ΤΕΙ ΔΥΤΙΚΗΣ ΕΛΛΑΔΑΣ

ΣΧΟΛΗ ΔΙΟΙΚΗΣΗΣ ΚΑΙ ΟΙΚΟΝΟΜΙΑΣ

ΤΜΗΜΑ ΔΙΟΙΚΗΣΗΣ ΕΠΙΧΕΙΡΗΣΕΩΝ / ΜΕΣΟΛΟΓΓΙ

Πτυχιακή εργασία

ΔΗΜΙΟΥΡΓΙΑ ΨΥΧΑΓΩΓΙΚΟΥ ΠΑΙΧΝΙΔΙΟΥ ΣΕ ΛΕΙΤΟΥΡΓΙΚΟ ΣΥΣΤΗΜΑ ANDROID [SEAGAME]

[Παπαζώης Παύλος]

AM:14515

Επιβλέπων καθηγητής

[Ιωάννης Ντόκας]

Μεσολόγγι [2015]

Η έγκριση της πτυχιακής εργασίας από το Τμήμα Διοίκησης Επιχειρήσεων/Μεσολογγίου του ΤΕΙ Δυτικής Ελλάδας δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα εκ μέρους του Τμήματος.

ΠΕΡΙΛΗΨΗ

Βασικός σκοπός της πτυχιακής εργασίας είναι η δημιουργία μιας εφαρμογής Android. Η εφαρμογή αυτή υλοποιήθηκε χρησιμοποιώντας την πλατφόρμα Eclipse. Τα κίνητρα για την εκπόνηση της εργασίας είναι πρώτον η ραγδαία εξάπλωση των εφαρμογών Android και δεύτερον λόγο του ανοιχτού κώδικα που μας δίνει την δυνατότητα ελεύθερης δημιουργίας. Στόχοι της εργασίας είναι οι εξής: α) Προσωπική εκμάθηση και β) Να παρουσιάσουμε βήμα - βήμα τον τρόπο δημιουργίας της εφαρμογής ώστε να ενθαρρυνθούν και άλλοι φοιτητές να ασχοληθούν με Android. Για να πετύχουμε τους στόχους μας ξεκινάμε με μία ιστορική αναδρομή στο λογισμικό Android και στις εκδόσεις του. Στην συνέχεια αναφέρουμε τα εργαλεία που χρησιμοποιήσαμε για την δημιουργίας της εφαρμογής μας καθώς και τις απαιτήσεις αυτής. Ακολουθεί το βασικότερο μέρος της πτυχιακής το οποίο είναι βέβαια η λεπτομερής μέθοδος δημιουργίας της. Έχοντας ολοκληρώσει την αρχική μορφή της εφαρμογής περιγράφουμε πως μπορούμε να την ανεβάσουμε στο google market από το οποίο θα είναι διαθέσιμη για να την κατεβάσει οποιοσδήποτε έχει λογισμικό Android 4.1.2.. Η εφαρμογή λοιπόν που δημιουργήσαμε την ονομάσαμε Seagame. Στην βασική της μορφή η εφαρμογή Seagame είναι ένα παιχνίδι για την έκδοση Android 4.1.2. στο οποίο ο χρήστης αγγίζοντας της οθόνη κατευθύνει ένα υποβρύχιο στην συλλογή θησαυρών αποφεύγοντας παράλληλα τα εμπόδια για να μην ηττηθεί. Στην τωρινή η εφαρμογή έχει μόνο μία πίστα και ένα στάδιο δυσκολίας. Μπορεί όμως να βελτιωθεί με την προσθήκη και άλλων πιστών με διαφορετικά εμπόδια και δυσκολία αλλά και με μικρότερες προσθήκες όπως η αποθήκευση του σκορ.

Πίνακας Περιεχομένων

ΠΙΝΑΚΑΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΑΠΟΔΟΣΗ ΟΡΩΝ

Πίνακας 1: Απόδοση Όρων

Project	Πρότζεκτ-έργο
Multimedia	Πολυμέσα
Multitasking	Πολυδιεργασία
Web Browsing	Διαδικτυακή Αναζήτηση
GPS	Παγκόσμιο Σύστημα Θεσιθεσίας
Qwerty	Αμετάφραστο
Google maps	Χάρτες της Google
Google talk	Ομιλία της Google
Micro SD.	Κάρτα Μνήμης
Bookmarks	Σελιδοδείκτης
UI	Επιφάνει Χρήστη
Multi-touch	Πολλαπλό Άγγιγμα
Wallpapers	Ταπετσαρία
Market	Αγορά
Google e-books,	Ηλεκτρονικά Βιβλία της Google
USB	Εννιαίος Σειριακός Δίαυλος
Zoom to Fill	Ζουμ
Java JDK	Εργαλεία Ανάπτυξης Java
Android SDK	Εργαλεία Ανάπτυξης Λογισμικού
Android Virtual Device	Εικονική Συσκευή Android
Window	Παράθυρο
Start	Έναρξη
Launch	Εκτόξευση
File	Αρχείο
New	Νέο
Application	Εφαρμογή
Driver	Πρόγραμμα Οδήγησης
Main Menu	Βασικό Μενού
Game	Παιχνίδι
Pause Menu	Μενού Παύσης
Continue	Συνέχεια
Background	Φόντο
WIN/LOSE	Νίκη - Ήττα
Score	Σκορ
game-play	Τρόπος Παιχνιδιού
Ship	Πλοίο
Barrier	Εμπόδιο
Bonus	Μπόνους
Obstacle	Εμπόδιο
Dpi	Τελείες ανα ίντσα
Folder	Φάκελος
Package	Πακέτο
Layout	Διάταξη

Activity (Java)	Ένα παράθυρο που αλληλεπηδρά μέσο της επιφάνειας
XML	Γλώσσα σήμανσης, που περιέχει σύνολο κανόνων για την ηλεκτρονική κωδικοποίηση κειμένων
Portrait	Πορτραίτο
Landscape	Τοπίο
Mode	Λειτουργία
Relative layout	Σχετικό Σχέδιο
Strings	Νήμα
Text	Κείμενο
Medium text	Μεσαίο Κείμενο
Identifiers	Αναγνωριστές
Id	Αναγνωριστικό
ImageButton	Κουμπί Εικόνας
TextView	Όψη Κειμένου
OnCreate	Μέθοδος Δημιουργίας στο Eclipse
FindViewById	Μέθοδος Εύρεσης Αντικειμένου Απο Τον Αναγνωριστή Του
Add	Προσθήκη
Root Element	Ριζικό Στοιχείο
Layer	Στρώμα
Linear layout	Γραμμικό Σχέδιο
MainThread	Κύριο Νήμα
Surface	Επιφάνεια
Events	Γεγονότα
OnTouchEventHandle	Μέθοδος Του Eclipse Η Οποία Χειρίζεται Τα Γεγονότα
Constructor	Κατασκευαστής
Draw	Σχεδιάζω
Update	Αναβάθμηση
canvas	Κανβάς
Bitmap	Δυφιοαπεικόνηση
Screen	Οθόνη
ScreenWidth	Πλάτος Οθόνης
Null	Κενό
Shipspeed	Ταχύτητα Πλοίου
ArrayList	Λίστα Σειράς
Death	Θάνατος
Booms	Εκρήξεις
SetBoomAnimation	Ορισμός Απεικόνησης Έκρηξης
AnimTime	Χρόνος Απεικόνησης
ScreenHeight	Ύψος Οθόνης
Bump	Εξόγκωμα
True	Αληθές
False	Ψευδές
MotionEvent	Γεγονός Κίνησης

shipheight	΄Υψος Πλοίου
Generate	Παράγω
Center	Κέντρο
Coin	Νόμισμα
Google Play	Ηλεκτρονικό Μαγαζί Της Google Απο Το Οποίο Μπορεί Κάποιος Να Κατεβάσει Εφαρμογές
Android Icon Set.	Ορισμός Εικονιδίου Android
Other	Άλλα
Finish	Τέλος
Application.	Εφαρμογή
Export	Εξαγωγή

ΕΙΣΑΓΩΓΗ

Στα παρακάτω κεφάλαια εξηγούνται αναλυτικότερα τα βήματα τα οποία ακολουθήθηκαν έτσι ώστε να δημιουργηθεί σε μορφή Beta η εφαρμογή **Seagame** η οποία θα τρέχει σε **Android 4.1.2. Samsung Galaxy S3 Mini** καθώς όμως αναφέρονται και κάποια γενικά πράγματα τα οποία είναι καλό να γνωρίζουμε για το λογισμικό Android. Η δημιουργία εφαρμογών Android αποτελεί ένα από τα σημαντικότερα πεδία έρευνας στον χώρο της πληροφορικής. Η χρήση και εξάπλωση των κινητών οδηγεί όλο και περισσότερους να ασχοληθούν με την δημιουργία διάφορων εφαρμογών για ψυχαγωγικούς, για εκπαιδευτικούς είτε ακόμα για επαγγελματικούς σκοπούς με αποτέλεσμα ο κλάδος να αναπτύσσεται ραγδαία. Στόχοι της πτυχιακής είναι να αποκτήσω ο ίδιος μια οικειότητα με το λογισμικό Android να εμβαθύνω τις γνώσεις μου στον προγραμματισμό καθώς επίσης και να μπορούνε άλλοι να επωφεληθούν από το κείμενο μου. Η εφαρμογή αυτή η οποία έχει ψυχαγωγικό χαρακτήρα είναι απλή στην χρήση της. Αγγίζοντας την οθόνη με το δάκτυλο ο χρήστης κατευθύνει είτε προς τα πάνω είτε προς τα κάτω το υποβρύχιο με σκοπό να αποφύγει τα εμπόδια και να μαζέψει θησαυρούς

ΚΕΦΑΛΑΙΟ 1: Ιστορικό Android

1.1. Τι είναι το Android;

Το Android είναι ένα λειτουργικό σύστημα ανοιχτού κώδικα, το οποίο έχει βασιστεί στο Linux, για φορητές συσκευές όπως smartphones και tablets. Αναπτύχθηκε από την Google και αργότερα από την Open Handset Alliance η οποία είναι μια κοινοπραξία εταιριών λογισμικού, κατασκευής hardware και τηλεπικοινωνιών. Οι τελευταίες ασχολούνται με την ανάπτυξη και την εξέλιξη ανοιχτών προτύπων στις φορητές συσκευές. Η πρώτη παρουσίαση της πλατφόρμας Android έγινε στις 5 Νοεμβρίου 2007, ταυτόχρονα με την ανακοίνωση της ίδρυσης του οργανισμού Open Handset Alliance. Η Google δημοσίευσε το μεγαλύτερο μέρος του κώδικα του Android υπό τους όρους της Apache License, μιας ελεύθερης άδειας λογισμικού. Η Google, τον Ιούλιο του 2005, εξαγόρασε την Android Inc, μια μικρή εταιρεία με έδρα το Palo Alto στην California των ΗΠΑ. Εκείνη την εποχή πολύ λίγα πράγματα ήταν γνωστά για τις λειτουργίες της Android Inc, εκτός του ότι ανέπτυσσαν λογισμικό για κινητά τηλέφωνα. Άρχισε έτσι να φημολογείται ότι η Google σχεδίαζε να διεισδύσει στην αγορά κινητής τηλεφωνίας. Στην Google, η ομάδα με επικεφαλής τον Andy Rubin ανέπτυξε μια κινητή πλατφόρμα που στηρίζεται στον πυρήνα του Linux, την οποία προώθησαν με την παροχή ενός ευέλικτου, αναβαθμίσιμου συστήματος. Έχει αναφερθεί ότι η Google είχε ήδη συγκεντρώσει μια σειρά από εταίρους hardware και software και επισήμανε στους παρόδους ότι ήταν ανοικτή σε διάφορους βαθμούς συνεργασίας εκ μέρους της. Ως αποτέλεσμα, εντυπα και ηλεκτρονικά μέσα ενημέρωσης σύντομα ανέφεραν φήμες ότι η Google ανέπτυσσε μια Google-branded συσκευή. Περισσότερες φήμες ακολούθησαν, οι οποίες ανέφεραν ότι η Google καθόριζε τις τεχνικές

Εικόνα 1: Λογότυπο Android

προδιαγραφές και έδειχνε πρωτότυπα στους κατασκευαστές κινητών τηλεφώνων και τους φορείς δικτύων. Εν κατακλείδι, η Google παρουσίασε το smartphone της Nexus One το οποίο χρησιμοποιεί το open source λειτουργικό σύστημα Android. Η συσκευή κατασκευάστηκε από την HTC , και έγινε διαθέσιμη στις 5 Ιανουαρίου 2010.

1.2. Εφαρμογές Android

Το Android έχει μια μεγάλη κοινότητα προγραμματιστών που γράφουν εφαρμογές, οι οποίες είναι σε θέση επεκτείνουν τη λειτουργικότητα των συσκευών. Οι εφαρμογές αυτές γράφονται σε μια προσαρμοσμένη έκδοση της JAVA και είναι διαθέσιμες από το online κατάστημα Google Play (πρώην Android Market) της Google όπως και από άλλα sites. Μέχρι τον Φεβρουάριο του 2012 περισσότερες από 450.000 εφαρμογές ήταν διαθέσιμες για Android ενώ ο αριθμός των downloads από το Android Market μέχρι το Δεκέμβριο του 2011 υπολογίζεται ότι είχε υπερβεί τα 10 δισεκατομμύρια. Αξίζει να σημειωθεί ότι το Android είναι η πρώτη σε πωλήσεις παγκοσμίως πλατφόρμα για smartphones αφού μέχρι το Φεβρουάριο του 2012 μετρούσε περισσότερες από 300 εκατομμύρια συσκευές σε χρήση.

1.3. Εξέλιξη του Android

Όπως αναφέραμε παραπάνω, το Android είναι ένα λειτουργικό σύστημα ανοιχτού κώδικα. Η εξέλιξη του, χάρις στην open source φύση του είναι ραγδαία. Αυτό είναι φανερό από το γεγονός ότι οι 7 κύριες εκδόσεις του έχουν κυκλοφορήσει σε διάστημα 2.5 ετών, από τον Απρίλη του 2009 μέχρι τον Νοέμβριο του 2011 (Εικόνα 1.3). Επίσης είναι απαραίτητο να αναφερθεί πως στην πληροφορική συνηθίζεται τα προϊόντα hardware και software να κυκλοφορούν εκτός από τον αριθμό έκδοσης τους, και με μία κωδική ονομασία. Ως εκ τούτου η κωδική ονομασία μπορεί να ποικίλει και να περιλαμβάνει για παράδειγμα ονόματα πόλεων (Windows Viena, Chicago) ή ονόματα ζώων (OSX Leopard,Lion). Στην περίπτωση βέβαια του Android τα κώδικα ονόματα έρχονται στη μορφή επιδόρπιου!

1.3.1. Android 1.5 Cupcake

Εικόνα 2: Λογότυπο Cupcake

Συνεχίζοντας, η έκδοση “Cupcake”, βασισμένη στο Linux Kernel 2.6.27, παρουσιάστηκε στις 30 Απριλίου του 2009. Η συγκεκριμένη έκδοση δύναται να υποστηρίξει νέες λειτουργίες για την κάμερα της συσκευής. Αναλυτικά, αυτές είναι η καταγραφή και η παρακολούθηση βίντεο από την λειτουργία της κάμερας και η άμεση μεταφόρτωση του βίντεο αλλά και των φωτογραφιών στο Youtube και στο Picasa αντίστοιχα απευθείας από το τηλέφωνο. Διαθέτει επίσης, νέο έξυπνο πληκτρολόγιο με πρόβλεψη κειμένου. Υποστηρίζει πρότυπο Bluetooth A2DP και AVRCP ενώ έχει και την ικανότητα να συνδέεται αυτόματα σε μικροσυσκευές Bluetooth από μια συγκεκριμένη απόσταση. Ακόμα, στην έκδοση αυτή υπάρχει νέο γραφικό περιβάλλον με κινούμενες μεταβάσεις οθόνης.

1.3.2. Android 1.6 Donut

Εικόνα 3: Λογότυπο Donut

Η έκδοση “Donut”, βασισμένη στο Linux Kernel 2.6.29, παρουσιάστηκε στις 15 Σεπτεμβρίου του 2009. Έχει ταχύτερη απόκριση σε σχέση με την προηγούμενη έκδοση. Στην συγκεκριμένη έκδοση υποστηρίζεται πλέον η επιλογή πολλαπλών αρχείων ταυτόχρονα. Διαθέτει ανανεωμένη γκάλερι και φωτογραφική μηχανή, καθώς και βελτιωμένο Android Market. Έχει ανανεωμένη φωνητική αναζήτηση, με ταχύτερη απόκριση και βαθύτερη ολοκλήρωση με εγγενείς (native) εφαρμογές, συμπεριλαμβανομένης της δυνατότητας κλήσης επαφών. Δίνει επίσης τη δυνατότητα αναζήτησης σελιδοδεικτών, ιστορικού, επαφών αλλά και πρόσβασης στο διαδίκτυο από την αρχική οθόνη. Παρέχεται υποστήριξη για ανάλυση οθονών WVGA, ανανεωμένη υποστήριξη τεχνολογιών για CDMA/EVDO, 802.1x, VPNs και μηχανή μετατροπής κειμένου σε ομιλία (text-tospeech).

1.3.3. Android 2.0/2.1 Eclair

Εικόνα 4: Λογότυπο Eclair

Η έκδοση “Eclair”, βασισμένη και αυτή στον Linux Kernel 2.6.29, παρουσιάστηκε στις 26 Οκτωβρίου του 2009 και τον Ιανουάριο του 2010 επανεκδόθηκε σε Android 2.1 Eclair (MR1).Σε αυτή την έκδοση υπάρχει ακόμα ταχύτερη απόκριση του υλικού σε σχέση με τις δυο προηγούμενες και επιπλέον υποστηρίζονται

περισσότερες οθόνες και αναλύσεις. Διαθέτει επίσης, νέο browser ο οποίος μπορει να υποστηρίξει το πρότυπο HTML5, νέο User Interface, και βελτιωμένους χάρτες Google (Google Maps 3.1.2). Παράλληλα, έχει ενσωματωθεί η υποστήριξη φλας για την κάμερα η οποία έχει πλέον και ψηφιακό zoom. Επίσης, έχει βελτιωθεί η κλάση MotionEvent ώστε να υπάρχει η δυνατότητα για γεγονότα πολλαπλής αφής (multitouch events). Υποστηρίζεται τέλος, Bluetooth 2.1 ενώ έχει βελτιωθεί επισης και το πληκτρολόγιο.

1.3.4. Android 2.2 Froyo

Εικόνα 5: Λογότυπο Froyo

Η έκδοση "Froyo", βασισμένη στο Linux Kernel 2.6.32, παρουσιάστηκε στις 20 Μαΐου του 2010. Αξίζει να αναφέρθει ότι υπάρχουν βελτιστοποιήσεις οι οποίες στοχεύουν στην ταχύτητα γενικά του λειτουργικού συστήματος, στην μνήμη και στην απόδοση. Έχει ενσωματωθεί επιπλεόν ο μηχανισμός JavaScript του Chrome V8 στον browser, υπάρχει παράλληλα Adobe Flash 10.1, ενώ υποστηρίζεται πιο καλά και το Microsoft Exchange. Έχει πραγματοποιηθεί ταυτοχρονα, ανανέωση του Android Market. Ο χρήστης είναι σε θέση πλέον, να ελέγχει αν θα γίνεται ή όχι κίνηση πακέτων δεδομένων από το δίκτυο κινητής τηλεφωνίας. Υπάρχει βέβαια και η δυνατότητα εγκατάστασης εφαρμογών στην κάρτα μνήμης αλλά και η μεταφορά τους εκεί από τη μνήμη του τηλεφώνου. Τέλος, το τηλέφωνο πλέον μπορεί να μετατραπεί σε WiFi hotspot.

1.3.5. Android 2.3 Gingerbread

Εικόνα 6: Λογότυπο Gingerbread

Η έκδοση "Gingerbread", είναι βασισμένη στο Linux Kernel 2.6.35.7. Παρουσιάστηκε στις 6 Δεκεμβρίου του 2010, ενώ τον Φεβρουάριο του 2011 επανεκδόθηκε σε Android 2.3.3. Στην έκδοση αυτή παρατηρούνται αλλαγές στο User Interface το οποίο έχει γίνει πιο απλό και ταχύ. Ακόμα υπάρχει πλέον δυνατότητα να υποστηρίχθούν πλέον οθόνες μεγάλων μεγεθών και αναλύσεων. Υπάρχει ταυτόχρονα, το πρωτόκολλο SIP για κλήσεις μέσω VoIP, υποστηρίζεται ο τύπος βίντεο WebM/VP8 και ο κωδικοποιητής AAC. Έχει πραγματοποιηθεί βελτίωση στον ήχο καθώς και στις λειτουργίες απεικόνισης για την ανάπτυξη παιχνιδιών. Υπάρχει η δυνατότητα για Copy-Paste σε όλο το σύστημα και όχι μόνο στην ίδια εφαρμογή. Υποστηρίζεται το NFC (Near Field Communication) αλλά και η ύπαρξη πολλαπλών καμερών. Τέλος, έχει βελτιωθεί η ενεργειακή υποστήριξη και έχει γίνει μετάβαση από το σύστημα αρχείων YAFFS στο ext4 στις νέες συσκευές.

1.3.6. Android 3.0 Honeycomb

Εικόνα 7: Λογότυπο Honeycomb

Η έκδοση “Honeycomb” είναι βασισμένη στο Linux Kernel 2.6.36 και παρουσιάστηκε στις 9 Μαΐου του 2011, με την ιδιαιτερότητα ότι προοριζόταν μόνο για tablets. Οι αλλαγές που έγιναν στην έκδοση αυτή στοχεύουν κυρίως στη βελτίωση της υποστήριξης των tablets. Υπάρχει ένα νέο, απόλυτα διαφορετικό, User Interface και υποστηρίζονται διπύρηνοι και τετραπύρηνοι επεξεργαστές. Παράλληλα, εχει απλοποιηθεί το multitasking με σκοπό ο χρήστης να μπορεί με τη χρήση ενός πλήκτρου (recent apps) να περνάει από μια εφαρμογή σε άλλη. Η συγκεκριμένη εκδοση δίνει ακόμα τη δυνατότητα για Video Chat μέσω της εφαρμογής Google Talk αλλά και για ανάγνωση βιβλίων μέσω του Google eBooks. Τέλος, μπορούν να κρυπτογραφηθούν όλα τα δεδομένα χρήστη.

1.3.7. Android 4.0 Ice Cream Sandwich

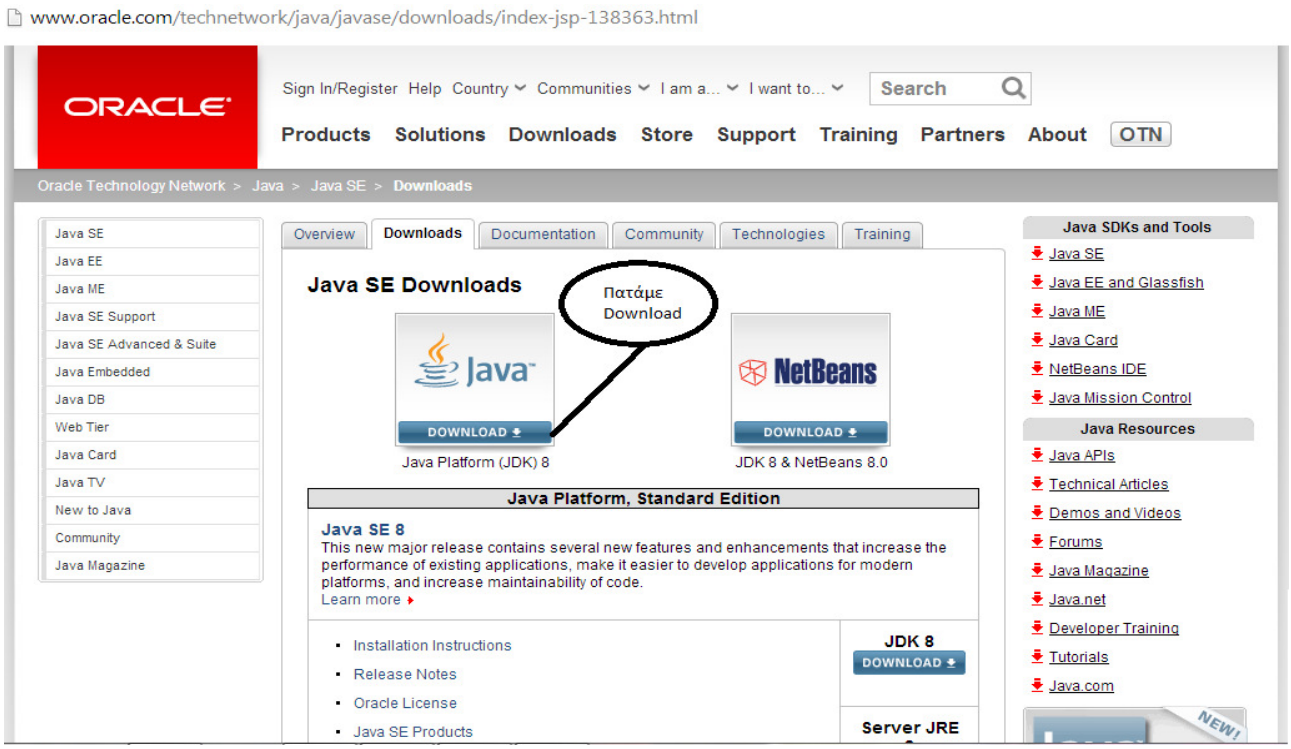
Η έκδοση “Ice Cream Sandwich”, η οποία ειίναι βασισμένη στο Linux Kernel 3.0.1, παρουσιάστηκε στις 19 Οκτωβρίου του 2011. Και σε αυτή την εφαρμογή, έχει βελτιωθεί η ταχύτητα και η απόδοση του συστήματος. Στο User Interface, το οποίο είναι και παλι διαφορετικό, υπάρχουν εικονικά πλήκτρα τα οποία παίρνουν τη θέση των φυσικών ή αφής που υπήρχαν στις συσκευές. Υπάρχει σαφής βελτίωση της ασφάλειας του συστήματος με την προσθήκη αναγνώρισης προσώπου για να ξεκλειδώσει η συσκευή. Ο browser δύναται να ανοίξει ταυτόχρονα μέχρι και 16 καρτέλες. Ο χρήστης έχει τη δυνατότητα να τερματίσει εφαρμογές οι οποίες τρέχουν στο background, ενώ μπορεί να θέσει και όρια στην κίνηση πακέτων δεδομένων. Η εφαρμογή Android Beam αξιοποιεί πλέον το NFC αφού επιτρέπει την αποστολή δεδομένων από τη συσκευή σε όσες συσκευές βρίσκονται εντός μιας μικρής ακτίνας εμβέλειας. Επιπλέον, με την ύπαρξη του Wi-Fi Direct συσκευές μπορούν να συνδεθούν μεταξύ τους ασύρματα χωρίς την μεσολάβηση κάποιου access point. Τέλος, υποστηρίζεται η εγγραφή βίντεο σε 1080p.

Εικόνα 8: Λογότυπο Sandwich

ΚΕΦΑΛΑΙΟ 2: Εγκατάσταση Λογισμικού

2.1. Εργαλεία για την δημιουργία της εφαρμογής μας

Αρχικά για την δημιουργία της εφαρμογής κατεβάσαμε το Java JDK το οποίο είναι τα εργαλεία αναπτύξης Java μέσα απο το Site της Oracle.

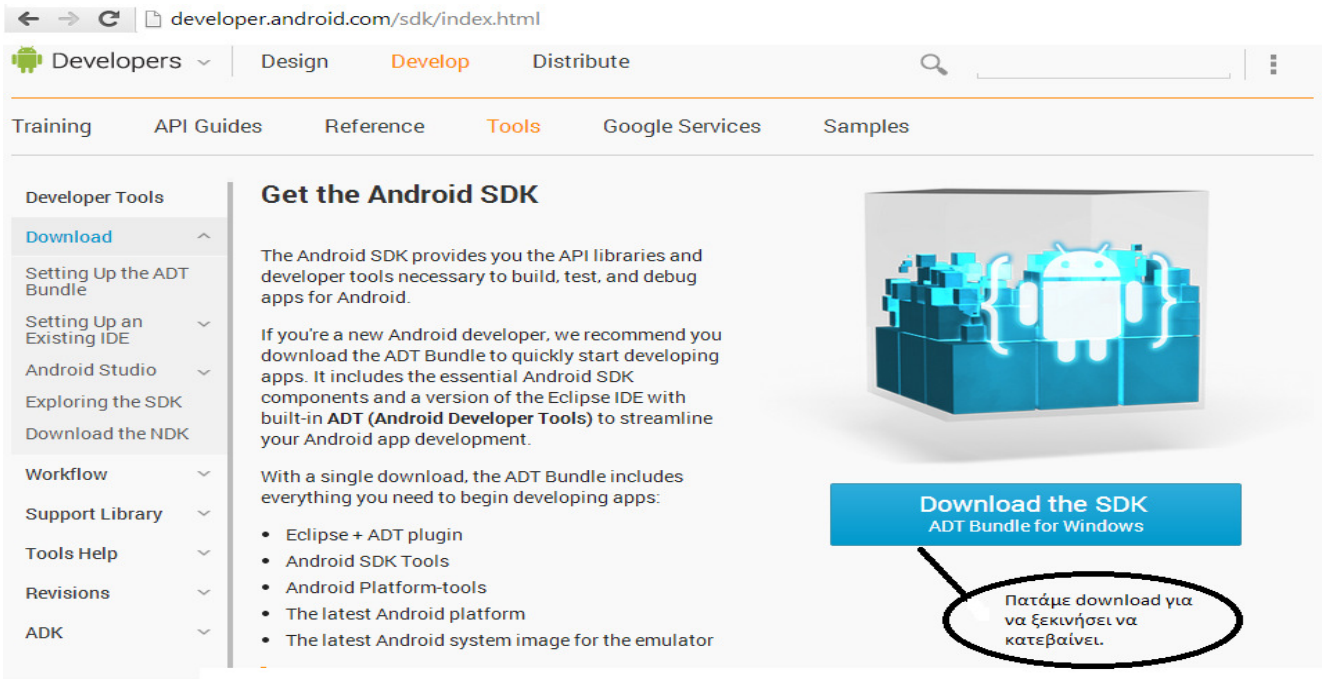


Εικόνα 9: Κατεβάζοντας το αρχείο Java JDK

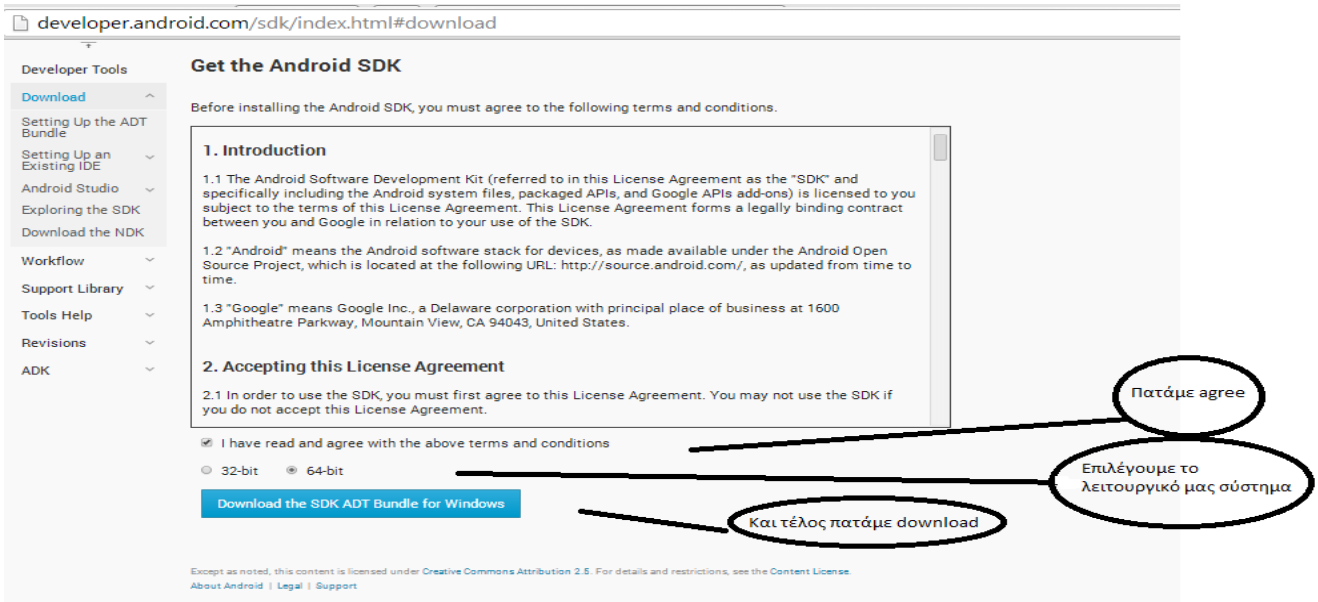
Το επόμενο βήμα αφού έχουμε κατεβάσει το αρχείο είναι η εγκατάσταση των εργαλείων JDK. Πατάμε accept και στην συνέχεια επιλέγουμε το λειτουργικό σύστημα.(Σημείωση τα x86 είναι για τα 32bit). Στην εγκατάσταση δεν χρειάζεται να αλλάξουμε κάτι οπότε πατάμε σε όλα Next.

Εικόνα 10: Εγκατάσταση JDK

Έχοντας κατεβάσει το Java JDK στην συνέχεια κατεβάζουμε το ADT Bundle το οποίο είναι ένα πακέτο όπου περιέχει το πρόγραμμα Eclipse καθώς και τα εργαλεία ανάπτυξης λογισμικού JDK που θα μας χρειαστούν προκειμένου να δημιουργήσουμε την εφαρμογή μας.

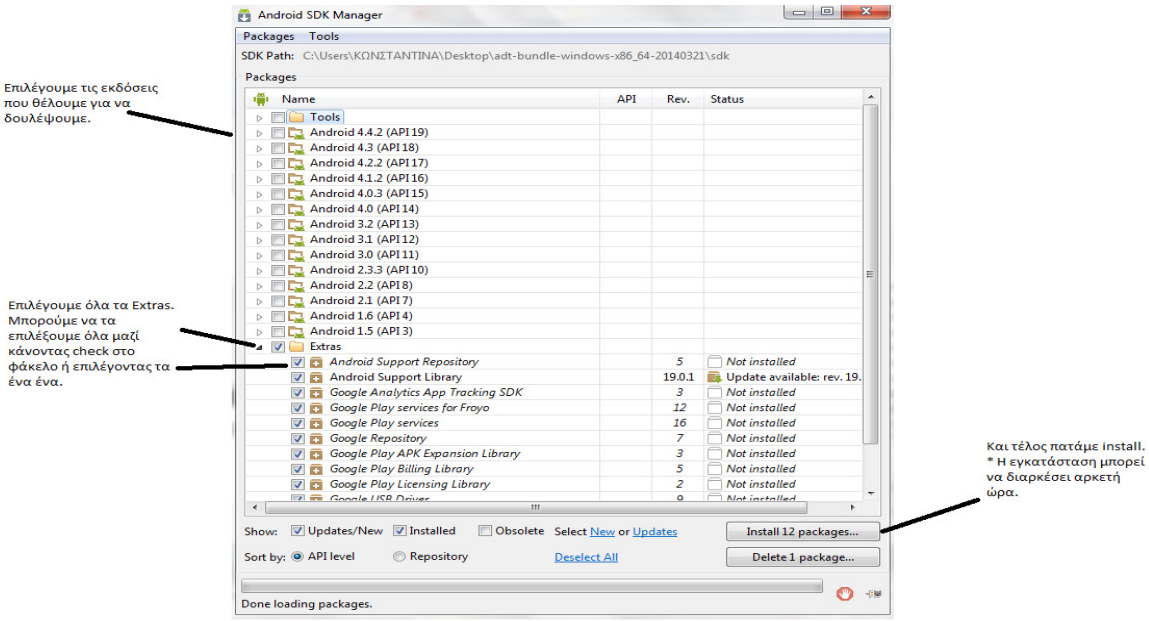


Εικόνα 11: Κατεβάζοντας το ADT Bundle



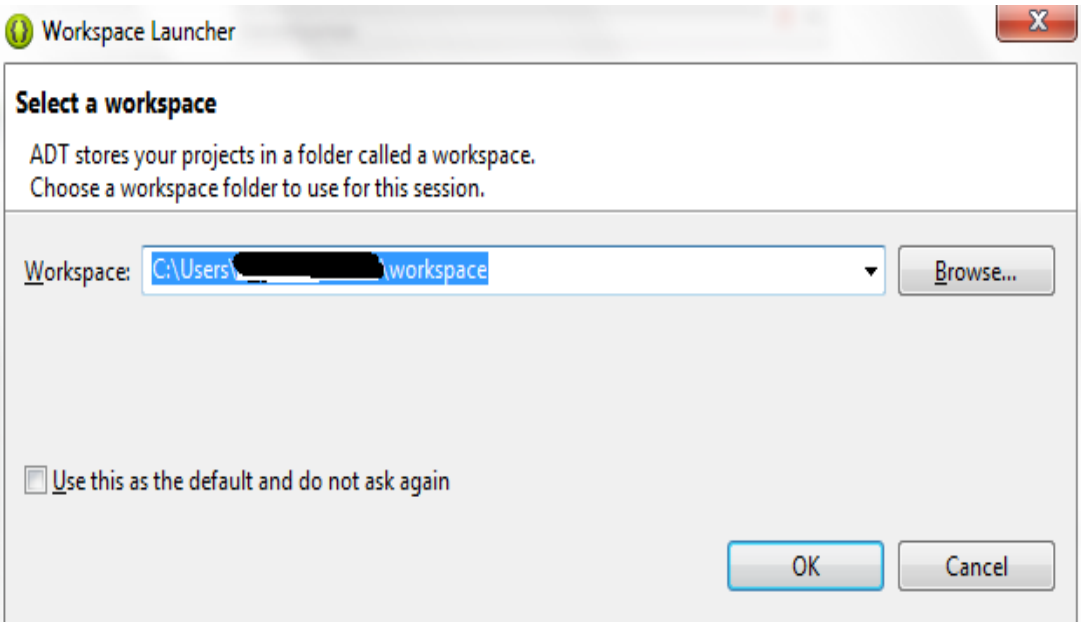
Εικόνα 12: Εγκατάσταση ADT Bundle

Αφού ολοκληρωθεί το κατέβασμα του ADT Bundle στην συνέχεια το αποσυμπιέζουμε στον φάκελο που επιθυμούμε. Έπειτα πηγαίνουμε στο φάκελο στον οποίο έχουμε αποσυμπιέσει τα αρχεία και ανοίγουμε το SDK Manager προκειμένου να κάνουμε εγκατάσταση την έκδοση Android 4.1.2. η οποία και είναι η έκδοση που θα τρέχει η εφαρμογή μας.



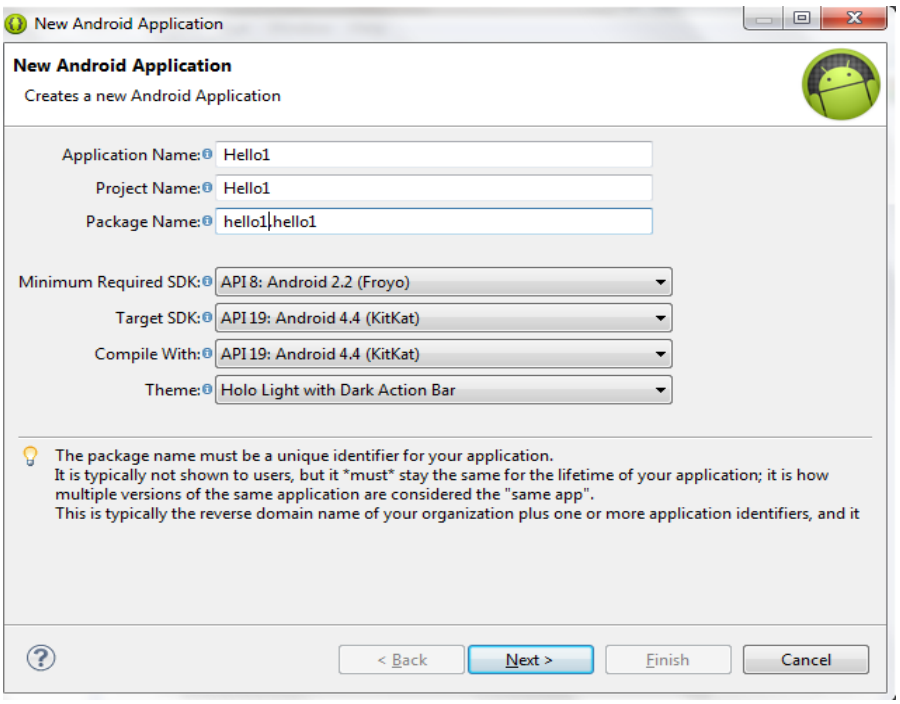
Εικόνα 13: Εγκατάσταση της έκδοσης Android 4.1.2.

Τελειώνοντας και με την εγκατάσταση της εκδοχής Android 4.1.2. πηγαίνουμε και ανοίγουμε το Eclipse. Με το που ανοίξει το πρόγραμμα μας ρωτάει να ορίσουμε ένα WorkSpace στο οποίο θα αποθηκεύονται οι εργασίες που θα κάνουμε.

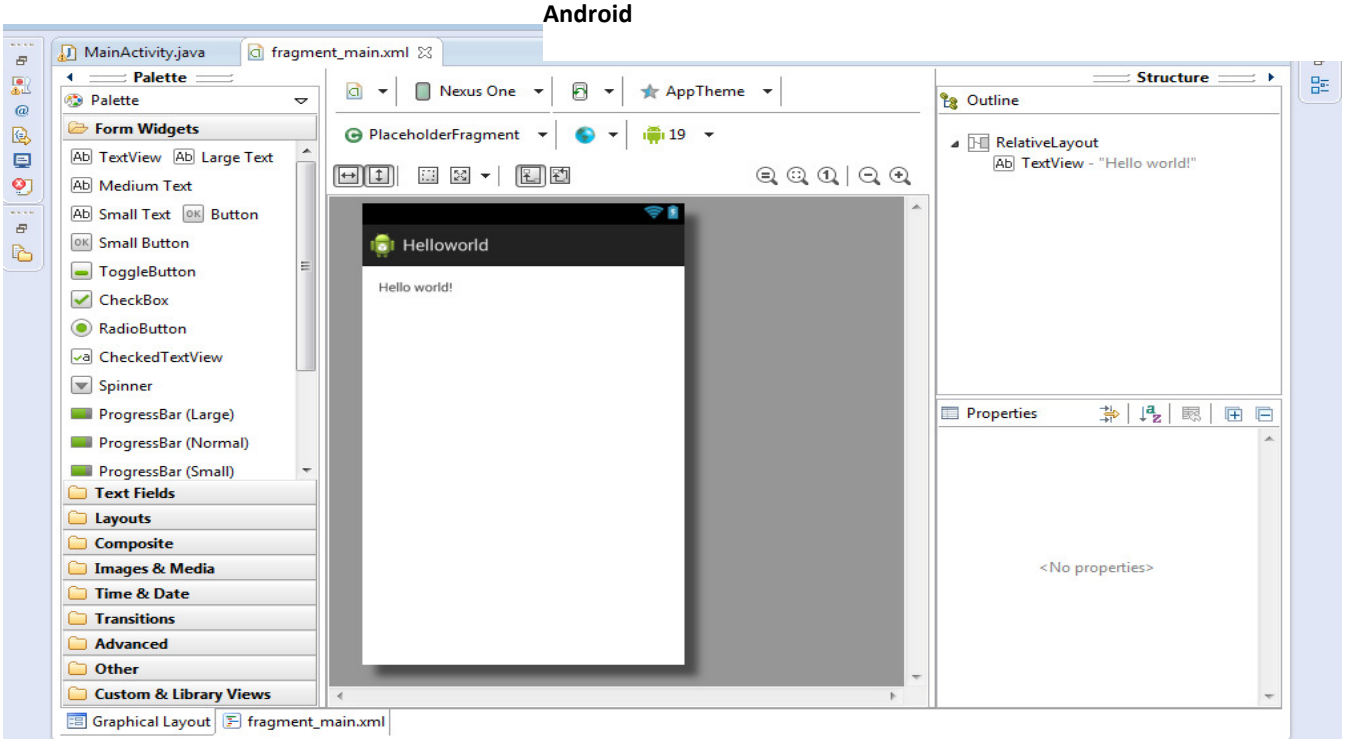


Εικόνα 14: Ορισμός workspace

Τέλος αφού έχουμε ορίσει και το workspace μπορούμε να φτιάξουμε μια απλή εφαρμογή Hello World. Για να γίνει αυτό πηγαίνουμε στην καρτέλα File→New→Android Application Project. Βάζουμε το όνομα και την έκδοση που επιθυμούμε και πετάμε Next και Finish. Στην συνέχει θα μας βγάλει στην αρχική οθόνη όπου θα μας δείχνει πως θα φαίνεται το πρόγραμμα στην συσκευή μας. Για να το τρέξουμε πατάμε  στην γραμμή εργαλείων.



Εικόνα 15: Επιλογή έκδοσης και ονόματος για την δημιουργία εφαρμογής Android



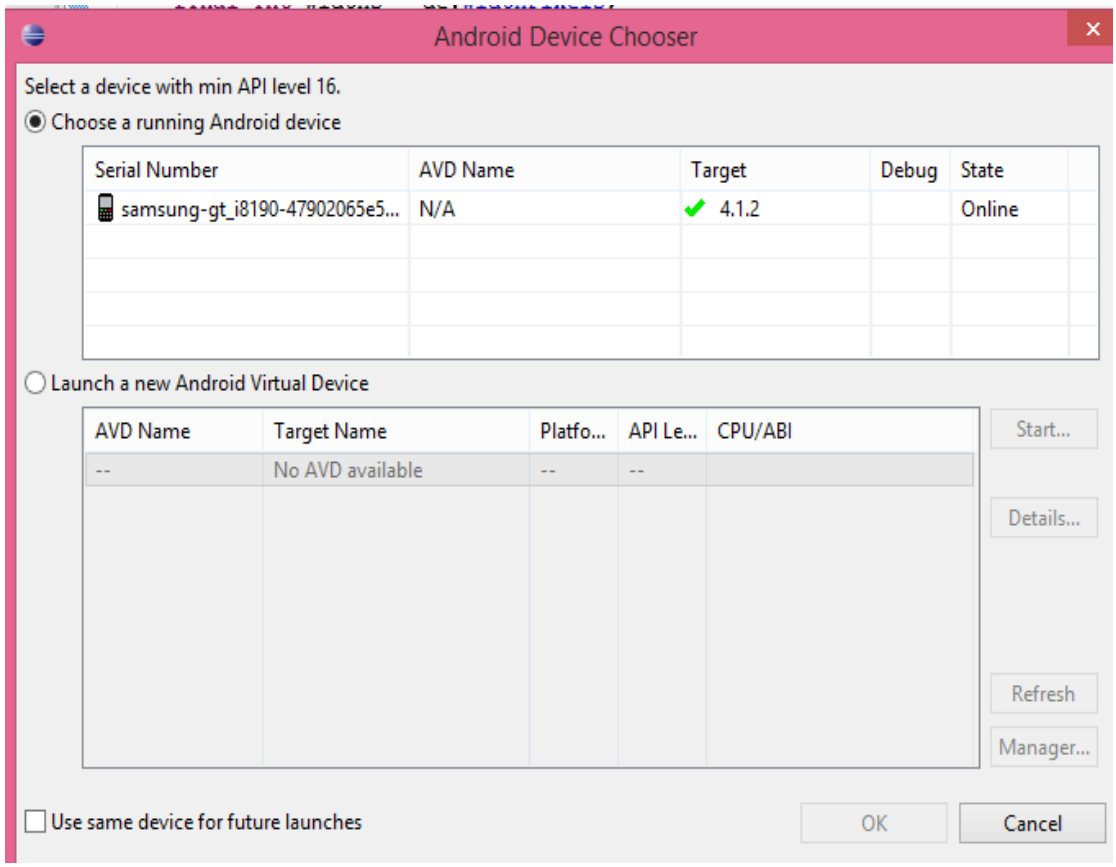
Εικόνα 16: Γραφική απεικόνιση του πρότζεκτ Hello World

2.2. Κατέβασμα του Usb Driver για να τρέχει η εφαρμογή κατευθείαν στο κινητό.

Για να καταφέρουμε να βλέπουμε αυτά που κάνουμε στο πρόγραμμα eclipse κατευθείαν στο κινητό μας χρειάστηκε να κατεβάσουμε το Samsung Android USB Driver για την συσκευή μας η οποία είναι το Samsung Galaxy S3 mini.

Εικόνα 17: Link για να κατεβάσουμε το Samsung Usb Driver For Mobile Phones

Όταν τελειώσει η εγκατάσταση πηγαίνουμε στο eclipse και όταν πατήσουμε run μας βγάζει στις συσκευές την συσκεύη μας.



Εικόνα 18: Απεικόνιση της συσκευής μας

ΚΕΦΑΛΑΙΟ 3: Δημιουργία Της Εφαρμογής Seagame

3.1. Απαιτήσεις εφαρμογής Seagame

Για την εφαρμογή αυτή θα χρειαστεί να δημιουργήσουμε τέσσερις επιφάνειες. Αυτές είναι:

1. Το Main Menu. Το Main menu αποτελείται απο δύο κουμπιά, το Start Game και τον τίτλο του παιχνιδιού τα οποία πλαισιώνονται από ένα background.

2. Το Pause Menu(μενού παύσης). Το Pause Menu περιλαμβάνει κι αυτό δύο κουμπιά. Αυτά είναι το Continue και Main Menu τα οποία πλαισιώνονται επίσης από ένα background.

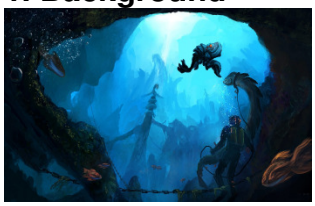
3. Τα Μενού WIN/LOSE το οποίο αποτελείται από ένα background, ένα Header Text αλλά και ένα Button το οποίο οδηγεί στο αρχικό μενού.

4. Τέλος, η οθόνη του παιχνιδιού περιλαμβάνει ένα background, ένα πινακάκι για το score, ένα κουμπί παύσης αλλά και τέσσερις κλάσεις προκειμένου να δημιουργηθεί το game-play. Οι κλάσεις αυτές είναι, Ship Class, Background Class, Barrier Class και Bonus Class.

Επίσης για την εφαρμογή θα χρειαστούμε και κάποια resources(πηγές). Οι πηγές αυτές είναι:

Για το βασικό μενού:

1. Background



Εικόνα 19: Φόντο αρχικής οθόνης

2. Τίτλος

Εικόνα 20: Τίτλος του παιχνιδιού

3. Start Button

Εικόνα 21: Το κουμπί εκκίνησης

Για τα μενού Pause,Win/Lose:

Δύο κουμπιά. Ένα για την παύση και ένα για την επιστροφή στο αρχικό μενού.

Εικόνα 22: Κουμπιά για τα μενού παύσης,νίκης και ήττας

Για το βασικό παιχνίδι:

1. Ship

Εικόνα 23: Το εικονίδιο για τον παίχτη

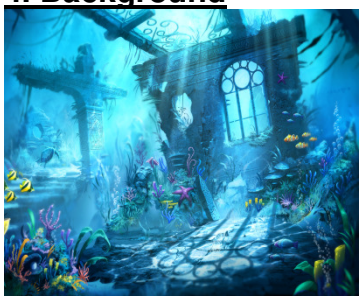
2. Bonus

Εικόνα 24: Εικονίδιο Θησαυρού

3. Obstacle

Εικόνα 25: Το εμπόδιο

4. Background



Εικόνα 26: Το φόντο της οθόνης παιχνιδιού

5. Pause Button

Εικόνα 27: Το κουμπί παύσης

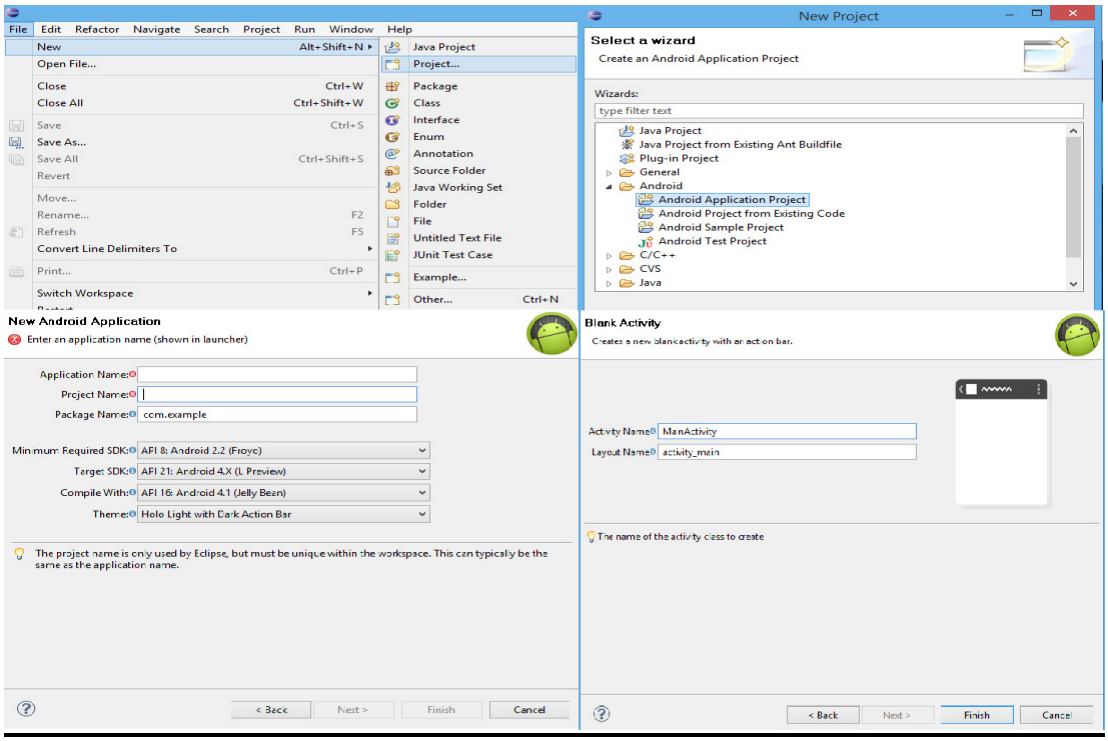
Στις πηγές(res) έχει προστεθεί μουσική, η οποία είναι ξεχωριστή για κάθε μενού. Συγκεκριμένα, η εφαρμογή περιλαμβάνει μία μουσική για το αρχικό μενού και μία για το βασικό παιχνίδι. Επομένως, ακούγεται έναν χαρακτηριστικός ήχος κάθε στιγμή που κερδίζεται ένας θησαυρός και ένας όταν ο παίκτης χάνει και κατ' επέκταση το Ship καταστρέφεται. Τέλος, έχει τοποθετηθεί άλλη μία διαφορετική, αυτή την φορά, μουσική όταν ο στόχος επιτυγχάνεται και ο παίκτης νικά.

3.1.2.Δημιουργία των φακέλων nodpi και raw

Η χρήση όλων αυτών των δεδομένων απαίτησε την δημιουργία δύο φακέλων μέσα στον φάκελο res.
Ο φάκελος res, προερχόμενος απο το περιβάλλον του eclipse, μας δίνει την δυνατότητα να χρησιμοποιούμε εικόνες για διάφορες συσκευές με την χρήση dpi(dots per inch) από προεπιλογή και διαχωρίζεται σε φακέλους.
Η δημιουργία των φακέλων είναι εξαιρετικά εύκολη. Πατώντας δεξί κλικ πάνω στον φάκελο res-->New-->Folder.
Κατ' επέκταση, το πρώτο βήμα είναι η δημιουργία του φακέλου nodpi μέσα στο φάκελο res. Με αυτό τον τρόπο, οι εικόνες που χρησιμοποιούνται αγνοούν το επίπεδο dpi των συσκευών που θα τρέξουν την εφαρμογή. Αξίζει να σημειωθεί στο σημείο αυτό, ότι απαιτείται επίσης η δημιουργία ενός δεύτερου φακέλου. Ο δεύτερος αυτός φάκελος, ονομάζεται raw και δεν υπάρχει στον πρόγραμμα ωστόσο χρειάζεται, στην περίπτωση που θέλουμε να προσθέσουμε μουσική.

3.2. Δημιουργία νέου Project

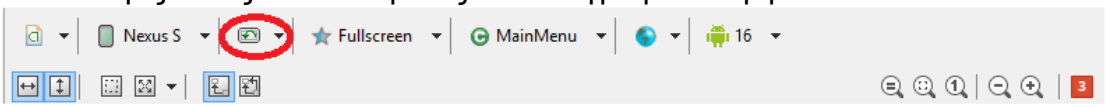
Για να δημιουργήσουμε ένα νέο android project πηγαίνουμε στην καρτέλα File --> New --> Project --> Android Application Project. Στην συνέχεια, ανοίγεται ένα νέο παράθυρο το οποίο μας ζητάει να δώσουμε το όνομα της εφαρμογής, του project καθώς και του Package.
Ακόμη μας δίνεται η δυνατότητα να επιλέξουμε και την έκδοση στην οποία θα δημιουργήσουμε την εφαρμογή μας. Πρέπει να λέχθει ότι στην παρούσα εφαρμογή επιλέχθηκε η έκδοση 4.1.2 η οποία είναι συμβατή με την προσωπική μου συσκευή.
Στα επόμενα μενού, αφού πατήσουμε συνέχεια υπάρχει η δυνατότητα να επιλέξουμε και εικονίδιο για την εφαρμογή μας εφόσον το θέλουμε αλλά και το Workspace.
Στην τελευταία καρτέλα που θα εμφανιστεί πρέπει να δώσουμε ένα συγκεκριμένο όνομα στην activity και στο layout της activity. Τέλος, πατώντας το κουμπί finish είμαστε έτοιμοι για να ξεκινήσουμε.



Εικόνα 28: Δημιουργία android project

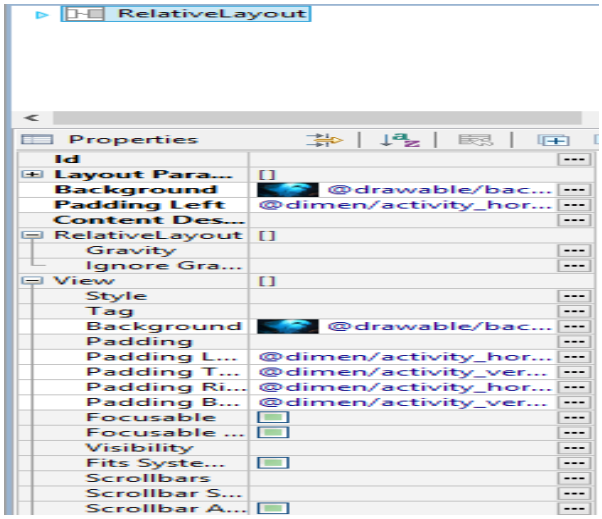
3.2.1 Δημιουργία Main Menu Layout

Στο σημείο αυτό έχει ήδη δημιουργηθεί η πρώτη activity "Main Menu" η οποία θα είναι και η αρχική οθόνη της εφαρμογής. Προχωρώντας στο επόμενο βήμα, το οποίο είναι η δημιουργία του περιβάλλοντος, συναντάμε δύο επιλογές. Πιο αναλυτικά, υπάρχει η δυνατότητα γραφικής προσέγγισης και αυτής με κείμενο XML. Αξίζει να σημειωθεί ότι το βασικό μενού της παρούσας εφαρμογής δημιουργήθηκε με την επιλογή της γραφικής προσέγγισης.
Συνεχίζοντας, αναφέρουμε ότι η συγκεκριμένη εφαρμογή επιλέχτηκε να λειτουργεί σε Landscape mode. Ως εκ τούτου, το βασικό βήμα στο στάδιο αυτό είναι η αλλαγή όψης από τη επιλογή Portrait στην επιλογή Landscape. Οι επιλογές αυτές είναι δοσμένες από το γραφικό περιβάλλον.

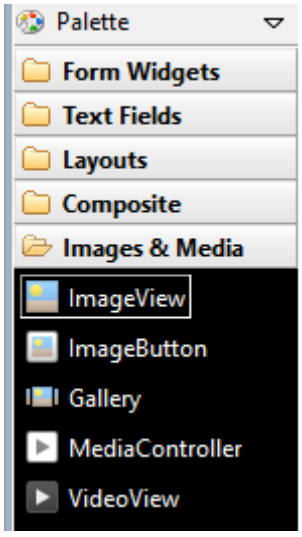


Εικόνα 29: Ορισμός οριζόντιας προβολής

Το επόμενο βήμα ήταν η δημιουργία background. Η επίτευξη αυτού του βήματος πραγματοποιείται με έναν συγκεκριμένο τρόπο. Αρχικά, μέσα στο γραφικό περιβάλλον τοποθετείται ένα relative layout. Στις ιδιότητες αυτού συναντάται η επιλογή προσθήκης background. Εκει επιλέχθηκε απευθείας η εικόνα που είχε αποθηκευτεί νωρίτερα στον φάκελο nodpi, ο οποίος έχει ήδη κατασκευαστεί σε προγενέστερο επίπεδο.
Στην συνέχεια τοποθετήθηκε μέσα στο relative layout ένα image view το οποίο περιείχε τον τίτλο της εφαρμογής.



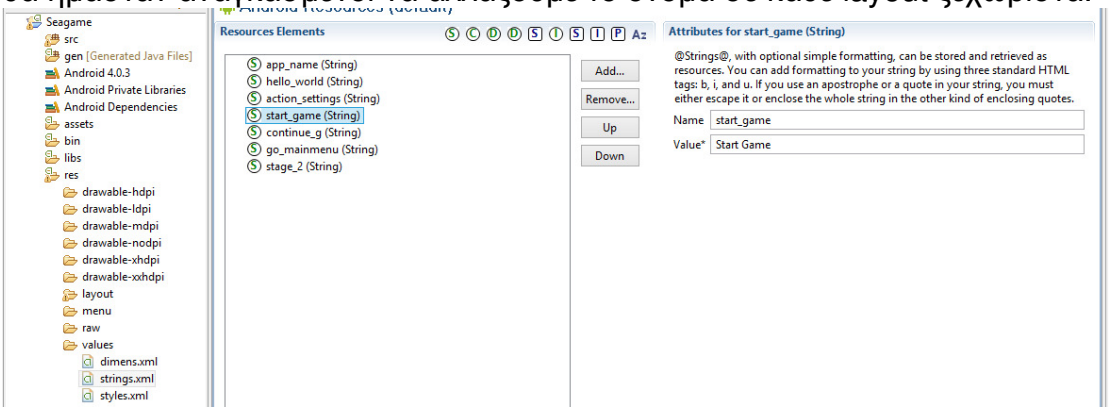
Εικόνα 30: Ιδιότητες του Relative Layout



Εικόνα 31: Εργαλεία της γραφικής προσέγγισης του XML

Στην συνέχεια χρειάζεται να προσθέσουμε στην αρχική οθόνη το κουμπί Start. Για τον σκοπό λοιπόν αυτό απαιτείται η τοποθέτηση ενός ακόμη relative layout. Μέσα στο relative layout θα προστεθούν, ένα image view όπου θα έχει την εικόνα btn1 από τον φάκελο res αλλά και ένα medium text όπου θα έχει τον τίτλο του κουμπιού. Επιπροσθέτως, χρειάζεται να δώσουμε όνομα σε ένα text. Στο σημείο αυτό υπάρχουν δύο επιλογές. Αρχικά έχουμε την δυνατότητα να πάμε στα properties και στο πεδίο text να δώσουμε το όνομα που θέλουμε. Η δεύτερη επιλογή είναι να πάμε στο φάκελο values του eclipse και βρίσκοντας το αρχείο strings.xml να δημιουργήσουμε μία μεταβλητή string στην οποία θα δώσουμε το όνομα που επιθυμούμε.

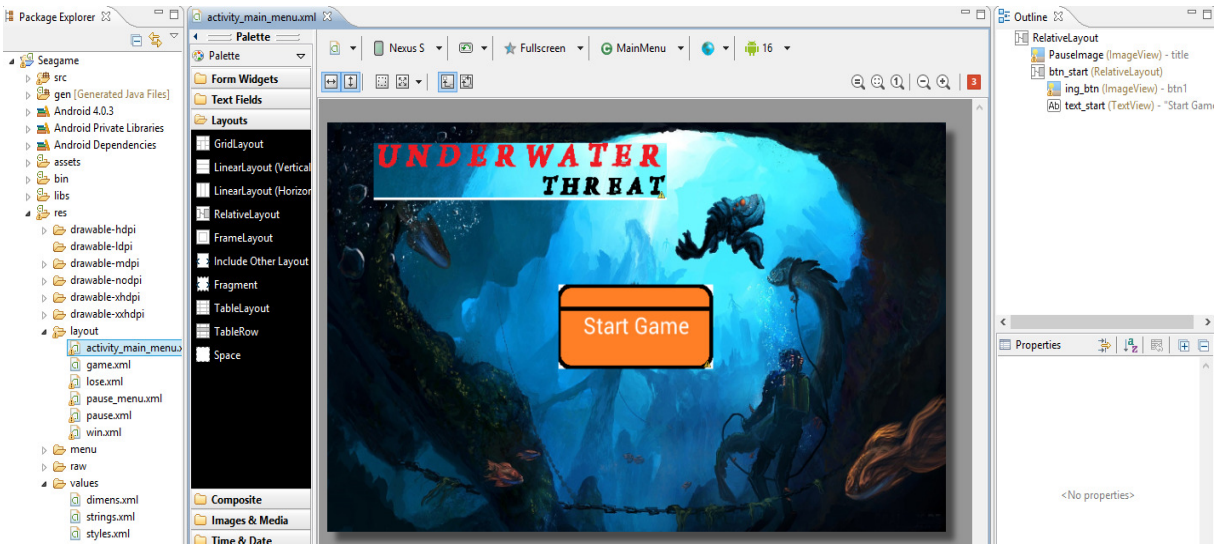
Αξίζει βέβαια στο σημείο αυτό να αναφέρουμε ότι ο δεύτερος τρόπος προτιμάται. Για παράδειγμα ας υποθέσουμε πως αυτό το text θέλουμε να το χρησιμοποιήσουμε και σε άλλα layout κατά την διάρκεια του προγράμματος. Στην περίπτωση που επιθυμήσουμε να του αλλάξουμε το όνομα τότε έχουμε την δυνατότητα να πάμε στο αρχείο, να αλλάξουμε το όνομα στην μεταβλητή stings.xml και κατ' επέκταση να αλλάξει το όνομα σε όλα τα layout. Αυτή η δυνατότητα μας γλιτώνει από χρόνο και κόπο γιατί σε διαφορετική περίπτωση θα ήμασταν αναγκασμένοι να αλλάξουμε το όνομα σε κάθε layout ξεχωριστά.



Εικόνα 32: Δημιουργία String μεταβλητών

Συνεχίζοντας αναφέρουμε ξανά ότι η αρχική μας οθόνη έχει φτιαχτεί με την χρήση της γραφικής προσέγγισης. Ως εκ τούτου τα υπάρχοντα στοιχεία μπορούμε να τα τοποθετήσουμε στην θέση που επιθυμούμε χειροκίνητα και το eclipse να φτιάξει από μόνο του το XML αρχείο.
Τέλος, για να ολοκληρωθεί η αρχική οθόνη πρέπει να δοθούν identifiers. Τα identifiers βοηθάνε έτσι ώστε να μπορούμε στην συνέχεια να βρίσκουμε εύκολα τα στοιχεία μας. Κατ επέκταση για να προσθέσουμε τα identifiers πηγαίνουμε στα properties του κάθε στοιχείου στην καρτέλα Id. Εκεί, δίνουμε το αναγνωριστικό όνομα της επιλογής μας. Οπότε δόθηκε στο δεύτερο layout που προστέθηκε το Id "btn_start", στο image view στο ίδιο Layout το Id "ing_btn" αλλά και στο text το Id "text_start" και μ αυτόν τον τρόπο οληκληρώθηκε η αρχική οθόνη.

Εικόνα 33: Αρχική οθόνη του παιχνιδιού



3.2.2. Δημιουργία της Main Activity

Στο σημείο αυτό πρέπει να δημιουργηθεί η Main Activity. Έτσι, πηγαίνουμε στον φάκελο "src" του Eclipse επεκτείνουμε το package και ανοίγουμε το αρχείο MainMenu.java.
Εκεί αρχικά θα δηλωθούν οι μεταβλητές. Η μεταβλητή RelativeLayout δηλώνεται ως "Btn" , η ImageView ως "ImageButton" και η TextView ως "txt".

```
public class MainMenu extends Activity {

        RelativeLayout Btn;
        ImageView ImageButton;
        TextView txt;
```

Εικόνα 34: Δήλωση μεταβλητών Btn, Image Button και txt

Στη συνέχεια, με την βοήθεια της μέθοδου OnCreate ενώνουμε τις μεταβλητές της Activity με τα στοιχεία που είχαμε προσθέσει πριν στο αρχείο XML, το οποίο έχει δημιουργηθεί βαση της γραφικής προσέγγισης. Επιπλέον, με την βοήθεια της μεθόδου findViewById βρίσκουμε τα στοιχεία που είχαν δηλωθεί πριν στο XML και έχουν το καθένα το δικό τους id.

```
protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main_menu);


      Btn = (RelativeLayout) findViewById(R.id.btn_start);
      ImageButton = (ImageView)findViewById(R.id.img_btn);
      txt = (TextView)findViewById(R.id.text_start);
```

'Επειτα θα χρησιμοποιήσουμε τους Listeners, στους οποίους περιγράφουμε τι θέλουμε να κάνει το πρόγραμμα στα συγκεκριμένα events. Στο παρών πρόγραμμα χρησιμοποιήθηκαν οι "OnTouch" και "OnClick" Listeners.
Με τον OnTouchListener αγγίζοντας το κουμπί "Start Game" βλέπουμε ότι αλλάζει χρώμα και όταν σταματάμε να το αγγίζουμε αυτό επανέρχεται στην αρχική του κατάσταση.

```
Btn.setOnTouchListener(new
TochButton(ImageButton));
```

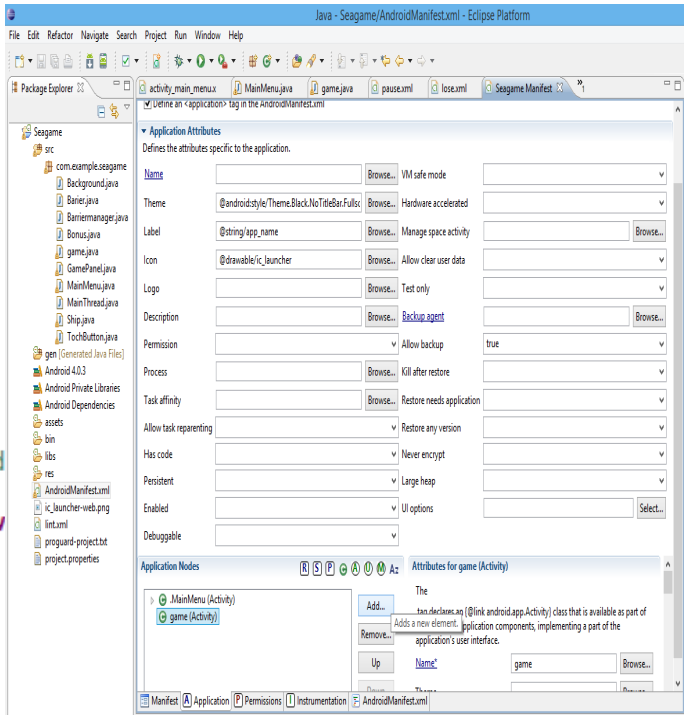
Εικόνα 35: OnTouchListener

Με τον OnClickListener πατώντας το "Start Game" το πρόγραμμα, μας οδηγεί στο βασικό παιχνίδι.Βέβαια, για να γίνει αυτό, πρέπει πρώτα να χρησιμοποιήσουμε την λειτουργία "intent" . Με αυτή την λειτουργία θα μπορέσουμε να ενώσουμε τις δυο κλάσεις και ύστερα να δηλώσουμε στο manifest ότι υπάρχει μία ακόμα Activity στο MainMenu. Η δήλωση στο

manifest ακολουθεί συγκεκριμένα βήματα. Αρχικά ανοίγουμε το αρχείο manifest. Στην συνέχεια πηγαίνουμε στην καρτέλα Application, πατάμε Add και επιλέγουμε την κλάση.

```
Btn.setOnClickListener(new OnClickListener() {

        @Override
        public void onClick(View v) {
                // TODO Auto-generated method
stub
                Intent myIntent = new
Intent(MainMenu.this, game.class);
                startActivity(myIntent);
```

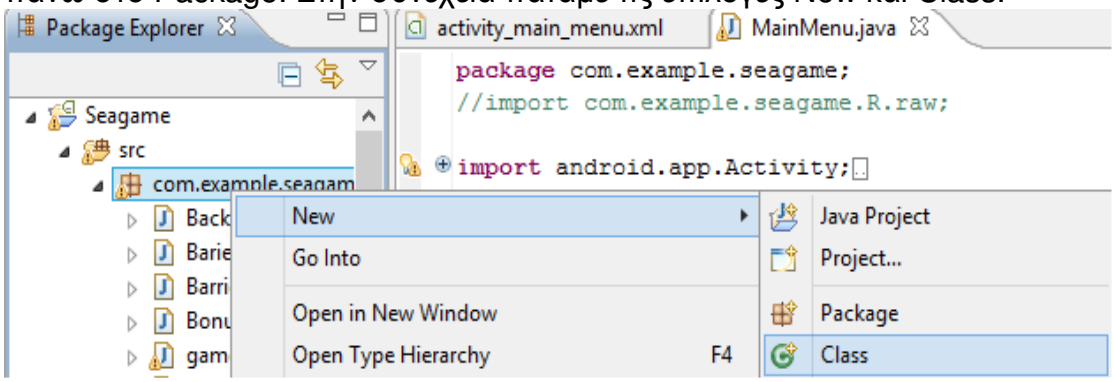
Εικόνα 37: OnClickListener



Εικόνα 36: Προσθήκη κλάσης στο Manifest

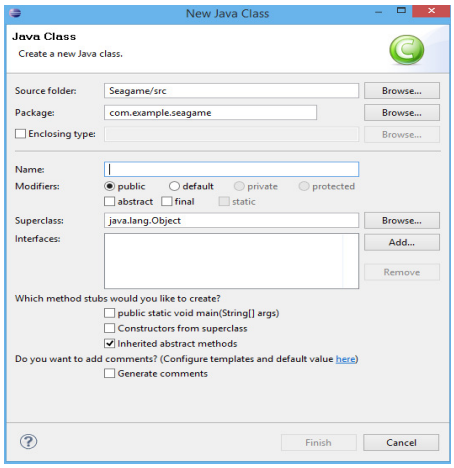
3.2.3. Δημιουργία της κλάσης και του XML Game

Σε αυτό το σημείο πρέπει να δημιουργηθεί μία νέα κλάση. Για την δημιουργία νέας κλάσης λοιπόν, πηγαίνουμε στην φάκελο src και έπειτα πατάμε δεξί κλικ πάνω στο Package. Στην συνέχεια πατάμε τις επιλογές New και Class.



Εικόνα 38: Δημιουργία νέας κλάσης

Αμέσως μετά, θα ανοίξει μία καρτέλα στην οποία το πρόγραμμα μας δίνει κάποιες επιλογές. Στο συγκεκριμένο σημείο δόθηκε μόνο το όνομα της κλάσης την οποία ονομάσαμε "game". Η κλάση η οποία δημιουργήθηκε δεν είχε από μόνη της, την Oncreate μέθοδο. Γι αυτό τον λόγο την δημιουργήσαμε μόνοι μας. Ακολούθεισαι επίσης και η δημιουργία ενός αρχείου XML. Η ενέργεια αυτή είναι αρκετά σημαντική, διότι το αρχείο XML θα συνδεθεί στην συνέχεια με την



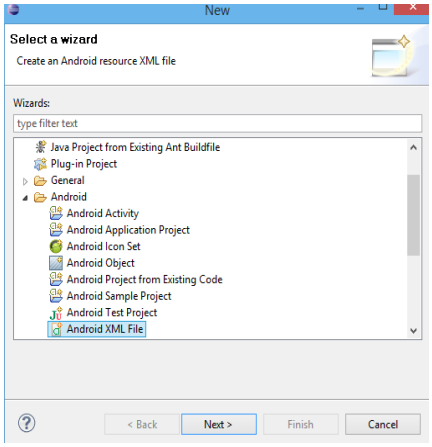
Εικόνα 39: Καρτέλα στην δημιουργία νέας κλάσης

OnCreate έτσι ώστε να δημιουργηθεί το περιβάλλον στο οποίο θα τρέχει η εφαρμογή μας.

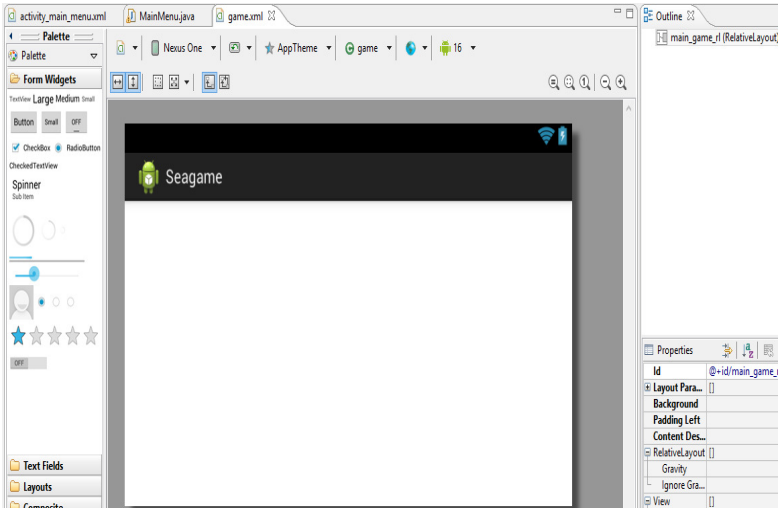
```
public class game extends Activity {

        RelativeLayout Rel_main_game;

protected void onCreate(Bundle savedInstanceState) {

                super.onCreate(savedInstanceState);

                setContentView(R.layout.game);

        Rel_main_game = (RelativeLayout)
```

Εικόνα 40: Δημιουργία μεθόδου OnCreate

Αξίζει λοιπόν στο σημείο αυτό να λεχθεί ο τρόπος με τον οποίο θα δημιουργήσουμε το αρχείο XML που αναφέραμε νωρίτερα. Πιο αναλυτικά, πηγαίνουμε στο φάκελο ‘’res’’, πατάμε δεξί κλικ στον φάκελο layout, επιλέγουμε την επιλογή ‘’New’’ και έπειτα ‘’othera . Στην καρτέλα που ανοίγει επιλέγουμε Android XML File. Αφού το επιλέξουμε μας δίνεται η δυνατότητα να διαλέξουμε ένα "Root Element" και να του δώσουμε ένα όνομα. Το όνομα που δώσαμε είναι "game" και το στοιχείο που επιλέξαμε να έχει από προεπιλογή ήταν το



Εικόνα 41: Δημιουργία XML αρχείου

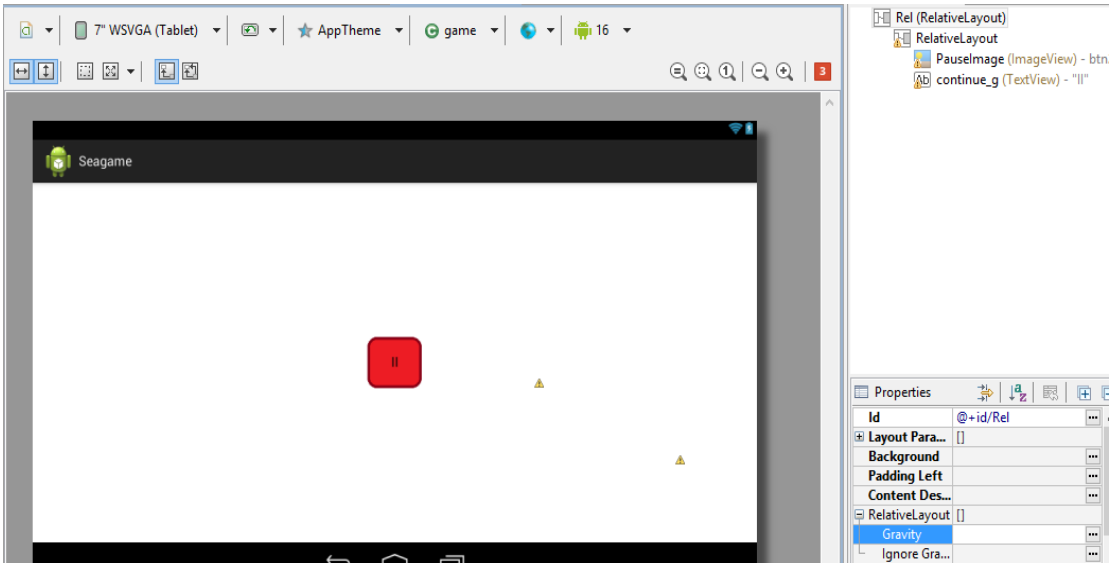


Εικόνα 42: Το game XML

Relative Layout. Έτσι, αφού δημιουργήσαμε το XML πηγαίνουμε και του δίνουμε ως Id το "main_game_rl". Είναι πολύ σημαντικό να σημειωθεί ότι τα features της δυναμικής δημιουργίας στοιχείων δεν υπάρχουν γραμμένα σε XML. Γι αυτόν τον λόγο χρειάζεται να τα προσθέσουμε κατά την διαδικασία δημιουργίας της Activity.

3.2.4. Δημιουργία του XML Pause

Γνωρίζοντας πλέον πως να φτιάξουμε ένα αρχείο XML δημιουργούμε το "Pause.xml" βάζοντας του δύο relative layout. Το πρώτο εκ των δύο θα περιλαμβάνει ως Id το "Rel", ένα ImageView το οποίο θα έχει την εικόνα btn2 αλλα και ένα text με όνομα "II".



Εικόνα 43: Το XML του κουμπιού παύσης

Το κουμπί Pause δεν θα χρησιμοποιηθεί ως Layer αλλά μόνο ως κουμπί.

3.2.5. Δήλωση και επεξεργασία του Pause Button

Το Pause Button είναι το κουμπί το οποίο όταν θα το πατάμε θα σταματάει η εφαρμογή.
Όπως ήδη γνωρίζουμε οτιδήποτε έχουμε δημιουργήσει σε XML πρέπει να το δηλώσουμε έτσι ώστε να μπορεί να τρέξει στο πρόγραμμα. Για τον λόγο αυτό αρχικά πηγαίνουμε στην κλάση "game.java" όπου και δηλώνουμε το Pause Button το οποίο έχουμε δημιουργήσει σε XML και στην συνέχεια με την μέθοδο LayoutInflater η οποία είναι μία μέθοδος η οποία μας επιτρέπει να δημιουργήσουμε και να επεξεργαστούμε μια εικόνα σε XML, ορίσαμε το

μέγεθος της εικόνας του "pause.xml" καθώς και την στην οποία θα βρίσκεται μέσα στο κεντρικό παιχνίδι.

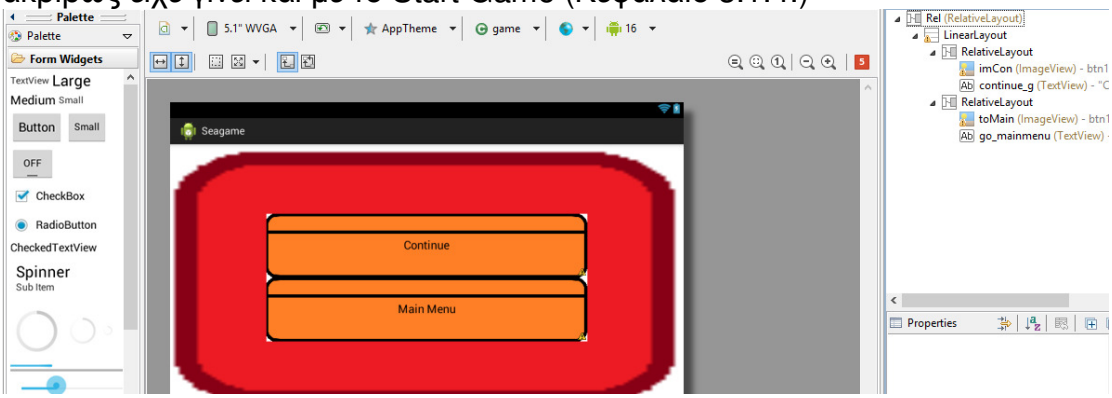
```
public class game extends Activity {

        View pauseButton;

protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);



        setContentView(R.layout.game);

        Rel_main_game = (RelativeLayout)
        DisplayMetrics de = new DisplayMetrics();
        this.getWindowManager().getDefaultDisplay().getMetrics(de);

        final int heights = de.heightPixels;
        final int widths = de.widthPixels;

 LayoutInflater myInflater = (LayoutInflater)
getApplicationContext().getSystemService(getApplicationContext().LAYOUT_INFLATER_SE
RVICE);
        pauseButton = myInflater.inflate(R.layout.pause, null, false);
        pauseButton.setX(widths-150);
        pauseButton.setY(0);
        Rel_main_game.addView(pauseButton);

        pauseButton.getLayoutParams().height=150;
        pauseButton.getLayoutParams().width=150;
```

Εικόνα 44: Δήλωση και επεξεργασία του κουμπιού παύσης

3.2.6. Δημιουργία του XML Pause Menu και δήλωση στο game.java

Μετά και την δημιουργία του Pause Button δημιουργούμε το Pause Menu όπου ο σκοπός του είναι να εμφανίζεται όταν θα πατάμε το Pause Button και θα μας δίνει την επιλογή να συνεχίσουμε ή να γυρίσουμε στο αρχικό Menu.
Με τον ίδιο τρόπο που δημιουργήσαμε το game.xml αλλά και το pause.xml δημιουργήθηκε επίσης και το pausemenu.xml. Στο pausemenu.xml υπάρχει ένα relative layout με Id "Rel" και ένα Linear layout. Επιπροσθέτως, μέσα στο Linear layout υπάρχουν ακόμη δύο relative layout τα οποία έχουν μέσα τους από ένα imageview το καθένα και δύο text. Το ένα imageview βρίσκεται με Id "ImCon", το άλλο με Id "ToMain". Επιπλέον υπάρχουν όπως αναφέραμε

προηγουμένως και δύο text, στα οποία η ονομασία έχει μπει με string όπως ακριβώς είχε γίνει και με το Start Game (Κεφάλαιο 3.1.4.)



Εικόνα 45: Το μενού παύσης

Αφού ολοκληρώθηκε το XML το επόμενο βήμα είναι η δήλωση και η επεξεργασία του PauseMenu στο game.java. Στην επεξεργασία δηλώνουμε το PauseMenu ως αόρατο καθώς θέλουμε μόνο να το βλέπουμε όταν πατάμε το pause button και όχι συνέχεια.

```
public class game extends Activity {

        View PauseMenu;
        View pauseButton;
        RelativeLayout Rel_main_game;

protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.game);

        Rel_main_game = (RelativeLayout)
        DisplayMetrics de = new DisplayMetrics();
        this.getWindowManager().getDefaultDisplay().getMetrics(de);

        final int heights = de.heightPixels;
        final int widths = de.widthPixels;

LayoutInflatermyInflater=(LayoutInflater)getApplicationContext().getSystemService(getApplica
tionContext().LAYOUT_INFLATER_SERVICE);

        PauseMenu = myInflater.inflate(R.layout.pause_menu, null,     false);
        Rel_main_game.addView(PauseMenu);
        PauseMenu.setVisibility(View.GONE);

ImageView Con = (ImageView)PauseMenu.findViewById(R.id.imCon);
ImageView MainMenuTo =(ImageView)PauseMenu.findViewById(R.id.toMain);
```

Εικόνα 46: Δήλωση και επεξεργασία του μενού παύσης

Στην συνέχεια δημιουργήθηκαν OnClickListeners οι οποίοι θα εκτελούν τις ενέργειες Continue αλλά και Main Menu οι οποίες υπάρχουν στο Pause Menu.

```
OnClickListener Continue_list = new OnClickListener() {
            @Override
            public void onClick(View v) {
                  PauseMenu.setVisibility(View.GONE);



                  pauseButton.setVisibility(View.VISIBLE);
            }
      };
      OnClickListener To_Main_Menu = new OnClickListener() {
            @Override
            public void onClick(View v) {
                  game.this.finish();
            }
      };
```

Εικόνα 47: Δημιουργία OnClickListener

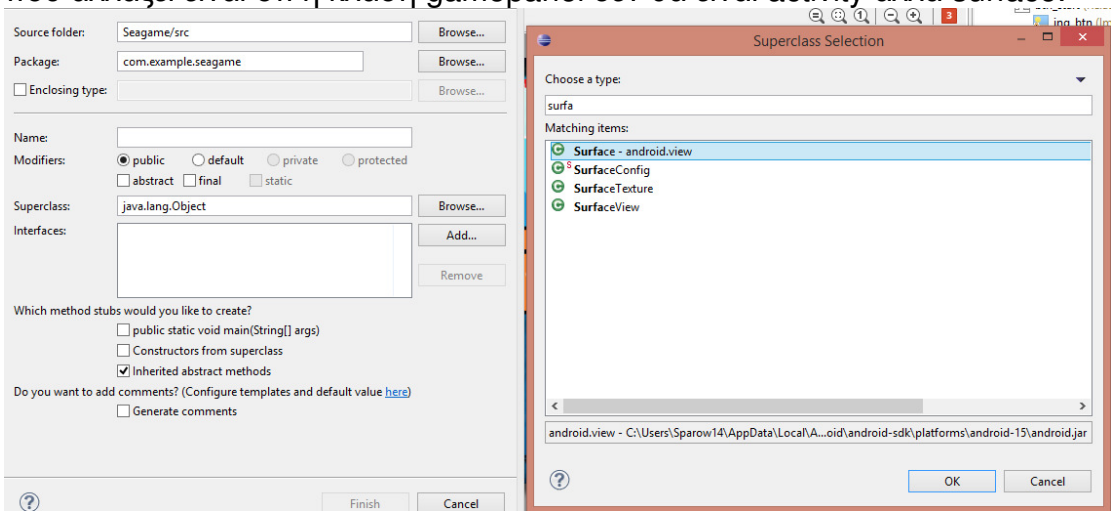
Αφού δημιουργήσουμε αυτούς τους δύο Listeners, δημιουργούμε ακόμη έναν ο οποίος θα εξαφανίζει το Pause Button κάθε φορά που το πατάμε. Ως αποτέλεσμα, θα εμφανίζεται μόνο το Pause Menu πράγμα που χρειάζεται να το δηλώσουμε στην μέθοδο OnCreate στο LayoutInflater.

```
OnClickListener Pause_click = new OnClickListener() {
            @Override
            public void onClick(View v) {
                  pauseButton.setVisibility(View.GONE);
                  PauseMenu.setVisibility(View.VISIBLE);

            }
      };

      protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);

            setContentView(R.layout.game);
      LayoutInflater myInflater = (LayoutInflater)
getApplicationContext().getSystemService(getApplicationContext().LAYO
UT_INFLATER_SERVICE);
      pauseButton = myInflater.inflate(R.layout.pause, null, false);
      pauseButton.setOnClickListener(Pause_click);
```

Εικόνα 48: Δημιουργία του Listener Pause_Click

3.2.7. Δημιουργία της κλάσης gamepanel και MainThread

Ο τρόπος με τον οποίο δημιουργήθηκε η κλάση gamepanel είναι σχεδόν ίδιος με τον τρόπο δημιουργίας της κλάσης game που είδαμε παραπάνω. Το μόνο που αλλάζει είναι ότι η κλάση gamepanel δεν θα είναι activity αλλά surface.



Εικόνα 49: Δημιουργία κλάσης MainThread

Αυτό γίνεται διότι το πεδίο surface μας παρέχει μία περιοχή στην οποία μπορούμε να σχεδιάσουμε τι ακριβώς θα γίνεται στην παρούσα εφαρμογή. Έχουμε λοιπόν την δυνατότητα να απεικονίσουμε όλες τις ενέργειες σε ξεχωριστό thread και κατ' επέκταση να χειριστούμε καλύτερα τα events που σχεδιάζουμε.
Η κλάση gamepanel πρέπει να περιέχει και να εφαρμόζει τις μεθόδους surfaceholder και callback. Ως εκ τούτου καλούμαστε να τις δημιουργήσουμε έτσι ώστε να έχουμε την δυνατότητα να διακόπτουμε τα events. Πριν όμως δημιουργήσουμε αυτές τις μεθόδους δημιουργούμε και έναν constructor.
Ακόμη χρειάστηκε και η δημιουργία της λειτουργίας OnTouchEventHandler, η οποία μας δίνει την δυνατότητα να χρησιμοποιούμε τα events τα οποία θα προστεθούν στο gamepanel απλά κάνοντας κλικ σ αυτά.

```
public boolean onTouchEvent(MotionEvent event) {
        return true;
    }
```

Εικόνα 50: Η λειτουργία OnTouchEvent

Τέλος, δημιουργούμε δύο ακόμη μεθόδους. Την draw μέθοδο η οποία είναι υπεύθυνη για όλα τα σχέδια που θα υπάρχουν στην εφαρμογή, καθώς και την update μέθοδο η οποία είναι υπεύθυνη για την αναβάθμιση του gameplay.
Η κλάση gamepanel είναι το πιο σημαντικό κομμάτι μαζί με την κλάση game καθώς ότι έχουμε δείξει και ότι θα δημιουργήσουμε στην συνέχεια θα υπάρχει μέσα σ αυτές τις δύο κλάσεις. Επίσης οι δύο αυτές κλάσεις περιέχουν όλα τα στοιχεία, τα οποία βλέπουμε όταν θα τρέχει η εφαρμογή.
Συνεχίζοντας αξίζει να αναφέρουμε ότι το gameplay της συγκεκριμένης εφαρμογής είναι μια άπειρη επανάληψη. Για να το πετύχουμε αυτό χρειάστηκε

η δημιουργία μιας άλλης κλάσης, η οποία ονομάστηκε MainThread και περιγράφει αυτή την διαδικασία.
Η MainThread λοιπόν όπως και όλες οι κλάσεις των στοιχείων που υπάρχουν σ αυτή την εφαρμογή θα αντλούνται από την gamepanel, έτσι ώστε να τρέχει η εφαρμογή.
Πιο αναλυτικά η MainThread περιέχει τρείς μεταβλητές.

1. Την SurfaceHolder όπου θα είναι για τον έλεγχο της επιφάνειας.
2. Την gamepanel όπου θα είναι υπεύθυνη για το gamepanel.
3. Την μεταβλητή Running όπου θα μας δείχνει αν το thread τρέχει η όχι.

Στο σημείο αυτό είναι απαραίτητο να αναφέρουμε ένα πλήθος πληροφοριών. Ξεκινώντας τονίζουμε ότι η Main Thread περιέχει πλην των τριών μεταβλητών, την μέθοδο Run η οποία με την σειρά της περιέχει την μέθοδο canvas που χρησιμεύει για τον σχεδιασμό. Παράλληλα, συμπεριλαμβάνει την μεταβλητή Pause Game με την βοήθεια της οποίας η εφαρμογή σταματάει αλλά και την μεταβλητή dt. Η τελευταία, είναι σε θέση να υπολογίσει την αλλαγής θέσης των αντικειμένων που θα υπάρχουν στην οθόνη καθώς και τον χρόνο που πέρασε από το τελευταίο update στο παιχνίδι.Τέλος συμπεριλαμβάνονται οι μέθοδοι Long Start Draw και Long End Draw οι οποίες καταγράφουνε την αρχή και το τέλος του χρόνου της σχεδίασης. Για την μέτρηση του χρόνου, χρησιμοποιήσαμε την μεταβλητή dt η οποία είναι ίση με την διαφορά του StartDraw απο το EndDraw και έχει διαιρεθεί διά 1000 διότι ο χρόνος μετριέται σε υποδιαιρέσεις του δευτερολέπτου. Η συγκεριμένη μεταβλητή έχει χρησιμοποιηθεί σε όλα τα αντικείμενα του παιχνιδιού.

```
import android.graphics.Canvas;
import android.view.SurfaceHolder;
public class MainThread extends Thread {
        private SurfaceHolder surfaceHolder;
        private GamePanel gamePanel;
        private boolean running;
        float dt;
        public MainThread(SurfaceHolder holder, GamePanel gamePanel)
                this.surfaceHolder = holder;
                this.gamePanel = gamePanel;
                dt=0;
        }
        void setRunning(Boolean running){
                this.running = running;
        }
        @Override
        public void run() {
                Canvas canvas;
                while (running) {
                        if(!gamePanel.Pause_game){
                                long StartDraw = System.currentTimeMillis();
                                canvas = null;
                                        try{
                                        canvas =this.surfaceHolder.lockCanvas();


                                                synchronized (surfaceHolder) {
                                                gamePanel.Update(dt);
                                                gamePanel.Draw(canvas);
                                                        }       }
                                        finally{
                                                if (canvas!=null)
                                surfaceHolder.unlockCanvasAndPost(canvas);
                                        }
                                long EndDraw = System.currentTimeMillis();
                                dt = (EndDraw-StartDraw)/1000.f;
                        }       }       }       }
```

Εικόνα 51: Δημιουργία της MainThread

3.2.8. Δημιουργία της κλάσης Background

Σ αυτό το υποκεφάλαιο θα εξηγήσουμε πώς δημιουργήθηκε αλλά και τι κάνει η κλάση Background η οποία θα δηλωθεί στην κλάση gamepanel.
Η δημιουργία της κλάσης Background θα πραγματοποιηθεί με τον ίδιο τρόπο που δημιουργήθηκαν και οι πρηγούμενες μέχρι τώρα κλάσεις. Η κλάση αυτή μας δίνει την δυνατότητα να ορίσουμε το φόντο της εφαρμογής μας.
Ξεκινώντας λοιπόν, δημιουργούμε έναν constructor στον οποίο δίνουμε τα ορίσματα Bitmap και screen_w καθώς επίσης και τις μεταβλητές που θα χρησιμοποιήσουμε στην κλάση. Οι μεταβλητές αυτές είναι το Bitmap, ScreenWidth, τα θέσεις x & y, καθώς και μία μεταβλητή που ορίζει πόσα Background μπορούμε να έχουμε στην οθόνη.
Έπειτα, δημιουργούμε το φόντο και οι αρχικές τιμές των x,y ορίζονται ως μηδέν. Στην συνέχεια, προσθέτουμε το πλάτος της οθόνης μας(width screen) και υπολογίζουμε τον αριθμό των εικόνων. Ο υπολογισμός αυτός πραγματοποιείται, διαιρώντας το πλάτος οθόνης (WidthScreen) με το πλάτος του Bitmap και προσθέτοντας +1 στο σύνολο. Επιπροσθέτως, περνάμε την μεταβλητή gamepanel μέσα στον constructor η οποία θα συνδέει την κλάση Background με την κλάση gamepanel.

```
public class Background {
	Bitmap BackBitmap;
	int x,y;
	int ScreenWidth;
	int Count_Background;
	GamePanel root_gamepanel;

	public Background(Bitmap bitmap , int Screen_w , GamePanel
Game_panel) {
		this.BackBitmap=bitmap;
		this.x=0;
		this.y=0;
		this.ScreenWidth=Screen_w;
		Count_Background = ScreenWidth/BackBitmap.getWidth()+1;
		root_gamepanel = Game_panel;
	}
```

Εικόνα 52: Οι μεταβλητές και ο constructor της κλάσης Background

Αφού ολοκληρώθηκαν οι παραπάνω ενέργειες φτιάχνουμε την μέθοδο Draw η οποία θα περιλαμβάνει τη λειτουργία canvas και την μέθοδο Update. Στην λειτουργία canvas γίνεται ο σχεδιασμός του Background ενώ με την βοήθεια της μέθοδου Update υπολογίζονται οι αλλαγές στο αντικείμενο(ship).
Συνεχίζοντας αναφέρουμε πώς στην λειτουργία draw χρησιμοποιούμε την συνάρτηση for με έναν μετρητή(i) έτσι ώστε να μετρήσουμε από το μηδέν έως και το σύνολο των αριθμών των εικόνων που υπάρχουν στην οθόνη. Σε αυτό προσθέτουμε τις συνθήκες της σχεδίασης καθώς και του canvas εφόσον αυτές δεν είναι ίσες με Null. Στην συνέχεια ξεκινάμε την σχεδίαση. Αρχικά, καθορίζουμε το Bitmap ως αριστερό σημείο σύμφωνα με την μεταβλητή x.

Μετά παίρνουμε το πλάτος του Bitmap και αφού το πολλαπλασιάσουμε με το μετρητή(i), προσθέτουμε την αξία της μεταβλητής x. Η τρίτη παράμετρος είναι η y και η τέταρτη είναι Null. Μετά ελέγχουμε αν η απόλυτη τιμή της x είναι μεγαλύτερη απο το πλάτος του Bitmap και ορίζουμε την μεταβλητή x ίση με το x+το πλάτος του Bitmap. Όλα αυτά τα βήματα δεν γίνονται τυχαία. Ο σκοπός είναι να δημιουργήσουμε την ψευδαίσθηση ότι το Background δεν τελειώνει ποτέ. Το επόμενο βήμα είναι η περιγραφή της μεθόδου update στην οποία η αξία της x ορίζεται ως x μείον την ταχύτητα του αντικειμένου(ship) επί την μεταβλητή dt. Η ταχύτητα του Ship θα οριστεί στην συνέχεια στην κλάση gamepanel.

```
public void draw(Canvas canvas){
            for (int i = 0; i <Count_Background+1;i++){
                  if (canvas!=null)
                        canvas.drawBitmap(BackBitmap,
BackBitmap.getWidth()*i+x,y,null);

            }
            if (Math.abs(x)>BackBitmap.getWidth())
            {
                  x = x +BackBitmap.getWidth();
            }

      }
      public void update(float dt){
            x = (int) (x - root_gamepanel.ShipSpeed*dt);
      }
}
```

Εικόνα 53: Σχεδιασμός της κλάσης Background

3.2.9. Δημιουργία και δήλωση των μεταβλητών Shipspeed και Background στην κλάση gamepanel

Στο παραπάνω κεφάλαιο αναφερθήκαμε στην δημιουργία της κλάσης και της μεταβλητής Background καθώς και της μεταβλητής shipspeed. Στο επόμενο στάδιο, για να μπορούμε να τις χρησιμοποιήσουμε πρέπει να τις δηλώσουμε στην κλάση gamepanel αλλά και να περιγράψουμε την λειτουργία τους στον constructor.
Η δήλωση λοιπόν, αυτών των δύο μεταβλητών έγινε ως εξής:

```
public class GamePanel extends SurfaceView implements
SurfaceHolder.Callback {
      private Background background;
      public float ShipSpeed;
```

Εικόνα 54: Δήλωση των μεταβλητών ShipSpeed και background στην κλάση gamepanel

Όσον αφορά στην περιγραφή των μεταβλητών καλούμαστε από την μία να ορίσουμε φόντο στην μεταβλητή Background και από την άλλη να ορίζουμε την μεταβλητή shipspeed έτσι ώστε να είναι ίση με το πλάτος της οθόνης(ScreenWidth) δια 2.

```
public GamePanel(Context context, game game,int ScreenWidth) {
            super(context);
            getHolder().addCallback(this);
            this.game = game;
thread = new MainThread(getHolder(),this);
background = new
Background(BitmapFactory.decodeResource(getResources(),
R.drawable.back),ScreenWidth,this);
            ShipSpeed = ScreenWidth/2.f;
```

Εικόνα 55: Πρόσθεση του φόντου

Έπειτα αφού έχουμε ολοκληρώσει συνεχίζουμε με την περιγραφή της Background μεταβλητής στην μεθόδο Draw και Update. Πιο αναλυτικά με την μέθοδο Draw λέμε στο πρόγραμμα να ξεκινάει την σχεδίαση του φόντου σε περίπτωση που το παιχνίδι δεν έχει γίνει Pause αλλά και η συνθήκη canvas δεν είναι ίση με Null. Τέλος με την μέθοδο Update δηλώνουμε το background με την παράμετρο dt.

```
void Draw(Canvas canvas){
            if (!Pause_game)
                  if(canvas!=null){
                        canvas.drawColor(Color.BLACK);
                        background.draw(canvas);
                  }
      }
      void Update(float dt){

      background.update(dt);
      }
```

Εικόνα 56: Σχεδιασμός της background στην μέθοδο Draw και Update

3.2.10. Δημιουργία της κλάσης ship

Η διαδικασία δημιουργίας της κλάσης ship δεν διαφέρει από τις υπόλοιπες που έχουν ήδη δημιουργηθεί. Αφού λοιπόν δημιουργήσουμε την κλάση αρχικά δημιουργούμε ένα constructor στον οποίο δίνουμε τα ορίσματα x,y,ScreenWidth και δηλώνουμε στην κλάση τις μεταβλητές bitmap,x,y,speed,inc,screenwidth,ArrayList<bitmap>Booms και death. Όλα αυτά θα τα χρησιμοποιήσουμε για την δημιουργία αυτής της κλάσης.

```
public class Ship {

      private Bitmap bitmap;
      private int x;
      private int y;
      private int Speed;

      private int inc;
      private int ScreenWidth;
      public ArrayList<Bitmap>Booms  = null;
      boolean death;
```


Εικόνα 57: Μεταβλητές της κλάσης ship

Έπειτα στον constructor αρχικοποιούμε όλες τις μεταβλητές εκτός την array για την οποία θα δημιουργήσουμε ξεχωριστή μέθοδο και θα την ονομάσουμε setBoomAnimation όπου θα περαστεί ως παράμετρος η μεταβλητή Bitmap.
Για την δημιουργία της μεθόδου SetBoomAnimation πρώτα αρχικοποιήσαμε την Array η οποία υπάρχει ήδη στην κλάση μας, στην συνέχεια νέα Array χρησιμοποιώντας την AnimationArray και ορίσαμε δευτερεύουσες μεταβλητές στην κλάση τις οποίες χρησιμοποιήσαμε για να σχεδιάσουμε την απεικόνιση της έκρηξης.

```
public class Ship {
      public ArrayList<Bitmap>Booms  = null;
      float animTime=0;
      float totalAnimTime = 1;
      float numFrames;

public void setBoomAnimation(ArrayList<Bitmap>animation){
            Booms = new ArrayList<Bitmap>(animation);
            numFrames = Booms.size();
      }
```

Εικόνα 58: Δημιουργία της μεθόδου SetBoomAnimation

Έχοντας ολοκληρώσει τα παραπάνω δημιουργούμε την μέθοδο draw της κλάσης ship. Η μέθοδος draw είναι υπεύθυνη για τον σχεδιασμό της θέσης του ship μέσα στην εφαρμογή εφόσον δεν έχει συγκρουστεί με κανένα εμπόδιο της πίστας. Καθώς εμείς θέλουμε το ship να βρίσκεται σε κεντρική θέση στην οθόνη αν ισχύει το παραπάνω κάναμε πρώτα τις αφαιρέσεις χ,y μείον το πλάτος του bitmap και τα αποτελέσματα των πράξεων δια δύο.

```
public void draw(Canvas canvas){
            if (!death)
            {
                  canvas.drawBitmap(bitmap,  x - bitmap.getWidth()/2,y
- bitmap.getHeight()/2,null);
            }
```

Εικόνα 59: Μέθοδος Draw της κλάσης ship

Επίσης στην μέθοδο draw σχεδιάζουμε και την θέση στην οποία θα βρίσκεται το ship σε περίπτωση που δεν συγκρουστεί με κάποιο εμπόδιο.

```
{
                  int index = (int)
(animTime/totalAnimTime*numFrames);
                  if (index<numFrames)
                        canvas.drawBitmap(Booms.get(index),  x -
bitmap.getWidth()/2,y  - bitmap.getHeight()/2,null);
            }
```

Εικόνα 60: Σχεδιασμός θέσης στην μέθοδο Draw

Έχοντας ολοκληρώσει και την μέθοδο draw δημιουργούμε και την μέθοδο update. Σε αυτήν την μέθοδο θα αναβαθμίζουμε τις συντεταγμένες της

απεικόνισης και της μεταβλητές της AnimTime. Εφόσον το ship έχει συγκρουστεί προσθέτουμε την μεταβλητή dt στην AnimTime αν όμως το ship δεν έχει συγκρουστεί του δίνουμε κίνηση. Για να του δώσουμε κίνηση λοιπόν χρησιμοποιήθηκαν 2 νέες μεταβλητές VertSpeed και screenHeight τα οποία αρχικοποιήσαμε στον constructor αλλά και τα δηλώσαμε στην κλάση.

```
public void update(float dt){
            if (death){
                  animTime += dt;
            }
            else
            {
                  VertSpeed+=ScreenHeight/2*dt;
                  if(up)
                        VertSpeed-=ScreenHeight*dt*2;
                  y+=VertSpeed*dt;

                  if (y-(bitmap.getHeight()/2)>ScreenWidth)
                        y=0-(bitmap.getHeight()/2);
            }
      }
```

Εικόνα 61: Δημιουργία της μεθόδου Update

Μετά την δημιουργία και των δύο μεθόδων μεθόδων πηγαίνουμε και τις ορίζουμε στις αντίστοιχες μεθόδους Draw και Update στην κλάση gamepanel.

```
void Draw(Canvas canvas){
            if (!Pause_game)
                  if(canvas!=null){
                        background.draw(canvas);
                        ship.draw(canvas);
                  }
      }
      void Update(float dt){
            ship.update(dt);
      }
```

Εικόνα 62: Σχεδιασμός της μεθόδου Draw στην κλάση gamepanel

Τέλος για την ολοκλήρωση της κλάσης ship δημιουργούμε μία μέθοδο σύγκρουσης. Αυτή η μέθοδος μας βοηθάει να δούμε αν ένας δείκτης βρίσκεται σε ορθογώνια περιοχή. Αν βρίσκεται τότε τον κάνουμε να συγκρουστεί με κάτι και αν όχι τότε δεν συγκρούεται. Έπειτα δημιουργήσαμε μια μέθοδο bump αλλά και getpoint στην οποία ορίσαμε τους δείκτες οι οποίοι είναι οι γωνίες του ορθογωνίου αλλά και δείκτες που αφορούν όλα τα σημεία στην ορθογώνια περιοχή.

```
private void getPoint(Point TL, Point TR, Point BL, Point BR) {
      TL.x = x-bitmap.getWidth() / 2;
      TL.y = y - bitmap.getHeight() / 2;

      TR.x = x+bitmap.getWidth() / 2;
      TR.y = y - bitmap.getHeight() / 2;

      BL.x = x-bitmap.getWidth() / 2;
      BL.y = y + bitmap.getHeight() / 2;

      BR.x = x+bitmap.getWidth() / 2;
      BR.y = y+bitmap.getHeight() / 2;


}
```

Εικόνα 63: Μέθοδος getPoint

```
public boolean bump(Point OTL,Point OTR,Point OBR,Point OBL){
            Point TL = new Point(), TR = new Point(), BL = new
Point(), BR = new Point();

            ArrayList<Point> PointList = new ArrayList<Point>();
            PointList.add(OTL);
            PointList.add(OTR);
            PointList.add(OBR);
            PointList.add(OBL);


      getPoint(TL,TR,BL,BR);

      for (int i=0;i<PointList.size();i++){
            if(BR.x>=PointList.get(i).x)
                  if (TL.x<=PointList.get(i).x)
                        if(PointList.get(i).y>=TL.y)
                              if(PointList.get(i).y<=BR.y)
                                    return true;
      }
      PointList.clear();
      PointList.add(TL);
      PointList.add(TR);
      PointList.add(BR);
      PointList.add(BL);
      for (int i=0;i<PointList.size();i++){
            if (OBR.x>=PointList.get(i).x)
                  if (OTL.x<=PointList.get(i).x)
                        if(PointList.get(i).y>=OTL.y)
                              if (PointList.get(i).y<=OBR.y)
                                    return true;
      }

      return false;

}
```

Εικόνα 64: Μέθοδος bump

3.2.11. Επεξεργασία της μεθόδου OnTouch της κλάσης gamepanel

Αφού έχουμε πλέον τις μεθόδους Draw και Update στην κλάση gamepanel, πηγαίνουμε να ορίσουμε την κίνηση του ship κάθε φορά που θα πατάμε στην οθόνη. Όταν λοιπόν θα αγγίζουμε την οθόνη θέλουμε το ship να πηγαίνει προς τα πάνω και όταν θα την αφήνουμε να πέφτει. Γι αυτό τον λόγο χρησιμοποιήσαμε την μέθοδο MotionEvent στην οποία ορίσαμε αρχικά την μεταβλητή up=true κάθε φορά που αγγίζουμε την οθόνη αλλά και up=false κάθε φορά που την αφήνουμε.

```
public boolean onTouchEvent(MotionEvent  event) {
            if (event.getAction()==MotionEvent.ACTION_DOWN){
                  ship.up=true;
            }
            if(event.getAction()==MotionEvent.ACTION_UP){
                  ship.up=false;
            }
            return true;
      }
```

Εικόνα 65: Επεξεργασία της μεθόδου OnTouchEvent της κλάσης gamepanel

3.2.12. Επεξεργασία της μεθόδου OnclickListener της κλάσης game

Στην μέθοδο OnclickListener της κλάσης game ορίζουμε την μεταβλητή running=false του thread της κλάσης gamepanel αλλά και τερματίζουμε την κλάση game διότι θέλουμε κάθε φορά που θα χάνουμε ή θα κερδίζουμε να μπορούμε να ξεκινήσουμε νέο παιχνίδι. Αν δεν γίνει αυτό το παιχνίδι κάθε φορά που θα χάνουμε ή θα κερδίζουμε στην ουσία δεν θα τελειώνει με αποτέλεσμα όταν θα πηγαίνουμε για να ξεκινήσουμε να σταματάει η εφαρμογή.

```
OnClickListener To_Main_Menu = new OnClickListener() {
            @Override
            public void onClick(View v) {
                  game_panel.thread.setRunning(false);
                  game.this.finish();
```

Εικόνα 66: Επεξεργασία της μεθόδου OnClickListener της κλάσης game

3.2.13. Προσθήκη της απεικόνισης καταστροφής του Ship

Στην κλάση ship έχουμε ήδη δημιουργήσει την μέθοδο setBoomAnimation η οποία όπως είδαμε παραπάνω είναι υπεύθυνη για την απεικόνιση της έκρηξης. Αυτό όμως που δεν έχει οριστεί ακόμα είναι τι θα βλέπουμε όταν θα γίνεται η έκρηξη. Για να το κάνουμε αυτό πηγαίνουμε στον constructor της gamepanel και με την μέθοδο ArrayList ορίζουμε τις τέσσερις εικόνες από τον φάκελο nodpi που είχαμε προσθέσει νωρίτερα. Με αυτόν τον τρόπο λοιπόν καταφέραμε να δημιουργήσουμε την απεικόνιση της έκρηξης.

```
ArrayList<Bitmap> animation=new ArrayList<Bitmap>();

      animation.add(BitmapFactory.decodeResource(getResources(),
R.drawable.boom11));

      animation.add(BitmapFactory.decodeResource(getResources(),
R.drawable.boom22));

      animation.add(BitmapFactory.decodeResource(getResources(),
R.drawable.boom33));

      animation.add(BitmapFactory.decodeResource(getResources(),
R.drawable.boom44));
            ship.setBoomAnimation(animation);
```

Εικόνα 67: Απεικόνιση καταστροφής του ship

3.2.14. Δημιουργία των κλάσεων Barrier Manager και Barrier

Όπως έχουμε δει στα προηγούμενα κεφάλαια έτσι και σε αυτό δημιουργούμε με τον ίδιο τρόπο τις κλάσεις Barrier Manager η οποία θα είναι υπεύθυνη για την τοποθέτηση των Barrier σε συγκεκριμένη περιοχή στην οθόνη αλλά και τν κλάση Barrier στην οποία θα δημιουργήσουμε την όψη των εμποδίων αλλά και πόσα και πώς θα εμφανίζονται στην οθόνη.
Αφού τις δημιουργήσουμε ξεκινάμε να διαμορφώνουμε αρχικά την κλάση Barrier Manager. Πηγαίνουμε λοιπόν και προσθέτουμε στην κλάση όλες τις μεταβλητές τις οποίες θα χρησιμοποιήσουμε στην συνέχεια αλλά και έναν constructor με τα ορίσματα bitmap και gamepanel.

```
public class Barriermanager {

      Bitmap center;
      int shipHeight;
      int Num;
      int screenH;
      int TargetY =-1;
      int dpos;
      int dl;
      public GamePanel game_panel;

      ArrayList<Barier> TopWalls = null;
      ArrayList<Barier> BottomWalls = null;

public Barriermanager(Bitmap decodeResource,GamePanel game_panel) {
            center = decodeResource;
            this.game_panel = game_panel;
```

Εικόνα 68: Οι μεταβλητές και ο constructor της κλάσης BarrierManager

Έπειτα δημιουργούμε την μέθοδο shipheight η οποία θα ορίζουμε το ύψος του ship,

```
void setShip (int h){
      shipHeight = h;
```

Εικόνα 69: Μέθοδος setShip

την μέθοδο Setscreen όπου θα αντλεί δεδομένα απο την κλάση Barrier και θα μας λέει πόσα Barrier θα σχεδιάζονται στην οθόνη και θα μας δημιουργεί την οφθαλμαπάτη ότι πηγαίνουμε συνέχεια προς τα κάτω στον διάδρομο,

```
public void setscreeN (int width,int height){
              screenH =height;
              Num=(width)/center.getWidth()+4;
              TopWalls=new ArrayList<Barier>();
              BottomWalls=new ArrayList<Barier>();
              for (int i=0;i<Num+1;i++){
                    Barier BB = new
Barier(center,width+200+center.getWidth()*i,0);
                    BB.setManager (this);
                    TopWalls.add(BB);

                    Barier BBB = new
Barier(center,width+200+center.getWidth()*i,0);
                    BBB.setManager (this);
                    BottomWalls.add(BBB);

              }
Generate();
```

Εικόνα 70: Μέθοδος setscreen

την μέθοδο generate η οποία θα παράγει και θα τοποθετεί τα Barrier στο πάνω αλλά και κάτω μέρος της οθόνης μας

```
private void Generate() {
      int h = center.getHeight()/2;
      dl = screenH;
      dpos = screenH/2;
      int new_dl = screenH*3/5;
      int inc = (dl-new_dl)/Num;
      for (int i=0; i <Num+1;i++)
      {
            dl=dl-inc;
            h = TopWalls.get(i).getBitmap().getHeight()/2;
            TopWalls.get(i).setY(dpos-dl/2-h);
            BottomWalls.get(i).setY(dpos+dl/2+h);

      }
```

Εικόνα 71: Μέθοδος Generate

και τις μεθόδους draw και update στις η οποίες είναι υπεύθυνες για τον σχεδιασμό και την αναβάθμιση.

```
public void draw(Canvas canvas){

        for (int i=0;i<Num+1;i++){

            TopWalls.get(i).draw(canvas);

            BottomWalls.get(i).draw(canvas);
        }

}
      public void update(float dt){

        for (int i=0;i<Num+1;i++){

            TopWalls.get(i).update(dt, true);

            BottomWalls.get(i).update(dt, false);
        }
       }
```

Εικόνα 72: Μέθοδοι Draw και Update

Έχοντας ολοκληρώσει την κλάση BarrierManager πηγαίνουμε να επεξεργαστούμε και την κλάση Barrier. Όπως και στην Barrier Manager πρώτα ορίζουμε στην κλάση τις μεταβλητές που θα χρησιμοποιήσουμε καθώς και έναν constructor με τα ορίσματα Bitmap Center, x,y.

```
private Bitmap bitmap;
      private int x;
      private int y;

      Barriermanager BM;
      boolean doit;

public Barier(Bitmap center, int x, int y) {


         bitmap=center;
         this.x=x;
         this.y=y;
   }
```

Εικόνα 73: Οι μεταβλητές και ο constructor της κλάσης barrier

Έπειτα δημιουργούμε τις μεθόδους setmanager η οποία είναι ο σύνδεσμος μεταξύ της Barrier Manager και Barrier έτσι ώστε η Barrier Manager να μπορεί να διαβάσει τα δεδομένα της κλάσης Barrier,

```
public void setManager(Barriermanager barriermanager) {

        BM = barriermanager;

    }
```

Εικόνα 74: Μέθοδος setManager

την μέθοδο Bitmap η οποία είναι υπεύθυνη για το φόντο της οθόνης,

```
public Bitmap getBitmap() {

        return bitmap;
    }
```

Εικόνα 75: Μέθοδος Bitmap

την SetY όπου ορίζεται η μεταβλητή y,

```
public void setY(int y) {

        this.y=y;
    }
```

Εικόνα 76: Μέθοδος setY

τις μεθόδους draw, update για τον σχεδιασμό και την εμφάνιση τους στην οθόνη

```
    public void draw(Canvas canvas) {

        canvas.drawBitmap(bitmap, x-(bitmap.getWidth()/2), y-
(bitmap.getHeight()/2),null);
    }
    public void update(float dt, boolean b) {
        if (x<-bitmap.getWidth()){
            if(b){
                if (Math.abs(BM.TargetY-BM.dpos)<50)
                    doit=true;
                if((BM.TargetY==-1)||doit){
                    BM.TargetY= new
Random().nextInt(BM.screenH-BM.dl/2)+BM.dl/4;
                }
                if (BM.dpos<BM.TargetY)
                    BM.dpos=BM.dpos+new
Random().nextInt(15);
                else
                    BM.dpos=BM.dpos-new
Random().nextInt(15);

                y=BM.dpos-BM.dl/2-bitmap.getHeight()/2;
            }
            else
            {
                y=BM.dpos+BM.dl/2+bitmap.getHeight()/2;

            }
            x=(int)(x+bitmap.getWidth()*(BM.TopWalls.size()-
1));
        }
        x=(int)(x-BM.game_panel.ShipSpeed*dt);
    }
```

Εικόνα 77: Μέθοδοι Draw και Update

και την μέθοδο ArrayList η οποία θα είναι αυτή που καθορίζει τα σημεία σύγκρουσης του ship με τα Barrier.

```
        public ArrayList<Point> GetArray() {
                Point TL = new Point(), TR = new Point(), BL = new
    Point(), BR = new Point();
                TL.x = x-bitmap.getWidth()  / 2;
                TL.y = y - bitmap.getHeight()  / 2;

                TR.x = x+bitmap.getWidth()  / 2;
                TR.y = y - bitmap.getHeight()  / 2;

                BL.x = x-bitmap.getWidth()  / 2;
                BL.y = y + bitmap.getHeight()  / 2;

                BR.x = x+bitmap.getWidth()  / 2;
                BR.y = y+bitmap.getHeight()  / 2;

                ArrayList<Point> temp = new ArrayList<Point>();
                temp.add(TL);
                temp.add(TR);
                temp.add(BR);
                temp.add(BL);
                return temp;
        }

    }
```

Εικόνα 78: Μέθοδος ArrayList

3.2.15. Δήλωση και δημιουργία της μεταβλητής Barrier manager στην κλάση gamepanel

Μετά και την δημιουργία της κλάσης Barrier manager και Barrier πηγαίνουμε στην κλάση gamepanel όπου αρχικά δηλώνουμε την Barriermanager.

```
public class GamePanel extends SurfaceView implements
SurfaceHolder.Callback {

      private Barriermanager BM;
```

Εικόνα 79: Δήλωση της BarrierManager στην κλάση gamepanel

Στην συνέχεια μέσα στον constructor της gamepanel ορίζουμε τις παραμέτρους ScreenWidth και ScreenHeight τις οποίες χρησιμοποιήσαμε στην κλάση Barrier manager και στην συνέχεια ορίζουμε την εικόνα που θα έχει το barrier.

```
public GamePanel(Context context, game game,int ScreenWidth,int
Screenheight) {
            super(context);
            getHolder().addCallback(this);
            this.game = game;
            BM = new
Barriermanager(BitmapFactory.decodeResource(getResources(),
R.drawable.obstacle),this);
BM.setscreeN(ScreenWidth, Screenheight);
```

Εικόνα 80: Ορισμός των παραμέτρων ScreenWidth και Screenheight

Παρακάτω στην μέθοδο draw δηλώσαμε τον σχέδιο το οποίο έχουμε ήδη δημιουργήσει στις κλάσεις Barrier manager και Barrier.

Τέλος στην μέθοδο update της κλάσης gamepanel δημιουργήσαμε το σημείο σύγκρουσης του ship με τα Barrier με την βοήθεια της μεθόδου ArrayList η οποία περιλάμβανε μία συνάρτηση if στην οποία περιγράψαμε τι θα συμβαίνει αν το ship συγκρουστεί και τι αν δεν συγκρουστεί. Σε περίπτωση που το ship έρθει σε επαφή με κάποιο Barrier το παιχνίδι θα σταματά και στην αντίθετη περίπτωση θα συνεχίζει κανονικά.

```
void Draw(Canvas canvas){
            if (!Pause_game)
                  if(canvas!=null){
                        canvas.drawColor(Color.BLACK);
                        BM.draw(canvas);
                  }
      }
      void Update(float dt){

            ship.update(dt);
            if(!ship.death)
            {
                  background.update(dt);
                  BM.update(dt);

                  ArrayList<Point>Coin_Point  = new
ArrayList<Point>(Coin.GetArray());

            for(int i=0;i<BM.TopWalls.size();i++){
                  ArrayList<Point>temp=new
ArrayList<Point>(BM.TopWalls.get(i).GetArray());
                  ArrayList<Point>temp2=new
ArrayList<Point>(BM.BottomWalls.get(i).GetArray());

            if((ship.bump(temp.get(0), temp.get(1), temp.get(2),
temp.get(3))||(ship.bump(temp2.get(0), temp2.get(1), temp2.get(2),
temp2.get(3))))
                  {
                        ship.death=true;
                        Message msg =
BM.game_panel.game.handler.obtainMessage();
                        msg.what = 1;
                        BM.game_panel.game.handler.sendMessage(msg);
                  }
            }
      }
```

Εικόνα 81: Σχεδιασμός σημείου σύγκρουσης του ship με τα Barrier

3.2.16. Δημιουργία κλάσης Bonus

Η κλάση bonus στην δική μας περίπτωση είναι θησαυροί που ο παίχτης πρέπει να συλλέξει έτσι ώστε να κερδίσει. Έχοντας λοιπόν δημιουργήσει την κλάση Bonus αρχικοποιούμε σε αυτή τις μεταβλητές που θα χρησιμοποιήσουμε. Αυτές είναι η μεταβλητή bitmap,x,y αλλά και

Barriermanager η οποία στην συνέχεια θα χρησιμεύσει ως προς το σημείο όπου θα εμφανίζονται οι θησαυροί.

```
public class Bonus {
	private Bitmap bitmap;
	private int x;
	private int y;
	Barriermanager BM;
```

Εικόνα 82: Μεταβλητές της κλάσης Bonus

Αφού ορίσουμε και τις μεταβλητές δημιουργούμε έναν constructor με παραμέτρους decodeResource,x και y.

```
public Bonus(Bitmap decodeResource, int x, int y) {

		this.bitmap = decodeResource;
		this.x = x;
		this.y = y;
```

Εικόνα 83: Δήλωση παραμέτρων στον constructor

Στην συνέχεια δημιουργούμε πρώτα την μέθοδο setBarriermanager. Αυτό το κάνουμε για να ξέρουμε τις συντεταγμένες τις οποίες δημιουργούνται τα Barrier έτσι ώστε οι θησαυροί μας να εμφανίζονται εκεί που δεν υπάρχουν Barrier έτσι ώστε να μπορεί ο παίχτης να τους πάρει.

```
public void setBarriermanager(Barriermanager candidate){
		BM=candidate;
	}
```

Εικόνα 84: Μέθοδος SetBarriermanager

Δημιουργούμε επίσης την μέθοδο Bitmap getBitmap η οποία είναι υπεύθυνη για το φόντο της οθόνης.

```
	public Bitmap getBitmap(){
		return bitmap;
	}
```

Εικόνα 85: Μέθοδος Bitmap

Ακόμη με την δημιουργία της ArrayList ορίσαμε τα σημεία σύγκρουσης των θησαυρών με το Ship.

```
	public ArrayList<Point> GetArray() {
		Point TL = new Point(), TR = new Point(), BL = new
Point(), BR = new Point();
		TL.x = x-bitmap.getWidth() / 2;
		TL.y = y - bitmap.getHeight() / 2;

}		TR.x = x+bitmap.getWidth() / 2;
		TR.y = y - bitmap.getHeight() / 2;

		BL.x = x-bitmap.getWidth() / 2;
		BL.y = y + bitmap.getHeight() / 2;

		BR.x = x+bitmap.getWidth() / 2;
		BR.y = y+bitmap.getHeight() / 2;

		ArrayList<Point> temp = new ArrayList<Point>();
		temp.add(TL);
		temp.add(TR);
		temp.add(BR);
		temp.add(BL);
		return temp;
	}
```

Εικόνα 86: Σημεία σύγκρουσης

Στην συνέχεια στις μεθόδους draw και update σχεδιάσαμε την θέση που θα εμφανίζονται οι θησαυροί στο παιχνίδι. Χρησιμοποιώντας την μεταβλητή Random στην ουσία είπαμε στο πρόγραμμα να εμφανίζει σε τυχαία σημεία τους θησαυρούς.

```
public void draw(Canvas canvas){
canvas.drawBitmap(bitmap, x - (bitmap.getWidth() / 2), y -
(bitmap.getHeight() / 2), null);
	}

public void update(float dt)
	{
		if (x<-BM.game_panel.ScreenWidth/4)
		{
			x=BM.game_panel.ScreenWidth+bitmap.getWidth();
			Random r = new Random();
			y=r.nextInt(BM.dl)+BM.dpos-BM.dl/2;
	}
		x -=BM.game_panel.ShipSpeed*dt;
	}
```

Εικόνα 87: Μέθοδοι Draw και Update για τον καθαρισμό της θέσης των θησαυρών στην πίστα

Τέλος στις μεθόδους SetX και SetY απλά ορίσαμε τις μεταβλητές x,y οι οποίες είναι υπεύθυνες για την θέση του θησαυρού όπως έχει οριστεί στην gamepanel.

```
	public void SetX(int x) {
		// TODO Auto-generated method stub
		this.x=x;
	}

	public void SetY(int y) {
		// TODO Auto-generated method stub
		this.y=y;
	}

}
```

Εικόνα 88: Μέθοδοι SetX και SetY

3.2.17. Δήλωση παραμέτρων και δημιουργία της μεταβλητής Bonus στην κλάση gamepanel

Στο παραπάνω κεφάλαιο έχουμε δημιουργήσει την κλάση Bonus την οποία πρέπει να την δηλώσουμε στην κλάση gamepanel αλλά και να δημιουργήσουμε το σημείο σύγκρουσης του ship με τον θησαυρό.

```
public class GamePanel extends SurfaceView implements
SurfaceHolder.Callback {

private Ship ship;
private Barriermanager BM;
private Bonus Coin;
```

Εικόνα 89: Δήλωση της μεταβλητής Bonus στην κλάση gamepanel

Έπειτα στον constructor ορίζουμε την εικόνα του θησαυρού αλλά και το μέγεθος που θα έχει μέσα στην εφαρμογή.

```
public GamePanel(Context context, game game,int ScreenWidth,int
Screenheight) {
            super(context);
            getHolder().addCallback(this);
            this.game = game;
            Coin = new
Bonus(BitmapFactory.decodeResource(getResources(),
R.drawable.bonus1),-200,-200);
            Coin.setBarriermanager(BM);
```

Εικόνα 90: Ορισμός εικόνας και μεγέθους θησαυρού

Στην συνέχεια στην μέθοδο draw δηλώνουμε το σχέδιο του θησαυρού που έχουμε φτιάξει στην κλάση Bonus.

```
void Draw(Canvas canvas){
            if (!Pause_game)
                  if(canvas!=null){

canvas.drawColor(Color.BLACK);
            Coin.draw(canvas);
```

Εικόνα 91: Δήλωση του σχεδίου της κλάσης Bonus στην μέθοδο Draw της κλάσης gamepanel

Τέλος στην μέθοδο update με την χρήση της ArrayList δημιουργήσαμε το σημείο σύγκρουσης του ship με τον θησαυρό καθώς επίσης ορίσαμε στο πρόγραμμα να εξαφανίζει τον θησαυρό σε περίπτωση όπου ο χρήστης αποτύχει να το πάρει.

```
        void Update(float dt){

              ship.update(dt);
              if(!ship.death)
              {
                    background.update(dt);
                    BM.update(dt);
                    Coin.update(dt);

                    ArrayList<Point>Coin_Point = new
  ArrayList<Point>(Coin.GetArray());


                    if(ship.bump(Coin_Point.get(0), Coin_Point.get(1),
  Coin_Point.get(2), Coin_Point.get(3))){
                          Coin.SetX(-200);
                          Coin.SetY(-200);
                          Message msg =
  BM.game_panel.game.handler.obtainMessage();
                          msg.what = 0;
                          BM.game_panel.game.handler.sendMessage(msg);
                    }
```

Εικόνα 92: Ορισμός σημείων σύγκρουσης στην μέθοδο Update

3.2.18. Προσθήκη δεδομένων στην κλάση game

Έχουμε ήδη δημιουργήσει την κλάση game σε παραπάνω κεφάλαιο και έχουμε προσθέσει και λίγα δεδομένα. Για να ολοκληρώσουμε όμως την εφαρμογή πρέπει να προσθέσουμε και κάποια ακόμη.
Αρχικά προσθέτουμε και δίνουμε τιμές σε τέσσερις μεταβλητές τις οποίες στην συνέχεια θα χρησιμοποιήσουμε μέσα στην κλάση game. Αυτές είναι: η UPDATE_SCORE,DEATH,LOSE αλλά και MainMusic την οποία θα χρησιμοποιήσουμε για να βάλουμε ήχους στην εφαρμογή μας.

```
public class game extends Activity {

	final static int UPDATE_SCORE = 0;
	final static int DEATH = 1;
	final static int LOSE = 2;
	MediaPlayer MainMusic;
```

Εικόνα 93: Δήλωση μεταβλητών στην κλάση game

Στην συνέχεια φτιάχνουμε την μέθοδο Handler. Η μέθοδος αυτή είναι υπεύθυνη για τον χειρισμό των μηνυμάτων. Ανάλογα το μήνυμα που θα τις δίνεται θα κάνει την συγκεκριμένη ενέργεια. Για παράδειγμα σε αυτήν την εφαρμογή της είπαμε άμα δέχεται μεταβλητή UPDATE SCORE να αναβαθμίζει το score μας τόσο όσο θα το ορίσουμε αργότερα στην μέθοδο i_get_coin στην και αν δέχεται μεταβλητή DEATH να μας βγάζει το μήνυμα της ήττας που επίσης θα δημιουργήσουμε στην συνέχεια.

```
final Handler handler = new Handler (){
		@Override
		public void handleMessage(Message msg){
			if(msg.what==UPDATE_SCORE){
				i_get_coin();
			}
			if (msg.what == DEATH) {
				postDelayed(new Runnable() {
					@Override
					public void run() {
						Message msg =
handler.obtainMessage();
						msg.what = LOSE;
						handler.sendMessage(msg);
					}
				}, 3000);
			}
				if (msg.what == LOSE) {
					i_lose();
				}

			super.handleMessage(msg);
		}
	};
```

Εικόνα 94: Μέθοδος Handler

Επίσης δημιουργούμε δυναμικά στην μέθοδο myInflater ένα Relative Layout όπου θα απεικονίζεται ένα πλαίσιο του οποίου η θέση και το μέγεθος ορίζεται από εμάς καθώς όμως και το κείμενο score με κίτρινο χρώμα και συγκεκριμένη γραμματοσειρά που δίπλα του θα αναγράφονται οι πόντοι μας.

```
RelativeLayout RR = new RelativeLayout(this);
      RR.setBackgroundResource(R.drawable.btn1);
      RR.setGravity(Gravity.CENTER);
      Rel_main_game.addView(RR, 180, 80);
      RR.setX(0);
      txt = new TextView(this);
      Typeface Custom = Typeface.createFromAsset(getAssets(),
"waltographUI.ttf");
      txt.setTypeface(Custom);
      txt.setTextColor(Color.YELLOW);
      txt.setText("Score   " + score);
      RR.addView(txt);
```

Εικόνα 95: Δημιουργία απεικόνισης του σκορ

3.2.19. Δημιουργία κλάσης TouchButton

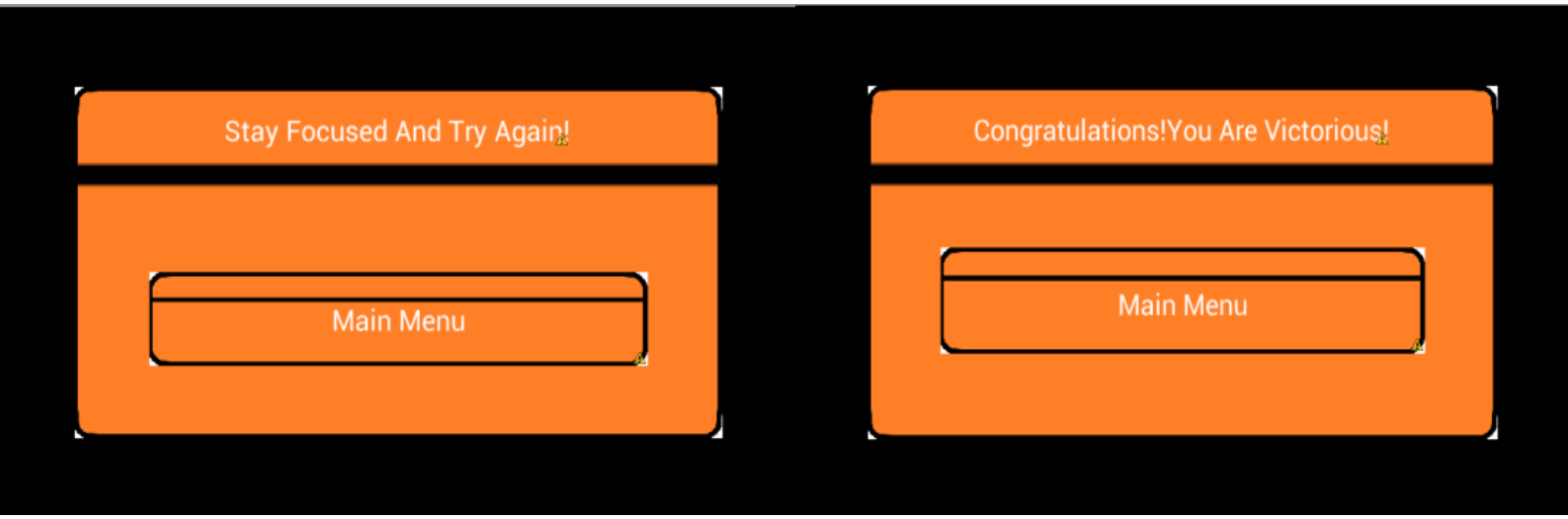
Αυτή η κλάση δημιουργήθηκε ως touchListener και όχι ως Activity. Στην κλάση λοιπόν TouchButton ορίσαμε την μορφή που θα έχουν τα κουμπιά που έχουμε τοποθετήσει στην εφαρμογή μας όταν θα τα κρατάμε πατημένα αλλά και όταν θα τα αφήνουμε. Αυτό που κάναμε παρακάτω είναι το εξής: Κάθε φορά που ο χρήστης κρατάει πατημένο κάποιο κουμπί αυτό αλλάζει μορφή και κάθε φορά που το αφήνει ξανάρχεται στην αρχική του κατάσταση.

```
import android.view.MotionEvent;
import android.view.View;
import android.view.View.OnTouchListener;
import android.widget.ImageView;

public class TochButton implements OnTouchListener {
      ImageView IM;

      public TochButton(ImageView imageButton) {
            IM = imageButton;
      }

      @Override
      public boolean onTouch(View arg0, MotionEvent event) {
            switch (event.getAction()) {
            case MotionEvent.ACTION_DOWN:
                  IM.setImageResource(R.drawable.btn2);
                  break;
            case MotionEvent.ACTION_UP:
                  IM.setImageResource(R.drawable.btn1);
                  break;
            case MotionEvent.ACTION_OUTSIDE:
                  IM.setImageResource(R.drawable.btn1);
                  break;
            default:
                  break;


            }
            return false;
} }
```


Εικόνα 96: Κλάση TouchButton

3.2.20. Δημιουργία XML και μεθόδων win και lose

Όπως είπαμε στο παραπάνω κεφάλαιο κάθε φορά που ο παίχτης θα χάνει η θα κερδίζει ανάλογα με το αποτέλεσμα θα εμφανίζεται και ένα μήνυμα. Αυτό για να συμβεί αρχικά δημιουργήσαμε στην γραφική προσέγγιση του XML δύο αρχεία. Ένα για την ήττα αλλά και ένα για την νίκη. Τα XML αυτά δημιουργήθηκαν όπως ακριβώς και τα άλλα που έχουμε φτιάξει.



Εικόνα 97: Μενού νίκης και ήττας

Έπειτα έχοντας δημιουργήσει τα XML αρχεία πηγαίνουμε στην κλάση game όπου και θα δηλώσουμε τις μεταβλητές win και lose που θα μας χρειαστούν για να εντάξουμε στην μέθοδο myInflater της κλάσης game την εμφάνιση τους.

```
WinDialog = myInflater.inflate(R.layout.win,  null, false);
      Rel_main_game.addView(WinDialog);
      ImageView Win_to_main =
(ImageView)WinDialog.findViewById(R.id.toMain);
      Win_to_main.setOnTouchListener(new TochButton(Win_to_main));
      Win_to_main.setOnClickListener(To_Main_Menu);
      WinDialog.setVisibility(View.GONE);

      LoseDialog = myInflater.inflate(R.layout.lose,  null, false);
      Rel_main_game.addView(LoseDialog);
      ImageView Lose_to_main =
(ImageView)LoseDialog.findViewById(R.id.toMain);
      Lose_to_main.setOnTouchListener(new TochButton(Lose_to_main));
      Lose_to_main.setOnClickListener(To_Main_Menu);
      LoseDialog.setVisibility(View.GONE);
```

Εικόνα 98: Δημιουργία των διαλόγων win και lose

Σε αυτές τις δύο μεθόδους συνδέσαμε το κουμπί Main Menu έτσι ώστε να μας γυρνάει στην αρχική όταν το πατάμε και συνδέσαμε την κλάση TouchButton για να αλλάζει η όψη των κουμπιών όταν τα πατάμε και όταν τα αφήνουμε.

3.2.21. Δημιουργία μεθόδων onBackPressed, onStop, i_lose,i win και i get_coin

Οι τελευταίες ενέργειες που πρέπει να γίνουν έτσι ώστε να ολοκληρωθεί το πρόγραμμα είναι η δημιουργία των παρακάτω μεθόδων: Αρχικά δημιουργούμε την μέθοδο OnBackPressed απο την οποία και ζητάμε κάθε φορά που πατάμε από την συσκευή μας το κουμπί που μας γυρνάει πίσω να μας πηγαίνει στο Pause Menu και όχι στο Main Menu.

```
@Override
      public void onBackPressed() {
            pauseButton.setVisibility(View.GONE);
            PauseMenu.setVisibility(View.VISIBLE);
            game_panel.Pause_game = true;
      }
```

Εικόνα 99: Μέθοδος onBackPressed

Έπειτα δημιουργούμε την μέθοδο onStop η οποία είναι αυτή η οποία θα σταματάει τον έναν ήχο έτσι ώστε όταν πηγαίνουμε για παράδειγμα από το Main Menu στο βασικό παιχνίδι να σταματάει τον ήχο του Menu έτσι ώστε να ξεκινήσει ο ήχος που έχουμε για το βασικό παιχνίδι.

```
@Override
protected void onStop() {
      if(MainMusic.isPlaying())
            MainMusic.stop();
      super.onStop();
}
```

Εικόνα 100: Μέθοδος onStop

Στην μέθοδο i_lose ορίζουμε να εμφανίζεται το XML αρχείο που έχουμε δημιουργήσει, καθώς όμως και έναν ήχο για όπου θα συνοδεύει το Menu κάθε φορά που θα χάνουμε.

```
protected void i_lose() {
      if (MainMusic.isPlaying())
            MainMusic.stop();
      MainMusic = MediaPlayer.create(game.this, R.raw.lose);
      MainMusic.start();
      game_panel.Pause_game = true;
      LoseDialog.setVisibility(View.VISIBLE);
}
```

Εικόνα 101: Μέθοδος i_lose

Στην συνέχεια στην μέθοδο i_get_coin ορίζουμε κατα πόσο θα αυξάνεται το score κάθε φορά που θα πέρνουμε έναν θησαυρό, τον ήχο που θα ακούγεται, αλλά και στους πόσους θησαυρούς θα νικάμε.

```
protected void i_get_coin() {
	get_coins++;
	score+=200;
	txt.setText("Score " + score);
	if (get_coins == 50) {
		i_win();
	}
	MediaPlayer mp=MediaPlayer.create(game.this, R.raw.item);
	MainMusic.setVolume(0.3f, 0.3f);
	mp.start();
	}
```

Εικόνα 102: Μέθοδος i_get_coin

Τέλος στην μέθοδο i_win κάνουμε ότι και στην μέθοδο i_lose απλά αλλάζουμε το αρχείο XML που θα εμφανίζεται και τον ήχο που θα το συνοδεύει σε περίπτωση νίκης.

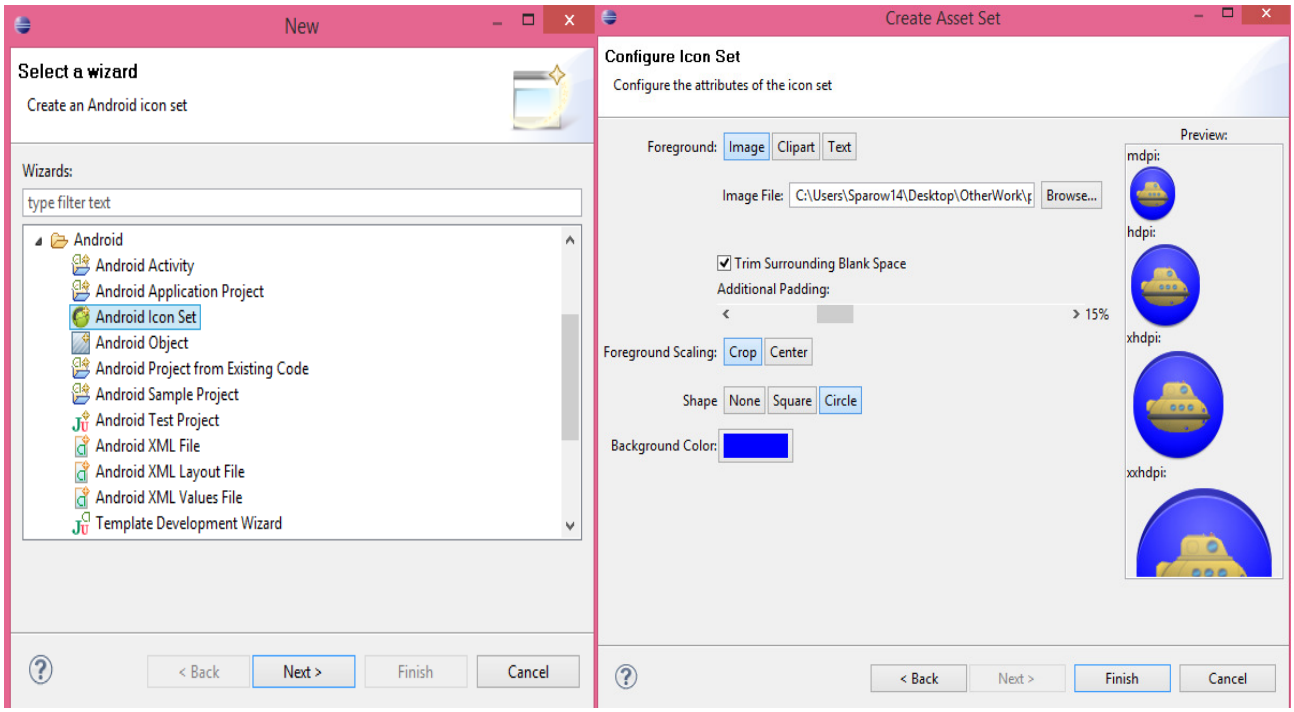
```
private void i_win() {
	if (MainMusic.isPlaying())
		MainMusic.stop();
	 MainMusic = MediaPlayer.create(game.this, R.raw.win);
	MainMusic.start();
	game_panel.Pause_game = true;
	WinDialog.setVisibility(View.VISIBLE);
}
}
```

Εικόνα 103: Μέθοδος i_win

ΚΕΦΑΛΑΙΟ 4: Ανεβάζοντας Μια Εφαρμογή Στο Google Play

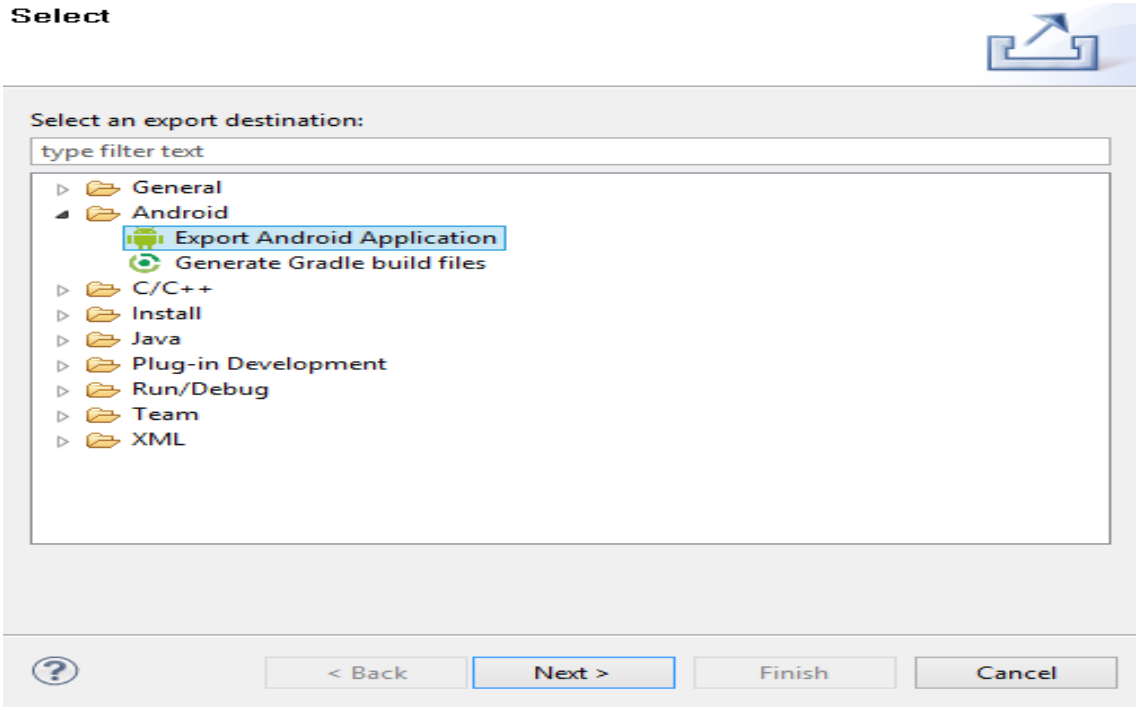
4.1.Πώς ανεβάζουμε μια εφαρμογή στο Google Play

Έχοντας ολοκληρώσει την εφαρμογή μας εάν το επιθυμούμε μας δίνεται η δυνατότητα να την ανεβάσουμε στο Google Play και από εκεί να την κατεβάσει όποιος επιθυμεί. Για να γίνει αυτό λοιπόν ακολουθούνται τα εξής βήματα.
Πρώτον για να την ανεβάσουμε την εφαρμογή μας πρέπει να της ορίσουμε ένα μοναδικό εικονίδιο. Αυτό για να το κάνουμε πατάμε δεξί κλικ πάνω στο project μας--> New-->Other και από εκεί στο φάκελο android επιλέγουμε το Android Icon Set.
Πατώντας λοιπόν σε αυτή την επιλογή μας δίνεται η δυνατότητα να επιλέξουμε εικονίδιο για την εφαρμογή. Αφού επιλέξουμε πατάμε Finish.



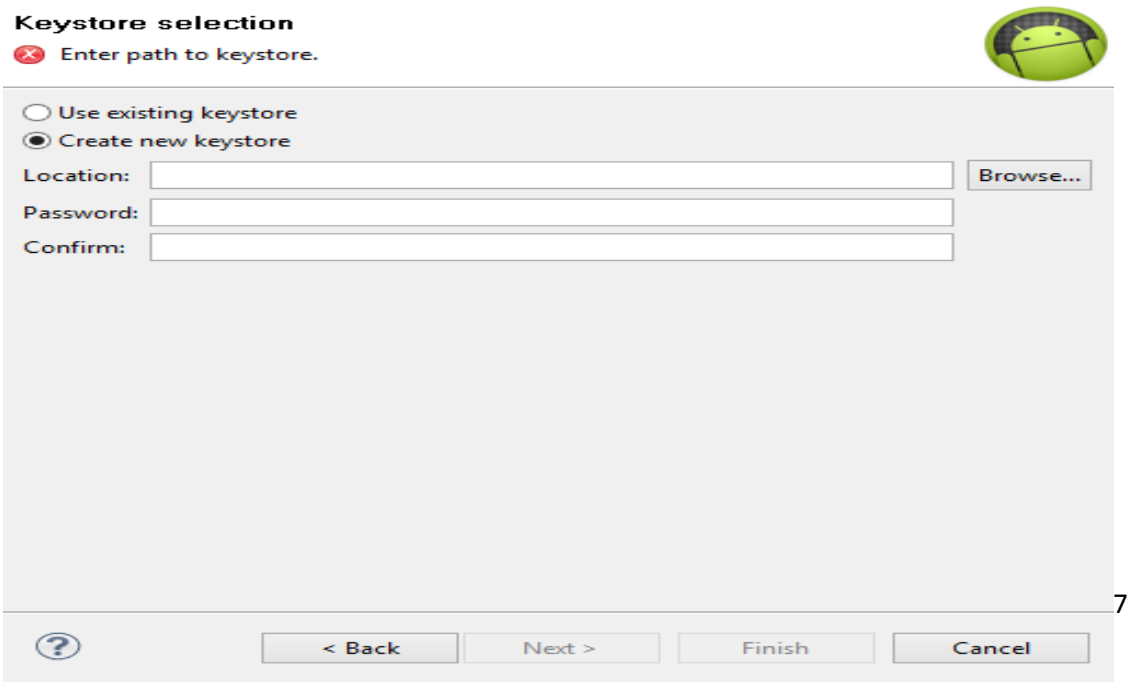
Εικόνα 104: Προσθήκη εικονιδίου εφαρμογής

Αφού δημιουργήσουμε και το εικονίδιο που θέλουμε επόμενο βήμα είναι να εξάγουμε την εφαρμογή μας.
Για να εξάγουμε λοιπόν την εφαρμογή πηγαίνουμε στο project μας πατάμε δεξί κλικ-->Export και στον φάκελο Android επιλέγουμε Export Android Application.



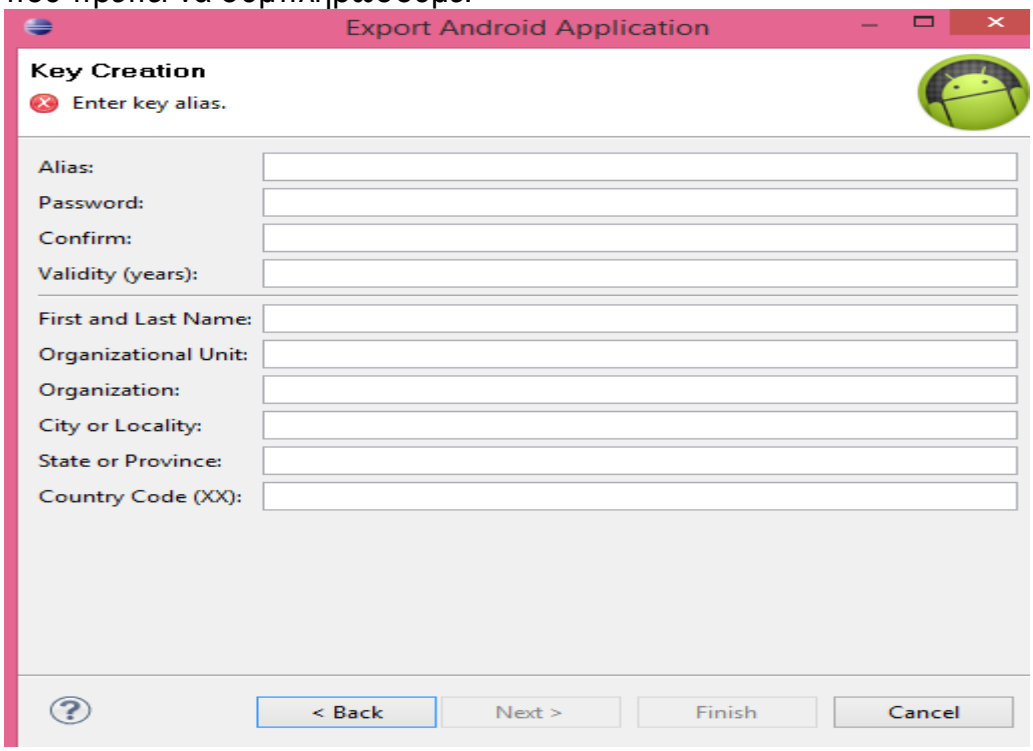
Εικόνα 105: Καρτέλα Export

Στην καρτέλα που θα μας ανοίξει μας δίνεται η επιλογή να επιλέξουμε το project που θέλουμε. Όταν επιλέγουμε λοιπόν και πατάμε next. Η επόμενη καρτέλα μας λέει οτι χρειαζόμαστε κάποιο κλειδί για να ολοκληρώσουμε την εξαγωγή. Σε περίπτωση που δεν έχουμε απλά δημιουργούμε ένα και του δίνουμε και έναν κωδικό.



Εικόνα 106: Καρτέλα επιλογής κλειδιού

Όταν συμπληρώσουμε τα παραπάνω πεδία πηγαίνουμε στην επόμενη καρτέλα στην οποία θα δημιουργήσουμε το κλειδί. Σε αυτήν την καρτέλα μας ζητούνται πάρα πολλά στοιχεία όπως τηλέφωνο, όνομα, πολιτεία και άλλα που πρέπει να συμπληρώσουμε.



Εικόνα 107: Καρτέλα δημιουργίας κλειδιού

Τελευταία καρτέλα μετά και από την συμπλήρωση των στοιχείων είναι η καρτέλα όπου επιλέγουμε το αρχείο seagame.apk ρο οποίο είναι η εφαρμογή μας και είναι αυτό θα ανεβάσουμε στο google play.
Έχοντας ολοκληρώσει όλα τα παραπάνω πηγαίνουμε να ανεβάσουμε την εφαρμογή στο google play. Για να το κάνουμε αυτό πηγαίνουμε στην διεύθυνση play.google.com/apps/publish. Εκεί θα μας ζητηθεί να πληρώσουμε ένα ποσό έτσι ώστε να ανεβάσουμε την εφαρμογή μας.
Η συγκεκριμένη εφαρμογή δεν έχει ανέβει στο google play διότι είναι σε δοκιμαστικό στάδιο αλλά αυτά ήταν τα βήματα τα οποία θα ακολουθήσουμε για να την ανεβάσουμε.

ΚΕΦΑΛΑΙΟ 5: Τρόποι Βελτίωσης Της Εφαρμογής

5.1. Προτάσεις Βελτίωσης της εφαρμογής

Η συγκεκριμένη όπως και όλες η εφαρμογές επειδή μπορούν να τρέξουν δεν σημαίνει απαραίτητα ότι είναι και στο τελικό σημείο που μπορούν να φτάσουν. Εξάλλου τίποτα δεν δημιουργείτε και είναι τέλειο κατευθείαν.
Το επόμενο στάδιο γι αυτήν την εφαρμογή είναι αρχικά να αποθηκεύονται τα καλύτερα 5 σκορ. Επίσης να υπάρχουν διαφορετικές πίστες όπου ο παίχτης θα μπορεί να πάει στην επόμενη με την προϋπόθεση να έχει περάσει την προηγούμενη. Αυτές οι πίστες θα έχουν διαφορετικό περιβάλλον μεταξύ τους και μεγαλύτερη δυσκολία. Μία εξίσου πολύ σημαντική βελτίωση που πρέπει να γίνει είναι να μπορεί να είναι συμβατή η εφαρμογή αυτή με τις πιο σύγχρονες εκδόσεις Android έτσι ώστε να μπορούν να την χρησιμοποιήσουν ακόμα περισσότεροι χρήστες. Τελευταίο βήμα όπου έχει ως προϋπόθεση να γίνουν τα παραπάνω είναι να ανέβει αυτή η εφαρμογή στο playstore του google market έτσι ώστε να μπορεί να την αποκτήσει ο οποιοσδήποτε.

ΒΙΒΛΙΟΓΡΑΦΙΑ - ΔΙΚΤΥΟΓΡΑΦΙΑ

1. http://dspace.cst.uop.gr:8080/bitstream/handle/123456789/25/cst04026_thesis_final_report_march_2013-cv.pdf?sequence=1 (Πάρθηκε στις 11-5-2015)
2. https://el.wikipedia.org/wiki/Android (Πάρθηκε στις 1-6-2015)
3. http://developer.samsung.com/technical-doc/view.do?v=T000000117 (Πάρθηκε στις 2-4-2015)
4. http://www.oracle.com/technetwork/java/javase/downloads/index.html(Πάρθηκε στις 2-4-2015)
5. http://developer.android.com/tools/help/adt.html(Πάρθηκε στις 2-4-2015)
6. https://eclipse.org/downloads/packages/release/Indigo/SR2(Πάρθηκε στις 2-4-2015)
7. http://www.allaboutandroid.gr/?p=6362 (Πάρθηκε στις 2-6-2015)
8. http://developer.android.com/guide/topics/graphics/2d-graphics.html(Πάρθηκε στις 7-5-2015)
9. http://developer.android.com/reference/android/view/LayoutInflater.html (Πάρθηκε στις 5-4-2015)
10. http://developer.android.com/reference/android/view/View.OnTouchListener.html (Πάρθηκε στις 14-4-2015)
11. http://developer.android.com/reference/android/widget/Button.html (Πάρθηκε στις 14-4-2015)
12. http://developer.android.com/reference/android/app/Activity.html (Πάρθηκε στις 22-4-2015)
13. http://stackoverflow.com/questions/6413700/android-proper-way-to-use-onbackpressed (Πάρθηκε στις 25-4-2015)
14. http://www.tutorialspoint.com/java/java_arraylist_class.htm (Πάρθηκε 20-4-2015)
15. http://developer.android.com/reference/android/view/MotionEvent.html (Πάρθηκε στις 1-6-2015)
16. C Προγραμματισμός, Πέμπτη Έκδοση, Εκδόσεις: Μ. Γκιούρδας
17. Δομές Δεδομένων Αλγόριθμοι, και Εφαρμογές στη C++

Πνευματικά δικαιώματα