσουμε έτσι ώστε να γνωρίζουμε τις περιοχές της.



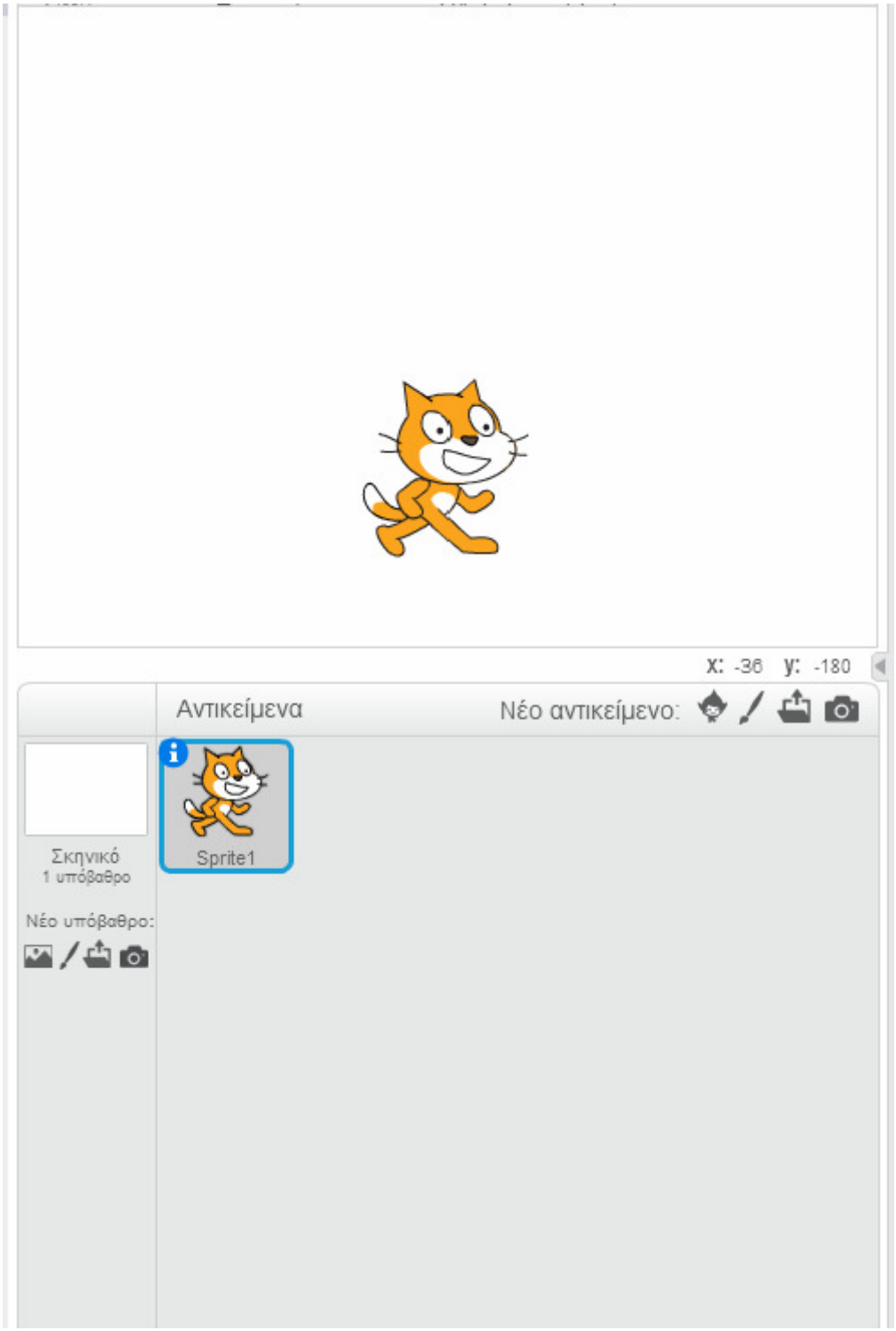
Εικόνα 1.11: Scratch χάρτης περιοχών



Εικόνα 1.12: Αλλαγή Τίτλου Έργου

Επιλέγοντας αυτό το σύνδεσμο θα οδηγηθούμε σε μία νέα σελίδα που έχει το καινούργιο έργο. Στην εικόνα παραπάνω απεικονίζεται το σημείο που μπορούμε να αλλάξουμε τον τίτλο του έργου.

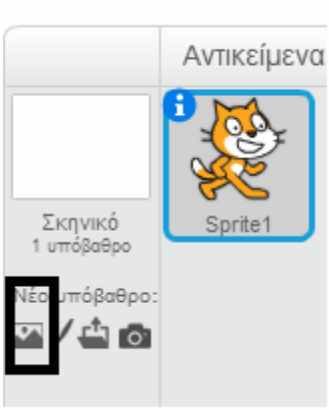
Στην αριστερή πλευρά της σελίδας που έχουμε μεταφερθεί μπορούμε να δούμε την παρακάτω εικόνα:



Εικόνα 1.13: Έργο και επιλογές

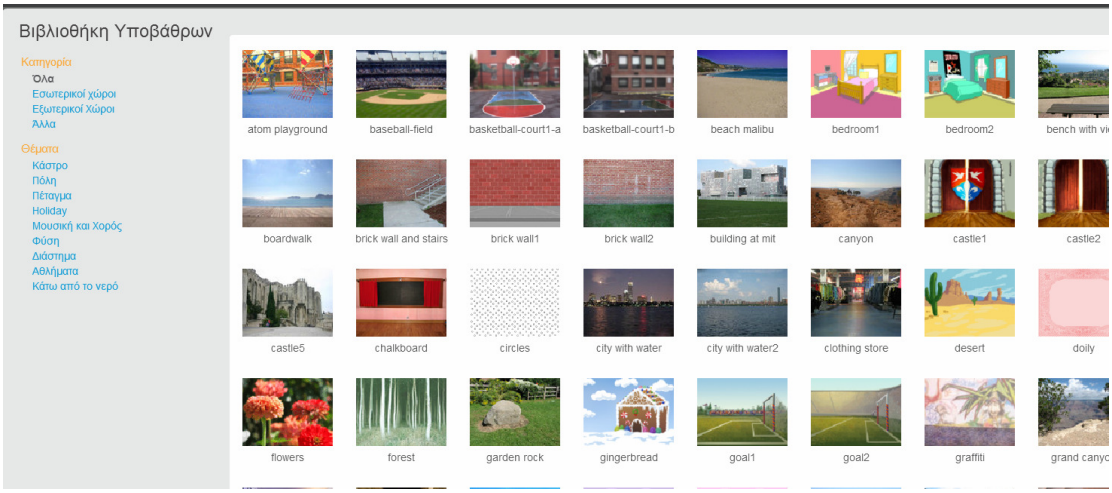
Το επάνω παράθυρο απεικονίζει το τρέχον στιγμιότυπο του έργου μας, ενώ το κάτω τα Sprites (τους χαρακτήρες) που χρησιμοποιούμε για το έργο και το background (το υπόβαθρο).

Μέχρι στιγμής έχουμε ένα χαρακτήρα σε ένα λευκό υπόβαθρο. Αυτό όμως δεν είναι πολύ όμορφο. Ας προσπαθήσουμε να αλλάξουμε το υπόβαθρο του έργου μας. Αν παρατηρήσουμε κάτω από το υπόβαθρο υπάρχουν κάποια κουμπιά. Ας δούμε τι δυνατότητες δίνουν αυτά.
Επιλέγοντας το πρώτο κουμπί θα ανοίξει μία βιβλιοθήκη από εικόνες για να αλλάξουμε το φόντο του παιχνιδιού.



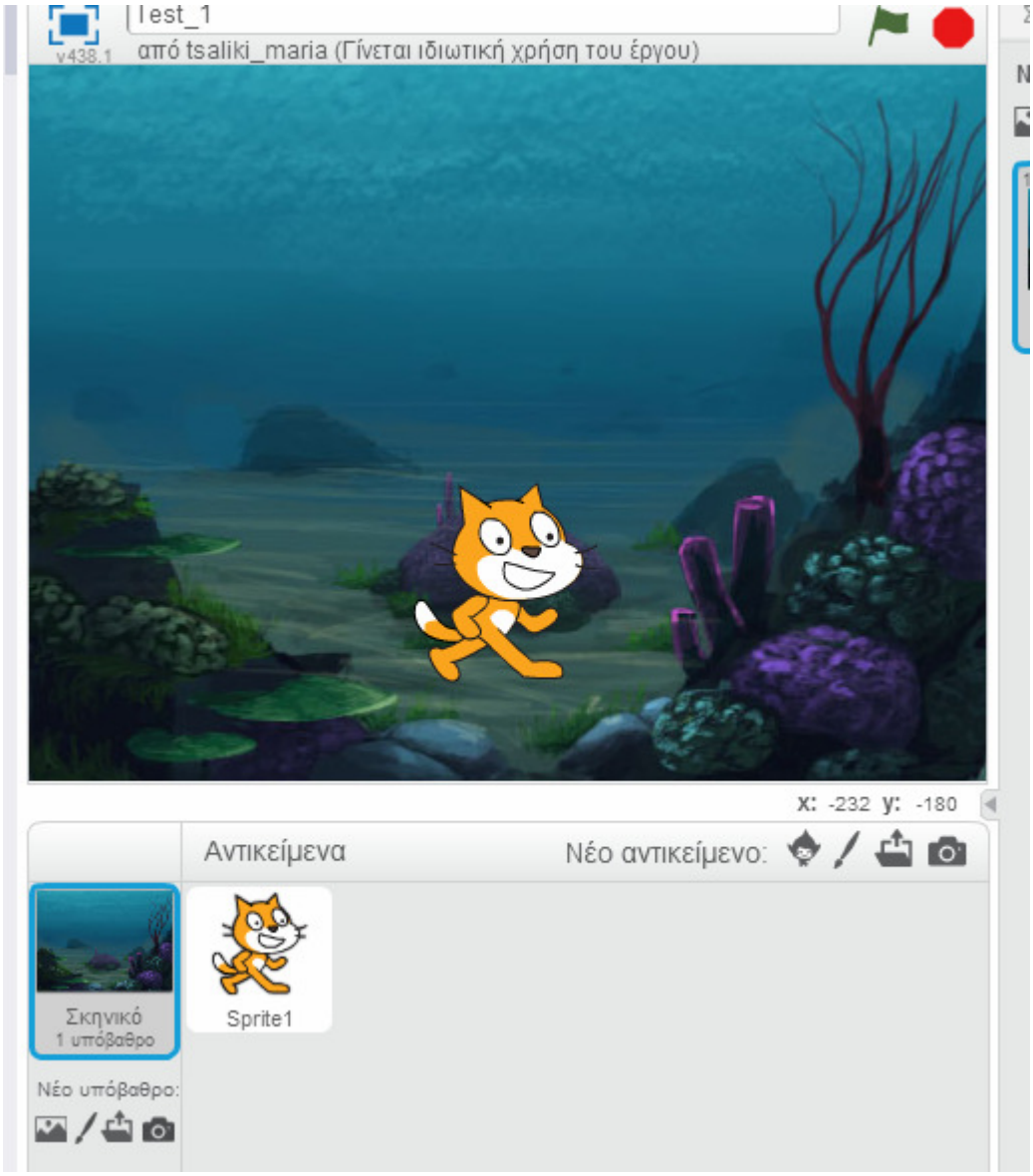
Εικόνα 1.14: Αλλαγή φόντου έργου

Οι εικόνες είναι ταξινομημένες ανά θεματική κατηγορία και μπορούμε να διαλέξουμε οποιαδήποτε εικόνα θέλουμε για φόντο που θεωρούμε ότι θα κάνει περισσότερο ενδιαφέρον το έργο μας.



Εικόνα 1.15: Βιβλιοθήκη Υπόβαθρων

Επιλέγοντας κάποια από τις παραπάνω εικόνες μπορούμε να δούμε μια αλλαγή στο έργο μας. Έστω λοιπόν ότι επιλέξαμε την underwater1, θα δούμε το έργο μας να γίνεται ως ακολούθως:



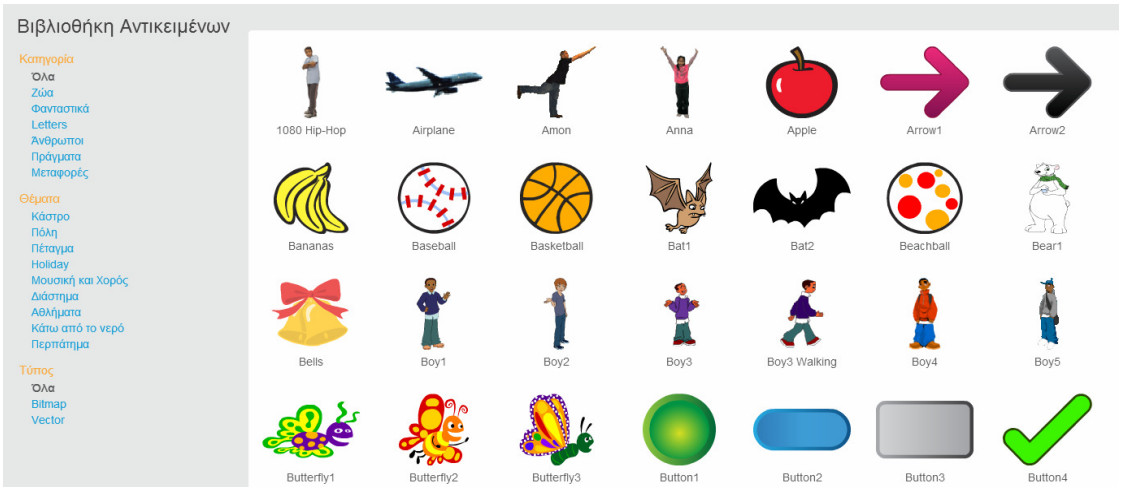
Εικόνα 1.16: Αποτέλεσμα αλλαγής φόντου

Παρομοίως, μπορούμε να προσθέσουμε ένα νέο αντικείμενο ή να αλλάξουμε το υπάρχον, ή ακόμα και να το διαγράψουμε με τις επιλογές που παρέχει το πάνελ αντικειμένων, όπως απεικονίζεται παρακάτω:

Εικόνα 1.17: Εισαγωγή χαρακτήρα

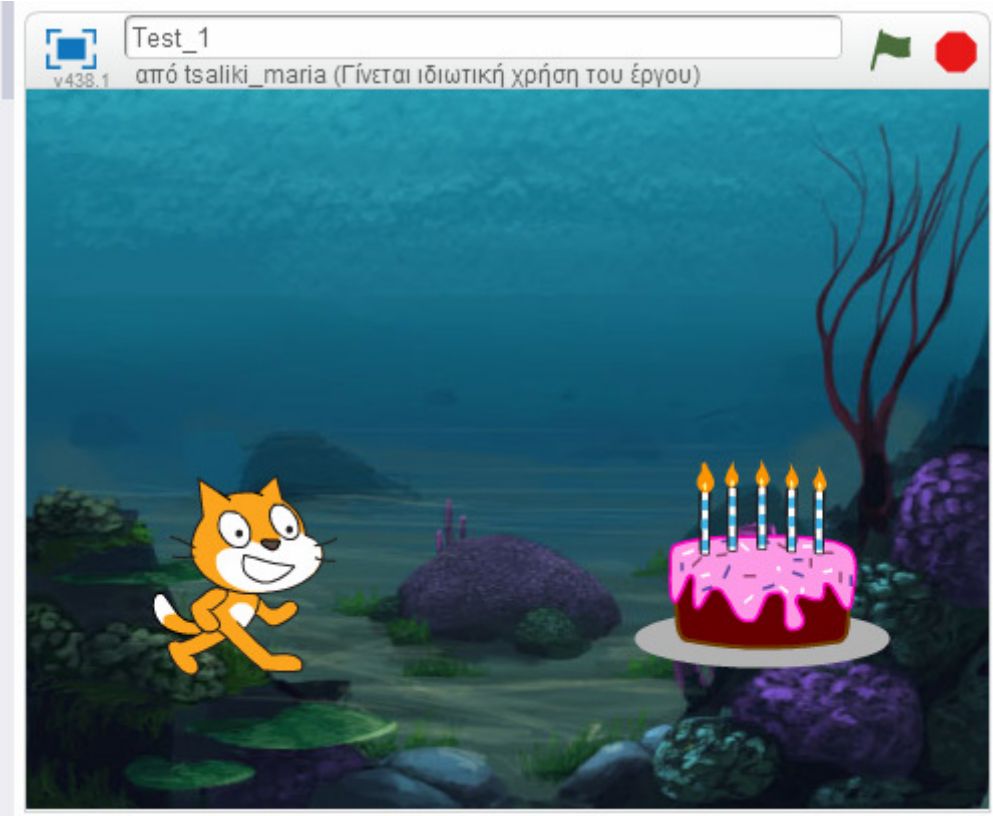
Έστω ότι θέλουμε να προσθέσουμε ένα καινούργιο αντικείμενο. Εάν κάνουμε κλικ στην πρώτη επιλογή του μενού της εικόνας παραπάνω θα ανοίξει η βιβλιοθήκη αντικειμένων, απ' όπου μπορούμε να επιλέξουμε τι αντικείμενο θέλουμε να προσθέσουμε.

Και πάλι, τα αντικείμενα είναι ταξινομημένα ανά θεματική κατηγορία.



Εικόνα 1.18: Βιβλιοθήκη Αντικειμένων

Έστω ότι επιλέξαμε το cake. Θα δούμε το έργο μας να αλλάζει λοιπόν ως ακολούθως:



Εικόνα 1.19: Αποτέλεσμα προσθήκης αντικειμένου.

Μπορούμε να καθορίσουμε την θέση των αντικειμένων κάνοντας κλικ πάνω σε αυτά και τραβώντας τα με το ποντίκι στη θέση που επιθυμούμε.
Μπορούμε πλέον να δούμε πώς δημιουργούνται σενάρια για το έργο μας. Όπως απεικονίζεται στην εικόνα 1.11 (χάρτης περιοχών Scratch) υπάρχει μία περιοχή που ονομάζεται Scripts Area. Αυτό είναι το σημείο όπου θα εμφανίζονται όλα τα σενάρια που αφορούν το έργο που κατασκευάζει ο χρήστης. Δηλαδή η αντιδράσεις των χαρακτήρων και του υπόβαθρου θα καθορίζονται από τα σενάρια που θα τοποθετηθούν σε αυτή την περιοχή. Για να προσθέσουμε ένα σενάριο στο έργο μας μπορούμε να επιλέξουμε από την στήλη των Κινήσεων, των Ενδυμασιών και των Ήχων. Κάθε καρτέλα που επιλέγει ο χρήστης παρέχει μία λίστα επιλογών τις οποίες πολύ απλά μπορεί να σύρει με τον ποντίκι του στην περιοχή των script και να τα θέσει σε ενεργεία.
Επίσης μπορεί να κάνει παραλλαγές σε κάποιο σενάριο. Για παράδειγμα το σενάριο «Κινήσου 10 βήματα», το νούμερο 10 είναι ανοιχτό στο χρήστη για να το επεξεργαστεί.

Με τον τρόπο αυτό μπορούμε να φτιάξουμε σενάρια για τους χαρακτήρες και το υπόβαθρο.
Τέλος, μπορούμε να μοιραστούμε το έργο μας με άλλα μέλη του Scratch πατώντας το κουμπί «Κάντε κοινή χρήση του έργου».

Αποτελέσματα

Προκειμένου να διερευνηθούν οι αντιλήψεις των εκπαιδευτικών για κάθε τύπο δραστηριοτήτων σχεδιάστηκε το παρακάτω ερωτηματολόγιο:

1. Ο τύπος αυτής της δραστηριότητας μάθησης μου άρεσε
2. Ο τύπος αυτής της δραστηριότητας μάθησης θεωρώ ότι θα είναι αποδοτικός και ενδιαφέρων για τους μαθητές μου
3. Ο τύπος αυτής της δραστηριότητας μάθησης θεωρώ ότι θα δυσκολέψει τους μαθητές
4. Οταν εμπλέξω τους μαθητές μου σε δραστηριότητες στο Scratch θα χρησιμοποιήσω αυτό τον τύπο δραστηριότητας
5. Τα πιο σημαντικά πλεονεκτήματα αυτού του τύπου δραστηριότητας είναι:
6. Τα πιο σημαντικά μειονεκτήματα αυτού του τύπου δραστηριότητας είναι:
7. Θα χρησιμοποιήσω αυτό τον τύπο δραστηριότητας στη διδασκαλία μου κατά τη φάση της: (α) Εισαγωγής στο μάθημα (ΕΙΣ), (β) Εμπέδωσης του μαθήματος (ΕΜΠ), (γ) Αξιολόγησης του μαθήματος (Α), (δ) Ως εργασία για το σπίτι (Ε-Σ).
8. Δεν θα χρησιμοποιήσω (Δ-Χ) αυτό τον τύπο δραστηριότητας στη διδ/λία μου διότι:

Οι 5 από τις παραπάνω ερωτήσεις είναι κλειστού τύπου ενώ οι υπόλοιπες 3 είναι ανοικτού τύπου. Οι 4 από τις κλειστές ερωτήσεις (βλέπε ερωτήσεις 1, 2, 3, και 4) θα έπρεπε να απαντηθούν μέσω μιας 5-βαθμης κλίμακας Likert (ΔΡ:Διαφωνώ ριζικά, 8 ο 6 Πανελλήνιο Συνέδριο με Διεθνή Συμμετοχή Δ:Διαφωνώ, Α:Αδιαφορος, Σ:Συμφωνώ, ΣΑ:Συμφωνώ απόλυτα) ενώ η 5η ερώτηση (βλέπε ερώτηση 7) είναι πολλαπλής επιλογής.

Δραστηριότητες – Ερωτήσεις	Α π α

	ντήσεις				
Κατηγορίες δραστηριοτήτων – Ερωτήσεις που τέθηκαν	ΔΡ	Δ	Α	Σ	ΣΑ
(i) Ελεύθερες δημιουργικές δραστηριότητες					
Ο τύπος αυτής της δραστηριότητας μάθησης μου άρεσε		1	1	16	2
Ο τύπος αυτής της δραστηριότητας μάθησης θεωρώ ότι θα είναι αποδοτικός και ενδιαφέρων για τους μαθητές μου		1	2	14	3
Ο τύπος αυτής της δραστηριότητας μάθησης θεωρώ ότι θα δυσκολέψει τους μαθητές μου		9	3	7	3
Οταν εμπλέξω τους μαθητές μου σε δραστηριότητες στο Scratch θα χρησιμοποιήσω αυτό τον τύπο δραστηριότητας	1	1	2	12	4
Θα χρησιμοποιήσω αυτό τον τύπο δραστηριότητας στη διδασκαλία μου κατά τη φάση της:	ΕΙΣ: 9, ΕΜΠ: 4,			Α: 2, Ε-Σ: 5	
(ii) Επίλυση συγκεκριμένου προβλήματος					
Ο τύπος αυτής της δραστηριότητας μάθησης μου άρεσε			1	14	5
Ο τύπος αυτής της δραστηριότητας		1	2	14	3

μάθησης θεωρώ ότι θα είναι αποδοτικός και ενδιαφέρων για τους μαθητές μου					
Ο τύπος αυτής της δραστηριότητας μάθησης θεωρώ ότι θα δυσκολέψει τους μαθητές μου	2	5	6	5	2
Οταν εμπλέξω τους μαθητές μου σε δραστηριότητες στο Scratch θα χρησιμοποιήσω αυτό τον τύπο δραστηριότητας			1	15	4
Θα χρησιμοποιήσω αυτό τον τύπο δραστηριότητας κατά τη φάση της					
(iii) Δραστηριότητες πολλαπλών λύσεων	ΕΙΣ: 3, ΕΜΠ: 13		Α: 3, Ε-Σ:1		
Ο τύπος αυτής της δραστηριότητας μάθησης μου άρεσε			1	14	5
Ο τύπος αυτής της δραστηριότητας μάθησης θεωρώ ότι θα είναι αποδοτικός και ενδιαφέρων για τους μαθητές μου			1	14	5
Ο τύπος αυτής της δραστηριότητας μάθησης θεωρώ ότι θα δυσκολέψει τους μαθητές μου		2	4	12	2
Οταν εμπλέξω τους μαθητές μου σε δραστηριότητες στο Scratch θα χρησιμοποιήσω αυτό τον τύπο δραστηριότητας	1	1		14	4
Θα χρησιμοποιήσω αυτό τον τύπο δραστηριότητας στη διδασκαλία μου κατά τη φάση της:	ΕΙΣ: 1, ΕΜΠ: 9			Α: 5, Ε-ΣΠ: 4, Δ-	

				X:1	

Στον Πίνακα 1 παρατίθενται οι κατηγορίες από τις δραστηριότητες που προαναφέρθηκαν και οι κλειστές ερωτήσεις που τέθηκαν (στήλη 1) και οι συχνότητες απαντήσεων (ΔΡ, Δ, Α, Σ, ΣΑ) των εκπαιδευτικών σε αυτές (στήλες 2, 3, 4, 5 και 6). Οι απαντήσεις, και οι αντίστοιχες συχνότητες στην ερώτηση πολλαπλής επιλογής αναφέρονται επίσης στον Πίνακα 1 (γραμμές, 8, 14 και 20). Στη συνέχεια παρατίθενται οι απαντήσεις που δόθηκαν στις ανοικτού τύπου ερωτήσεις από τους εκπαιδευτικούς (οι αριθμοί μέσα στις παρενθέσεις αναφέρονται στον αριθμό των εκπ/κών που υποστήριξαν την κάθε συγκεκριμένη άποψη). (i) Ελεύθερες δημιουργικές δραστηριότητες Πλεονεκτήματα: Ελευθερία επιλογής θέματος, σεναρίου και δυσκολίας επίλυσης από το μαθητή (5), κεντρίζει τη φαντασία και τη δημιουργικότητα των μαθητών (16), απευθύνεται σε όλους τους μαθητές (1), μπορούν να χρησιμοποιηθούν διαφορετικές γνώσεις σε συνδυασμό, ανάληψη της πρωτοβουλίας, της υπευθυνότητας και ανάπτυξη της κριτικής Οι ΤΠΕ στην Εκπαίδευση 7 σκέψης των μαθητών (1), δημιουργία κινήτρου για μάθηση (4), ευχαρίστηση των μαθητών από το αποτέλεσμα της κατασκευής τους (1), επικοινωνία γνώσης μεταξύ των μαθητών (1). Μειονεκτήματα: έλλειψη εστίασης σε συγκεκριμένους μαθησιακούς στόχους (3), κάποιοι μαθητές (ενδεχομένως οι πιο αδιάφοροι) μπορεί να καταβάλλουν την ελάχιστη προσπάθεια (6), δεν μπορούμε να ελέγξουμε την κατανόηση συγκεκριμένων εννοιών (4), οι μαθητές μπορούν να χαθούν και να ξεφύγουν όταν δεν υπάρχουν συγκεκριμένοι στόχοι (3), οι μαθητές δυσκολεύονται όταν δεν τους δίνονται οδηγίες (3), η εξεύρεση λύσεων ποικίλει από τις πολύ απλές έως τις πολύ σύνθετες (3), είναι χρονοβόρες (3) δύσκολο να ασχοληθεί ένας εκπαιδευτικός με όλους τους μαθητές και τις διάφορες λύσεις που προτείνουν (1), δύσκολο για τους μαθητές να βρουν μια ιδέα και να ξεκινήσουν (1). (ii) Επίλυση συγκεκριμένου προβλήματος Πλεονεκτήματα: Υπάρχει ένα συγκεκριμένο πλαίσιο για τους μαθητές (7), υπάρχουν συγκεκριμένοι μαθησιακοί στόχοι που θέλουμε να πετύχουμε (16), επιτυγχάνεται η μάθηση συγκεκριμένης γνώσης (4), οι μαθητές δεν χάνονται (8), ο μαθητής

κατευθύνεται για να σκεφτεί κάτι (1), συγκράτηση του μαθήματος και της τάξης σε συγκεκριμένα πλαίσια (1). Μειονεκτήματα: Η συγκεκριμένη δραστηριότητα μπορεί να είναι αδιάφορη για τους μαθητές (10), περιορίζει τη φαντασία, τη σκέψη, την πράξη και τη δημιουργικότητα των μαθητών σε προκαθορισμένα πλαίσια της μιας ορθής λύσης (7), οι μαθητές δεν μπορούν να δημιουργήσουν τις δικές τους λύσεις (5), η επιτυχία της εξαρτάται από την καταλληλότητά της (1), απευθύνεται κάθε φορά σε μαθητές ενός συγκεκριμένου επιπέδου ικανοτήτων και γνώσεων (3), εμμονή στη μια και ορθή λύση (1). (iii) Δραστηριότητες πολλαπλών λύσεων Πλεονεκτήματα: Ενθάρρυνση της δημιουργικότητας των μαθητών (6), είναι δυνατόν να συμμετέχουν όλοι οι μαθητές (5), στοχεύει στη μάθηση συγκεκριμένων γνώσεων (5), μπορούν να δημιουργηθούν λύσεις διαφόρων επιπέδων ανάλογα με το επίπεδο και τη φαντασία των μαθητών (9), είναι καθορισμένη αλλά και ελεύθερη (7), δημιουργία χωρίς άγχος να βρεθεί η μοναδική ορθή λύση (2), μοιάζει με παιχνίδι (1), ικανοποιούνται οι μαθητές μέσα από την πρωτοτυπία των λύσεων (4), δίνει δυνατότητες στους μαθητές να φτάσουν σε ένα προχωρημένο επίπεδο γνώσης και να εξελιχθούν (4), οι μαθητές μπορούν να χρησιμοποιήσουν πολλαπλά εργαλεία (1), δίνεται μεγάλη ελευθερία στους μαθητές να δημιουργήσουν (4), απευθύνεται σε μαθητές διαφόρων γνωστικών επιπέδων (3), δεν περιορίζονται οι καλοί μαθητές αλλά δεν αποκλείονται οι πιο αδύνατοι (3), είναι ευέλικτες (1), μπορούν να χρησιμοποιηθούν για εμπέδωση των γνώσεων των μαθητών (5). Μειονεκτήματα: Είναι δύσκολο στους μαθητές να φτάσουν σε πολύπλοκες λύσεις (5), το να κατασκευάσει ένας μαθητής πολλαπλές λύσεις είναι σύνθετη, χρονοβόρα και δύσκολη δουλειά (3), οι αδύνατοι μαθητές μπορεί να μην μπορούν να κατασκευάσουν σύνθετες λύσεις (1), μπορεί το σενάριο της δραστηριότητας να μην κεντρίζει αρκετά τους μαθητές (1).

3. 2 Kids Ruby

Το KidsRuby, όπως δηλώνει και το όνομά του, είναι ένα εργαλείο το οποίο δίνει τη δυνατότητα σε ένα παιδί να μάθει γρήγορα και εύκολα μία γλώσσα προγραμματισμού πολύ διαδεδομένη, την γλώσσα Ruby.

Η Ruby είναι μία δυναμική γλώσσα προγραμματισμού, ανοικτού κώδικα, που επικεντρώνονται στην απλότητα και την παραγωγικότητα. Διαθέτει μία εύκολη σύνταξη, γεγονός που την κάνει εύκολη στο να διαβαστεί και να γραφεί.

Λίγα λόγια για τη Ruby

Από τη στιγμή που προτείνεται μέσω της παρούσας εργασίας ένα εργαλείο με το οποίο τα παιδιά θα είναι σε θέση να μάθουν Ruby, αξίζει να αναφέρουμε λίγα λόγια σχετικά με αυτή τη γλώσσα.

Το λογότυπο της Ruby είναι ένα κόκκινο ρουμπίνι και απεικονίζεται παρακάτω:

Η Ruby προήλθε από την Ιαπωνία στα μέσα της δεκαετίας τους 1990 και αρχικά σχεδιάστηκε και αναπτύχθηκε από τον Yukihiro "Matz" Matsumoto. Θα μπορούσαμε να πούμε πως επηρεάστηκε κυρίως από την Perl και την Lisp, που είναι επίσης διαδεδομένες γλώσσες προγραμματισμού.

Η γλώσσα αυτή υποστηρίζει πολλούς τύπους προγραμματισμού, όπως είναι ο **αντικειμενοστραφής** προγραμματισμός (πρόκειται για μία μεθοδολογία ανάπτυξης προγραμμάτων, όπου ο χειρισμός σχετιζόμενων δεδομένων και των διαδικασιών που επενεργούν σε αυτά γίνεται από κοινού, μέσω μίας δομής δεδομένων που τα περιβάλλει ως αυτόνομη οντότητα με ταυτότητα και δικά της χαρακτηριστικά. Αυτή η δομή δεδομένων καλείται *αντικείμενο* και αποτελεί πραγματικό στιγμιότυπο στη μνήμη ενός σύνθετου, και πιθανώς οριζόμενου από τον χρήστη, τύπου δεδομένων, που ονομάζεται κλάση), ο **προστακτικός** προγραμματισμός (το ζητούμενο κατασκευάζεται / υπολογίζεται αλλάζοντας την κατάσταση του υπολογιστή μέσω εντολών), ο **συναρτησιακός** (που αντιμετωπίζει τον υπολογισμό ως την αποτίμηση

μαθηματικών συναρτήσεων και αποφεύγει την κατάσταση προγράμματος και τα μεταβλητά δεδομένα) και ο **ανακλαστικός**.

Αυτή τη γλώσσα προγραμματισμού, λοιπόν, έχει ως στόχο να διδάξει στα παιδιά το kidsRuby.

Εάν κάποιος επισκεφτεί την επίσημη ιστοσελίδα του KidsRuby, (http://kidsruby.com/) θα δει ένα ευχάριστο περιβάλλον που σε προδιαθέτει θετικά και παρακινεί το ενδιαφέρον ενός παιδιού.

Η αρχική σελίδα της ιστοσελίδας μοιάζει όπως η εικόνα 3.1.3.

Με πολύ απλό τρόπο μπορεί κάποιος να κατεβάσει το KidsRuby από την ιστοσελίδα που προαναφέρθηκε. Η εγκατάσταση του περιβάλλοντος είναι εξίσου εύκολη. Απλά τρέχουμε το αρχείο (.exe) που κατέβηκε.

Αφού τρέξουμε το αρχείο και εγκατασταθεί το περιβάλλον, μπορούμε πλέον να αποκτήσουμε πρόσβαση στο περιβάλλον του KidsRuby.

Το περιβάλλον είναι ιδιαίτερα εύκολο και η αρχική σελίδα του είναι αυτή της εικόνας 3.1.4.

Στα δεξιά του αρχικού μενού που απεικονίζεται στην εικόνα 3.1.4, υπάρχει ένα μαύρο πλαίσιο, το οποίο είναι για να πληκτρολογείται ο κώδικας.

Μενού Start Here

Το πρώτο μενού του KidsRuby, είναι το λεγόμενο “Start Here”.

Πατώντας σε αυτό το μενού, μπορούμε να δούμε κάποιες γενικές πληροφορίες για το τι είναι η Ruby, τι γλώσσα προγραμματισμού είναι και πότε αναπτύχθηκε. Μπορούμε εύκολα να αλλάζουμε σελίδες πατώντας το Next, ή το Back, όπως φαίνεται πάνω στην εικόνα 3.1.5.

Μόλις τελειώσουν οι σελίδες από την εισαγωγή στη Ruby, το KidsRuby μας συμβουλεύει να πατήσουμε το link “Back to Lessons”, όπου θα βρούμε τα μαθήματα που θα εκπαιδεύσουν το παιδί στη Ruby.

Μενού Keyboard Shortcuts

Πατώντας σε αυτό το μενού, το “Keyboard Shortcuts”, μπορούμε να δούμε έναν πίνακα με όλες τις συντομεύσεις πληκτρολογίου που υπάρχουν στο KidsRuby, ανάλογα πάντα με το λειτουργικό σύστημα στο οποίο είναι εγκατεστημένο (Windows / Mac). Για παράδειγμα, πατώντας ταυτόχρονα Ctrl + U, μπορούμε να αλλάξουμε τη γραμματοσειρά από μικρά σε κεφαλαία.

Μία απεικόνιση αυτού του μενού φαίνεται στην εικόνα 3.1.6.

Μενού Hackety-Hack

Το μενού αυτό, είναι εκείνο που μας συμβουλεύει το τέλος του “Start Here” να επισκεφτούμε. Το Hackety – Hack είναι το πρώτο μάθημα που θα κάνει ένα παιδί στο KidsRuby.

Το KidsRuby, δεν έχει ως στόχο μόνο να διδάξει τη σύνταξη της γλώσσας Ruby στα παιδιά, αλλά κυρίως να μεταδώσει το βασικό και γενικό νόημα του προγραμματισμού.

Χαρακτηριστικά αναφέρει στα μαθήματα ότι όλη η ιδέα του προγραμματισμού είναι οι αλγόριθμοι και παρακινεί το παιδί να φτιάχνει, όπως λέμε “to-do list”, προσπαθεί δηλαδή να αναπτύξει στα παιδιά την αλγοριθμική σκέψη.

Όταν, λοιπόν το παιδί ανοίξει το μενού Hackety – Hack, έχει τρεις επιλογές.

Basic Programming

Basic Ruby

Turtle Colors

Ξεκινώντας από την πρώτη επιλογή το “Basic Programming”, το παιδί θα διδαχθεί πώς να φτιάξει ένα απλό τετράγωνο.

Το εντυπωσιακό όμως στο εργαλείο αυτό για κάποιον που ξέρει προγραμματισμό, είναι ότι το πρώτο πράγμα που κάνει είναι να παρακινεί το παιδί να πάρει χαρτί και μολύβι και να σχεδιάσει ένα τετράγωνο μία προς μία πλευρά.

Αυτό γιατί, και ο κώδικας που θα πληκτρολογεί το παιδί θα εμφανίζει σταδιακά μία προς μία πλευρά. Και αυτή είναι η πρώτη και σημαντική επαφή με την αλγοριθμική σκέψη που αναφέρθηκε παραπάνω.

Σταδιακά, προτείνεται από το KidsRuby στον χρήστη να πληκτρολογήσει κομμάτια κώδικα. Η οθόνη στην οποία γράφουμε τον κώδικα μοιάζει όπως αυτή της εικόνας 3.1.7:

Ο κώδικας που θα πληκτρολογήσει το παιδί φαίνεται σε μία οθόνη στα αριστερά.

Καθώς το παιδί αντιγράφει τον κώδικα που του προτείνεται και τρέχοντας το κάθε σημείο του προγράμματος θα βλέπει και μία έξοδο. Η έξοδος αυτή είναι σταδιακά μία προς μία πλευρά του τετραγώνου μέχρι να ολοκληρωθεί το τετράγωνο και να δούμε το αποτέλεσμα της εικόνας 3.1.9.

Στο σημείο αυτό θα έχει ολοκληρωθεί το πρώτο μάθημα σε Ruby καθώς και η πρώτη επαφή με τον προγραμματισμό και την αλγοριθμική σκέψη.

Στη συνέχεια ο χρήστης μπορεί να συνεχίσει στη δεύτερη επιλογή που παρέχει το μενου Hackety – Hack, το “Basic Ruby”.

Φυσικά, όπως κάθε αρχάριος προγραμματιστής και εδώ το πρώτο παράδειγμα που θα δοκιμάσει ο χρήστης είναι το να εκτυπώσει το περίφημο

“Hello World!”. Έτσι, αυτή τη φορά καθοδηγείται το παιδί να πληκτρολογήσει το εξής:

```
alert "Hello, world!"
```

Η εντολή alert είναι ίσως από τις πιο συχνά χρησιμοποιημένες στην Ruby και είναι η εντολή που εκτυπώνει ένα λεκτικό σε ένα παράθυρο. Εάν κάποιος τρέξει αυτή την εντολή θα πάρει το αποτέλεσμα της εικόνας 3.1.10.

Σε αυτό το σημείο είναι καλό να αναφέρουμε το ρόλο της καρτέλας “Output”, που μπορούμε να δούμε στην εικόνα 3.1.5.

Η καρτέλα αυτή υπάρχει για να εκτυπώνονται τυχόν Errors – σφάλματα που μπορεί να βρεθούν στον κώδικα. Βέβαια, για ένα παιδί είναι μάλλον εξαιρετικά δύσκολο το να καταλάβει τα σφάλματα του κώδικα, αλλά είναι μια επαφή με την έννοια της αποσφαλμάτωσης κώδικα (debugging), που είναι απαραίτητη γνώση για έναν άρτιο προγραμματιστή.

Προχωρόντας το μάθημα, το μέγιστο σημείο που μπορεί να φτάσει ένα παιδί είναι σε ένα σημαντικό σημείο, αυτό της πραγματικής επικοινωνίας ανθρώπου υπολογιστή. Το παιδί δηλαδή θα μάθει ότι ο χρήστης μπορεί να δίνει εισαγωγή δεδομένων στον υπολογιστή και αυτός με τη σειρά του να τα αποθηκεύει και να τα επεξεργάζεται ώστε να δώσει μία έξοδο. Μαθαίνει δηλαδή το ακόλουθο βασικό μοντέλο:

Είσοδος -> Επεξεργασία -> Έξοδος

Αυτό θα γίνει πληκτρολογώντας τις ακόλουθες γραμμές κώδικα:

```
name =      "What is your name?"
alert "Hello, " + name
```

Αξιοσημείωτο είναι το γεγονός ότι η Ruby είναι μία γλώσσα πολύ διαφορετική από τις πλέον πολύ διαδεδομένες τύπου Java, C# κ.ό.κ. Οι διαφορές είναι πολλές και βασικές, αλλά αξίζει να αναφέρουμε τουλάχιστον ότι η Ruby δεν έχει τύπους μεταβλητών και δηλώσεις μεταβλητών (βλέπουμε για παράδειγμα το name, που είναι μεταβλητή και δεν προυπάρχει κάποια δήλωση της) και κυρίως οι εντολές δεν ξεχωρίζουν μεταξύ τους με το ερωτηματικό (;).

Τρέχοντας το παραπάνω κομμάτι κώδικα βλέπουμε το αποτέλεσμα που φαίνεται στην εικόνα 3.1.1.

Εάν ο χρήστης πληκτρολογήσει λοιπόν το όνομά του, (π.χ “George”) θα δούμε το παρακάτω:

Τέλος, το παιδί σε αυτό το μάθημα μπορεί να αποκτήσει και μια πρώτη επαφή με την δομή if – else.

Η τελευταία επιλογή του μενού Hackety – Hack είναι το “Turtle Colors”, το οποίο είναι ένας πίνακας από βασικά χρώματα που μπορούν να εμφανιστούν στην καρτέλα Turtle, με βάση το δείκτη **R**ed **G**reen **B**lue. Η επιλογή αυτή φαίνεται στην εικόνα 3.1.13.

Εκτός από το Hackety – Hack υπάρχουν και άλλα μαθήματα στα επόμενα μενού, τα οποία λειτουργούν με την ίδια λογική που περιγράψαμε παραπάνω.

Ολοκληρώνοντας τα μαθήματα του KidsRuby το παιδί θα έχει πια αποκτήσει μία επαφή με την αλγοριθμική σκέψη και τον προγραμματισμό σε Ruby.

3.3 Alice

Το Alice είναι ένα πρωτοποριακό 3D περιβάλλον προγραμματισμού που κάνει εύκολο να δημιουργήσετε μια κινούμενη εικόνα για την αφήγηση

μιας ιστορίας, ένα διαδραστικό παιχνίδι, ή ένα βίντεο για να μοιραστείτε στο διαδίκτυο.

Το Alice είναι ίσως από τα πιο ενδιαφέροντα εργαλεία που βρέθηκε κατά τη διάρκεια της έρευνας, γιατί αφορά τον 3D σχεδιασμό, ένα θέμα που έχει στρέψει τα βλέματα της τεχνολογίας επάνω του. Μπορούμε να σκεφτούμε πόση προσπάθεια καταβάλεται για τη δημιουργία τρισδιάστατων ταινιών, και τη συρροή κόσμου που προκαλλούν εκείνες, τρισδιάστατων παιχνιδιών.

Άλλωστε, μία από τις πιο ενδιαφέρουσες καινοτομίες είναι η εικονική πραγματικότητα η οποία στηρίζεται κατά πολύ μεγάλο βαθμό στο σχεδιασμό τρισδιάστατων αντικειμένων.

Το Alice χρησιμοποιεί κατά κόρον αντικειμενοστραφή προγραμματισμό, δηλαδή γλώσσες όπως Java, C++ και C#.

Το Alice, είναι ένα εργαλείο για τρισδιάστατους σχεδιασμούς αντικειμένων και κατασκευή ταινιών, βίντεο και παιχνιδιών.

Η εγκατάστασή του είναι σχετικά απλή. Εάν επισκεφτούμε την επίσημη ιστοσελίδα του Alice (http://www.alice.org/) μπορούμε πολύ εύκολα να κατεβάσουμε το Alice, στην έκδοση που επιθυμούμε. Στο αρχείο που θα κατέβει μπορούμε να βρούμε το εκτελέσιμο αρχείο του περιβάλλοντος.

Το πρώτο πράγμα που χρειάζεται όταν τρέξουμε το εκτελέσιμο αρχείο (.exe) του Alice, είναι να καθορίσουμε τις ρυθμίσεις του. Για να συμβεί αυτό, πρέπει να ανοίξουμε την καρτέλα Preferences.

Το πρόγραμμα εγκατάστασης Alice έχει ένα προκαθορισμένο σύνολο προτιμήσεων για το "look and feel" του περιβάλλοντος Alice, για το πώς θα εμφανίζεται δηλαδή το περιβάλλον. Αυτά είναι οι προεπιλεγμένες προτιμήσεις. Οι προεπιλεγμένες προτιμήσεις εμφανίζονται στην εικόνα 3.2.1. Από προεπιλογή, σχεδόν όλες οι επιλογές στις προτίμησεις είναι απενεργοποιημένες (που υποδεικνύεται από την έλλειψη ενός σήματος ελέγχου για τα στοιχεία προτίμησης από το μενού). Ωστόσο, τα δύο πρώτα στοιχεία Gallery προτίμησης ενεργοποιημένη (υποδεικνύεται με ένα σημάδι επιλογής στην αρχή του στοιχείου στο μενού).

Με τις προεπιλεγμένες ρυθμίσεις προτίμησης, το Alice ξεκινά με το πρόγραμμα επεξεργασίας κώδικα να εμφανίζει μόνο μία καρτέλα, όπως φαίνεται. Η καρτέλα είναι χώρος editor (επεξεργαστή) για τη δημιουργία κώδικα σε κάποια myFirstMethod, μια μέθοδο που ανήκει στην σκηνή. Όταν ο κώδικας έχει δημιουργηθεί και ο χρήστης κάνει κλικ στο κουμπί Εκτέλεση, η σκηνή θα εμφανίσει ένα αναδυόμενο παράθυρο (το παράθυρο runtime) και ο κώδικας της myFirstMethod θα εκτελεστεί.

Για να ορίσουμε μια προτίμηση, μπορούμε να ανοίξουμε την καρτέλα Window στη γραμμή μενού και να επιλέξουμε Preferences. Στη συνέχεια, να κάνουμε κλικ σε μία από τις προτιμήσεις του μενού. Η εικόνα που ακολουθεί (εικόνα 3.2.3) δείχνει ότι έχει συμπεριληθφεί η προτίμηση "*Include Type Decoration*" . Μια προτίμηση ενεργοποιείται όταν ένα σημάδι ελέγχου εμφανίζεται αμέσως προς τα αριστερά του μενού.

Οποιοσδήποτε συνδυασμός των προτιμήσεων μπορεί να καθοριστεί. Ένα προτεινόμενο σύνολο των προτιμήσεων φαίνεται παρακάτω για όσους επιθυμούν να επικεντρωθούν σε αντικειμενοστραφή έννοιες του προγραμματισμού με σκοπό να προετοιμαστούν για μια γλώσσα προγραμματισμού, όπως η Java.

Διαφορετικές ρυθμίσεις προτίμησης οδηγούν σε μια διαφορετική εμφάνιση και αίσθηση για το περιβάλλον Alice. Για παράδειγμα, οι ρυθμίσεις που εμφανίζονται στην προηγούμενη εικόνα θα έχει ως αποτέλεσμα στην οθόνη που φαίνεται παρακάτω. Οι δύο καρτέλες θα εμφανίζονται αυτόματα στο πρόγραμμα επεξεργασίας κώδικα, καθώς και ένα κουμπί επιλογής κλάσης.

Το κουμπί επιλογής κλάσης είναι ένα κίτρινο κουμπί που ονομάζεται “Scene” και διαθέτει ένα κάτω μαύρο βέλος που αποκαλύπτει ότι πατώντας το θα αναδειχθεί ένα μενού. Όταν λοιπόν πατηθεί αυτό το κουμπί, θα εμφανιστεί ένα μενού με όλες τις κλάσεις που χρησιμοποιούνται. Στην εικόνα 3.2.7 φαίνεται ένα χαρακτηριστικό παράδειγμα.

Εάν επιλέξουμε μία κλάση θα ανοίξει μία καινούργια καρτέλα στον επεξεργαστή κώδικα. Αυτή η καρτέλα εμφανίζει ένα σχήμα των μεθόδων που ανήκουν στην κλάση. Εάν για παράδειγμα κάποιος επιλέξει την κλάση “Alien” θα δει το αποτέλεσμα της εικόνας 3.2.8.

Οι μέθοδοι που αναφέρονται σε μια καρτέλα κλάσης υποδιαιρούνται σε τρεις ομάδες (διαδικασίες, λειτουργίες, και ιδιότητες), που εμφανίζονται στον πίνακα Μέθοδοι. Επιπλέον, εμφανίζεται και μια τέταρτη ομάδα, που ονομάζεται κατασκευαστές (Constructors). Ένας κατασκευαστής είναι ένα ιδιαίτερο είδος της μεθόδου που περιέχει οδηγίες για τη δημιουργία ενός νέου αντικειμένου, όπως ορίζεται από αυτή στην κλάση.

Αυτό το σχήμα δίνει τη δυνατότητα σε ένα παιδί να έρθει σε επαφή με αυτό που ονομάζουμε διάγραμμα UML μίας κλάσης και σκιαγραφεί το βασικό

σχήμα της κλάσης. Μπορούμε σε αυτό το σημείο να αναφέρουμε ένα πολύ απλό παράδειγμα κλάσης σε Java.

```
package grammateia;

public class Student {

	String name;
	String surname;
	int id;
	Subject aSubject;

	//Constructor
	/**/
	public Student(String name,String surname, int id)
	{
		this.name = name;
		this.surname = surname;
		this.id = id;



	}

	public String getName()
	{
		return name;
	}
	public String getSurname()
	{
		return surname;
	}
	public int getId()
```

```
        {
                return id;
        }

        public void setSubject(int id,String name,char examine)
        {
                this.id = id;
                aSubject = new Subject (name,examine);
        }
        public void setSubjectWithGrade(int id,String name,char
examine,double grade)
        {
                this.id =id;
                aSubject= new Subject (name,examine,grade);
        }
}
```

Εικόνα 3.2.9: Παράδειγμα Κλάσης Java

Στο παράδειγμα της εικόνας 3.2.9, η κλάση ονομάζεται Student και έχει κατασκευαστή το ακόλουθο τμήμα κώδικα:

```
public Student(String name,String surname, int id)
        {
                this.name = name;
                this.surname = surname;
                this.id = id;


        }
```

Δηλαδή για να κατασκευάσει κάποιος ένα στιγμιότυπο της κλάσης Student (ένα μαθητή) θα πρέπει να δηλώσει όνομα, επίθετο και id.

```
Student ma8itis = new Student("George","Papadopoulos",1);
```

Αντίστοιχα, οι ιδιότητες της κλάσεις είναι οι μεταβλητές που έχουν δηλωθεί δηλαδή το ακόλουθο τμήμα κώδικα:

```
String name;
String surname;
int id;
Subject aSubject;
```

Και οι μέθοδοι το υπόλοιπο τμήμα κώδικα της εικόνας 3.2.10.

Αυτές οι δύο πλατφόρμες, το Alice και το KidsRuby, είναι πλατφόρμες που στοχεύουν στην εκμάθηση γλωσσών προγραμματισμού (Java και Ruby) και είναι σχεδιασμένες για desktop. Ενδιαφέρον παρουσιάζουν όμως και οι τρόποι εκμάθησης γλωσσών προγραμματισμού σε tablets, όπως το iPad.

3.4 Kodu για Η/Υ

Το Kodu για Η/Υ είναι μία δωρεάν οπτική γλώσσα προγραμματισμού η οποία ξεκίνησε να αναπτύσσεται το 2010. Αρχικά ήταν διαθέσιμη στο Xbox αλλά στη συνέχεια εξελίχθηκε και για Η/Υ. Η γλώσσα αυτή, είναι κατασκευασμένη έτσι ώστε να δώσει τη δυνατότητα στα παιδιά να αναπτύξουν δεξιοτήτες στον προγραμματισμό.

Το Kodu προσπαθεί να επικεντρώνεται σε παιχνίδια έτσι ώστε να κρατάει το ενδιαφέρον των μαθητών σε υψηλά επίπεδα.

Φυσικά υπάρχουν και άλλα εργαλεία εκμάθησης, αλλά επιλέχθηκαν τα παραπάνω ως πιο διαδεδομένα.

Το Kodu είναι μία εφαρμογή για desktop που αναπτύχθηκε από τη Microsoft. Είναι από τις λίγες εφαρμογές που έχουν επιλογή γλώσσας Ελληνικά. Επίσης, συγκριτικά με όλες τις υπόλοιπες εφαρμογές που αναλύονται στην παρούσα πτυχιακή εργασία έχει την καλύτερη υποστήριξη και τα περισσότερα εγχειρίδια για να μάθει ο χρήστης να το χρησιμοποιεί.

Τι ακριβώς όμως είναι το Kodu;

Το Kodu είναι ένα εργαλείο με σκοπό την ανάπτυξη παιχνιδιών, που παρέχει τη δυνατότητα στο χρήστη, εκτός από το να δημιουργεί τα δικά του παιχνίδια να μπορεί να τα μοιράζεται με φίλους. Τα κύρια σημεία είναι:

Ένας επεξεργαστής κάποιας τρισδιάστατης σκηνής, για να ξεκινήσει η δημιουργία του περιβάλλοντος του παιχνιδιού

Ένα νέο οπτικό προγραμματιστικό σύστημα που επιτρέπει στο χρήστη να δημιουργεί μοναδικές συμπεριφορές διαδραστικών παιχνιδιών

Ένα σύστημα κοινοποίησης για να ανταλλάσσονται παιχνίδια μεταξύ των χρηστών

Το Kodu περιλαμβάνει κάποια πρότυπα παιχνίδια που έχουν ως στόχο να εκπαιδεύσουν τον χρήστη και να του δείξουν τι ικανότητες έχει αυτό το εργαλείο. Θα δούμε στη συνέχεια κάποια από αυτά.

Στην εικόνα 3.3.2 μπορούμε να δούμε την Αρχική Σελίδα που εμφανίζεται στην έναρξη της εφαρμογής.

Το πολύ σημαντικό πλεονέκτημα που παρέχει το kodu είναι ότι έχει ενσωματωμένο στα βασικά μενού του το μενού “Community” ή αλλιώς “Κοινότητα”.

Το μενού αυτό περιέχει τα Μαθήματα που θα χρειαστεί κάποιος για να μάθει να χειρίζεται το Kodu.

Εάν λοιπόν επιλέξουμε το πρώτο μάθημα, μπορούμε να δούμε την παρακάτω οθόνη (εικόνα 3.3.3).

Ο χρήστης δηλαδή, έχει μία βασική τρισδιάστατη σκηνή η οποία θα αποτελέσει και τη βάση του παιχνιδιού του. Στο κάτω μέρος της οθόνης βέβαια, υπάρχει μία μπάρα με διάφοερες επιλογές. Έτσι ο χρήστης μπορεί να επεξεργαστεί την τρισδιάστατη σκηνή λειαίνοντας το δάπεδο, προσθέτωντας πέτρες, δέντρα και άλλα βασικά αντικείμενα.

Η μπάρα αυτή παρέχει διάφορες επιλογές. Εάν ο χρήστης για παράδειγμα επιλέξει την πρώτη επιλογή (χέρι εικόνα 3.3.4) θα μπορεί να μετακινεί την κάμερα με το ποντίκι του. Να αλλάζει δηλαδή οπτική γωνία με την οποία βλέπει τη σκηνή.

Εάν ο χρήστης επιλέξει το εικονίδιο με το σήμα του Kodu, (εικόνα 3.3.5) μπορεί να προσθέσει ή να επεξεργαστεί χαρακτήρες και αντικείμενα στη σκηνή. Δηλαδή μπορεί για παράδειγμα να προσθέσει ένα δέντρο, μία πέτρα κ.ό.κ.

Το αντικείμενο που θέλει να προσθέσει μπορεί να το επιλέξει από ένα διαδραστικό μενού με εικόνες και χρώματα όπως φαίνεται στην εικόνα 3.3.6. Αφού το επιλέξει, μπορεί να το προσθέσει με ένα απλό κλικ στην τρισδιάστατη σκηνή, όπου εκείνος επιθυμεί.

Παρόμοια, εάν επιλέξουμε το τρίτο εικονίδιο (εικόνα 3.3.7) , μπορούμε να προσθέσουμε στην σκηνή μονοπάτια και δρόμους και να επιλέξουμε το

χρώμα αυτών από μία μπάρα με χρώματα που εμφανίζεται στο επάνω μέρος της οθόνης (εικόνα 3.3.8).

Επιλέγοντας το εικονίδιο της βούρτσας (εικόνα 3.3.9), μπορούμε να μεγαλώσουμε τη σκηνή προσθέτωντας έδαφος, αλλά ακόμα και να τη μικρύνουμε εάν το επιθυμούμε διαγράφοντάς το.

Επιπλέον, το επόμενο ακριβώς εικονίδιο δίπλα στη βούρτα (εικόνα 3.3.10) υπάρχει για να προσθέτει λόφους ή κοιλάδες στην τρισδιάστατη σκηνή. Ο χρήστης μόλις το επιλέξει μπορεί κάνοντας κλικ στο επιθυμητό σημείο της σκηνής να τοποθετήσει εκεί ένα λόφο ή μία κοιλάδα.

Η επόμενη επιλογή υπάρχει για τον ακριβώς αντίθετο σκοπό. Δηλαδή, πατώντας το προηγούμενο κουμπί προσθέσαμε στη σκηνή ένα λόφο ή μία κοιλάδα. Πατώντας αυτό το κουμπί (εικόνα 3.3.10), το έδαφος γίνεται ομαλό.

Στην εικόνα 3.3.12, απεικονίζεται το κουμπί που δίνει τη δυνατότητα στο παιδί να κάνει αγκαθωτό το έδαφος.

Το Kodu παρέχει τόσες πολλές επιλογές για το χρήστη, και κάνει την σκηνή τόσο διαδραστική που φυσικά οι επιλογές δεν περιορίζονται σε αυτό το σημείο. Το εικονίδιο της εικόνας 3.3.13, υπάρχει για να προστεθεί στη σκηνή νερό, δηλαδή το παιδί μπορεί να φτιάξει θάλασσες, ποτάμια, λίμνες κ.ά.

Μπορούμε με τις παραπάνω επιλογές να φτιάξουμε μία σκηνή όπως αυτή της εικόνας 3.3.14 με μονοπάτια, λίμνες, νερά κλπ.

Με το εικονίδιο της εικόνας 3.3.15 μπορούμε να διαγράψουμε αντικείμενα από τη σκηνή και με αυτό της εικόνας 3.3.16 μπορούμε να αλλάξουμε τις ρυθμίσεις.

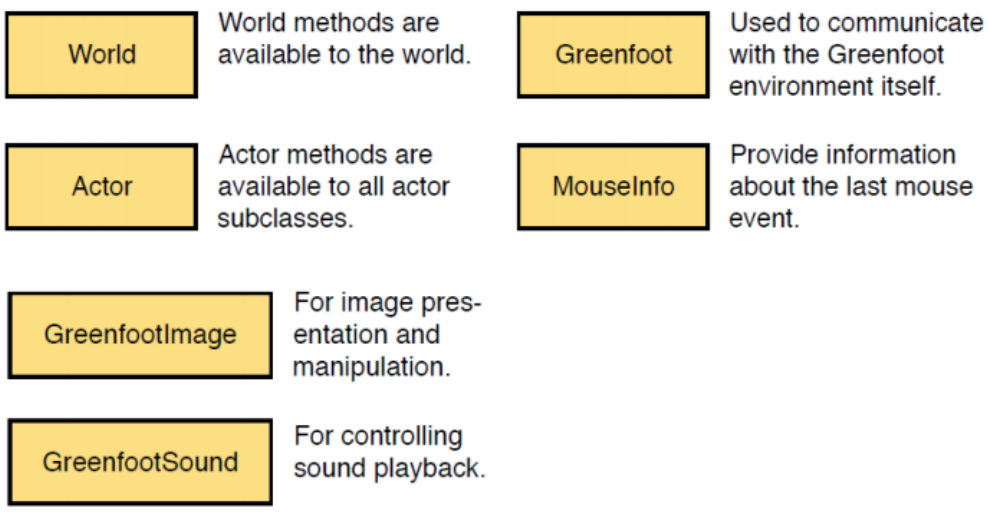
Τέλος το σπιτάκι (εικόνα 3.3.17) θα επαναφέρει τον χρήστη στο αρχικό μενού, και το κουμπί Play (εικόνα 3.3.18) σημαίνει ότι ο χρήστης μπορεί να παίξει στη σκηνή που έφτιαξε!

3.5 Greenfoot

Το Greenfoot είναι ένα εκπαιδευτικό περιβάλλον προγραμματισμού που μπορεί να χρησιμοποιηθεί για τη δημιουργία παιχνιδιών και προσομοιώσεων. Στο περιβάλλον του Greenfoot απλοποιείται

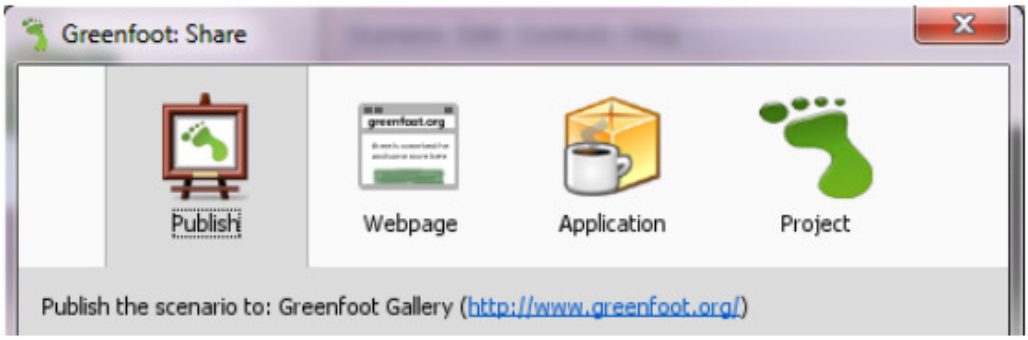
- η δημιουργία της γραφικής διασύνδεσης και
- ο προγραμματισμός της συμπεριφοράς των χαρακτήρων ενός παιχνιδιού/προσομοίωσης που δημιουργούνται ως υποκλάσεις των υπαρχόντων κλάσεων World και Actor αντίστοιχα (οι οποίες είναι διαθέσιμες σε κάθε νέο project που δημιουργείται με το περιβάλλον).

Η γλώσσα προγραμματισμού που χρησιμοποιείται είναι η Java, ενώ χρησιμοποιείται μια Διεπαφή Προγραμματισμού Παιχνιδιών – API (Application Programming Interface) που υποστηρίζει όλες τις βασικές λειτουργίες που συναντάμε στον προγραμματισμό παιχνιδιών & προσομοιώσεων.



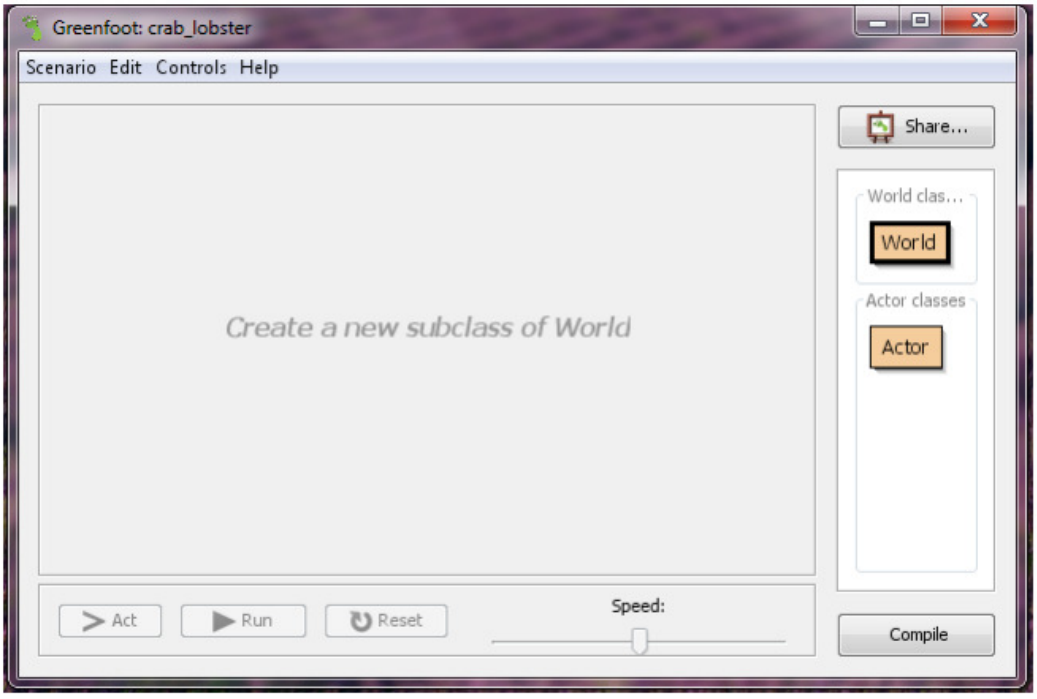
Ο χρήστης έχει τη δυνατότητα:

- να δημοσιεύσει ένα project στη σχετική gallery που υπάρχει στο site του περιβάλλοντος
- να εξάγει ένα project ως applet σε μια ιστοσελίδα
- να εξάγει ένα project ως αυτόνομη εφαρμογή (jar file)
- να εξάγει ένα project ως ένα αρχείο gfar (Greenfoot project)

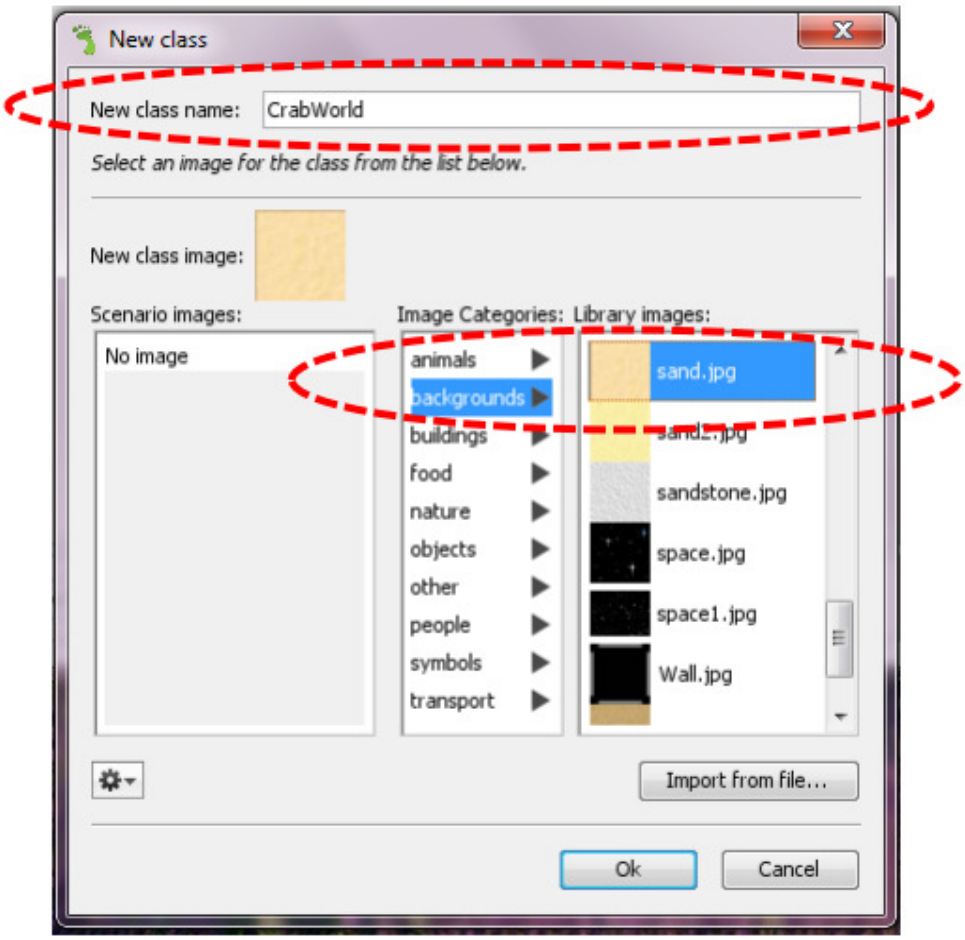


Ας δημιουργήσουμε το πρώτο μας παιχνίδι…

- Δημιουργία νέου project



- Δημιουργία υποκλάσης CrabWorld για τη δημιουργία του κόσμου/σκηνικού

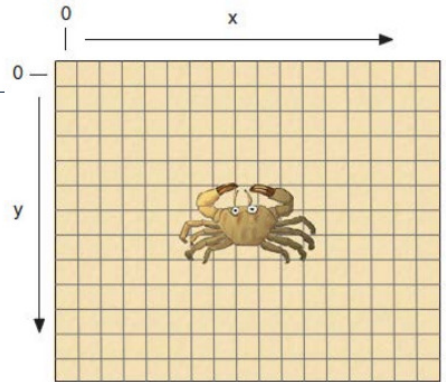


- Καθορισμός διάστασης κόσμου

```
import greenfoot.*;  // World, Actor, GreenfootImage, Greenfoot, MouseInfo

public class CrabWorld extends World
{

    public CrabWorld()
    {
        //Διάσταση του κόσμου 600x600 κελιά με μέγεθος κελιού 1x1 pixels.
        super(600, 400, 1);
    }
}
```



- Οι χαρακτήρες του παιχνιδιού

Καβούρι

- Κινείται
- Στρίβει κατά x μοίρες
- Στρίβει ελεγχόμενο από τον παίκτη
- Εντοπίζει τα όρια του κόσμου
- Ελέγχει αν υπάρχει σκουλήκι στην τρέχουσα θέση
- Τρώει το σκουλήκι που υπάρχει στην ίδια θέση
- Ψάχνει για σκουλήκι και το τρώει – αυξάνει το σκορ (αριθμό σκυληκιών)

Αστακός

- Κινείται
- Στρίβει κατά x μοίρες
- Στρίβει τυχαία
- Στρίβει όταν φτάσει στα όρια
- Εντοπίζει τα όρια του κόσμου
- Ελέγχει αν υπάρχει καβούρι στην τρέχουσα θέση
- Τρώει το καβούρι που υπάρχει στην ίδια θέση
- Ψάχνει για καβούρι και το τρώει – τερματίζει το παιχνίδι

Σκουλήκι

- Παραμένει στη θέση του

Είναι προφανές ότι τα καβούρια και οι αστακοί έχουν πολλά κοινά σημεία (λειτουργίες/μεθόδους -> συμπεριφορά). Οι κλάσεις που μοντελοποιούν τα καβούρια (έστω Crab) και τους αστακούς (έστω Lobster) θα οριστούν ως υποκλάσεις μιας άλλης κλάσης (έστω Animal) που με τη σειρά της θα υλοποιηθεί ως υποκλάση της Actor. Έλεγχος εντοπισμού των ορίων του

κόσμου. Έλεγχος ύπαρξης αντικειμένου ενός συγκεκριμένου τύπου στην τρέχουσα θέση. Ένα ζώο τρώει κάποιο άλλο εφόσον βρίσκεται στην ίδια θέση.

Ορισμός υπερκλάσης Animal για την αναπαράσταση των ζώων του παιχνιδιού

Ιδιότητες/πεδία

- Ταχύτητα κίνησης – αριθμός pixels σε κάθε βήμα (σταθερά κλάσης): WALKING_SPEED

Λειτουργίες/μέθοδοι

- Στροφή προς τη φορά των δεικτών του ρολογιού κατά x μοίρες.
- Μετακίνηση προς την τρέχουσα κατεύθυνση κατά WALKING_SPEED pixels.
- Έλεγχος εντοπισμού των ορίων του κόσμου.
- Έλεγχος ύπαρξης αντικειμένου ενός συγκεκριμένου τύπου στην τρέχουσα θέση.
- Ένα ζώο τρώει κάποιο άλλο εφόσον βρίσκεται στην ίδια θέση

Στροφή κατά 'angle' μοίρες

```
/**
 * Στροφή προς τη φορά των δεικτών του ρολογιού κατά 'angle' μοίρες
 */
public void turn(int angle)
{
    setRotation(getRotation() + angle);
}
```

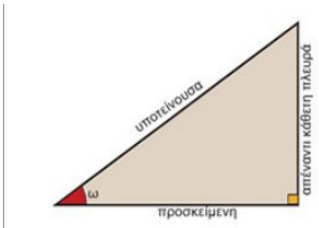
Class Actor	
void **setRotation**(int rotation)	Set the rotation of the object.
int **getRotation**()	Return the current rotation of the object.

Μετακίνηση προς την τρέχουσα κατεύθυνση

```
/**
 * Κίνηση προς τα εμπρός.
 */
public void move()
{
    double angle = Math.toRadians( getRotation() );
    int x = (int) Math.round(getX() + Math.cos(angle) * WALKING_SPEED);
    int y = (int) Math.round(getY() + Math.sin(angle) * WALKING_SPEED);

    setLocation(x, y);
}
```

Class Actor	
`int` **`getX`**`()`	Return the x-coordinate of the object's current location.
`int` **`getY`**`()`	Return the y-coordinate of the object's current location.
`void` **`setLocation`**`(int x, int y)`	Assign a new location for this object.

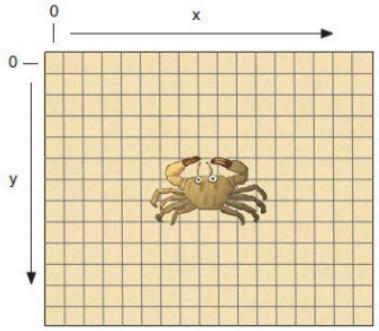


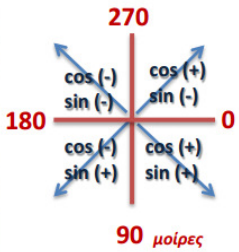
sin

$$\text{ημω} = \frac{\text{απέναντι κάθετη πλευρά}}{\text{υποτείνουσα}}$$

cos

$$\text{συνω} = \frac{\text{προσκείμενη κάθετη πλευρά}}{\text{υποτείνουσα}}$$






```
    double angle = Math.toRadians( getRotation() );
    int x = (int) Math.round(getX() + Math.cos(angle) * WALKING_SPEED);
    int y = (int) Math.round(getY() + Math.sin(angle) * WALKING_SPEED);
```

Έλεγχος εντοπισμού των ορίων του κόσμου

```
/**
 * Επιστρέφει true αν βρίσκεται στα όρια του κόσμου.
 */
public boolean atWorldEdge()
{
    if(getX() < 20 || getX() > getWorld().getWidth() - 20)
        return true;
    if(getY() < 20 || getY() > getWorld().getHeight() - 20)
        return true;
    return false;
}
```

Class Actor	
World **getWorld**()	Return the world that this object lives in.
int **getX**()	Return the x-coordinate of the object's current location.
int **getY**()	Return the y-coordinate of the object's current location.

Class World	
int **getHeight**()	Return the height of the world (in number of cells).
int **getWidth**()	Return the width of the world (in number of cells).

Έλεγχος ύπαρξης άλλου τύπου αντικειμένου στην τρέχουσα θέση

```
/**
 * Επίστρεψε true αν υπάρχει αντικείμενο τύπου 'clss' στην τρέχουσα θέση,
 * false διαφορετικά.
 */
public boolean canSee(Class clss)
{
    Actor actor = getOneObjectAtOffset(0, 0, clss);
    return actor != null;
}
```

Class Actor	
protected Actor **getOneObjectAtOffset** (int dx, int dy, Class cls)	Return one object that is located at the specified cell (relative to this objects location).

Ένα ζώο τρώει κάποιο άλλο που βρίσκεται στην ίδια θέση

```
/**
 * Προσπάθησε να φας ένα αντικείμενο τύπου'clss'. Αν δεν υπάρχει αντικείμενο
 * αυτού του τύπου στην τρέχουσα θέση η μέθοδος δεν έχει κάποιο αποτέλεσμα
 */
public void eat(Class clss)
{
   Actor actor = getOneObjectAtOffset(0, 0, clss);
   if(actor != null) {
        getWorld().removeObject(actor);
   }
}
```

Class Actor	
protected Actor **getOneObjectAtOffset** (int dx, int dy, Class cls)	Return one object that is located at the specified cell (relative to this objects location).
World **getWorld**()	Return the world that this object lives in.

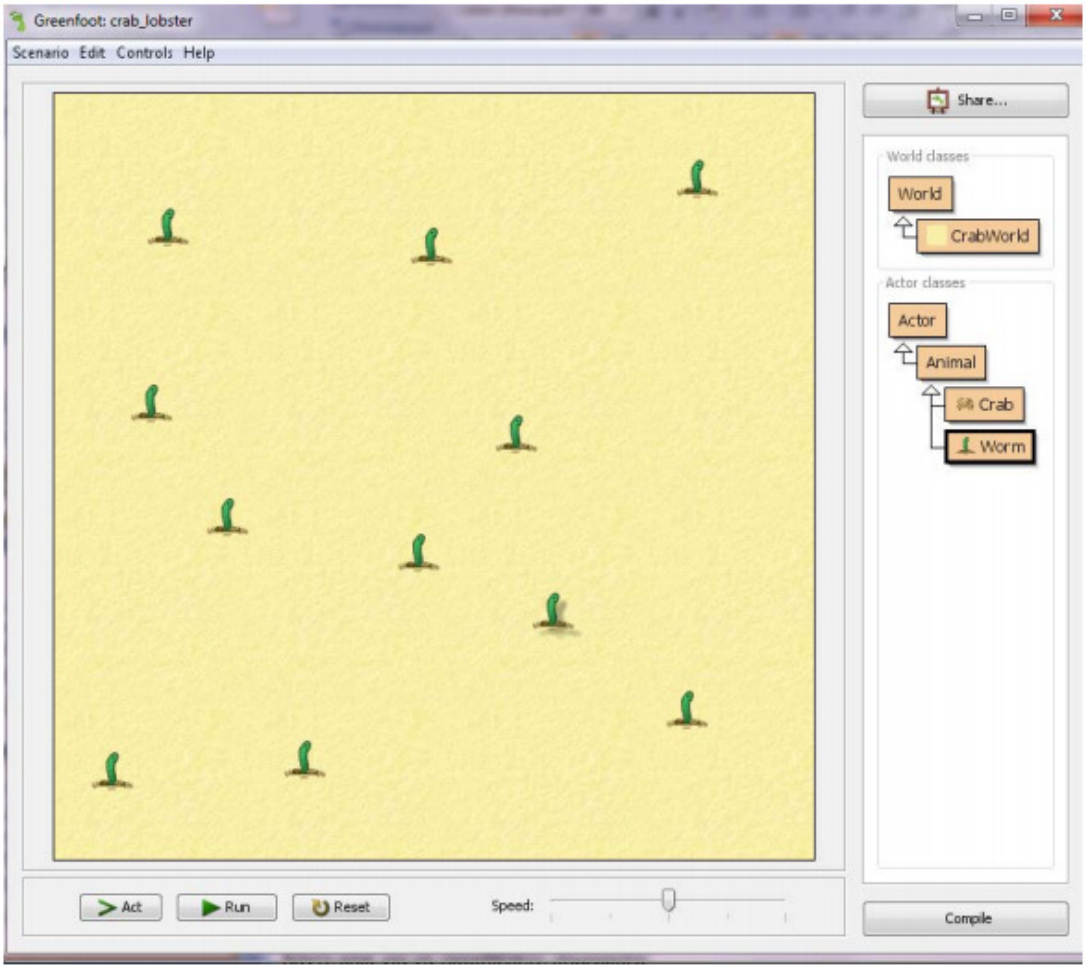
Class World	
void **removeObject**(Actor object)	Remove an object from the world.

Ορισμός κλάσης Worm για την αναπαράσταση των σκουληκιών

Προς το παρόν δεν έχουν κάποια συμπεριφορά. Κάθονται και περιμένουν να τους φάνε τα καβούρια. Φυσικά, υπάρχουν οι μέθοδοι που κληρονομούνται από τις κλάσεις Animal και Actor. Θα μπορούσαν τα σκουλήκια να κινούνται και αυτά και μάλιστα με μεγάλη ταχύτητα για να ξεφεύγουν από τα καβούρια.

Προσθήκη σκουληκιών στο παιχνίδι

Με δεξί κλικ πάνω στην κλάση Worm επιλέγετε new Worm() από το αναδυόμενο μενού και «αφήνετε» το σκουλήκι που δημιουργείται μέσα στον κόσμο. Για να προσθέσετε πολλά στιγμιότυπα, έχοντας επιλέξει την κλάση, κρατάτε πατημένο το Shift και κάνετε κλικ σε εκείνα τα σημεία που θέλετε να προστεθούν σκουλήκια. Με δεξί κλικ στην περιοχή του παιχνιδιού, επιλέγετε Save World για να αποθηκευτεί το σκηνικό του παιχνιδιού.



Ορισμός κλάσης Crab για την αναπαράσταση των καβουριών

Ιδιότητες/πεδία

- Αριθμός σκουληκιών που έχει φάει – wormsEaten

Λειτουργίες/μέθοδοι.

- Στρίβει προς τα αριστερά και δεξιά ελεγχόμενο από τον παίκτη.
- Ψάχνει για σκουλήκι και το τρώει – αν φάει 8 σκουλήκια το παιχνίδι τερματίζει
- … και φυσικά συνεχώς κάθε στιγμή ενεργεί (act) ως εξής:
- Στρίβει αν το ζητήσει ο παίκτης
- Κινείται
- Ψάχνει για σκουλήκι και αν βρει το τρώει

Φυσικά, υπάρχουν και οι μέθοδοι που κληρονομούνται από τις κλάσεις Animal και Actor.

Στρίβει αριστερά και δεξιά ελεγχόμενο από τον χρήστη

```
/**
 * Ελεγξε αν έχει πατηθεί το αριστερό ή δεξί βελάκι από το πληκτρολόγιο
 * και στρίψε κατάλληλα.
 */
public void checkKeypress()
{
    if (Greenfoot.isKeyDown("left"))
    {
          turn(-4);
    }
    if (Greenfoot.isKeyDown("right"))
    {
         turn(4);
    }
}
```

Class Greenfoot	
`static boolean` **`isKeyDown`**`(String keyName)`	Check whether a given key is currently pressed down.

Ψάχνει για σκουλήκι και το τρώει

```
/**
 * Ελεγξε αν έχεις πέσει πάνω σε ένα σκουλήκι προκειμένου να το φας.
 * Αν έχεις φάει 8 σκουλήκια κερδίζεις.
 */
public void lookForWorm()
{
    if ( canSee(Worm.class) )
    {
          eat(Worm.class);
          Greenfoot.playSound("slurp.wav");

          wormsEaten = wormsEaten + 1;
          if (wormsEaten == 8)
          {
                  Greenfoot.playSound("fanfare.wav");
                  Greenfoot.stop();
          }
     }
}
```

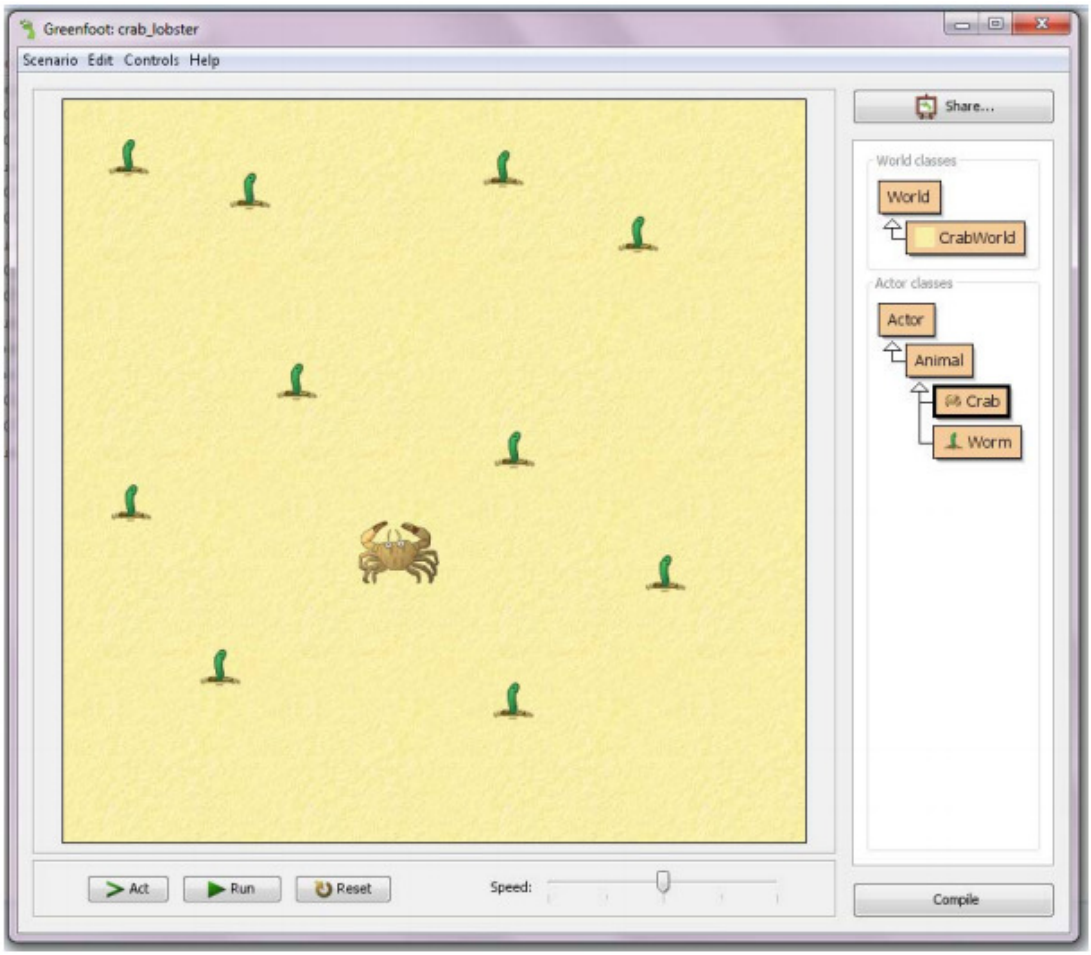
Class Greenfoot	
`static void` **`playSound`**`(String soundFile)`	Play sound from a file.
`static void` **`stop`**`()`	Stop the simulation.

Βασική συμπεριφορά καβουριού

```
/**
 * Υλοποίηση βασικής συμπεριφοράς καβουριού. Καλείται κάθε φορά που πατάμε
 * το κουμπί 'Act' ή 'Run' στο περιβάλλον.
 */
public void act()
{
    checkKeypress();
    move();
    lookForWorm();
}
```

Προσθήκη καβουριού στο παιχνίδι

Μπορείτε πλέον να προσθέσετε ένα καβούρι και να δοκιμάσετε την πρώτη έκδοση του παιχνιδιού. Δεν θα ήταν ωραία ο βασικός μας χαρακτήρας να κινείται πιο “φυσικά”; Αυτό μπορεί να γίνει εύκολα χρησιμοποιώντας και μια 2η εικόνα για το καβούρι με τα πόδια του σε διαφορετική θέση…




```
public class Crab extends Animal
{
    private GreenfootImage image1;
    private GreenfootImage image2;
    ...
    public Crab()
    {
        image1 = new GreenfootImage("crab.png");
        image2 = new GreenfootImage("crab2.png");
        setImage(image1);
        wormsEaten = 0;
    }

    public void act()
    {
        ...
        switchImage();
    }
```

a) crab with legs out

b) crab with legs in

```
public void switchImage()
{
    if (getImage() == image1)
    {
        setImage(image2);
    }
    else
    {
        setImage(image1);
    }
}
```

Βελτιώνοντας την κίνηση του καβουριού

Προσθήκη αντίπαλου ….

… και επειδή παιχνίδι χωρίς αντίπαλο δεν έχει ενδιαφέρον ας προσθέσουμε και μερικούς αστακούς που τους αρέσουν τα καβούρια (όσο περισσότερα τόσο μεγαλύτερη η πρόκληση!)

Ορισμός κλάσης Lobster για την αναπαράσταση των αστακών

Λειτουργίες/μέθοδοι

- Στρίβει τυχαία
- Στρίβει όταν φτάσει στα όρια του κόσμου
- Ψάχνει για καβούρι και το τρώει – το παιχνίδι τερματίζει
- … και φυσικά συνεχώς κάθε στιγμή ενεργεί (act) ως εξής:
- Στρίβει όταν φτάσει στα όρια του κόσμου
- Στρίβει τυχαία
- Κινείται
- Ψάχνει για καβούρι και αν βρει το τρώει.

Φυσικά, υπάρχουν και οι μέθοδοι που κληρονομούνται από τις κλάσεις Animal και Actor

Στρίβει αν είναι στα όρια του κόσμου

```
/**
 * Έλεγξε αν είσαι στα όρια του κόσμου και στρίψε.
 */
public void turnAtEdge()
{
   if ( atWorldEdge() )
   {
         turn(17);
   }
}
```

Στρίβει (ή όχι) τυχαία αριστερά ή δεξιά

```
/**
 * Αποφασίζεται τυχαία να στρίψει ή όχι (πιθανότητα να στρίψει 9 /100)
 * Αν στρίψει, στρίβει τυχαία λίγο προς τα αριστερά ή δεξιά (-45..45 μοίρες)
 */
public void randomTurn()
{
    if (Greenfoot.getRandomNumber(100) > 90) {
        turn(Greenfoot.getRandomNumber(90)-45);
    }
}
```

Class Greenfoot	
`static int` **`getRandomNumber`**`(int limit)`	Return a random number between 0 (inclusive) and limit (exclusive).

Βασική συμπεριφορά αστακού

```
/**
 * Έλεγξε αν έπεσες πάνω σε ένα καβούρι. Αν ναι απομάκρυνε το και τελείωσε
 * το παιχνίδι
 */
public void lookForCrab()
{
    if ( canSee(Crab.class) )
    {
        eat(Crab.class);
        Greenfoot.playSound("au.wav");
        Greenfoot.stop();
    }
}
```

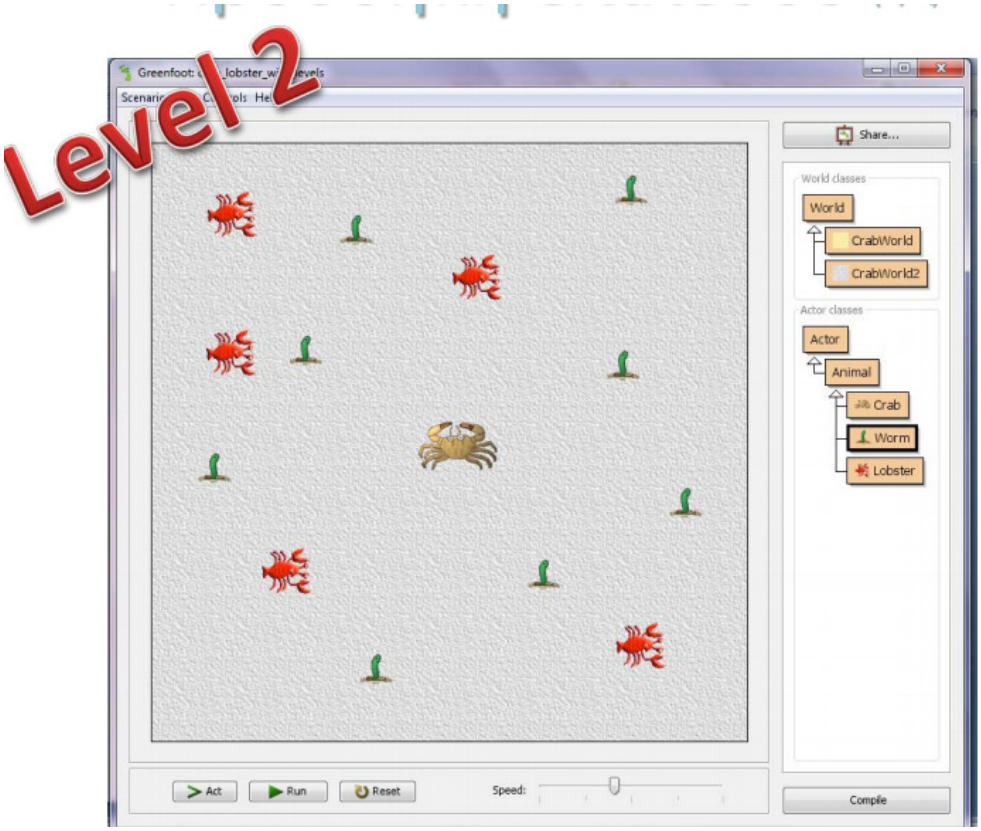
```
/**
 * Υλοποίηση βασικής συμπεριφοράς αστακού. Καλείται κάθε φορά που πατάμε
 * το κουμπί 'Act' ή 'Run' στο περιβάλλον.
 */
public void act()
{
    turnAtEdge();
    randomTurn();
    move();
    lookForCrab();
}
```

Προσθήκη επιπέδου

Δημιουργήστε το 2ο επίπεδο με τον ίδιο τρόπο που δημιουργήσατε και τον αρχικό κόσμο:

- Δημιουργήστε μια 2η υποκλάση της World, έστω CrabWorld2

- Το 2ο επίπεδο πρέπει να έχει μεγαλύτερο βαθμό δυσκολίας – για παράδειγμα, περισσότερους εχθρούς ή/και μικρότερη περιοχή!
- Αφού δημιουργήσετε την νέα κλάση κάντε δεξί κλικ πάνω σε αυτή και επιλέξτε new CrabWorld2 για να δημιουργήσετε ένα στιγμιότυπο της κλάσης αυτής και να δημιουργήσετε το 2ο επίπεδο
- Μην ξεχάσετε να αποθηκεύσετε την κατάσταση του κόσμου
- Αν θέλετε να επεξεργαστείτε ξανά το 1ο επίπεδο, για να το φέρετε στο προσκήνιο χρειάζεται δημιουργήσετε στιγμιότυπο της αντίστοιχης κλάσης



Έχοντας πλέον επίπεδα στο παιχνίδι, ο πρωταγωνιστής του θα πρέπει να:

- μεταφέρεται από το ένα επίπεδο στο άλλο

ο πρωταγωνιστής (καβούρι) θα πρέπει να περνάει ως όρισμα στον κατασκευαστή του κάθε επιπέδου

- γνωρίζει ανά πάσα στιγμή σε ποιο επίπεδο βρίσκεται

προσθήκη μιας σχετικής ιδιότητας/πεδίου στην κλάση του πρωταγωνιστή

- ελέγχει σε κάθε βήμα αν πρέπει να προχωρήσει στο επόμενο επίπεδο και αν ναι να αλλάζει επίπεδο

προσθήκη σχετικής μεθόδου

Μεταφορά πρωταγωνιστή μεταξύ επιπέδων

```
public class CrabWorld extends World
{
    public CrabWorld()
    {
        this(new Crab());
    }

    public CrabWorld(Crab crab)
    {
        super(600, 600, 1);

        prepare(crab);
    }

    private void prepare(Crab crab)
    {
        //Crab crab = new Crab();  ο αστακός περνάει πλέον μέσω παραμέτρου

        addObject(crab, 269, 366);
        …
    }
}
```

```
public class CrabWorld2 extends World
{
    public CrabWorld2()
    {
        this(new Crab());
    }

    public CrabWorld2(Crab crab)
    {
        super(600, 600, 1);

        prepare(crab);
    }

    private void prepare(Crab crab)
    {
        //Crab crab = new Crab();  ο αστακός περνάει πλέον μέσω παραμέτρου

        addObject(crab, 308, 303);
        …
    }
```

Παρακολούθηση/αλλαγή επιπέδων

```
public class Crab extends Animal
{
    ...
    private int level;
    private boolean startGame;

    public Crab()
    {
        ...
        level = 1;
        startGame = true;
    }

    public void act()
    {
        ...
        checkNextLevel();
    }
```

3.6 Βασικές Αρχές Προγραμματισμού

Παραπάνω αναλύσαμε κάποιες εφαρμογές που μπορεί να χρησιμοποιήσει κάποιος είτε για να μάθει είτε για να διδάξει τις βασικές αρχές προγραμματισμού. Όπως και σε κάθε άλλο τομέα, έτσι και στην εκπαίδευση στον προγραμματισμό, θα πρέπει κάποιος να λάβει υπόψην του τις διαφορετικές εκπαιδευτικές ανάγκες που μπορεί να έχει ένα παιδί ανάλογα με την ηλικία του.

Προτού όμως αναλύσουμε τις ανάγκες, καλό θα ήταν να αναφέρουμε τις βασικές αρχές προγραμματισμού, έστω και περιληπτικά.

Κατ' αρχήν πρέπει να αναφέρουμε ότι ο προγραμματισμός διακρίνεται σε τέσσερις μεγάλες κατηγορίες: στον συναρτησιακό προγραμματισμό, τον αντικειμενοστραφή, τον προστακτικό και τον ανακλαστικό προγραμματισμό.

Συναρτησιακός Προγραμματισμός

Στην πληροφορική ο συναρτησιακός προγραμματισμός είναι ένα προγραμματιστικό παράδειγμα που αντιμετωπίζει τον υπολογισμό ως την αποτίμηση μαθηματικών συναρτήσεων και αποφεύγει την κατάσταση προγράμματος και τα μεταβλητά δεδομένα. Δίνει έμφαση στην εφαρμογή συναρτήσεων, σε αντίθεση με τον προστατικό προγραμματισμό, ο οποίος δίνει έμφαση στις αλλαγές κατάστασης. Ο συναρτησιακός προγραμματισμός έχει τις ρίζες του στο λογισμικό λάμδα, ένα τυπικό σύστημα που αναπτύχθηκε το 1930 για τη διερεύνηση του ορισμού συναρτήσεων, της εφαρμογής συναρτήσεων και της αναδρομής. Πολλές συναρτησιακές γλώσσες προγραμματισμού μπορούν να θεωρηθούν επεκτάσεις του λογισμικού λάμδα.

Πρακτικά, η διαφορά μεταξύ μιας μαθηματικής συνάρτησης και της έννοιας της "συνάρτησης" που χρησιμοποιείται στον προστακτικό προγραμματισμό είναι ότι οι προστακτικές συναρτήσεις μπορούν να έχουν παρενέργειες, αλλάζοντας την τιμή των ήδη αποτιμημένων υπολογισμών. Λόγω αυτού, δεν έχουν διαφάνεια αναφορικότητας, δηλαδή η ίδια έκφραση της γλώσσας μπορεί να δώσει διαφορετικές τιμές ανάλογα με την κατάσταση του εκτελούμενου προγράμματος. Αντίστροφα, σε συναρτησιακά προγράμματα, η τιμή που επιστρέφει μια συνάρτηση εξαρτάται μόνο από τα ορίσματα που αποτελούν την είσοδο της συνάρτησης. Έτσι, η κλήση μιας συνάρτησης *f* με ένα όρισμα *x* θα δώσει το ίδιο αποτέλεσμα *f(x)* και τις δύο φορές. Η εξάλειψη των παρενεργειών μπορεί να κάνει πολύ ευκολότερο το να κατανοηθεί και να προβλεφθεί η συμπεριφορά ενός προγράμματος. Αυτό είναι ένα από τα κίνητρα για την ανάπτυξη του συναρτησιακού προγραμματισμού.

Οι συναρτησιακές γλώσσες προγραμματισμού, και ειδικότερα οι αμιγώς συναρτησιακές συναντώνται περισσότερο στο ακαδημαϊκό παρά στο εμπορικό ή βιομηχανικό περιβάλλον. Παρ' όλα αυτά, αξιοσημείωτες συναρτησιακές γλώσσες που χρησιμοποιούνται στη βιομηχανία και στην ανάπτυξη εμπορικών εφαρμογών είναι ανάμεσα σε άλλες η Erlang, η Ocaml, αλλά και γλώσσες ειδικών πεδιών όπως η R για τη στατιστική, η Mathematica για τα

συμβολικά μαθηματικά, ή η K για την χρηματιστηριακή ανάλυση. Διαδεδομένες γλώσσες ειδικού πεδίου όπως η SQL και τα Lex/Yacc χρησιμοποιούν στοιχεία συναρτησιακού προγραμματισμού, ειδικά για να αποφύγουν μεταβλητές τιμές. Τα λογιστικά φύλλα μπορούν επίσης να θεωρηθούν συναρτησιακές γλώσσες προγραμματισμού.

Προγραμματισμός σε συναρτησιακό στυλ μπορεί να επιτευχθεί και σε μη συναρτησιακές γλώσσες όπως η C, η Java και η Python, που δεν είναι ειδικά σχεδιασμένες για τέτοια χρήση.

```
factorial:=1;
while (n > 0) do begin
factorial := factorial * n;
n:=n-1
end
```

Εικόνα 3.5.1: Παράδειγμα Συναρτησιακού Προγραμματισμού (Pascal)

Αντικειμενοστρεφής Προγραμματισμός

Στην πληροφορική **αντικειμενοστρεφής προγραμματισμός** (object-oriented programming), ή ΑΠ, ονομάζουμε ένα προγραμματιστικό υπόδειγμα το οποίο εμφανίστηκε στα τέλη της δεκαετίας του 1960 και καθιερώθηκε κατά τη δεκαετία του 1990, αντικαθιστώντας σε μεγάλο βαθμό το παραδοσιακό υπόδειγμα του δομημένου προγραμματισμού. Πρόκειται για μία μεθοδολογία ανάπτυξης προγραμμάτων, υποστηριζόμενη από κατάλληλες γλώσσες προγραμματισμού, όπου ο χειρισμός σχετιζόμενων δεδομένων και των διαδικασιών που επενεργούν σε αυτά γίνεται από κοινού, μέσω μίας δομής δεδομένων που τα περιβάλλει ως αυτόνομη οντότητα με ταυτότητα και δικά της χαρακτηριστικά. Αυτή η δομή δεδομένων καλείται *αντικείμενο* και αποτελεί

πραγματικό στιγμιότυπο στη μνήμη ενός σύνθετου, και πιθανώς οριζόμενου από τον χρήστη, τύπου_δεδομένων ονόματι κλάση. Η κλάση προδιαγράφει τόσο δεδομένα όσο και τις διαδικασίες οι οποίες επιδρούν επάνω τους· αυτή υπήρξε η πρωταρχική καινοτομία του ΑΠ.

Έτσι μπορεί να οριστεί μία προδιαγραφή δομής αποθήκευσης (π.χ. μία κλάση «τηλεόραση») η οποία να περιέχει τόσο ιδιότητες (π.χ. Μία μεταβλητή «τρέχον κανάλι») όσο και πράξεις ή χειρισμούς επί αυτών των ιδιοτήτων (π.χ. μία διαδικασία «άνοιγμα της τηλεόρασης»). Στο εν λόγω παράδειγμα κάθε υλική τηλεόραση (κάθε αντικείμενο αποθηκευμένο πραγματικά στη μνήμη) αναπαρίσταται ως ξεχωριστό, «φυσικό» στιγμιότυπο αυτής της πρότυπης, ιδεατής κλάσης. Επομένως μόνο τα αντικείμενα καταλαμβάνουν χώρο στη μνήμη του υπολογιστή ενώ οι κλάσεις αποτελούν απλώς «καλούπια». Οι αιτίες που ώθησαν στην ανάπτυξη του ΑΠ ήταν οι ίδιες με αυτές που οδήγησαν στην ανάπτυξη του δομημένου προγραμματισμού (ευκολία συντήρησης, οργάνωσης, χειρισμού και επαναχρησιμοποίησης κώδικα μεγάλων και πολύπλοκων εφαρμογών), όμως τελικώς η αντικειμενοστρέφεια επικράτησε καθώς μπορούσε να αντεπεξέλθει σε προγράμματα πολύ μεγαλύτερου όγκου και πολυπλοκότητας.

```
class Cat():
    name
    age
    sex

    def eat():
        print("%s is eating" % name)

    def sleep():
        print("%s is sleeping" % name)
```

Εικόνα 3.5.2: Παράδειγμα Αντικειμενοστραφούς Προγραμματισμού (Python)

Προστακτικός Προγραμματισμός

Στην πληροφορική καλούμε προστακτικό προγραμματισμό, σε αντίθεση με το δηλωτικό προγραμματισμό, ένα προγραμματιστικό υπόδειγμα όπου το ζητούμενο κατασκευάζεται / υπολογίζεται αλλάζοντας την κατάσταση του υπολογιστή μέσω εντολών.

Το υπόδειγμα αυτό ακολουθούν οι διαδικαστικές γλώσσες προγραμματισμού, όπως η Pascal, η C, η Fortran, κ.α., αλλά και πολλές αντικειμενοστρεφείς γλώσσες όπως η Java, η C++, η C#, κ.α.

```
male(james1).
 male(charles1).
 male(charles2).
 male(james2).
 male(george1).

 female(catherine).
 female(elizabeth).
 female(sophia).

 parent(charles1, james1).
 parent(elizabeth, james1).
 parent(charles2, charles1).
 parent(catherine, charles1).
 parent(james2, charles1).
 parent(sophia, elizabeth).
 parent(george1, sophia).
```

Εικόνα 3.5.3: Παράδειγμα Προστακτικού Προγραμματισμού (Prolog)

Ανακλαστικός Προγραμματισμός

Στην πληροφορική, ο όρος ανάκλαση (αγγλ. reflection) αναφέρεται στη δυνατότητα ενός προγράμματος υπολογιστή να παρατηρεί και να μεταβάλλει τη δομή και τη συμπεριφορά του κατά την εκτέλεση.

Σε στατικές γλώσσες όπως η C/C++ όταν γράφουμε το πρόγραμμα πρώτα το μεταγλωττίζουμε σε γλώσσα μηχανής και στην συνέχεια το τρέχουμε. Καθώς το πρόγραμμα εκτελείται δεν είναι δυνατό να τροποποιηθεί το πρόγραμμα και κάθε αλλαγή θα πρέπει να γίνει στο πηγαίο κώδικα ο οποίος στην συνέχεια θα πρέπει να μεταγλωττιστεί σε γλώσσα μηχανής και να εκτελεστεί. Η ιδέα της ανάκλασης είναι ότι κάνουμε αλλαγές στο κώδικα του προγράμματος την ώρα που αυτό εκτελείται. Για παράδειγμα αν έχουμε μια κλάση μπορούμε την ώρα που το πρόγραμμα τρέχει, μέσω της ανάκλασης, να πάρουμε πληροφορίες για την κλάση και να τροποποιήσουμε την λειτουργία της (π.χ. να τροποποιήσουμε τον κώδικα των συναρτήσεων-μεθόδων). Η ιδέα είναι ότι οι εντολές του προγράμματος αποθηκεύονται σαν δεδομένα τα οποία μπορούν να τροποποιηθούν κατά την διάρκεια εκτέλεσης. Η δυνατότητα να προσθέτουμε νέο εκτελέσιμο κώδικα την ώρα που τρέχει το πρόγραμμα ονομάζεται μεταπρογραμματισμός (metaprogramming).

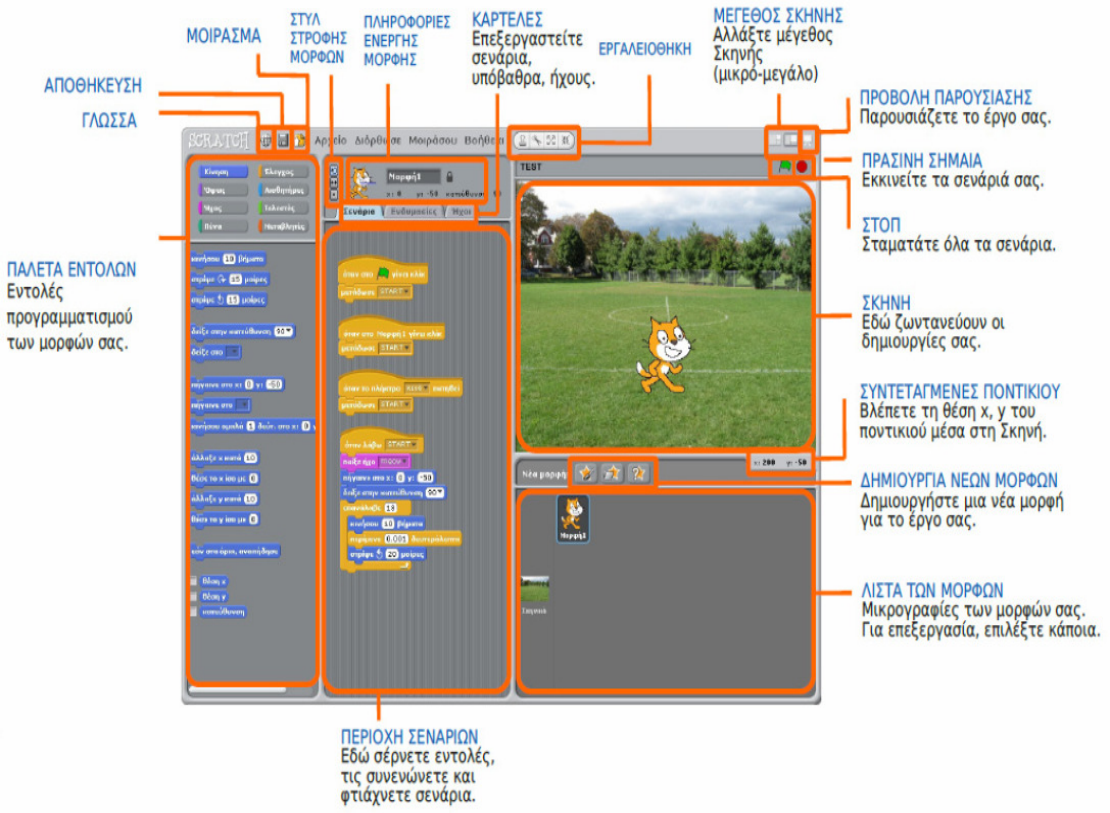
Η ανάκλαση εφαρμόζεται συνήθως σε γλώσσες υψηλού επιπέδου που τρέχουν σε κάποια εικονική μηχανή όπως η Smalltalk και άλλες γλώσσες σεναρίων όπως η Ruby, και λιγότερο σε στατικές γλώσσες όπως η Java, η C, η ML ή η Haskell.

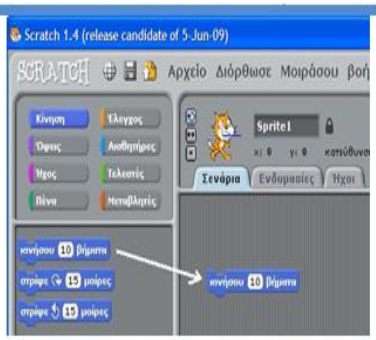
```
name = 'Patricia Rosanna Jessica Mildred Oppenheimer'
puts 'My name is ' + name + '.'
```

```
puts 'Wow! ' + name + ' is a really long name!'
```

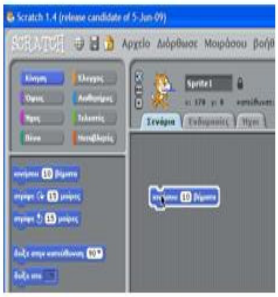
4 Ασκήσεις

4.1 Scratch

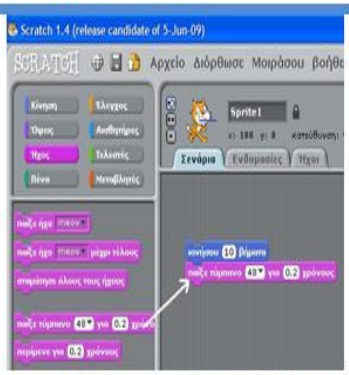




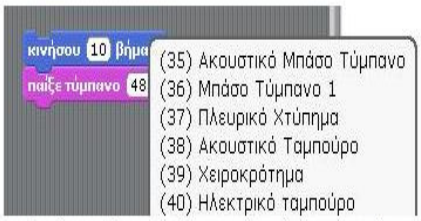
Σύρε μια εντολή "κινήσου" και μετέφερε τη στη περιοχή σεναρίων.



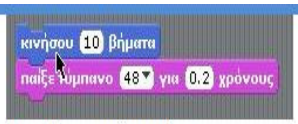
Με διπλό 'κλικ' στην εντολή κάνε τη γάτα να κινηθεί.



Σύρε την εντολή "παίξε τύμπανο" (εντολή ήχου).



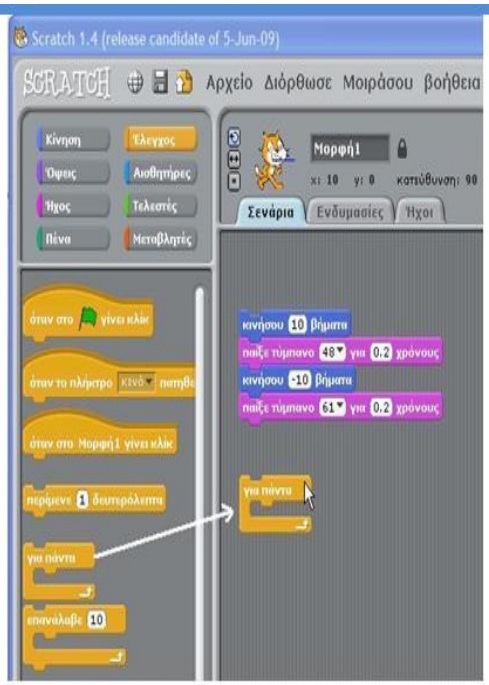
Διάλεξε έναν ήχο από τις πιθανές επιλογές που εμφανίζονται πατώντας το βελάκι δίπλα στον αριθμό.



Με διπλό κλικ, δες τι έφτιαξες.

Πρόσθεσε ακόμη μια εντολή "κινήσου" και πληκτρολόγησε αρνητικό ποσό (-10)
Κάνε διπλό κλικ οπουδήποτε στο σωρό των εντολών.

Πρόσθεσε ακόμα μια εντολή "παίξε τύμπανο" και διάλεξε ήχο από το μενού. Κάνε πάλι διπλό κλικ.



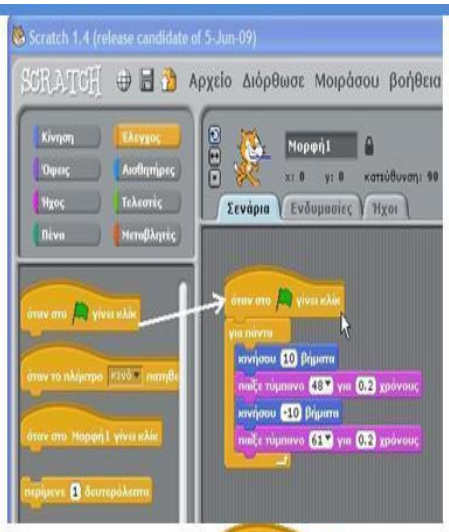
Σύρε την εντολή "για πάντα"

Βάλε το σωρό των εντολών στο στόμα του "για πάντα"

Για να σύρεις το σωρό, πάρε τη πάνω εντολή.

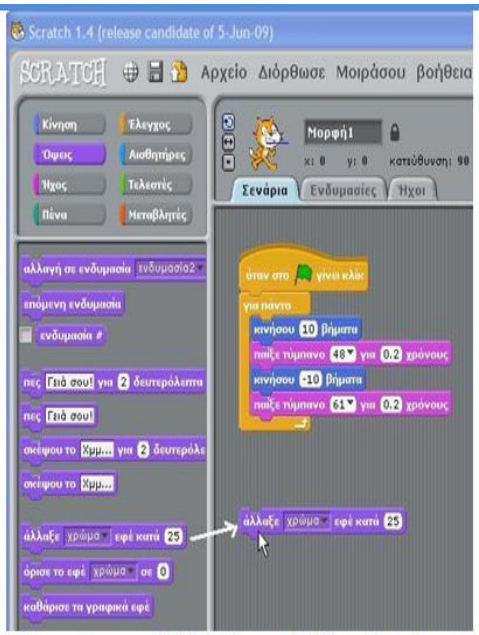
Κάνε διπλό κλικ.

Για να σταματήσει κάνε κλικ σ' αυτό το κουμπί που βρίσκεται πάνω δεξιά στο περιβάλλον.

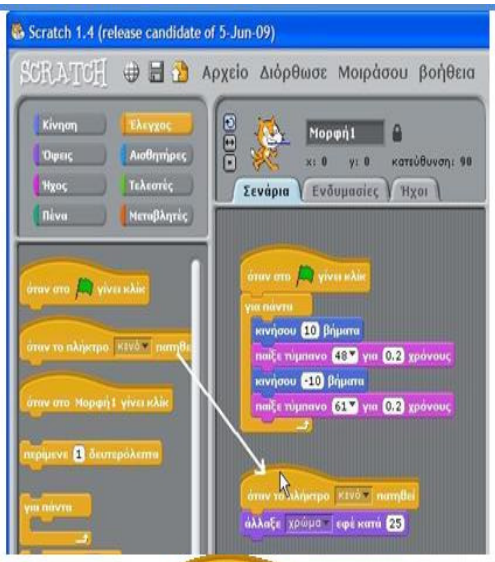


Σύρε έξω την εντολή  και τοποθέτησέ τη στη κορυφή του σωρού των εντολών.

Όποτε κάνεις κλικ στην πράσινη σημαία οι εντολές σου θα ξεκινήσουν για να σταματήσουν κάνε κλικ στο κουμπί  .



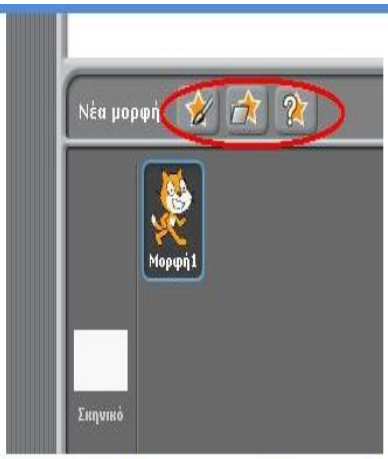
Σύρε την εντολή "άλλαξε εφε". Κάνε διπλό κλικ για να δεις τι κάνει.



Σύρε την εντολή  και τοποθέτησέ τη πάνω από την εντολή "άλλαξε εφέ".

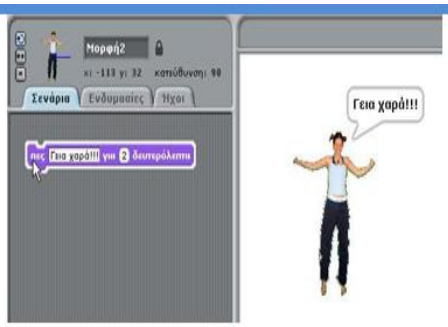
Πίεσε το πλήκτρο κενό από το πληκτρολόγιο.

Μπορείς να διαλέξεις διαφορετικό πλήκτρο από το μενού.

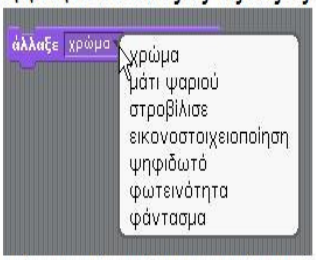


Για να προσθέσεις αντικείμενο κάνε κλικ σε αυτά τα κουμπιά.

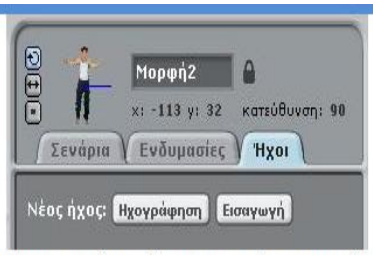
Για να προσθέσεις αυτό το αντικείμενο κάνε κλικ στο  . Μετά κάνε κλικ στο φάκελο με τις μορφές των ανθρώπων (People) και διάλεξε το "jodi 1".



Κάνε κλικ μέσα στην εντολή "πες" και πληκτρολόγησε για να αλλάξεις τις λέξεις.

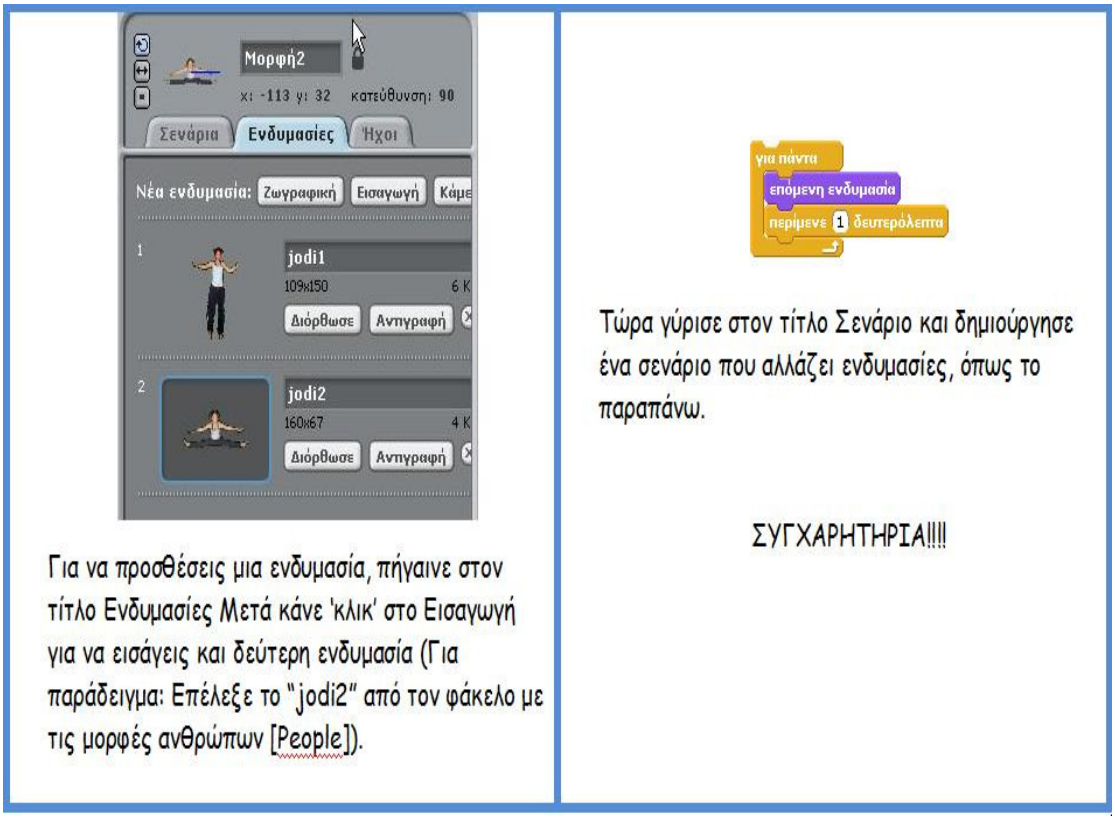


Στην εντολή αλλαγής χρώματος διάλεξε ένα διαφορετικό εφέ, κάνε διπλό κλικ στην εντολή. Για να καθαρίσεις τα εφέ πάτα το κόκκινο κουμπί.



Κάνε κλικ στον τίτλο Ήχοι, και είτε ηχογράφησε ένα δικό σου ήχο (Ηχογράφηση) ή εισήγαγε έναν δικό σου (Εισαγωγή).

Γύρισε στον τίτλο σενάριο και χρησιμοποίησε την εντολή "παίξε ήχο" για να ακούσεις τον ήχο.

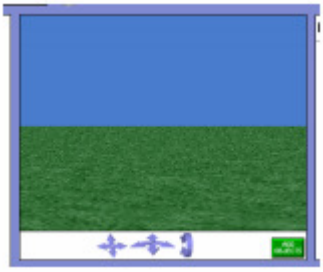


4.3 Alice

Για να δημιουργήσουμε σωστά ένα παιχνίδι Alice, πρέπει να έχουμε στο μυαλό μας ότι:

- Όλα τα αντικείμενα στο Alice πρέπει να δημιουργηθούν πρωτού το παιχνίδι ξεκινίσει. Αυτό σημαίνει ότι έχουμε ορατά και αόρατα αντικείμενα μέσω μιας ιδιότητας που ονομάζεται **opacity**. Εάν έχουμε opacity 0 έχουμε αόρατο αντικείμενο, εάν έχουμε opacity 1, έχουμε ορατό αντικείμενο. Θα φτιάξουμε ένα παιχνίδι στόχος του οποίου θα είναι ο παίκτης να κάνει 5 κλικ στο ζόμπι για να κερδίσει.

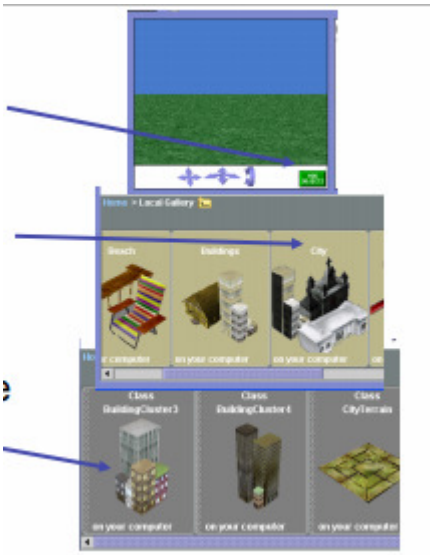
Δημιουργία Κόσμου



- Κλικ File -> New World
- Επιλογή «Template»
- Κλικ στο grass world

- κλικ open

Προσθήκη Αντικειμένου στον Κόσμο



- Κλικ στην Προσθήκη Αντικειμένου κάτω στην οθόνη.
- Κυλίστε στην κατηγορία πόλης και κλικ πάνω στην πόλη
- Προσθέστε κάποια κτίρια

Προσθήκη Ζόμπι

- Κλικ στην τοπική Γκαλερί για να πάμε πίσω στη λίστα κατηγοριών.
- Ανοίξτε την Κατηγορία «Άνθρωποι»
- Κύλιση στο ζόμπι και προσθήκη στον κόσμο.

Απόκρυψη Ζόμπι

Δημιουργήστε μια μέθοδο για να κρύψει το ζόμπι

- Κάντε κλικ στο ζόμπι στο δέντρο αντικείμενων

- Κάντε κλικ στις μεθόδους στο παράθυρο «λεπτομέρειες»

- Κάντε κλικ στο κουμπί Δημιουργία νέας μεθόδου

- Το όνομα της μεθόδου ας είναι *hide*

Κάντε κλικ στις ιδιότητες στο παράθυρο λεπτομέρεια

• Σύρετε έξω opacity = 1 (100%)

• Βάλτε στην hide μέθοδος

• Αλλάξτε το opacity 0 (0%)

• Κάντε κλικ στην επιλογή σχετίζεται με τις μεθόδους

• Τραβήξτε προς τα έξω κίνηση ζόμπι κάτω από 2 μέτρα

Δημιουτργία μεθόδου PopUp

- Κλικ στις μεθόδους

- Κλικ στο κουμπί «Δημιουργία Νέας Μεθόδου»
- Ονομάστε τη μέθοδο popup

  - μετακινεί το ζόμπι τυχαία

  - μετακινεί το ζόμπι πάνω από το έδαφος

  - κάνει το ζόμπι ορατό

Τυχαία Μετακίνηση Ζόμπι

- Ξεκινήστε από το να τραβήξετε το ζόμπι
- Κλικ στον κόσμο στο δένδρο αντικειμένων για να το επιλέξετε
- Κλικ στις λεπτομέρειες
- Random Number : min = 0 και max = 2

Εμφάνιση Ζόμπι

- Κλικ στο ζόμπι στο δέντρο αντικειμένων και επιλογή του.
- Κλικ στις μεθόδους από τις λεπτομέρειες
- Opacity = 0

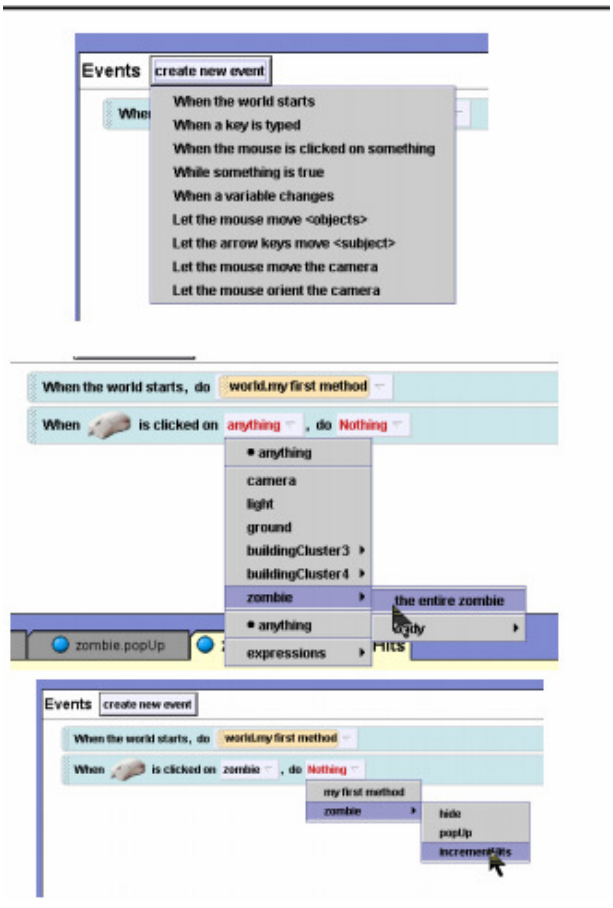
Προσθήκη ιδιότητας

- Πρέπει να αποθηκεύσουμε κάπου τις φορές που θα πρέπει να κάνουμε κλικ στο ζόμπι.
- Κλικ στις ιδιότητες
- Κλικ στη Δημιουργία Νέας Ιδιότητας
- Ονομάστε την νέα ιδιότητα numHits.

  - καθορισμός τιμής ιδιότητας σε 0.

Αντίδραση στο κλικ του mouse πάνω σε ένα ζόμπι

- Στο παράθυρο με τα Συμβάντα κλικ στο κουμπί Δημιουργία Νέου Συμβάντος.
- Αλλαγή το anything to the zombie (στο entire zombie)
- Αλλαγή Nothing σε zombie

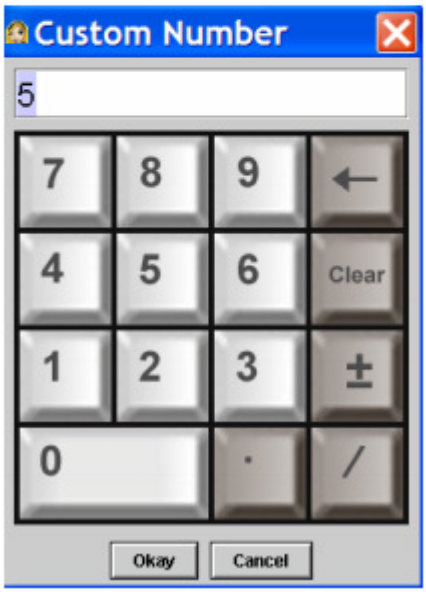


incrementHits

Μέθοδος για να τρέξει το παιχνίδι

- Κλικ στο δέντρο αντικειμένων του κόσμου
- Επιλογή methods
- Ονομάστε τη μέθοδο (myfirstmethod)
- Κλικ στο functions
  - Επιλογή α < b και εάν είναι true

    Επιλογή 1 και 1.



Όσο ο αριθμός των χτυπημάτων είναι μικρότερος από 5

- Κλικ στο ζόμπι στο δένδρο αντικειμένων
- Κλικ στις ιδιότητες

-numHits = 0

- Αλλαγή του second από 1 σε 5.

Απόκρυψη και προσμονή για τυχαίο χρονικό διάστημα

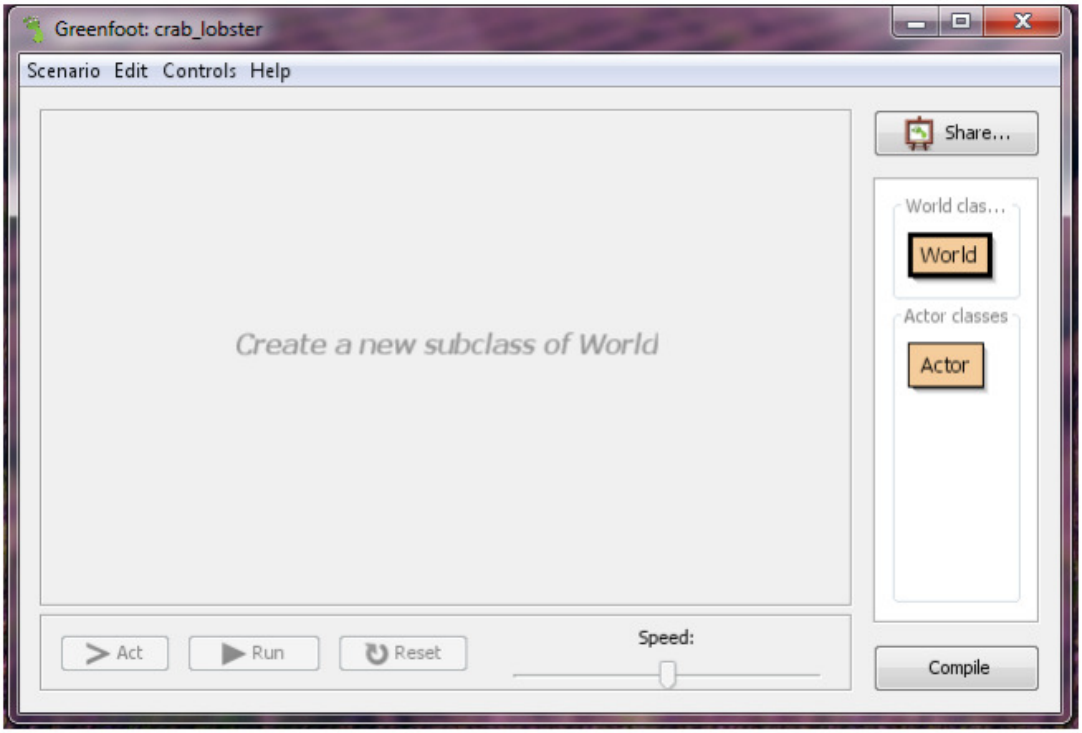
- Κλικ στις μεθόδους
- Επιλογή της μεθόδου hide
- Επιλογή 1 sec για wait.

Ενημέρωση παίκτη ότι Κέρδισε!

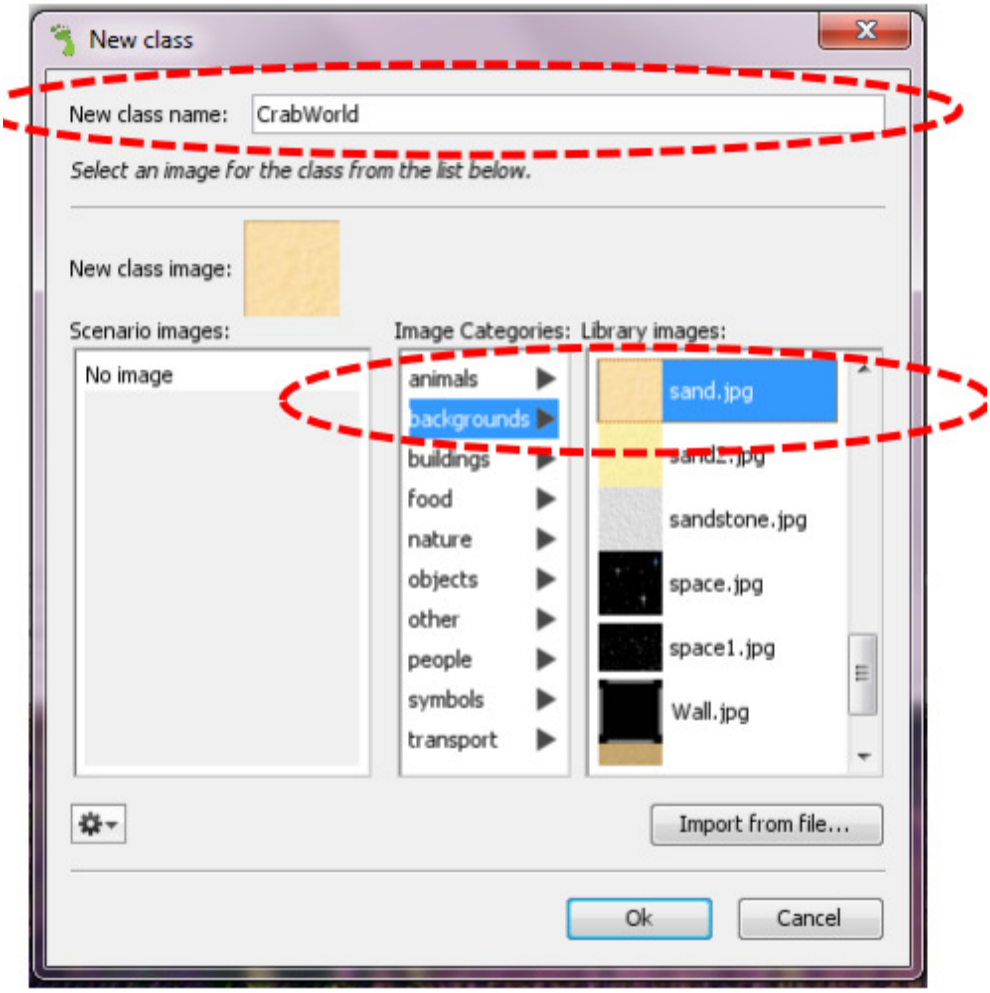
- Προσθήκη 3D κειμένου που να λέει πως ο χρήστης κέρδισε.
- Κλικ File -> Add 3D Text
- Πληκτρολογήστε το κείμενο που θέλετε
- Μην ξεχάσετε να κάνετε το αντικείμενο αόρατο στην αρχή!

4.5 Greenfoot

Δημιουργία νέου project



Δημιουργία υποκλάσης CrabWorld για τη δημιουργία σκηνικού.

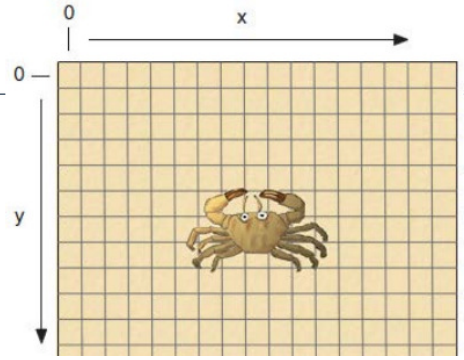


Καθορισμός διάστασης κόσμου.

```
import greenfoot.*;  // World, Actor, GreenfootImage, Greenfoot, MouseInfo

public class CrabWorld extends World
{

    public CrabWorld()
    {
        //Διάσταση του κόσμου 600x600 κελιά με μέγεθος κελιού 1x1 pixels.
        super(600, 400, 1);
    }
}
```



Ορισμός υπερκλάσης Animal για την αναπαράσταση των ζώων του παιχνιδιού.

Ιδιότητες/πεδία

- Ταχύτητα κίνησης – αριθμός pixels σε κάθε βήμα (σταθερά κλάσης): WALKING_SPEED

Λειτουργίες/μέθοδοι

- Στροφή προς τη φορά των δεικτών του ρολογιού κατά x μοίρες
- Μετακίνηση προς την τρέχουσα κατεύθυνση κατά WALKING_SPEED pixels
- Έλεγχος εντοπισμού των ορίων του κόσμου
- Έλεγχος ύπαρξης αντικειμένου ενός συγκεκριμένου τύπου στην τρέχουσα θέση
- Ένα ζώο τρώει κάποιο άλλο εφόσον βρίσκεται στην ίδια θέση

Στροφή κατά 'angle' μοίρες

```
/**
 * Στροφή προς τη φορά των δεικτών του ρολογιού κατά 'angle' μοίρες
 */
public void turn(int angle)
{
    setRotation(getRotation() + angle);
}
```

Class Actor	
`void` **`setRotation`**`(int rotation)`	Set the rotation of the object.
`int` **`getRotation`**`()`	Return the current rotation of the object.

Μετακίνηση προς την τρέχουσα κατεύθυνση

```
/**
 * Κίνηση προς τα εμπρός.
 */
public void move()
{
    double angle = Math.toRadians( getRotation() );
    int x = (int) Math.round(getX() + Math.cos(angle) * WALKING_SPEED);
    int y = (int) Math.round(getY() + Math.sin(angle) * WALKING_SPEED);

    setLocation(x, y);
}
```

Class Actor	
`int` **`getX`**`()`	Return the x-coordinate of the object's current location.
`int` **`getY`**`()`	Return the y-coordinate of the object's current location.
`void` **`setLocation`**`(int x, int y)`	Assign a new location for this object.

Έλεγχος εντοπισμού των ορίων του κόσμου

```
/**
 * Επιστρέφει true αν βρίσκεται στα όρια του κόσμου.
 */
public boolean atWorldEdge()
{
    if(getX() < 20 || getX() > getWorld().getWidth() - 20)
        return true;
    if(getY() < 20 || getY() > getWorld().getHeight() - 20)
        return true;
    return false;
}
```

Class Actor	
World **getWorld**()	Return the world that this object lives in.
int **getX**()	Return the x-coordinate of the object's current location.
int **getY**()	Return the y-coordinate of the object's current location.

Class World	
int **getHeight**()	Return the height of the world (in number of cells).
int **getWidth**()	Return the width of the world (in number of cells).

Έλεγχος ύπαρξης άλλου τύπου αντικειμένου στην τρέχουσα θέση

```
/**
 * Επίστρεψε true αν υπάρχει αντικείμενο τύπου 'clss' στην τρέχουσα θέση,
 * false διαφορετικά.
 */
public boolean canSee(Class clss)
{
    Actor actor = getOneObjectAtOffset(0, 0, clss);
    return actor != null;
}
```

Class Actor	
protected Actor **getOneObjectAtOffset** (int dx, int dy, Class cls)	Return one object that is located at the specified cell (relative to this objects location).

Ένα ζώο τρώει κάποιο άλλο που βρίσκεται στην ίδια θέση

```
/**
 * Προσπάθησε να φας ένα αντικείμενο τύπου'clss'. Αν δεν υπάρχει αντικείμενο
 * αυτού του τύπου στην τρέχουσα θέση η μέθοδος δεν έχει κάποιο αποτέλεσμα
 */
public void eat(Class clss)
{
   Actor actor = getOneObjectAtOffset(0, 0, clss);
   if(actor != null) {
         getWorld().removeObject(actor);
   }
}
```

Class Actor	
protected Actor **getOneObjectAtOffset** (int dx, int dy, Class cls)	Return one object that is located at the specified cell (relative to this objects location).
World **getWorld**()	Return the world that this object lives in.

Class World	
void **removeObject**(Actor object)	Remove an object from the world.

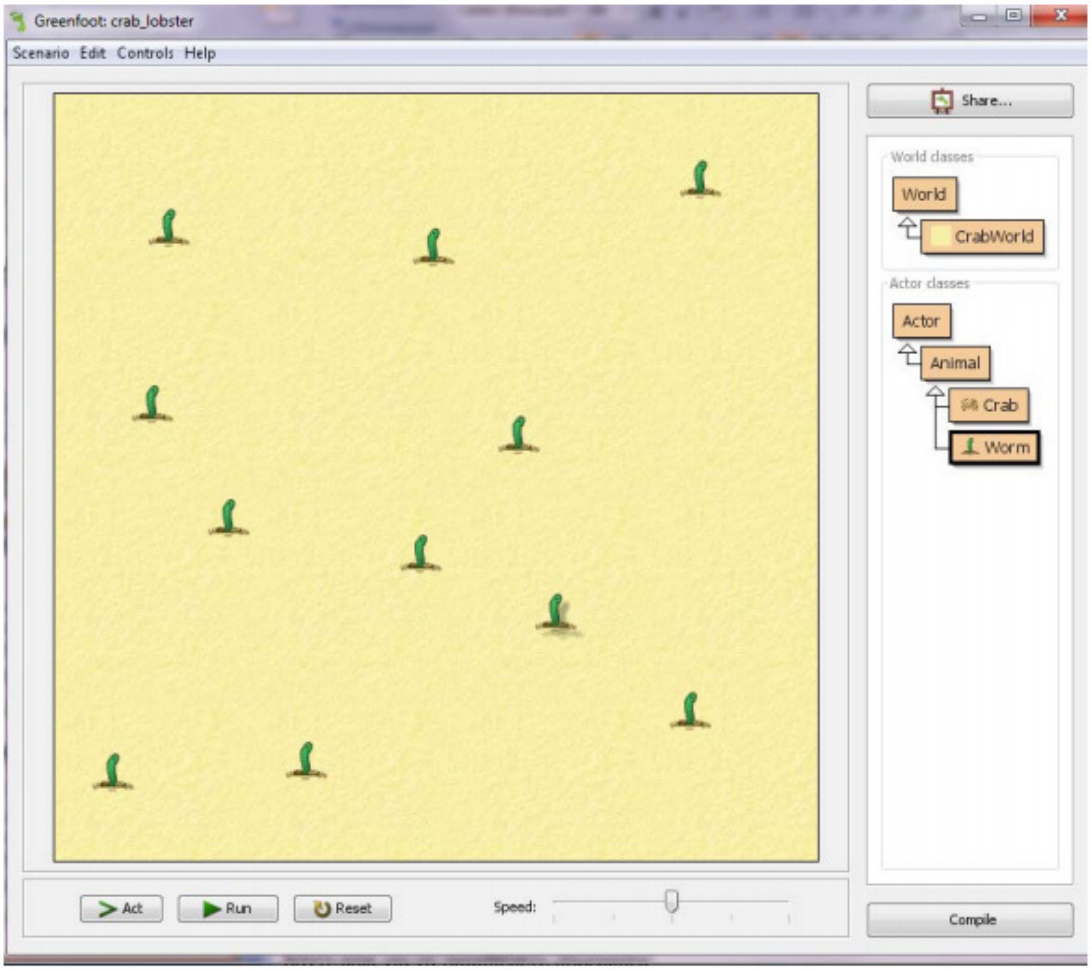
Ορισμός κλάσης Worm για την αναπαράσταση των σκουληκιών

Προς το παρόν δεν έχουν κάποια συμπεριφορά.

Κάθονται και περιμένουν να τους φάνε τα καβούρια.

Φυσικά, υπάρχουν οι μέθοδοι που κληρονομούνται από τις κλάσεις Animal και Actor. Θα μπορούσαν τα σκουλήκια να κινούνται και αυτά και μάλιστα με μεγάλη ταχύτητα για να ξεφεύγουν από τα καβούρια.

Με δεξί κλικ πάνω στην κλάση Worm επιλέγετε new Worm() από το αναδυόμενο μενού και «αφήνετε» το σκουλήκι που δημιουργείται μέσα στον κόσμο. Για να προσθέσετε πολλά στιγμιότυπα, έχοντας επιλέξει την κλάση, κρατάτε πατημένο το Shift και κάνετε κλικ σε εκείνα τα σημεία που θέλετε να προστεθούν σκουλήκια. Με δεξί κλικ στην περιοχή του παιχνιδιού, επιλέγετε Save World για να αποθηκευτεί το σκηνικό του παιχνιδιού.



Ορισμός κλάσης Crab για την αναπαράσταση των καβουριών

Ιδιότητες/πεδία

Αριθμός σκουληκιών που έχει φάει - wormsEaten

Λειτουργίες/μέθοδοι

Στρίβει προς τα αριστερά και δεξιά ελεγχόμενο από τον παίκτη

Ψάχνει για σκουλήκι και το τρώει – αν φάει 8 σκουλήκια το παιχνίδι τερματίζειι Κάθε στιγμή ενεργεί (act) ως εξής:

Στρίβει αν το ζητήσει ο παίκτης

Κινείται

Ψάχνει για σκουλήκι και αν βρει το τρώει

Στρίβει αριστερά και δεξιά ελεγχόμενο από τον χρήστη

```
/**
 * Ελεγξε αν έχει πατηθεί το αριστερό ή δεξί βελάκι από το πληκτρολόγιο
 * και στρίψε κατάλληλα.
 */
public void checkKeypress()
{
    if (Greenfoot.isKeyDown("left"))
    {
          turn(-4);
    }
    if (Greenfoot.isKeyDown("right"))
    {
         turn(4);
    }
}
```

Class Greenfoot	
`static boolean` **`isKeyDown`**`(String keyName)`	Check whether a given key is currently pressed down.

Ψάχνει για σκουλήκι και το τρώει

```
/**
 * Ελεγξε αν έχεις πέσει πάνω σε ένα σκουλήκι προκειμένου να το φας.
 * Αν έχεις φάει 8 σκουλήκια κερδίζεις.
 */
public void lookForWorm()
{
    if ( canSee(Worm.class) )
    {
          eat(Worm.class);
          Greenfoot.playSound("slurp.wav");

          wormsEaten = wormsEaten + 1;
          if (wormsEaten == 8)
          {
                  Greenfoot.playSound("fanfare.wav");
                  Greenfoot.stop();
          }
     }
}
```

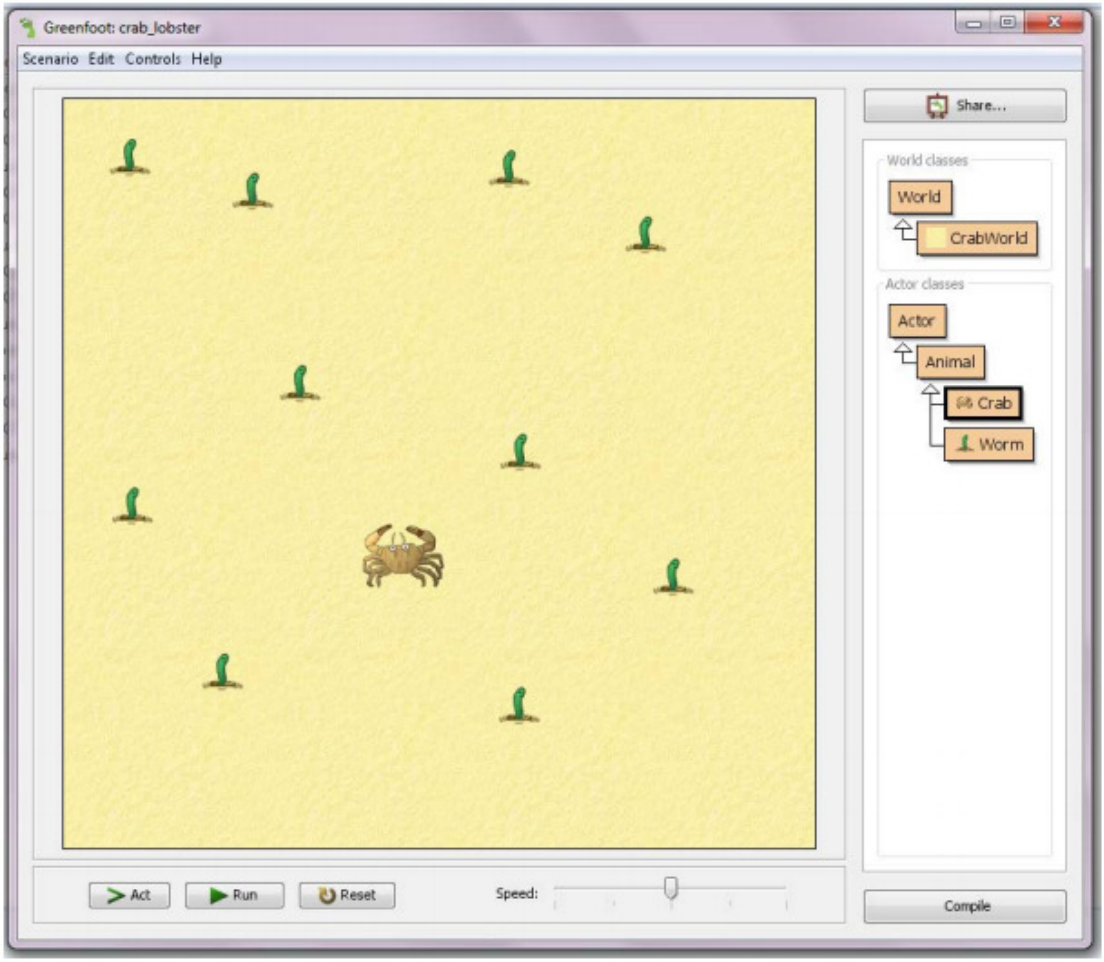
Class Greenfoot	
`static void` **`playSound`**`(String soundFile)`	Play sound from a file.
`static void` **`stop`**`()`	Stop the simulation.

Βασική συμπεριφορά καβουριού

```
/**
 * Υλοποίηση βασικής συμπεριφοράς καβουριού. Καλείται κάθε φορά που πατάμε
 * το κουμπί 'Act' ή 'Run' στο περιβάλλον.
 */
public void act()
{
    checkKeypress();
    move();
    lookForWorm();
}
```

Προσθήκη καβουριού στο παιχνίδι

Μπορείτε πλέον να προσθέσετε ένα καβούρι και να δοκιμάσετε την πρώτη έκδοση του παιχνιδιού. Δεν θα ήταν ωραία ο βασικός μας χαρακτήρας να κινείται πιο “φυσικά”; Αυτό μπορεί να γίνει εύκολα χρησιμοποιώντας και μια 2η εικόνα για το καβούρι με τα πόδια του σε διαφορετική θέση…



Βελτιώνοντας την κίνηση του καβουριού

```
public class Crab extends Animal
{
    private GreenfootImage image1;
    private GreenfootImage image2;
    …
    public Crab()
    {
        image1 = new GreenfootImage("crab.png");
        image2 = new GreenfootImage("crab2.png");
        setImage(image1);
        wormsEaten = 0;
    }

    public void act()
    {
        …
        switchImage();
    }
```

a) crab with legs out

b) crab with legs in

```
public void switchImage()
{
   if (getImage() == image1)
   {
        setImage(image2);
   }
   else
   {
        setImage(image1);
   }
}
```

Προσθήκη αντίπαλου

Ορισμός κλάσης Lobster για την αναπαράσταση των αστακών

Λειτουργίες/μέθοδοι

Στρίβει τυχαία

Στρίβει όταν φτάσει στα όρια του κόσμου

Ψάχνει για καβούρι και το τρώει – το παιχνίδι τερματίζει

και φυσικά συνεχώς κάθε στιγμή ενεργεί (act) ως εξής:

Στρίβει όταν φτάσει στα όρια του κόσμου

Στρίβει τυχαία

Κινείται

Ψάχνει για καβούρι και αν βρει το τρώει

Φυσικά, υπάρχουν και οι μέθοδοι που κληρονομούνται από τις κλάσεις Animal και Actor.

Στρίβει αν είναι στα όρια του κόσμου

```
/**
 * Ελεγξε αν είσαι στα όρια του κόσμου και στρίψε.
 */
public void turnAtEdge()
{
    if ( atWorldEdge() )
    {
          turn(17);
    }
}
```

Στρίβει ή όχι τυχαία δεξιά ή αριστερά

```
/**
 * Αποφασίζεται τυχαία να στρίψει ή όχι (πιθανότητα να στρίψει 9 /100)
 * Αν στρίψει, στρίβει τυχαία λίγο προς τα αριστερά ή δεξιά (-45..45 μοίρες)
 */
public void randomTurn()
{
    if (Greenfoot.getRandomNumber(100) > 90) {
          turn(Greenfoot.getRandomNumber(90)-45);
    }
}
```

Class Greenfoot	
`static int` **`getRandomNumber`**`(int limit)`	Return a random number between 0 (inclusive) and limit

```
/**
 * Ελεγξε αν έπεσες πάνω σε ένα καβούρι. Αν ναι απομάκρυνε το και τελείωσε
 * το παιχνίδι
 */
public void lookForCrab()
{
    if ( canSee(Crab.class) )
    {
          eat(Crab.class);
          Greenfoot.playSound("au.wav");
          Greenfoot.stop();
     }
}
```

Βασική συμπεριφορά αστακού

```
/**
 * Υλοποίηση βασικής συμπεριφοράς αστακού. Καλείται κάθε φορά που πατάμε
 * το κουμπί 'Act' ή 'Run' στο περιβάλλον.
 */
public void act()
{
    turnAtEdge();
    randomTurn();
    move();
    lookForCrab();
}
```

5 Συμπεράσματα

Μετά την παραπάνω ανάλυση είναι πλέον σκόπιμο ίσως να καταλήξουμε στην πιο προσιτή εφαρμογή για την εκμάθηση προγραμματισμού σε νεαρές ηλικίες. Το σίγουρο είναι ότι έχουν γίνει πολλές προσπάθειες τα τελευταία χρόνια και οι περισσότερες είναι αξιόλογες, για να προσεγγίσουμε τους μικρούς μας φίλους ώστε να θέλουν να μάθουν προγραμματισμού.

Ενώ η πιο διαδεδομένη εφαρμογή είναι η Scratch, και μάλιστα είναι από τις πολύ λίγες με τόσο καλό documentation και μεγάλη γκάμα σε υλικό, στην παρούσα ανάλυση διαπιστώθηκε μεγαλύτερη ευκολία στο Greenfoot.

Φυσικά, η Scratch είναι για πιο νεαρές ηλικίες, είναι ας πούμε η πρώτη επαφή κάποιου με προγραμματισμό. Δεν μπορεί κάποιος δάσκαλος να μάθει στα παιδιά της αρχές του προγραμματισμού με Java.

Παρ' όλα αυτά, επειδή το παρόν του προγραμματισμού κυριαρχείτε από την αντικειμενοστρέφεια, είναι πολύ σημαντικό το επόμενο βήμα του παιδιού να είναι με τέτοιο προγραμμαστισμό. Για αυτό το λόγο προτείνετε η χρήση του εργαλείου greenfoot.

6 Βιβλιογραφία

Cohen, L., & Manion, L. (1989). Research Methods in Education, Routledge, London.

Crawford, C. (1982). The Art of Computer Game Design. Retrieved from:

www.vancouver.wsu.edu/ fac/peabody/game-book/ Coverpage.html Eckerdal, A. (2009). Novice Programming Students' Learning of Concepts and Practise. Dissertation presented at Mathematics and Computer Science, Dept of Information Technology, Upsalla University, Sweeden, http://uu.diva-ortal.org/smash/record.jsf?pid=diva2: 173221 Freund, S. N. & Roberts, E. S. (1996). THETIS: An ANSI C programming environment designed for introductory use. ACM, SIGCSE '96 2/96, Philadelphia, PA USA, 300-304. Ford, J.L. (2008). Scratch Programming for Teens. Canada: Course Technology PTR. Hadjerrouit, S. (2008). Towards a Blended Learning Model for Teaching and Learning Computer Programming: A Case Study. Informatics in Education, 2008, Vol. 7, No. 2, 181–210. Prensky, M. (2001). Digital game-based learning, Mc Graw-Hill, New York. Kordaki, M. (2003). The effect of tools of a computer microworld on students' strategies regarding the concept of conservation of area. Educational Studies in Mathematics, 52, 177-209 Kordaki, M. (2010). A drawing and multi-representational computer environment for beginners' learning of programming using C: Design and pilot formative evaluation. Computers and Education Vol. 54(1), pp. 69-87. Kordaki, M. (2012; in print). Diverse categories of programming learning activities could be performed within Scratch. In Proceedings of 4 th World Conference on Educational Sciences, 02-02/2012, Barcelona, Spain, Procedia- Social and Behavioral Sciences. Κορδάκη, Μ. & Ψώμος, Π. (2012). Scratch: 11 διαφορετικές κατηγορίες μαθησιακών δραστηριοτήτων. 6ο Πανελλήνιο Συνέδριο 'Διδακτική της Πληροφορικής', Διοργάνωση: ΕΠΥ και Παιδαγωγικό τμήμα Παν/μίου Δυτικής Μακεδονίας, (σελ.591-594), 20-22 Απριλίου 2012, Φλώρινα, Ελλάδα. Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., Kafai, Y., (2009). Scratch: Programming for All, November 2009, Communications of the ACM, 52(11), pp. 60-67. Soloway, E. & Spohrer, J. C. (1989). Studying the novice programmer. Hillside, N.J. Erlbaum. Wing, J. (2006). Viewpoint: computational thinking. Communications of the ACM. Vol. 49(3), pp. 33-35. Winslow, L. E. (1996). Programming Pedagogy. SIGCSE Bulletin, Vol. 28 (3), 17-22 Robins, A., Rountree, J. and Rountree, N. (2003). Learning and teaching Programming: A Review and Discussion. Computer Science Education, 13(2), pp. 137-172.
http://www.greenfoot.org/door
http://www.edutopia.org/blog/15-ways-teaching-students-coding-vicki-davis
http://www.edutopia.org/blog/7-apps-teaching-children-coding-anna-adam