Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

Σχολή Τεχνολογικών Εφαρμογών

Τμήμα Εφαρμοσμένης Πληροφορικής & Πολυμέσων

Πτυχιακή εργασία

Τίτλος: Δημιουργία εφαρμογής Android για επεξεργασία εικόνων με την χρήση της βιβλιοθήκης OpenCV

Παπαδημητρίου Μιχάλης ΑΜ: 2394

Επιβλέπων καθηγητής : Γιώργος Τριανταφυλλίδης

Ευχαριστίες

Καταρχάς θα ήθελα να ευχαριστήσω τον επιβλέπων καθηγητή μου Γιώργο Τριανταφυλλίδη ο οποίος με βοήθησε και μου έδωσε μια ευκαιρία να ασχοληθώ με κάτι που μου αρέσει πολύ και θα προσπαθήσω μελλοντικά να επεκταθώ από μόνος μου στο αντικείμενο αυτό.

Θα ήθελα να ευχαριστήσω τους γονείς μου για την υποστήριξη που μου παρείχαν προκειμένου να ολοκληρώσω τις σπουδές μου. Άλλα πάνω από όλα θέλω να ευχαριστήσω του φίλους μου. Οι οποίοι με στήριξαν και με βοήθησαν αυτά τα χρόνια και προπάντων που με ανεχτήκαν τον τελευταίο καιρό που ήμουν αγχωμένος λόγο της σχολής.

Abstract

In this graduate paper I will be referring firstly in the history of mobiles, also in functions which smart phones have and the possibilities that they offer to us ,by summarizing certain of the most popular mobile OS that exist in the market and I will briefly speak to you about them.

Next I will analyze OS Android .I will be referring to:

- What are the differences between Android and other mobile technologies and how programmers can exploit these differences .
- How programmers use development environment Eclipse for Java for developing and debugging applications Android in the simulator and in telephone devices (or tablet).
- What is the structure of Android applications.
- And what possibilities does the SDK has and how can they be used by programmers.

As long as the OpenCV library concerns I will mention some general information. We will refer to what kind of a library it is, what is her use and what are the possibilities provided to the programmer. And I will refer to certain functions of OpenCV and their use.

As long as the program concerns I will create a mobile device application that will be used in OS Android. This application is going to use the camera, to take a picture first, afterwards this picture is going to be processed with the help of the OpenCV library. But firstly we will see a small tutorial on how to install the programs needed for the development environment. And then I will refer to the application and I will explain the code.

In order every time to be able to check the code and the results, I continuously had the device used for the trials connected (Motorola Xoom) and each time I changed something ,such as applying a filter in my picture I immediately put it to my device to check the results. The biggest difficulty I met was understanding the function of the OpenCV library. And the reason why is because OpenCV library provides high level interconnection applications. The library has about 300 functions written in C and some in C++ classes.

As for the final result is concerned , I could say that I have a feeling of satisfaction because I managed to create an application ,that is user friendly providing him with numerous image processing possibilities .

Σύνοψη

Στην πτυχιακή αυτή θα αναφερθώ καταρχάς για το την ιστορία τον κινητών, για την λειτουργία τον smartphone και για τις δυνατότητες που μας προσφέρουν. Θα σας πω περιληπτικά μερικά από τα δημοφιλέστερα λειτουργικά κινητών που υπάρχουν στην αγορά και θα σας μιλήσω με λίγα λόγια για αυτά.

Στην συνέχεια θα δούμε για το λειτουργικό Android. θα αναφερθώ:

- Σε τι διαφέρει το Android από άλλες τεχνολογίες κινητών τηλεφώνων και πως οι προγραμματιστές μπορούν να εκμεταλλευτούν αυτές τις διαφορές.
- Πως οι προγραμματιστές χρησιμοποιούν το περιβάλλον ανάπτυξης Eclipse για τη Java για την ανάπτυξη και εξφαλμάτωση εφαρμογών Android στον εξομοιωτή και στις συσκευές τηλεφώνου (η tablet).
- Ποια είναι η δομή των εφαρμογών Android.
- Και ποιες δυνατότητες έχει το SDK και πως μπορούν να τις χρησιμοποιήσουν οι προγραμματιστές.

Όσο αφορά πάλι για την βιβλιοθήκη OpenCV θα σας πω γενικά στοιχειά. Θα μιλήσουμε για το τι βιβλιοθήκη είναι, για το ποια είναι η χρήση της και για τις δυνατότητες που παρέχει στον προγραμματιστή. Και θα σας αναφέρω μερικές συναρτήσεις της OpenCV και για το ποια είναι η χρήση τους.

Όσο αφορά τώρα για το πρόγραμμα. Θα δημιουργήσω μια εφαρμογή κινητής συσκευής για το λειτουργικό σύστημα Android η οποία θα κάνει χρήση της κάμερας μας και στην συνέχεια θα παίρνει την φωτογραφία που τράβηξα και θα την επεξεργάζεται με την βοήθεια της βιβλιοθήκης OpenCV. Καταρχάς θα δούμε ένα μικρό tutorial για το πώς μπορούμε να εγκαταστήσουμε τα προγράμματα που χρειάζονται για το περιβάλλον ανάπτυξης. Και στην συνέχεια θα σας μιλήσω για την εφαρμογή και θα σας εξηγήσω τον κώδικα.

Για να μπορέσω να ελέγχω κάθε φορά τον κώδικα και τα αποτελέσματα του έπρεπε να έχω συνδεμένο συνεχώς την συσκευή που έγιναν οι δοκιμές(Motorola Xoom) και κάθε φορά που άλλαζα κάτι, έβαζα ένα φίλτρο στην εικόνα μου ή οτιδήποτε άλλο αμέσως το πέρναγα στην συσκευή για να δω κατά πόσο είναι σωστά τα αποτελέσματα. Η μεγαλύτερη δυσκολία που συνάντησα ήταν να κατανοήσω την λειτουργία της βιβλιοθήκης OpenCV. Και ο λόγος αυτός οφείλετε γιατί η OpenCV παρέχει μια μεσαίου έως υψηλού επίπεδου διασύνδεση εφαρμογών με περίπου τριακόσιες συναρτήσεις γραμμένες σε C και μερικές κλάσεις C++.

Άλλα ως τελικό αποτέλεσμα μπορώ να πω ότι έμεινα αρκετά ικανοποιημένος γιατί κατάφερα να δημιουργήσω μια εφαρμογή πολύ εύκολη ως προς τον χρήστη παρέχοντας του πολλές δυνατότητες επεξεργασίας εικόνας.

Περιεχόμενα

Εικόνες

1. Εισαγωγή

1.1 Σκοπός

Ο σκοπός της πτυχιακής αυτής είναι να μπορέσουμε να κατανοήσουμε την λειτουργιά του λειτουργικού συστήματος για φορητές συσκευές Android. Να κατανοήσουμε κάποια πράγματα για την βιβλιοθήκη OpenCV όπως ποια είναι η χρήση της και πως μπορούμε να την ενσωματώσουμε στην εφαρμογή μας. Επίσης μετά το τέλος της πτυχιακής θα πρέπει να έχουμε μάθει να χειριζόμαστε το περιβάλλον ανάπτυξης Android γρήγορα και ευέλικτα ώστε να μπορούμε μελλοντικά να δημιουργούμε εφαρμογές Android σε όποια έκδοση μας ζητηθεί. Θα έχουμε την ικανότητα να εισάγουμε στο πρόγραμμα μας άλλες βιβλιοθήκες για να μπορέσουμε να κάνουμε πιο εύκολη την δημιουργία πιο εξειδικευμένων εφαρμογών που θα μας ζητηθούν.

1.2 Διαρρύθμιση κεφαλαίων

Στο επόμενο κεφαλαίο θα αναφερθούμε με λίγα λόγια για την ιστορία της κινητής τηλεφωνίας ποιος ο δημιουργός της πρώτης κινητής συσκευής τηλεφώνου και πως εξελίχθηκε μέχρι σήμερα. Θα μιλήσουμε για τα Smartphone που είναι η εξέλιξη τον κινητών, για της δυνατότητες τους και θα πούμε για ποιο λόγο θεωρούνται ανώτερα από τα παλιά κινητά μας τηλέφωνα. Και στο τέλος του κεφαλαίου θα αναφερθούμε για το πώς μπορούμε να επιλέξουμε ένα Smartphone σύμφωνα με της ανάγκες μας.

Στο 3ο κεφαλαίο θα μιλήσουμε για τα λειτουργικά συστήματα κινητών συσκευών που έχουμε σήμερα. Λίγα λόγια για τις εκδόσεις που έχουν αυτά τα λειτουργικά, για την απόδοση τους, για το τι υποστηρίζει η οθόνη τους, για το περιβάλλον και τις εφαρμογές τους και για την ασφάλεια που παρέχουν αυτά τα λειτουργικά. Και θα βγάλουμε κάποια συμπεράσματα σύμφωνα με την δίκια μας κρίση.

Στο 4ο κεφάλαιο θα αναφερθούμε καθαρά για το Android. Καταρχάς θα σας πούμε λίγα λόγια για την ιστορία του Android και για το πώς έφτασε εκεί που είναι σήμερα. Θα αναφερθούμε και για το Google Play και για την πληθώρα τον εφαρμογών που μας παρέχουν. Στην συνέχεια του κεφαλαίου θα πούμε για τις εκδόσεις που έχουμε μέχρι και σήμερα και για τις διαφορές αυτόν τον εκδόσεων. Και τέλος θα μιλήσουμε για την αρχιτεκτονική του android.

Στο 5ο κεφάλαιο θα αναφερθούμε για το SDK, για τις άδειες χρήσης, θα κάνουμε μια τεκμηρίωση για το Android και λίγα λόγια για το περιβάλλον υλοποίησης και για τα εργαλεία-δείγματα εφαρμογών. Και στην συνέχεια θα αναφερθούμε αναλυτικά για τους καταλόγους που έχει ένα Android project.

Στο 6ο κεφάλαιο θα αναφερθούμε καθαρά για την OpenCV. Θα μιλήσουμε για γενικά στοιχειά για την OpenCV για το τι είδους βιβλιοθήκη είναι και για το ποια είναι η χρήση της. Θα πούμε μερικές από της υπάρχουσες δομές που έχει η βιβλιοθήκη. Επίσης θα αναφερθούμε

σε μερικές ομάδες συναρτήσεων για τον εντοπισμό κίνησης, εντοπισμό δέρματος και για την επεξεργασία εικόνας.

Στο 7^{ο} κεφάλαιο θα πούμε πια το πώς μπορούμε να εγκαταστήσουμε τα προγράμματα για την δημιουργία ενός περιβάλλοντος σχεδίασης εφαρμογών Android. Θα γίνει μια επίδειξη με εικόνες βήμα προς βήμα για την σωστή εγκατάσταση τον προγραμμάτων.

Στο 8^{ο} και τελευταίο κεφάλαιο μας μιλήσουμε πια για την εφαρμογή μας. Στην αρχή του κεφαλαίου θα σας δείξουμε πως μπορούμε να δημιουργήσουμε το project μας με την έκδοση Android που επιθυμούμε και για το πώς μπορούμε να εισάγουμε την βιβλιοθήκη OpenCV. Για το πώς μπορούμε να χρησιμοποιήσουμε έναν πόρο της συσκευής στην εφαρμογή μας όπως είναι η κάμερα. Επίσης θα αναφερθούμε για την δημιουργία Buttons, ImageButton, Seekbar και EditText. Θα πούμε ακόμα για το πώς λειτούργει η OpenCV στην εφαρμογή μας και πως εισάγουμε τα φίλτρα μας στην εικόνα και άλλα πολλά. Τέλος θα δούμε πως μπορούμε να αποθηκεύσουμε την εικόνα μας με το όνομα που επιθυμούμε εμείς.

Στο Τέλος θα σας δώσω όλο τον κώδικα για την εφαρμογή που δημιούργησα.

2. Γενικά

2.1 Η ιστορία της κινητής τηλεφωνίας

Εικόνα 1 Ο Μάρτιν Κούπερ με το πρώτο κινητό της Motorola

Η περιπέτεια της κινητής τηλεφωνίας ξεκίνησε αμέσως μετά τον Β' Παγκόσμιο Πόλεμο, με τις πρώτες προσπάθειες των Σουηδών, Φιλανδών και Αμερικανών. Όμως, ως ληξιαρχική πράξη γέννησής της θεωρείται η 3η Απριλίου 1973.

Ήταν ένα μουντό ανοιξιάτικο πρωινό στη Νέα Υόρκη. Ο δόκτωρ Μάρτιν Κούπερ της Motorola, περπατώντας σ' ένα δρόμο της αμερικάνικης μεγαλούπολης ήξερε ότι έγραφε ιστορία. Στα δυο του χέρια κρατούσε μια συσκευή που έμοιαζε με φορητό ασύρματο. Είχε ύψος 25 εκατοστά και βάρος 900 γραμμάρια. Ήταν το πρώτο σύγχρονο κινητό τηλέφωνο με τον κωδικό MotorolaDynaTAC. Σχημάτισε τον αριθμό του βασικού ανταγωνιστή του, Τζόελ Ένγκελ, που δούλευε για λογαριασμό της Bell Labs.

«Γεια σου Τζο, σου μιλάω από ένα αληθινό κινητό τηλέφωνο» του είπε. «Παρότι δεν είχαμε τις καλύτερες των σχέσεων, μου συμπεριφέρθηκε πολύ ευγενικά», δήλωσε χρόνια αργότερα ο Κούπερ σε μια συνέντευξή του. Η Bell πήρε τη ρεβάνς το 1978, κατασκευάζοντας το πρώτο δοκιμαστικό δίκτυο κινητής τηλεφωνίας, που ήταν αναγκαίο για την εξέλιξη και την εμπορική εκμετάλλευση του κινητού.

Το πρώτο αυτοματοποιημένο δίκτυο κινητής τηλεφωνίας λειτούργησε στις αρχές της δεκαετίας του '80 στη Σκανδιναβία. Μέχρι τα τέλη της δεκαετίας του '80 τα κινητά τηλέφωνα ήταν ογκώδη για να μεταφέρονται στην τσέπη κι έτσι ήταν εγκατεστημένα κυρίως σε αυτοκίνητα. Το πρώτο κινητό που έλαβε άδεια έγκρισης ήταν το μοντέλο της Μοτορόλα DynaTAC8000X. Υπήρξε η ναυαρχίδα των λεγόμενων κινητών πρώτης γενιάς (1G).

Εικόνα 2 Μερικά κινητά

Στην αρχή της δεκαετίας του '90 άρχισε η απογείωση των κινητών τηλεφώνων, με την ψηφιοποίηση δικτύων (GSM) και συσκευών. Τα κινητά έγιναν μικρότερα (100-200 γραμμάρια), χωρούσαν στην παλάμη και έμπαιναν έστω και με δυσκολία στην τσέπη του χρήστη τους. Περάσαμε έτσι στα κινητά της δεύτερης γενιάς (2G), που παρείχαν και άλλες ευκολίες, όπως την αποστολή σύντομων γραπτών μηνυμάτων (SMS) και τη λήψη φωτογραφιών.

Στις αρχές του 21ου αιώνα ήλθαν τα κινητά τρίτης γενιάς (3G), με τις απεριόριστες δυνατότητες των πολυμέσων. Σήμερα, η διείσδυση του κινητού τηλεφώνου στον

πλανήτη ξεπερνά το 30%, με αλματώδη άνοδο στις φτωχές χώρες του πλανήτη και κυρίως στην Αφρική. Η φιλανδική εταιρεία Nokia, με μερίδιο αγοράς 36%, κατέχει την πρώτη θέση στις πωλήσεις κινητών τηλεφώνων παγκοσμίως.

Στην Ελλάδα η κινητή τηλεφωνία έκανε την εμφάνισή της το 1992, με την προκήρυξη διαγωνισμού από την κυβέρνηση Μητσοτάκη για τη χορήγηση δύο αδειών. Ο αποκλεισμός του ΟΤΕ από τη διαδικασία αδειοδότησης προκάλεσε θύελλα διαμαρτυριών κατά της κυβέρνησης. Η κυβέρνηση αντέτεινε την αφερεγγυότητα του οργανισμού (καθυστερήσεις στις συνδέσεις σταθερών τηλεφώνων που έφθανε και τα 15 χρόνια, Υπόθεση Τόμπρα κ.ά.), αλλά και τα οικονομικά οφέλη, που θα είχε από τη χορήγηση των αδειών σε ιδιωτικές εταιρείες. Τελικά, οι δύο άδειες κατακυρώθηκαν στην Panafon (νυν Vodafone), πολυμετοχική εταιρεία με επικεφαλής την αγγλική Vodafone, και στην ιταλική Telestet (μετέπειτα TIM και νυν WIND).

Η Telestet ξεκίνησε την εμπορική της εκμετάλλευση στις 29 Ιουνίου 1993 και η Panafon την 1η Ιουλίου του ίδιου χρόνου. Η Cosmote, συμφερόντων ΟΤΕ, ήταν ο τρίτος παίκτης της αγοράς (Ιανουάριος 1998) και η Q, εταιρεία του ομίλου Φέσσα, ο τέταρτος (19 Ιουνίου 2002). Η Q στη συνέχεια εξαγοράσθηκε από την TIM (Ιανουάριος 2006) κι έτσι σήμερα δραστηριοποιούνται τρεις εταιρείες, WIND, Vodafone και Cosmote, που είναι η ηγέτιδα στο χώρο της κινητής τηλεφωνίας.

Τους πρώτους μήνες του 1993 τα κινητά τηλέφωνα λειτουργούσαν μόνο στην Αττική και τα νησιά του Σαρωνικού. Το κόστος ήταν απαγορευτικό για τους πολλούς. Οι συσκευές στοίχιζαν από 700-1400€, το τέλος ενεργοποίησης 85€, το μηνιαίο πάγιο 40€ και το λεπτό ομιλίας 0,25€. Έτσι, μόνο 1000 ήταν οι συνδρομητές τις πρώτες μέρες του Ιουλίου.

Οι εκτιμήσεις των «ειδικών» έκαναν λόγο για 200.000 συνδρομητές μέσα σε μια δεκαετία. Απέτυχαν παταγωδώς στις προβλέψεις τους. 13 χρόνια μετά, λειτουργούσαν στη χώρα μας 13.551.000 συσκευές (Δεκέμβριος 2006), που καλύπτουν το 120,5% του ελληνικού πληθυσμού, γεγονός που κατατάσσει την Ελλάδα στις πρώτες θέσεις παγκοσμίως σε αναλογία πληθυσμού και κινητών τηλεφώνων. [9]

2.2 Γενικά για τα Smartphone

Αυτό είχε ξεκινήσει αρκετά χρόνια πριν, με κάποια εξειδικευμένα μοντέλα που απευθύνονταν κυρίως σε επαγγελματίες και κόστιζαν πολύ. Τώρα πια, διεισδύουν σε όλους τους καταναλωτές, ορίζοντας τη νέα μόδα. Έτσι, βλέπουμε όλο και περισσότερους να σνομπάρουν τα παλιομοδίτικα "κουμπάκια" και να χειρίζονται το τηλέφωνό τους πιέζοντας ή σέρνοντας το δάχτυλο στην οθόνη.

Το βασικότερο χαρακτηριστικό των smart phones είναι ότι επιτρέπουν την εγκατάσταση εφαρμογών, όπως ακριβώς συμβαίνει και στους υπολογιστές. Αυτή η αγορά αναπτύσσεται ραγδαία, με χιλιάδες νέες εφαρμογές κάθε είδους να ξεφυτρώνουν καθημερινά. Όμως, όλες οι

εφαρμογές δεν είναι συμβατές με όλα τα τηλέφωνα. Υπάρχουν διαφορετικά λειτουργικά και κάθε ένα έχει τις δικές του εφαρμογές, όπως άλλωστε και τα δικά του χαρακτηριστικά.

Ένα άλλο δυνατό σημείο των smart phones είναι η συνδεσιμότητα. Τα πάνε πολύ καλά με ότι έχει να κάνει με internet. Από το απλό browsing και την αποστολή - λήψη email, μέχρι σύνδεση με instant messengers, social networks όπως Facebook και Twitter, YouTube, ακόμα και voip κλήσεις - Skype.

Πολλά περιλαμβάνουν και δέκτη GPS που, σε συνδυασμό με τα κατάλληλα λογισμικά, μας δείχνει τη θέση μας πάνω στο χάρτη, σχεδιάζει διαδρομές προς προορισμούς επιλογής μας ή μας καθοδηγεί σε όλη τη διαδρομή (navigation).Χρησιμοποιούν απευθείας σύνδεση (υψηλής ταχύτητας) στο internet μέσω του αντίστοιχου provider κινητής τηλεφωνίας (με την αντίστοιχη χρέωση δεδομένων) ή συνδέονται εύκολα σε ασύρματα δίκτυα Wi-Fi (σπίτι, δουλειά ή άλλα hot-spots σε cafe, καταστήματα κλπ). [10]

2.3 Πώς επιλέγουμε smartphone?

Αν μας ενδιαφέρει η πρόσβαση στο internet, θέλουμε συσκευή που υποστηρίζει γρήγορη σύνδεση (3G υψηλής ταχύτητας) και σύνδεση με δίκτυα Wi-Fi. Επίσης, σε αυτή την περίπτωση, καλό θα ήταν να έχουμε και μια μεγαλούτσικη οθόνη για να μη βγάζουμε τα μάτια μας.

Σημαντική είναι και η σύνδεση με τον υπολογιστή (μέσω καλωδίου USB ή BlueTooth) για μεταφορά αρχείων και συγχρονισμό, αν και αυτό μάλλον θεωρείται πλέον δεδομένο για όλες τις συσκευές.

Αν γράφουμε συχνά (sms, email, σημειώσεις), θα πρέπει να δούμε αν μας βολεύει το virtual πληκτρολόγιο οθόνης. Ίσως θέλουμε μια μεγαλύτερη οθόνη (με πιο μεγάλα virtual πλήκτρα) ή μια συσκευή που έχει και κανονικό qwerty πληκτρολόγιο. Σε κάθε περίπτωση, πριν επιλέξετε συσκευή δοκιμάστε αν σας βολεύει (από κάποιο φίλο ή σε κάποιο κατάστημα).

Οθόνη: Πολλές οθόνες είναι ευαίσθητες στην αφή, ώστε να μπορεί να χειρίζεται κανείς τη συσκευή χωρίς τη χρήση πλήκτρων. Κάτι τέτοιο μπορεί να είναι πολύ βολικό για κάποιους (ή ακόμα και απαραίτητο). Αυξάνει τις δυνατότητες και ανεβάζει αντίστοιχα την τιμή της συσκευής. Ανάλογα με τη συσκευή, μπορεί να επιτρέπεται η χρήση με τα δάκτυλα ή και με ειδική γραφίδα. Σε κάποιες περιπτώσεις, μπορεί να υπάρχει παράλληλα και ένα κανονικό πληκτρολόγιο qwerty. Θα βρείτε δύο τύπους οθονών αφής: 1. Resistive: Ανταποκρίνεται στην πίεση, χρησιμοποιούμε το δάχτυλο (ή το νύχι) ή μια γραφίδα (με τη χρήση της γραφίδας, συχνά μπορείτε να γράψετε "χειρόγραφα" στην οθόνη και το τηλέφωνο να αναγνωρίσει τους χαρακτήρες και να τους μετατρέψει στον αντίστοιχο ψηφιακό χαρακτήρα). 2. Capacitive: Ανταποκρίνεται στο ίδιο το δάχτυλο, χρησιμοποιούμε το μαλακό μέρος του δάχτυλου. Η εισαγωγή κειμένου γίνεται εύκολα με ένα virtual πληκτρολόγιο πάνω στην οθόνη. Συνήθως, το πρώτο είδος το συναντάμε σε συσκευές symbian ή mobile windows και το δεύτερο σε i-phone και android (χωρίς να είναι απόλυτο). Μπορεί επίσης η οθόνη να υποστηρίζει "multitouch" που αυξάνει τη λειτουργικότητά της.

Αν μας ενδιαφέρουν εξωτερικές εφαρμογές, διαλέγουμε το λειτουργικό σύστημα που μας ταιριάζει περισσότερο (δες παρακάτω).

Αν μας ενδιαφέρει η φωτογραφία, εξετάζουμε τις αντίστοιχες δυνατότητες (φακό, ανάλυση, φλας κλπ).

Βλέπουμε αν υπάρχει ενσωματωμένο GPS, χρήσιμο για χρήση on line χαρτών, navigation και όχι μόνο.

Ενσωματωμένες εφαρμογές: Κάθε συσκευή έρχεται με κάποιες βασικές εφαρμογές προ-εγκατεστημένες (ημερολόγιο, σημειώσεις κλπ). Μπορεί το smart phone μας να μην προσφέρει αρχικά κάποιες από τις εφαρμογές που θα θέλαμε ή που είχαμε συνηθίσει απο παλιότερα τηλέφωνά μας. Όμως, στις περισσότερες περιπτώσεις, μπορούμε να προσθέσουμε αυτές τις εφαρμογές (μαζί με εκατοντάδες άλλες) εκ των υστέρων, εγκαθιστώντας τις.

Λαμβάνουμε υπ' όψιν το μέγεθος και το βάρος. Μεγαλύτερη οθόνη σημαίνει πιο άνετη χρήση, αλλά και πιο μεγάλο μέγεθος και βάρος συσκευής (συνήθως και ψηλότερο κόστος).

Επίσης, την ταχύτητα απόκρισης της συσκευής που εξαρτάται από το hardware (επεξεργαστής, ROM) και το λειτουργικό σύστημα. Θα την βρείτε σε reviews ή με δοκιμή.

Τέλος, κοιτάζουμε την τιμή, που πιθανότητα είναι από λίγο ως πολύ τσουχτερή. [10]

3. Mobile OS

3.1 iOS

Εικόνα 3 iOS

Το iOS (γνωστό και ως iPhone OS) είναι το λειτουργικό σύστημα για κινητές πλατφόρμες της Apple. Αν και αρχικά αναπτύχθηκε μόνο για το iPhone έχει από τότε επεκταθεί ώστε να υποστηρίζει και άλλες συσκευές της Apple όπως τα iPod Touch και iPad. Το συγκεκριμένο λειτουργικό σύστημα δεν υποστηρίζει άλλες συσκευές εκτός από αυτές της Apple. Ένα από τα μεγάλα πλεονεκτήματα του είναι το App Store το οποίο περιέχει περισσότερες από 500.000 εφαρμογές σύμφωνα με την τελευταία μέτρηση που έχει γίνει στα τέλη Μαΐου του 2011. Στο τελευταίο τετράμηνο του 2010 το iOS κατείχε το 16% της αγοράς των smartphones πίσω από το Google Android και το Nokia Symbian. [10]

3.1.1 Εκδόσεις

Οι εκδόσεις χαρακτηρίζονται από ένα αριθμό ξεκινώντας από την 1.0. Η τελευταία είναι η 4 (με συνεχείς αναβαθμίσεις σε 4.X).

3.1.2 Απόδοση

Το iOS είναι γρήγορο και σταθερό.

3.1.3 Οθόνη

Χρησιμοποιεί την οθόνη αφής σαν input device. Για την εισαγωγή κειμένου, χρησιμοποιείται ένα virtual πληκτρολόγιο πάνω στην οθόνη.

3.1.4 Περιβάλλον

Το περιβάλλον (μενού κλπ) είναι σε μεγάλο βαθμό προσαρμόσιμο από το χρήστη. Ευχάριστο, εύκολο, φιλικό, εργονομικό.

3.1.5 Εφαρμογές

Το iOS επιτρέπει την εγκατάσταση εφαρμογών. Αυτές είναι ελεγχόμενες από την ίδια την Apple. Αυτό σημαίνει ότι δύσκολα θα βρείτε άκρη χωρίς να περάσετε από την Apple, αλλά μπορείτε να στηριχτείτε στην ασφάλεια και αξιοπιστία της εταιρίας (η λογική που ισχύει και για τους υπολογιστές Mac). Υπάρχουν πάρα πολλές εφαρμογές κάθε είδους (εκτιμώνται πάνω από 250.000). Βασικότερη πηγή είναι το iTunes App Store της Apple.

3.1.6 Ασφάλεια

Το iOS θεωρείται γενικά "ασφαλές", κυρίως λόγω του "στενού ελέγχου" των εφαρμογών από την Apple. Θεωρητικά όμως θα μπορούσε να είναι ευάλωτο σε κακόβουλο λογισμικό, ειδικά στις περιπτώσεις που έχει γίνει "επέμβαση" στο λειτουργικό ("σπασμένο" - "hacked"). Εμείς προτείνουμε πάντα να μην εγκαθιστά κανείς εφαρμογές από μη αξιόπιστες πηγές.

3.1.7 Συμπέρασμα

Το iOS είναι η μόδα και κοιτά με αισιοδοξία το μέλλον. Έχει κερδίσει ένα αρκετά υψηλό μερίδιο αγοράς και διεκδικεί ακόμα υψηλότερο. Είναι μοντέρνο, αξιόπιστο, σταθερό και προσφέρει ένα πλήθος από εφαρμογές. Στηρίζεται και ελέγχεται αποκλειστικά από την Apple (με τα καλά και τα κακά αυτού).

3.2 Windows Mobile

Εικόνα 4 Windows Mobile

Τον Φεβρουάριο του 2010, η Microsoft ανακοίνωσε τον διάδοχο των Windows Mobile, την νέα γενιά λειτουργικών συστημάτων για κινητές πλατφόρμες, τα Windows Phone 7. Το νέο λειτουργικό σύστημα περιλαμβάνει ένα εντελώς νέο περιβάλλον χρήσης το οποίο έχει δημιουργηθεί με μια γλώσσα σχεδίασης της ίδιας της εταιρίας, που ονομάζεται Metro.

3.2.1 Εκδόσεις

Οι εκδόσεις χαρακτηρίζονται από ένα αριθμό. Η τελευταία ήταν η Windows Mobile 6.5.5, ενώ στα τέλη του 2010 κυκλοφόρησε και η Windows Phone 7 (που έχει πάρει πολύ καλά σχόλια).

3.2.2 Απόδοση

Είναι αρκετά βαρύ, με υψηλές απαιτήσεις σε hardware. Αυτό σημαίνει ότι μπορεί να είναι κάπως αργό σε συσκευές χωρίς επαρκή ισχύ.

3.2.3 Οθόνη

Χρησιμοποιεί την οθόνη αφής σαν input device.

3.2.4 Περιβάλλον

Το περιβάλλον θυμίζει τα κλασικά Windows και είναι αρκετά οικείο για κάποιον που τα χρησιμοποιεί ήδη στον υπολογιστή του, δεν απαιτεί ιδιαίτερη εκπαίδευση και εξοικείωση. Πολύ καλή συμβατότητα με εφαρμογές της Microsoft σε σταθερό υπολογιστή (πχ Outlook).

3.2.5 Εφαρμογές

Επιτρέπει την εγκατάσταση εφαρμογών. Όμως, η ποικιλία είναι πολύ μικρότερη από εκείνη των Android και iOS (εκτιμούνται γύρω στις 5.000). Πάντως, η Microsoft κάνει ότι μπορεί

για αυτό, προσαρμόζοντας συνεχώς εφαρμογές (π.χ. παιχνίδια φτιαγμένα αρχικά για το x-box).

3.2.6 Ασφάλεια

Όπως και τα κλασικά Windows, είναι ευάλωτα σε κακόβουλο λογισμικό, ειδικά στις περιπτώσεις που έχει γίνει "επέμβαση" στο λειτουργικό ("σπασμένο" - "hacked"). Κυκλοφορούν mobile εκδόσεις εφαρμογών προστασίας. Εμείς προτείνουμε πάντα να μην εγκαθιστά κανείς εφαρμογές από μη αξιόπιστες πηγές.

3.2.7 Συμπέρασμα

Τα Windows Mobile δεν απολαμβάνουν την αποδοχή των κλασικών Windows και δυσκολεύονται να ανταγωνιστούν τα iOS της Apple και Android της Google. Η Microsoft κάνει ότι μπορεί, αλλά πολλοί αναλυτές θεωρούν ότι, με τα σημερινά δεδομένα, είναι ένας χαμένος πόλεμος. Το μερίδιο αγοράς τους είναι χαμηλό και δεν δείχνει διάθεση να αυξηθεί. Παρόλα αυτά, παραμένει ένα αξιόλογο λειτουργικό, που πολλοί θα επέλεγαν. Ίσως, μεγαλύτερο πλεονέκτημά του είναι η συγγένεια με τα Windows που οι περισσότεροι χρησιμοποιούν καθημερινά και το παρόμοιο περιβάλλον...

3.3 Symbian

Εικόνα 5 symbian

Το Symbian OS είναι λειτουργικό σύστημα για φορητές συσκευές, αποτελεί εξέλιξη του λειτουργικού συστήματος EPOC από την Psion. Το Symbian OS δημιουργήθηκε με τη γλώσσα προγραμματισμού C++ από τη Symbian Ltd. Πριν το 2009 το Symbian OS υποστήριζε διαφορετικά περιβάλλοντα χρήστη. Όμως με την δημιουργία του Symbian Platform, το ίδιο έτος, τα 3 βασικά περιβάλλοντα χρήστη ενώθηκαν σε ένα, το οποίο εξαγοράστηκε από την Nokia και στην συνεχεία μετατράπηκε σε λογισμικό ανοικτού κώδικα. Αν και οι συσκευές με λογισμικό Symbian εξακολουθούν να πωλούνται σε μεγάλους αριθμούς στην αγορά, τα τελευταία χρόνια το μερίδιο του λειτουργικού αυτού συστήματος στην αγορά μειώνεται. Για την ανάπτυξη εφαρμογών στο περιβάλλον του λειτουργικού υπάρχει το Symbian SDK το οποίο χρησιμοποιεί ως γλώσσα προγραμματισμού την C++ σε συνδυασμό με το Qt, ένα Framework εφαρμογών που χρησιμοποιείται από πολλές πλατφόρμες. Μπορεί να χρησιμοποιηθεί είτε με το Qt Creator είτε με το Carbide, ένα παλιότερο IDE που χρησιμοποιείται για ανάπτυξη εφαρμογών Symbian. Ένας εξομοιωτής χρησιμοποιείται, για τη δοκιμή των εφαρμογών, που τρέχει τον κώδικα απευθείας αντί να προσομοιώνει την λειτουργία του κινητού τηλεφώνου. [10]

3.3.1 Εκδόσεις

Εδώ τα πράγματα είναι λίγο μπερδεμένα. Θα ακούσετε για εκδόσεις του λειτουργικού (π.χ. 9.3, 9.4) και της πλατφόρμας (π.χ. UIQ 1-2-3, series 60-80-90 edition 1-2-3-5, ή Symbian^1-^2-^3). Οι περισσότερες σημερινές συσκευές χρησιμοποιούν series 60 5th edition (S60-e5, το αντίστοιχο του Symbian^1) ή Symbian^3 (η πιο προχωρημένη έκδοση).

3.3.2 Απόδοση

Το Symbian είναι γρήγορο και σταθερό, κάνει καλή διαχείριση του hardware και είναι αρκετά οικονομικό (κατανάλωση μπαταρίας).

3.3.3 Οθόνη

Οι μοντέρνες εκδόσεις χρησιμοποιούν την οθόνη αφής σαν βασικό input device, ενώ λίγες συσκευές μπορεί να έχουν και πληκτρολόγιο. Το μέγεθος και η ανάλυση της οθόνης εξαρτάται από το μοντέλο της συσκευής.

3.3.4 Περιβάλλον

Το περιβάλλον (μενού κλπ) είναι σε μεγάλο βαθμό προσαρμόσιμο από το χρήστη. Αρκετά φιλικό και εργονομικό, αν και, για πολλούς, ελαφρά ξεπερασμένο.

3.3.5 Εφαρμογές

Το Symbian επιτρέπει την εγκατάσταση εφαρμογών. Αυτές θα τις βρείτε από πολλές διαφορετικές πηγές (sites στο internet) ή από το Ovi Store της Nokia. Είναι αρκετές (~15.000), από δωρεάν ως φθηνές. Καλύπτουν πολλές ανάγκες, αν και δεν πλησιάζουν καν την ποικιλία των αντίστοιχων που κυκλοφορούν για Android ή iPhone.

3.3.6 Ασφάλεια

Το Symbian είναι ευάλωτο σε ιούς - trojans και γενικά κακόβουλο λογισμικό, αν και όχι στο βαθμό των PC. Στις περισσότερες περιπτώσεις είναι με τη μορφή εφαρμογών (που ζητούν την άδεια του χρήστη για να εγκατασταθούν), άρα μπορούμε να πούμε ότι ένας προσεκτικός χρήστης που δεν εγκαθιστά εφαρμογές από μη αξιόπιστες πηγές, είναι σχετικά ασφαλής. Πάντως, κυκλοφορούν και τα αντίστοιχα λογισμικά προστασίας.

3.3.7 Συμπέρασμα

Το Symbian φαίνεται πλέον να γίνεται συνώνυμο της Nokia. Είναι αξιόπιστο και σταθερό. Τα προηγούμενα χρόνια υπήρξε leader στα smartphones και διατηρεί την αποτελεσματικότητά του, αλλά σταδιακά χάνει τη λάμψη του και το μερίδιο αγοράς του, όσο ανεβαίνουν τα νεότερα λειτουργικά της Google (Android) και Apple (iOS του iPhone). Παραμένει ένα πολύ καλό λειτουργικό, αν όμως σας ενδιαφέρει να εγκαταστήσετε πολλές και "περίεργες" εφαρμογές, δεν θα βρείτε εδώ την ποικιλία των Android και iPhone.

3.4 Android

Εικόνα 6 Android

Ένα σχετικά πρόσφατο, ανοιχτό λειτουργικό, από εταιρία που ανήκει στη Google και βασισμένο σε πλατφόρμα Linux! Αναπτύσσεται ραγδαία, κερδίζοντας μερίδιο αγοράς και μπαίνοντας σε συσκευές πολλών εταιριών. Πολλοί "ειδικοί" θεωρούν ότι "είναι το μέλλον".[10]

3.4.1 Εκδόσεις

Οι εκδόσεις χαρακτηρίζονται από ένα αριθμό και ένα αντίστοιχο όνομα: 1.1, 1.5(Cupcake), 1.6(Donut), 2.0-2.1(Eclair), 2.2(Froyo), 2.3(Gingerbread), 3.0(Honeycomb), 4.0(Ice Scream)... Κάθε έκδοση έχει και διαφορετικά χαρακτηριστικά και δυνατότητες. Οι σημερινές συσκευές συνήθως τρέχουν εκδόσεις μεταξύ 1.6 και 2.2.

3.4.2 Απόδοση

Το Android είναι γρήγορο και σταθερό.

3.4.3 Οθόνη

Χρησιμοποιεί την οθόνη αφής σαν βασικό input device, ενώ λίγες συσκευές μπορεί να έχουν και qwerty πληκτρολόγιο. Το μέγεθος και η ανάλυση της οθόνης εξαρτάται από το μοντέλο της συσκευής. Πάντως, αν γράφετε συχνά και χρησιμοποιείτε εφαρμογές internet, προτιμήστε μια μεγαλούτσικη οθόνη.

3.4.4 Περιβάλλον

Το περιβάλλον (μενού κλπ) είναι σε μεγάλο βαθμό προσαρμόσιμο από το χρήστη. Αρκετά φιλικό και εργονομικό, ενώ σε κάθε νέα έκδοση εξελίσσεται. Ανάλογα με τον κατασκευαστή της συσκευής, μπορεί να υπάρχουν διαφορές.

3.4.5 Εφαρμογές

Το Android επιτρέπει την εγκατάσταση εφαρμογών και διαθέτει μια μεγάλη κοινότητα που προσφέρει πάρα πολλές εφαρμογές κάθε είδους (εκτιμούνται πάνω από 100.000). Βασικότερη πηγή είναι το Android Market (θα βρείτε ήδη εγκατεστημένο link στο Android τηλέφωνό σας). Οι περισσότερες είναι δωρεάν ή αρκετά φθηνές.

3.4.6 Ασφάλεια

Το Android θεωρείται σχετικά "ασφαλές", αν και θεωρητικά είναι ευάλωτο σε κακόβουλο λογισμικό. Αρκετοί ανησυχούν ότι σύντομα θα δούμε πολλά δυσάρεστα περιστατικά και ζητούν από τη Google να ελέγχει εξονυχιστικά όλες τις εφαρμογές του Android Market. Εμείς προτείνουμε πάντα να μην εγκαθιστά κανείς εφαρμογές από μη αξιόπιστες πηγές. Πάντως, ήδη έχουν εμφανιστεί τα πρώτα λογισμικά προστασίας.

3.4.7 Συμπέρασμα

Το Android φαίνεται να δείχνει το μέλλον, κόντρα στο αντίπαλο δέος της Apple και την παράδοση των Symbian και RIM-Blackberry. Αυξάνει θεαματικά το μερίδιο αγοράς του,

στοχεύοντας την κορυφή. Είναι μοντέρνο, αξιόπιστο, σταθερό και προσφέρει ένα πλήθος από εφαρμογές. Και μόνο το γεγονός ότι στηρίζεται από μια εταιρία όπως η Google, υποχρεώνει τον ανταγωνισμό να το πάρει πραγματικά σοβαρά... [10]

Ποιο αναλυτικά θα αναφερθούμε στο επόμενο κεφάλαιο.

3.5 BlackBerry OS

Εικόνα 7 BlackBerry

Παλιό, κλασικό λειτουργικό, πιο "σοβαρό", απευθύνεται κυρίως σε επαγγελματίες χρήστες. Είναι το κορυφαίο σε διαχείριση e-mail και πολύ δυνατό σε ότι έχει να κάνει με internet, downloads και συνδεσιμότητα. Δέχεται εξωτερικές εφαρμογές (Blackberry App World). Το περιβάλλον του δεν στηρίζεται στο fun και την ευκολία χρήσης, αλλά στην αποτελεσματικότητα, κάτι που για κάποιους το κάνει κάπως "δύσχρηστο". Επίσης, η "υπηρεσία BlackBerry" χρεώνεται από τους παρόχους κινητής τηλεφωνίας με επιπλέον πάγιο. Έχει ένα σεβαστό μερίδιο αγοράς, από ένα συγκεκριμένο κοινό.

4. Android

Στον κόσμο της τεχνολογίας, όπου τα αμύθητα ποσά και οι τεχνολογικές καινοτομίες διαδέχονται η μία την άλλη με ταχύτατους ρυθμούς, οι κολοσσοί διαρκώς μάχονται μεταξύ τους: Apple vs Microsoft, Google vs Apple vs Samsung, και πάει λέγοντας. Άλλος κερδίζει, άλλος χάνει. Και το έπαθλο; …Ο χρήστης!

Πώς θα μπορούσαμε να ερμηνεύσουμε αλλιώς το γεγονός ότι ο εμπνευστής και δημιουργός του Android, του λειτουργικού συστήματος για κινητά τηλέφωνα που βασίζεται στον πυρήνα του Linux, που έχει τη σφραγίδα της Google, έκανε τα πρώτα του επαγγελματικά του βήματα στην Apple

4.1 Η ιστορία

Τα πράγματα ξεκίνησαν όταν ο ευφυής Andy Rubin θέλησε την Άνοιξη του 2005 να χρησιμοποιήσει την Google ως κατ' εξοχήν μηχανή αναζήτησης για το T-Mobile Sidekick, μια φερέλπιδα συσκευή κινητού, την οποία είχε αναπτύξει με την ομάδα συνεργατών του. Εν συνεχεία, ζήτησε να συναντηθεί με τον Larry Page, ο οποίος είναι ο ένας από τους δύο ιδρυτές της Google. Σε αυτήν τη συνάντηση ο Rubin παρουσίασε το Android ως ένα εν δυνάμει παγκόσμιο ανοικτό λειτουργικό σύστημα που θα άλλαζε για πάντα τον τρόπο που διαντιδρούνε οι χρήστες με το κινητό σας, τονίζοντας, ταυτόχρονα, τη σταθερή υπεροχή που παρατηρείται στις συνήθειες του αγοραστικού κοινού των κινητών τηλεφώνων, σε αντιδιαστολή με τις πωλήσεις ηλεκτρονικών υπολογιστών.

Την ίδια στιγμή, ο Page δεν ήθελε να γίνει απλώς ο υποστηρικτής του Android, ήθελε να γίνει ο ιδιοκτήτης του. Ο Andy… βρήκε «την καλή», την ώρα που ένας ισχυρός παίκτης εμφανίστηκε στο προσκήνιο και έθεσε έτσι τους όρους του ανταγωνισμού σε άλλο επίπεδο. Ο καινούργιος «παίκτης» δεν είναι άλλος από εκείνον που τελικά λάνσαρε το καλοκαίρι του 2005, το iPhone της Apple.

Ο επιχειρηματικός-τεχνολογικός κόσμος περίμενε πως η Google θα απαντούσε με ένα gPhone, αλλά αυτό δεν έγινε, διότι έγινε κάτι άλλο, πολύ σημαντικότερο. Το Φθινόπωρο του 2005 ανακοινώνεται ότι 34 εταιρίες, όπως η Texas Instruments, η Intel, η T-Mobile και η Sprint Nextel, ενώνουν τις δυνάμεις τους με την Google για τη δημιουργία μιας πλατφόρμας ανοιχτού κώδικα που θα έχει ενσωματωμένο το λογισμικό Linux και θα εκπροσωπείται από μια νέα συστάδα εταιριών που θα καλείται Open Handset Alliance.

Στο χορό δεν άργησαν να μπουν και άλλες εταιρείες, όπως η HTC, η Motorola και η LG, ανακοινώνοντας την πρόθεσή τους να δώσουν προς πώληση στην αγορά smartphones με λειτουργικό σύστημα Android σε διάφορα σχήματα και μεγέθη, με τα οποία θα μπορεί να έχει ο χρήστης να ενσωματώνει στο κινητό του πλήθος εφαρμογών.

4.2 Τι πρέπει να γνωρίζουμε για το Android

Το Android είναι ένα λειτουργικό σύστημα που ενσωματώνεται σε συσκευές κινητής τηλεφωνίας, τα οποία διαθέτουν οθόνη αφής, τρέχουν τον πυρήνα (kernel) του λειτουργικού Linux και ακόμη, επιτρέπει στους κατασκευαστές λογισμικού να συνθέτουν κώδικα με τη χρήση της γλώσσας προγραμματισμού Java, ελέγχοντας τη συσκευή μέσω βιβλιοθηκών λογισμικού της Google.

Συσκευές με Android υπάρχουν πλέον πάρα πολλές, η καθεμία με διαφορετικά χαρακτηριστικά και από διάφορες κατασκευάστριες εταιρίες: η LG, Samsung, HTC, Sony Ericsson, Motorola, είναι μερικές από τις εταιρίες που χρησιμοποιούν το λειτουργικό Android για τα smartphones τους.

Το πολύ θετικό με τις συσκευές Android είναι ότι είναι αφενός multimedia (σας δίνουν τη δυνατότητα να αναπαράγετε πολλαπλά μέσα) και multitasking (δίνουν τη δυνατότητα εκτέλεσης πολλών εφαρμογών ταυτόχρονα, π.χ. ακούτε τραγούδια ενώ σερφάρετε στο ίντερνετ και ταυτόχρονα απαντάτε σε ένα SMS χωρίς να κλείσετε καμία εφαρμογή ή να χάσετε τη σελίδα που επισκεφτήκατε).

Το web browsing στο Android είναι ταχύτατο, υποστηρίζεται από flash και υπάρχουν πολλοί browsers για να καλύψουν και τους πλέον απαιτητικούς.

Ανεξάρτητα από το κόστος, όλες οι συσκευές Android διαθέτουν GPS και Wi-fi, δικαιώνοντας έτσι το βασικό λόγο δημιουργίας του εν λόγω λειτουργικού συστήματος που δεν είναι άλλος παρά η ανεμπόδιστη και εύκολη πρόσβαση στο διαδίκτυο, σε συνδυασμό με ένα πλήθος εφαρμογών (apps), όπως χάρτες, αναζήτηση, chat και e-mail, που πραγματικά επιτρέπουν στο χρήστη να μένει διαρκώς δικτυωμένος και ενημερωμένος.

Βασικό χαρακτηριστικό του Android, επίσης, είναι η πληθώρα εφαρμογών που διατηρούν τη συνεχή σύνδεση με Facebook, MySpace, Twitter και δεκάδες άλλες υπηρεσίες social networking. Ακόμη, το Android σας δίνει τη δυνατότητα να προσθέσετε widgets, δηλαδή εικονίδια για την ταχύτερη πρόσβαση στα προγράμματα, τα οποία τοποθετούνται στη home screen του κινητού (launcher).

Επιπλέον η notification bar είναι εξαιρετικά χρήσιμη, καθώς με ένα απλό drag βλέπετε όλες τις ειδοποιήσεις για τη συσκευή σας, αλλά και τα προγράμματα (applications) που έχετε εγκαταστήσει.

Όσον αφορά το hardware, οι διπύρηνοι επεξεργαστές και οι διακεκριμένες GPU είναι πλέον γεγονός, ενώ αναμένουμε και επεξεργαστές τεσσάρων πυρήνων, καθώς αυτό έχει ήδη

ανακοινωθεί από την Nvidia με τον επεξεργαστή Kal – El ο οποίος μάλιστα θα περιέχει και έναν πέμπτο –stealth- πυρήνα. [1]

4.3 Google Play

Το Google Play είναι ένα νέο ενιαίο κατάστημα που συγκεντρώνει όλη τη μουσική, τα videos/ταινίες, τις εφαρμογές και τα eBooks σε ένα μέρος, αντικαθιστώντας το Android Market. Όλες οι υπηρεσίες της Google μετονομάζονται και μπαίνουν κάτω από την "ομπρέλα" του Google Play (π.χ. Google Music -> Play Music κλπ.), μια υπηρεσία που είναι προσβάσιμη από οποιονδήποτε υπολογιστή, Android smartphone ή Android tablet, με την αποθήκευση των αρχείων του χρήστη να γίνεται σε cloud και φυσικά να είναι διαθέσιμα κάθε στιγμή από οπουδήποτε. Οι χρήστες μπορούν να αποθηκεύουν δωρεάν μέχρι και 20.000 τραγούδια, ενώ έχουν πρόσβαση σε εκατομμύρια κομμάτια, σε περισσότερες από 450.000 εφαρμογές και παιχνίδια Android, στην τεράστια ηλεκτρονική βιβλιοθήκη και σε χιλιάδες ταινίες, με πολλές νέες κυκλοφορίες σε ποιότητα HD.

Για να πούμε και λίγα λόγια για τον προκάτοχο του Google Play το Android Market, είναι ένα online κατάστημα της Google, προσφέρει σε κάθε χρήστη εφαρμογές για το κινητό τους που είναι συμβατό με το λειτουργικό της Google. Το συντριπτικό ποσοστό των εφαρμογών είναι δωρεάν ενώ πλέον οι πληρωμένες εφαρμογές είναι διαθέσιμες και στο ελληνικό κοινό.

4.4 Εκδόσεις Android

Το Android που κυκλοφορεί διάφορες εκδόσεις με ονομασίες που σου ανοίγουν την όρεξη για… νέα χαρακτηριστικά, όπως τα παλαιότερα CupCake (1.5), Donut (1.6), Éclair (2.0, 2.1), GingerBread (2.3) αλλά και FroYo (2.2) Honeycomb (3.0) που υλοποιείται σε ταμπλέτες ενώ υπάρχουν πλέον και οι εκδόσεις Honeycomb (3.1) και Honeycomb (3.2) και Ice Cream Sandwich(4.0).

Από την «παρθενική» έκδοση Android 1.0, η οποία κυκλοφόρησε το Σεπτέμβριο του 2008, μέχρι την αμέσως επόμενη, 1.1 που παρουσιάστηκε το Φεβρουάριο του 2009, χρειάστηκε ένας χρόνος για να γίνει η έκρηξη των καινοτόμων εκδόσεων και των σημαντικών αλλαγών που επέφεραν για τον χρήστη.

Το πρώτο smartphone που «έτρεξε» Android είναι το T-Mobile G1 κατασκευασμένο από την HTC με οθόνη αφής TFT-LCD 3,2", full qwerty πληκτρολόγιο, πρόσβαση σε Gmail, YouTube, Google maps, Google talk, Google calendar, κάμερα 3,2MP με αυτόματη εστίαση και κάρτα μνήμης micro SD.

4.4.1 CupCake (1.5)

Εικόνα 8 CupCake

Το Cupcake (1.5) εισάγει κάποια καινούργια χαρακτηριστικά και ανανεώσεις στην διεπιφάνεια χρήστη (User Interface):

- Ικανότητα για καταγραφή και παρακολούθηση βίντεο μέσα από την λειτουργία της βιντεοκάμερας, μεταφόρτωση βίντεο στο YouTube και φωτογραφιών στο Picasa απευθείας από το τηλέφωνο, καινούργιο μαλακό πληκτρολόγιο (αφής) με πρόβλεψη κειμένου
- Υποστήριξη προτύπου Bluetooth A2DP και AVRCP
- Ικανότητα αυτόματης σύνδεσης σε μικροσυσκευή Bluetooth από μια συγκεκριμένη απόσταση
- Καινούργια widgets και φάκελοι που μπορούν να δημοσιευτούν στην αρχική οθόνη
- Κινούμενες μεταβάσεις οθόνης

4.4.2 Donut (1.6)

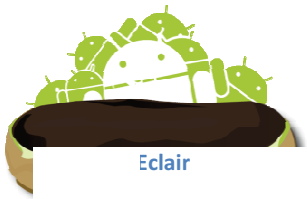
Εικόνα 9 Donut

Το 'Donut', Android 1.6, ήρθε τον Σεπτέμβριο του 2009.

Η έκδοση αυτή εισάγει κάποια καινούργια χαρακτηριστικά όπως:

- Βελτιωμένο Android Market
- Ενσωματωμένη φωτογραφική μηχανή, βιντεοκάμερα και διεπαφή (interface) γκαλερί
- Η γκαλερί επιτρέπει πλέον στους χρήστες την επιλογή πολλαπλών φωτογραφιών για διαγραφή
- Ανανεωμένη αναζήτηση με φωνή, με ταχύτερη απόκριση και βαθύτερη ολοκλήρωση με εγγενής (native) εφαρμογές, συμπεριλαμβανομένης της δυνατότητας να καλούμε επαφές
- Ανανεωμένη αναζήτηση με την δυνατότητα αναζήτησης σελιδοδεικτών, ιστορικού, επαφών και στο διαδίκτυο από την αρχική οθόνη
- Ανανεωμένη υποστήριξη τεχνολογιών για CDMA/EVDO, 802.1x, VPNs και με μηχανή μετατροπής κειμένου σε ομιλία (text-to-speech)
- Υποστήριξη για ανάλυση οθονών WVGA
- Βελτιώσεις στην ταχύτητα για αναζήτηση και για εφαρμογές φωτογραφικής μηχανής

4.4.3 Éclair (2.0, 2.1)



Eclair

Ακολουθεί το 'Eclair', Android 2.0 τον Νοέμβριο 2009, με τις επανεκδόσεις του σε Android 2.0.1 τον Δεκέμβριο 2009 (Eclair 0.1) και τον Ιανουάριο 2010 με το Android 2.1 (Eclair MR1).

Ανάμεσα στις άλλες αλλαγές είναι και:

- Βέλτιστη ταχύτητα υλικού
- Υποστήριξη για περισσότερες οθόνες και αναλύσεις
- Βελτιωμένη διεπιφάνεια χρήστη
- Καινούργια διεπιφάνεια χρήσης για την μηχανή αναζήτησης και υποστήριξη του προτύπου HTML5
- Καινούργιες λίστες επαφών
- Καλύτερος λόγος άσπρου – μαύρου για φόντα

- Βελτιωμένοι χάρτες Google (google maps) 3.1.2
- Υποστήριξη Microsoft Exchange
- Ενσωματωμένη υποστήριξη flash για την Camera
- Ψηφιακή μεγέθυνση (zoom)
- Κλάση MotionEvent βελτιωμένη ώστε οι κατασκευαστές να μπορούν να παρακολουθούν αποτελεσματικότερα τα γεγονότα πολλαπλής αφής
- Ανανεωμένο εικονικό πληκτρολόγιο
- Bluetooth 2.1

4.4.4 FroYo (2.2)

Εικόνα 11 FroYo

Ακολουθεί το Android 2.2 με το όνομα 'Froyo' τον Μάιο του 2010

Η έκδοση FROYO ανάμεσα σε άλλες αλλαγές περιλαμβάνει:

- Βελτιστοποιήσεις στην ταχύτητα γενικά του λειτουργικού συστήματος, στην μνήμη και στην απόδοση
- Ενσωμάτωση στην μηχανή αναζήτησης, της μηχανής Javascript του Chrome V8
- Αυξημένη υποστήριξη Microsoft Exchange (σε πολιτικές ασφαλείας, συγχρονισμού ημερολογίου, auto – discovery, GAL look-up, remote wipe)
- Βελτιωμένος προωθητής εφαρμογής (application launcher), με συντομεύσεις προς τις εφαρμογές τηλεφώνου και εφαρμογές της Μηχανής Αναζήτησης
- Πρόσδεση USB και λειτουργία δυναμικής ζώνης (hotspot) WiFi
- Ανανεωμένη εφαρμογή Αγοράς (Market) με αυτόματη ανανέωση
- Επιλογή για απαγόρευση πρόσβασης δεδομένων πάνω από ένα δίκτυο κινητής τηλεφωνίας
- Γρήγορη εναλλαγή ανάμεσα σε πολλαπλές γλώσσες του πληκτρολογίου και των λεξικών τους
- Φωνητική κλήση και διαμοιρασμός επαφών με Bluetooth
- Υποστήριξη για αριθμητικούς και αλφαριθμητικούς κωδικούς
- Η μηχανή αναζήτησης μπορεί να αποτυπώσει κινούμενα GIFs
- Υποστήριξη για πεδία μεταφόρτωσης αρχείων στην μηχανή αναζήτησης
- Υποστήριξη για εγκατάσταση εφαρμογών στην επεκτάσιμη μνήμη
- Υποστήριξη Adobe Flash 10.1

4.4.5 GingerBread (2.3)

Εικόνα 12 GingerBread

Η τρέχουσα έκδοση για κινητά smartphones, από τον Δεκέμβριο του 2010 μ στιγμή που γράφεται αυτή η εργασία, είναι η έκδοση Android 2.3 με το όνομα "Gingerbread" με την επανέκδοση του σε Android 2.3.3 τον Φεβρουάριο του 2011

Οι αλλαγές που έχουν γίνει είναι οι ακόλουθες:

- Βελτιωμένο UI για απλότητα και ταχύτητα
- Πιο γρήγορη, πιο διαισθητική εισαγωγή κειμένου
- Επιλογή λέξεων και αντιγραφή/επικόλληση με ένα άγγιγμα
- Βελτιωμένη ενεργειακή διαχείριση
- υποστήριξη NFC (Near Field Communication)
- Υποστήριξη video κλήσης
- Υποστήριξη του πρωτόκολλου WebM για αναπαραγωγή video

Εικόνα 13 Honeycomb

4.4.6 Honeycomb (3.1, 3.2)

Η πιο πρόσφατη έκδοση, είναι η έκδοση Android 3.0 με το όνομα “Honeycomb” στην διάθεση των χρηστών και προγραμματιστών από τον Φεβρουάριο του 2011, λίγες μέρες μετά την επανέκδοση του Android 2.3.3, και προορίζεται αποκλειστικά για ταμπλέτες, για τις οποίες να αναφερθούμε αναλυτικότερα στην συνέχεια.

Μερικά από τα χαρακτηριστικά του είναι:

- Υποστηρίζει διπύρηνους και τετραπύρηνους επεξεργαστές
- Βελτιωμένη υποστήριξη των ταμπλετών
- ανάπτυξη λογισμικού (scripting) για 3D, σε γλώσσα η οποία καλείται "Renderscript"
- Video chat μέσω Google Talk
- Google eBooks
- "Ιδιωτική περιήγηση"

Εικόνα 14 Ice Cream Sandwich

4.4.7 Ice Cream Sandwich (4.0)

Και τέλος φτάνουμε στην νεότερη έκδοση μας 4.0 Ice Cream Sandwich. Τα νέα χαρακτηριστικά του Android 4.0 Ice Cream είναι άκρως εντυπωσιακά και τα πρώτα σχόλια που δέχεται παγκοσμίως το λογισμικό είναι εγκωμιαστικά. Κατεγράψα τα πιο σημαντικά και σας παρουσιάζω αναλυτικά τους λόγους για τους οποίους αυτό είναι το καλύτερο OS. Το Android για πάρα πολλούς χρήστες κινητών ήταν ήδη το καλύτερο λειτουργικό σύστημα της κατηγορίας. Με το χθεσινό "μεγάλο" update, η Google φαίνεται να καταφέρνει να πείσει ακόμα και τους πιο δύσπιστους για την υπεροχή του Android. Πάμε να εξερευνήσουμε σιγά σιγά τα νέα χαρακτηριστικά του Android 4.0 Ice Cream Sandwich. Η Google σε αυτή την αναβάθμιση δεν έμεινε μόνο σε αυτό που περίμεναν αρκετοί, δηλαδή στην αναβάθμιση του γραφικού περιβάλλοντος και στην ενοποίηση του Gingerbread με το Honeycomb. Αντίθετα άλλαξε σχεδόν τα πάντα, όπως για παράδειγμα όλες τις δικές της Android εφαρμογές.

Γενικά στοιχεία για το Android 4.0 Ice Cream Sandwich

Η Google εισήγαγε την νέα γραμματοσειρά Roboto, η οποία σχεδιάστηκε συνδυάζοντας στυλ από Web Design, Typography και TV. Η περιήγηση ανάμεσα στις αρχικές οθόνες γίνεται ακόμα με τον παραδοσιακό τρόπο, αλλά στο μενού των εφαρμογών προστέθηκαν επίσης tabs. Επίσης πατώντας τα “Volume Down” + “Power” μπορούμε να πάρουμε screenshot της οθόνης, ενώ το Software επιτρέπει την σύλληψη φωτογραφιών με μηδενικό lag.

Πληκτρολόγιο

Το πληκτρολόγιο είναι ανασχεδιασμένο ώστε να παρέχει πιο άμεση διόρθωση λέξεων και καλύτερο Copy, Cut και Paste. Οι αλλαγές που έγιναν βοηθάνε στην πιο γρήγορη πληκτρολόγηση, ενώ ακόμα καλύτερο είναι το voice-to-text που επιτρέπει πλέον να μιλάμε με παύσεις ανάμεσα στις λέξεις. (πολύ πιο βολικό).

Ειδοποιήσεις

Όπως πάντα οι ειδοποιήσεις κατεβαίνουν από το πάνω μέρος της οθόνης, αλλά πλέον εμφανίζουν τις φωτογραφίες αυτών που μας καλούν ή μας στέλνουν κάποιο μήνυμα. Η μπάρα εμφανίζεται πλέον ακόμα και στην οθόνη κλειδώματος, ενώ οι ειδοποιήσεις θα μπορούν να σβηστούν με ένα απλό swype προς τα δεξιά ή αριστερά.

Browser

Ο νέος browser του Android 4.0 ICS μάλλον είναι προσωρινός, αφού ο Chrome θα πάρει τη θέση του στα κινητά μας σε λίγους μήνες. Παρ' όλα αυτά, θα μπορείτε πλέον στον Android browser να σώσετε μια σελίδα για offline διάβασμα με ένα κλικ, ενώ ταυτόχρονα θα μπορείτε να έχετε ανοικτά μέχρι 16 tabs. Σημαντική είναι επίσης η προσθήκη συγχρονισμού των bookmarks από τον dekstop Chrome και η δυνατότητα εναλλαγής από mobile σε desktop περιβάλλον των σελίδων μέσα από το ειδικό μενού.

Face Unlock

Κάτι που περιμέναμε χρόνια, έκανε πράξη η Google, φέρνοντας με το Android 4.0 την αναγνώριση προσώπου στα κινητά μας. Ξεχάστε τους κωδικούς και τα gestures, η πιο καλή ασφάλεια είναι το πρόσωπο σας.

Gmail

Η αναβάθμιση του Gmail είναι πολύ μεγάλη, αφού η εφαρμογή εισάγει νέο τρόπο εμφάνισης και νέες λειτουργίες. Στο κάτω μέρος της οθόνης θα υπάρχει πάντα μια μπάρα με τις πιο συνηθισμένες επιλογές, ενώ θα μπορείτε απ' ευθείας από την εφαρμογή να αλληλεπιδράσετε με τις επαφές σας.

Ημερολόγιο

Και εδώ έχουμε καινούργιο design, αλλά αυτό που εντυπωσιάζει είναι η προσθήκη Pinch to Zoom!

Data Usage

Η Google ενσωμάτωσε στο ICS τη δική της εφαρμογή για την διαχείριση των δεδομένων από/προς το διαδίκτυο. Μπορείτε να δείτε με έξυπνο τρόπο πόσα δεδομένα στέλνετε και από ποιες εφαρμογές και να προσθέσετε κάποια επιτρεπτά όρια.

Φωτογραφίες, Video & Gallery

Μια ακόμα αναβάθμιση αφορά τον τρόπο με το οποίο μπορούμε να μοιραζόμαστε τις φωτογραφίες και τα βίντεο μας. Μετά τη λήψη μια φωτογραφίας/βίντεο, με ένα κλικ πάνω στη μικρογραφία της μας παρουσιάζεται ένα ειδικό μενού με όλους τους τρόπους διαμοιρασμού που είναι δυνατοί. Ταυτόχρονα η Google για πρώτη φορά εισάγει μια ενσωματωμένη εφαρμογή επεξεργασίας φωτογραφιών on-the-go. Δεν θα λέγαμε ότι είναι στη μορφή του Photoshop, αλλά πιο πολύ θυμίζει το δημοφιλές Instagram με επιλογές όπως περιστροφή, περικοπή, αφαίρεση κόκκινων ματιών κλπ. Μάλιστα, κάθε φορά που χρησιμοποιείται κάποια από αυτές τις επιλογές, η εφαρμογή αποθηκεύει αυτόματα μια προσωρινή έκδοση της εικόνας σας, ώστε να μην χάσετε κάτι από λάθος. Τέλος, προστέθηκε η δυνατότητα για λήψη πανοράματος, κάτι που γίνεται με πολύ εύκολα και γρήγορο τρόπο, ενώ τα αποτελέσματα που είδαμε είναι εξαιρετικά.

Η Συλλογή (gallery) άλλαξε τελείως, με τις εικόνες να εμφανίζονται σε μεγάλες μικρογραφίες, η μία δίπλα στην άλλη, σαν κολλάζ. Μπορείτε να τις "τακτοποιήσετε" με βάση την ημερομηνία ή το γεωγραφικό σημείο που καταγράφηκαν, ενώ μπορείτε να ενεργοποιήσετε τον αυτόματο συγχρονισμό τους με το cloud, μέσω φυσικά του Google+.

Στον τομέα του Video προστέθηκαν κάποια λίγα πραγματάκια, όπως το συνεχές zoom κατά τη διάρκεια της λήψης και η δυνατότητα για time lapse videography. (που σημαίνει ότι μπορείτε να επιταχύνεται ένα video 10 ωρών σε σταθερό τοπίο, χωρίς να φαίνεται σε fast forward, αλλά σαν εφέ ταινίας)

People

Η εφαρμογή contacts μετονομάστηκε σε People (αλά Windows Phone) και η λειτουργία της εισάγει νέα στοιχεία στο Android. Οι επαφές σας εμφανίζονται με μεγάλες φωτογραφίες και περιέχουν μια προβολή των τελευταίων κινήσεων κάθε προσώπου στα social networks.

Android Beam

Η τεχνολογία NFC πρωτοεμφανίστηκε από την Google, με το Nexus S. Η επιλογή αυτή, αν και τότε φαινόταν σαν κίνηση εντυπωσιασμού, σήμερα με το Android 4.0 Ice Cream Sandwich και το Android Beam, μοιάζει απόλυτα σχεδιασμένη. Με το Android Beam θα μπορείτε να μοιράζεστε κάθε δεδομένο σας με τα Android κινητά με NFC, με μια απλή επαφή των δύο συσκευών. Όλες οι εφαρμογές της Google είναι σχεδιασμένες για να

καταλαβαίνουν τους διάφορους τύπους δεδομένων που μοιράζεστε και να αλληλεπιδρούν με τον κατάλληλο τρόπο. Η επίδειξη του Beam ήταν αρκετά εντυπωσιακή και ελπίζουμε η πραγματική εφαρμογή του να είναι το ίδιο εύκολη.

4.5 Αρχιτεκτονική του Android

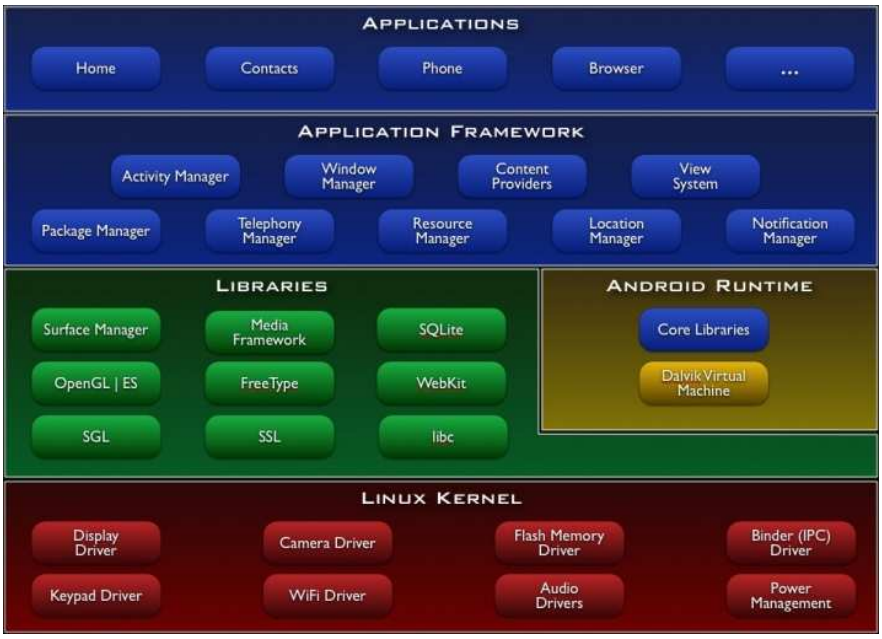
Όπως προανέφερα, το Android είναι μια στοίβα λογισμικού. Η λογική πίσω από αυτήν την έκφραση και σε όλη την φιλοσοφία του Android, κρύβεται στο ακόλουθο διάγραμμα με τα βασικά συστατικά του

Στην στοίβα του Android (Σχήμα 1.3), παρατηρούμε 4 επίπεδα. Ακολούθως θα περιγράψουμε συνοπτικά τα βασικά αυτά επίπεδα χωρίς να μπούμε σε λεπτομέρειες για όλα τα περιεχόμενα του κάθε επιπέδου. Αν ο αναγνώστης επιθυμεί να μάθει περισσότερα, μπορεί να επισκεφθεί την επίσημη ιστοσελίδα του Android για κατασκευαστές (http://developer.android.com). Κάθε επίπεδο στην αρχιτεκτονική αυτή, χρησιμοποιεί τις υπηρεσίες που του προσφέρονται από τα πιο κάτω επίπεδα. Ας δούμε τώρα αυτά τα επίπεδα ξεκινώντας από το πιο χαμηλό.

4.5.1 Πυρήνας Linux (Linux kernel)

Το Android είναι βασισμένο στα γερά θεμέλια του Linux. Ο πυρήνας Linux είναι δοκιμασμένος, σταθερός και πετυχημένος και μπορεί να βρεθεί παντού, από ρολόγια χειρός μέχρι υπερυπολογιστές. Το Linux παρέχει στο Android το αφαιρετικό επίπεδο υλικού, επιτρέποντάς του να μπορεί να χρησιμοποιηθεί σε μεγάλη ποικιλία πλατφόρμων στο μέλλον. Ειδικότερα, το Android χρησιμοποιεί τον πυρήνα Linux για την διαχείριση μνήμης, την διαχείριση διεργασιών, την δικτύωση και άλλες υπηρεσίες του λειτουργικού συστήματος

4.5.2 Εγγενής Βιβλιοθήκες – Native Libraries



Εικόνα 15 Αρχιτεκτονική

Στο αμέσως ψηλότερο επίπεδο βρίσκουμε τις Native Libraries – Εγγενής Βιβλιοθήκες. Όλες αυτές είναι γραμμένες στην γλώσσα προγραμματισμού C και C++ και μεταγλωττίστηκαν για την συγκεκριμένη αρχιτεκτονική υλικού που χρησιμοποιείται από το τηλέφωνο. Οι βιβλιοθήκες αυτές δεν είναι εφαρμογές που μπορούν να σταθούν από μόνες τους. Υπάρχουν για να μπορούν να κληθούν από προγράμματα υψηλότερου επιπέδου. Από την έκδοση Donut και μετά, οι κατασκευαστές μπορούν να γράφουν τις δικές τους τέτοιες βιβλιοθήκες με την χρήση της Εργαλειοθήκης NDK (Native Development Kit).

4.5.3 Χρόνος Εκτέλεσης – Android Runtime

Στο ίδιο επίπεδο με τις εγγενής βιβλιοθήκες, βρίσκουμε και τον χρόνο εκτέλεσης Android. Εδώ ζουν βασικές βιβλιοθήκες της Java και η εικονική μηχανή Dalvik. Η Dalvik είναι μια βελτιστοποιημένη υλοποίηση μιας εικονικής μηχανής Java για φορητές συσκευές από την Google. Η Dalvik τρέχει .dex αρχεία, τα οποία είναι bytecodes που προέρχονται από αρχεία .class και .jar. Εν αντιθέσει όμως με τα .class αρχεία, τα .dex είναι πολύ πιο συμπαγή και αποδοτικά, γεγονός σημαντικό για συσκευές με περιορισμένη μνήμη και μπαταρία. Το Android περιλαμβάνει ένα σύνολο βασικών βιβλιοθηκών που παρέχουν τις περισσότερες από τις διαθέσιμες λειτουργίες των βασικών βιβλιοθηκών της Java. Κάποια πακέτα και κλάσεις υπάρχουν και στο Android κάποια άλλα δεν υποστηρίζονται καθόλου, ενώ ταυτόχρονα το Android παρέχει και επιπρόσθετα προσαρμοσμένα στις δικές του ανάγκες.

4.5.4 Πλαίσιο Εφαρμογής – Application Framework

Πάνω από τις εγγενής βιβλιοθήκες και το χρόνο εκτέλεσης Android, είναι το πλαίσιο εφαρμογής. Αυτό το επίπεδο μας παρέχει υψηλού επιπέδου δομικές μονάδες τις οποίες μπορούμε να χρησιμοποιούμε για την κατασκευή των εφαρμογών μας. Αυτό το πλαίσιο είναι

προ-εγκατεστημένο στο Android, αλλά είναι επεκτάσιμο, αφού ο κάθε κατασκευαστής μπορεί να το συμπληρώσει με δικά του κομμάτια.

Τα σημαντικότερα δομικά στοιχεία του πλαισίου αυτού είναι:

1) Διαχειριστής Δραστηριοτήτων - Activity Manager: Υπεύθυνο για τον έλεγχο του χρόνου ζωής (Σχήμα 1.4) των εφαρμογών και για την διατήρηση μιας στοίβας που επιτρέπει την πλοήγηση του χρήστη σε προηγούμενες οθόνες.
2) Παροχέας Περιεχομένου - Content Providers: Αυτά τα αντικείμενα περιέχουν δεδομένα που μπορούν να διαμοιραστούν μεταξύ εφαρμογών.
3) Διαχειριστής Πόρων - Resource Manager: Οι πόροι, είναι οτιδήποτε υπάρχει σε ένα πρόγραμμα και δεν είναι κώδικας. Για παράδειγμα μπορεί να είναι κωδικοί χρωμάτων, αλφαριθμητικοί χαρακτήρες ή ακόμα και έτοιμα σχεδιαγράμματα οθονών φτιαγμένα σε XML, τα οποία μπορεί το πρόγραμμα να καλεί. 20
4) Διαχειριστής Τοποθεσίας - Location Manager: Χρησιμοποιείται για να μπορεί να ξέρει το τηλέφωνο που βρίσκεται ανά πάσα στιγμή.
5) Διαχειριστής Κοινοποιήσεων - Notification Manager: Ιδανικός τρόπος για να ενημερώνεις τον χρήστη για γεγονότα που συμβαίνουν, διακριτικά χωρίς να διακόπτεις την εργασία του.

4.5.5 Εφαρμογές και Widgets

Στο υψηλότερο επίπεδο της στοίβας Android, φιγουράρουν οι εφαρμογές και τα widgets. Αυτό είναι που βλέπουν οι χρήστες χωρίς να γνωρίζουν την όλη από κάτω διαδικασία. Αυτές είναι εφαρμογές που γράφουν οι κατασκευαστές λογισμικού, εφαρμογές που ήδη είναι εγκατεστημένες στο τηλέφωνο ή που ο χρήστης παίρνει από το Android Market. Οι εφαρμογές είναι προγράμματα που καταλαμβάνουν ολόκληρη την οθόνη και αλληλεπιδρούν με το χρήστη. Από την άλλη τα widget λειτουργούν σε μικρά τετράγωνα μέσα στην αρχική οθόνη – εφαρμογή.

5. Το Android ως πλατφόρμα ανάπτυξης

5.1 Android SDK

Το Android είναι ένα πακέτο λογισμικού για κινητά τηλέφωνα που περιλαμβάνει λειτουργικό σύστημα, έτοιμες βιβλιοθήκες και έτοιμες εφαρμογές. Το Android SDK παρέχει τα εργαλεία και το API που χρειάζεται κάποιος για να ξεκινήσει να αναπτύσσει εφαρμογές στην πλατφόρμα Android χρησιμοποιώντας την γλωσσά προγραμματισμού Java. [1]

Το Android SDK περιλαμβάνει 4 βασικά στοιχειά:

1) Τη συμφωνεί άδειας χρήσης του Android SDK
2) Την τεκμηρίωση Android
3) Το περιβάλλον υλοποίησης εφαρμογών
4) Τα Εργαλεία και τα δείγματα εφαρμογών

5.1.1 Τη συμφωνεί άδειας χρήσης του Android SDK

Για να μπορείτε να λάβετε το Android SDK, πρέπει πρώτα να διαβάσετε και να συμφωνήσετε με τη Συμφωνία άδειας χρήσης του Android SDK. Αυτή η συμφωνία αποτελεί ένα συμβόλαιο ανάμεσα σε σας και την Google.

Ακόμα και αν κάποιος στην εταιρεία σας έχει συμφωνήσει με την Συμφωνία άδειας χρήσης εκ μέρους σας, είναι σημαντικά για σας τον προγραμματιστή να λάβετε υπόψη σας τα παρακάτω σημαντικά σημεία:

1. Τα δικαιώματα που χορηγούνται: Η Google (ως κάτοχος πνευματικών δικαιωμάτων του Android) σας χορηγεί μία περιορισμένη, παγκόσμια, ελεύθερη από την υποχρέωση πληρωμής πνευματικών δικαιωμάτων, μη εκχωρήσαμε σε άλλους και μη αποκλειστική άδεια χρήσης του SDK αποκλειστικά για την ανάπτυξη εφαρμογών για την πλατφόρμα Android. Η Google (και άλλοι συμμετέχοντες στο έργο) σας χορηγούν την άδεια χρήσης, αλλά εξακολουθούν να διατηρούν στην κατοχή τους όλα τα πνευματικά δικαιώματα του υλικού. Η χρήση του Android SDK δεν σας δίνει την άδεια να χρησιμοποιήσετε άλλα λογότυπα ή εμπορικά ονόματα της Google. Δεν θα πρέπει να αφαιρέσετε οποιαδήποτε σημείωση πνευματικών δικαιωμάτων που υπάρχει. Οι τρίτες εφαρμογές, με τις οποίες αλληλεπιδρούν οι εφαρμογές σας (άλλες εφαρμογές του Android) υπόκεινται σε ξεχωριστούς όρους και δεν υπακούν σ' αυτήν τη συμφωνία.

2. Χρήση του SDK: Μπορείτε να αναπτύξετε μόνο εφαρμογές Android. Δεν μπορείτε να παράγετε άλλα έργα από το SDK ή να διανείμετε το SDK σε οποιαδήποτε άλλη συσκευή ή να διανείμετε μέρος του SDK με άλλο λογισμικό.

3. Αλλαγές SDK και συμβατότητα με προηγούμενες εκδόσεις: Η Google μπορεί να αλλάξει το Android SDK ανά πάσα στιγμή, χωρίς καμία ειδοποίηση, χωρίς σεβασμό στη συμβατότητα με προηγούμενες εκδόσεις. Αν και οι αλλαγές στο Android API αποτελούσαν σημαντικό πρόβλημα για τις δοκιμαστικές εκδόσεις του SDK, οι πρόσφατες εκδόσεις είναι αρκετά σταθερές. Ωστόσο, κάθε ανανέωση του SDK επηρεάζει συνήθως ένα μικρό αριθμό υπαρχόντων εφαρμογών, καθιστώντας αναγκαίες τις ενημερώσεις.

4. Δικαιώματα προγραμματιστών εφαρμογών Android: Διατηρείτε όλα τα δικαιώματα για οποιοδήποτε λογισμικό Android που αναπτύσσετε με το SDK, συμπεριλαμβανομένων των πνευματικών δικαιωμάτων. Έχετε επίσης όλη την ευθύνη για το έργο σας.

5. Απαιτήσεις απορρήτου εφαρμογών Android: Συμφωνείτε ότι οι εφαρμογές σας θα προστατεύουν το απόρρητο και τα νόμιμα δικαιώματα των χρηστών τους. Εάν η εφαρμογή σας χρησιμοποιεί ή προσπελαύνει ιδιωτικές πληροφορίες για το χρήστη (όνομα χρήστη, κωδικό πρόσβασης κ.ά.), τότε η εφαρμογή σας θα πρέπει παρέχει να παρέχει μία επαρκή σημείωση απορρήτου και θα πρέπει να αποθηκεύει αυτά τα δεδομένα με ασφάλεια. Σημειώστε ότι οι νόμοι και οι ρυθμίσεις απορρήτου μπορεί να διαφέρουν ανάλογα με τη θέση του χρήστη και ότι εσείς ως προγραμματιστές είστε αποκλειστικά υπεύθυνοι για την κατάλληλη διαχείριση αυτών των δεδομένων.

6. Προϋποθέσεις εφαρμογών Android για κακόβουλο λογισμικό: Είστε υπεύθυνοι για όλες τις εφαρμογές που αναπτύσσετε. Συμφωνείτε να μην δημιουργείτε επικίνδυνες εφαρμογές ή κακόβουλο λογισμικό. Είστε αποκλειστικά υπεύθυνοι για όλα τα δεδομένα που μεταφέρονται μέσα από την εφαρμογή σας.

7. Πρόσθετοι όροι για συγκεκριμένα Google API: Η χρήση του Android Maps API υπόκειται σε περαιτέρω Όρους υπηρεσίας (συγκεκριμένα, στη χρήση των εξής πακέτων: com. google. android.maps και com.android.location.Geocoder). Πρέπει να συμφωνήσετε μ' αυτούς τους -πρόσθετους όρους, ώστε να χρησιμοποιήσετε αυτά τα συγκεκριμένα API και να συμπεριλαμβάνετε πάντα τη σημείωση πνευματικών δικαιωμάτων του Google Maps.Η χρήση των Google Data API (εφαρμογές Google όπως Gmail, Blogger, Google Calendar, Google Finance Portfolio Data, Picasa, YouTube και άλλες) παρέχουν πρόσβαση, η οποία περιορίζεται στα δικαιώματα πρόσβασης που ο χρήστης έχει παραχωρήσει ρητά στην εφαρμογή σας μέσω αποδοχής δικαιωμάτων που παρέχονται από τον προγραμματιστή κατά την εγκατάσταση.

8. Η ανάπτυξη γίνεται με δική σας ευθύνη: Οποιαδήποτε ζημία που προέρχεται από την ανάπτυξη με το Android SDK επιβαρύνει μόνο εσάς και όχι τη Google.

5.1.2 Την τεκμηρίωση Android

Η τεκμηρίωση χωρίζεται τώρα σε επτά βασικές ενότητες:

■ Η καρτέλα Home (Αρχική) είναι το σημείο εκκίνησης μέσα στην τεκμηρίωση του Android. Εδώ θα βρείτε ανακοινώσεις προγραμματιστών και σημαντικές συνδέσεις για τα τελευταία «καυτά» θέματα σε σχέση με την ανάπτυξη με το Android.

■ Η καρτέλα SDK παρέχει πληροφορίες για τις διαφορετικές διαθέσιμες εκδόσεις του Android SDK, όπως και πληροφορίες για το εσωτερικό πακέτο ανάπτυξης του Android,

Native Development Kit (NDK). 0α δείτε επίσης σημειώσεις για τις εκδόσεις του Android SDK.

■ Η καρτέλα Dev Guide (Οδηγός συσκευών) παρουσιάζει την πλατφόρμα Android και Ι καλύπτει τις καλύτερες πρακτικές για τη σχεδίαση και ανάπτυξη εφαρμογών Android, όπως και πληροφορίες για τη δημοσίευση εφαρμογών.

■ Η καρτέλα Reference (Αναφορά) παρέχει μία αναλυτική λίστα με όλα τα API για το Android, με λεπτομερή περιγραφή συγκεκριμένων κλάσεων και διεπαφών.

■ Η καρτέλα Resources (Πόροι) παρέχει πρόσβαση σε τεχνικά άρθρα και μαθήματα για το Android. Θα βρείτε επίσης συνδέσεις για την κοινότητα του Android (ομάδες, λίστες αλλη-1 λογραφίας και επίσημη τροφοδοσία από το Twitter), όπως και τα δείγματα εφαρμογών! που παρέχονται μαζί με το Android SDK.

■ Η καρτέλα Videos (Βίντεο) παρέχει πρόσβαση σε βίντεο, που παρουσιάζουν θέματα ανάπτυξης με το Android, όπως πληροφορίες για την πλατφόρμα, συμβουλές για προγραμματιστές, συνόδους ανάπτυξης με το Android από τις διασκέψεις της Google και συνεντεύξεις με προγραμματιστές.

■ Η καρτέλα Blog (Ιστολόγιο) παρέχει πρόσβαση στο ιστολόγιο που δημοσιεύεται απ' την ομάδα ανάπτυξης του Android. Εδώ θα βρείτε ανακοινώσεις για κυκλοφορίες SDK, χρήσιμες συμβουλές ανάπτυξης και πληροφορίες για επερχόμενα συμβάντα που σχετίζονται με το Android.

Η τεκμηρίωση Android παρέχεται σε μορφή HTML τοπικά και στο Web στη διεύθυνση http:// developer.android.com. Ορισμένα χαρακτηριστικά της τεκμηρίωσης Android που αφορούν σε δικτυακά θέματα (όπως οι καρτέλες Blog και Video) διατίθενται μόνο στο Web.

5.1.3 Το περιβάλλον υλοποίησης εφαρμογών

Το περιβάλλον υλοποίησης εφαρμογών Android παρέχεται στο αρχείο android. jar. Το Android SDK αποτελείται από αρκετά σημαντικά πακέτα.

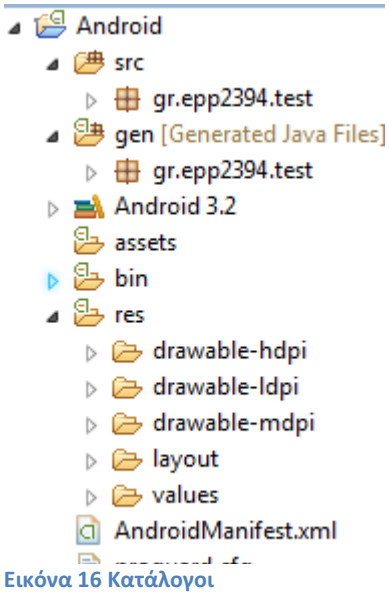
Υπάρχει επίσης ένα προαιρετικό πρόσθετο για Google API, το οποίο αποτελεί επέκταση του Android SDK και διευκολύνει την ανάπτυξη με το Google Maps και άλλα API και υπηρεσίες της Google. Για παράδειγμα, εάν θέλετε να συμπεριλάβετε το στοιχείο ελέγχου Map View στην εφαρμογή σας, πρέπει να εγκαταστήσετε και να χρησιμοποιήσετε αυτό το χαρακτηριστικό. Αυτό το πρόσθετο πρόγραμμα αντιστοιχεί στο πακέτο com. google. * (συμπεριλαμβανομένου του com.google.android.maps) και απαιτεί συμφωνία με πρόσθετους όρους και εγγραφή για τη λήψη ενός κλειδιού API. Για περισσότερες πληροφορίες γι' αυτό το πρόσθετο πρόγραμμα, διαβάστε τη σελίδα http://code.google.com/android/add-ons/google-apis/.

5.1.4 Τα Εργαλεία και τα δείγματα εφαρμογών

Το Android SDK παρέχει πολλά εργαλεία για σχεδίαση, ανάπτυξη, εκσφαλμάτωση και υλοποίηση των εφαρμογών σας Android. Το πρόσθετο πρόγραμμα Eclipse ενσωματώνει πολλά απ' αυτά τα εργαλεία αρμονικά στο περιβάλλον ανάπτυξης και παρέχει διάφορους οδηγούς για τη δημιουργία και την εκσφαλμάτωση έργων Android.
Θα βρείτε τις ρυθμίσεις για το πρόσθετο πρόγραμμα ADT στο στοιχείο Eclipse μέσα στο μενού Window, Preferences, Android. Εδώ μπορείτε να καθορίσετε τη θέση στο δίσκο όπου εγκαταστήσατε το SDK και τα εργαλεία του Android, όπως και πολλές άλλες ρυθμίσεις δομή και εκσφαλμάτωσης.
org.xmlpull.
Το πρόσθετο πρόγραμμα ADT αυξάνει τον αριθμό των χρήσιμων λειτουργιών στο προεπιλεγμένο Eclipse IDE. Πολλά νέα κουμπιά υπάρχουν στη γραμμή εργαλείων, μεταξύ των οποίων βλέπουμε κουμπιά για:

- Εκκίνηση του προγράμματος Android SDK and AVD Manager.
- Δημιουργία νέου έργου με τον οδηγό Android Project Wizard.
- Δημιουργία νέου έργου δοκιμών με τον οδηγό Android Project Wizard.
- Δημιουργία νέου αρχείου πόρου Android XML.

Υπάρχει επίσης μία ειδική πλευρά του Eclipse για την εκσφαλμάτωση εφαρμογών Android, if οποία ονομάζεται DDMS (Dalvik Debug Monitor Server, διακομιστής παρακολούθησης εκσφαλμάτωσης Dalvik). Μπορείτε να ενεργοποιήσετε αυτήν την πλευρά του Eclipse επιλέγοντας Window^ Open Perspective, DDMS ή αλλάζοντας την επιλογή DDMS στην πάνω δεξιά γωνία της οθόνης. Θα μιλήσουμε για το DDMS παρακάτω σ' αυτό το κεφάλαιο. Αφού σχεδιάσετε μία εφαρμογή Android, μπορείτε επίσης να χρησιμοποιήσετε το πρόσθετο πρόγραμμα ADT για το Eclipse, ώσπ να ξεκινήσετε έναν οδηγό συσκευασίας και υπογραφής της εφαρμογής σας Android, προκειμέ¬νου να τη δημοσιεύσετε. Θα μιλήσουμε περισσότερο γι' αυτό στο Κεφάλαιο 29, «Πουλήστε την εφαρμογή σας Android».



Εικόνα 16 Κατάλογοι

5.2 Ανάλυση των καταλογών σε ένα Android project

Σε αυτήν την ενότητα θα δούμε τους καταλόγους που δημιουργούνται όταν ξεκινάμε ένα νέο Android project. Θα εξετάσουμε τα περιεχόμενα τους αλλά και τι μπορούμε να

προσθέσουμε σε αυτούς και θα αναλύσουμε το xml αρχείο AndroidManifest.

Στην Εικόνα βλέπουμε τους καταλόγους που δημιουργήθηκαν για το project με το όνομα Android μέσα από το περιβάλλον ανάπτυξης. [1]

5.2.1 Ο κατάλογος src

Το όνομα αυτού του καταλόγου έχει προέρθει από την συντομογραφία της Αγγλικής λέξης source που σημαίνει «πηγή». Σε αυτόν βρίσκεται όλος ο πηγαίος κώδικας μας, είτε είναι ένα αρχείο δηλαδή μια κλάση είτε είναι πολλές, πάντα όμως κάτω από ένα πακέτο.

Όπως βλέπουμε από την παραπάνω εικόνα, κάτω από τον κατάλογο /src υπάρχει το πακέτο που έχουμε δηλώσει ότι ανήκουν οι κλάσεις μας και το όνομα αυτού είναι συνήθως η προσωπική μας ιστοσελίδα. Επίσης στην ίδια εικόνα βλέπουμε και το αρχείο AndroidActivity.java που δημιουργήθηκε αυτόματα από το SDK του Android και κληρονομεί από την κλάση Activity

```
1  package gr.epp2394.test;
2
3  import android.app.Activity;
5
6  public class AndroidActivity extends Activity {
7      /** Called when the activity is first created. */
8      @Override
9      public void onCreate(Bundle savedInstanceState) {
10         super.onCreate(savedInstanceState);
11         setContentView(R.layout.main);
12     }
13 }
```

Εικόνα 17

5.2.2 Ο κατάλογος res

Ο κατάλογος αυτός περιέχει τους πόρους (resources) και το όνομα του είναι μια συντομογραφία της Αγγλικής λέξης resources. Οι πόροι μπορούν να είναι κείμενο, εικόνες, ήχος ή ποιο απλά ότι δεν είναι κώδικας.

Σε αυτόν τον κατάλογο δημιουργούμε και αποθηκεύουμε τους πόρους ανάλογα με τον τύπο τους. Για παράδειγμα, αν έχουμε μια εικόνα τύπου jpeg ή png τότε πρέπει να πάει στον αντίστοιχο κατάλογο που ξεκινάει έτσι /res/drawable και μπορεί να έχει κατάληξη -hdpi, - mdpi, -ldpi. Αν έχουμε ένα xml αρχείο το οποίο καθορίζει την εμφάνιση που θα έχει η εφαρμογή μας τότε πρέπει να πάει στον κατάλογο /res/layout. Τέλος πέρα από τα

υπάρχοντα επιθήματα μπορούμε να προσθέσουμε και δικά μας ανάλογα με τον τύπο των πόρων και τις ανάγκες που έχουμε.

Όλοι οι πόροι μας μεταγλωττίζονται από τον resource compiler (μεταγλωττιστή πόρων), ο οποίος τους συμπιέζει και τους πακετάρει δημιουργώντας μια κλάση με το όνομα R η οποία περιέχεται στον παρακάτω κατάλογο με το όνομα GEN. Η κλάση R περιέχει τα αναγνωριστικά που χρησιμοποιούμε έτσι ώστε να μας επιτρέπει να αναφερόμαστε σε αυτούς τους πόρους μέσα από το πρόγραμμα μας.

5.2.3 Ο κατάλογος gen

Στο κατάλογο αυτό θα βρούμε την κλάση R που αναφέραμε προηγουμένως στο 5.1.2 και όπως συμβαίνει στο κατάλογο /src έτσι και εδώ η κλάση αυτή βρίσκεται κάτω από το πακέτο που ορίσαμε.

Κάτι πολύ σημαντικό που πρέπει να σημειώσουμε εδώ είναι ότι η κλάση R διαχειρίζεται αυτόματα από το Android plug-in του Eclipse. Κάθε φορά που βάζουμε ένα αρχείο οπουδήποτε στον κατάλογο /res το plug-in παρατηρεί την αλλαγή και προσθέτει τα κατάλληλα αναγνωριστικά IDs των πόρων στο αρχείο R.java για εμάς. Το R.java μένει πάντα συγχρονισμένο. Αν το ανοίξουμε θα δούμε ότι περιέχει κάτι σαν το παρακάτω:

R.java

```
1  /* AUTO-GENERATED FILE.  DO NOT MODIFY.
2   *
3   * This class was automatically generated by the
4   * aapt tool from the resource data it found.  It
5   * should not be modified by hand.
6   */
7
8  package gr.epp2394.test;
9
10 public final class R {
11     public static final class attr {
12     }
13     public static final class drawable {
14         public static final int ic_launcher=0x7f020000;
15     }
16     public static final class layout {
17         public static final int main=0x7f030000;
18     }
19     public static final class string {
20         public static final int app_name=0x7f040001;
21         public static final int hello=0x7f040000;
22     }
23 }
```

Εικόνα 18 R

Οι παραπάνω δεκαεξαδικοί αριθμοί είναι ακέραιοι αριθμοί οπού ο Resource Manager του Android τους χρησιμοποιεί για να φορτώσει τα πραγματικά δεδομένα όπως αλφαριθμητικά και αλλά στοιχεία τα οποία συγκεντρώνονται μέσα στο πακέτο μας. Οι τιμές που βλέπουμε ότι ισούνται δεν αντιστοιχούν στις πραγματικές τιμές των δεδομένων μας αλλά σε τιμές που αφορούν τους ίδιους τους πόρους μας. Πρέπει να σημειώσουμε πως σχεδόν κάθε πρόγραμμα Android, συμπεριλαμβανομένου και του ίδιου του Framework του Android, έχει μία κλάση R [18].

5.2.4 Το xml αρχείο androidmanifest

Κάθε εφαρμογή πρέπει να έχει ένα αρχείο AndroidManifest.xml [19] με ακριβώς αυτό το όνομα στον αρχικό (root) κατάλογο του. Στο αρχείο παρουσιάζονται οι απαραίτητες πληροφορίες σχετικά με την εφαρμογή μας στο σύστημα του Android, πληροφορίες τις οποίες χρειάζεται το σύστημα πριν μπορέσει να εκτελέσει οποιοδήποτε κώδικα της εφαρμογής. Μεταξύ άλλων το AndroidManifest κάνει και τα ακόλουθα:

- Ονομάζει το Java πακέτο της εφαρμογής. Το όνομα του πακέτου λειτουργεί ως μοναδικό αναγνωριστικό ID για την εφαρμογή
- Περιγράφει τις συνιστώσες της εφαρμογής. Για παράδειγμα, τις activities , services (υπηρεσίες), content providers (παρόχους περιεχομένου) κτλ
- Ονομάζει τις κλάσεις που εφαρμόζουν καθεμία από τις συνιστώσες και δημοσιεύει τις ικανότητές τους. Για παράδειγμα, ποιες Intents (Προθέσεις) μπορούν να χειριστούν
- Οι δηλώσεις αυτές αφήνουν το σύστημα Android να γνωρίζει ποιες είναι οι συνιστώσες και υπό ποιες συνθήκες μπορούν να ενεργοποιούνται
- Δηλώνει ποια δικαιώματα η εφαρμογή πρέπει να έχει προκειμένου να γίνει δυνατή η πρόσβαση σε προστατευμένες περιοχές του API και να μπορεί να αλληλεπιδρά με άλλες εφαρμογές.
- Δηλώνει τα δικαιώματα που οι άλλοι οφείλουν να έχουν προκειμένου να αλληλεπιδράσουν με στοιχεία της εφαρμογής
- Δηλώνει το ελάχιστο επίπεδο του Android API που απαιτεί η εφαρμογή [1]

Πολλά από τα παραπάνω μπορούμε να τα δούμε σε ένα τυπικό αρχείο AndroidManifest.xml που αυτόματα το SDK δημιούργησε μόλις φτιάξαμε ένα νέο Android project, και έχει ως εξής:

AndroidActivity.java | R.java | Android Manifest

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3      package="gr.epp2394.test"
4      android:versionCode="1"
5      android:versionName="1.0" >
6
7      <uses-sdk android:minSdkVersion="13" />
8
9      <application
10         android:icon="@drawable/ic_launcher"
11         android:label="@string/app_name" >
12         <activity
13             android:name=".AndroidActivity"
14             android:label="@string/app_name" >
15             <intent-filter>
16                 <action android:name="android.intent.action.MAIN" />
17
18                 <category android:name="android.intent.category.LAUNCHER" />
19             </intent-filter>
20         </activity>
21     </application>
22
23 </manifest>
```

Εικόνα 19 AndroidManifest

- Από τον παραπάνω xml κώδικα θα αναλύσουμε τις ετικέτες που χρησιμοποιούνται και είναι αυτές που απαιτούνται να υπάρχουν ώστε να λειτουργήσει η εφαρμογή μας σε μια Android συσκευή. Επίσης πρέπει να αναφέρουμε ότι σχεδόν όλα τα γνωρίσματα πρέπει να φέρουν μπροστά τους την λέξη android. Τέλος, ενδεικτικά θα αναφέρουμε ποιές άλλες ετικέτες και γνωρίσματα μπορούμε να χρησιμοποιήσουμε.
- Η ετικέτα <manifest> αποτελεί την αρχική ετικέτα του αρχείου και πρέπει να περιέχει την ετικέτα <application> και προαιρετικά μπορεί να περιέχει άλλες ετικέτες, ενδεικτικά τις <permission>, <permission-group>, <uses-permission>, κτλ. Τα γνωρίσματα που περιέχει η ετικέτα<manifest> του παραπάνω κώδικα είναι τα εξής:

    - package: σε αυτό το γνώρισμα δηλώνεται το πακέτο που χρησιμοποιείται. Στο συγκεκριμένο παράδειγμα είναι το πακέτο org.example.example

    - android:versionCode: Εδώ δηλώνεται η έκδοση του κώδικα μας και μπορεί να είναι οποιαδήποτε ακέραια τιμή θέλουμε. Συνήθως οι μεγαλύτεροι αριθμοί δηλώνουν και ποιο πρόσφατη έκδοση. Αυτό το γνωρησμα χρησιμοποειται εσωτερικα από το συστημα και δεν είναι ορατο από τον χρηστη. Στο παραδειγμα μας είναι η εκδοση 1.

    - android:vesionName: Εδώ δηλώνουμε τον αριθμό έκδοσης που θέλουμε να εμφανίζεται στον χρήστη. Όπως και πριν η έκδοση μας είναι η 1.0
- Η ετικέτα <application> ορίζει την εφαρμογή μας. Συμπεριλαμβάνεται στην ετικέτα <manifest> και μπορεί να περιέχει άλλες ετικέτες όπως <activity>, <service>, <provider> κα. Τα γνωρίσματα που περιέχει η εν λόγω ετικέτα στον παραπάνω κώδικα είναι τα εξής:

    - android:icon: Με αυτό το γνώρισμα δηλώνουμε το εικονίδιο που θα εμφανίζει η εφαρμογή μας στον χρήστη. Αυτή η εικόνα θα πρέπει να βρίσκεται σε έναν από τους φακέλους /res/drawable

    - android:label: Εδώ καθορίζουμε το όνομα της εφαρμογής που θα εμφανίζεται στον χρήστη.

- Η ετικέτα <activity> δηλώνει ένα activity το οποίο αποτελεί εν μέρει και το user interface της εφαρμογής. Όλα τα activities της εφαρμογής μας οφείλουν να αντιπροσωπεύονται από μια ετικέτα <activity> μέσα στο αρχείο AndroidManifest της εφαρμογής μας. ^ σύστημα δεν μπορεί να δει ένα activity το οποίο δεν έχει δηλωθεί στο αρχείο, οπότε δεν πρόκειται και να το εκτελέσει. Αυτή η ετικέτα συμπεριλαμβάνεται στην ετικέτα <application> και μπορεί να περιέχει τις ετικέτες <intent-filter> και <meta-data>. Στον παραπάνω κώδικα βλέπουμε τα εξής γνωρίσματα:

    - android:name: Εδώ δηλώνουμε το όνομα του activity μας. Κανονικά τα ονόματα δηλώνονται με το πλήρες όνομα τους, δηλαδή για το παράδειγμα μας έτσι org.example.example.Example, αλλά για λόγους συντομίας βάζουμε

μια τελεία μπροστά από το όνομα του activity μας και αυτόματα το σύστημα παίρνει το πακέτο που έχουμε δηλώσει.

- android:label: Σε αυτό το γνώρισμα δηλώνουμε το όνομα που θέλουμε να έχει το activity μας και εμφανίζεται στον τίτλο της όταν εκτελείτε. Συνήθως βάζουμε το όνομα της εφαρμογής μας.

• Η ετικέτα <intent-filter> καθορίζει τους τύπους των intents που ανταποκρίνεται το activity μας και δηλώνει της δυνατότητες του τελευταίου καθώς και το τι μπορεί ή όχι να χειριστεί. Περισσότερα για τα intents θα δούμε παρακάτω. Αυτή η ετικέτα συμπεριλαμβάνεται σε μία ετικέτα <activity> ή <service> ή <receiver> και πρέπει να περιέχει την ετικέτα <action>. Επιπρόσθετα μπορεί να περιέχει τις <category> και <data>.

- Η ετικέτα <action> πρέπει να συμπεριλαμβάνεται στην <intent-filter> αν θέλουμε να ορίσουμε ποια intents θα περνάνε από το φίλτρο. Αν δεν έχουμε ορίσει κανένα <action> τότε το activity μας δεν λαμβάνει και δεν χειρίζεται κανένα intent. Στο παράδειγμα μας περιέχει το γνώρισμα name με την τιμή android.intent.action.MAIN που δηλώνει πως το activity μας δεν χρειάζεται δεδομένα ώστε να ξεκινήσει. Σαν να είναι η main μιας java εφαρμογής.

- Η ετικέτα <category> περιέχει επιπρόσθετες πληροφορίες για το είδος της συνιστώσας που πρέπει να χειριστεί το intent. Στο παράδειγμα μας έχουμε το γνώρισμα name με την τιμή android.intent.category.LAUNCHER που δηλώνει ότι το activity μας είναι το πρωταρχικό activity ενός project ή αλλιώς της εφαρμογή μας και βρίσκεται στην λίστα των ανώτερων επιπέδων (top level) της εκκίνησης εφαρμογών.

Τα activities που μπορούν να αρχικοποιούν, δηλαδή να ξεκινούν εφαρμογές και ταυτόχρονα να τις αντιπροσωπεύουν στον application launcher (προωθητή εφαρμογών), περιέχουν τα στοιχεία, <action> και <category> που περιγράψαμε με αυτές τις τιμές.

- Η ετικέτα <uses-sdk> μας δίνει τη δυνατότητα να εκφράσουμε τη συμβατότητα της εφαρμογής μας με μία ή με περισσότερες εκδόσεις της Android πλατφόρμας, δηλώνοντας τον ακέραιο αριθμό που αντιπροσωπεύει το επίπεδο του API Εικόνα. Συμπεριλαμβάνεται στην ετικέτα <manifest> και μπορεί να περιέχει τα γνωρίσματα minSdkVersion, targetSdkVersion και maxSdkVersion.

  Από το παράδειγμα μας βλέπουμε ότι έχει οριστεί το γνώρισμα minSdkVersion με τιμή 13. Αυτό σημαίνει ότι για να τρέξει η εφαρμογή μας σε μια Android συσκευή πρέπει να έχει λειτουργικό σύστημα με έκδοση 3.2 και πάνω.

Distribution	API level	
4.0.x *Ice Cream Sandwich*	14-15	1.0%
3.x.x *Honeycomb*	11-13	3.4%
2.3.x *Gingerbread*	9-10	58.6%
2.2 *Froyo*	8	27.8%
2.0, 2.1 *Eclair*	7	7.6%
1.6 *Donut*	4	1.0%
1.5 *Cupcake*	3	0.6%

Εικόνα 20 Api level

6 OpenCV

6.1 Γενικά στοιχεία

Η OpenCV είναι μια βιβλιοθήκη ελευθέρου λογισμικού η οποία αναπτύχθηκε από την Intel και αφορά στην επεξεργασία εικόνας. Η OpenCV παρέχει μία μεσαίου έως υψηλού επιπέδου διασύνδεση εφαρμογών με περίπου τριακόσιες συναρτήσεις γραμμένες σε C και μερικές κλάσεις C++. Η ανάπτυξη επικοινωνίας ανθρώπου με υπολογιστή, η ανίχνευση, απομόνωση και αναγνώριση αντικειμένων, η ανίχνευση και αναγνώριση προσώπων, η κατανόηση και παρακολούθηση κίνησης είναι μερικά από τα πεδία μηχανικής όρασης που καλύπτει. Οι αλγόριθμοι της OpenCV είναι βελτιστοποιημένοι για επεξεργαστές αρχιτεκτονικής Intel Pentium (MMX, Pro, 3, 4). Τέλος, η δημιουργία της OpenCV αποσκοπεί και στην δημιουργία μιας κοινότητας ανοιχτού λογισμικού, σχετική με την μηχανική όραση, η οποία θα αναπτύσσει σύγχρονες μεθόδους επεξεργασίας εικόνας σε ένα συνεχώς αναπτυσσόμενο τεχνολογικό περιβάλλον. [8]

6.2 Χρησιμοποιούμενες δομές

Μερικές βασικές δομές τις οποίες χρησιμοποιούμε στο πρόγραμμα είναι:

1) IplImage= Πίνακας στον οποίο αποθηκεύονται εικόνες. Μπορεί να έχει περισσότερα του ενός κανάλια η εικόνα .

2) CvSize= Δομή για την αποθήκευση του μεγέθους των εικόνων

3)CvScalar= Δομή που έχει 4 μεταβλητές double μια για κάθε κανάλι μιας εικόνας.

Για περισσότερες πληροφορίες όσο αφορά την βιβλιοθικη OpenCV μποειτε να επισκευτητε το http://cgi.cs.indiana.edu/~oleykin/website/OpenCVHelp/ . [8]

6.3 Συναρτήσεις προγράμματος σε OpenCV

Παρά κάτω θα μιλήσουμε για κάποιες συναρτήσεις της OpenCV και θα αναλύσουμε μερικές από αυτές. [8]

6.3.1 Συναρτήσεις χρησιμοποιούμενες για τον εντοπισμό δέρματος

Μερικές συναρτήσεις που χρησιμοποιούνται για τον εντοπισμό δέματος είναι : cvNormal01, cvRGB2YCbCrChanelY, cvRGB2YCbCrChanelCr, cvRGB2YCbCrChanelCb, cvGetSkinProps, cvprocskindetection,

6.3.1.1 cvNormal01

IplImage* cvNormal01 (IplImage* x, IplImage* y, double mx, double my, double sx, double sy)

Η συνάρτηση cvNormal01 επιστρέφει έναν double πίνακα ιδίων διαστάσεων με αυτές των frame του βίντεο (ιδίων διαστάσεων με αυτές των πινάκων x και y όπου τα x και y είναι ο Cb και Cr πίνακας αντίστοιχα) ο οποίος περιέχει τις πιθανότητες το pixel να είναι δέρμα. Κάθε στοιχείο αυτού του πίνακα είναι θετικό και μικρότερο του ένα. Ο υπολογισμός γίνεται χρησιμοποιώντας τα mx, sx, sy και my και υπολογίζουμε με πράξεις πινάκων την τελική πιθανότητα υψώνοντας σε αρνητική εκθετική δύναμη το τελικό αποτέλεσμα προκειμένου να είμαστε εντός των περιορισμών που έχουμε λόγω του ότι το αποτέλεσμα πρέπει να είναι στην περιοχή [0,1]. Αυτή η συνάρτηση είναι καθοριστική μιας και μέσω αυτής μπορούμε να εντοπίσουμε το δέρμα με ικανοποιητική ακρίβεια και χρησιμοποιείται πολύ στο πρόγραμμα.

IplImage* x =ο πίνακας Cb της εικόνας.

IplImage* y = ο πίνακας Cr της εικόνας.

double mx= μεταβλητή που υπολογίζεται από την συνάρτηση

CvGetSkinProps.

double my= μεταβλητή που υπολογίζεται από την συνάρτηση

CvGetSkinProps.

double sx=0.001 συνήθως.

double sy=0.001 συνήθως.

6.3.1.2 CvRGB2YCbCrChanelY

IplImage* cvRGB2YCbCrChanelY (IplImage* X)

Η συνάρτηση cvRGB2YCbCrChanelY παίρνει σαν είσοδο μια RGB εικόνα τριών καναλιών και υπολογίζει την αντίστοιχη YCbCr εικόνα και επιστρέφει το κανάλι Y. Χρησιμοποιείται ο γνωστός τρόπος μετατροπής μιας εικόνας από RGB σε YCbCr. Χρησιμοποιείται ο πίνακας τρία επί τρία TAB ο οποίος ουσιαστικά είναι η μήτρα μετατροπής από την βάση RGB στην βάση YCbCr και με ένα πολλαπλασιασμό πινάκων επιτυγχάνουμε αυτήν την μετατροπή. Συγκεκριμένα για τον υπολογισμό του καναλιού Y χρησιμοποιούμε την πρώτη γραμμή του πίνακα TAB. Επίσης διαιρώντας με 255 κανονικοποιούμε τα αποτελέσματα και τα κάνουμε μορφής double.

IplImage* X = RGB εικόνα τριών καναλιών.

TAB=

0.299	0.587	0.114
-0.169	-0.331	0.5
0.5	-0.419	-0.081

6.3.1.3 cvGetSkinProps

IplImage* cvGetSkinProps (IplImage* X, double* zhue, double* zsat, double* zval, double* mx, double* my, double* mhue, double* msat, double* mval)

Η συνάρτηση cvGetSkinProps παίρνει σαν είσοδο μια RGB εικόνα τριών καναλιών και υπολογίζει την αντίστοιχη YCbCr εικόνα και επιστρέφει έναν double πίνακα ιδίων διαστάσεων με αυτές των frame του βίντεο (ιδίων διαστάσεων με αυτές των πινάκων x και y όπου τα x και y είναι ο Cb και Cr πίνακας αντίστοιχα) ο οποίος περιέχει τις πιθανότητες το pixel να είναι δέρμα. Κάθε στοιχείο αυτού του πίνακα είναι θετικό και μικρότερο του ένα. Η συνάρτηση αυτή επίσης υπολογίζει και τα mx και my τα οποία είναι απαραίτητα για τον υπολογισμό των πιθανοτήτων για το αν ένα pixel είναι δέρμα η όχι. Ο Υπολογισμός των mx,my γίνεται μέσω της εξής διαδικασίας. Κατά την εκτέλεση της cvGetSkinProps ζητείται από τον χρήστη να επισημάνει πάνω στην εικόνα ποια περιοχή θεωρείται δέρμα. Παίρνοντας το μέσω όρο των τιμών αυτής της περιοχής υπολογίζουμε το mx από το κανάλι Cr και το my από το κανάλι Cb.

IplImage* X = RGB εικόνα τριών καναλιών.

double* zhue ->Βοηθητική μεταβλητή.

double* zsat ->Βοηθητική μεταβλητή.

double* zval ->Βοηθητική μεταβλητή.

double* mx ->Βοηθητική μεταβλητή.

double* my ->Βοηθητική μεταβλητή.

double* mhue ->Βοηθητική μεταβλητή.

double* msat ->Βοηθητική μεταβλητή.

double* mval ->Βοηθητική μεταβλητή.

6.3.2 Συναρτήσεις χρησιμοποιούμενες για τον εντοπισμό κίνησης

Μερικες σθναρτησεις που χρησιμοποιουντια για τον εντοπισμο κινισης είναι: cvFindMovingSkin, cvProcessImgSeq

6.3.2.1 cvFindMovingSkin

void cvFindMovingSkin(IplImage* ImgA,IplImage* ImgB,double mx,double my,IplImage* ms,IplImage* cmout,IplImage* mmout)

Η συνάρτηση cvFindMovingSkin υπολογίζει τρεις πολύ βασικούς πίνακες καθοριστικής σημασίας για τον εντοπισμό του δέρματος αλλά και της κίνησης. Όι πίνακες αυτοί είναι λογικοί πίνακες(περιέχουν άσους και μηδενικά). Ο ένας περιέχει την κίνηση που παρατηρήθηκε μεταξύ των δύο εικόνων ο άλλος περιέχει της περιοχές που θεωρούνται δέρμα και ο τρίτος την κίνηση του δέρματος που παρουσιάστηκε ανάμεσα στις δυο εικόνες. Η συνάρτηση αυτή είναι η πιο σημαντική συνάρτηση του προγράμματος μου μιας και χρησιμοποιεί σχεδόν όλες τις υπόλοιπες και βγάζει τα πιο ουσιαστικά αποτελέσματα μέσα από τα οποία παίρνουμε τα τελικά συμπεράσματά μας.

IplImage* ImgA= Το πρώτο frame.

IplImage* ImgB= Το δεύτερο frame.

double mx = μεταβλητή που υπολογίζεται από την cvGetSkinProps.

double my = μεταβλητή που υπολογίζεται από την cvgetskinprops.

IplImage* ms = πίνακας κινούμενου δέρματος.

IplImage* cmout = πίνακας που εντοπίζει που υπάρχει δέρμα.

IplImage* mmout = πίνακας που εντοπίζει που γίνεται κίνηση.

6.3.2.2 cvProcessImgSeq

IplImage* cvProcessImgSeq(char MainImgName[],int NumberOfImgs,char ImgType[])

Η συνάρτηση cvProcessImgSeq είναι η συνάρτηση μέσω της οποίας όλες οι συναρτήσεις του προγράμματός μου συνεργάζονται προκείμενα να πάρουμε το επιθυμητό αποτέλεσμα. Ουσιαστικά εργαζόμαστε ανα δύο εικόνων με όλα τα frame του βίντεο και εντοπίζουμε την κίνησης του δέρματος που παρατηρείται σε αυτά τα ζευγάρια και παίρνουμε τα αποτελέσματα. Η συνάρτηση επιστρέφει έναν πίνακα διαστάσεων 6 επί το πλήθος των ζευγαριών εικόνων τα οποία επεξεργαζόμαστε και ο οποίος περιέχει τα κέντρα του κεφαλιού και των δύο χεριών ανά κάθε ζευγάρι εικόνων. Αυτό γίνεται γιατί μετά το τέλος αυτής της συνάρτησης θα απεικονίσουμε τα αποτελέσματα και γι' αυτό χρειαζόμαστε αυτόν τον πίνακα. Επίσης δίνουμε στην συνάρτηση το όνομα και τον τύπο των εικόνων του το οποίο θέλουμε να επεξεργαστούμε καθώς και το πλήθος αυτών των εικόνων.

char MainImgName[] = Το κυρίως όνομα των εικόνων.

int NumberOfImgs= Το πλήθος των εικόνων.

char ImgType[]= Ο τύπος των αρχείων.

6.3.3 Συναρτήσεις χρησιμοποιούμενες για εντοπισμό δερματικής

Μερικες συναρτησεις που χρησιμοποιουνται για εντοπισμο δερματικης είναι : cvGetSkinProps, cvobjxy, cvFindMovingSkin, cvProcSkinDetection, cvProcessImgSeq, ShowSkinMovementOnImgManual.

6.3.3.1 cvProcSkinDetection

IplImage* cvProcSkinDetection(IplImage* ms,IplImage* cmout,IplImage* mmout,IplImage* prevcoords)

Η συνάρτηση cvProcSkinDetection είναι αυτή που καθορίζει μεταξύ δύο εικόνων ποια θα είναι η νέα θέση του κεφαλιού και των δύο χεριών. επιστρέφει έναν μονοδιάστατο πίνακα έξι θέσεων όπου ανα δυο βρίσκονται οι συντεταγμένες των τριών βασικών σημείων. Χρησιμοποιούνται οι βασικοί πίνακες που έχουν υπολογιστεί από την cvFindMovingSkin και με μερικούς λογικούς ελέγχους καταλήγουμε να αποφανθούμε για το ποια από τα υπάρχοντα αντικείμενα είναι αυτά που μας ενδιαφέρουν και αντίστοιχα επιλέγουμε τις συντεταγμένες τους και της επιστρέφει η συνάρτηση σε έναν πίνακα ώστε να χρησιμοποιηθούν παρακάτω στο πρόγραμμα.

IplImage* ms = πίνακας κινούμενου δέρματος.

IplImage* cmout = πίνακας που εντοπίζει που υπάρχει δέρμα.

IplImage* mmout = πίνακας που εντοπίζει που γίνεται κίνηση.

IplImage* prevcoords = Η συνταταγμένες του κεφαλιού και των δύο χεριών πριν την εκτέλεση της cvProcSkinDetection.Αν δεν έχει εκτελεσθεί καμία φορά η cvProcSkinDetection τότε η prevcoords περιέχει άσους.

6.3.3.2 ShowSkinMovementOnImgManual

void ShowSkinMovementOnImgManual(IplImage* Coords,char MainImgName[],char ImgType[],bool SaveFile)

Η συνάρτηση ShowSkinMovementOnImgManual χρησιμοποιείται προκειμένου να δείξουμε τα αποτελέσματα της cvProcessImgSeq (και γενικότερα του εντοπισμού κίνησης του δέρματος) αλλά και αν επιθυμούμε να τα αποθηκεύσουμε. Τα αποτελέσματα είναι ο εντοπισμός σε κάθε frame του πού βρίσκεται το Κεφάλι το Αριστερό και το Δεξί χέρι. Η απεικόνιση των αποτελεσμάτων γίνεται πάνω σε μια GrayScale εικόνα της αρχικής όπου πάνω στο κέντρο του κάθε μέλους έχει δημιουργηθεί ένα γράμμα το οποίο προσδιορίζει την θέση αυτού του μέλους και έχουμε την δυνατότητα με αυτό τον τρόπο να διαπιστώσουμε αν ο εντοπισμός δερματικής κίνησης που έχει γίνει είναι σωστός βλέποντας στην πράξη αν όντος το Κεφάλι συμπίπτει με τις αντίστοιχες συντεταγμένες για το κεφάλι που δίνει το πρόγραμμα και αντίστοιχα τα χέρια.

IplImage* Coords = Πίνακας που περιέχει τις συντεταγμένες κεφαλιού και χεριών απο προηγούμενη εκτέλεση της συνάρτησης (αρχική τιμή είναι ο πίνακας γεμάτος με άσους)

char MainImgName[] = Το κυρίως όνομα των εικόνων.

char ImgType[]= Ο τύπος των αρχείων.

bool SaveFile = οταν είναι true έχουμε αποθήκευση των αποτελεσμάτων.

6.3.4 Συναρτήσεις επεξεργασίας εικόνων

Συναρτήσεις επεξεργασίας εικόνων είναι οι : cvTakeRowFromImage, cvMakeRowImagecvAddColumToImage, cvMakeImageCol, cvImReconstruct, cvBwLabel, cvMakeMyCloseFriendLikeMe, MakeRoiPOly, MakeEveryKToG

6.3.4.1 MakeRoiPOly

void MakeRoiPOly(IplImage* mask,int a[11][2])

Η συνάρτηση MakeRoiPOly εκτελεί μια πολύ σημαντική λειτουργία. Δεδομένης μιας εικόνας και ενός πίνακα ο οποίος περιέχει 10 συντεταγμένες σημείων δημιουργεί ένα δεκάγωνο πάνω στην εικόνα με βάση τις δέκα συντεταγμένες που έχει πάρει ως είσοδο. Τις τιμές αυτές του τετραγώνου τις κρατάει δημιουργώντας μια μάσκα η οποία έχει ένα μόνο στα σημεία του δεκάγωνου και μηδέν οπουδήποτε αλλού. Με την συνάρτηση αυτή ο χρήστης καλείται να δώσει την περιοχή μέσα στην οποία υπάρχει δέρμα και έτσι ο εντοπισμός δέρματος που θα κάνουμε θα είναι μεγάλης ακρίβειας. Το δεκάγωνο είναι ένας ικανοποιητικός χώρος ώστε να επιλεγεί μια κατάδηλη δερματική περιοχή σε μια εικόνα. Μεγαλύτερο πλήθος γωνιών είναι εφικτό αλλά δεν θα επιφέρει σημαντικές αλλαγές. Οι συντεταγμένες παίρνονται από την onmouse συνάρτηση η οποία θα αναλυθεί παρακάτω.

IplImage* mask = Μάσκα η οποία είναι ένα εκεί που έχει επιλέξει ο χρήστης ότι υπάρχει δέρμα

int a[11][2] = γωνίες δεκάγωνου.

6.3.4.2 cvMakeMyCloseFriendLikeMe

void cvMakeMyCloseFriendLikeMe(IplImage* Destination,int x,int y,int Labeler)

Η συνάρτηση cvMakeMyCloseFriendLikeMe παίρνει έναν πίνακα με μηδέν και ένα και τις συντεταγμένες ενός σημείου του πίνακα που είναι ένα και κάνει όλα τα γειτονικά σημεία που έχουν ένα με την τιμή του Labeled που δίνεται στην συνάρτηση. Η συνάρτηση χρησιμοποιεί αναδρομή καλώντας τον εαυτό της για τα 8 γειτονικά σημεία ενός pixel. Τα γειτονικά σημεία είναι :

(X -1, Y-1)	(X -1, Y)	(X-1 , Y+1)
(X , Y-1)	(X , Y)	(X , Y+1)
(X +1, Y-	(X +1, Y)	(X +1,

1)		Y+1)

IplImage* Destination = εικόνα που περιέχει μηδέν και ένα της οποία τα στοιχεία προσπαθούμε να χρωματίσουμε

int x = σημείο του οποίου τους γείτονες θέλουμε να αλλάξουμε τιμή.

int y= σημείο του οποίου τους γείτονες θέλουμε να αλλάξουμε τιμή.

int Labeled= η τιμή που θα πάρει το σημείο και οι γείτονες του.

6.3.4.3 cvImReconstruct

void cvImReconstruct(IplImage* A,IplImage* B,IplImage* Destination)

Η συνάρτηση cvImReconstruct επιτελεί μια πολύ σημαντική διεργασία και είναι ένα από τα πιο βασικά σημεία στον εντοπισμό κίνησης δέρματος. Δέχεται δύο πίνακες τον Α και τον Β. Οι πίνακες είναι λογικοί(με άσους και μηδενικά). Η συνάρτηση δημιουργεί έναν νέο πίνακα ο οποίος είναι και αυτός λογικός και έχει ένα μόνο στις περιοχές όπου υπάρχουν στον Α και σε αυτές του Β που η τομή τους με μια περιοχή του Α δεν δίνει κενό σύνολο. Η σημαντικότητα της συνάρτησης έγκειται στο ότι με αυτήν την λειτουργία που εκτελεί από έναν πίνακα που εντοπίζει την κίνηση και από έναν που εντοπίζει το δέρμα μας δίνει έναν που εντοπίζει το κινούμενο δέρμα πράγμα που είναι και το ζητούμενο του όλου προγράμματος.

IplImage* A = Η εικόνα με την βασική προτεραιότητα. Οι άσοι της εικόνας αυτής θα είναι σίγουρα και στην τελική.

IplImage* B = Η εικόνα της οποία οι άσοι θα υπάρχουν στην τελική εικόνα μόνο και μόνο αν υπάρχει τομή με κάποια περιοχή του πίνακα Α.

IplImage* Destination= Ο πίνακας στο οποίο αποθηκεύονται τα αποτελέσματα της συνάρτησης.

6.3.4.4 cvMakeImageCol

IplImage* cvMakeImageCol(IplImage* Source)

Η συνάρτηση cvMakeImageCol μετατρέπει έναν δισδιάστατο πίνακα σε μία στήλη όπου ουσιαστικά έχουμε την προσάρτηση της κάθε στήλης στην επόμενη. Το OpenCV δεν παρέχει κάποια έτοιμη συνάρτηση που να κάνει την αντίστοιχη διεργασία. Η συνάρτηση αυτή χρησιμοποιείται πολύ από το πρόγραμμα μου μιας και λόγο των πολλαπλών γινόμενο πινάκων επιβάλλεται να μετατρέπω δισδιάστατους πίνακες σε μονοδιάστατους και αντίστροφα.

IplImage* Source = ο δισδιάστατος πίνακα της εικόνας που θα μετατραπεί σε στήλη.

7. Εγκατασταση Android & OpenCV

Στο κεφαλαίο αυτό θα μιλήσουμε για τον τρόπο δημιουργίας μιας εφαρμογής Android και για την εισαγωγη της βιβλιοθήκης OpenCV στην εφαρμογή μας με σκοπό την επεξεργασία εικόνας.

Θα κανουμε μια αναλυτική περιγραφή βήμα προς βήμα για την αρχική εγκατάσταση του περιβάλλον ανάπτυξης για τον τρόπο που δημιουργούμαι μια εικονική συσκευή Android (Android Virtual Device) ..

Αρχική εγκατάσταση του περιβάλλον ανάπτυξης

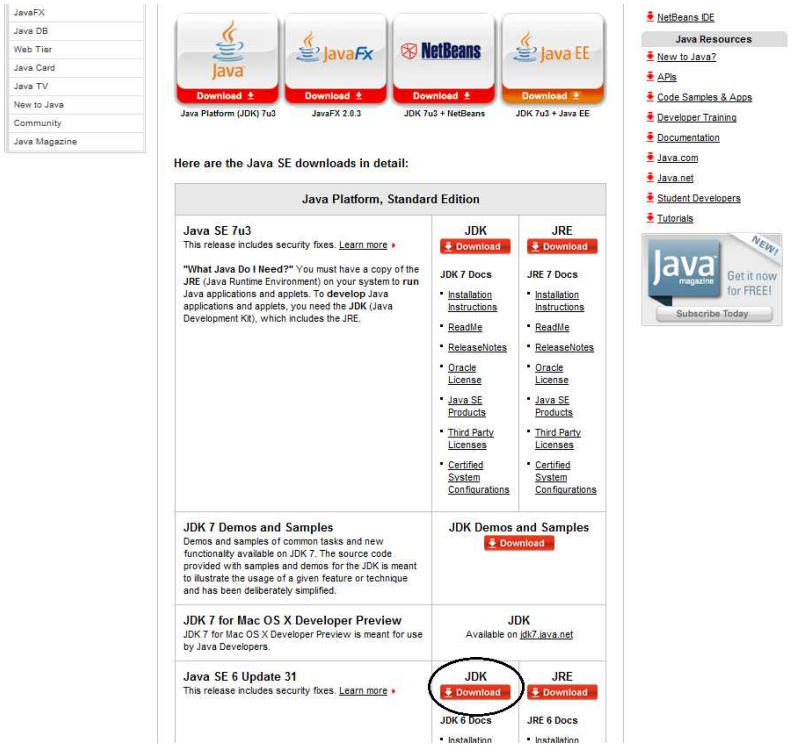
Προκειμένου να αναπτύξουμε εφαρμογές Android , πρέπει να έχουμε το παρακάτω λογισμικό στον υπολογιστή μας:

- Το πακέτο ανάπτυξης Java Development Kit(JDK) στης εκδόσεις 5 ή 6, το όποιο μπορείτε να το λάβετε απ' τη σελίδα http://www.oracle.com/technetwork/java/javase/downloads/index.html .
- Ένα συμβατό περιβάλλον ανάπτυξης με το Java IDE όπως το Eclipse, μαζί με το πρόσθετο πρόγραμμά του JDT, το οποίο μπορείτε να λάβετε από τη σελίδα http://www. eclipse.org/downloads/
- Το Android SDK, τα εργαλεία και την τεκμηρίωση του, τα οποία μπορείτε να λάβετε απ' τη σελίδα http://developer.android.com/sdk/index.html.
- Το πρόσθετο πρόγραμμα Android Development Tools (ADT) για το Eclipse, το οποίο μπορείτε να λάβετε μέσω του μηχανισμού ανανέωσης λογισμικού του Eclipse. Για οδηγίες για το πώς θα εγκαταστήσετε αυτό το πρόσθετο πρόγραμμα, επισκεφτείτε τη σελίδα http://developer.android.com/sdk/eclipse-adt.html . Αν και αυτό το εργαλείο είναι προαιρετικό για την ανάπτυξη, το προτείνουμε ανεπιφύλακτος ενώ ο' αυτό το βιβλίο θα δείτε ότι το χρησιμοποιούμε συχνά.

Μία ολοκληρωμένη λίστα με τις απαιτήσεις συστημάτων για ανάπτυξη με το Android μπορείτε να βρείτε στη σελίδα http://developer.android.com/sdk/requirements.html . Οδηγίες εγκατάστασης θα βρείτε στη σελίδα http://developer.android.com/sdk/installing.html .

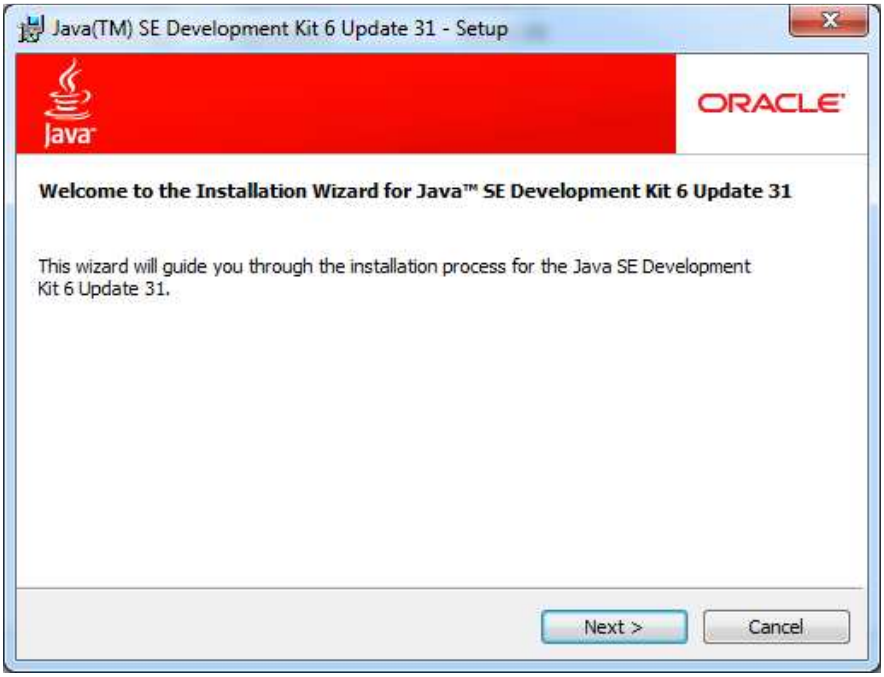
7.1 Java Development Kit(JDK)

Καταρχάς κατεβάζουμε το JDK.

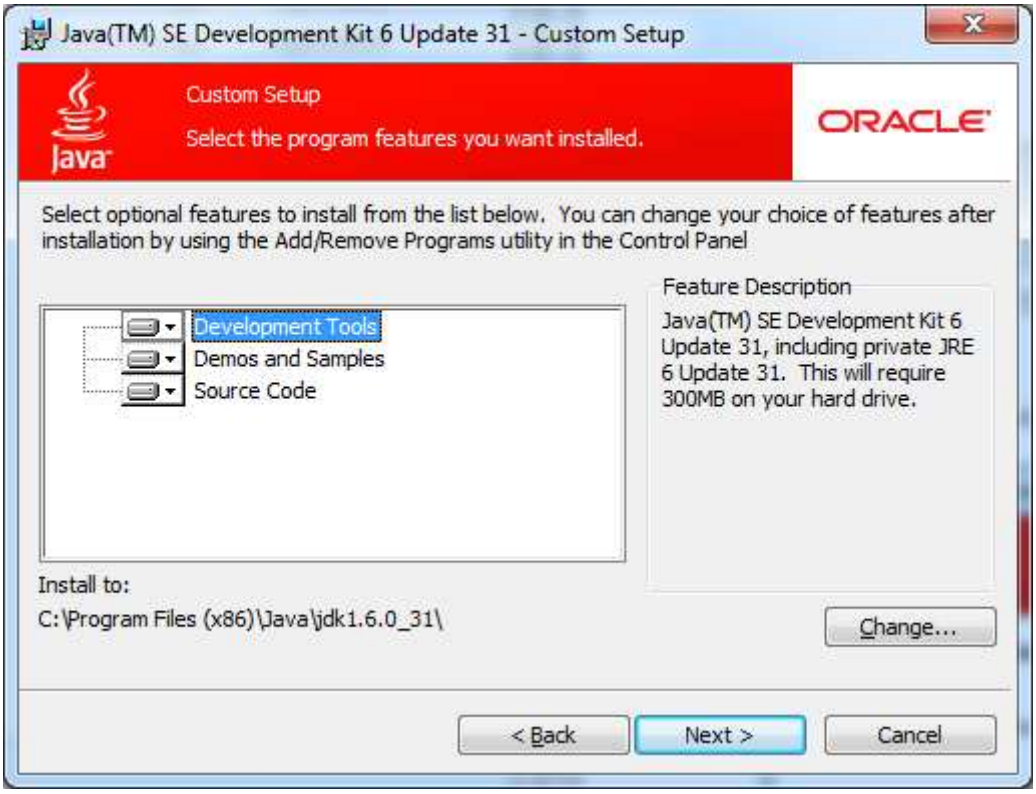


Εικόνα 21

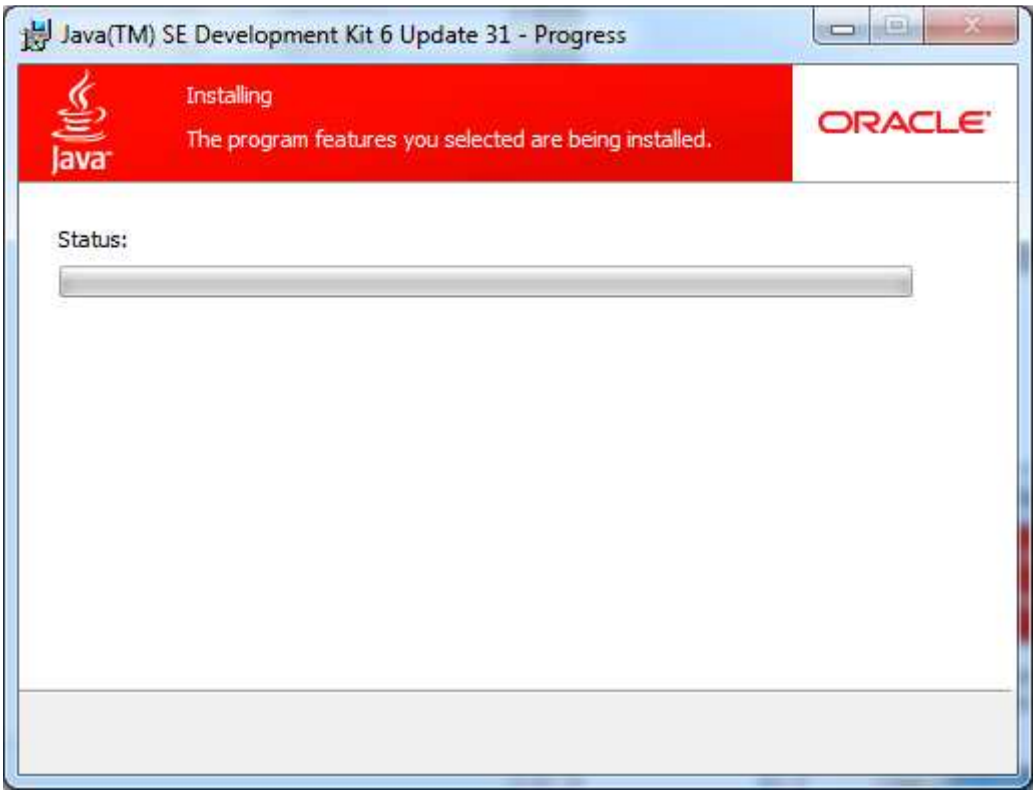
Και στην συνέχεια το εγκαθιστούμε.



Εικόνα 22



Εικόνα 23



Εικόνα 24

Και έχουμε εγκαταστήσει επιτυχώς το Java Development Kit(JDK). [4]

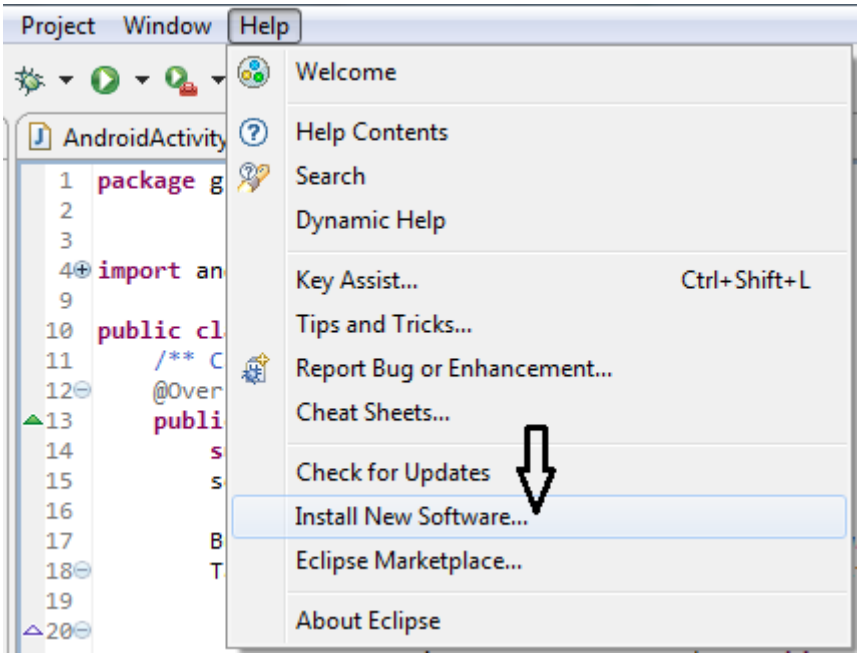
7.2 Eclipse

Πάμε στην σελίδα http://www.eclipse.org/downloads και κατεβάζουμε το Eclipse IDE for Java EE Developers.

Εικόνα 25 eclipse download

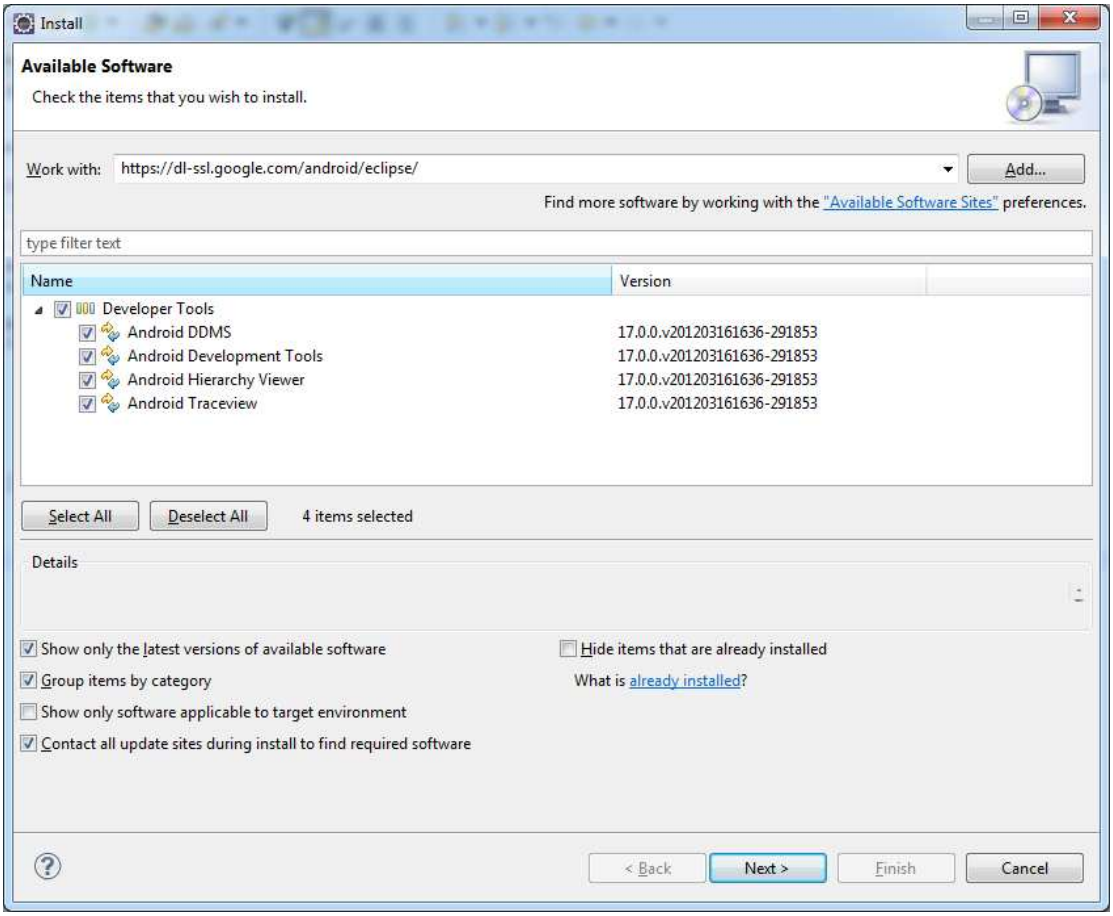
Μετά αποσυμπιέζουμε το zip αρχείο που κατεβάσαμε και το τρέχουμε.

Στην συνέχεα πρέπει να εγκαθηστησουμαι το ADT για το Android και πάμε στο eclipse Help→ Install New Software.



Εικόνα 26 Instal New software

Στο Work with πληκτρολογούμαι https://dl-ssl.google.com/android/eclipse/ και μετά next



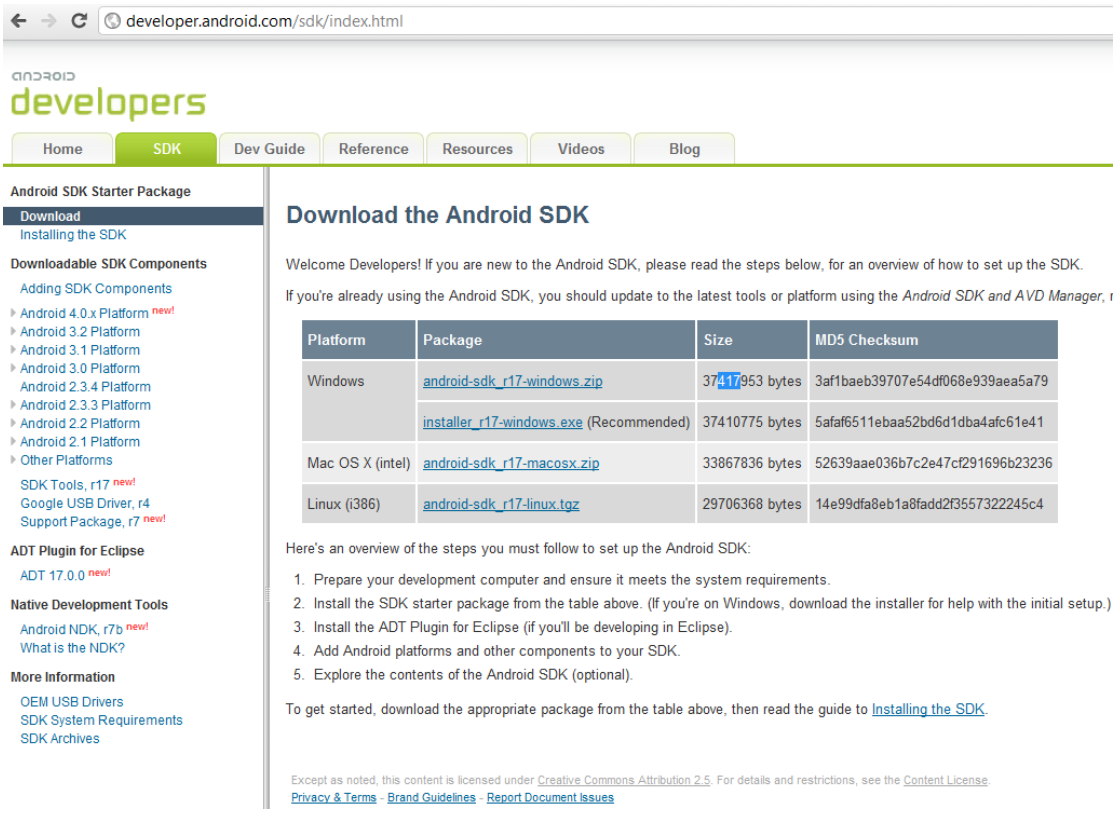
Εικόνα 27 developer tools

Κάνει μια επανεκκίνηση τα eclipse μας και είναι έτοιμο. [4]

7.3 Android SDK

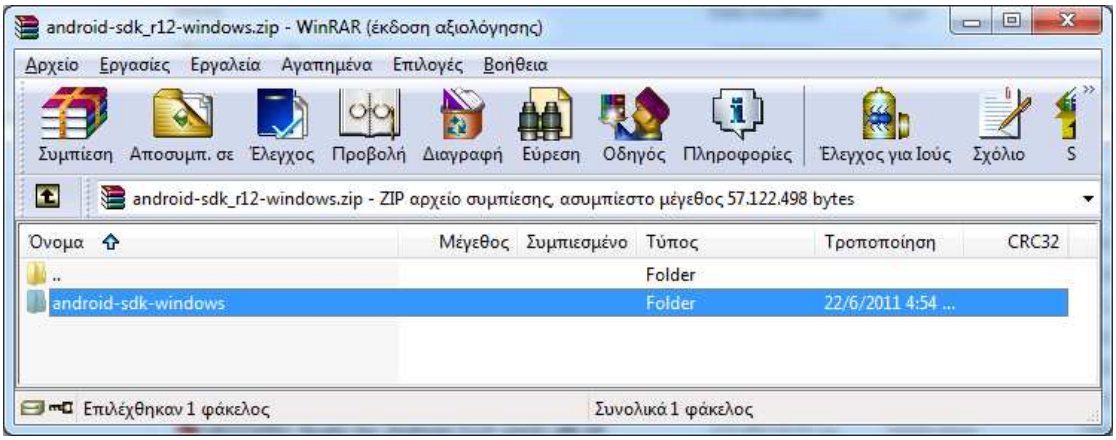
Τέλος πρέπει να εγκαταστήσουμε και το Android SDK

Πάμε στην σελίδα http://developer.android.com/sdk/index.html και κατεβάζουμε την έκδοση SDK που μας ενδιαφέρει.



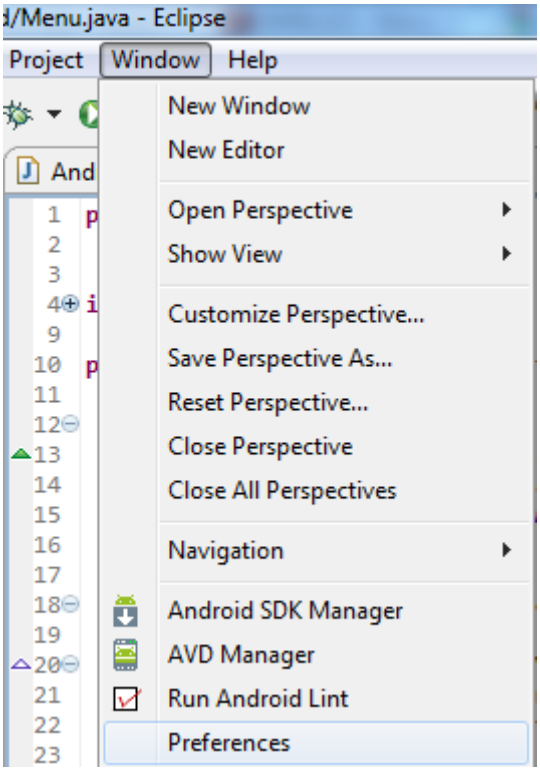
Εικόνα 28 SDK download

Αποσυμπιεζουμε το zip αρχειο μας και μεταφερουμαι τον φακελο android-sdk στον σκληρο μας δισκο στο C:/



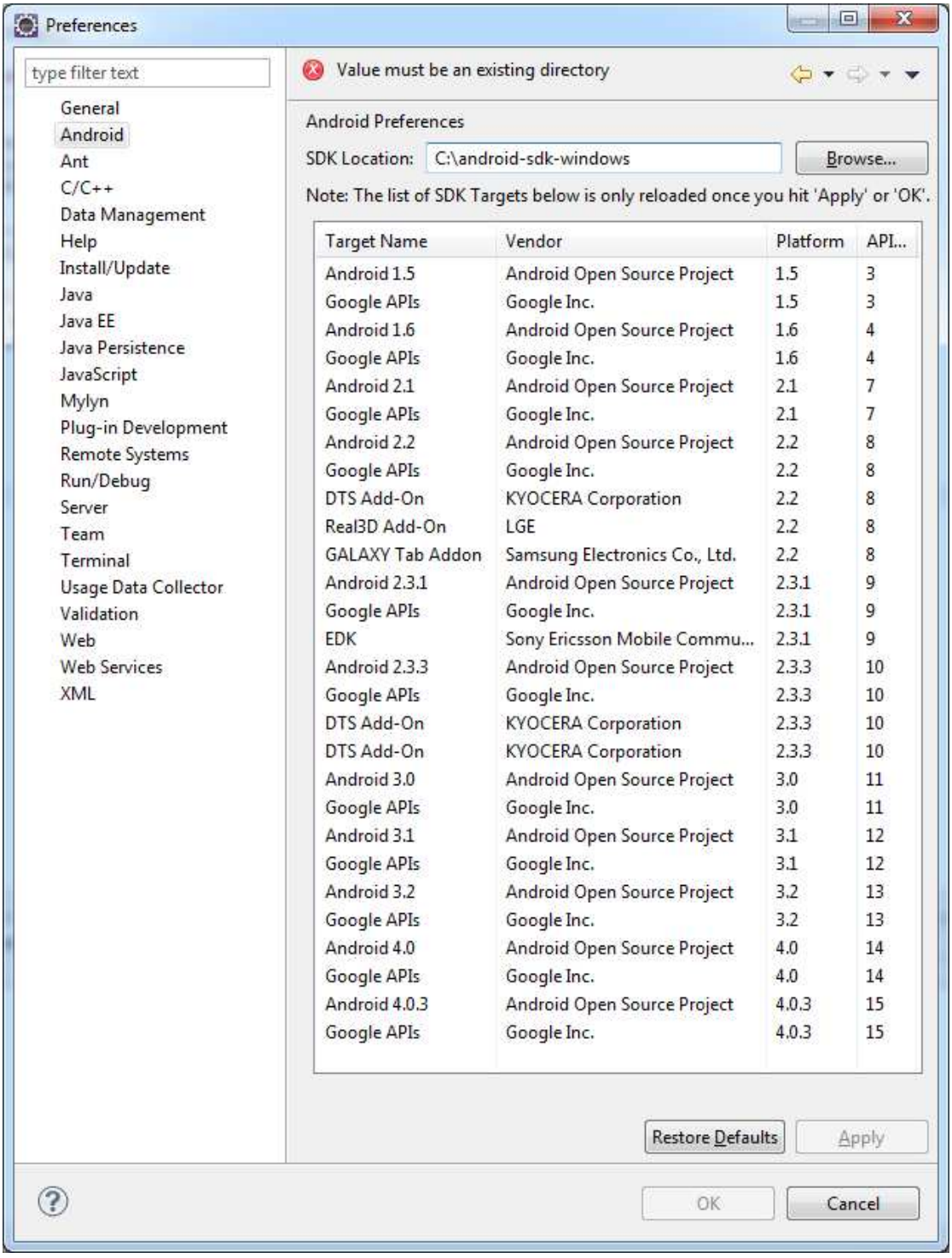
Εικόνα 29 SDK unzip

Μετά πάμε στο eclipse Window→Preferences



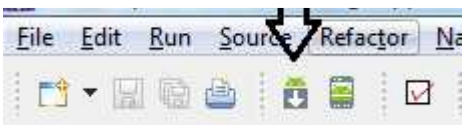
Εικόνα 30 preference

Από τα αριστερά μας επιλέγουμε το Android και εκεί που λέει SDK Location επιλέγουμε την διαδρομή του φάκελου που αποσυμπιέσαμε προηγουμένως και πατάμε OK



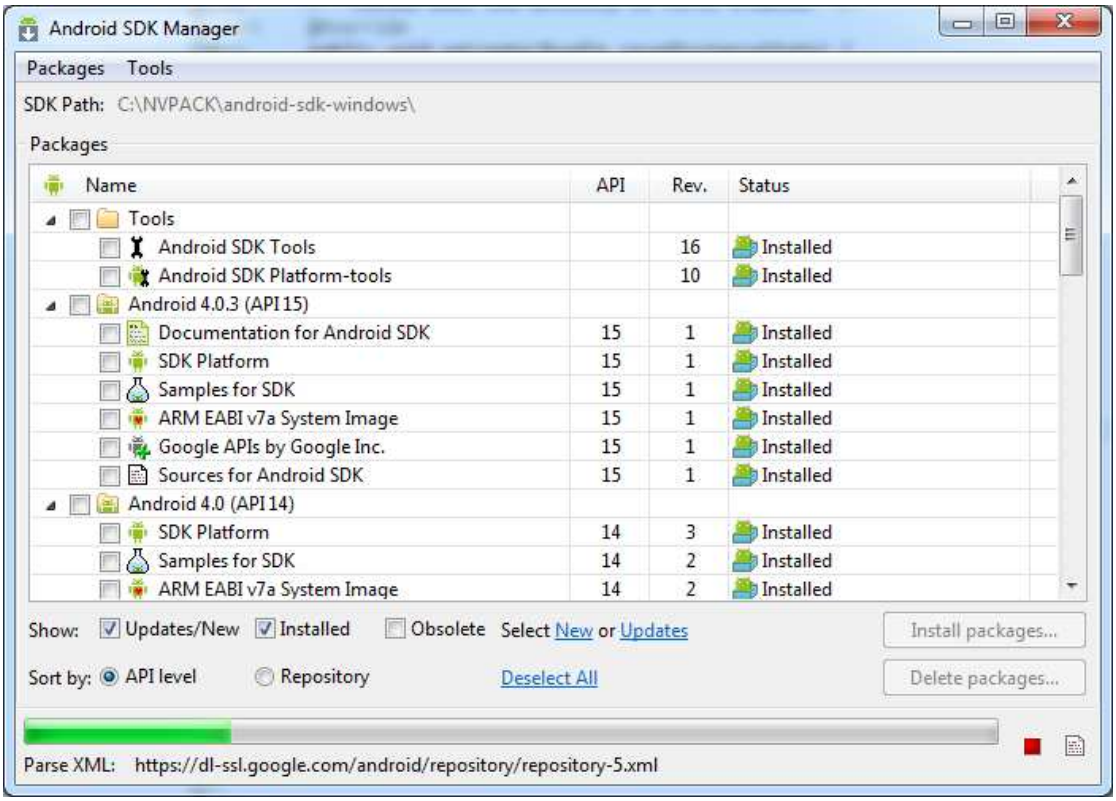
Εικόνα 31 εκδοσεις

Στην συνέχεια πάμε να εγκαταστήσουμε της εκδόσεις Android που μας ενδιαφέρουν πατώντας από το eclipse το εικονίδιο .



Εικόνα 32 προγραμμα SDK

Και μετά εγκαθιστάμε τα package που μας ενδιαφέρουν .

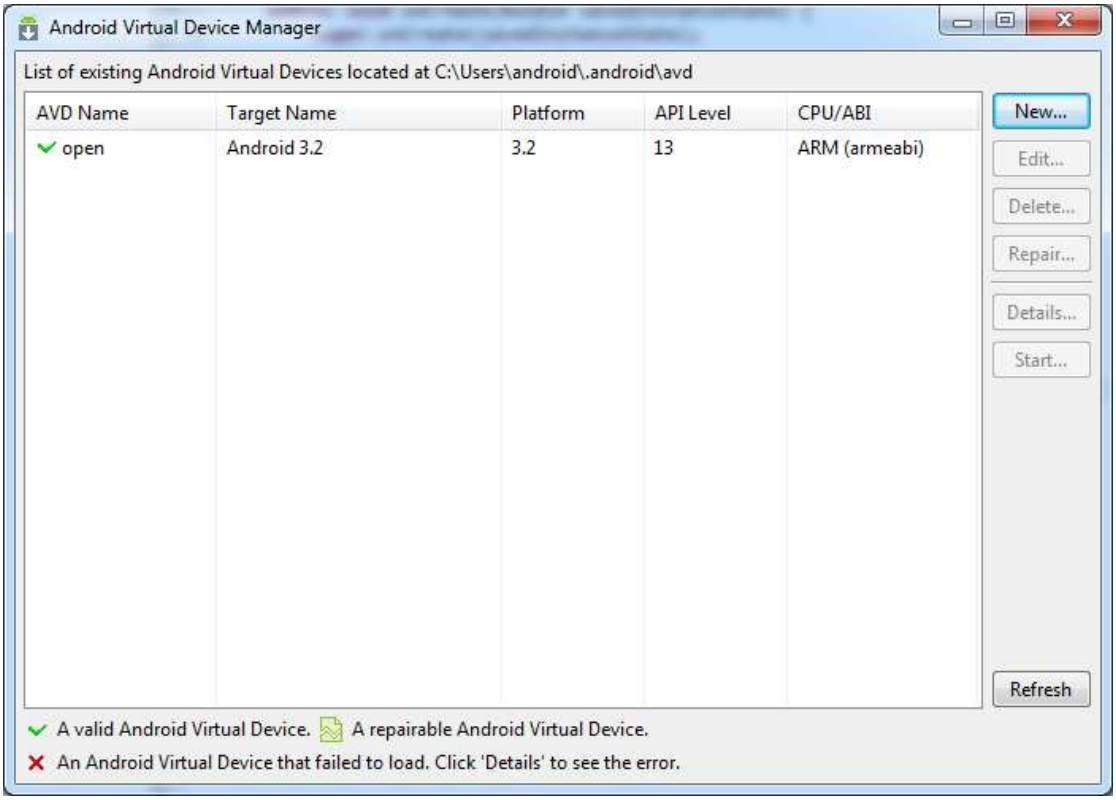


Εικόνα 33 SDK Manager

Και τέλος για να εγκαταστήσουμε μια εικονική συσκευή για να τρέχουμε της εφαρμογές μας από τον υπολογιστή μας πατάμε από το eclipse το εικονίδιο

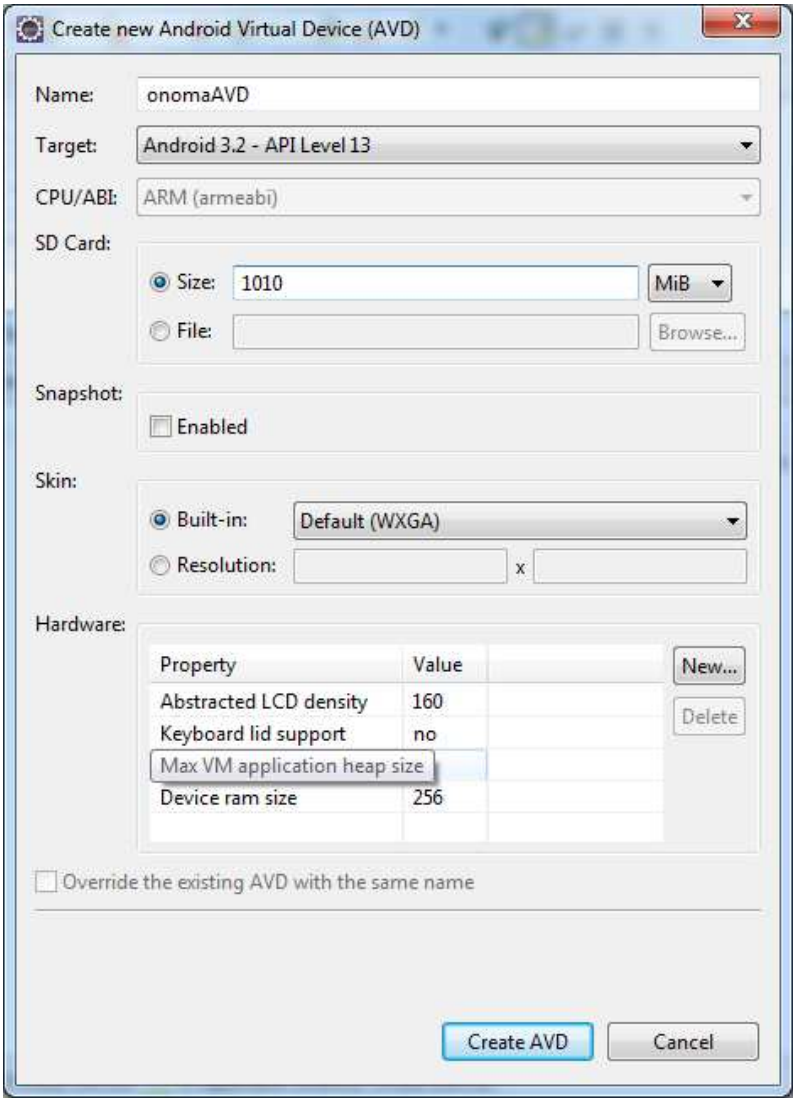
Εικόνα 34 Εικονικη συσκευη

Και στο παράθυρο που θα μας εμφανιστεί πατάμε New



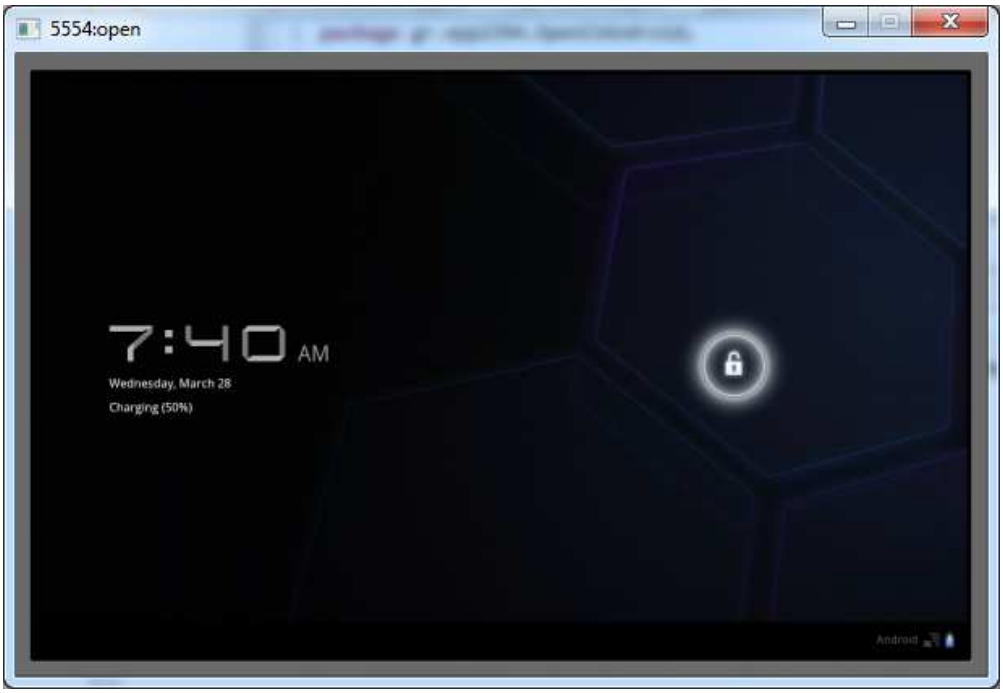
Εικόνα 35 εικονικες συσκευες

Και μετά πληκτρολογούμαι το όνομα του AVD ποια έκδοση θα υποστηρίζει και το μέγεθος της SD Card.



Εικόνα 36 ρυθμησεις συσκευης

Και από κάτω θα δείτε πώς εμφανίζετε στον υπολογιστή μας.

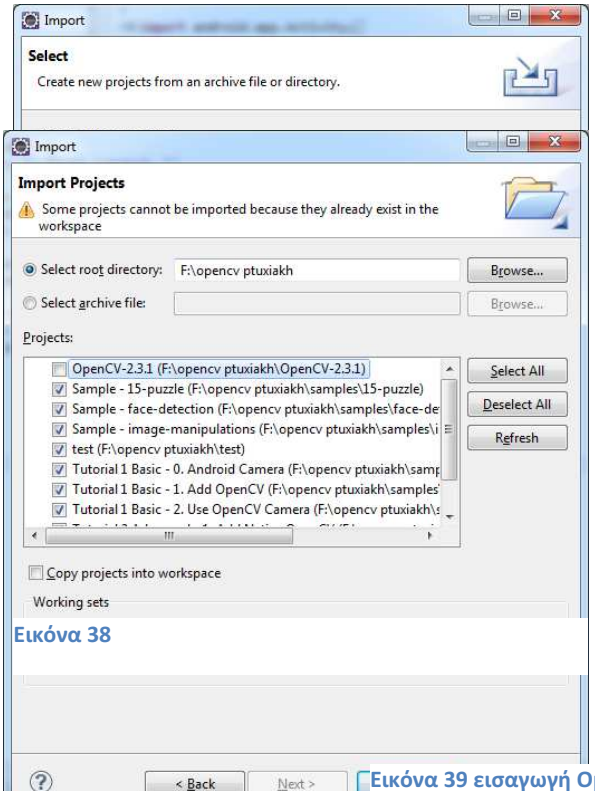


Εικόνα 37 preview

Εμείς στην συνέχεια δεν θα χρησιμοποιήσουμε εικονική συσκευή στο πρόγραμμα μας αλλά θα δουλέψουμε πάνω στην συσκευή Motorola Xoom. [4]

7.4 Εισαγωγή Βιβλιοθήκης OpenCV

Καταρχάς πρέπει να κατεβάσουμε την βιβλιοθήκη μας από την σελίδα http://sourceforge.net/projects/opencvlibrary/files/opencv-android/ . Αποσυμπιέζουμε το αρχείο και με το αντιγραφούμε στον φάκελο Opencv που τον δημιουργούμε στον C:/.



Εικόνα 38

Μετά πάμε στο eclipse και πάμε File→import… και στον φάκελο General επιλέγουμε Existing Projects into Workspace

Πατάμε το Browse.. και επιλέγουμε τον φάκελο του

Εικόνα 39 εισαγωγή OpenCV

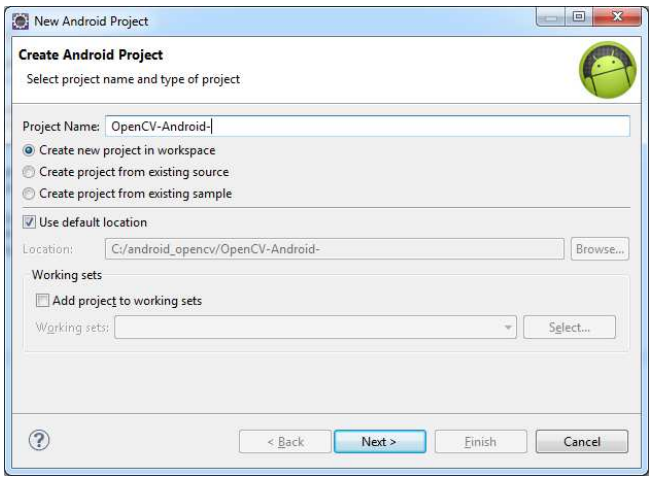
OpenCV που κατεβάσαμε προηγουμένως και πατάμε finish.

8. Η εφαρμογή μας

Στο κεφάλαιο θα δούμε πως θα δημιουργήσουμε μια εφαρμογή Android. Η όποια αρχικά θα μας μεταφέρει στην κάμερα της συσκευή μας, και στην συνέχεια όταν τραβήξουμε μια φωτογραφία να μας την μεταφέρει στην αρχική σελίδα της εφαρμογής μας. Με την βοήθεια της βιβλιοθήκης της OpenCV να επεξεργαζόμαστε την εικόνα και να της προσθέτουμε διάφορα φίλτρα, χρωματισμούς ακόμα και να αλλάζουμε το μέγεθος της με ένα απλό κλικ.

8.1 Δημιουργία νέου project και εισαγωγή βιβλιοθήκης

Αρχικά πάμε από το eclipse File→New→Android project

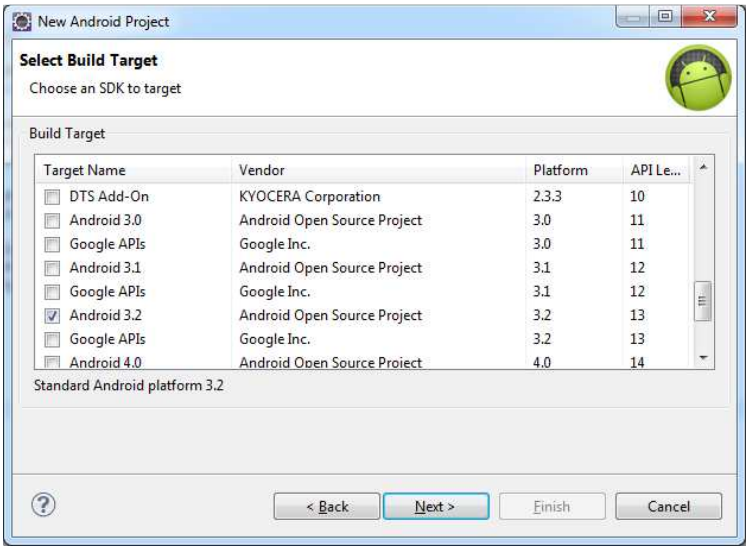


Εικόνα 40 new project 1

Στο Project Name γράφουμε

OpenCV-Android επιλέγουμε το Create new project in workspace και τα υπόλοιπα ως έχουν και πατάμε Next.

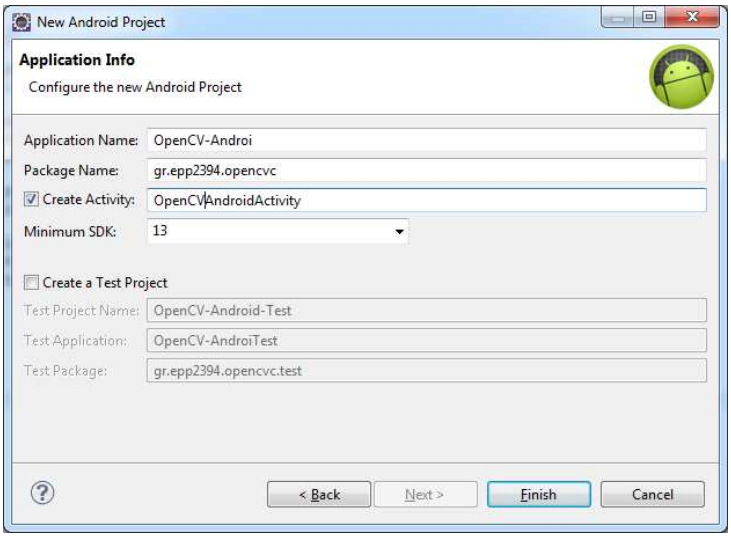
Στο επόμενο παράθυρο που θα μας εμφανιστεί επιλέγουμε για την έκδοση με την οποία θέλουμε να τρέχει η εφαρμογή μας.



Εικόνα 41new project 2

Στην εφαρμογή μας επιλέγουμε την έκδοση 3.2 και μετά Next

Στο επόμενο και τελευταίο παράθυρο που θα μας εμφανιστεί δίνουμε το όνομα της εφαρμογής μας το όνομα του package, το όνομα της πρώτης μας class η οποία συνήθως είναι τύπου activity και για της ελάχιστη έκδοση Android που μπορεί να τρέξει η εφαρμογή μας.



Εικόνα 42new project 3

Για Application Name βάζουμε OpenCV-Android

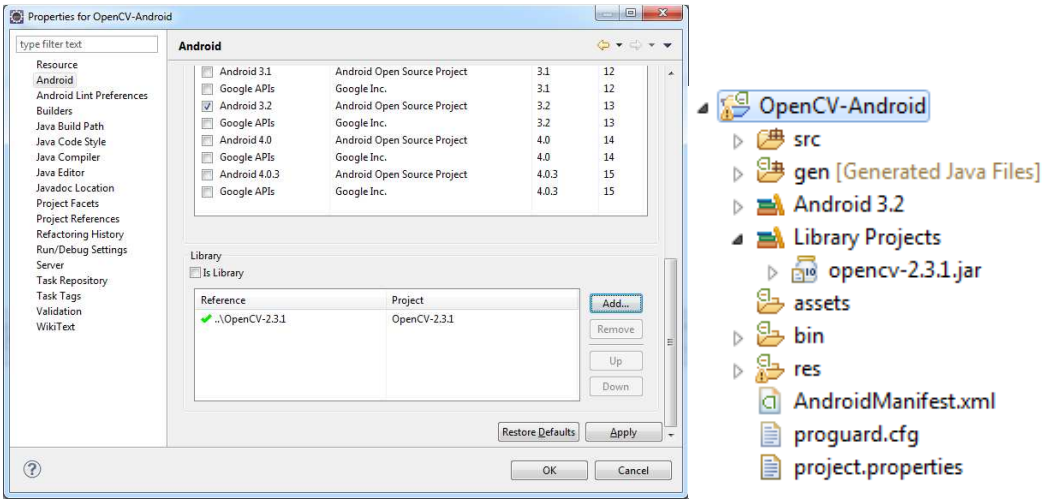
Package Name gr.epp2394.opencv

Όνομα class OpenCVAndroidActivity

Minimum SDK : 13

Και έτσι δημιουργήσαμε την εφαρμογή μας η οποία ακόμα δεν κάνει τίποτα.

Τώρα θέλουμε να προσθέσουμε την βιβλιοθήκη OpenCV στην εφαρμογή μας. Πάμε αριστερά στο eclipse και εκεί που έχει όλα τα projects πάμε στο project μας και πατάμε δεξί κλικ και properties. Στο παράθυρο που θα μας εμφανιστεί από τα αριστερά επιλέγουμε την καρτέλα Android και κάτω δεξιά πατάμε το κουμπί add και επιλέγουμε την βιβλιοθήκη μας. Και τέλος πατάμε Ok και προσθέτουμε την βιβλιοθήκη μας στην εφαρμογή μας.



Εικόνα 43 εισαγωγή βιβλιοθήκης

8.2 Χρήση κάμερας της συσκευής και εξαγωγή εικόνας στης εφαρμογή.

Αρχικά πάμε στην OpenCVAndroidActivity και γράφουμε τον παρακάτω κώδικα

```
1  package gr.epp2394.OpenCVAndroid;
2
3  import android.app.Activity;
4  import android.content.Intent;
5  import android.os.Bundle;
6
7  public class OpenCVAndroidActivity extends Activity {
8      /** Called when the activity is first created. */
9      @Override
10     public void onCreate(Bundle savedInstanceState) {
11         super.onCreate(savedInstanceState);
12         setContentView(R.layout.main);
13         Intent intent = new Intent(OpenCVAndroidActivity.this, Camera1.class);
14         startActivity(intent);
15
16     }
17 }
```

Στην γραμμή 13 δημιουργούμε ένα Intent όπου μας μεταφέρει στην class Camera1

Στην γραμμή 14 τρέχουμε αυτό το intent.

Μετά πάμε στο Androidmanifest.xml και δηλώνουμε ότι κάνουμε χρήσης ενός άλλου πόρου της συσκευής ώστε να έχουμε δικαιώματα πρόσβασης στην κάμερα. [2]

```
27 <action android:name="android.media.action.IMAGE_CAPTURE" />
28 <category android:name="android.intent.category.DEFAULT" />
```

Δημιουργούμε την Class Camera1 που είναι και αυτή τύπου Activity και γράφουμε τον παρακάτω κώδικα.

```
import java.io.File;

import android.app.Activity;
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.net.Uri;
import android.os.Bundle;
import android.os.Environment;
import android.util.Log;
import android.view.Display;
import android.widget.ImageView;

public class AndroidActivity extends Activity {

    final static int CAMERA_RESULT = 0;

    ImageView imv;
    String imageFilePath;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        imageFilePath = Environment.getExternalStorageDirectory()
                .getAbsolutePath() + "/myfavoritepicture.jpg";
        File imageFile = new File(imageFilePath);
        Uri imageFileUri = Uri.fromFile(imageFile);
        Intent i = new Intent(android.provider.MediaStore.ACTION_IMAGE_CAPTURE);
        i.putExtra(android.provider.MediaStore.EXTRA_OUTPUT, imageFileUri);
        startActivityForResult(i, CAMERA_RESULT);
    }

    protected void onActivityResult(int requestCode, int resultCode,
            Intent intent) {
        super.onActivityResult(requestCode, resultCode, intent);
```

Εικόνα 44 κώδικας

```
37
38          if (resultCode == RESULT_OK) {
39              // Get a reference to the ImageView
40              imv = (ImageView) findViewById(R.id.ReturnedImageView);
41              Display currentDisplay = getWindowManager().getDefaultDisplay();
42              int dw = currentDisplay.getWidth();
43              int dh = currentDisplay.getHeight();
44              // Load up the image's dimensions not the image itself
45              BitmapFactory.Options bmpFactoryOptions = new BitmapFactory.Options();
46              bmpFactoryOptions.inJustDecodeBounds = true;
47              Bitmap bmp = BitmapFactory.decodeFile(imageFilePath,
48                      bmpFactoryOptions);
49              int heightRatio = (int) Math.ceil(bmpFactoryOptions.outHeight
50                      / (float) dh);
51              int widthRatio = (int) Math.ceil(bmpFactoryOptions.outWidth
52                      / (float) dw);
53              Log.v("HEIGHTRATIO", "" + heightRatio);
54              Log.v("WIDTHRATIO", "" + widthRatio);
55              // If both of the ratios are greater than 1,
56              // one of the sides of the image is greater than the screen
57              if (heightRatio > 1 && widthRatio > 1) {
58                  if (heightRatio > widthRatio) {
59                      // Height ratio is larger, scale according to it
60                      bmpFactoryOptions.inSampleSize = heightRatio;
61                  } else {
62                      // Width ratio is larger, scale according to it
63                      bmpFactoryOptions.inSampleSize = widthRatio;
64                  }
65              }
66              // Decode it for real
67              bmpFactoryOptions.inJustDecodeBounds = false;
68              bmp = BitmapFactory.decodeFile(imageFilePath, bmpFactoryOptions);
69              // Display it
70              imv.setImageBitmap(bmp);
71          }
72      }
73  }
```

Εικόνα 45 κωδικας

Στην γραμμή 16 αρχικοποιούμε μια final static int μεταβλητή [2]

18 δημιουργούμε το imv που είναι τύπου ImageView όπου και στην γραμμή 40 θα πάρει την θέση από το ImageView που θα δημιουργήσουμε αργότερα στο xml αρχείο μας

19 δημιουργούμε ένα imageFilePath τύπου string για να του δώσουμε αργότερα στην γραμμή 25 την θέση όπου θα αποθηκευτή η εικόνα μας.

27 δημιουργούμε ένα imageFile που είναι τύπου File και βρίσκετε στην διεύθυνση που έχουμε δώσει στο imageFilePath

28 δημιουργούμε ένα imageFIleUri τύπου Uri το οποίο θα διώχνει στο imageFile μας.

29-31 δημιουργούμε ένα intent το οποίο μας μεταφέρει στην κάμερα μας και το αποτέλεσμα της καμερας (της φωτογραφίας) τα αποθηκεύει εκεί που του διώχνει το imageFileUri.

41-43 παίρνουμε τις διαστάσεις της εικόνας μας

45-68 περνάμε την εικόνα μας και μέσω της BitmapFactory.Option μπορούμε και αλλάζουμε τις ρυθμίσεις της εικόνας μας και της δίνουμε το κανονικό μέγεθος της

70 εμφανίζεται η εικόνα μας στο ImageView που θα δηλώσουμε παρακάτω στο xml αρχείο μας

Μετά πάμε στον φάκελο res→layout→main.xml του project μας και πληκτρολογούμε τον παρακάτω κώδικα

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
>
<ImageView android:id="@+id/ReturnedImageView" android:layout_width="wrap_content"..
android:layout_height="wrap_content"></ImageView>
</LinearLayout>
```

Εικόνα 46 xml

Δημιουργούμε ένα linearLayout στο οποίο θα έχει οριζόντια διάταξη και μέσα σε αυτό τοποθετούμε το ImageView μας με id=ReturnedImageView. Στο οποίο θα μας εμφανίσει την εικόνα που βγάλαμε από την κάμερα προηγουμένως.

8.3 Δημιουργία ImageButton, Button, Seekbar, EditText

8.3.1 Button

Μπορούμε στα buttons(κουμπιά) να δώσουμε κάποιες λειτουργιές όταν πατιούνται.

Καταρχάς πρέπει να δηλώσουμε τα κουμπιά στο xml αρχείο μας και να τους δώσουμε όνομα (id) διαστάσεις και το κείμενο που θα γράφει το κουμπί

```
<Button
    android:id="@+id/size1"
    android:layout_width="100dp"
    android:layout_height="wrap_content"
    android:text="1" />
```

Εικόνα 47 xml Button

Αφού το δηλώσουμε στο xml αρχείο μας πρέπει να το δηλώσουμε και στην κλάση μας. Και όταν το δηλώσουμε πρέπει να του πούμε και ποια είναι ακριβώς η λειτουργιά του.

```
Button fl2button = (Button) findViewById(R.id.fl2button);
Button fl3button = (Button) findViewById(R.id.fl3button);
Button transposebutton = (Button) findViewById(R.id.transpose);
```

Εικόνα 48 Java Button

```
fl2button.setOnClickListener(new View.OnClickListener() {

    public void onClick(View v) {
        mRgba = flip(0);
        bmpOut = Bitmap.createBitmap(mRgba.cols(), mRgba.rows(),
                Bitmap.Config.ARGB_8888);
        Utils.matToBitmap(mRgba, bmpOut);
        imv.setImageBitmap(bmpOut);

    }
});
fl3button.setOnClickListener(new View.OnClickListener() {

    public void onClick(View v) {
        mRgba = flip(1);
        bmpOut = Bitmap.createBitmap(mRgba.cols(), mRgba.rows(),
                Bitmap.Config.ARGB_8888);
        Utils.matToBitmap(mRgba, bmpOut);
        imv.setImageBitmap(bmpOut);
    }
});
```

Εικόνα 49 Java onclick

[5]

8.3.2 ImageButton

Το imageButton μοιάζει παρά πολύ με το απλό Button με την μόνη διαφορά ότι αντί ένα απλό κουμπί μπορούμε να προσθέσουμε μια εικόνα πάνω σε αυτό.

```
<ImageButton
    android:id="@+id/b7"
    android:layout_width="150px"
    android:layout_height="120px"
    android:src="@drawable/chestnut1" />
```

Εικόνα 50 xml ImageButton

Το src είναι όταν θέλουμε να προσθέσουμε εικόνα και το συγκεκριμένο μας λέει να πάρουμε την εικόνα μας από τον φάκελο drawable.

Αλλάζει και η δήλωση λίγο

```
ImageButton b6 = (ImageButton) findViewById(R.id.b6);
ImageButton b7 = (ImageButton) findViewById(R.id.b7);
```

Εικόνα 51 Java ImageButton

[5]

8.3.3 Seekbar

Το seekbar είναι μια μπάρα η οποία παίρνει τιμές από ένα σύνολο τιμών που εμείς αρχικοποιούμε και χρησιμεύει για τον έλεγχο με μεγαλύτερη ακρίβεια.

Εικόνα 52 SeekBar

Η δήλωση στο xml αρχείο είναι αρκετά απλή.

```
<SeekBar
    android:id="@+id/green"
    android:layout_width="164dp"
    android:layout_height="fill_parent"
    android:max="255" />
```

Εικόνα 53 xml SeekBar

Την μόνη διαφορά από τα Buttons είναι ότι υπάρχει μια μέγιστη τιμή max και θα μας λέει μέχρι πόσο θα πάει η μπάρα μας. Να αναφέρουμε ότι η τιμή του ελάχιστου μπορεί να αλλάξει και αυτή. Στο παράδειγμα μας παίρνει default τιμή που είναι 0.

Όσο αφορά τώρα την δήλωση της στην java είναι η ακόλουθη:

```
SeekBar grseek = (SeekBar) findViewById(R.id.green);
```

Εικόνα 54 java SeekBar

Δημιουργούμε ένα grseek που είναι τύπου Seekbar το οποίο θα συνδέεται με το xml αρχείο μας με αυτό το Seekbar που έχει id green.

Για τον τρόπο λειτουργίας του τώρα:

```
grseek.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {

    public void onStopTrackingTouch(SeekBar seekBar) {
        // TODO Auto-generated method stub

    }

    public void onStartTrackingTouch(SeekBar seekBar) {
        // TODO Auto-generated method stub

    }

    public void onProgressChanged(SeekBar seekBar, int progress,
            boolean fromUser) {
        im.RGB(im.getRed(), progress, im.getBlue());

        show();

    }
});
```

Εικόνα 55 java change SeekBar

Γράφουμε ένα setOnSeekBarChangeListener ο όποιος στην συνάρτηση onProgressChanged τρέχει οπότε κουνήσουμε την μπάρα μας για να κάνει και την αντίστοιχη κίνηση που του πούμε. [5]

8.3.4 EditText

Όσο αφορά για το EditText μας είναι ένας χώρος που τον δηλώνουμε εμείς και έχουμε την δυνατότητα αν εισάγουμε ένα κείμενο ενώ τρέχει η εφαρμογή μας.

Εικόνα 56 EditText

Όσο αφορά το xml αρχείο μας τα μονά που χρειάζονται είναι id διαστάσεις και αν θέλουμε ένα αρχικό κείμενο.

```
<EditText
    android:id="@+id/namepic"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="nameImage" />
```

Εικόνα 57 xml EditText

Για την δήλωση στην Java επίσης δεν θέλουμε να γράψουμε και πολλά.

Μόνο την δήλωση του EditText για την σύνδεση με το xml μας και να αναφέρουμε σε τι μας χρησιμεύει το EditText που μόλις δημιουργήσαμε. Στην προκείμενη περίπτωση το έχουμε για να πάρουμε το κείμενο που γράψαμε και να το μετατρέψουμε σε ένα String. [5]

```
saveName = (EditText) findViewById(R.id.namepic);
```

Εικόνα 58 java EdtiText

```
String filename= saveName.getText().toString();
```

Εικόνα 59 String

8.4 OpenCV

Παρακάτω θα δούμε την λειτουργία της βιβλιοθήκης της OpenCV. Πώς μετατρέπουμε ένα Bitmap σε Mat(της OpenCV) και αντιστρόφως, πως βάζουμε ένα χρωματισμό της επιλογής μας σε μια εικόνα, πώς μπορούμε να αυξήσουμε την φωτεινότητα της εικόνας μας, πως μπορούμε να αλλάξουμε τις διαστάσεις της εικόνας και την εισαγωγή διαφόρων φίλτρων.

8.4.1 Bitmap σε Mat

Για να μπορέσουμε να δουλέψουμε με την βιβλιοθήκη της OpenCV πρέπει πρώτα να μετατρέψουμε την Bitmap εικόνα μας σε Mat και αυτό γίνετε με τον παρακάτω κώδικα:

```
import org.opencv.android.Utils;

import org.opencv.core.Mat;

public Mat imgToProcess;

 public Mat imageToMat(Bitmap img) {
     imgToProcess = Utils.bitmapToMat(img);
     return imgToProcess;
 }
```

Εικόνα 60 Bitmap to Mat

Καταρχάς πρέπει να δηλώσουμε την βιβλιοθήκη που θα χρησιμοποιήσουμε για να μπορεί το πρόγραμμα μας να αναγνωρίσει τις Mat.

Στην συνέχεια δηλώνουμε το imgToProcess που είναι τύπου Mat. Και τέλος τρέχουμε την συνάρτηση ImageToMat(την δημιουργούμε για δίκια μας ευκολία) η όποια παίρνει ως είσοδο μια Bitmap εικόνα και επιστρέφει ένα Mat. Με την βοήθεια της Utils.bitmapToMat(img) δέχεται την εικόνα μας και την επιστρέφει ως εικόνα Mat.

Για την μετατροπή της εικόνας Mat σε εικόνα Bitmap γίνετε με τον παρακάτω τρόπο:

```
bmpOut2 = Bitmap.createBitmap(mRgba.cols(), mRgba.rows(),
        Bitmap.Config.ARGB_8888);
Utils.matToBitmap(mRgba, bmpOut2);
```

Το bmpOut2 είναι τύπου Bitmap και του δίνουμε τς διαστάσεις του mRgba που είναι τύπου Mat και στην συνέχεια με το Utils.matToBitmap όπου δέχεται ένα Bitmap και ένα Mat κάνουμε την μετατροπή μας. [6]

8.4.2 Εισαγωγή ενός χρωματισμού της επιλογής μας σε μια εικόνα & αύξηση φωτεινότητας

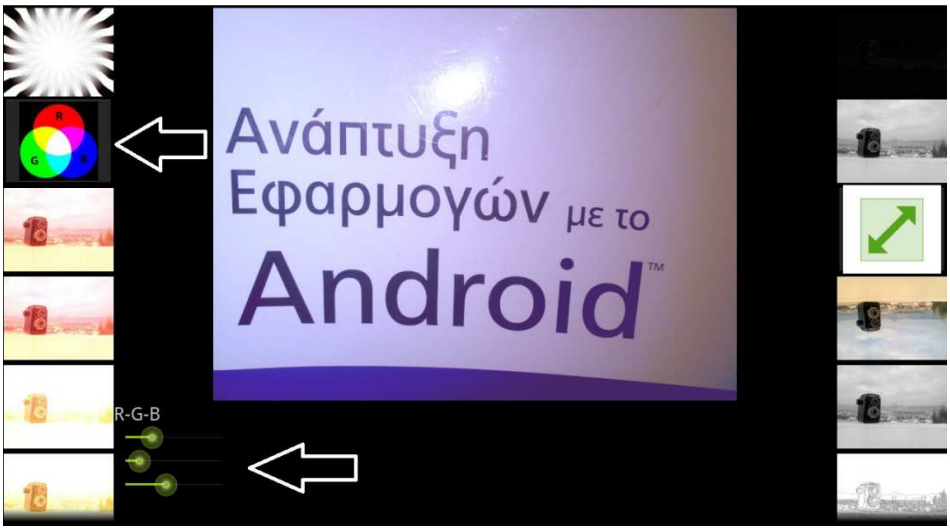
Καταρχάς για να προσθέσουμε ένα χρώμα σε μια εικόνα πρέπει να δημιουργήσουμε μια εικόνα η οποία να έχει το χρώμα που θέλουμε πληκτρολογώντας του τον RGB του χρώματος. Αν δηλαδή θέλαμε να δημιουργήσουμε μια εικόνα με χρώμα κόκκινο στον κώδικα του RGB πληκτρολογούμε (255,0,0). Η εικόνα με το κόκκινο χρώμα πρέπει να είναι τύπου Mat όπως τύπου Mat πρέπει να είναι και η φωτογραφία μας.

```
public Mat RGBconv(Mat src, Mat dst, Mat kernel) {
    m = new Scalar(this.red, this.green, this.blue);
    kernel = new Mat(src, Range.all());
    kernel = new Mat(kernel.size(), src.type(), m);

    Core.add(src, kernel, dst);
    return dst;
}
```

Στον παραπάνω κώδικα μας καλούμε μια συνάρτηση η οποία θα παίρνει ως ορίσματα την φωτογραφία μας (src), τη νέα εικόνα Mat όπου θα μας εμφανίσει το τελικό αποτέλεσμα(dst) και την Mat εικόνα που θα περιέχει το χρώμα που θα ζητήσουμε να προστεθεί στην φωτογραφία μας(kernel).

Δημιουργούμε ένα Scalar για να του δώσουμε το RGB μας(το χρώμα μας) και το αποθηκεύουμε στο m. Μετά δίνουμε στον kernel της διαστάσεις της φωτογραφίας μας. Στην επομένη γραμμή του δίνουμε και το χρώμα που θα έχει η εικόνα μας συμφώνα με το m.

Τέλος καλούμε την συνάρτηση Core.add(src,kernel,dst) η οποία ενώνει 2 εικόνες μαζί. Στην περίπτωση μας ενώνει την φωτογραφεία με έναν χρωματισμό που του ζητήσαμε.



Εικόνα 61 RGB

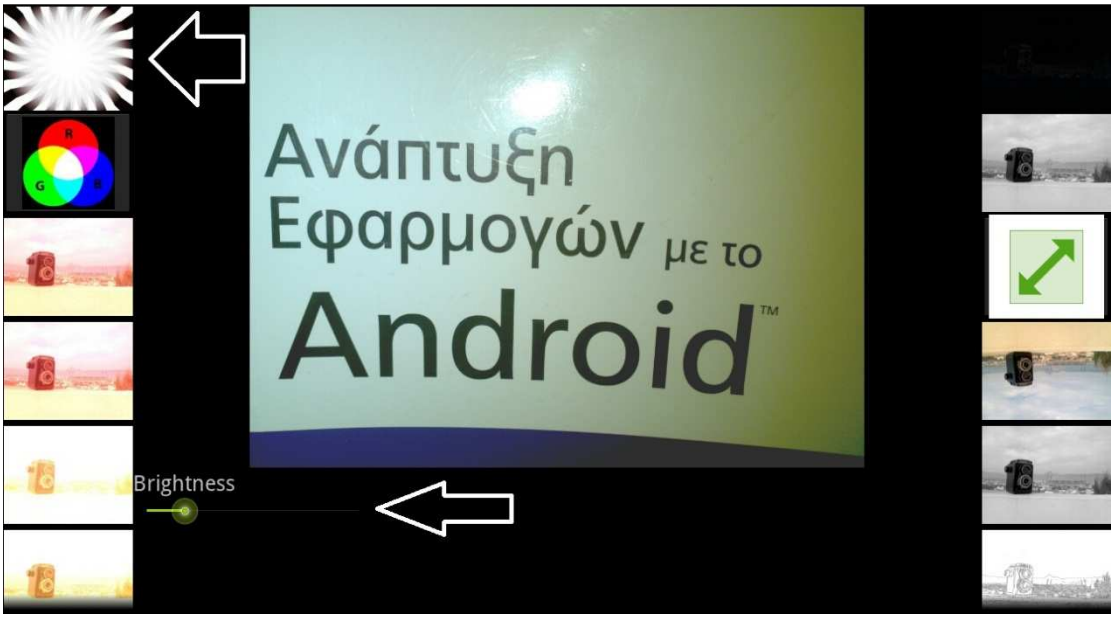
Όπως θα δούμε και στην εικόνα μας παραπάνω με την βοήθεια του seekbar(που το αναφέραμε σε προηγούμενο κεφάλαιο) μπορούμε και πειράζουμε το RGB μας και άμεσως μας το εμφανίζει στο preview image μας στην μέση.

Με έναν παρόμοιο τρόπο μπορούμε να μεγαλώσουμε την φωτεινότητα μια εικόνας. Με την μόνη διαφορά ότι το RGB μας πρέπει να ανεβάζει και της 3 τιμές του τα ίδιο. Δηλαδή:

```
brseek.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {

    public void onStopTrackingTouch(SeekBar seekBar) {
    }

    public void onStartTrackingTouch(SeekBar seekBar) {
    }

    public void onProgressChanged(SeekBar seekBar, int progress,
            boolean fromUser) {
        im.RGB(progress, progress, progress);

        show();

    }
});
```

Εικόνα 62 Java RGB

Βλέπουμε ότι στην περίπτωση μας η τιμή που θα δώσουμε στην seekbar μας θα την δίνουμε και για το Red και για το Green και για το Blue. [6]



Εικόνα 63 Brightness

8.4.3 Εισαγωγή φίλτρων σε εικόνα

Υπάρχουν πολλών ειδών φίλτρα που έχουμε προσθέσει στο πρόγραμμα μας. Τα οποία είναι :

- Flip
- Grayscale
- Canny
- Laplace
- Negative

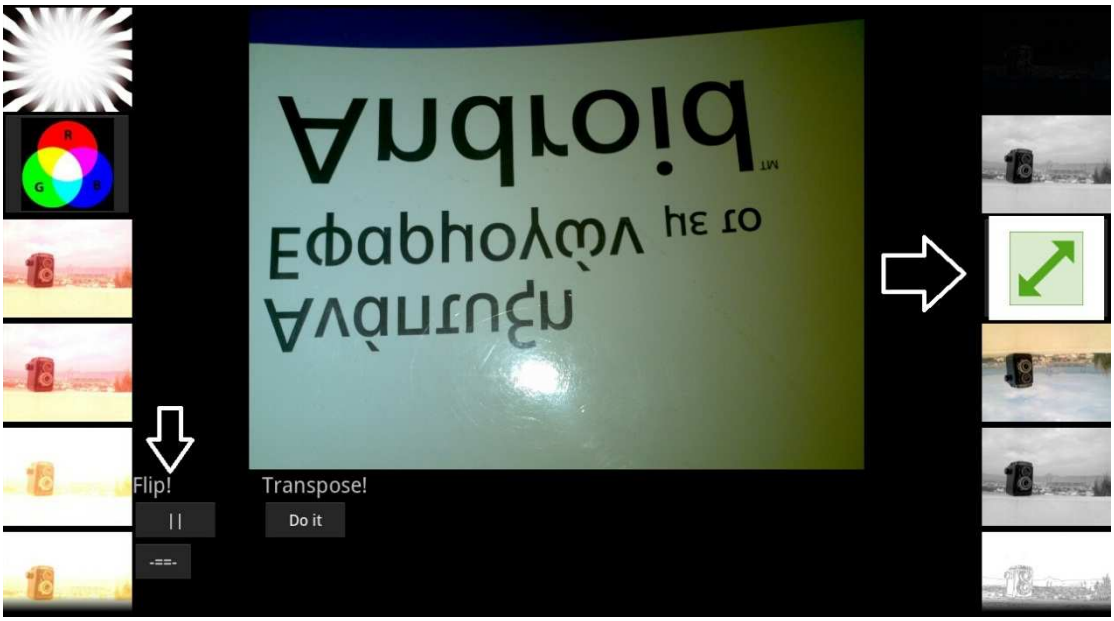
- GaussianBlur
- Onesfilter
- Eyefilter
- morphologyEx
- copyMakeBorder
- sobel
- threshold

[6]

8.4.3.1 Flip

Το φίλτρο αυτό έχει την δυνατότητα να μας αναποδογυρίσει μια εικόνα

```
Core.flip(imgToProcess, mRgba, 0);
```



Εικόνα 64 Flip

Δέχεται ως 2 εικόνες Mat και ένα ακέραιο που του λέει τρόπο αναποδογυρίσματος. Η μια εικόνα Mat είναι η φωτογραφίας μας και η άλλη είναι η νέα εικόνα μας που θα εφαρμοστεί πάνω της το φίλτρο μας.

8.4.3.2 Grayscale

Μετατρέπει μια εικόνα από έγχρωμη σε εικόνα με κλίμακα του γκρι.

```
Imgproc.cvtColor(imgToProcess, mRgba, Imgproc.COLOR_BGR2GRAY);
Imgproc.cvtColor(mRgba, mRgba, Imgproc.COLOR_GRAY2RGBA, 4);
```

Εικόνα 65 GrayScale

Και εδώ δέχεται 2 εικόνες Mat άλλα και την χρωματική αλλαγή που θα του κάνει(BGR2GRAY)

8.4.3.3 Canny

Το φίλτρο canny ανιχνεύει ακμές.

```
Imgproc.Canny(mRgba, mRgba, 590, 600, 3);
```

Εικόνα 66 Java canny

Εικόνα 67 Canny

Παίρνει πάλι 2 εικόνες Mat την φωτογραφία μας και την νέα μας εικόνα και τα υπόλοιπα ορίσματα μας λένε κατά πόσο βαθμό θα ανιχνεύει της ακμές. Όσο μεγαλύτερο νούμερο τόσο λιγότερες ακμές θα ανιχνεύει.

Δίνουμε την δυνατότητα στον χρηστή να επιλέξει το νούμερο για το πλήθος τον ακμών μέσο 2 seekbar.

8.4.3.4 Laplace

Και το φίλτρο Laplace κάνει και αυτό ανίχνευση ακμών με διαφορετικό αποτέλεσμα από ότι στο canny όμως που είπαμε παραπάνω.

```
Imgproc.Laplacian(mRgba, mRgba, mRgba.depth());
```

Εικόνα 68 Java Laplace

Εικόνα 69 Laplace

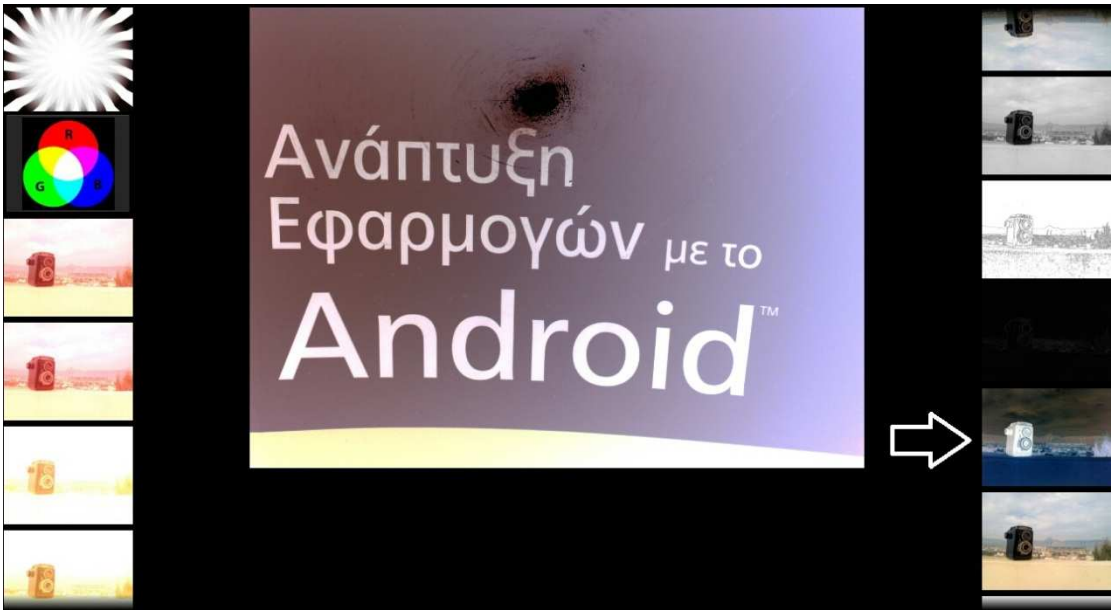
Και εδώ 2 Mat εικόνες άλλα έχουμε και το βάθος χρώματος της φωτογραφία μας.

8.4.3.5 Negative

Το φίλτρο negative μας δίνει την δυνατότητα να αντιστρέψουμε τα χρώματα μιας εικόνας.

```
Core.bitwise_not(imgToProcess, mRgba);
```

Εικόνα 70 Java negative



Εικόνα 71 Negative

Και εδώ 2 εικόνες Mat μια όπου είναι η φωτογραφίας μας και άλλη που θα δοθεί το αποτέλεσμα του φίλτρου μας.

8.4.3.6 GaussianBlur

Σε αυτό το φίλτρο έχουμε την δυνατότητα να προσθέσουμε ένα θόρυβο στην εικόνα μας που να παρουσιάζετε σαν μια θολούρα.

```
Imgproc.GaussianBlur(imgToProcess, mRgba, new Size(1, 1), 40);
```

Εικόνα 72 Java GaussianBlur

Εικόνα 73 GaussianBlur

Και εδώ 2 εικόνες Mat μια για την φωτογραφία μας και μια για το αποτέλεσμα του φίλτρου. Στο size του λέμε το μέγεθος που θέλουμε να έχει το θόρυβο. Όσο πιο μεγάλο τόσο περισσότερος θόρυβος.

Δονούμε και εδώ την δυνατότητα στον χρηστή να επιλέξει το μέγεθος του θορύβου.

8.4.3.7 Onesfilter , Eyefilter

Άλλα 2 φίλτρα είναι το Onesfilter και το Eyefilter που είναι 2 αρκετά κοινά φίλτρα.

```
kernMat = Mat.ones(new Size(4, 4), CvType.CV_64FC1);
Imgproc.filter2D(imgToProcess, mRgba, 64, kernMat);
```

Εικόνα 74 java OneFilter

&

```
kernMat = Mat.eye(new Size(4, 4), CvType.CV_64FC1);
Imgproc.filter2D(imgToProcess, mRgba, 64, kernMat);
```

Εικόνα 75java eyeFilter

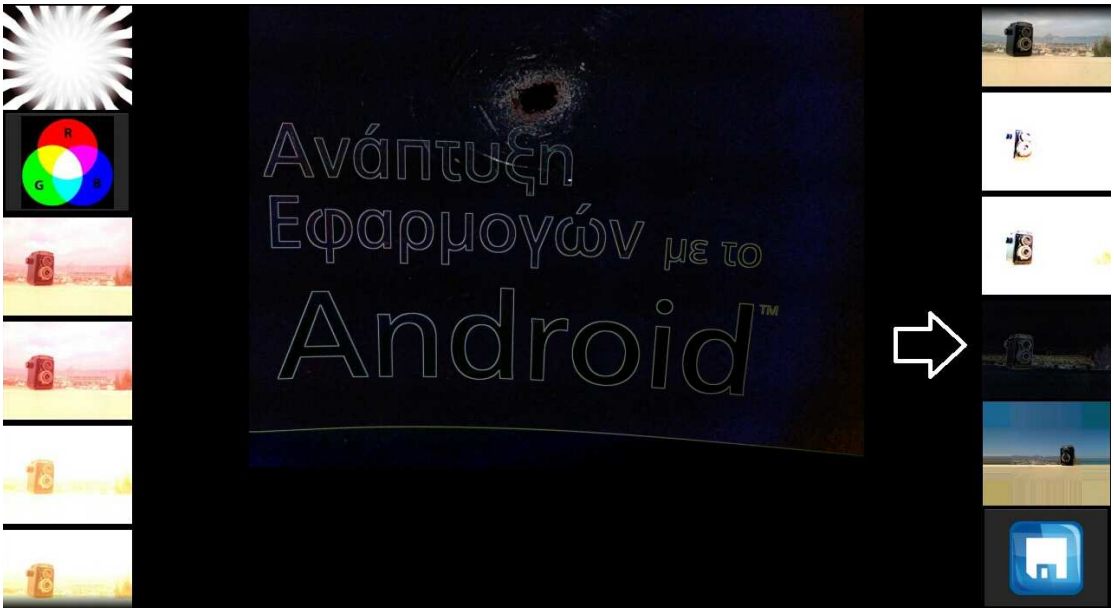
Σε αυτά τα φίλτρα γίνετε μια αλλαγή όσο αφορά τα pixel της εικόνας μας.

8.4.3.8 morphologyEx

Και αυτό το φίλτρο είναι ένα φίλτρο ανίχνευσης ακμών.

```
kernel = Mat.ones(3, 3, CvType.CV_32F);
Imgproc.morphologyEx(imgToProcess, mRgba, Imgproc.MORPH_GRADIENT,
        kernel);
```

Εικόνα 76 Java morphologyEx



Εικόνα 77 morphologyEx

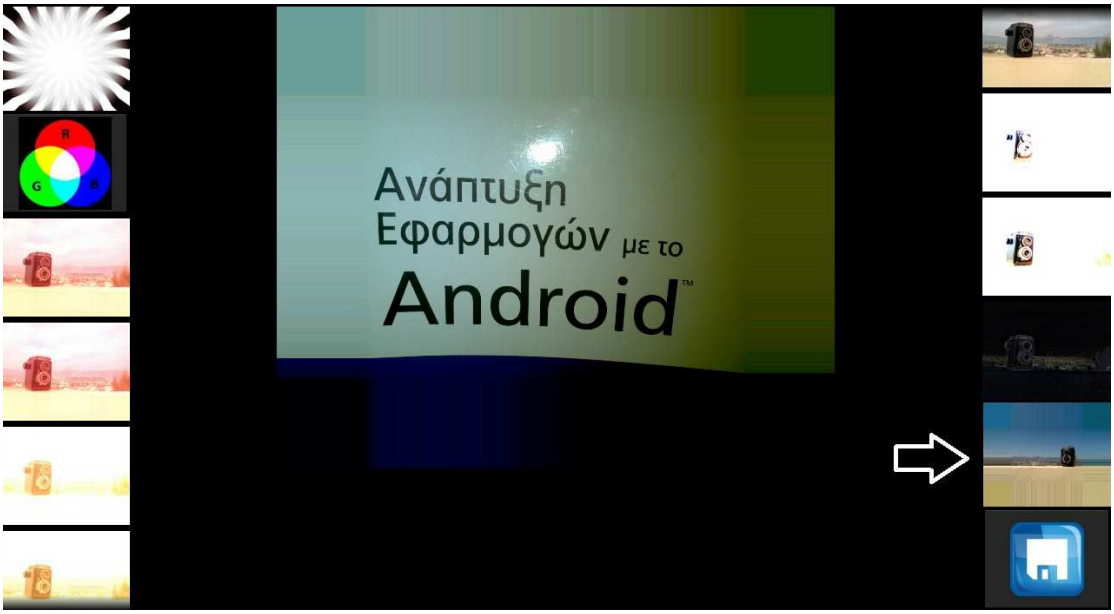
Η διαφορά με τα άλλα φίλτρα ανίχνευσης ακμών είναι ότι εδώ δίνεται με την βοήθεια ενός αλλού πίνακα Mat 3x3 διαστάσεων με άλλες της τιμές του 1.

8.4.3.9 copyMakeBorder

Σε αυτό το φίλτρο προσθέτουμε ένα παραμορφωμένο περίγραμμα στην εικόνα μας.

```
Imgproc.copyMakeBorder(imgToProcess, mRgba, 300, 300, 300, 300,
        Imgproc.BORDER_REPLICATE);
```

Εικόνα 78 Java copyMakeBorder



Εικόνα 79 copyMakeBorder

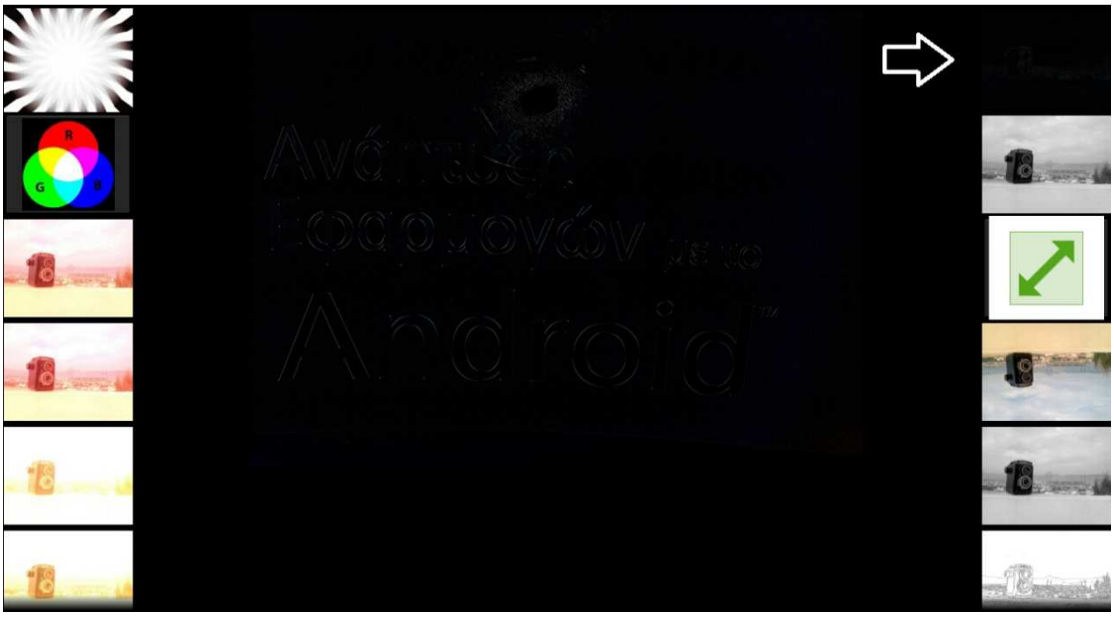
Εκτός από της γνώστες 2 εικόνες Mat που δέχεται του λέμε και πόσο θα είναι το μέγεθος του περιγράμματος και για της 4 πλευρές και τον τύπο του περιγράμματος.

8.4.3.10 sobel

Άλλο ένα φίλτρο είναι το sobel.

```
Imgproc.Sobel(imgToProcess, mRgba, imgToProcess.depth(), 1, 1);
```

Εικόνα 80 java sobel



Εικόνα 81sobel

Και εδώ έχουμε 2 εικόνες Mat για την φωτογραφία μας και για το αποτέλεσμα του φίλτρου μας. Επίσης πρέπει να του αναφέρουμε και το βάθος χρώματος της εικόνας μας.

8.4.3.11 threshold

Με το φίλτρο threshold έχουμε την δυνατότητα να κόψουμε τα ελάχιστα μιας εικόνας. Δηλαδή αν έχουμε μια ασπρόμαυρη εικόνα μπορούμε να μην συμπεριέλαβει κόπιες αποχρώσεις του γκρι. Αυτό γίνετε με την βοήθεια ενός seekbar. Η seekbar παίρνει τιμές από 0-255(όσες και οι αποχρώσεις του γκρι. Αν το seekbar πάρει την τιμή 50 τότε όσες αποχρώσεις του γκρι έχουν τιμή κάτω από 50 γίνονται μαύρα.

```
Imgproc.cvtColor(imgToProcess, mRgba, Imgproc.COLOR_BGR2GRAY);
Imgproc.cvtColor(mRgba, mRgba, Imgproc.COLOR_GRAY2RGBA, 4);
Imgproc.threshold(mRgba, mRgba, 0, 0, Imgproc.THRESH_TOZERO);
```

Εικόνα 82 java threshold

Εικόνα 83 threshold

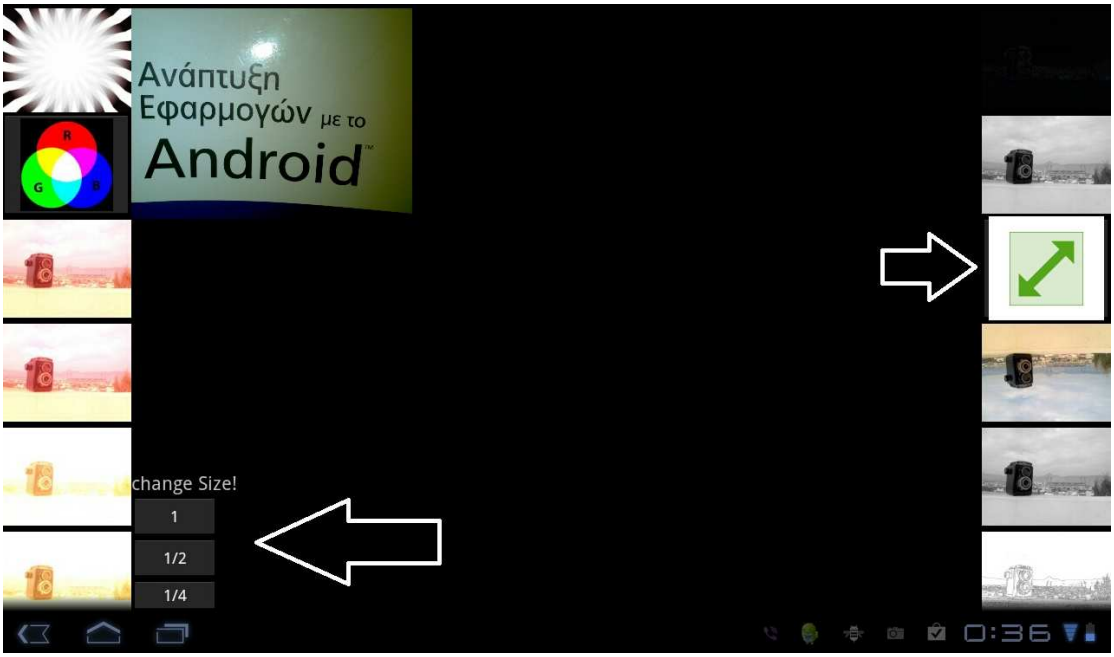
Πρώτα πρέπει να κάνουμε την μετατροπή σε grayscale την εικόνα μας και μετά να χρησιμοποιήσουμε το φίλτρο μας.

8.4.4 Αλλαγή διαστάσεων

Η βιβλιοθήκη OpenCV μας δίνει την δυνατότητα να αλλάξουμε το μέγεθος μιας εικόνας.

```
Imgproc.resize(imgToProcess, mRgba, new Size(
        (1 * imgToProcess.cols()),
        (1 * imgToProcess.rows())));
```

Εικόνα 84 java resize



Εικόνα 85 resize

Και το resize παίρνει ως όρισμα 2 εικόνες Mat. Μια για την φωτογραφία μας και μια που θα μας δίνει το νέο μέγεθος μας. Επίσης έχει το Size όπου αν το πολλαπλασιάσουμε επί 1 τότε θα πάρουμε το κανονικό μέγεθος ενώ αν το πολλαπλασιάσουμε επί 0,5 θα πάρει το μισό μέγεθος της εικόνας μας.

8.5 Αποθήκευση εικόνας

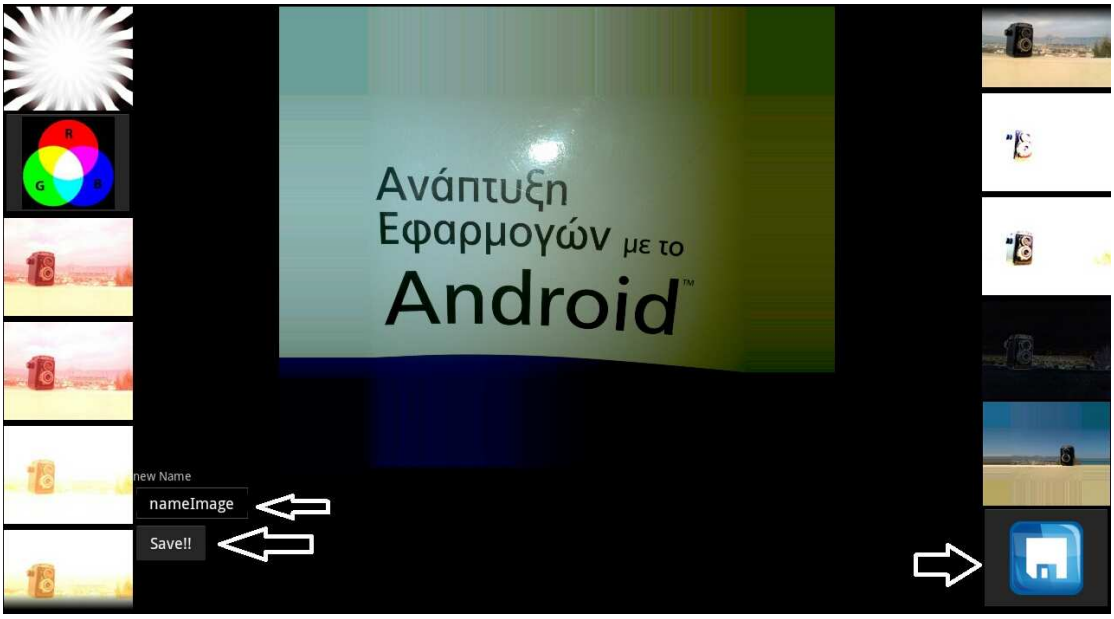
Έχουμε την δυνατότητα να αποθηκεύσουμε την εικόνα μας με ένα όνομα που εμείς επιθυμούμε.

Αυτό γίνετε με την βοήθεια ενός Button και ενός EditText.

Στο EditText γραφούμε το όνομα που θέλουμε να αποθηκεύσουμε την εικόνα μας και με το που πατήσουμε το κουμπί μας τότε παίρνει το κείμενο που έχουμε γράψει στο EditText μας και το μετατρέπει σε String. Στην συνέχεια αποθηκεύει την εικόνα μας με τον τύπο που θα του ζητήσουμε εμείς (πχ jpg) στον φάκελο που θα του πούμε και με το όνομα που έχουμε πάρει από το EditText(το String μας).

```
save.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        String filename= saveName.getText().toString();

        try {

            String path = Environment.getExternalStorageDirectory()
                    .getAbsolutePath();
            OutputStream fOut = null;
            File file = new File(path, filename + ".jpg");
            fOut = new FileOutputStream(file);
            bmpOut2.compress(Bitmap.CompressFormat.JPEG, 100, fOut);
            fOut.flush();
            fOut.close();
            MediaStore.Images.Media.insertImage(
                    getContentResolver(), file.getAbsolutePath(),
                    file.getName(), file.getName());
        } catch (Exception e) {
        }
    }
});
```

Εικόνα 86 java save



Εικόνα 87 save

Συμπεράσματα & μελλοντικές επεκτάσεις

Από τα παραπάνω κεφάλαια καταλήγουμε ότι το λειτουργικό Android παρέχει τα έξης:

- Είναι ένα ολοκληρωμένο λειτουργικό σύστημα για κινητές συσκευές.
- Αποτελεί μια ολοκληρωμένη πλατφόρμα ανάπτυξης εφαρμογών και προσφέρει πληθώρα εργαλείων και μεθόδων
- Παρέχει πολλά και τεκμηριωμένα κείμενα ανάπτυξης εφαρμογών και του τρόπου λειτουργιάς του συστήματος
- Μπορεί να καλύψει τις ανάγκες τόσο των μέσων όσο και των εξειδικευμένων χρηστών.
- Μας δίνει την δυνατότητα να εισάγουμε βιβλιοθήκες ακόμα και αν είναι γραμμένες με C++ με την βοήθεια του Android NDK.
- Οι κατασκευαστές των κινητών συσκευών έχουν την δυνατότητα να προσαρμόσουν ανάλογα με τις ανάγκες τους το λειτουργικό σύστημα Android χωρίς κόστος.

Όσον αφορά πάλι για την βιβλιοθήκη OpenCV καταλήγουμε ότι:

- Είναι μια τεραστία βιβλιοθήκη με μεγάλο πλήθος συναρτήσεων
- Έχει απεριόριστες δυνατότητες όσο αφορά την επεξεργασία εικόνας video και ανίχνευσης
- Μπορεί εύκολα να ενσωματωθεί στην εφαρμογή μας.
- Μας παρέχει εύκολη υλοποίηση

Χάρη στα παραπάνω καταφέραμε να επιτύχουμε τους στόχους μας και να υλοποιήσουμε την εφαρμογή μας έτσι όπως την θέλαμε. Μια εφαρμογή η οποία έχει μια πληθώρα επίλογων και δυνατοτήτων χωρίς όμως να είναι απαραίτητη μεγάλη επεξεργαστική ισχύ και να καταλαμβάνει μεγάλο χώρο στον δίσκο της συσκευής. Είναι πολύ εύκολη στην χρήση με την δυνατότητα όπου ο χρήστης να μπορεί να βλέπει με preview τα αποτελέσματα κάθε κίνησης που σκέφτεται να κάνει. Με το να υπάρχουν 2 scroll menu ένα αριστερά και ένα δεξιά μπορεί να γίνει η χρήση του προγράμματος μόνο με τους 2 αντίχειρες μας.

Η εφαρμογή μας θα μπορούσε μελλοντικά να επεκταθεί και να προστεθούν και άλλες δυνατότητες όπως είναι :

- Εισαγωγή νέων φίλτρων
- Αφαίρεση του κόκκινου των ματιών
- Κουμπί για εισαγωγή εικόνας από αρχείο
- Δυνατότητα αποστολής εικόνας είτε με mail είτε και ανέβασμα σε Facebook, Gmail κλπ
- Εισαγωγή έτοιμων περιγραμμάτων
- Αλλαγή χρωμάτων σε συγκεκριμένα τμήματα της εικόνας
- Ζουμ πάνω στην εικόνα μας
- Αποκοπή με επιλογή

Βιβλιογραφία

[1] Ανάπτυξη εφαρμογών με το Android εκδόσεις: Μ.Γκιουρδας

[2] Android: Apress - Pro Android Media Developing Graphics, Music, Video, and Rich Media Apps for Smartphones and Tablets

[3] Android: Android NDK Beginner's Guide

[4] Android: Professional Android 2 Application Development (2010) (Malestrom)

[5] Android: http://www.lynda.com/Android-2-tutorials/Android-App-Development-with-Java-Essential-Training/79825-2.html

[6] OpenCV: http://opencv.willowgarage.com/wiki/Android

[7] OpenCV : http://opencv.willowgarage.com/documentation/cpp/

[8] OpenCV: http://opencv.willowgarage.com/wiki/

[9] Ιστορικα: http://www.sansimera.gr/articles/241

[10] SmartPhone: http://shoppingtips.gr/index.php?option=com_content&view=article&id=83:-smart-phone&catid=10:2010-05-13-16-11-09&Itemid=13

Παράρτημα

Java Κώδικας

OpenCVAndroidActivity

```
package gr.epp2394.OpenCVAndroid;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;

public class OpenCVAndroidActivity extends Activity {
	/** Called when the activity is first created. */
	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.main);
		Intent intent = new Intent(OpenCVAndroidActivity.this, Camera1.class);
		startActivity(intent);

	}
}
```

Camera1

```
package gr.epp2394.OpenCVAndroid;


import java.io.File;
//import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.OutputStream;

import org.opencv.android.Utils;
import org.opencv.core.*;
import org.opencv.imgproc.*;

import android.app.Activity;
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.net.Uri;
import android.os.Bundle;
import android.os.Environment;
import android.provider.MediaStore;
import android.util.Log;
import android.view.Display;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.ImageView;
import android.widget.LinearLayout;
```

```
import android.widget.SeekBar;

public class Camera1 extends Activity {

        final static int CAMERA_RESULT = 0;
        ImageView imv;
        String imageFilePath;
        String imageFilePath1;

        // private Mat mYuv;
        public Mat imgToProcess;
        public Mat mRgba;// neo
        public Mat kernMat;
        private Mat kernel;
        Bitmap bmpOut;
        Scalar m;
        Bitmap bmp;
        String imagename;
        Bitmap bmpOut2;
        int r, g, b, c1, c2, cwb = 0, fl, bl;
        float si;
        public String filename;
        public LinearLayout flipLL, rgbLL, brightnessLL, cannyLL, blurLL, saveLL,
                        resizeLL, thresholdLL;
        EditText saveName;
        Filters im;

        // private double[][] mike = {{0,-2,0}, {1,0,1},{0,-2,0}};
```

```
	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.main2);

		imageFilePath = Environment.getExternalStorageDirectory()
				.getAbsolutePath() + "/myfavoritepicture.jpg";

		File imageFile = new File(imageFilePath);
		Uri imageFileUri = Uri.fromFile(imageFile);

		Intent i = new
Intent(android.provider.MediaStore.ACTION_IMAGE_CAPTURE);
		i.putExtra(android.provider.MediaStore.EXTRA_OUTPUT, imageFileUri);
		startActivityForResult(i, CAMERA_RESULT);
	}

	protected void onActivityResult(int requestCode, int resultCode,
			Intent intent) {
		super.onActivityResult(requestCode, resultCode, intent);

		// dhlwseis
		imv = (ImageView) findViewById(R.id.ReturnedImageView);
		flipLL = (LinearLayout) findViewById(R.id.flipll);
		cannyLL = (LinearLayout) findViewById(R.id.cannyll);
		rgbLL = (LinearLayout) findViewById(R.id.rgbll);
		brightnessLL = (LinearLayout) findViewById(R.id.brightnessll);
```

```
blurLL = (LinearLayout) findViewById(R.id.gaussianll);
saveLL = (LinearLayout) findViewById(R.id.savell);
resizeLL = (LinearLayout) findViewById(R.id.resizell);

thresholdLL = (LinearLayout) findViewById(R.id.thresholdll);
im = new Filters();

ImageButton b1 = (ImageButton) findViewById(R.id.b1);
ImageButton b2 = (ImageButton) findViewById(R.id.b2);
ImageButton b3 = (ImageButton) findViewById(R.id.b3);
ImageButton b4 = (ImageButton) findViewById(R.id.b4);
ImageButton b5 = (ImageButton) findViewById(R.id.b5);
ImageButton b6 = (ImageButton) findViewById(R.id.b6);
ImageButton b7 = (ImageButton) findViewById(R.id.b7);
ImageButton b8 = (ImageButton) findViewById(R.id.b8);
ImageButton b9 = (ImageButton) findViewById(R.id.b9);
ImageButton b10 = (ImageButton) findViewById(R.id.b10);
ImageButton b11 = (ImageButton) findViewById(R.id.b11);
ImageButton b12 = (ImageButton) findViewById(R.id.b12);
ImageButton b13 = (ImageButton) findViewById(R.id.b13);
ImageButton b14 = (ImageButton) findViewById(R.id.b14);
ImageButton b15 = (ImageButton) findViewById(R.id.b15);

ImageButton br1 = (ImageButton) findViewById(R.id.br1);
ImageButton br2 = (ImageButton) findViewById(R.id.br2);
ImageButton br3 = (ImageButton) findViewById(R.id.br3);
ImageButton br4 = (ImageButton) findViewById(R.id.br4);
ImageButton br5 = (ImageButton) findViewById(R.id.br5);
```

```
ImageButton br6 = (ImageButton) findViewById(R.id.br6);
ImageButton br7 = (ImageButton) findViewById(R.id.br7);
ImageButton br8 = (ImageButton) findViewById(R.id.br8);
ImageButton br9 = (ImageButton) findViewById(R.id.br9);
ImageButton br10 = (ImageButton) findViewById(R.id.br10);
ImageButton brsize = (ImageButton) findViewById(R.id.brsize);

ImageButton br11 = (ImageButton) findViewById(R.id.br11);
ImageButton br12 = (ImageButton) findViewById(R.id.br12);
Button size1 = (Button) findViewById(R.id.size1);
Button size12 = (Button) findViewById(R.id.size12);
Button size14 = (Button) findViewById(R.id.size14);

ImageButton saveico = (ImageButton) findViewById(R.id.saveico);
ImageButton brightnessico = (ImageButton)
findViewById(R.id.brighnessico);
ImageButton rgbico = (ImageButton) findViewById(R.id.rgbico);

SeekBar brseek = (SeekBar) findViewById(R.id.brseek);
SeekBar reseek = (SeekBar) findViewById(R.id.red);
SeekBar grseek = (SeekBar) findViewById(R.id.green);
SeekBar blseek = (SeekBar) findViewById(R.id.blue);
SeekBar ca1seek = (SeekBar) findViewById(R.id.ca1seek);
SeekBar ca2seek = (SeekBar) findViewById(R.id.ca2seek);
SeekBar blurseek = (SeekBar) findViewById(R.id.blurseek);
Button cabutton = (Button) findViewById(R.id.cabutton);
Button fl2button = (Button) findViewById(R.id.fl2button);
Button fl3button = (Button) findViewById(R.id.fl3button);
Button transposebutton = (Button) findViewById(R.id.transpose);
```

```
SeekBar thresholdseek = (SeekBar) findViewById(R.id.thresholdseek);

saveName = (EditText) findViewById(R.id.namepic);

Button save = (Button) findViewById(R.id.save);
if (resultCode == RESULT_OK) {



// eisagogh eikonas kai metatropei se bitmap
Display currentDisplay = getWindowManager().getDefaultDisplay();
int dw = currentDisplay.getWidth();
int dh = currentDisplay.getHeight();

BitmapFactory.Options bmpFactoryOptions = new
BitmapFactory.Options();
bmpFactoryOptions.inJustDecodeBounds = true;
bmp = BitmapFactory.decodeFile(imageFilePath,
bmpFactoryOptions);
int heightRatio = (int) Math.ceil(bmpFactoryOptions.outHeight
/ (float) dh);
int wightRatio = (int) Math.ceil(bmpFactoryOptions.outWidth
/ (float) dw);
Log.v("HEIGHTRATIO", "" + heightRatio);
Log.v("WIDTHRATIO", "" + wightRatio);
if (heightRatio > 1 && wightRatio > 1) {
if (heightRatio > wightRatio) {
bmpFactoryOptions.inSampleSize = heightRatio;
```

```
                } else {
                        bmpFactoryOptions.inSampleSize = wightRatio;
                }
        }
        bmpFactoryOptions.inJustDecodeBounds = false;
        bmp = BitmapFactory.decodeFile(imageFilePath,
bmpFactoryOptions);
        VisiBility(5);

        imgToProcess = im.imageToMat(bmp);

        kernMat = new Mat();

        mRgba = new Mat(imgToProcess, Range.all());
        // mRgba=new Mat(mRgba.rows(), mRgba.cols(),
CvType.CV_32FC1);

        bmpOut2 = Bitmap.createBitmap(mRgba.cols(), mRgba.rows(),
                        Bitmap.Config.ARGB_8888);
        Utils.matToBitmap(mRgba, bmpOut2);
        imv.setImageBitmap(bmpOut2);
        // koumpia onclick
        b1.setOnClickListener(new View.OnClickListener() {
                public void onClick(View v) {

                        imv.setImageBitmap(filter(1));

                }
        });
```

```
b2.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
                imv.setImageBitmap(filter(2));
        }
});
b3.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
                imv.setImageBitmap(filter(3));
        }
});
b4.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
                imv.setImageBitmap(filter(4));
        }
});
b5.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
                imv.setImageBitmap(filter(5));
        }
});
b6.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
                imv.setImageBitmap(filter(6));
        }
});
b7.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
                imv.setImageBitmap(filter(7));
```

```
        }
    });
    b8.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            imv.setImageBitmap(filter(8));
        }
    });
    b9.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            imv.setImageBitmap(filter(9));
        }
    });
    b10.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            imv.setImageBitmap(filter(10));
        }
    });
    b11.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            imv.setImageBitmap(filter(11));
        }
    });
    b12.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            imv.setImageBitmap(filter(12));
        }
    });
    b13.setOnClickListener(new View.OnClickListener() {
```

```
            public void onClick(View v) {
                imv.setImageBitmap(filter(13));
            }
        });
        b14.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                imv.setImageBitmap(filter(14));
            }
        });
        b15.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                imv.setImageBitmap(filter(15));
            }
        });

        br1.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {

                imv.setImageBitmap(filter(30));

            }
        });
        br2.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                imv.setImageBitmap(filter(31));
            }
        });
        br3.setOnClickListener(new View.OnClickListener() {
```

```
            public void onClick(View v) {
                imv.setImageBitmap(filter(32));
            }
        });
        br4.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                imv.setImageBitmap(filter(33));
            }
        });
        br5.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                imv.setImageBitmap(filter(34));
            }
        });
        br6.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                imv.setImageBitmap(filter(35));
            }
        });
        br7.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                imv.setImageBitmap(filter(36));
            }
        });
        br8.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                imv.setImageBitmap(filter(37));
            }
```

```
});
br9.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
                imv.setImageBitmap(filter(38));
        }
});
br10.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
                imv.setImageBitmap(filter(39));
        }
});
br11.setOnClickListener(new View.OnClickListener() {

        public void onClick(View v) {
                imv.setImageBitmap(filter(40));

        }
});
br12.setOnClickListener(new View.OnClickListener() {

        public void onClick(View v) {
                imv.setImageBitmap(filter(41));

        }
});
brsize.setOnClickListener(new View.OnClickListener() {

        public void onClick(View v) {
```

```
                    VisiBility(8);

                }
            });
            saveico.setOnClickListener(new View.OnClickListener() {

                public void onClick(View v) {
                    VisiBility(7);

                }
            });
            size1.setOnClickListener(new View.OnClickListener() {

                public void onClick(View v) {
                    Imgproc.resize(imgToProcess, mRgba, new Size(
                            (1 * imgToProcess.cols()),
                            (1 * imgToProcess.rows())));

                    bmpOut2 = Bitmap.createBitmap(mRgba.cols(),
mRgba.rows(),
                            Bitmap.Config.ARGB_8888);
                    Utils.matToBitmap(mRgba, bmpOut2);
                    imv.setImageBitmap(bmpOut2);

                }
            });
            size12.setOnClickListener(new View.OnClickListener() {

                public void onClick(View v) {
```

```
                        Imgproc.resize(imgToProcess, mRgba,
                                new Size((0.5 *
imgToProcess.cols()),
                                        (0.5 *
imgToProcess.rows())));

                        bmpOut2 = Bitmap.createBitmap(mRgba.cols(),
mRgba.rows(),
                                Bitmap.Config.ARGB_8888);
                        Utils.matToBitmap(mRgba, bmpOut2);
                        imv.setImageBitmap(bmpOut2);

                    }
                });
                size14.setOnClickListener(new View.OnClickListener() {

                    public void onClick(View v) {
                        Imgproc.resize(imgToProcess, mRgba,
                                new Size((0.25 *
imgToProcess.cols()),
                                        (0.25 *
imgToProcess.rows())));

                        bmpOut2 = Bitmap.createBitmap(mRgba.cols(),
mRgba.rows(),
                                Bitmap.Config.ARGB_8888);
                        Utils.matToBitmap(mRgba, bmpOut2);
                        imv.setImageBitmap(bmpOut2);

                    }
                });
```

```
save.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
                String filename= saveName.getText().toString();


                try {

                        String path =
Environment.getExternalStorageDirectory()
                                        .getAbsolutePath();
                        OutputStream fOut = null;
                        File file = new File(path, filename + ".jpg");
                        fOut = new FileOutputStream(file);

    bmpOut2.compress(Bitmap.CompressFormat.JPEG, 100, fOut);
                        fOut.flush();
                        fOut.close();
                        MediaStore.Images.Media.insertImage(
                                        getContentResolver(),
file.getAbsolutePath(),
                                        file.getName(),
file.getName());
                } catch (Exception e) {
                }
        }
});
brightnessico.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
                VisiBility(4);
```

```
                }
            });
            rgbico.setOnClickListener(new View.OnClickListener() {
                public void onClick(View v) {
                    VisiBility(2);
                }
            });
            cabutton.setOnClickListener(new View.OnClickListener() {

                public void onClick(View v) {
                    Imgproc.cvtColor(imgToProcess, mRgba,
                            Imgproc.COLOR_BGRA2GRAY,
4);
                    Imgproc.Canny(mRgba, mRgba, im.getC1(),
im.getC2(), 5);
                    Imgproc.cvtColor(mRgba, mRgba,
Imgproc.COLOR_GRAY2RGBA, 4);
                    if (cwb == 0) {

                        Core.bitwise_not(mRgba, mRgba);

                        cwb = 1;

                    } else {

                        cwb = 0;
                    }

                    bmpOut = Bitmap.createBitmap(mRgba.cols(),
mRgba.rows(),
```

```
                                        Bitmap.Config.ARGB_8888);
                                Utils.matToBitmap(mRgba, bmpOut);
                                imv.setImageBitmap(bmpOut);
                        }
                });

                fl2button.setOnClickListener(new View.OnClickListener() {

                        public void onClick(View v) {
                                mRgba = flip(0);
                                bmpOut = Bitmap.createBitmap(mRgba.cols(),
mRgba.rows(),
                                        Bitmap.Config.ARGB_8888);
                                Utils.matToBitmap(mRgba, bmpOut);
                                imv.setImageBitmap(bmpOut);

                        }
                });
                fl3button.setOnClickListener(new View.OnClickListener() {

                        public void onClick(View v) {
                                mRgba = flip(1);
                                bmpOut = Bitmap.createBitmap(mRgba.cols(),
mRgba.rows(),
                                        Bitmap.Config.ARGB_8888);
                                Utils.matToBitmap(mRgba, bmpOut);
                                imv.setImageBitmap(bmpOut);
                        }
                });
```

```
                transposebutton.setOnClickListener(new View.OnClickListener() {

                        public void onClick(View v) {
                                Core.transpose(imgToProcess, mRgba);
                                bmpOut = Bitmap.createBitmap(mRgba.cols(),
mRgba.rows(),
                                                Bitmap.Config.ARGB_8888);
                                Utils.matToBitmap(mRgba, bmpOut);
                                imv.setImageBitmap(bmpOut);
                        }
                });

                // seekbars
                // Brightness
                brseek.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener() {

                        public void onStopTrackingTouch(SeekBar seekBar) {
                        }

                        public void onStartTrackingTouch(SeekBar seekBar) {
                        }

                        public void onProgressChanged(SeekBar seekBar, int
progress,
                                        boolean fromUser) {
                                im.RGB(progress, progress, progress);

                                show();
```

```
                }
            });
            // RGB Red
            reseek.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener() {

                public void onStopTrackingTouch(SeekBar seekBar) {
                }

                public void onStartTrackingTouch(SeekBar seekBar) {
                }

                public void onProgressChanged(SeekBar seekBar, int
progress,
                        boolean fromUser) {
                    im.RGB(progress, im.getGreen(), im.getBlue());

                    show();

                }
            });
            // RGB Green
            grseek.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener() {

                public void onStopTrackingTouch(SeekBar seekBar) {
                    // TODO Auto-generated method stub

                }
```

```
                public void onStartTrackingTouch(SeekBar seekBar) {
                        // TODO Auto-generated method stub

                }

                public void onProgressChanged(SeekBar seekBar, int
progress,
                                boolean fromUser) {
                        im.RGB(im.getRed(), progress, im.getBlue());

                        show();

                }
        });
        // RGB Blue
        blseek.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener() {

                public void onStopTrackingTouch(SeekBar seekBar) {
                        // TODO Auto-generated method stub

                }

                public void onStartTrackingTouch(SeekBar seekBar) {
                        // TODO Auto-generated method stub

                }

                public void onProgressChanged(SeekBar seekBar, int
progress,
```

```
                                        boolean fromUser) {
                                    im.RGB(im.getRed(), im.getGreen(), progress);

                                    show();

                                }
                            });
                            // canny
                            ca1seek.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener() {

                                public void onStopTrackingTouch(SeekBar seekBar) {
                                    // TODO Auto-generated method stub

                                }

                                public void onStartTrackingTouch(SeekBar seekBar) {
                                    // TODO Auto-generated method stub

                                }

                                public void onProgressChanged(SeekBar seekBar, int
progress,
                                        boolean fromUser) {
                                    c2 = im.getC2();
                                    im.canny(progress, c2);

                                    bmpOut2 = im.showcanny(imgToProcess, mRgba,
cwb);
                                    imv.setImageBitmap(bmpOut2);
```

```
                }

            });

            ca2seek.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener() {


                public void onStopTrackingTouch(SeekBar seekBar) {
                    // TODO Auto-generated method stub


                }


                public void onStartTrackingTouch(SeekBar seekBar) {
                    // TODO Auto-generated method stub


                }


                public void onProgressChanged(SeekBar seekBar, int
progress,
                        boolean fromUser) {
                    c1 = im.getC1();
                    im.canny(c1, progress);
                    bmpOut2 = im.showcanny(imgToProcess, mRgba,
cwb);
                    imv.setImageBitmap(bmpOut2);


                }

            });

            // blur

            blurseek.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener() {
```

```
                    public void onStopTrackingTouch(SeekBar seekBar) {
                        // TODO Auto-generated method stub

                    }

                    public void onStartTrackingTouch(SeekBar seekBar) {
                        // TODO Auto-generated method stub

                    }

                    public void onProgressChanged(SeekBar seekBar, int
progress,
                            boolean fromUser) {
                        if (progress % 2 == 0) {
                            progress = progress + 1;
                        }
                        Imgproc.GaussianBlur(imgToProcess, mRgba, new
Size(
                                progress, progress), 40);
                        bmpOut = Bitmap.createBitmap(mRgba.cols(),
mRgba.rows(),
                                Bitmap.Config.ARGB_8888);
                        Utils.matToBitmap(mRgba, bmpOut);
                        imv.setImageBitmap(bmpOut);

                    }
                });
                thresholdseek.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener() {
```

```
                public void onStopTrackingTouch(SeekBar seekBar) {
                        // TODO Auto-generated method stub

                }

                public void onStartTrackingTouch(SeekBar seekBar) {
                        // TODO Auto-generated method stub

                }

                public void onProgressChanged(SeekBar seekBar, int
progress,
                                boolean fromUser) {

                        Imgproc.cvtColor(imgToProcess, mRgba,
Imgproc.COLOR_BGR2GRAY);
                        Imgproc.cvtColor(mRgba, mRgba,
Imgproc.COLOR_GRAY2RGBA, 4);
                        Imgproc.threshold(mRgba, mRgba, progress, 0,
Imgproc.THRESH_TOZERO);
                        bmpOut = Bitmap.createBitmap(mRgba.cols(),
mRgba.rows(),
                                        Bitmap.Config.ARGB_8888);
                        Utils.matToBitmap(mRgba, bmpOut);
                        imv.setImageBitmap(bmpOut);

                }
        });
```

```
            }
        }

        protected Bitmap filter(int i) {
            imgToProcess = im.imageToMat(bmp);
            switch (i) {
            case 1:

                filename="Auburn";
                VisiBility(2);
                im.RGB(147, 39, 36);

                mRgba = im.RGBconv(imgToProcess, mRgba, kernMat);

                break;
            case 2:

                filename="Red-Brown";
                VisiBility(2);
                im.RGB(165, 42, 42);
                mRgba = im.RGBconv(imgToProcess, mRgba, kernMat);
                break;
            case 3:
                filename="Sandy Brown";
                VisiBility(2);
                im.RGB(244, 164, 96);
                mRgba = im.RGBconv(imgToProcess, mRgba, kernMat);
                break;
```

```
case 4:
        filename="Peru";
        VisiBility(2);
        im.RGB(205, 133, 63);
        mRgba = im.RGBconv(imgToProcess, mRgba, kernMat);
        break;
case 5:
        filename="Rosy Brown";
        VisiBility(2);
        im.RGB(188, 143, 143);
        mRgba = im.RGBconv(imgToProcess, mRgba, kernMat);
        break;
case 6:
        filename="Beaver";
        VisiBility(2);
        im.RGB(159, 139, 112);
        mRgba = im.RGBconv(imgToProcess, mRgba, kernMat);
        break;
case 7:
        filename="Chestnut";
        VisiBility(2);
        im.RGB(149, 69, 53);
        mRgba = im.RGBconv(imgToProcess, mRgba, kernMat);
        break;
case 8:
        filename="Russet";
        VisiBility(2);
        im.RGB(128, 70, 27);
```

```
            mRgba = im.RGBconv(imgToProcess, mRgba, kernMat);
            break;
        case 9:

            filename="Bistre";
            VisiBility(2);
            im.RGB(61, 43, 31);
            mRgba = im.RGBconv(imgToProcess, mRgba, kernMat);
            break;
        case 10:
            filename="Bronze";
            VisiBility(2);
            im.RGB(205, 127, 50);
            mRgba = im.RGBconv(imgToProcess, mRgba, kernMat);
            break;
        case 11:
            filename="Buff";
            VisiBility(2);
            im.RGB(240, 220, 130);
            mRgba = im.RGBconv(imgToProcess, mRgba, kernMat);
            break;
        case 12:
            filename="Sienna";
            VisiBility(2);
            im.RGB(136, 45, 23);
            mRgba = im.RGBconv(imgToProcess, mRgba, kernMat);
            break;
```

```
case 13:
        filename="Dark Sienna";
        VisiBility(2);
        im.RGB(60, 20, 20);
        mRgba = im.RGBconv(imgToProcess, mRgba, kernMat);
        break;
case 14:
        filename="Cyan";
        VisiBility(2);
        im.RGB(0, 255, 255);
        mRgba = im.RGBconv(imgToProcess, mRgba, kernMat);
        break;
case 15:
        filename="Gray";
        VisiBility(2);
        im.RGB(119, 119, 119);
        mRgba = im.RGBconv(imgToProcess, mRgba, kernMat);
        break;
case 30:
        filename="Flip";
        VisiBility(1);
        Core.flip(imgToProcess, mRgba, 0);

        break;
case 31:

        filename = "Grayscale";
        VisiBility(5);
```

```
                Imgproc.cvtColor(imgToProcess, mRgba,
Imgproc.COLOR_BGR2GRAY);

                Imgproc.cvtColor(mRgba, mRgba,
Imgproc.COLOR_GRAY2RGBA, 4);



                break;

            case 32:



                filename="Canny";

                VisiBility(3);

                Imgproc.cvtColor(imgToProcess, mRgba,
Imgproc.COLOR_BGRA2GRAY, 4);

                Imgproc.Canny(mRgba, mRgba, 590, 600, 3);

                Imgproc.cvtColor(mRgba, mRgba,
Imgproc.COLOR_GRAY2RGBA, 4);



                break;

            case 33:



                filename="Laplace";

                VisiBility(5);

                Imgproc.cvtColor(imgToProcess, mRgba,
Imgproc.COLOR_BGRA2GRAY);

                Imgproc.Laplacian(mRgba, mRgba, mRgba.depth());

                Imgproc.cvtColor(mRgba, mRgba,
Imgproc.COLOR_GRAY2RGBA, 4);



                break;

            case 34:



                filename="Negative";
```

```
                VisiBility(5);
                Core.bitwise_not(imgToProcess, mRgba);

                break;
            case 35:

                filename="GaussianBlur";
                VisiBility(6);
                Imgproc.GaussianBlur(imgToProcess, mRgba, new Size(1, 1), 40);

                break;
            case 36:
                filename="onesfilter";
                VisiBility(5);
                kernMat = Mat.ones(new Size(4, 4), CvType.CV_64FC1);
                Imgproc.filter2D(imgToProcess, mRgba, 64, kernMat);

                break;
            case 37:
                filename="eyefilter";
                VisiBility(5);
                kernMat = Mat.eye(new Size(4, 4), CvType.CV_64FC1);
                Imgproc.filter2D(imgToProcess, mRgba, 64, kernMat);

                break;
            case 38:

                filename="morphologyEx";
```

```
                VisiBility(5);
                kernel = Mat.ones(3, 3, CvType.CV_32F);
                Imgproc.morphologyEx(imgToProcess, mRgba,
Imgproc.MORPH_GRADIENT,
                        kernel);

                break;
            case 39:
                filename="copyMakeBorder";
                VisiBility(5);
                Imgproc.copyMakeBorder(imgToProcess, mRgba, 300, 300, 300,
300,
                        Imgproc.BORDER_REPLICATE);

                break;
            case 40:
                filename="sobel";
                VisiBility(5);
                Imgproc.Sobel(imgToProcess, mRgba, imgToProcess.depth(), 1, 1);

                break;
            case 41:
                filename="threshold";
                VisiBility(9);

                Imgproc.cvtColor(imgToProcess, mRgba,
Imgproc.COLOR_BGR2GRAY);
                Imgproc.cvtColor(mRgba, mRgba,
Imgproc.COLOR_GRAY2RGBA, 4);
```

```
                    Imgproc.threshold(mRgba, mRgba, 0, 0,
Imgproc.THRESH_TOZERO);

                    break;

            default:
                    break;
            }

            bmpOut = Bitmap.createBitmap(mRgba.cols(), mRgba.rows(),
                            Bitmap.Config.ARGB_8888);
            Utils.matToBitmap(mRgba, bmpOut);

            bmpOut2 = bmpOut;
            imgToProcess = Utils.bitmapToMat(bmp);
            kernMat = new Mat();
            im = new Filters();
            mRgba = new Mat(imgToProcess, Range.all());
            bmpOut = Bitmap.createBitmap(mRgba.cols(), mRgba.rows(),
                            Bitmap.Config.ARGB_8888);
            Utils.matToBitmap(mRgba, bmpOut);
            return bmpOut2;
    }

    public void show() {
            m = new Scalar(im.getRed(), im.getGreen(), im.getBlue());
            kernMat = new Mat(imgToProcess, Range.all());
            kernMat = new Mat(kernMat.size(), imgToProcess.type(), m);
```

```
                Core.add(imgToProcess, kernMat, mRgba);
                bmpOut = Bitmap.createBitmap(mRgba.cols(), mRgba.rows(),
                                Bitmap.Config.ARGB_8888);
                Utils.matToBitmap(mRgba, bmpOut);

                bmpOut2 = bmpOut;
                imgToProcess = Utils.bitmapToMat(bmp);
                kernMat = new Mat();

                mRgba = new Mat(imgToProcess, Range.all());
                bmpOut = Bitmap.createBitmap(mRgba.cols(), mRgba.rows(),
                                Bitmap.Config.ARGB_8888);
                Utils.matToBitmap(mRgba, bmpOut);
                imv.setImageBitmap(bmpOut2);

        }

        public void canny(int canny1, int canny2) {
                c1 = canny1;
                c2 = canny2;
        }

        public void showcanny() {

                Imgproc.cvtColor(imgToProcess, mRgba, Imgproc.COLOR_BGRA2GRAY,
4);
                Imgproc.Canny(mRgba, mRgba, c1, c2, 5);
                Imgproc.cvtColor(mRgba, mRgba, Imgproc.COLOR_GRAY2RGBA, 4);
                if (cwb == 0) {
```

```
                Core.bitwise_not(mRgba, mRgba);
        }
        bmpOut2 = Bitmap.createBitmap(mRgba.cols(), mRgba.rows(),
                        Bitmap.Config.ARGB_8888);
        Utils.matToBitmap(mRgba, bmpOut2);
        imv.setImageBitmap(bmpOut2);

}

public Mat flip(int flip) {

        Core.flip(imgToProcess, mRgba, flip);
        return mRgba;

}

public void VisiBility(int visible) {
        if (visible == 1) {
                flipLL.setVisibility(View.VISIBLE);
                cannyLL.setVisibility(View.GONE);
                brightnessLL.setVisibility(View.GONE);
                rgbLL.setVisibility(View.GONE);
                blurLL.setVisibility(View.GONE);
                saveLL.setVisibility(View.GONE);
                resizeLL.setVisibility(View.GONE);
                thresholdLL.setVisibility(View.GONE);

        } else if (visible == 2) {
```

```
            flipLL.setVisibility(View.GONE);
            cannyLL.setVisibility(View.GONE);
            brightnessLL.setVisibility(View.GONE);
            rgbLL.setVisibility(View.VISIBLE);
            blurLL.setVisibility(View.GONE);
            saveLL.setVisibility(View.GONE);
            resizeLL.setVisibility(View.GONE);
            thresholdLL.setVisibility(View.GONE);
        } else if (visible == 3) {
            flipLL.setVisibility(View.GONE);
            cannyLL.setVisibility(View.VISIBLE);
            brightnessLL.setVisibility(View.GONE);
            rgbLL.setVisibility(View.GONE);
            blurLL.setVisibility(View.GONE);
            saveLL.setVisibility(View.GONE);
            resizeLL.setVisibility(View.GONE);
            thresholdLL.setVisibility(View.GONE);
        } else if (visible == 4) {
            flipLL.setVisibility(View.GONE);
            cannyLL.setVisibility(View.GONE);
            brightnessLL.setVisibility(View.VISIBLE);
            rgbLL.setVisibility(View.GONE);
            blurLL.setVisibility(View.GONE);
            saveLL.setVisibility(View.GONE);
            resizeLL.setVisibility(View.GONE);
            thresholdLL.setVisibility(View.GONE);
        } else if (visible == 5) {
            flipLL.setVisibility(View.GONE);
```

```
            cannyLL.setVisibility(View.GONE);
            brightnessLL.setVisibility(View.GONE);
            rgbLL.setVisibility(View.GONE);
            blurLL.setVisibility(View.GONE);
            saveLL.setVisibility(View.GONE);
            resizeLL.setVisibility(View.GONE);
            thresholdLL.setVisibility(View.GONE);
        } else if (visible == 6) {
            flipLL.setVisibility(View.GONE);
            cannyLL.setVisibility(View.GONE);
            brightnessLL.setVisibility(View.GONE);
            rgbLL.setVisibility(View.GONE);
            blurLL.setVisibility(View.VISIBLE);
            saveLL.setVisibility(View.GONE);
            resizeLL.setVisibility(View.GONE);
            thresholdLL.setVisibility(View.GONE);
        } else if (visible == 7) {
            flipLL.setVisibility(View.GONE);
            cannyLL.setVisibility(View.GONE);
            brightnessLL.setVisibility(View.GONE);
            rgbLL.setVisibility(View.GONE);
            blurLL.setVisibility(View.GONE);
            saveLL.setVisibility(View.VISIBLE);
            resizeLL.setVisibility(View.GONE);
            thresholdLL.setVisibility(View.GONE);
        } else if (visible == 8) {
            flipLL.setVisibility(View.GONE);
            cannyLL.setVisibility(View.GONE);
```

```
                    brightnessLL.setVisibility(View.GONE);
                    rgbLL.setVisibility(View.GONE);
                    blurLL.setVisibility(View.GONE);
                    saveLL.setVisibility(View.GONE);
                    resizeLL.setVisibility(View.VISIBLE);
                    thresholdLL.setVisibility(View.GONE);
                }else if (visible == 9) {
                    flipLL.setVisibility(View.GONE);
                    cannyLL.setVisibility(View.GONE);
                    brightnessLL.setVisibility(View.GONE);
                    rgbLL.setVisibility(View.GONE);
                    blurLL.setVisibility(View.GONE);
                    saveLL.setVisibility(View.GONE);
                    resizeLL.setVisibility(View.GONE);
                    thresholdLL.setVisibility(View.VISIBLE);
                }

        }
}
```

Filters

```
package gr.epp2394.OpenCVAndroid;

import org.opencv.android.Utils;
import org.opencv.core.Core;
import org.opencv.core.Mat;
import org.opencv.core.Range;
import org.opencv.core.Scalar;
import org.opencv.imgproc.Imgproc;

import android.graphics.Bitmap;

public class Filters {
	public int red, green, blue, c1, c2;
	public float resize;
	public Mat kernMat, imgToProcess, mRgba;
	public Scalar m;
	public Bitmap bmpOut;

	// public LinearLayout flipLL, rgbLL, brightnessLL, cannyLL, blurLL, saveLL,
	// resizeLL;

	public Mat imageToMat(Bitmap img) {
		imgToProcess = Utils.bitmapToMat(img);
		return imgToProcess;
	}

	public Mat clcmRgba() {
```

```
            mRgba = new Mat(imgToProcess, Range.all());
            return mRgba;
    }

    public Mat clckernMat() {
            kernMat = new Mat(imgToProcess, Range.all());
            kernMat = new Mat(kernMat.size(), imgToProcess.type(), m);
            return kernMat;

    }

    public void RGB() {
            red = 0;
            green = 0;
            blue = 0;
    }

    public void RGB(int r, int g, int b) {
            red = r;
            green = g;
            blue = b;

    }

    public int getBlue() {
            return blue;
    }
```

```
public int getGreen() {
        return green;
}

public int getRed() {
        return red;
}

public void setBlue(int blue) {
        this.blue = blue;
}

public void setGreen(int green) {
        this.green = green;
}

public void setRed(int red) {
        this.red = red;
}

public Mat RGBconv(Mat src, Mat dst, Mat kernel) {
        m = new Scalar(this.red, this.green, this.blue);
        kernel = new Mat(src, Range.all());
        kernel = new Mat(kernel.size(), src.type(), m);

        Core.add(src, kernel, dst);
        return dst;
}
```

```
public Bitmap showcanny(Mat src, Mat dst, int x) {

        Imgproc.cvtColor(src, dst, Imgproc.COLOR_BGRA2GRAY, 4);
        Imgproc.Canny(dst, dst, c1, c2, 5);
        Imgproc.cvtColor(dst, dst, Imgproc.COLOR_GRAY2RGBA, 4);
        if (x == 0) {
                Core.bitwise_not(dst, dst);
        }
        bmpOut = Bitmap.createBitmap(dst.cols(), dst.rows(),
                        Bitmap.Config.ARGB_8888);
        Utils.matToBitmap(dst, bmpOut);
        return bmpOut;

}

public void canny(int canny1, int canny2) {
        c1 = canny1;
        c2 = canny2;
}

public int getC1() {
        return c1;
}

public int getC2() {
        return c2;
}
```

```
	public void setC1(int c1) {
		this.c1 = c1;
	}

	public void setC2(int c2) {
		this.c2 = c2;
	}

}
```