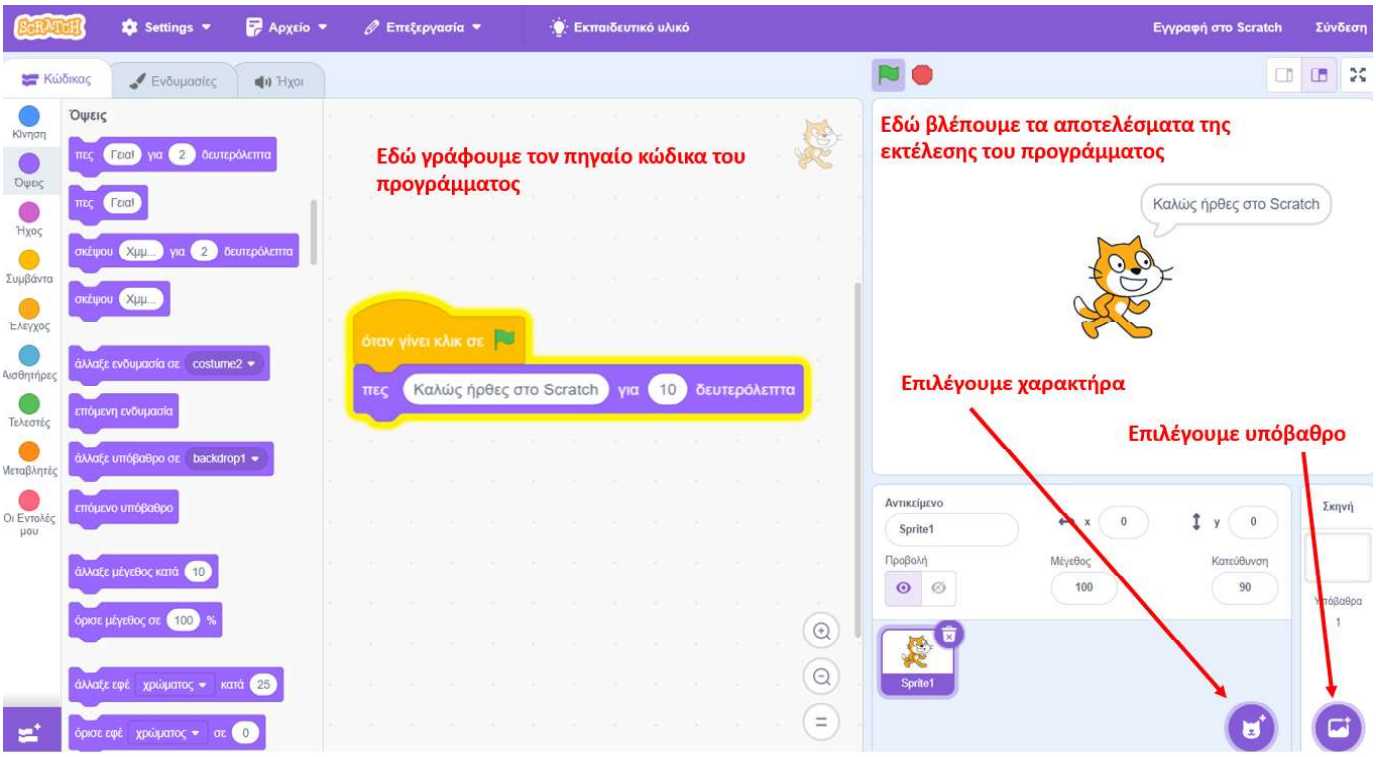
9.2 Το περιβάλλον προγραμματισμού Scratch

Το Scratch είναι ένα εκπαιδευτικό περιβάλλον προγραμματισμού με πλακίδια (block-based) που μας δίνει τη δυνατότητα να δημιουργούμε διαδραστικές ιστορίες, κινούμενα σχέδια, προσομοιώσεις και παιχνίδια. Όλα αυτά μπορούμε να τα μοιραστούμε μέσω του Διαδικτύου με άλλους χρήστες. Επίσης, μπορούμε να χρησιμοποιήσουμε ή να επεκτείνουμε τα προγράμματα άλλων χρηστών στην κοινότητα του Scratch. Στον δικτυακό τόπο https://scratch.mit.edu/ μπορείτε να δημιουργήσετε το δικό σας έργο ή να περιηγηθείτε στα έργα που έχουν μοιραστεί άλλοι προγραμματιστές και να επεκτείνετε κάποια από αυτά ή να δείτε τον κώδικα με τον οποίο έχουν δημιουργηθεί. Το Scratch έχει αναπτυχθεί από μια μικρή ομάδα ερευνητών του MIT.



Εικόνα 9.2. Το περιβάλλον προγραμματισμού Scratch (https://scratch.mit.edu/)

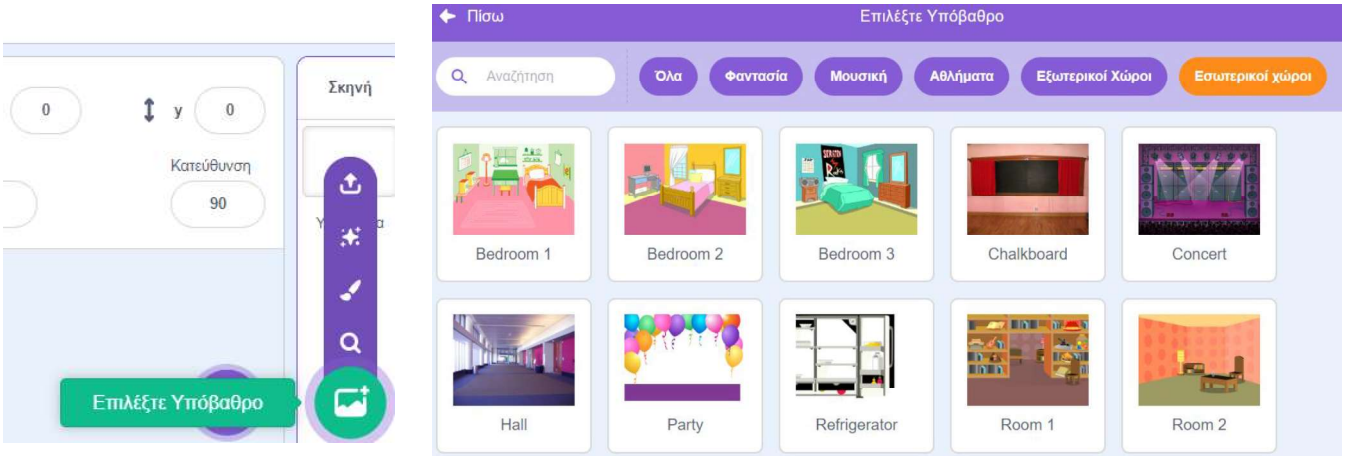
Στο πλαϊνό μενού φαίνονται όλες οι θεματικές κατηγορίες εντολών, όπως είναι οι εντολές κίνησης, οι όψεις, τα συμβάντα, οι εντολές ελέγχου κ.λπ. Κάθε ομάδα εντολών έχει αντιστοιχιστεί σε ένα χρώμα. Το πρώτο μας πρόγραμμα χρησιμοποίησε δύο εντολές:

Ένα συμβάν το οποίο ξεκινάει την εκτέλεση των εντολών όταν γίνει κλικ στην πράσινη σημαία.

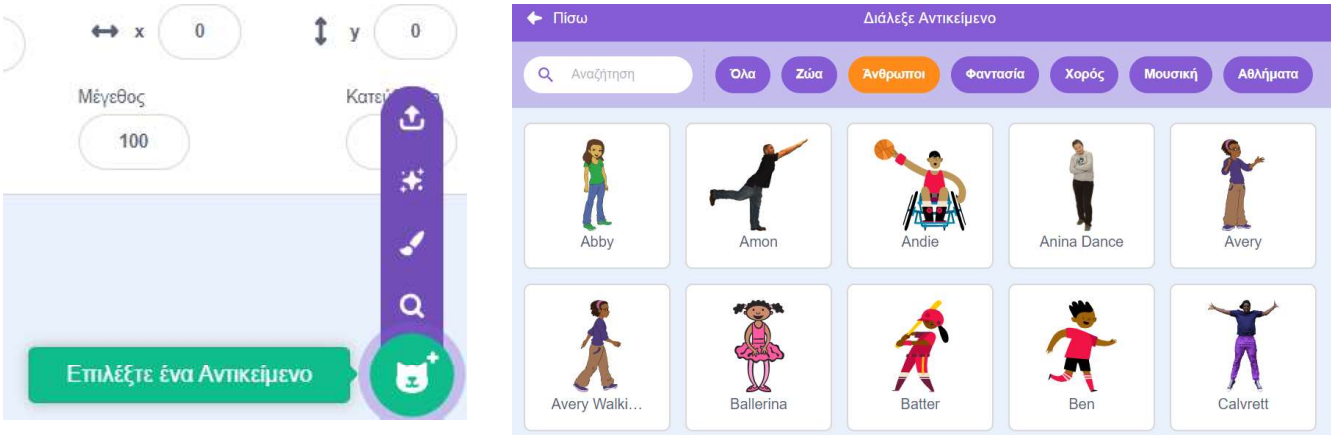
και μια εντολή από τις όψεις η οποία εμφανίζει ένα μήνυμα σε μορφή διαλόγου από τη γάτα για 10 δευτερόλεπτα.

9.3 Ψηφιακή αφήγηση στο Scratch

Αξιοποιώντας κάποιες από τις ομάδες εντολών των συμβάντων και των εντολών ελέγχου μπορούμε να δημιουργήσουμε μικρές διαδραστικές ιστορίες σαν σύντομα κόμικ. Κάτω δεξιά επιλέγουμε υπόβαθρο από τα ήδη προτεινόμενα ή μπορούμε να σχεδιάσουμε το δικό μας στη ζωγραφική ή να μεταφορτώσουμε (upload) μια εικόνα-υπόβαθρο που βρήκαμε στο Διαδίκτυο.



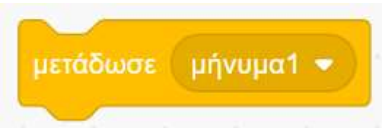
Αφού επιλέξουμε το υπόβαθρο που ταιριάζει στην ιστορία μας, πρέπει να επιλέξουμε και δύο χαρακτήρες:



Για κάθε χαρακτήρα μπορούμε να επιλέξουμε πού θα κοιτάει, αν θα κάθεται, θα περπατάει ή θα έχει κάποια άλλη στάση. Αφού επιλέξουμε τον χαρακτήρα που θέλουμε, στη συνέχεια, επιλέγουμε πάνω αριστερά στις ενδυμασίες και εκεί ορίζουμε τη στάση και την ενδυμασία του. Οι εντολές που θα χρησιμοποιήσουμε για να εμφανίζονται οι διάλογοι με τη σειρά και όχι ο ένας πάνω στον άλλον ανήκουν στην ομάδα συμβάντων.

Όταν γίνει κλικ στην πράσινη σημαία εκτελούνται οι εντολές που ακολουθούν.

Όταν λάβω το μήνυμα από κάποιον/-α εκτελούνται οι εντολές που ακολουθούν.

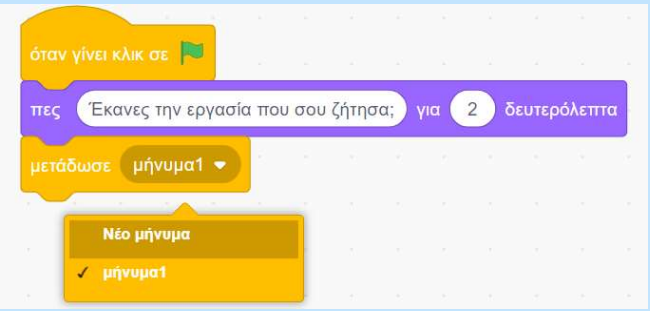


Μεταδίδω το μήνυμα «μήνυμα1» το οποίο περιμένει ο άλλος χαρακτήρας.

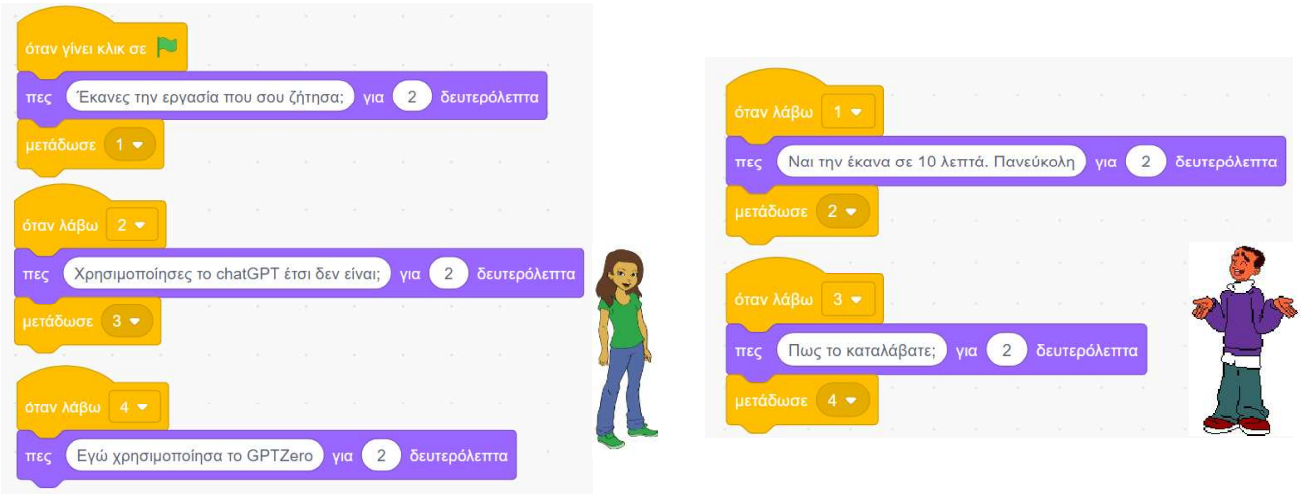
Παράδειγμα 1

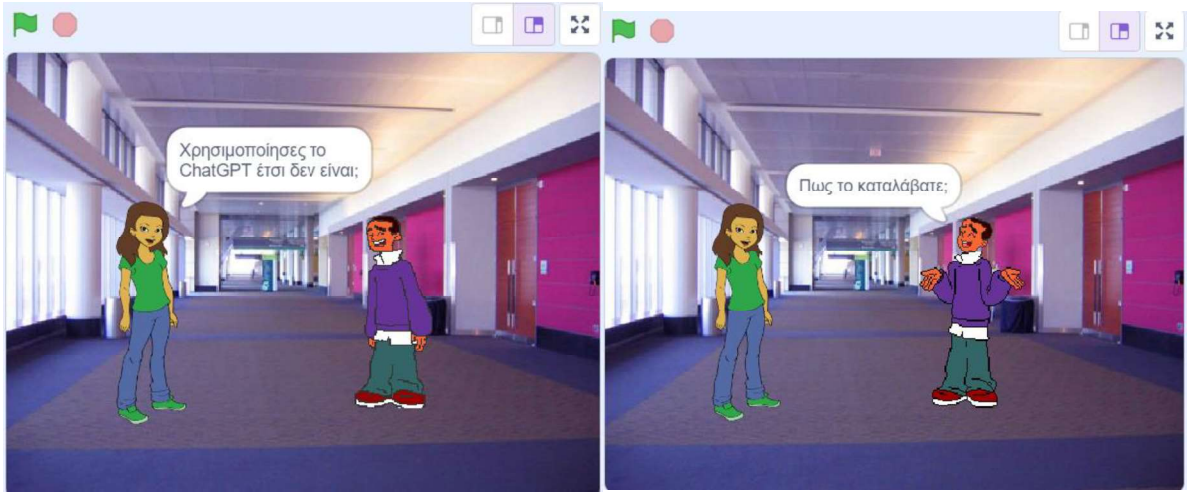
Ας επιλέξουμε δύο χαρακτήρες, την κυρία Χρύσα και τον μαθητή της Οδυσσέα.

Η κυρία Χρύσα κάνει την έναρξη της συνομιλίας με τον μαθητή. Χρησιμοποιούμε την εντολή **πες** από τις όψεις και την εντολή **μετάδωσε** μήνυμα από τα συμβάντα. Για αυτό δημιουργούμε ένα νέο μήνυμα με την εντολή <**μετάδωσε μήνυμα**> στο οποίο μπορούμε να δώσουμε ό,τι όνομα θέλουμε. Εδώ δίνουμε τον αριθμό 1. Μετά τη μετάδοση του μηνύματος η εκτέλεση του μπλοκ εντολών σταματάει.



Ο Οδυσσέας περιμένει να λάβει το μήνυμα 1 για να απαντήσει στην καθηγήτριά του και με τη σειρά του της μεταδίδει το μήνυμα 2, ώστε η καθηγήτρια να καταλάβει ότι ήρθε η σειρά της να μιλήσει. Για αυτό χρησιμοποιούμε την εντολή <**όταν λάβω μήνυμα**>. Με αυτόν τον τρόπο ο δεύτερος συνομιλητής μιλάει μόνο όταν τελειώσει η πρώτη, έτσι ώστε να μη μιλάνε ταυτόχρονα.

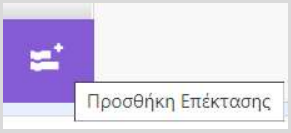




9.5 Η δύναμη της επανάληψης

Ομάδα Εντολών με λειτουργίες πένας

Στο Scratch 3 η πένα δε βρίσκεται μέσα στις βασικές εντολές στο πλαϊνό μενού. Θα πρέπει να πάτε κάτω αριστερά στην προσθήκη επέκτασης και στη συνέχεια να επιλέξετε την πένα, ώστε να εμφανιστούν οι εντολές.



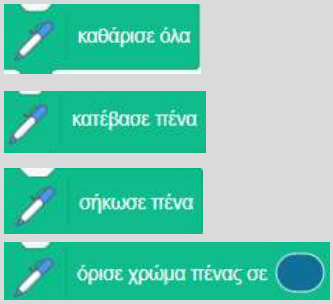
Βασικές Λειτουργίες Πένας

Καθαρίζει όλη την οθόνη από όλα τα σχήματα που έχουμε σχηματίσει.

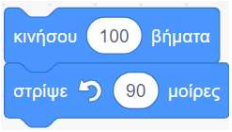
Κατεβάζει την πένα για το αντικείμενο, έτσι ώστε να αφήνει πίσω του ένα ίχνος όταν κινείται και με αυτόν τον τρόπο να ζωγραφίζει σχήματα.

Ανεβάζει την πένα έτσι ώστε το αντικείμενο να μην αφήνει ίχνος όταν κινείται.

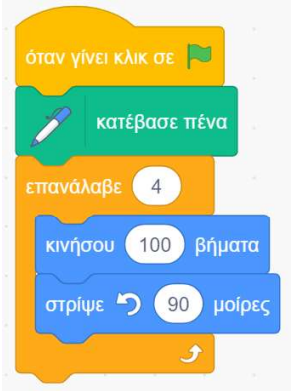
Ορίζει το χρώμα της πένας.

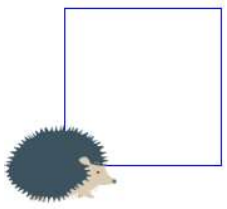


Για να σχηματίσει ένα τετράγωνο, ο σκαντζόχοιρος θα πρέπει να σχεδιάσει τέσσερις γραμμές. Όταν φτάσει στο τέλος της γραμμής, θα πρέπει να στρίψει 90° έτσι ώστε να σχηματιστεί ορθή γωνία. Παρατηρούμε ότι

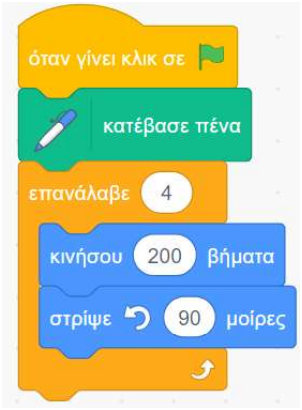


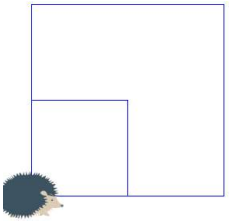
οι εντολές κινήσου και στρίψε επαναλαμβάνονται τέσσερις φορές. Αντί λοιπόν να τις γράφουμε τέσσερις φορές μπορούμε να χρησιμοποιήσουμε την εντολή **Επανάλαβε** από την ομάδα εντολών ελέγχου. Η εντολή αυτή εκτελεί τις εντολές που είναι οριοθετημένες μέσα στην επανάληψη τόσες φορές όσες θέλουμε. Εδώ έχουμε δώσει τον αριθμό 4. Αν θέλαμε να σχεδιάσουμε ένα δεκάγωνο, θα γλιτώναμε αρκετές γραμμές κώδικα.



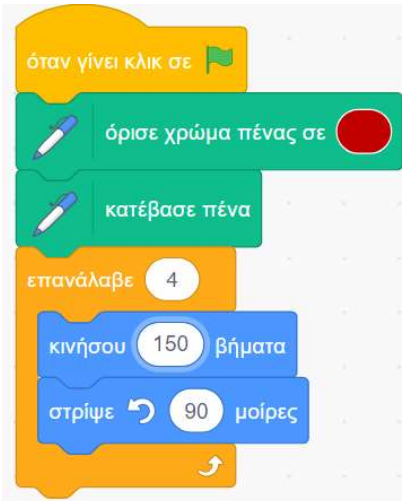


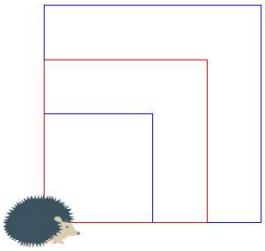
Είναι, όμως, το μόνο πλεονέκτημα της εντολής επανάληψης ότι γλιτώνουμε χώρο; Ας δούμε το διπλανό παράδειγμα, όπου αποφασίσαμε να σχεδιάσουμε ένα τετράγωνο με διπλάσια πλευρά. Παρατηρώντας το τελικό αποτέλεσμα, διαπιστώνουμε ένα πρόβλημα. Ενώ θέλαμε να σχηματιστεί ένα μόνο τετράγωνο πλευράς 200, βλέπουμε και ένα τετράγωνο πλευράς 100.





Πώς προέκυψε αυτό; Για να διερευνήσουμε περισσότερο αυτό το πρόβλημα αλλάζουμε πάλι την πλευρά του τετραγώνου και την κάνουμε 150 για να δούμε τι θα συμβεί. Ταυτόχρονα, όμως, αλλάζουμε το χρώμα της πένας σε κόκκινο. Αυτό που κάνουμε αυτή τη στιγμή λέγεται **εκσφαλμάτωση** (debugging). Ψάχνουμε, δηλαδή, να βρούμε ένα bug (σφάλμα) στον κώδικά μας. Έτσι καταλήγουμε στο διπλανό πρόγραμμα.

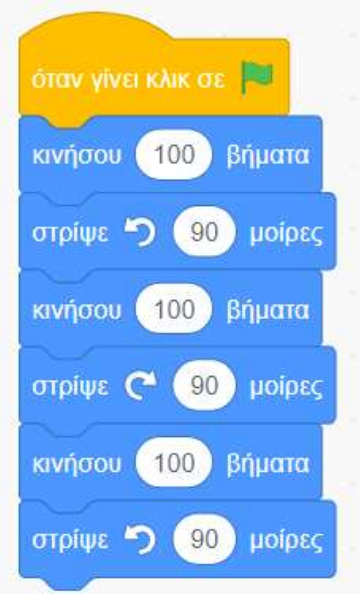


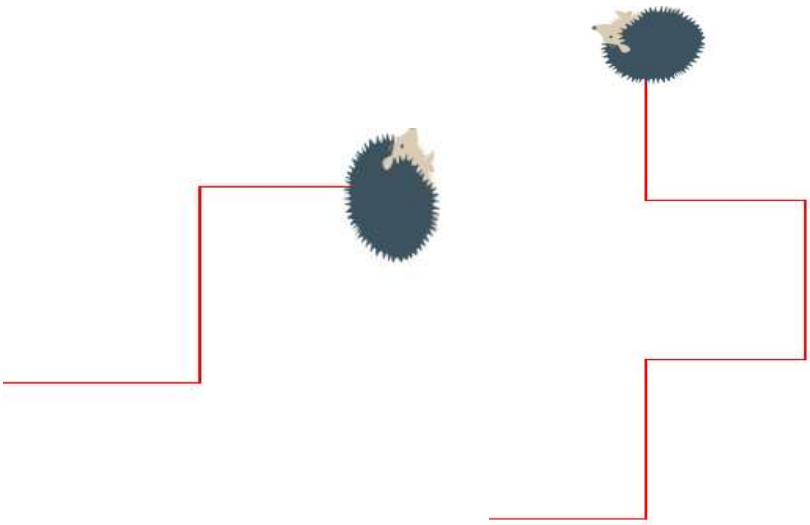


Έχουμε προσθέσει και μια εντολή **περίμενε** ώστε να φαίνεται η κίνηση του σκαντζόχοιρου, ενώ σχεδιάζει το σχήμα. Εκτός από την εντολή **περίμενε**, έχουμε αυξήσει και την πλευρά του τετραγώνου από 100 σε 150. Πόσες αλλαγές θα χρειάζονταν αν δεν χρησιμοποιούσαμε την εντολή επανάληψης αλλά γράφαμε όλες τις εντολές όπως πριν; Αυτό αποτελεί ένα σημαντικό πλεονέκτημα όταν γράφουμε κώδικα. Να μπορούμε να πετύχουμε τον σκοπό μας με τις ελάχιστες δυνατές αλλαγές. Η δυνατότητα αυτή λέγεται **επεκτασιμότητα** (extensibility) του κώδικά μας, γιατί μας δίνει τη δυνατότητα να προσαρμόζουμε τις εφαρμογές μας εύκολα στις νέες εξελίξεις χωρίς πολλές αλλαγές.

Τώρα φαίνεται τι έχει συμβεί. Το κόκκινο τετράγωνο είναι αυτό που σχηματίστηκε με την τελευταία εκτέλεση του προγράμματός μας. Τα άλλα δύο σχηματίστηκαν από τις προηγούμενες εκτελέσεις και παρέμειναν εκεί γιατί δεν δώσαμε εντολή να καθαρίσει η οθόνη.

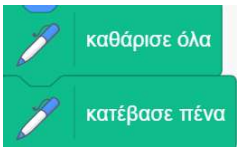
Ας δούμε και το επόμενο παράδειγμα, όπου ο σκαντζόχοιρος δε σχεδιάζει ένα κλειστό σχήμα, με αποτέλεσμα να μην επιστρέψει στη θέση από την οποία ξεκίνησε. Το δεύτερο σχήμα προκύπτει από την εκτέλεση του ίδιου προγράμματος για δεύτερη φορά. Ο σκαντζόχοιρος όχι μόνο δεν έχει επιστρέψει στο σημείο εκκίνησης, αλλά δείχνει προς την αντίθετη κατεύθυνση.



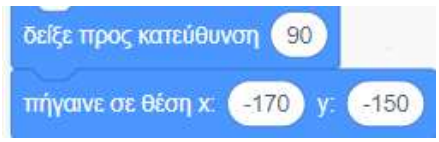


Αυτό που έπρεπε να γίνει στο ξεκίνημα λέγεται **αρχικοποίηση** (initialization). Δηλαδή, να επαναφέρουμε όλα τα αντικείμενα του προγράμματός μας στην αρχική κατάσταση.

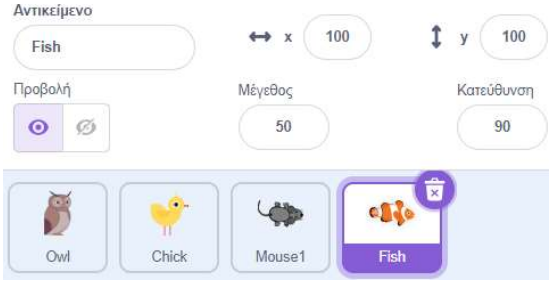
Αυτό σημαίνει ότι θα πρέπει να καθαρίσουμε το σκηνικό και να κατεβάσουμε την πένα, ώστε ο σκαντζόχοιρος να σχεδιάζει το σχήμα που θέλουμε κατά την κίνηση.



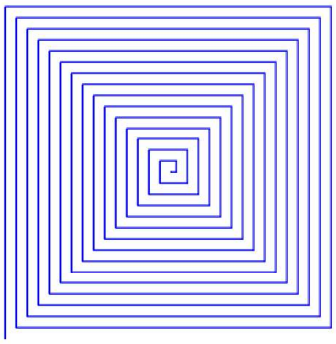
Επίσης, θα πρέπει να μεταφέρουμε τον σκαντζόχοιρο στο αρχικό σημείο από όπου ξεκίνησε και να τον στρέψουμε προς τα δεξιά.
Έτσι, θα σχηματίσει γωνία 90° με τον κατακόρυφο άξονα. Οι συντεταγμένες (x,y)=(-170,-150) αναφέρονται στο σημείο εκκίνησης που βρίσκεται κάτω αριστερά.



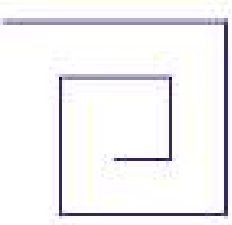
Αρχικά μεταφέρουμε τον σκαντζόχοιρο στην κάτω αριστερή γωνία.
Το σημείο αυτό έχει συντεταγμένες (-170, -150). Δηλαδή βρίσκεται 170 βήματα αριστερά από το κέντρο και 150 βήματα κάτω από το κέντρο.
Οι συντεταγμένες ενός σημείου στο Scratch είναι η οριζόντια και κατακόρυφη απόσταση από το κέντρο της οθόνης. Άρα το κέντρο έχει συντεταγμένες (0,0).
Αν θέλουμε να βρούμε τις ακριβείς συντεταγμένες ενός αντικειμένου αφού το μεταφέρουμε στη θέση που θέλουμε, κοιτάμε κάτω αριστερά στα χαρακτηριστικά του. Εδώ φαίνεται ότι το ψάρι του οποίου έχουμε μειώσει το μέγεθος στο μισό βρίσκεται στο σημείο με συντεταγμένες (100, 100).



9.6 Η έννοια της μεταβλητής



Θέλουμε να σχεδιάσουμε το διπλανό σχήμα. Παρατηρούμε ότι, ενώ φαίνεται ότι κάθε φορά η στροφή είναι 90° όπως όταν σχεδιάζεται ένα τετράγωνο, το μήκος κάθε πλευράς αλλάζει συνέχεια. Θέλουμε η πλευρά να ξεκινάει από ένα αρχικό μήκος και, στη συνέχεια, να αυξάνει σταδιακά. Θα πρέπει να βρούμε έναν τρόπο να αναπαραστήσουμε την πλευρά που μεταβάλλεται μετά από κάθε κίνηση, δηλαδή μια ποσότητα που μεταβάλλεται. Γι' αυτό χρησιμοποιούμε τη **μεταβλητή**. Η μεταβλητή είναι μια θέση στη μνήμη, η οποία περιέχει μια τιμή που μπορεί να αλλάξει, όποτε θελήσουμε.



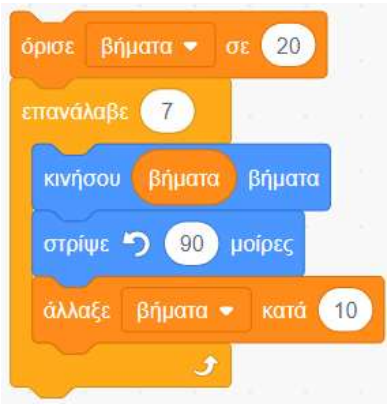
Δίπλα φαίνεται η πρώτη προσπάθεια που έκανε μια μαθήτρια. Ξεκίνησε με πλευρά 20 και σε κάθε κίνηση αύξανε την πλευρά κατά 10. Έτσι, πήρε το διπλανό σχήμα. Προφανώς, δεν μπορούμε να συνεχίσουμε έτσι, γιατί η έκταση του προγράμματος θα είναι 5-6 σελίδες.

Σκεφτόμαστε, λοιπόν, πώς θα μπορούσαμε να γράψουμε αυτές τις εντολές με την μορφή επανάληψης. Θα πρέπει να βρούμε έναν τρόπο να γράψουμε έναν αλγόριθμο της μορφής:

Επανάλαβε 7 φορές
  Κινήσου βήματα
  Στρίψε αριστερά 90°
  Αύξησε την **πλευρά** κατά 10

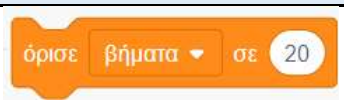
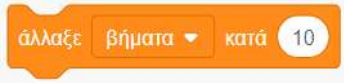
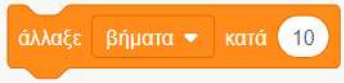
Για αυτό θα χρειαστεί να ορίσουμε μια μεταβλητή με όνομα **βήματα**.

Μετά την εισαγωγή της μεταβλητής, μπορούμε να γράψουμε το διπλανό τμήμα κώδικα με τη χρήση εντολής επανάληψης, έτσι ώστε να επιτελεί την ίδια ακριβώς λειτουργία.



Σκεφτείτε πόσες αλλαγές θα χρειαστούν σε κάθε περίπτωση αν θελήσουμε να γίνει πιο πυκνό το σπιράλ και αλλάξουμε το 10 σε 5.

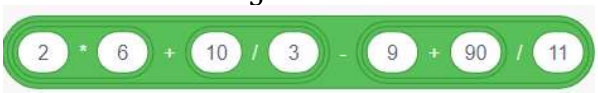
Κάθε μεταβλητή έχει ένα όνομα και αναφέρεται σε μια θέση στη μνήμη του υπολογιστή. Όταν θέσουμε μια τιμή σε αυτή τη θέση η προηγούμενη τιμή που είχε διαγράφεται, όπως φαίνεται στο παρακάτω παράδειγμα:

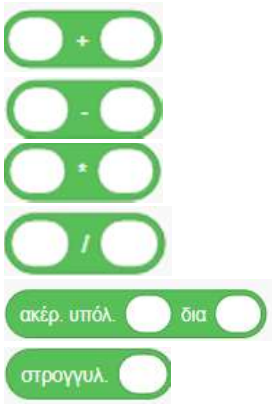
Εντολή	Αποτέλεσμα στη μνήμη	Επεξήγηση εντολής
		Θέτει στη μεταβλητή βήματα την τιμή 20 βήματα ← 20
		Αυξάνει τη μεταβλητή βήματα κατά 10 βήματα ← βήματα + 10
		Αυξάνει τη μεταβλητή βήματα κατά 10 βήματα ← βήματα + 10
		Αυξάνει τη μεταβλητή βήματα κατά 10 βήματα ← βήματα + 10

Από τα παραπάνω φαίνεται ότι οι δύο τελευταίες εντολές είναι ισοδύναμες, δηλαδή εκτελούν την ίδια ακριβώς λειτουργία, αυξάνουν τη μεταβλητή κατά 10.

Εκτός από την πρόσθεση, υπάρχουν και άλλοι τελεστές που μπορούμε να χρησιμοποιήσουμε, αν χρειαστεί να κάνουμε αριθμητικούς υπολογισμούς. Επίσης, μπορούν να υπολογιστούν πιο σύνθετες αριθμητικές παραστάσεις, όπως για παράδειγμα:

$$2 \cdot 6 + \frac{10}{3} - (9 + 90)/11$$





Πρόσθεση

Αφαίρεση

Πολλαπλασιασμός

Διαίρεση

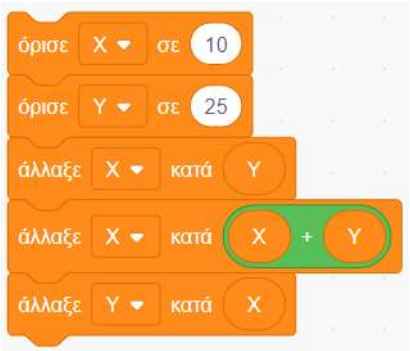
Υπόλοιπο Ακέραιας Διαίρεσης

Στρογγυλοποίηση

Δραστηριότητα 4

Ποιες είναι οι τελικές τιμές των μεταβλητών μετά την εκτέλεση των παρακάτω προγραμμάτων;

Α

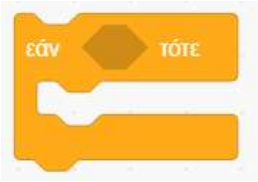
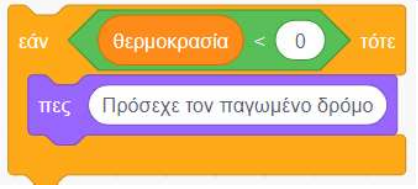
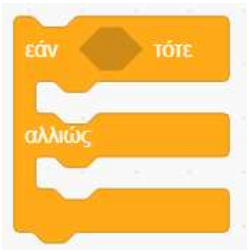


Β

Γ

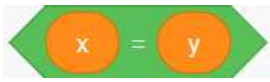
9.7 Η ώρα των αποφάσεων

Πολλές φορές θέλουμε να ελέγξουμε διάφορες περιπτώσεις και να εκτελέσουμε συγκεκριμένες ομάδες εντολών ανάλογα με την τιμή κάποιας συνθήκης. Για αυτό χρησιμοποιούμε τις εντολές διακλάδωσης /επιλογής. Στο διπλανό παράδειγμα, μια μπάλα ποδοσφαίρου κινείται συνεχώς στο σκηνικό. Όταν αγγίζει το όριο της οθόνης, τοποθετείται αμέσως σε μια τυχαία θέση στο σκηνικό και συνεχίζει να κινείται μέχρι να φτάσει στα όρια της οθόνης κ.ο.κ. Η εντολή <πήγαινε σε τυχαία θέση> εκτελείται μόνο όταν η μπάλα αγγίξει κάποιο από τα όρια της οθόνης. Δηλαδή, κάθε φορά ελέγχεται αν ισχύει η έκφραση <αγγίζει <όριο>.

Περιγραφή	Εντολή	Παράδειγμα
Όταν έχουμε μόνο μια περίπτωση. Οι εντολές μέσα στο τότε θα εκτελεστούν μόνο αν η συνθήκη ισχύει.		
Όταν έχουμε δύο περιπτώσεις. Αν η συνθήκη ισχύει, εκτελούνται οι εντολές μέσα στο τότε. Αν δεν ισχύει, εκτελούνται οι εντολές που αντιστοιχούν στο αλλιώς.		

Όταν έχουμε περισσότερες από μία περιπτώσεις, χρειάζεται να χρησιμοποιήσουμε εμφωλευμένες ή ένθετες (nested) εντολές επιλογής, όπως φαίνεται στο διπλανό παράδειγμα, όπου παίρνουμε περιπτώσεις για τη θερμοκρασία. Αν η θερμοκρασία είναι θετικός αριθμός, εμφανίζεται αντίστοιχο μήνυμα. Αν όμως δεν είναι, δηλαδή η συνθήκη θερμοκρασία > 0 δεν ισχύει, τότε υπάρχουν δύο περιπτώσεις, είτε να είναι μηδέν, είτε να είναι αρνητική. Γι' αυτό το λόγο, πρέπει πάλι να χρησιμοποιήσουμε και δεύτερο έλεγχο για να διακρίνουμε ανάμεσα στις δύο αυτές περιπτώσεις. Αν δεν ισχύει ούτε η συνθήκη θερμοκρασία < 0, τότε η μόνη περίπτωση που μένει είναι η μηδενική θερμοκρασία που αντιστοιχεί στο τελευταίο αλλιώς. Άρα, στο αλλιώς φτάνουμε μόνο όταν δεν ισχύει καμία από τις συνθήκες που έχουν προηγηθεί.

Μια λογική συνθήκη μπορεί να ισχύει ή να μην ισχύει. Όταν ισχύει λέμε ότι έχει τιμή **Αληθής** ενώ όταν δεν ισχύει έχει την τιμή **Ψευδής**. Δίπλα φαίνονται οι τιμές κάποιων συνθηκών, στις οποίες εμπλέκονται οι μεταβλητές x και y με περιεχόμενο:

Συνθήκη	Αποτέλεσμα
	28>6, **Αληθής**
	28<6, **Ψευδής**
	28=6, **Ψευδής**

Μπορούμε να χρησιμοποιήσουμε και σύνθετες συνθήκες, αν χρειάζεται. Για παράδειγμα, αν θέλουμε η θερμοκρασία να είναι εντός συγκεκριμένων ορίων χρειάζεται η σύζευξη δύο συνθηκών με χρήση του λογικού τελεστή **και** όπως φαίνεται παρακάτω:

Μπορούμε, επίσης, να σχηματίσουμε την αντίθετη συνθήκη, δηλαδή να είναι η θερμοκρασία εκτός των παραπάνω ορίων, άρα είτε μικρότερη ή ίση από 20°C είτε μεγαλύτερη ή ίση από 30°C. Για αυτό μπορούμε να χρησιμοποιήσουμε τον λογικό τελεστή της διάζευξης (**ή**) ή τον λογικό τελεστή της άρνησης (**όχι**).

Είναι φανερό ότι η χρήση του τελεστή **όχι** όταν θέλουμε να ελέγξουμε την αντίθετη μιας λογικής συνθήκης, μπορεί να μας γλιτώσει από πολύ δουλειά και να μειώσει το μέγεθος του κώδικά μας.

Δραστηριότητα 7

Να γράψετε ένα πρόγραμμα το οποίο θα ζητάει από τον χρήστη έναν αριθμό, θα ελέγχει αν αυτός ο αριθμός είναι άρτιος ή περιττός και θα εμφανίζει αντίστοιχο μήνυμα.
Υπόδειξη: Να χρησιμοποιήσετε τον τελεστή υπολογισμού του υπολοίπου της ακέραιας διαίρεσης.

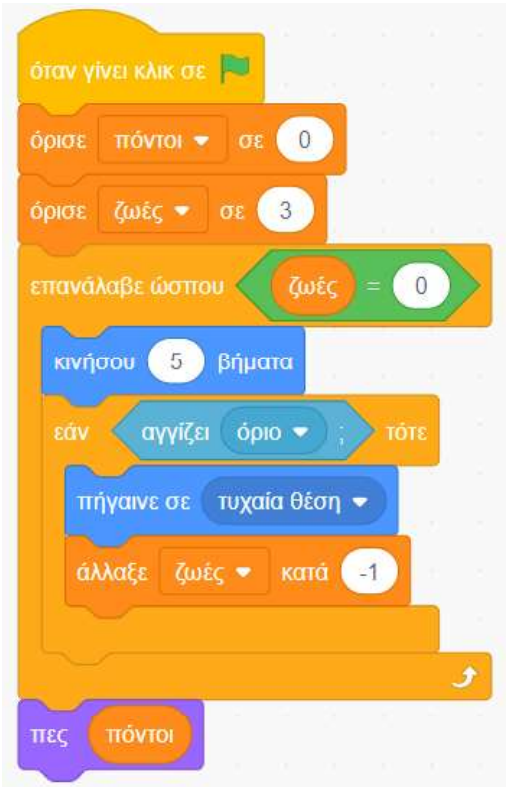
Δραστηριότητα 8

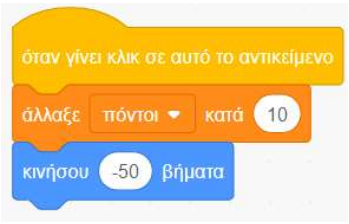
Να γράψετε ένα πρόγραμμα το οποίο θα κάνει έναν συγκεκριμένο ήχο κάθε φορά που ο χρήστης θα πατάει το πλήκτρο space. Όταν ο χρήστης πατήσει το space 10 φορές το πρόγραμμα να εμφανίζει ένα μήνυμα και να εξαφανίζει τον χαρακτήρα – πρωταγωνιστή του προγράμματος.
Υπόδειξη: Μπορείτε να χρησιμοποιήσετε τις παρακάτω εντολές:

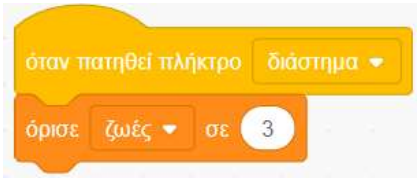
9.8 Χειρισμός γεγονότων

Μια σημαντική ομάδα εντολών, που μπορεί να αξιοποιηθεί στον σχεδιασμό παιχνιδιών, είναι τα συμβάντα (events) και οι αισθητήρες (sensing).

Θα μπορούσαμε να επεκτείνουμε το πρόγραμμα με τη μπάλα ποδοσφαίρου που κινείται συνεχώς στη σκηνή με την εξής λειτουργικότητα: Να προσπαθεί ο χρήστης-παίκτης να κάνει κλικ με το ποντίκι στην μπάλα που κινείται. Κάθε φορά που τα καταφέρνει θα παίρνει 10 πόντους, αλλά κάθε φορά που δε θα προλαβαίνει και η μπάλα θα φτάνει στο όριο της σκηνής θα χάνει μια ζωή. Για αυτό θα χρειαστούμε δύο μεταβλητές, τις ζωές και τους πόντους. Ο παίκτης ξεκινάει αρχικά με 3 ζωές και 0 πόντους. Σε περίπτωση που χάσει και την τελευταία ζωή του το πρόγραμμα τερματίζει και του εμφανίζει τους βαθμούς που συγκέντρωσε. Για αυτό χρησιμοποιούμε τη εντολή επανάληψης **Επανάλαβε ώσπου <ζωές=0>** η οποία ελέγχει σε κάθε επανάληψη αν η μεταβλητή ζωές έχει πάρει την τιμή 0.

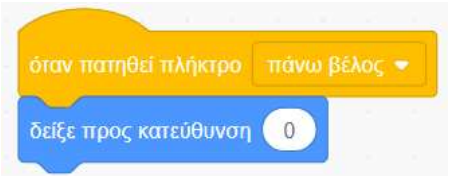


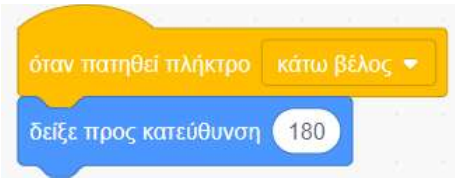


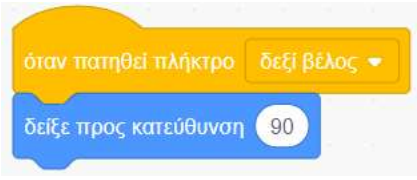


Παραπάνω προσθέσαμε και έναν ειδικό κωδικό (cheat code), ώστε όταν ο χρήστης χάνει, να μπορεί να πάρει τρεις ζωές με ένα πάτημα του πλήκτρου διάστημα (space).

Μπορούμε να γράψουμε κώδικα για τη διαχείριση και άλλων γεγονότων, όπως το πάτημα ενός πλήκτρου. Για παράδειγμα, αντί η μπάλα να κινείται τυχαία, θα μπορούσε να τη μετακινεί ο παίκτης με τα βελάκια και ένας άλλος παίκτης να χειρίζεται το ποντίκι.







Μπορείτε να δώσετε εσείς τον αντίστοιχο κώδικα για τον χειρισμό του πλήκτρου που μετακινεί τον χαρακτήρα αριστερά (αριστερό βέλος);

Προγραμματισμός οδηγούμενος από γεγονότα

Σήμερα, οι σύγχρονες εφαρμογές δεν είναι προγράμματα τα οποία ακολουθούν μια σειριακή εκτέλεση των εντολών. Αντιθέτως, αποτελούνται από πολλά μικρά προγράμματα, τα οποία είναι σχεδιασμένα να ανταποκρίνονται/χειρίζονται συγκεκριμένα γεγονότα, όπως το πάτημα ενός πλήκτρου ή του ποντικιού ή ένα μήνυμα από ένα άλλο πρόγραμμα μέσω Διαδικτύου.