**App Inventor: Εφαρμογές για Android**

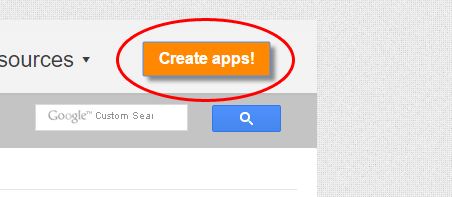
Το App Inventor είναι ένα χρήσιμο εργαλείο για κάθε αρχάριο προγραμματιστή που θέλει να ασχοληθεί με mobile apps, με περιβάλλον τύπου Scratch.

**Τι χρειάζεται για να ξεκινήσουμε;**

* Η λειτουργία του App Inventor είναι κυρίως διαδικτυακή
* δεν απαιτείται κάποιο συγκεκριμένο λογισμικό (Windows, Linux κτλ.)
* Δεν απαιτείται υπολογιστής με ιδιαίτερες δυνατότητες.
* Δεν απαιτείται η γνώση κάποιας γλώσσας προγραμματισμού εκ μέρους του χρήστη.
* Χρειαζόμαστε [ένα λογαριασμός Gmail](https://www.pcsteps.gr/22989-%CE%B4%CE%B7%CE%BC%CE%B9%CE%BF%CF%85%CF%81%CE%B3%CE%AF%CE%B1-gmail-google-%CE%BB%CE%BF%CE%B3%CE%B1%CF%81%CE%B9%CE%B1%CF%83%CE%BC%CE%BF%CF%8D/)

**Περιβάλλον Σχεδίασης και Προγραμματισμού**

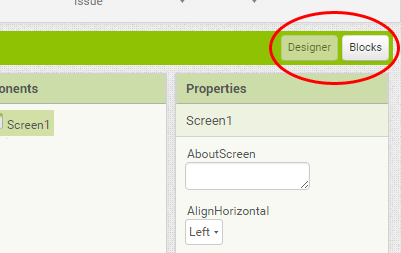
Για να ξεκινήσουμε λοιπόν την πρώτη από τις εφαρμογές Android, αρχικά [μπαίνουμε στο site](http://appinventor.mit.edu/explore/ai2/windows.html) του Αpp Inventor (<http://appinventor.mit.edu>) και κάνουμε κλικ στο **Create Apps**.



Αφού συνδεθούμε με τον λογαριασμό μας στο Gmail πατάμε το **Start new project**.

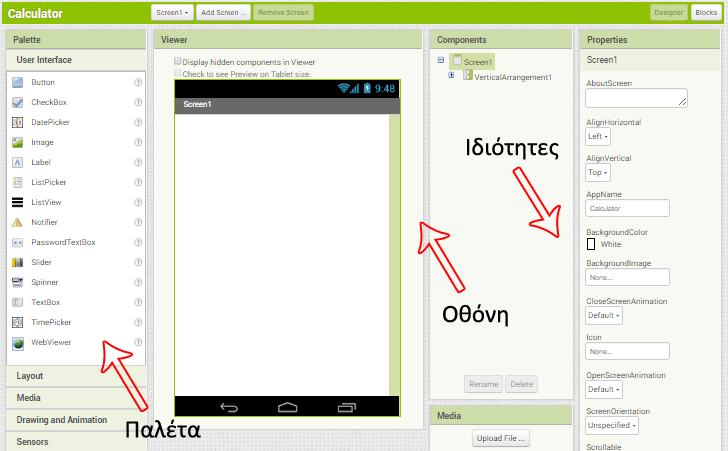
Στην συνέχεια, δίνουμε ένα οποιοδήποτε όνομα. Στο παράδειγμα θα το ονομάσουμε "**Calculator**".

Αφού ορίσουμε την ονομασία, ανοίγει αυτόματα το Project. Βλέπουμε ότι το περιβάλλον εργασίας του App Inventor αποτελείται από δύο μέρη. Την Σχεδίαση (**Designer**) και τα **Blocks**, που είναι το κομμάτι του προγραμματισμού.



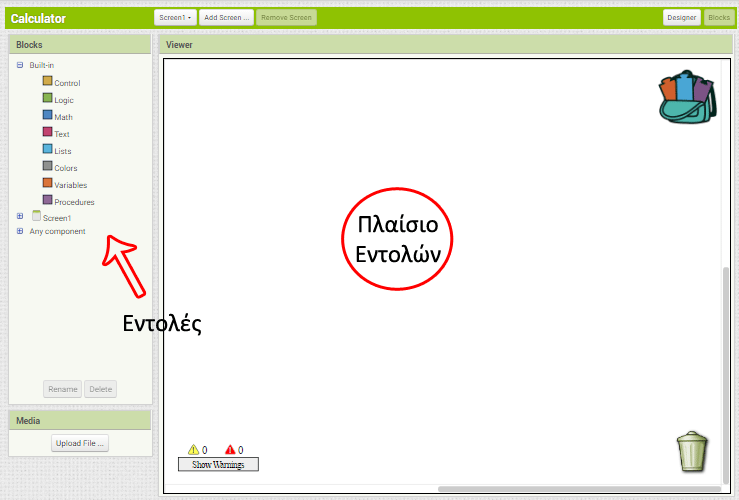
Συγκεκριμένα, η σχεδίαση αποτελείται από την **παλέτα**, από την **οθόνη της εφαρμογής**, από την **λίστα των αντικειμένων** που έχουμε τοποθετήσει μέσα στο πλαίσιο της οθόνης, και από της ιδιότητες αυτής.

Τα αντικείμενα της παλέτας τα βάζουμε μέσα στο πλαίσιο της οθόνης με την κίνηση **Drag And Drop** (Μεταφορά και Απόθεση).



Όσον αφορά τα blocks, δηλαδή τον προγραμματισμό σε παζλ, τα "κομμάτια" που υπάρχουν στην αριστερή άκρη της οθόνης αποτελούν **απλές εντολές στα αγγλικά**.

Οι εντολές αυτές τοποθετούνται στον Viewer, δηλαδή στο πλαίσιο τοποθέτησης των εντολών, με την διαδικασία drag and drop όπως και τα αντικείμενα της σχεδίασης. Επίσης στο πλαίσιο των εντολών υπάρχει ένα **σακίδιο που βοηθάει στην εύκολη αναπαραγωγή των ιδίων εντολών** και ένας κάδος ο οποίος τις καταργεί.



**Δημιουργία αλγορίθμου**

Η επιτυχία μίας εφαρμογής βασίζεται σε αυτό που ονομάζουμε αλγόριθμο. Ο αλγόριθμος είναι μια **πλήρης και περιεκτική περιγραφή, βήμα προς βήμα**, της διαδικασίας που θα ακολουθήσει το πρόγραμμα για να κάνει τις λειτουργίες που θέλουμε.

Για παράδειγμα, αν θέλουμε να φτιάξουμε ένα πολύ απλό κομπιουτεράκι, που να κάνει πράξεις ανάμεσα σε δύο αριθμούς και να εμφανίζει το αποτέλεσμα, ο αλγόριθμος χονδρικά είναι:

1. Διάβασε τον πρώτο αριθμό
2. Διάβασε το σύμβολο της πράξης
3. Διάβασε τον δεύτερο αριθμό
4. Ανάλογα με το σύμβολο, κάνε την κατάλληλη πράξη
5. Εμφάνισε το αποτέλεσμα

Είναι σημαντικό να κατανοήσουμε πως **ο αλγόριθμος δεν είναι από μόνος του γλώσσα προγραμματισμού**, δεν περιλαμβάνει συγκεκριμένες εντολές ή σύνταξη. Είναι ένα σχεδιάγραμμα, το οποίο, αν έχει γίνει σωστά, μπορούμε στη συνέχεια να αναπτύξουμε με οποιαδήποτε μέθοδο προγραμματισμού.

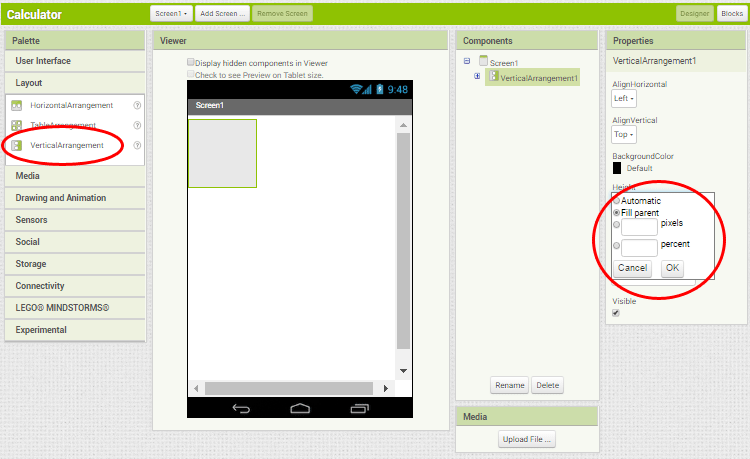
**Δημιουργία Εφαρμογής**

Εφόσον έχουμε τον αλγόριθμο, στο επόμενο βήμα θα σχεδιάσουμε την εμφάνιση της εφαρμογής μας, που είναι ιδιαίτερα εύκολο με το App Inventor.

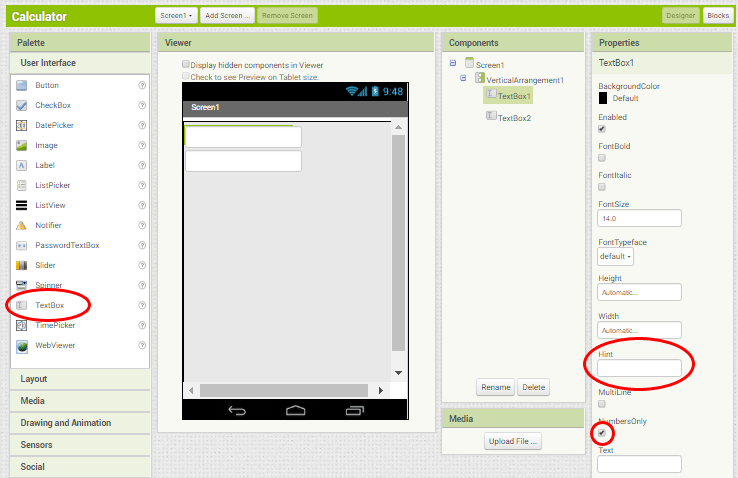
Δίνουμε στην οθόνη (**screen**) το όνομα “**Αριθμομηχανή**”.

Αρχικά τοποθετούμε στην εικόνα ένα VerticalArragment από την καρτέλα Layout και ορίζουμε το ύψος (**Height**) και το πλάτος (**Weight**) του ως **Fill Parent** (για να πάρει το μέγεθος ολόκληρης της οθόνης).

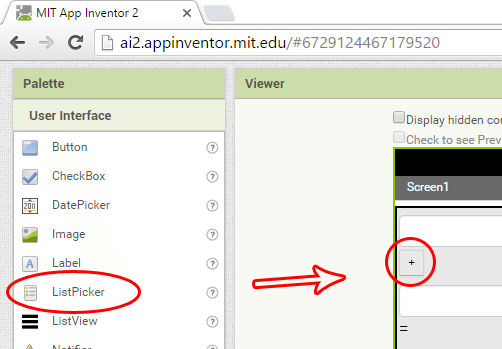
Έτσι δημιουργούμε το επίπεδο πάνω στο οποίο θα σχεδιάσουμε την εφαρμογή μας, βάζοντας τα αντικείμενα μας σε σωστή στοίχιση.



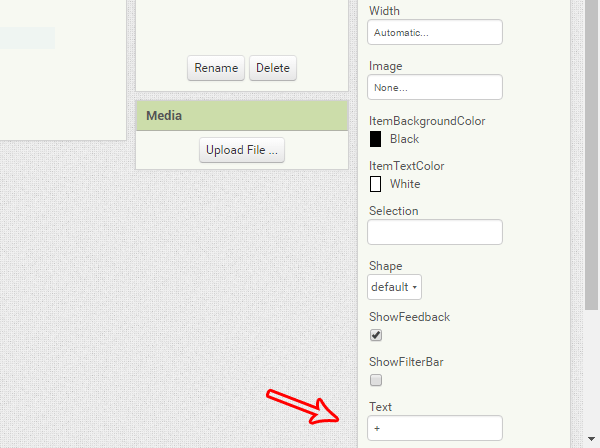
Στην συνέχεια βάζουμε μέσα σε αυτό το επίπεδο δύο **TextBox** (1,2) (**Πλαίσιο κειμένου**) στα οποία σβήνουμε το κείμενο από το πεδίο Hint και **ενεργοποιούμε την αποδοχή μόνο αριθμητικών τιμών** (Number Only). Αυτά τα TextBox θα είναι τα πεδία εισαγωγής των δύο αριθμών που θα υπολογίζει το κομπιουτεράκι, βάσει αλγορίθμου.



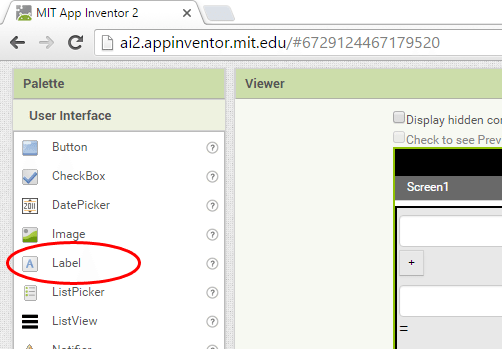
Έπειτα σύρουμε ένα "**ListPicker**", που είναι ένα **κουμπί πολλαπλής επιλογής** από μια λίστα που θα ορίσουμε. Στην προκειμένη περίπτωση, θα το χρησιμοποιήσουμε για τις πράξεις που θα υποστηρίζει το κομπιουτεράκι.



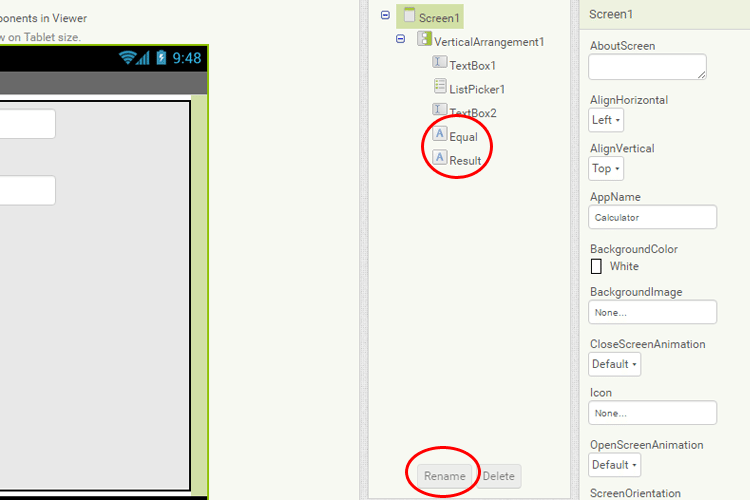
Στο πεδίο text του ListPicker **σβήνουμε το κείμενο και γράφουμε απλά "+"**. Θα δούμε ποιο κάτω τον λόγο.



Εισάγουμε επίσης δύο Label, τα οποία είναι απλό κείμενο για την εφαρμογή μας.



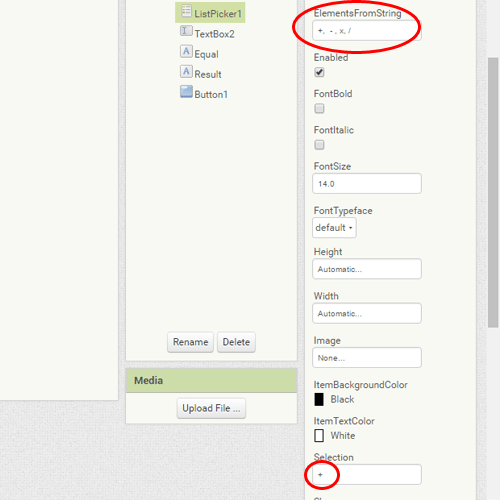
Tα ονομάζουμε **Equal** και **Result** αλλάζοντας αντίστοιχα και το κείμενο τους σε "**=**" και "**Αποτέλεσμα**".



Στο επόμενο βήμα προσθέτουμε το **Button1(Κουμπί)** στο κείμενο του οποίου γράφουμε "**Υπολόγισε**".

Τέλος, το μόνο που μας έχει μείνει όσον αφορά την σχεδίαση είναι να ορίσουμε τις άλλες πράξεις που μπορεί να κάνει το πρόγραμμα. Γι’ αυτό αλλάζουμε τις ιδιότητες του ListPicker. Συγκεκριμένα στο κενό **ElementsFromString γράφουμε "+, -, \*, /"** ενώ στο κενό Selection γράφουμε "+".

Σκοπός αυτής της ρύθμισης είναι η εφαρμογή να έχει ως προεπιλεγμένη πράξη την πρόσθεση και πατώντας πάνω στο ListPicker να επιλέγουμε εμείς την πράξη που θέλουμε.



Όπως είδαμε ποιο πάνω, ο κώδικας στο App Inventor αποτελεί ένα είδος παζλ. Η βασική ιδέα για την δημιουργία της αριθμομηχανής είναι ότι **το πρόγραμμα κάνει την ίδια δουλειά κάθε φορά, και το μόνο που αλλάζει είναι η πράξη που έχουμε επιλέξει**.

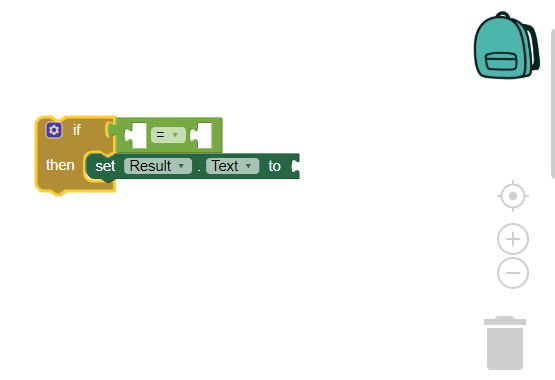
Για να περάσουμε αυτή την λογική στις εφαρμογές για Android, χρησιμοποιούμε τις κατάλληλες εντολές από δίπλα.

Είναι σημαντικό να γνωρίζουμε σε ποια καρτέλα βρίσκεται η κάθε εντολή. Για αυτό τον λόγο θα γράφουμε την καρτέλα σε παρένθεση δίπλα σε κάθε εντολή.

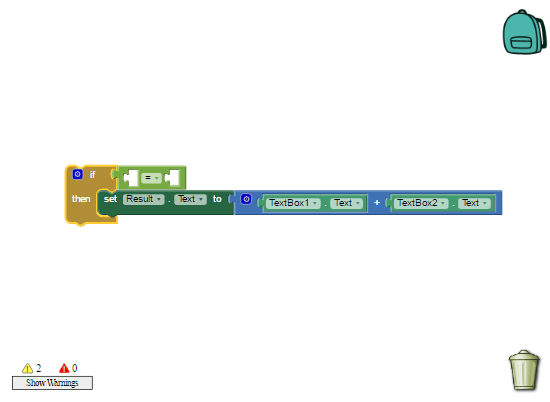
Αρχικά παίρνουμε την εντολή **if/then (Control)**. Η δομή if/then είναι από τις πιο θεμελιώδεις δομές του προγραμματισμού. Στην προκειμένη περίπτωση θα τη χρησιμοποιήσουμε για να πούμε στο κομπιουτεράκι "Αν (If) η πράξη που επιλέγει ο χρήστης είναι η τάδε, τότε (then) κάνε το εξής".



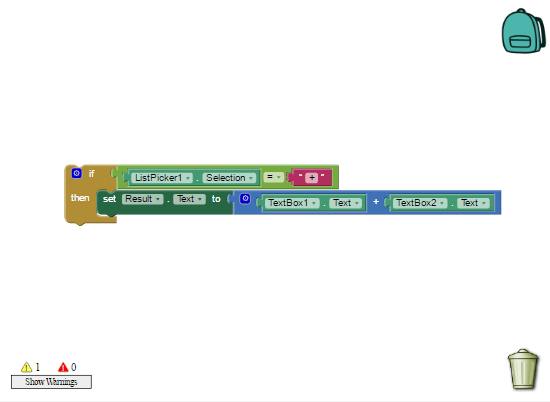
Ενώνουμε το if με την εντολή **"\_=\_" (Logic**), με το οποίο θα ελέγξουμε τις πιθανές πράξεις, και το then με την εντολή **"Set Result.Text to" (Result)**, για να ορίσει το αποτέλεσμα.



Το πρώτο αυτό κομμάτι αφορά την πρόσθεση. Κατά συνέπεια, την εντολή **‘Set Result.Text to’** την ενώνουμε με την εντολή **"\_+\_" (Math)**. Συμπληρώνουμε τα κενά με τα "κομμάτια" **"ΤextBox1.Text" (TextBox1)** και **"TextBox2.Text" (TextBox2)**.

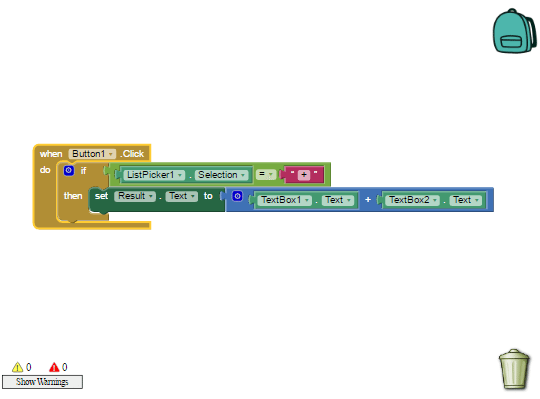


Συμπληρώνουμε επίσης και την ισότητα στο if ως εξής: **“List.Picker1.Selection” = “+” (ListPicker1, Text)**

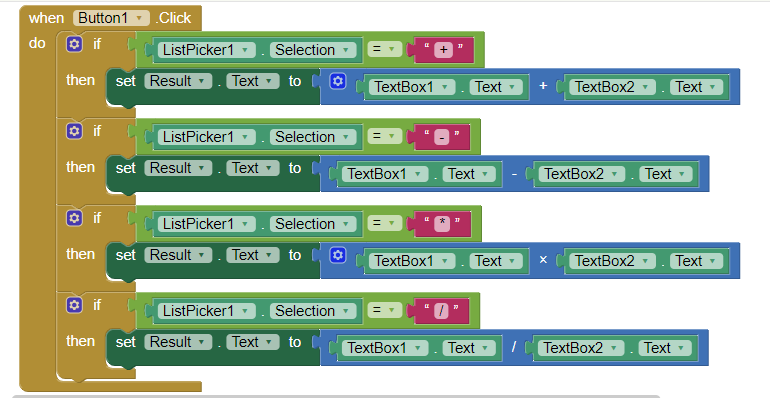


Μέχρι εδώ, έχουμε πει ουσιαστικά στο πρόγραμμα: Αν η επιλογή του ListPicker1 είναι "+", τότε στο πεδίο Result εμφάνισε το άθροισμα του TextBox1 με το TextBox2.

Τέλος, βάζουμε την εντολή If/Then μέσα στην εντολή when Button1. Clicked (Button1), ώστε να εκτελεστεί όταν ο χρήστης πατήσει το κουμπί Button1.



Στην συνέχεια κάνουμε την **αντίστοιχη διαδικασία και για τις άλλες πράξεις**, μέσα στο ίδιο When Button1 .Click. Παρατηρήστε πως σε κάθε μία αλλάζει μόνο το δεύτερο μέρος του If, με το αντίστοιχο σύμβολο, και φυσικά η πράξη ανάμεσα στα TextBox1 και TextBox2.



Με το τελευταίο κομμάτι του κώδικα ρυθμίζουμε και την εμφάνιση του κατάλληλου συμβόλου στο ListPicker1, ανάλογα με την πράξη που έχουμε επιλέξει.

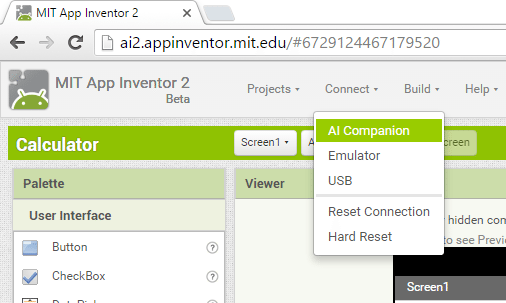
Αυτή είναι λοιπόν η πλήρης εφαρμογή μας. Όταν πατάμε το κουμπί, ανάλογα με το κείμενο του ListPicker κάνει την αντίστοιχη πράξη, και το ListPicker εμφανίζει το σύμβολο της πράξης.

**Τρόποι εκτέλεσης και δοκιμής**

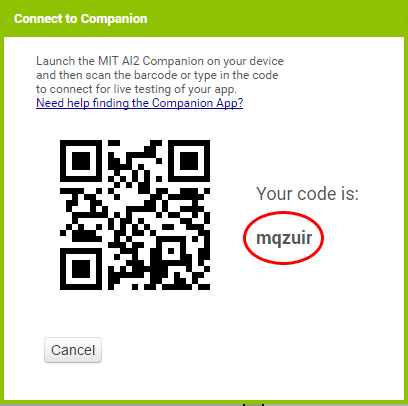
Οι τρόποι για να ελέγξουμε αν λειτουργεί η εφαρμογή που φτιάξαμε με το App Inventor είναι τρεις. Με κανέναν από αυτούς τους τρόπους οι εφαρμογές για Android δεν εγκαθίστανται κάπου. Απλά εκτελούνται δοκιμαστικά.

**Μέσω Wi-Fi**

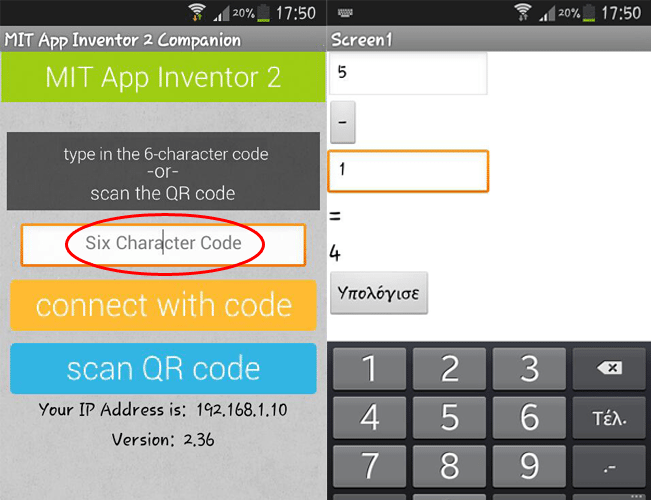
Μάλλον πρόκειται για τον ποιο έμπιστο τρόπο. Αρχικά κατεβάζουμε [την εφαρμογή All Companion](https://play.google.com/store/apps/details?id=edu.mit.appinventor.aicompanion3) στο Android κινητό μας. Μετά επιλέγουμε την σύνδεση All Companion από το App Inventor.



Τέλος, αντιγράφουμε τον κωδικό που μας δίνεται στην συσκευή μας...



...και μετά από λίγο η εφαρμογή μας θα είναι έτοιμη.

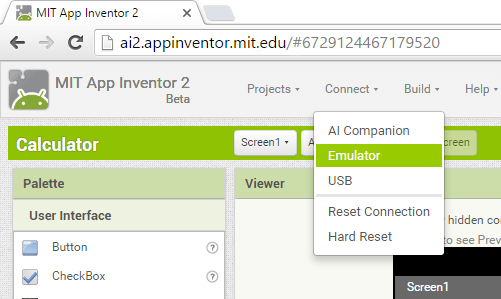


**Με προσομοιωτή**

Ο δρόμος αυτός μπορεί να φαίνεται βολικός γιατί δεν απαιτεί σύνδεση με κινητό. Η δοκιμή όμως είναι αρκετά επίπονη, γιατί ο Emulator (προσομοιωτής) είναι αρκετά αργός και δεν το συνιστούμε.

Για να το επιλέξουμε πρέπει πρώτα να τρέξουμε το aiStarter που βρίσκεται στην επιφάνεια εργασίας μας από την εγκατάσταση που κάναμε στην αρχή.

Μετά να επιλέξουμε την σύνδεση με Emulator από το περιβάλλον του App Inventor.

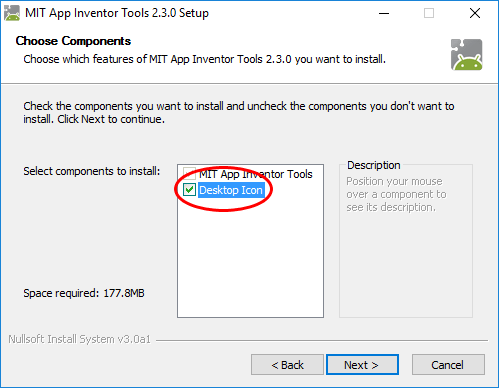


**Εγκατάσταση του προσομοιωτή App Inventor**

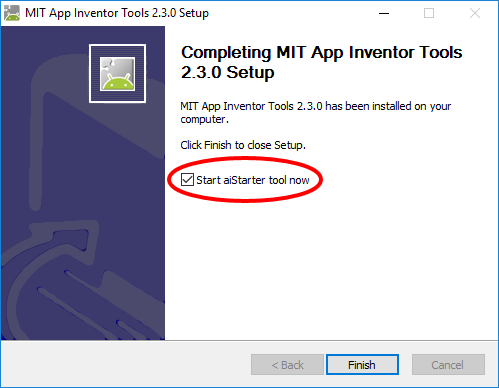
Η εγκατάσταση του App Inventor είναι πρακτικά η εγκατάσταση όλων όσων χρειάζονται για να δοκιμάζουμε τις εφαρμογές για Android που δημιουργούμε και να ελέγχουμε αν λειτουργούν.

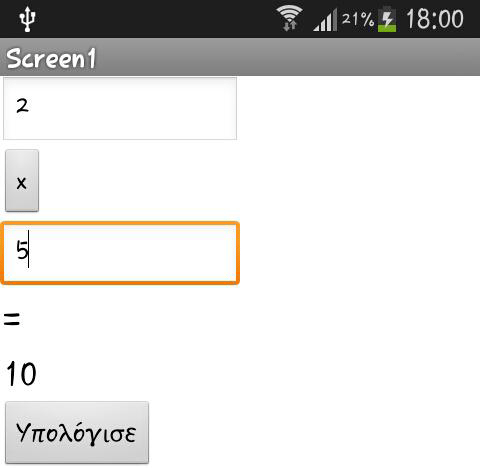
Αρχικά μπαίνουμε [στην επίσημη σελίδα](http://appinventor.mit.edu/explore/ai2/windows.html) του app inventor (<http://ai2.appinventor.mit.edu>) και κατεβάζουμε το setup από το σύνδεσμο "**Download the installer**".

Στην συνέχεια, εκτελούμε το αρχείο αυτό ως διαχειριστές, δεχόμενοι όλες τις προϋποθέσεις. Όταν ερωτηθούμε για το τι θέλουμε να εγκαταστήσει, ενεργοποιούμε το εικονίδιο της επιφάνειας εργασίας.



Με την ολοκλήρωση της διαδικασίας μπορούμε αν θέλουμε να επιλέξουμε να ανοίξει ή όχι η εφαρμογή που έχει εγκατασταθεί.

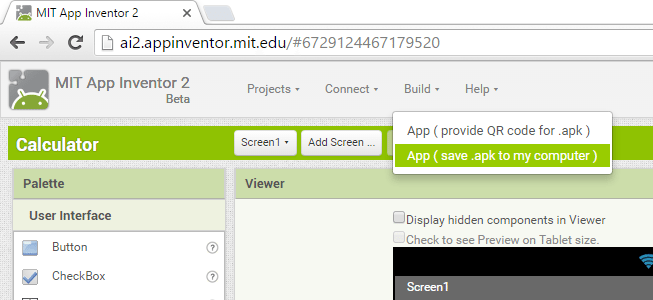




**Πως Εξάγουμε τις Εφαρμογές για Android**

Η εξαγωγή της εφαρμογής έτσι ώστε να την εγκαταστήσουμε σε μια συσκευή Android είναι αρκετά απλή και εύκολη. Προσοχή όμως, για να την τρέξουμε στην συσκευή μας θα πρέπει να έχουμε **ενεργοποιήσει την εγκατάσταση εφαρμογής από άγνωστες πηγές**.

Πηγαίνουμε στο App Inventor και επιλέγουμε Build. Διαλέγουμε την δεύτερη επιλογή καθώς θέλουμε να εξάγουμε την εφαρμογή ως αρχείο.

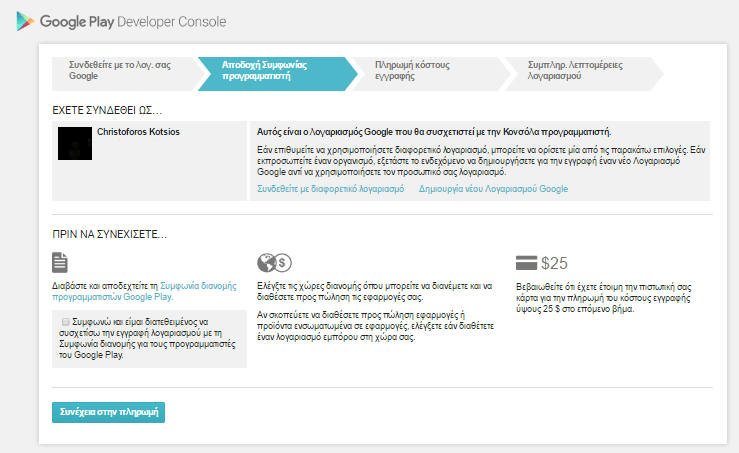


Η διαδικασία θα ολοκληρωθεί με την λήψη του αρχείου .apk στον υπολογιστή μας.

**Υποβολή εφαρμογής στο Google Play**

Πλέον για να εισάγουμε εφαρμογές για Android στο Google Play, θα πρέπει να πληρώσουμε εφάπαξ $25 ως developers. Αυτό ήταν μια κίνηση της Google προκειμένου να ελαττώσει την ύπαρξη εφαρμογών πολύ χαμηλού επιπέδου.

Με εξαίρεση όμως αυτή την επιβάρυνση, η διαδικασία παραμένει η ίδια. Συνδεόμαστε [στη σελίδα υποβολής](https://play.google.com/apps/publish/signup/) με τον λογαριασμό μας στο Gmail και δεχόμαστε όλες τις προϋποθέσεις.

[](https://play.google.com/apps/publish/signup/)

Αφού πληρώσουμε μέσω του Google Wallet μετά συμπληρώνουμε τα απαιτούμενα στοιχεία και ανεβάζουμε τις φωτογραφίες με τις σωστές διαστάσεις.

Μεγάλη προσοχή θέλει να δώσουμε όταν θα συμπληρώσουμε το κενό της περιγραφής έτσι ώστε να κάνουμε την εφαρμογή μας όσο ποιο ελκυστική γίνεται ανάμεσα σε άλλες εφαρμογές για Android.

**ΕΦΑΡΜΟΓΗ 2η: Ρίξιμο ζαριών**

Θα κατασκευάσουμε μια εφαρμογή που θα δίνει τη δυνατότητα στον χρήστη της να ρίχνει δύο ζάρια με το πάτημα ενός κουμπιού, όπως ακριβώς και στο τάβλι. Το περιβάλλον της εφαρμογής μας θα μοιάζει με το παρακάτω.



**Βήμα 1: Δημιουργία νέου Project και ρυθμίσεις οθόνης**

Δημιουργήστε ένα project με όνομα Zaria.

Αρχικά, το μοναδικό διαθέσιμο συστατικό της εφαρμογής είναι η οθόνη (**Screen1**) και θα χρειαστεί να μεταβάλλετε κάποιες από τις ιδιότητές της.

Επιλέξτε το αντικείμενο **Screen1** από το τμήμα με την επικεφαλίδα **Components (Συστατικά)** και αλλάξτε τις ιδιότητες στο δεξί μέρος της σελίδας στο τμήμα με την επικεφαλίδα **Properties (Ιδιότητες)**.

Screen Orientation: Portrait

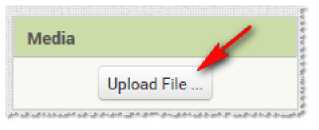
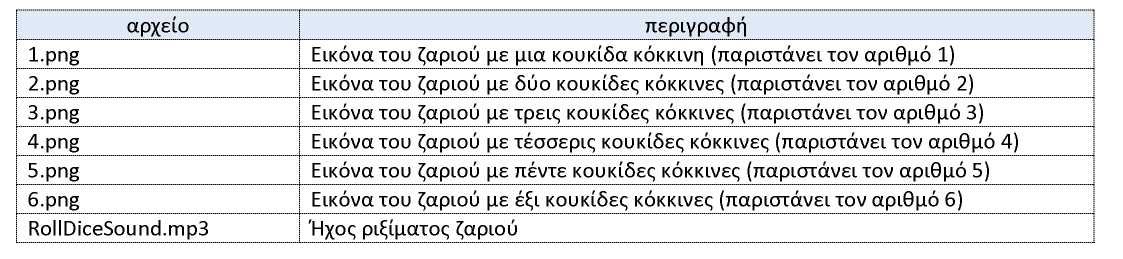
Title: Ζάρια

BackgroundColor: Dark Gray

**Βήμα 2: Προσθήκη των απαραίτητων αρχείων πολυμέσων**

Προσθέστε στο Project τα αρχεία εικόνας και τους ήχους που θα χρησιμοποιεί η εφαρμογή (**βρίσκονται στο drive – σε κοινή χρήση με μένα**).

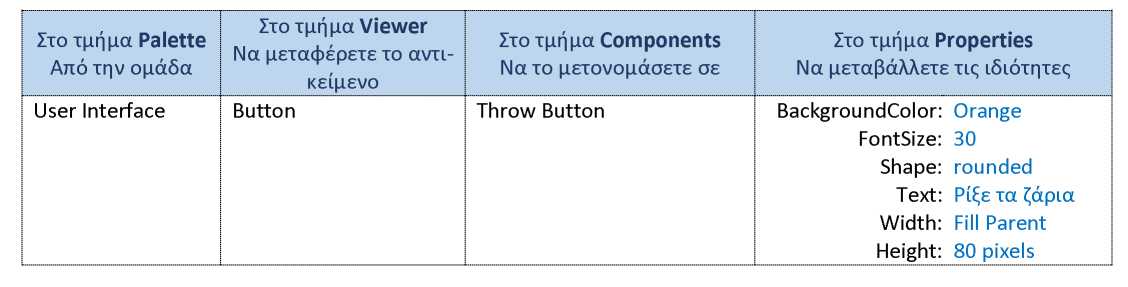
Κατεβάστε το συμπιεσμένο αρχείο RollDice.zip και αποσυμπιέστε το σε έναν φάκελο της αρεσκείας σας. (π.χ. στην Επιφάνεια εργασίας). Θα εμφανιστούν τα ακόλουθα αρχεία.



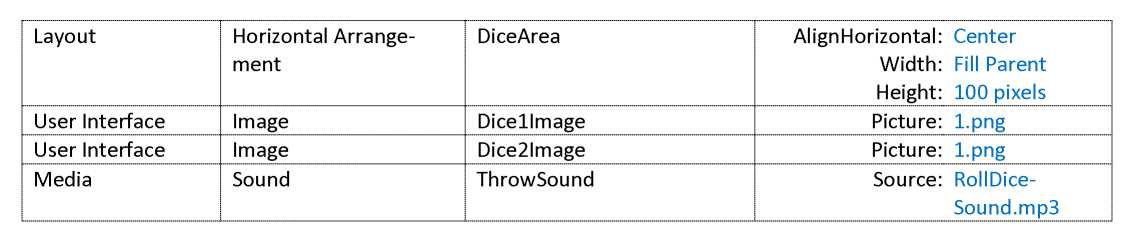
Πρέπει να ανεβάσετε τα 7 παραπάνω αρχεία στην εφαρμογή σας.

Από το τμήμα της σελίδας με ετικέτα Media επιλέξτε το κουμπί **Upload File...** και μετά επιλέξτε να ανεβάσετε στο Project τον ήχο και τις εικόνες.

**Βήμα 3: Προσθήκη αντικειμένων στο γραφικό περιβάλλον**



Για το σχεδιασμό της εμφάνισης του project κάντε τις ενέργειες που εμφανίζονται στον παρακάτω πίνακα:



Από το πάνω δεξί μέρος του App Inventor επιλέξτε το κουμπί **Blocks** για να υλοποιήσετε τη συμπεριφορά της εφαρμογής.

**Βήμα 4: Συγγραφή Προγράμματος (Blocks)**

Μέχρι στιγμής, στο παράθυρο σχεδίασης (Designer) έχουμε μεν σχεδιάσει ένα μέρος της «ο­θόνης» της εφαρμογής μας, αλλά δεν έχουμε ορίσει καθόλου την συμπεριφορά της. Θα μεταφερθούμε λοιπόν στο παράθυρο συγγραφής προγράμματος **Blocks** (το κουμπί μετά­βασης βρίσκεται πάνω δεξιά) για να συσχετίσουμε ενέργειες με γεγονότα και, ουσιαστικά, να προγραμματίσουμε, προσθέτοντας τις κατάλληλες εντολές.

**Βήμα 5: Γεγονός - Άγγιγμα του κουμπιού «Ρίξε ζάρια»**

Το ρίξιμο των ζαριών θα ενεργοποιείται όταν ο χρήστης αγγίζει το κουμπί «Ρίξε ζάρια». Θα πρέπει να συνδέσουμε το γεγονός αυτό (το άγγιγμα του κουμπιού) με τις κατάλληλες ενέρ­γειες.

Πρέπει, λοιπόν, να τοποθετήσουμε στην περιοχή εντολών την κατάλληλη εντολή που θα «ανιχνεύει» το άγγιγμα του κουμπιού.

Όταν προγραμματίζουμε μια συγκεκριμένη συμπεριφορά για την εφαρμογή μας πρακτικά συναρμολογούμε πλακίδια εντολών (και όχι μόνο). Για κάθε πλακίδιο που θέλουμε να προ­σθέσουμε **ανατρέχουμε στην κατάλληλη ομάδα στα αριστερά της οθόνης**, ανοίγει το αντί­στοιχο «συρτάρι» με τις διαθέσιμες εντολές, αναζητούμε και επιλέγουμε το πλακίδιο που χρειαζόμαστε και το σέρνουμε στον χώρο σύνταξης των προγραμμάτων.

Για το σκοπό μας (Όταν πατηθεί το κουμπί ThrowButton να εκτελεστεί κάποια ενέργεια), ε­πιλέγουμε το αντικείμενο ThrowButton από την αριστερή πλευρά της σελίδας, όπου περι­λαμβάνονται όλα τα αντικείμενα της εφαρμογής μας. Στη συνέχεια σέρνουμε με το ποντίκι, την εντολή **when [throwButton].click ... do** στο χώρο σύνταξης του προγράμματος, δεξιά.

**Βήμα 6: Ρίξιμο ζαριού – τυχαίοι αριθμοί**

Για να προσομοιώσουμε το ρίξιμο των ζαριών θα πρέπει να παράγουμε 2 τυχαίους αριθμούς, έναν για το κάθε ζάρι, και να θέτουμε, ανάλογα με τον αριθμό που θα παραχθεί, και την κατάλληλη εικόνα ζαριού στα αντίστοιχα σημεία.

Αρχικά για να αλλάξουμε την εικόνα που απεικονίζει το αντικείμενο Dicellmage, θα επιλέξουμε το αντικείμενο Dice1Image, για να βρούμε την εντολή **set Dice1Image.Picture to ...**



Η εντολή αυτή θα «κουμπώσει» κάτω από την εντολή ενεργοποίησης του γεγονότος που είδαμε προηγουμένως. Το αποτέλεσμα θα είναι όπως παρακάτω.



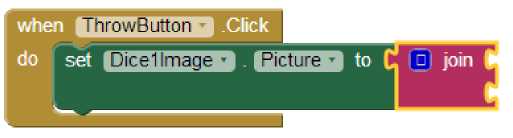
Στη συνέχεια θα πρέπει να προσθέσουμε στην «εγκοπή» της εντολής το όνομα του αρχείου εικόνας που απεικονιστεί. Εδώ πρέπει να προσθέσουμε ένα κείμενο που θα περιέχει το ό­νομα του αρχείου.

Το ένα τμήμα του ονόματος θα είναι ο **αριθμός της εικόνας**, τον οποίο θα παράγουμε με τυχαίο τρόπο και το **δεύτερο τμήμα του κειμένου θα είναι η επέκτασή του («.png»)**.

Επομένως θα πρέπει να συνδυάσουμε δύο τμήματα κειμένου. Για το σκοπό αυτό θα χρησι­μοποιήσουμε, από την ομάδα **Text**, την εντολή ...



... την οποία και θα κουμπώσουμε στην εγκοπή της εντολής αλλαγής εικόνας.

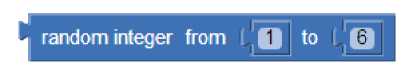


Στην πρώτη εγκοπή της εντολής θα βάλουμε το όνομα του αρχείου εικόνας και στη δεύτερη εγκοπή την κατάληξή του.

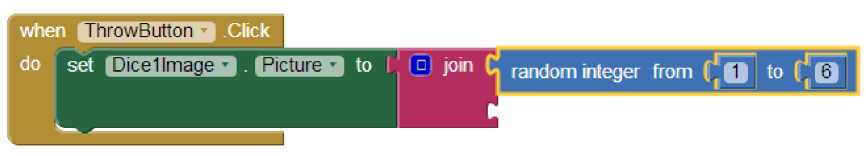
Το όνομα του αρχείου της εικόνας είναι ένας αριθμός από 1 μέχρι 6. Στο σημείο αυτό θα πρέπει να παράγουμε έναν τυχαίο αριθμό σε αυτό το διάστημα που θα τον χρησιμοποιή­σουμε σαν όνομα της εικόνας. Για το σκοπό αυτό θα ανατρέξουμε στην ομάδα **Math**, όπου εντοπίζουμε την εντολή ...



Θα προσαρμόσουμε τους αριθμούς που παράγονται στο διάστημα 1 - 6 (όσοι είναι και οι αριθμοί των ζαριών μας), αλλάζοντας το δεύτερο όριο τις εντολής από 100 σε 6:



Η εντολή αυτή θα «κουμπώσει» κάτω από την εντολή ενεργοποίησης του γεγονότος που είδαμε προηγουμένως. Το αποτέλεσμα θα είναι όπως παρακάτω.



Στη 2η εγκοπή της εντολής join θα προσθέσουμε το κείμενο «.png» που είναι η κατάληξη του ονόματος αρχείου. Για το σκοπό αυτό, θα χρειαστούμε, από την ομάδα **Text**, το πλακίδιο κειμένου που φαίνεται παρακάτω.



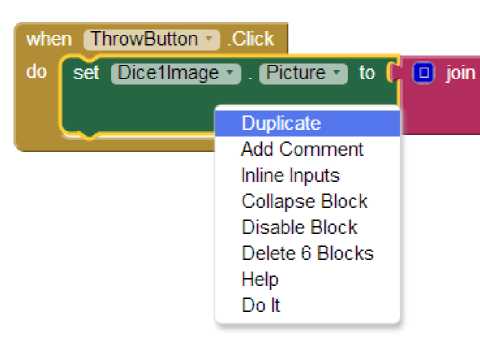
Η τελική μορφή του κώδικα μετά την προσθήκη της εντολής κειμένου φαίνεται στο επόμενο σχήμα.



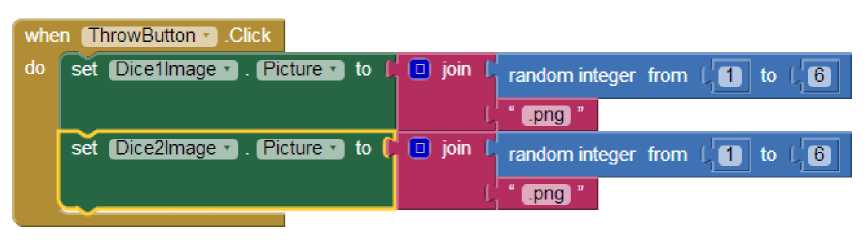
**Βήμα 7: Ρίξιμο δεύτερου ζαριού - Διπλασιασμός εντολών**

Είναι προφανές ότι θέλουμε να εκτελεστούν ακριβώς οι ίδιες ενέργειες για το αντικείμενο Dice2Image. Αντί να ξανακάνουμε τη διαδικασία που περιεγράφηκε προηγουμένως μπο­ρούμε απλά να διπλασιάσουμε την εντολή **set Dice1Image.Picture to** και να την τροποποιή­σουμε κατάλληλα για να λειτουργεί με την εικόνα του 2ου ζαριού.

Κάνουμε δεξί κλικ πάνω στην εντολή που θέλουμε να διπλασιάσουμε και επιλέγουμε την εντολή Duplicate (διπλασιασμός).



Το μόνο που χρειάζεται να αλλάξουμε είναι το όνομα του αντικειμένου από Dice1Image σε Dice2Image.



**Βήμα 8: Αναπαραγωγή ήχου ζαριών**

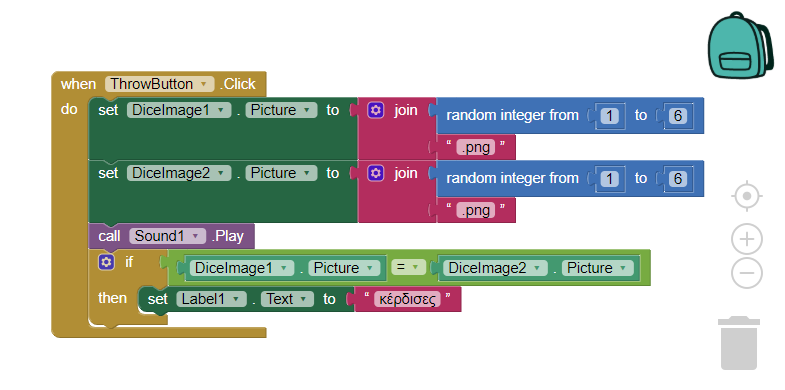
Η αναπαραγωγή του ήχου των ζαριών θα ενεργοποιείται και αυτή με το γεγονός του αγγίγ­ματος του κουμπιού «Ρίξε ζάρια».

Για να ξεκινήσει η αναπαραγωγή του ήχου θα χρησιμοποιήσουμε την εντολή



… την οποία και θα κουμπώσουμε κάτω από τις υπόλοιπες.

Το ολοκληρωμένο πρόγραμμα φαίνεται στην εικόνα που ακολουθεί.



**Επέκταση της εφαρμογής:**

**Βήμα 8: Ρίξιμο των ζαριών με κούνημα της συσκευής**

Προς το παρόν το ρίξιμο των ζαριών ενεργοποιείται με το άγγιγμα του κουμπιού «Ρίξε ζάρια».

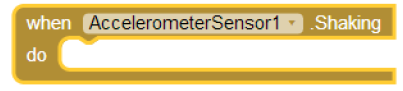
Οι κινητές συσκευές τύπου tablet ή κινητού τηλεφώνου περιλαμβάνουν μια σειρά από αισθητήρες, δηλαδή από εξαρτήματα που μετρούν διάφορα με-γέθη που αφορούν την κινητή συσκευή.

Ένας τέτοιος αισθητήρας είναι το Accelerometer ή επιταχυνσιόμετρο στα ελ-ληνικά, με τη βοήθεια του οποίου η κινητή συσκευή παίρνει πληροφορίες για το αν επιταχύνεται και σε ποιον άξονα (Χ,Υ,Ζ) καθώς και για το αν την μετακινούμε πέρα δώθε

Επανερχόμαστε στο τμήμα σχεδίασης της εφαρμογής (design).

Από την ομάδα **Sensors** της Παλέτας(**Palette**) επιλέγουμε και τοποθετούμε στην περιοχή σχεδίασης το αντικείμενο Accelerometer.

Στη συνέχεια μεταβαίνουμε στο τμήμα εντολών (Blocks) και επιλέγουμε το κατάλληλο γεγο-νός «ανίχνευσης» του κουνήματος της συσκευής, που βρίσκεται κάτω από το αντικείμενο AccelerometerSensor1, για να το προσθέσουμε στις εντολές της εφαρμογής.

****

Οι εντολές που θα εκτελούνται με την κίνηση της συσκευής θα είναι ακριβώς οι ίδιες με αυτές του αγγίγματος του κουμπιού. Τις πρσθέτουμε, αξιοποιώντας τη λειτουργία του διπλασια-σμού εντολών (duplicate).

**Βήμα 9: Προσθήκη σκορ στην εφαρμογή**

Καλό θα είναι η εφαρμογή να δείχνει στον χρήστη μετά από πόσες προσπάθειες έφερε διπλές, δηλαδή να μετράει μετά από πόσες προσπάθειες κέρδισε το παιχνίδι.

Για το σκοπό αυτό θα προσθέσουμε μια μεταβλητή, που θα έχει το ρόλο του μετρητή, δηλαδή θα μετράει πόσες φορές έχει πατήσει ο χρήστης το κουμπί ή έχει κουνήσει τη συσκευή.

Η μεταβλητή είναι ένας χώρο μνήμης στον υπολογιστή μας που μπορεί να αποθηκεύσει μια τιμή. Η αναφορά σε αυτόν τον χώρο γίνεται με ένα συμβολικό όνομα που επιλέγουμε εμείς.

Μπορείτε να φανταστείτε την μεταβλητή σαν ένα πολύ μικρό σάκο που χωράει μονάχα ένα αντικείμενο. Μπορούμε να μεταβάλλουμε το περιεχόμενο του σάκου, δηλαδή να αλλάξουμε την τιμή μιας μεταβλητής, αντικαθιστώντας πρακτικά το προηγούμενο αντικείμενο που έχει, με το νέο αντικείμενο που θα τοποθετήσουμε.

Παρακάτω φαίνεται ένα τέτοιο παράδειγμα. Έστω ότι έχουμε μια μεταβλητή με όνομα σκορ στην οποία δίνουμε αρχικά την τιμή 0. Στη συνέχεια της δίνουμε την τιμή 50, οπότε η προηγούμενη τιμή της (δηλαδή το 0) αντικαθίσταται από τη νέα τιμή.

****

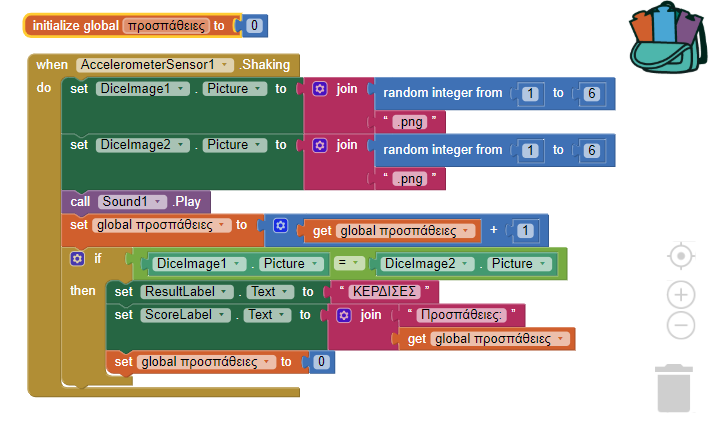
Εμείς θα χρησιμοποιήσουμε τη μεταβλητή προσπάθειες για να αποθηκεύουμε πόσα ριξίματα των ζαριών έχουν γίνει. Αρχικά θα της δώσουμε την τιμή 0.



Κάθε φορά που θα ρίχνουμε τα ζάρια, η τιμή της θα αυξάνεται κατά 1 και θα παίρνει πάλι την τιμή 0 για να ξεκινάει ο μετρητής από την αρχή.

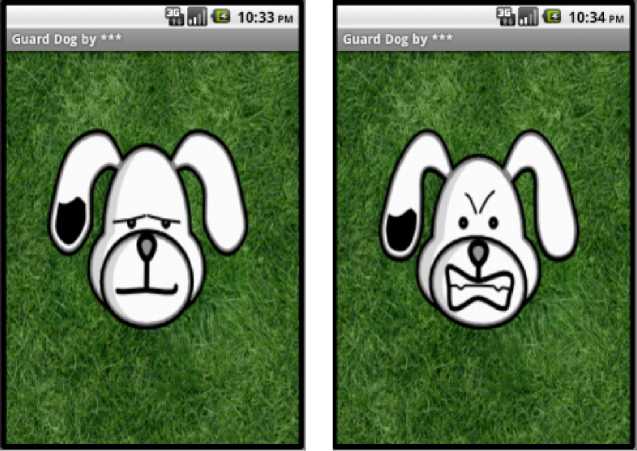
Επίσης θα προσθέσουμε ένα Label που θα ονομάζεται scoreLabel για να εμφανίζει τον αριθμό των προσπαθειών που έγιναν. Στο κείμενο αυτού του Label θα εμφανίζεται το κείμενο Προσπάθειες: και ο αριθμός των προσπαθειών που έγιναν.

Πιο κάτω φαίνεται το κομμάτι του προγράμματος με τις προσθήκες των τελευταίων βημάτων:

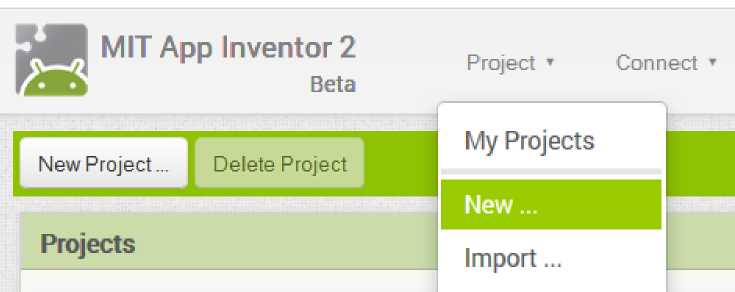


**Εφαρμογή 3: Φύλακας Αγγελος**

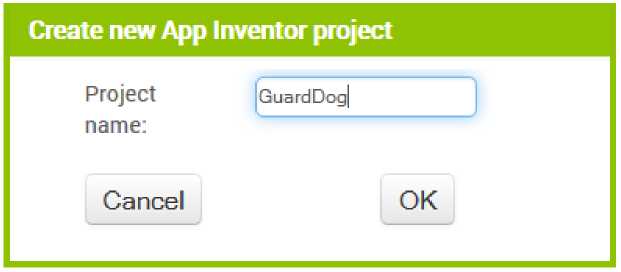
Θα αναπτύξουμε σταδιακά μια **εφαρμογή «συναγερμό»**, χρησιμοποιώντας έναν σκύλο που δεν θα επιτρέπει σε κανέναν να πειράζει την συσκευή μας.



Δημιουργία νέου project



...το οποίο ονομάζουμε GuardDog.



Αρχικά, βρισκόμαστε στην ενότητα **Designer**, στην οποία σχεδιάζουμε τη *διεπαφή* (interface) της εφαρμογής μας, προσθέτοντας τα απαραίτητα *συστατικά* (components) και ορίζοντας *ιδιότητες* (properties) για αυτά.

Το μοναδικό συστατικό μέχρι στιγμής, είναι η οθόνη (Screen1). Προτού προ­σθέσουμε άλλα συστατικά, κάνουμε ορισμένες απαραίτητες τροποποιήσεις στις ιδιότητες της οθόνης, που βρίσκονται στο πλαίσιο **Properties**, στα δεξιά.



Επιλέγουμε το χρώμα του φόντου (BackgroundColor) να είναι μαύρο.

Η ιδιότητα Scrollable δεν πρέπει να είναι επιλεγμένη. Στον τίτλο (Title) προσθέτουμε το όνομα της ομάδας μας, τα ονόματά μας ή τα αρχικά μας.

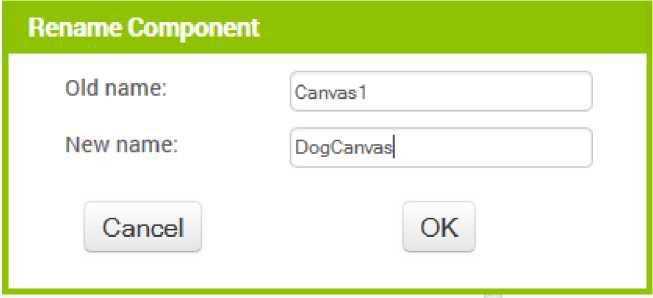
Ο καμβάς (Canvas) είναι μια ορθογώνια επιφάνεια, εντός της οποίας μπορούμε να χειριζόμαστε φιγούρες (Sprites) ή να σχεδιάζουμε αγγίζοντάς την.

Προκειμένου να ορίσουμε μια περιοχή στην οποία θα είναι δυνατή η μετακί­νηση της φιγούρας του σκύλου, θα προσθέσουμε στην οθόνη μας έναν καμβά.

Σέρνουμε λοιπόν από την παλέτα αριστερά (Palette), από την ομάδα πλακιδίων Drawing and Animation, το συστατικό Canvas, και το αποθέτουμε στην οθόνη.



Είναι σημαντικό (αλλά όχι υποχρεωτικό), να ονομάζουμε τα συστατικά που χρησιμοποιούμε με τέτοιο τρόπο, ώστε να τα αναγνωρίζουμε ευκολότερα. Γι' αυτό το λόγο, *μετονομάζουμε* τον καμβά Canvas1 σε DogCanvas.



Παρατήρηση: Τα ονόματα των συστατικών πρέπει να αποτελούνται από λατινικούς χαρακτήρες, αριθμούς ή κάτω παύλες και πρέπει να ξεκινάνε με χαρακτήρα, οπότε δεν μπορούμε να χρησιμοποιήσουμε π.χ. ελληνικούς χαρακτήρες ή κενά.

Τέλος, μεταβάλλουμε τις ιδιότητες του καμβά DogCanvas ως εξής:

Η επιλογή Fill Parent για τις ιδιότητες Width και Height του καμβά του επιτρέπει να επεκταθεί και να καταλάβει όλο το διαθέσιμο χώρο. Έτσι, ο σκύλος θα μπορεί να κινείται κατά μήκος και κατά πλάτος όλης της οθόνης.

Για να χρησιμοποιηθεί ως φόντο του καμβά DogCanvas η εικόνα με το γρασίδι (Grass.jpg), θα πρέπει πρώτα αυτή να «ανέβει» στο project μας. Κάνουμε κλικ στην ιδιότητα BackgroundImage, επιλέγουμε Upload File...



...μετά Αναζήτηση... (Browse...) κι επιλέγουμε το αρχείο Grass.jpg.



Παρόμοιες κινήσεις κάνουμε για κάθε αρχείο εικόνας ή ήχου που σκοπεύουμε να χρησιμοποιήσουμε στο project.

Στο βήμα αυτό θα προσθέσουμε τη φιγούρα του σκύλου-φύλακα.

Οι φιγούρες τοποθετούνται εντός του καμβά και μπορούν να αλληλεπιδρούν με τον καμβά, με άλλες φιγούρες εντός του καμβά και φυσικά να αντιδρούν στις δικές μας ενέργειες.

Προκειμένου να προσθέσουμε τον σκύλο, σέρνουμε μια φιγούρα (ImageSprite) μέσα στον καμβά. Μέσω της ιδιότητας Picture της φιγούρας, ορίζουμε σαν εικόνα της την ήρεμη εκδοχή του σκύλου, κατά παρόμοιο τρόπο με το Βήμα 2.



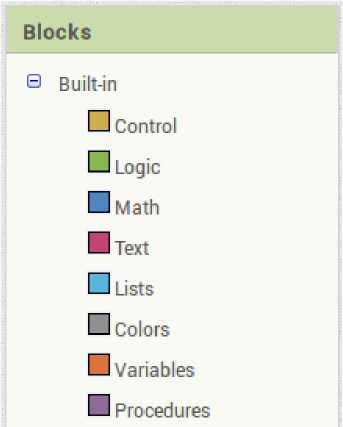
Αν ο σκύλος φαίνεται πολύ μεγάλος, μπορούμε να αλλάξουμε τις διαστάσεις του σε pixel, μέσω των ιδιοτήτων Width και Height.



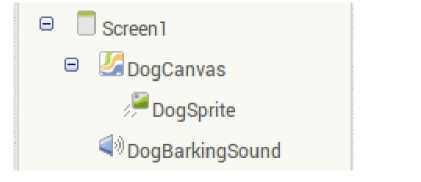
Η ιδιότητα MinimumInterval είναι ο ελάχιστος χρόνος πριν την επανάληψη του ήχου. Αν λοιπόν θέσουμε MinimumInterval: 300 msec (0,3 sec) τότε ο ήχος δεν θα μπορεί να ξαναπαίξει προτού περάσουν τουλάχιστον 0,3 δευτερόλεπτα.

Μέχρι στιγμής, στο Designer έχουμε σχεδιάσει ένα μέρος της «οθόνης» της εφαρμογής μας, αλλά δεν έχουμε ορίσει καθόλου την συμπεριφορά της. Μεταβαίνουμε λοιπόν στο Blocks (το κουμπί μετάβασης βρίσκεται πάνω δεξιά) για να συσχετίσουμε ενέργειες με γεγονότα και, ουσιαστικά, να προγραμ­ματίσουμε, προσθέτοντας τις κατάλληλες εντολές.

Στα αριστερά της οθόνης διακρίνουμε τρεις ομάδες πλακιδίων (blocks). Πρώτα βρίσκουμε τα ενσωματωμένα (built-in) πλακίδια, την λειτουργία των οποίων θα εξετάσουμε αργότερα.

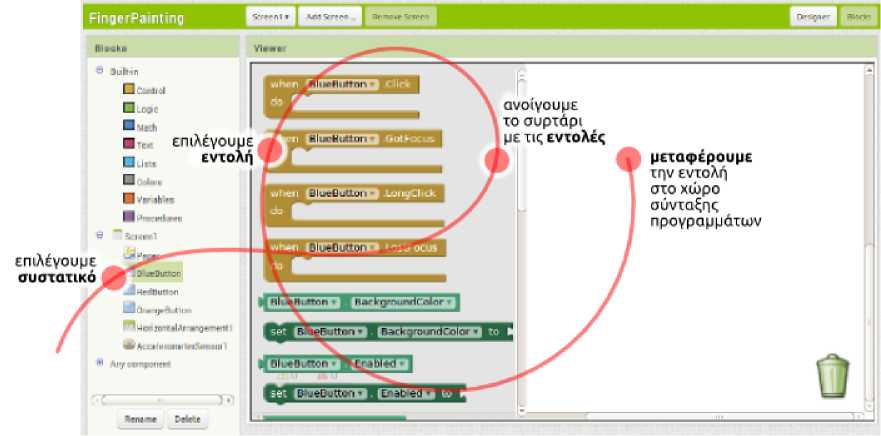


Στη συνέχεια, βλέπουμε τα πλακίδια που αφορούν τα συστατικά που έχουμε προσθέσει στην εφαρμογή μας.



Ακολουθεί η τελευταία ομάδα, που δε θα μας απασχολήσει εδώ.

Όταν προγραμματίζουμε μια συγκεκριμένη συμπεριφορά για την εφαρμογή μας πρακτικά συναρμολογούμε πλακίδια εντολών (και όχι μόνο). Για κάθε πλακίδιο που θέλουμε να προσθέσουμε **ανατρέχουμε στην κατάλληλη ομάδα στα αριστερά της οθόνης**, ανοίγει το αντίστοιχο «συρτάρι» με τις διαθέσιμες εντολές, αναζητούμε και επιλέγουμε το πλακίδιο που χρειαζόμαστε και το σέρνουμε στον χώρο σύνταξης των προγραμμάτων.



Στην συγκεκριμένη περίπτωση, θέλουμε να ακούγεται ο ήχος του γαβγίσματος DogBarkingSound κάθε φορά που ο χρήστης αγγίζει τον καμβά DogCanvas.

Επιλέγουμε λοιπόν το συστατικό DogCanvas, ανοίγει το αντίστοιχο συρτάρι και μεταφέρουμε το πλακίδιο when DogCanvas.TouchDown.



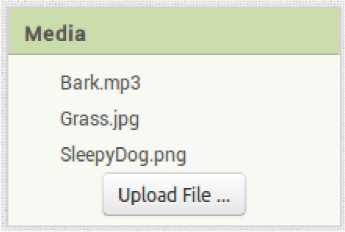
Στη συνέχεια επιλέγουμε το συστατικό DogBarkingSound, εντοπίζουμε το πλακίδιο call DogBarkingSound.Play και το «κουμπώνουμε» μέσα στο προηγούμενο πλακίδιο.



Ας διαβάσουμε τα πλακίδια που συναρμολογήσαμε. Δίνουν τις εντολές που είναι απαραίτητες έτσι ώστε η εφαρμογή μας να κάνει αυτό που θελουμε.

Όταν ο χρήστης αγγίξει τον καμβά DogCanvas, τότε αναπαράγεται ο ήχος DogBarkingSound.

Στο βήμα αυτό θα επεκτείνουμε την συμπεριφορά του σκύλου έτσι ώστε να αλλάζει μορφή και να αγριεύει όταν τον αγγίζουμε.



Προσθέτουμε στο project τη θυμωμένη εκδοχή του σκύλου (AngryDog.png). Στο το πλαίσιο Media πατάμε Upload File...



στη συνέχεια Browse. και επιλέγουμε το αρχείο AngryDog.png.

Εδώ θέλουμε η φιγούρα του σκύλου DogSprite να αλλάζει μορφή όταν ο χρήστης την αγγίζει και να επανέρχεται όταν η επαφή σταματά. Όλα τα πλακίδια αφορούν την μορφή DogSprite και θα τα βρούμε στο συρτάρι που εμφανίζεται όταν την επιλέξουμε.



Αρχικά, εντοπίζουμε και συναρμολογούμε τα παρακάτω πλακίδια:

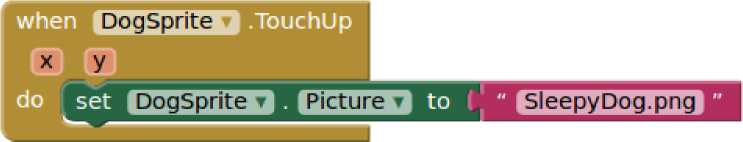


Παρατηρούμε ότι οι εντολές μας είναι ελλιπείς. Πώς θα προσδιορίσουμε την εικόνα που θα έχει η φιγούρα DogSprite όταν αγγιχθεί; Θα πρέπει να παρέχουμε το αντίστοιχο όνομα αρχείου. Ανατρέχουμε λοιπόν στην ενσωματωμένη ομάδα πλακιδίων Text και επιλέγουμε το πλακίδιο του κενού κειμένου...

...στο οποίο συμπληρώνουμε το όνομα του αρχείου εικόνας "AngryDog.png" και το προσθέτουμε στις εντολές μας.



Αναλόγως προσδιορίζουμε την εικόνα στην οποία θα επανέρχεται ο σκύλος όταν σταματάει η επαφή.



Όταν ο χρήστης αγγίξει τη φιγούρα DogSprite, τότε αυτή παίρνει την μορφή AngryDog.png. Όταν σταματήσει η επαφή με την φιγούρα DogSprite τότε αυτή παίρνει την μορφή SleepyDog.png.

**Εφαρμογή: Υπολογισμός εμβαδού τριγώνου**

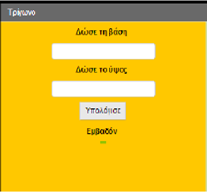
**Όνομα:………………………………………………………………..**

**Τμήμα:…………………………………Ημερομηνία:………………**

**Λογαριασμός χρήστη:……………………………………………**

**Όνομα εφαρμογής: …………………………………………………**

Να δημιουργήσετε εφαρμογή όπως φαίνεται στην παρακάτω εικόνα. Υπενθυμίζεται το εμβαδόν τριγώνου δίνεται από τη σχέση Ε=(βάση\*ύψος)/2.

****

Αναλυτικά:

1. Η οθόνη θα πρέπει να έχει όνομα Τρίγωνο, χρώμα πορτοκαλί και οριζόντια διάταξη κεντρική
2. Θα πρέπει να βάλουμε τις απαραίτητες ετικέτες (labels) για το Δώσε τη βάση, Δώσε το ύψος, Εμβαδόν, =, αποτέλεσμα (μη ορατό αρχικά).
3. Θα βάλουμε ένα κουμπί (button) για το Υπολόγισε.
4. Θα βάλουμε δύο πλαίσια κειμένου (text boxes) ένα για να εισάγει ο χρήστης τη βάση και ένα για να εισάγει ο χρήστης το ύψος.
5. Θα πρέπει να γίνονται οι απαραίτητοι υπολογισμοί προκειμένου να εμφανίζεται το αποτέλεσμα.

Καλή επιτυχία!