**3o Φύλλο Εργασίας**

**Τίτλος: Ping Pong**

**Διάρκεια: 1 ώρα**

**Ονοματεπώνυμα Ομάδας:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Στόχος**

* Με τη δραστηριότητα αυτή, θα μάθετε τι είναι τα υποπρογράμματα και πως λειτουργούν
* Θα θυμηθείτε τις εντολές ελέγχου (**if ….then**)
* Θα θυμηθείτε τις μεταβλητές και θα μάθετε τον τρόπο με τον οποίο ανακτάτε, μεταβάλετε τις τιμές τους

**Σκοπός Εφαρμογής**

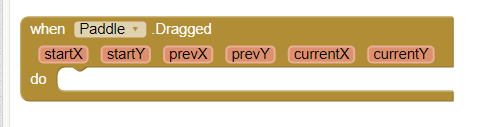
Θα δημιουργήσετε μια ένα παιχνίδι, που υλοποιεί την εφαρμογή **Ping Pong.** Το παιχνίδι αποτελείται από μια μπάλα που κινείται συνεχώς και αναπήδα στα τοιχώματα. Ο παίκτης χειρίζεται μία σανίδα, την οποία μετακινεί αριστερά και δεξιά με το δάκτυλο. Ο παίκτης μετακινώντας τη σανίδα, προσπαθεί να χτυπήσει την μπάλα και να την εμποδίσει να αγγίξει την κάτω πλευρά της οθόνης (στην περίπτωση αυτή το παιχνίδι τερματίζει και ο παίκτης χάνει)

**Βήμα 1ο**

**Άνοιγμα Project**

Ξεκινώντας, μεταφορτώνουμε από τον κοινόχρηστο δίσκο το project με όνομα **ping\_pong\_original.** Αρχικά η εφαρμογή περιέχει μόνο τις εντολές που μετακινούν την μπάλα σε τυχαίες θέσεις. Ακόμη υπάρχουν εντολές που αλλάζουν την κατεύθυνση την μπάλας, όταν χτυπήσει οποιοδήποτε άλλο αντικείμενο της οθόνης. Σε αυτό το βήμα θα ορίσουμε πως θα γίνεται η μετακίνηση της σανίδας, με το σύρσιμο της από την πλευρά του χρήστη. Το γεγονός που ανιχνεύει το σύρσιμο της σανίδας **(Paddle)** είναι αυτό που φαίνεται στην εικόνα.

* Οι μεταβλητές **startX** και **startY** δείχνουν τις συντεταγμένες του σημείου που αγγίξαμε, όταν ακουμπήσαμε την πρώτη φορά την οθόνη



* Οι μεταβλητές **currentX** και **currentY** δείχνουν τις τρέχουσες συντεταγμένες του σημείου που αγγίζουμε.

Προσέξτε, όταν συμβεί το γεγονός, οι προηγούμενες συντεταγμένες (**PrevY, PrevX**) είναι ίδιες με τις αρχικές (**startX, startY).** Αν δεν συμπεριλάβουμεκαμιά εντολή μετακίνησης του αντικειμένου (αλλάζοντας τις θέσεις x και y), δεν πρόκειται να συμβεί τίποτα.

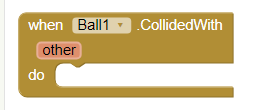
Επομένως θα χρειαστείτε προσδιορίσετε μέσα στο γεγονός



τις σωστές τιμές για το Χ και **Υ** του **paddle**, ώστε η σανίδα να κινείται μόνο αριστερά, δεξιά αλλά όχι πάνω και κάτω. (μεταβάλλονται οι τιμές για τον άξονα Χ του αντικειμένου αλλά όχι για το **Υ**)

**Βήμα 2ο**

**Προσδιορισμός ενεργειών, όταν ακουμπήσει η μπάλα τη σανίδα**

Η σύγκρουση της μπάλας με τη σανίδα (γεγονός-**CollidedWith**) 

θα πρέπει να πυροδοτεί την εκτέλεση διάφορων ενεργειών, όπως οι ακόλουθες:

* Nα γίνεται ανακατεύθυνση της μπάλας. . Η ιδιότητα **Ηeading** της μπάλας καθορίζει την κατεύθυνση, προς την οποία κινείται η μπάλα. Το **Ηeading** μπορεί να πάρει τις εξής τιμές σε εύρος από 0 μέχρι 360:

0 : δεξιά

90 : πάνω

180 : αριστερά

270 : κάτω

Για να είναι ρεαλιστική η μετακίνηση της μπάλας, θα πρέπει να κινείται σε αντιδιαμετρικά αντίθετη κατεύθυνση ως προς τον οριζόντιο άξονα. Δηλαδή όπως φαίνεται στο σχήμα

135ο

45ο

Για αυτό το λόγο χρησιμοποιήθηκε η εντολή



* Να αυξάνεται το σκορ, δηλαδή θα αλλάζει η μεταβλητή **score**. Θα πρέπει να φροντίσετε, ώστε η τιμή της μεταβλητής να τροποποιείται με επιλογή της κατάλληλης εντολής (μέθοδος) από την ομάδα εντολών **variable**s και να αυξάνεται με εντολή από την ομάδα εντολών **math.**
* Να αναγράφεται το σκορ στην ετικέτα **scoreLabel**

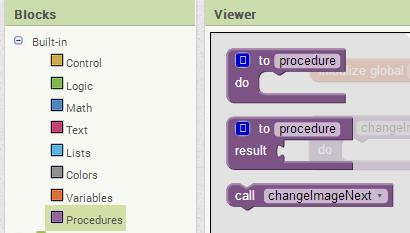
H τελευταία ενέργεια, δηλαδή η αναγραφή του σκορ γίνεται και όταν πατιέται το κουμπί της επανεκκίνησης **startButton**, το οποίο μηδενίζει το σκορ. Όπως είναι λογικό κάθε φορά που αλλάζει το σκορ, θα πρέπει αναγράφεται η νέα τιμή του στην ετικέτα **scoreLabel.**

Η ενέργεια αυτή επαναλαμβάνεται σε δυο διαφορετικά γεγονότα. Δεν είναι όμως λογικό να επαναλαμβάνουμε τις ίδιες εντολές ξανά και ξανά. Επίσης, εάν φορτώνουμε το πρόγραμμα με πολλές εντολές, γίνεται δυσνόητο. Φανταστείτε εάν αυτό συμβαίνει όχι μόνο δυο αλλά 10ή 20 φορές!

Για αυτό το λόγο υπάρχουν τα υποπρογράμματα, Στο App Inventor μπορούμε να δημιουργήσουμε ένα τμήμα προγράμματος που επιτελεί λίγες και αυτόνομες λειτουργίες, το οποίο ονομάζεται διαδικασία. Μέσα στη διαδικασία ενσωματώνουμε τις εντολές που εκτελούνται συχνά. Μία διαδικασία είναι επομένως μία σειρά εντολών, που μπορούμε να χρησιμοποιήσουμε σε οποιοδήποτε σημείο του προγράμματος μας. Η χρήση υποπρογραμμάτων έχει σημαντικά πλεονεκτήματα

* Γρήγορη ανάπτυξη προγραμμάτων αφού δεν επαναλαμβάνουμε τις ίδιες εντολές ξανά και ξανά
* Πιο κατανοητός κώδικας και πιο οργανωμένος
* Αποκτούμε δεξιότητες αφαίρεσης αφού μαθαίνουμε να ξεχωρίζουμε μέσα από την πολυπλοκότητα των προβλημάτων τα σημαντικά μέρη τους. Ταυτόχρονα καλλιεργούμε δεξιότητες σύνθεσης, επειδή καθώς προχωράμε στην ανάπτυξη ξεχωριστών υποπρογραμμάτων, θα απαιτηθεί να τα συνδέσουμε κατάλληλα μεταξύ τους για να φτιάξουμε το ενιαίο πρόγραμμα μας.

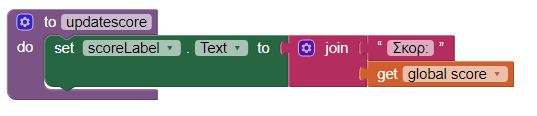
Η ενημέρωση λοιπόν, του **scoreLabel** θα γίνεται μέσα από ένα υποπρόγραμμα. Η δημιουργία μιας νέας διαδικασίας, πραγματοποιείται από την ομάδα εντολών **Procedures** (Διαδικασίες)



όπου και επιλέγουμε την εντολή **to [**procedure**] do** που φαίνεται παρακάτω.



Στη συνέχεια αλλάζουμε το όνομα της διαδικασίας από **procedure** σε **updatescore**. Μέσα στο σώμα της θα προσθέσουμε τις κατάλληλες εντολές, ώστε να αλλάζει το περιεχόμενο του **scoreLabel** (μέθοδος) και να αναγράφει το σκορ (μέθοδος από ομάδα εντολών **Variables**). Το τελικό αποτέλεσμα φαίνεται στην παρακάτω εικόνα.



Για να χρησιμοποιήσουμε τη διαδικασία σε ένα ή περισσότερα σημεία της εφαρμογής, θα χρειαστεί να την καλέσουμε. Η κλήση μιας διαδικασίας γίνεται αν βάλουμε σε οποιοδήποτε σημείο την εντολή **call** από την ομάδα εντολών **Procedures.**

Φροντίστε να κάνετε κλήση της διαδικασίας όταν

* η μπάλα συγκρούεται με την σανίδα
* πατιέται το κουμπί **StartButton,** που μηδενιζει το σκορ**.**

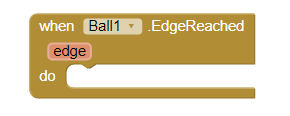
**Βήμα 4ο**

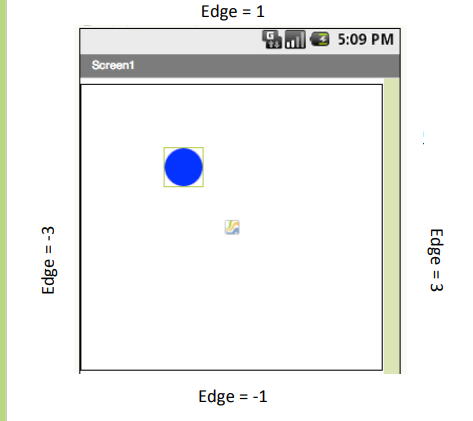
**Προσδιορισμός ενεργειών όταν η μπάλα ακουμπάει το κάτω μέρος οθόνης**

Η πρόσκρουση της μπάλας στα όρια της οθόνης (γεγονός – **Ball1.EdgeReached)**, προκαλεί την αναπήδηση της μπάλας μέχρι στιγμής. Όμως θα πρέπει να τροποποιηθεί ο κώδικας σε αυτό το γεγονός, ώστε

* Εάν η μπάλα αγγίξει το κάτω μέρος της οθόνης, το παιχνίδι να τερματίζεται. Συνεπώς θα γίνεται ο παραπάνω έλεγχος και αν ισχύει (η μπάλα ακουμπάει το κάτω μέρος οθόνης) θα παύει η κίνηση της μπάλας (μέθοδος από ομάδα εντολών **Ball**) και θα αναγράφεται στο **ScoreLabe**l (μέθοδος) η φράση **GAME OVER.**

Το γεγονός **Ball1.EdgeReached** εντοπίζει πότε η μπάλα αγγίζει τα όρια του καμβά (οθόνη) και μας δίνει μία τιμή για τη μεταβλητή **edge** ανάλογα ποιο σημείο της οθόνης έχει ακουμπήσει η μπάλα.





(κάτω-1, πάνω=1, αριστερα=-3, δεξιά=3)

* Διαφορετικά, όταν η μπάλα αγγίξει οποιοδήποτε άλλο άκρο της οθόνης, η μπάλα θα αναπηδά. Αυτό πραγματοποιείται με την εντολή . Η εντολή αυτή προκαλεί την αναπήδηση της μπάλας από ένα συγκεκριμένο άκρο της οθόνης. Συνήθως το άκρο αυτό, προσδιορίζεται από τη μεταβλητή-παράμετρο **edge** του γεγονότος **Ball1.EdgeReached.**

**ΕΡΓΑΣΙΑ ΓΙΑ ΣΠΙΤΙ**

Θα τροποποιήσετε την εφαρμογή, ώστε κάθε φορά που ανεβαίνει το σκορ κατά μια δεκάδα, θα αυξάνει η ταχύτητα κίνησης της μπάλας κατά 5 μονάδες. Θα χρειαστείτε μια μεταβλητή **speed**, η οποία θα καθορίζει την εκάστοτε ταχύτητα της μπάλας. Επίσης θα χρειαστείτε από την ομάδα εντολών **Math**, την εντολή  για να ελέγχετε πότε το σκορ ανεβαίνει δεκάδες.

Η εντολή **Modulo** δέχεται δυο αριθμούς και μας επιστρέφει το ακέραιο υπόλοιπο της διαίρεσης του πρώτου αριθμού με τον δεύτερο. **Υπόδειξη**: Σκεφτείτε οι αριθμοί που είναι πολλαπλάσια του 10, τι υπόλοιπο αφήνουν όταν διαιρεθούν με το 10.

