ΑΝΩΤΑΤΟ ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΚΡΗΤΗΣ

ΤΜΗΜΑ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΠΟΛΥΜΕΣΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Σπουδαστής : Μιρζογιάν Βαγκράμ

Εισηγητής : Μαλάμος Αθανάσιος

Θέμα: Σχεδίαση τρισδιάστατου εικονικού εξομοιωτή πτήσης σε περιβάλλον iPhone.

Ηράκλειο 2010

Η σελίδα αυτή είναι σκόπιμα λευκή.

Περιεχόμενα

1 ΕΙΣΑΓΩΓΗ

Ο σκοπός της πτυχιακής εργασίας είναι η δημιουργία ενός τρισδιάστατου παιχνιδιού τύπου Flight Simulator σε περιβάλλον iPhone με χρήση iPhoneSDK Xcode και μιας game – engine όπως SIO2, Unity 3D, Stone 3D κλπ. Τα γραφικά του παιχνιδιού θα είναι σχεδιασμένα σε κάποιο 3D πακέτο όπως 3ds max, Maya ή Blender και τα textures σε Photoshop.

Στο παιχνίδι αυτό ο χρήστης με τη βοήθεια των επιταχυνσιόμετρου και της multi-touch οθόνης της συσκευής θα πρέπει να μπορεί να κατευθύνει ένα αεροπλάνο μέσα σε μια εικονική πόλη. Στην πόλη θα υπάρχουν σκόρπια κάποια αντικείμενα. Σκοπός του παιχνιδιού θα είναι να συλλέξουμε αυτά τα αντικείμενα.

2 Το iPhone

2.1 Περιγραφή:

Το iPhone συγκαταλέγεται στην κατηγορία των smartphones και ενσωματώνει κινητό τηλέφωνο, φωτογραφική μηχανή και κάμερα, ασύρματη πρόσβαση στο διαδίκτυο μέσω wifi ή 3g δικτύου με δυνατότητα web browsing και email, αναπαραγωγή πολυμέσων και παιχνιδομηχανή σε μια συσκευή. Είναι κατασκεύασμα της εταιρίας Apple και πρωτοκυκλοφόρησε στις 29 Ιουνίου 2007. Η διεπαφή χρήστη βασίζεται κυρίως στην οθόνη πολυαφής 3.5 ιντσών που διαθέτει. Έχει ελάχιστο υλικό διεπαφής, μόνο 4 κουμπιά. Σαν συσκευή εισόδου χρησιμοποιείται επίσης το επιταχυνσιόμετρο τριών αξόνων το οποίο είναι πολύ χρήσιμο κυρίως σε παιχνίδια. Το iPhone εν γένει διαθέτει μόνο τις βασικές εφαρμογές που χρειάζεται ένα smartphone όπως εφαρμογή για την πραγματοποίηση κλήσεων, αποστολή – λήψη sms, φωτογραφική κάμερα, αναπαραγωγή πολυμέσων κτλ. Για το κατέβασμα εφαρμογών τρίτου κατασκευαστή η Apple έχει δημιουργήσει το app store το οποίο δίνει πρόσβαση σε πάνω από 300.000 εφαρμογές και παιχνίδια, μεγάλο μέρος των οποίων διατίθενται δωρεάν.

2.2 iPhone OS:

2.2.1 Εισαγωγή.

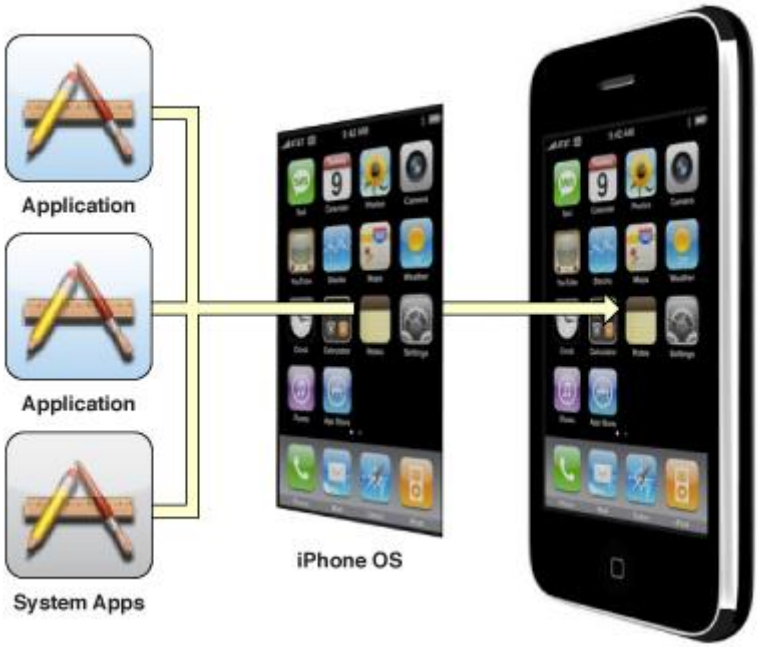
Η πλατφόρμα iPhone OS είναι η καρδιά των iPhone και iPod touch. Χτίστηκε με βάση το Mac OS X και πολλά από τα εργαλεία και τεχνολογίες που χρησιμοποιούνται για την ανάπτυξη εφαρμογών στην πλατφόρμα έχουν τις ρίζες τους εκεί. Παρά τις ομοιότητες του με το Mac OS X, δεν χρειάζεται να είναι κάποιος έμπειρος Mac OS X προγραμματιστής να γράψει εφαρμογές για το iPhone OS. Το iPhone Software Development Kit (SDK) προσφέρει όλα όσα χρειάζεται κανείς για να ξεκινήσει δημιουργία iPhone εφαρμογών.

iPhone OS είναι το λειτουργικό σύστημα που τρέχει στις iPhone και iPod touch συσκευές. Αυτό το λειτουργικό σύστημα διαχειρίζεται το υλικό της συσκευής και παρέχει τις βασικές τεχνολογίες που απαιτούνται για την υλοποίηση εγγενών εφαρμογών της συσκευής. Ανάλογα με το αν είναι εγκατεστημένο σε iPhone ή iPod touch, το λειτουργικό σύστημα είναι εμπλουτισμένο με διάφορες προεπιλεγμένες εφαρμογές του συστήματος, όπως τηλέφωνο, ταχυδρομείο, Safari κτλ που παρέχουν τις βασικές υπηρεσίες του συστήματος για τον χρήστη.

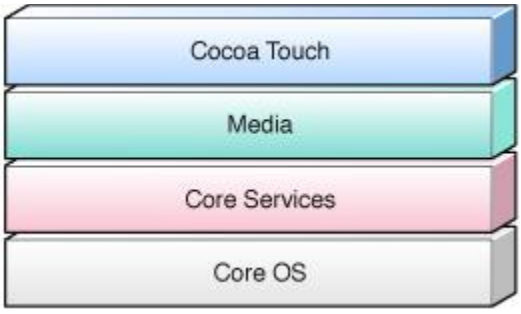
Το iPhone SDK περιλαμβάνει τα εργαλεία και τις διασυνδέσεις που απαιτούνται για την ανάπτυξη, εγκατάσταση, και την εκτέλεση μιας εφαρμογής.

2.2.2 Αρχιτεκτονική του iPhone OS

Η αρχιτεκτονική του iPhone OS είναι παρόμοια με την βασική αρχιτεκτονική του Mac OS X. Σε υψηλό επίπεδο, το iPhone OS ενεργεί ως μεσάζων μεταξύ του υλικού του iPhone και των εφαρμογών που εμφανίζονται στην οθόνη. Οι εφαρμογές που δημιουργούμε ποτέ δεν αλληλεπιδρούν άμεσα με το υλικό, αλλά αντ 'αυτού περνούν από τις διασυνδέσεις του συστήματος, οι οποίες αλληλεπιδρούν με τα κατάλληλα προγράμματα οδήγησης. Αυτός ο τρόπος διαχείρισης προστατεύει τις εφαρμογές από το να κάνουν αλλαγές στο υλικό της συσκευής.



Η υλοποίηση των iPhone OS τεχνολογιών μπορεί να θεωρηθεί ως ένα σύνολο στρωμάτων. Στα χαμηλότερα στρώματα του συστήματος είναι οι θεμελιώδεις υπηρεσίες στις οποίες στηρίζονται όλες οι εφαρμογές, ενώ τα υψηλότερου επιπέδου στρώματα περιέχουν πιο εξελιγμένες υπηρεσίες και τεχνολογίες.



Στον κώδικά μας προτείνεται η χρήση στρωμάτων υψηλότερου επιπέδου έναντι χαμηλότερου επιπέδου όποτε είναι δυνατόν. Τα πλαίσια υψηλότερου επιπέδου είναι για να μας παρέχουν αντικειμενοστραφείς κλήσεις για δομές κατώτερου επιπέδου. Αυτό συνήθως μας διευκολύνει στο να γράψουμε ευκολότερο επειδή μειώνει τον αριθμό των γραμμών κώδικα που χρειάζεται να γράψουμε για την προσθήκη πολύπλοκων χαρακτηριστικών όπως τα socket. Παρόλο όμως που μας αποκρύπτει τις χαμηλότερου επιπέδου δεν μας τις στερεί. Τα πλαίσια χαμηλότερου επιπέδου εξακολουθούν να είναι διαθέσιμα για όποιον θέλει να τα χρησιμοποιήσει.

2.2.3 iPhone SDK

Το iPhone SDK έρχεται με όλες τις διασυνδέσεις, τα εργαλεία και τους πόρους που απαιτούνται για την ανάπτυξη εφαρμογών για iPhone με χρήση ενός Macintosh υπολογιστή.

Η Apple προσφέρει τις περισσότερες διασυνδέσεις του συστήματος σε ειδικά πακέτα που ονομάζεται πλαίσια. Το πλαίσιο είναι ένας κατάλογος που περιέχει μια δυναμική κοινή βιβλιοθήκη και τους πόρους (όπως αρχεία κεφαλίδας, εικόνες κλπ) που απαιτούνται για την υποστήριξή της. Για να χρησιμοποιηθούν τα πλαίσια όπως και οποιαδήποτε άλλη κοινή βιβλιοθήκη πρέπει πρώτα να συνδεθούν με την εφαρμογή. Η σύνδεσή τους με την εφαρμογή δίνει πρόσβαση στις δυνατότητες του πλαισίου

Πέρα από τα πλαίσια, η Apple προσφέρει και ορισμένες τεχνολογίες στην μορφή των πρότυπων κοινόχρηστων βιβλιοθηκών. Επειδή το iPhone OS βασίζεται σε UNIX, πολλές από τις τεχνολογίες που αποτελούν το χαμηλότερο επίπεδο του λειτουργικού συστήματος προέρχονται από open-source τεχνολογίες.

Ορισμένα άλλα βασικά στοιχεία του SDK είναι:

- **Xcode Tools** - Παρέχει τα εργαλεία για την ανάπτυξη εφαρμογών για iPhone συμπεριλαμβανομένων των ακόλουθων βασικών εφαρμογών:

    - **Xcode** – Ένα ολοκληρωμένο περιβάλλον ανάπτυξης που διαχειρίζεται τα Project των εφαρμογών. Περιέχει εργαλεία για την επεξεργασία, μεταγλώττιση, εκτέλεση, και τον εντοπισμό σφαλμάτων στον κώδικά. Το Xcode ενσωματώνει και πολλά άλλα εργαλεία και είναι η κύρια εφαρμογή που χρησιμοποιείται κατά τη διάρκεια της ανάπτυξης.

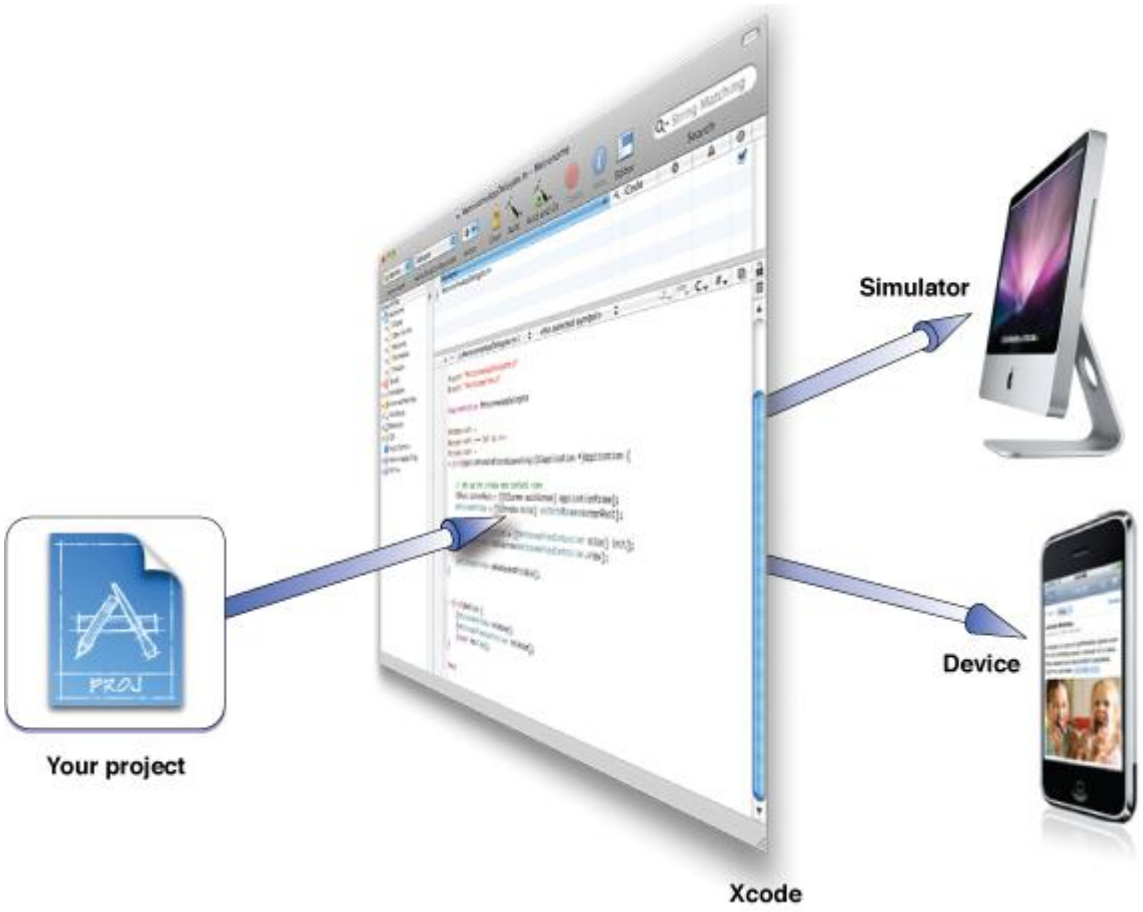
    - **Interface Builder** - Εργαλείο για την δημιουργία διεπαφών χρήστη οπτικά (με χρήση drag&drop). Τα αντικείμενα διεπαφών που δημιουργούνται στη συνέχεια αποθηκεύονται σε μια ειδική μορφή

αρχείου και φορτώνονται στην εφαρμογή την ώρα της εκτέλεσης.

- **Instruments** – Εργαλείο για την ανάλυση επιδόσεων και τον εντοπισμό σφαλμάτων κατά την εκτέλεση της εφαρμογής. Μπορείτε να χρησιμοποιήσετε μέσα για να συλλέξουν πληροφορίες για τη συμπεριφορά χρόνου εκτέλεσης της εφαρμογής σας και να εντοπίσει πιθανά προβλήματα.

- **iPhone Simulator** – Εφαρμογή σε περιβάλλον Mac OS X που προσομοιώνει το iPhone επιτρέποντάς να δοκιμάσουμε το iPhone εφαρμογές σε τοπικό επίπεδο.

- **iPhone Reference Library** - Τεκμηρίωση αναφοράς για το iPhone OS που συμπεριλαμβάνεται στο SDK. Η βιβλιοθήκη ενημερώνεται αυτόματα όταν διαθέσιμα.



Πέρα από το γεγονός ότι το SDK παρέχει το λογισμικό που χρειάζεται για να γράψουμε τις εφαρμογές, το Xcode και τα Instruments μας δίνουν τη δυνατότητα άμεσης αλληλεπίδρασης με μια συνδεδεμένη συσκευή για την εκτέλεση και διόρθωση του κώδικα σε πραγματικές. Η ανάπτυξη εφαρμογών σε πραγματική

συσκευή απαιτεί εγγραφή στο Apple Paid iPhone Developer Program, καταβολή 100$ ετησίως και ειδικές ρυθμίσεις του Mac και της συσκευής.
Περισσότερες πληροφορίες για τις απαραίτητες διαδικασίες ώστε να ξεκινήσει κανείς την ανάπτυξη εφαρμογών σε iPhone υπάρχουν στην παρακάτω σελίδα.
iPhone Development Guide.

2.3 Στρώματα του iPhone OS.

Παρακάτω θα παραθέσω τα βασικά στοιχεία του κάθε στρώματος αρχίζοντας από τα υψηλότερα προς τα χαμηλότερα.

2.3.1 Στρώμα Cocoa touch

Το στρώμα Cocoa Touch περιέχει τα βασικά πλαίσια για τη δημιουργία εφαρμογών iPhone OS. Οι τεχνολογίες σε αυτό το στρώμα παρέχουν την υποδομή που χρειάζεται για την υλοποίηση της διεπαφής χρήστη της εφαρμογής και την αλληλεπίδραση με πολλές υπηρεσίες του συστήματος υψηλού επιπέδου. Κατά την ανάπτυξη εφαρμογών προτείνεται πάντα να αρχίζουμε με αυτά τα πλαίσια και να μεταβαίνουμε στα χαμηλότερα επίπεδα μόνο όταν χρειάζεται.

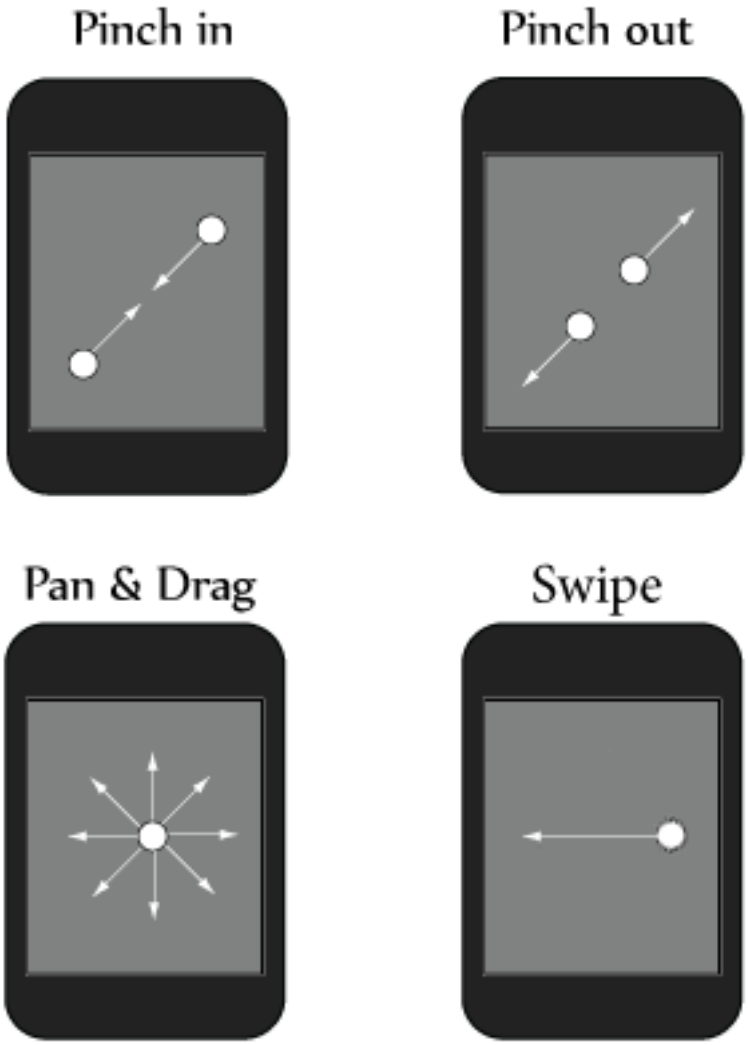
Παρακάτω φαίνονται κάποια από τα βασικότερα χαρακτηριστικά του στρώματος Cocoa Touch.

Αναγνώριση χειρονομιών.

Η αναγνώριση χειρονομίας είναι αντικείμενα που μπορούμε να προσθέσουμε σε μια εφαρμογή και να αναγνωρίσουν κάποιες κοινές μορφές χειρονομίας. Αφού το προσθέσουμε πρέπει να δηλώσουμε ποια μέθοδος θα καλείται όταν εντοπίσει κάποια χειρονομία. Όταν αναγνωρίσει κάποια χειρονομία καλείται η μέθοδος που έχουμε δηλώσει με όρισμα ένα αντικείμενο τύπου UIGestureRecognizer το οποίο περιέχει πληροφορίες σχετικά με το είδος της χειρονομίας, τις συντεταγμένες του σημείου στο οποίο συνέβη και άλλες πληροφορίες.

Τα είδη των χειρονομιών που αναγνωρίζει εξ ορισμού το iPhone OS είναι τα παρακάτω.

- Απλό πάτημα(οποιοσδήποτε αριθμός πατημάτων)
- Pinch in and out (για ζουμ)
- Panning or dragging
- Swiping(για αλλαγή σελίδας, προβολής κλπ)
- Περιστροφή (2 δάκτυλα κινούνται προς αντίθετες κατευθύνσεις)
- Μεγάλης διάρκειας πάτημα



Ωστόσο αν δεν μας αρκούν οι προεπιλεγμένες χειρονομίες μπορούμε να δημιουργήσουμε και δικές μας.

Υπηρεσίες Peer to Peer

Η peer-to-peer συνδεσιμότητα μέσω Bluetooth παρέχεται από το πλαίσιο Game Kit. Η συνδεσιμότητα αυτή αναλαμβάνει να δημιουργήσει και να αρχικοποιήσει περιόδους σύνδεσης με κοντινές συσκευές και να προκειμένου να ενεργοποιήσει τα multiplayer χαρακτηριστικά ενός παιχνιδιού, ή και κάποιας εφαρμογής.

Βασικά View Controller συστήματος

Πολλά από τα πλαίσια του στρώματος Cocoa Touch περιλαμβάνουν view controllers για την παρουσίαση των βασικών διεπαφών του συστήματος. Για παράδειγμα αν χρειαστεί να βάλουμε μια φωτογραφία στην εφαρμογή η οποία θα είναι είτε από τη φωτογραφική της συσκευής, είτε από τη βιβλιοθήκη φωτογραφιών πρέπει να δημιουργήσουμε ένα UIImagePickerController. Η Apple ενθαρρύνει τη χρήση τους

όποτε είναι δυνατόν. Οι ελεγκτές που μπορούμε να χρησιμοποιήσουμε είναι οι παρακάτω:

- Εμφάνιση ή επεξεργασία πληροφοριών επαφής - Address Book UI framework
- Δημιουργία ή να επεξεργασία γεγονότων στο ημερολόγιο - Event Kit UI framework
- Σύνθεση ενός μηνύματος e-mail ή SMS - Message UI framework
- Άνοιγμα ή προεπισκόπηση τα περιεχόμενα ενός αρχείου - UIDocumentInteractionController UIKit framework.
- Τράβηγμα μιας φωτογραφίας ή επιλογή μιας φωτογραφίας από τη βιβλιοθήκη φωτογραφιών του χρήστη - UIImagePickerController UIKit framework.
- Τράβηγμα ενός βίντεο κλιπ - UIImagePickerController UIKit framework.

Πλαίσιο Game Kit

Το πλαίσιο Game Kit επιτρέπει να προσθήκη peer-to-peer δυνατοτήτων. Παρόλο που οι δυνατότητες αυτές χρησιμοποιούνται πιο συχνά σε multiplayer παιχνίδια, μπορούν να ενσωματωθούν και σε εφαρμογές. Το πλαίσιο που παρέχει χαρακτηριστικά δικτύωσης μέσα από απλές, αλλά πολύ ισχυρές κλάσεις που είναι κτισμένες πάνω στο Bonjour. Αυτές οι κλάσεις αποκρύπτουν πολλές λεπτομέρειες σχετικά με το δίκτυο καθιστώντας έτσι εύκολο για προγραμματιστές άπειρους σε προγραμματισμό δικτύου να τις προσθέσουν στις εφαρμογές τους.

Πλαίσιο iAd

Δημιουργήθηκε στην τελευταία έκδοση του iPhone OS (iOS 4.0) και επιτρέπει την δημιουργία και προσθήκη διαφημιστικών banner στις εφαρμογές του iPhone. Οι διαφημίσεις είναι ενσωματωμένες στις βασικές προβολές και μπορούν να εμφανιστούν όποτε το θελήσουμε.

Πλαίσιο Map Kit

Το πλαίσιο Map Kit παρέχει ένα περιβάλλον χαρτών που μπορούν να ενσωματωθούν σε μια εφαρμογή. Το Map Kit κληρονομεί τα χαρακτηριστικά του από την εφαρμογή Χάρτες(maps) του iPhone επιτρέποντας όμως την προσθήκη δικών μας σημειώσεων και πληροφοριών. Μπορούμε πολύ εύκολα να προσθέσουμε μια τέτοια προβολή

στην εφαρμογή μας με λίγες γραμμές κώδικα. Αφού το αρχικοποιήσουμε πρέπει να ορίσουμε τα βασικά χαρακτηριστικά του όπως την περιοχή που θα εμφανίσει όταν το ανοίξει ο χρήστης το ζουμ κλπ.

Πλαίσιο UIKit

Το πλαίσιο UIKit (UIKit.framework) περιέχει διεπαφές προγραμματισμού της Objective-C που παρέχουν τη βασική υποδομή για την δημιουργία γραφικών και event-driven εφαρμογών στο iPhone OS. Κάθε εφαρμογή στο iPhone OS χρησιμοποιεί αυτό το πλαίσιο για να εφαρμόσει το σύνολο των χαρακτηριστικών του πυρήνα του. Αυτά είναι:

- Διαχείριση εφαρμογών
- Διαχείριση διεπαφών χρήστη
- Υποστήριξη γραφικών και παραθύρων
- Υποστήριξη Multitasking
- Υποστήριξη για το χειρισμό αφής και γεγονότων κίνησης
- Αντικείμενα που εκπροσωπούν βασικές προβολές και χειρισμούς
- Υποστήριξη για κείμενο και Web περιεχόμενο
- Υποστήριξη αποκοπής, αντιγραφής και επικόλλησης
- Υποστήριξη για animation περιεχομένων διεπαφών χρήστη
- Ενσωμάτωση εφαρμογών συστήματος μέσω συνδέσμων URL
- Υποστήριξη για την υπηρεσία push της Apple
- Υποστήριξη προσβασιμότητας για άτομα με ειδικές ανάγκες
- Τοπικό χρονοπρογραμματισμό ειδοποιήσεων
- Δημιουργία αρχείων PDF
- Δημιουργία προσαρμοσμένων text view που αλληλεπιδρούν με το πληκτρολόγιο του συστήματος

Εκτός από την παροχή του βασικού κώδικα για την δημιουργία εφαρμογών, το UIKit ενσωματώνει και την υποστήριξη για κάποια χαρακτηριστικά εξαρτώμενα από τη συσκευή όπως:

- Δεδομένα Επιταχυνσιόμετρου
- Ενσωματωμένη κάμερα (εφόσον υπάρχει)
- Βιβλιοθήκη φωτογραφιών του χρήστη
- Όνομα συσκευής και το μοντέλο
- Πληροφορίες για την κατάσταση της μπαταρίες
- Πληροφορίες αισθητήρα φωτός
- Πληροφορίες τηλεχειρισμού από συνδεδεμένα ακουστικά

2.3.2 Στρώμα Media

Το στρώμα Media περιέχει τις τεχνολογίες που σχετίζονται με τα γραφικά, τον ήχο και το βίντεο και προορίζεται στην δημιουργία της καλύτερων εμπειρίας πολυμέσων σε μια φορητή συσκευή. Αυτές οι τεχνολογίες έχουν σχεδιαστεί για να κάνουν ευκολότερη την δημιουργία εφαρμογών με τέλεια εικόνα και ήχο. Τα πλαίσια υψηλού επιπέδου καθιστούν εύκολη την δημιουργία προηγμένων γραφικών και animations γρήγορα, και τα πλαίσια χαμηλού επιπέδου μας παρέχουν πρόσβαση στα εργαλεία που χρειαζόμαστε για να κάνουμε τα πράγματα ακριβώς όπως τα θέλουμε.

Τεχνολογίες γραφικών

Τα γραφικά υψηλής ποιότητας είναι σημαντικό μέρος των εφαρμογών iPhone. Ο απλούστερος για να φτιάξουμε μια εφαρμογή είναι να χρησιμοποιήσουμε εικόνες σε συνδυασμό με βασικές προβολές και στοιχεία ελέγχου του πλαισίου UIKit αφήνοντας έτσι το σύστημα να κάνει όλη τη δουλειά. Ωστόσο υπάρχουν φορές που θέλουμε να κάνουμε κάτι πέρα από αυτά που προσφέρει το UIKit. Σ'αυτές τις περιπτώσεις μπορούμε να χρησιμοποιήσουμε τις παρακάτω τεχνολογίες για να διαχειριστούμε τα γραφικά περιεχόμενα της εφαρμογής μας:

- Core Graphics(Quartz) – διαχειρίζεται τα
- Core Animation(μέρος του Quartz Core) – Προσφέρει υποστήριξη για δημιουργία κίνησης σε προβολές και άλλα περιεχόμενα.
- OpenGL ES – προσφέρει υποστήριξη για 2D και 3D render με επιτάχυνση υλικού
- Core Text – προσφέρει ένα εξελιγμένο στρώμα κειμένου και μηχανή rendering
- Image I/O – προσφέρει διεπαφές για ανάγνωση και εγγραφή σχεδόν σε όλους τους τύπους εικόνων.
- Asset Library – επιτρέπει την πρόσβαση στις φωτογραφίες και τα βίντεο της βιβλιοθήκης του χρήστη

Τεχνολογίες ήχου

Οι τεχνολογίες ήχου του iPhone OS είναι σχεδιασμένα για να μας βοηθήσουν να παρέχουμε πλούσια εμπειρία ήχου στους χρήστες. Μας δίνουν τη δυνατότητα να αναπαράγουμε και να ηχογραφήσουμε υψηλής ποιότητας ήχο.

Το σύστημα μα προσφέρει διαφορετικούς τρόπου για αναπαραγωγή και εγγραφή περιεχόμενων ήχου ανάλογα με τις ανάγκες μας. Όταν επιλέγουμε μια τεχνολογία

πρέπει να λάβουμε υπόψη ότι τα πλαίσια υψηλών επιπέδων απλουστεύουν την διαδικασία ενώ τα χαμηλότερα επίπεδα μας δίνουν περισσότερες επιλογές. Παρακάτω παρουσιάζονται όλα τα πλαίσια αρχίζοντας από τα υψηλότερα προς τα χαμηλότερα επίπεδα.

- Πλαίσιο Media Player – προσφέρει εύκολη πρόσβαση στην μουσική βιβλιοθήκη του χρήστη και υποστηρίζει την αναπαραγωγή κομματιών και λιστών
- AV Foundation – προσφέρει ένα σύνολο διεπαφών Obj - C εύκολων στην χρήση για αναπαραγωγή και εγγραφή ήχου.
- OpenAL – προσφέρει ένα σύνολο διεπαφών ανεξαρτήτου πλατφόρμας για αναπαραγωγή ήχου stereo ή 3D
- Πλαίσιο Core Audio – προσφέρει απλές, αλλά και εξελιγμένες διεπαφές για αναπαραγωγή και εγγραφή ήχου. Οι διεπαφές αυτές παρέχουν και υποστήριξη για buffering, αναπαραγωγή πολυκάναλου τοπικού ή streamed ήχου.

Οι υποστηριζόμενοι τύποι αρχείων ήχου είναι οι παρακάτω:

- AAC
- Apple Lossless (ALAC)
- A-law
- IMA/ADPCM (IMA4)
- Linear PCM
- μ-law
- DVI/Intel IMA ADPCM
- Microsoft GSM 6.10
- AES3-2003

Τεχνολογίες βίντεο

Το iPhone OS μας παρέχει μερικές τεχνολογίες για αναπαραγωγή βίντεο είτε τοπικά είτε streamed από το διαδίκτυο. Σε συσκευές που το υποστηρίζουν το iOS επιτρέπει και την εγγραφή βίντεο και την ενσωμάτωσή του σε εφαρμογές.

Όπως με τις τεχνολογίες ήχου έτσι και εδώ τα υψηλότερα στρώματα απλουστεύουν τη διαδικασία ενώ τα χαμηλότερα στρώματα μας δίνουν περισσότερες επιλογές. Παρακάτω παρουσιάζονται όλα τα πλαίσια αρχίζοντας από τα υψηλότερα προς τα χαμηλότερα επίπεδα.

- Πλαίσιο Media Player – προσφέρει ένα σύνολο απλών στην χρήση διεπαφών για αναπαραγωγή βίντεο πλήρους οθόνης ή μη μέσα από μια εφαρμογή.

- AV Foundation – προσφέρει ένα σύνολο διεπαφών Obj – C για διαχείριση της εγγραφής και αναπαραγωγής ταινιών.
- Core Media – προσφέρει τις διεπαφές χαμηλού επιπέδου που χρησιμοποιούνται από τα πλαίσια υψηλού επιπέδου για την διαχείριση μέσων.

Οι τεχνολογίες βίντεο του iPhone OS υποστηρίζουν την αναπαραγωγή αρχείων βίντεο με κατάληξη .mov, .mp4, .m4v και .3gp που χρησιμοποιούν μια από τις παρακάτω συμπιέσεις:

- H.264 video, up to 1.5 Mbps, 640 by 480 pixels, 30 frames per second, Low-Complexity version of the H.264 Baseline Profile with AAC-LC audio up to 160 Kbps, 48kHz, stereo audio in `.m4v`, `.mp4`, and `.mov` file formats
- H.264 video, up to 768 Kbps, 320 by 240 pixels, 30 frames per second, Baseline Profile up to Level 1.3 with AAC-LC audio up to 160 Kbps, 48kHz, stereo audio in `.m4v`, `.mp4`, and `.mov` file formats
- MPEG-4 video, up to 2.5 Mbps, 640 by 480 pixels, 30 frames per second, Simple Profile with AAC-LC audio up to 160 Kbps, 48kHz, stereo audio in `.m4v`, `.mp4`, and `.mov` file formats

2.3.3 Στρώμα Core Services

Το στρώμα Core Services παρέχει τις θεμελιώδεις υπηρεσίες συστήματος που χρησιμοποιούνται από όλες τις εφαρμογές. Ακόμα και αν δεν τις χρησιμοποιούμε άμεσα πολλά μέρη του συστήματος είναι βασισμένα σ'αυτές.

Χαρακτηριστικά υψηλού επιπέδου.

In App Purchase

Το In App Purchase μας δίνει τη δυνατότητα να πουλήσουμε πρόσθετο περιεχόμενο και υπηρεσίες μέσα από την εφαρμογή μας. Για την υλοποίηση αυτής της δυνατότητας χρησιμοποιείται το πλαίσιο Store Kit, το οποίο παρέχει την απαραίτητη υποδομή για την πραγματοποίηση οικονομικών συναλλαγών μέσω του λογαριασμού του χρήστη στο iTunes.

Location Services

Χρησιμοποιώντας το πλαίσιο Core Location μας δίνει τη γεωγραφική θέση του χρήστη. Για να βρει τη γεωγραφική θέση του χρήστη χρησιμοποιεί όλες τις διαθέσιμες κεραίες της συσκευής(όπως Wi-Fi, GSM και GPS).

SQLite

Επιτρέπει την ενσωμάτωση μιας SQL βάσης δεδομένων στην εφαρμογή μας. Μέσα από την εφαρμογή μπορούμε να δημιουργήσουμε τοπικά αρχεία βάσης δεδομένων και να διαχειριστούμε τους πίνακες και τις εγγραφές σ'αυτά.

Υποστήριξη XML

Το πλαίσιο Foundation μας προσφέρει την κλάση NSXMLParser για ανάκτηση στοιχείων από ένα έγγραφο XML. Περισσότερη υποστήριξη για την διαχείριση XML περιεχομένου παρέχεται από την ανοικτού κώδικα βιβλιοθήκη libXML2.

Πλαίσια Core Services

Πλαίσιο Address Book

Προσφέρει πρόσβαση στις επαφές του βιβλίου διευθύνσεων της συσκευής μέσω κώδικα. Μέσω του πλαισίου αυτού ο χρήστης μπορεί να δει ή και να τροποποιήσει εγγραφές από τη βάση δεδομένων των επαφών.

Πλαίσιο CFNetwork

Το πλαίσιο CFNetwork είναι ένα σύνολο διεπαφών υψηλής απόδοσης για χρήση πρωτοκόλλων δικτύου στην εφαρμογή μας. Μας δίνει λεπτομερή έλεγχο στην στοίβα πρωτοκόλλων και κάνουν εύκολη τη χρήση των δομών χαμηλού επιπέδου όπως τα BSD Sockets. Διευκολύνουν επίσης την επικοινωνία με εξυπηρετητές FTP και HTTP. Παρακάτω φαίνονται κάποιες εργασίες που μπορούν να πραγματοποιηθούν με χρήση του πλαισίου CFNetwork.

- Χρήση BSD Sockets
- Δημιουργία κρυπτογραφημένης σύνδεσης με χρήση SSL ή TLS
- Επίλυση διευθύνσεων DNS
- Εργασία με HTTP, εξυπηρετητές HTTPS και επικύρωση HTTP
- Εργασία με εξυπηρετητές FTP
- Υπηρεσίες Bonjour

Πλαίσιο Core Foundation

Το πλαίσιο Core Foundation είναι ένα σύνολο διεπαφών σε C που παρέχουν υπηρεσίες διαχείρισης δεδομένων στις εφαρμογές. Υποστηρίζονται τα παρακάτω:

- Τύποι δεδομένων συνόλων(arrays, sets κλπ)
- Bundles
- Διαχείριση συμβολοσειρών
- Διαχείριση raw δεδομένων
- Χειρισμός URL και Stream
- Threads και run loops
- Επικοινωνία Socket και Port

Πλαίσιο Event Kit

Προσφέρει μια διεπαφή για πρόσβαση σε γεγονότα του ημερολογίου της συσκευής. Μπορεί να χρησιμοποιηθεί για προσθήκη ενός γεγονότος ή για ανάκτηση ενός υπάρχοντος.

Στρώμα Core OS

Το στρώμα Core OS παρέχει πρόσβαση στα χαρακτηριστικά χαμηλού επιπέδου στα οποία είναι βασισμένες οι περισσότερες τεχνολογίες υψηλότερων στρωμάτων. Σε περιπτώσεις που χρειάζεται να δοθεί περισσότερη βάση στην ασφάλεια ή χρειάζεται να επικοινωνήσουμε με μια εξωτερική συσκευή πρέπει να χρησιμοποιήσουμε τις τεχνολογίες αυτές.

Πλαίσιο External Accessory

Το πλαίσιο External Accessory παρέχει υποστήριξη για επικοινωνία με εξωτερικές συσκευές συνδεδεμένες στην συσκευή μας. Οι εξωτερικές συσκευές μπορούν να συνδεθούν μέσω της 30-pin θήρας ή ασύρματα μέσω Bluetooth. Αφού δημιουργήσουμε μια σύνδεση και την αρχικοποιήσουμε μπορούμε να επικοινωνήσουμε με την εξωτερική συσκευή στέλνοντας της εντολές που υποστηρίζει.

Πλαίσιο Security

Πέρα από τα ενσωματωμένα χαρακτηριστικά ασφάλειας του iPhone OS, το πλαίσιο Security μας προσφέρει διεπαφές που διασφαλίζουν την ασφάλεια των δεδομένων στις εφαρμογές. Το πλαίσιο προσφέρει διεπαφές για την διαχείριση πιστοποιητικών , δημόσιων και ιδιωτικών κλειδιών και πολιτικών εμπιστοσύνης. Υποστηρίζει και την δημιουργία ασφαλών ψευδό -τυχαίων αριθμών.

Για την πτυχιακή εργασία θα χρησιμοποιηθεί το iPhone 3G

2.4 Τεχνικά χαρακτηριστικά του iPhone 3G:

Αποθηκευτικός χώρος: 8, 16 ή 32 GB μνήμης Flash για την αποθήκευση αρχείων και το λειτουργικό σύστημα.
Μέγεθος οθόνης: 8.9 cm (Διαγώνιος: 3.5 ίντσες)
Ανάλυση οθόνης: 320×480 pixels
Μέγεθος τηλεφώνου: 115×61×11.6 mm
Βάρος τηλεφώνου: 135 g
Λειτουργικό σύστημα: iPhone OS
Λογισμικό πλοήγησης στο Διαδίκτυο Safari
Λογισμικό ηλεκτρονικού ταχυδρομείου Push
Μηχανή widgets
Core Animation
Διαχείριση ενέργειας
Θύρα συνδέσεων USB
Δυνατότητα αναπαραγωγής ήχων: Αναπαράγει μορφές αρχείων AAC (16 έως 320 Kbps), Protected AAC (MP4 από τα iTunes), MP3 (16 έως 320 Kbps), MP3 VBR, Audible (formats 2, 3, και 4), Apple Lossless, formats αρχείων AIFF και WAV
Δυνατότητα αναπαραγωγής βίντεο: Αναπαράγει βίντεο σε μορφή H.264
Quad band GSM / GPRS / EDGE: GSM 850, GSM 900, GSM 1800, GSM 1900
Ενσωματωμένα Wi-Fi (802.11b/802.11g) και Bluetooth 2.0 (χωρίς FTP μεταφορά αρχείων) με EDR
Κάμερα 2 megapixel
IPod - εφαρμογές πολυμέσων
Cover Flow διεπαφή και τρισδιάστατα γραφικά
Multi-touch διεπαφή οθόνης

Εικονικό πληκτρολόγιο QWERTY στην οθόνη αφής
Λειτουργίες: Σύρσιμο (κύλιση) οθόνης με το δάκτυλο και μεγέθυνση/σμίκρυνση με κτύπο ή διπλό κτύπο δακτύλου
Ενσωματωμένη, μη αποσπώμενη μπαταρία με αυτονομία μέχρι 5 ώρες ομιλίας/βίντεο/πλοήγησης ή μέχρι 16 ώρες αναπαραγωγής ήχου.
Ενσωματωμένος ανιχνευτής κίνησης για αυτόματη "έξυπνη" εναλλαγή οριζόντιου/κατακόρυφου προσανατολισμού οθόνης.
Ενσωματωμένος αισθητήρας προσέγγισης που απενεργοποιεί την οθόνη όταν πλησιάζει στο πρόσωπο του χρήστη, ώστε να εξοικονομείται ενέργεια και να αποφεύγεται ανεπιθύμητη εντολή εισόδου.

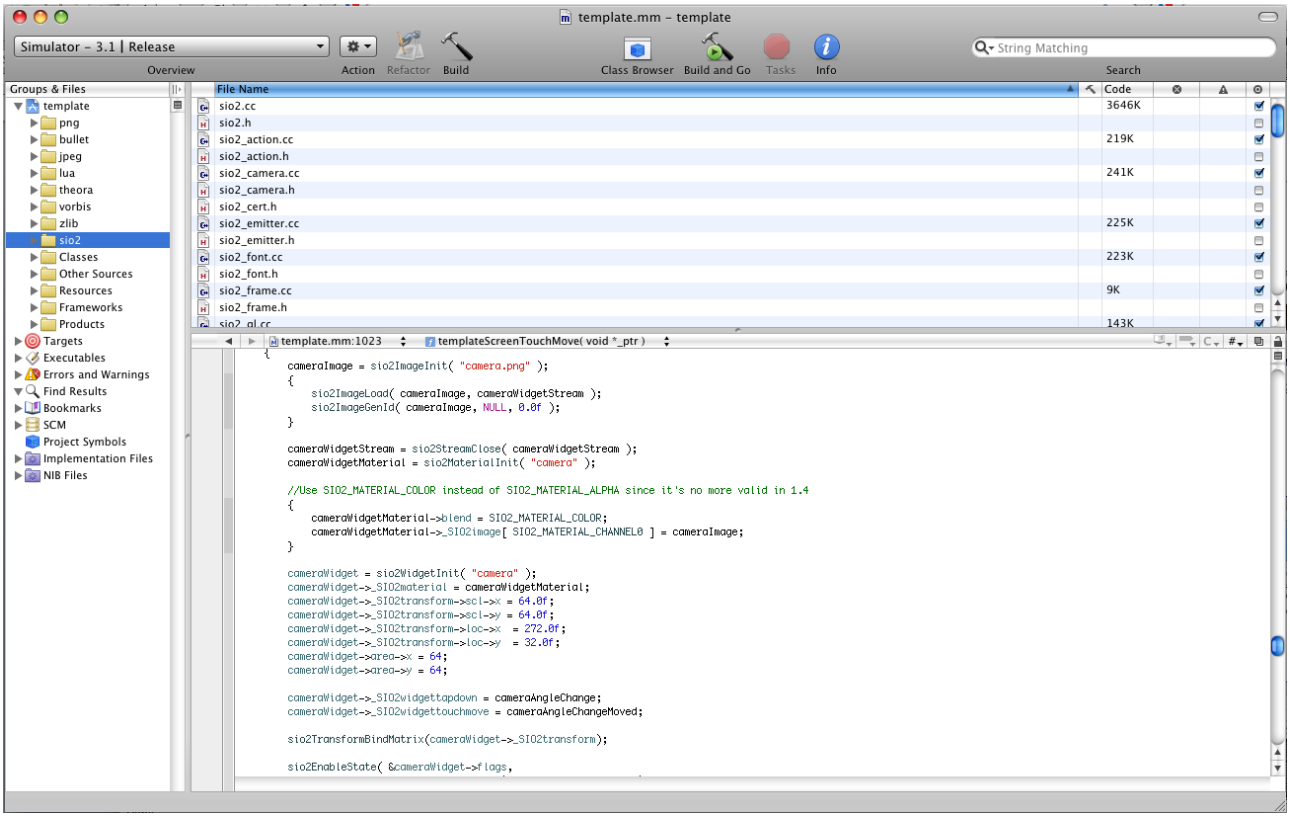
3 ΕΡΓΑΛΕΙΑ

3.1 Xcode

Το Xcode είναι μια σουίτα εργαλείων για την ανάπτυξη λογισμικού για Mac OS X, που αναπτύχθηκε από την Apple. Xcode 3.2, η τελευταία μεγάλη έκδοση, διατίθεται δωρεάν με το λειτουργικό σύστημα Mac OS X v10.6, αλλά δεν εγκαθίσταται από προεπιλογή. Επειδή η έκδοση 3.2 δεν υποστηρίζεται σε παλαιότερες εκδόσεις Mac OS, παλιότερες εκδόσεις του Xcode διατίθενται δωρεάν από την ιστοσελίδα της Apple Developer Connection.

Η κύρια εφαρμογή της σουίτας είναι το ολοκληρωμένο περιβάλλον ανάπτυξης (IDE), που ονομάζονται επίσης Xcode. Η σουίτα Xcode περιλαμβάνει επίσης τα περισσότερα από τα βοηθήματα προγραμματιστών της Apple, και το Interface Builder, μια εφαρμογή που χρησιμοποιείται για την κατασκευή γραφικών διεπαφών χρήστη.

Η σουίτα Xcode περιλαμβάνει μια τροποποιημένη έκδοση του ελεύθερου λογισμικού GNU Compiler Collection (GCC, apple-darwin9-GCC-4.2.1, καθώς και apple-darwin9-GCC-4.0.1), και υποστηρίζει C, C++, Fortran, Objective-C, Objective-C++, Java, AppleScript, Python και Ruby πηγαίου κώδικα με μια ποικιλία μοντέλων προγραμματισμού, που περιλαμβάνουν αλλά δεν περιορίζονται σε Cocoa, Carbon, and Java.



Εικόνα 3.1.1: Διεπαφή χρήστη Xcode 3.1.4

3.2 Μηχανή παιχνιδιού – Game Engine

3.2.1 Περιγραφή:

Μια μηχανή παιχνιδιού είναι ένα σύστημα λογισμικού σχεδιασμένο για τη δημιουργία και την ανάπτυξη βιντεοπαιχνιδιών. Υπάρχουν πολλές μηχανές παιχνιδιών οι οποίες είναι σχεδιασμένες να δουλεύουν σε κονσόλες βιντεοπαιχνιδιών και λειτουργικά συστήματα επιτραπέζιων υπολογιστών όπως τα Microsoft Windows, το Linux, και το Mac OS X. Η κεντρική λειτουργικότητα που παρέχεται τυπικά από μια μηχανή παιχνιδιού περιλαμβάνει μια μηχανή rendering ("renderer") για 2D ή 3D γραφικά, μια μηχανή φυσικής ή collision detection (και collision response), ήχο, scripting, animation, τεχνητή νοημοσύνη, δικτύωση, streaming, διαχείριση μνήμης, threading, υποστήριξη τοπικοποίησης, και scene graph. Η διαδικασία της ανάπτυξης παιχνιδιού συχνά οικονομικοποιείται με το ότι σε μεγάλο μέρος η ίδια μηχανή παιχνιδιού ξαναχρησιμοποιείται για να δημιουργηθούν διαφορετικά παιχνίδια.

3.2.2 Δημοφιλείς μηχανές παιχνιδιού:

Οι δημοφιλέστερες μηχανές παιχνιδιού για διάφορες πλατφόρμες είναι:

No.1: Unreal Engine 3	PC, Mac, Xbox 360, PS3
No.2: Gamebryo Lightspeed	Xbox 360, PS3, Wii, PC
No.3: CryENGINE 3	Xbox 360, PS3, PC
No.4: Unity 3D	PC, Mac, **iPhone**, Wii
No.5: BlitzTech	PS3, Xbox 360, Wii, PSP, PC
No.6: Infernal Engine	Xbox 360, PS3, PC, Wii, PS2, PSP
No.7: Vision Engine 7.5	PC (DX9 & 10), Xbox 360, PlayStation 3, Wii
No.8: Bigworld Technology Suite	MMO games
No.9: Vicious Engine 2	PC, Xbox 360, PS3 (VE2); PSP, PS2, Wii
No.10: Torque 3D	PC, Mac, Xbox 360, Wii, **iPhone**, PS3, PSP

Οι δημοφιλέστερες μηχανές παιχνιδιού για iPhone είναι:

Bork 3D - Είναι μια μηχανή ειδικά για προγραμματιστές και δεν περιέχει οπτικά εργαλεία τύπου WYSIWYG (***W**hat **Y**ou **S**ee **I**s **W**hat **Y**ou **G**et).*
Η μηχανή Bork3D χτίστηκε ειδικά για πλατφόρμες κινητών συσκευών. Έχει τις ρίζες στο Rude Engine, μια υψηλής απόδοσης βιβλιοθήκη γραφικών για το Pocket PC, Symbian και N-Gage. Οι επιδόσεις και επεκτασιμότητα είναι από τα βασικότερα χαρακτηριστικά της. Παρόλα αυτά δεν είναι μια πλήρης μηχανή παιχνιδιού με την έννοια ότι δεν περιέχει όλα τα απαραίτητα εργαλεία για την δημιουργία ενός παιχνιδιού όπως, εργαλεία για δημιουργία περιεχομένων, scripting κλπ. Απευθύνεται κυρίως σε άτομα με εμπειρία στον προγραμματισμό και το κόστος του ξεκινά από $ 49.

Cocos 2D iPhone - Το Cocos 2D iPhone είναι ένα εκτεταμένο engine ανοικτού κώδικα για την παραγωγή 2D παιχνιδιών για το iPhone. Είναι βασισμένο στην αρχιτεκτονική του Cocos 2D για python. Είναι εύκολο στην εκμάθηση και στην χρήση.
Μερικά από τα κύρια χαρακτηριστικά της είναι:

- Διαχείριση σκηνών
- Sprites και spritesheets
- Οπτικά εφέ όπως Lens, Ripple, Waves, Liquid και άλλα
- Βασικά μενού και κουμπιά
- 2 ολοκληρωμένες μηχανές φυσικής(Box2d και Chipmunk)
- Σύστημα σωματιδίων.(Particle system)
- Υποστήριξη Tile Map και Texture Atlas

Το Cocos 2D έχει χρησιμοποιηθεί ως σημείο εκκίνησης για πολλούς προγραμματιστές που

ενδιαφέρονται για τη δημιουργία παιχνιδιών σε iPhone.

iTGB (Torque Game Builder) – Είναι μια 2D μηχανή με ένα εξαιρετικά εύκολο στη χρήση scene editor. Πρόκειται για μια 2D μηχανή με πολύ πλούσιο σύνολο λειτουργιών. Αν εξαιρέσουμε το κόστος το οποίο αρχίζει από $ 750 το iTGB είναι ένας εξαιρετικός τρόπος για να ξεκινήσει κανείς με την ανάπτυξη παιχνιδιών για το iPhone.

iTGE (Torque Game Engine) - Είναι μια 3D μηχανή από τους δημιουργούς των iTGB. Η Torque Game Engine έχει χρησιμοποιηθεί σε πολλούς τίτλους παιχνιδιών για την πλατφόρμα των Windows. Το κόστος ξεκινά από $ 650.

Oolong engine - Oolong είναι μια μηχανή παιχνιδιού γραμμένη από το συντάκτη/προγραμματιστή Wolfgang Engel. Περιέχει ένα εκτεταμένο σύνολο χαρακτηριστικών και μια πολύ φιλελεύθερη άδεια χρήσης MIT. Ωστόσο είναι σίγουρα μια μηχανή που δημιουργήθηκε περισσότερο για προγραμματιστές, και απαιτεί βαθιά γνώση του OpenGL ES να χρησιμοποιηθεί. Υπάρχει μια λίστα συζητήσεων, καθώς και μια ωραία συλλογή από παραδείγματα. Ένα σοβαρό μειονέκτημα του είναι η έλλειψη τεκμηρίωσης.

Shiva Ston3d - Shiva είναι μια μηχανή παιχνιδιού με editor για σχεδιασμό σκηνών. Χρησιμοποιείται για τη δημιουργία παιχνιδιών σε πολλές πλατφόρμες όπως: Windows, Mac OS, Linux, Wii, iPhone-iPad, Android, και Palm. Η Shiva χρησιμοποιεί προηγμένη τεχνολογία για την απεικόνιση γραφικών, τον ήχο και τη μηχανή φυσικής. Το scripting language που χρησιμοποιεί είναι τύπου Lua και η τιμή ξεκινά από $ 250. Μερικά πολύ ωραία παιχνίδια για iPhone έχουν δημιουργηθεί με τη χρήση της.

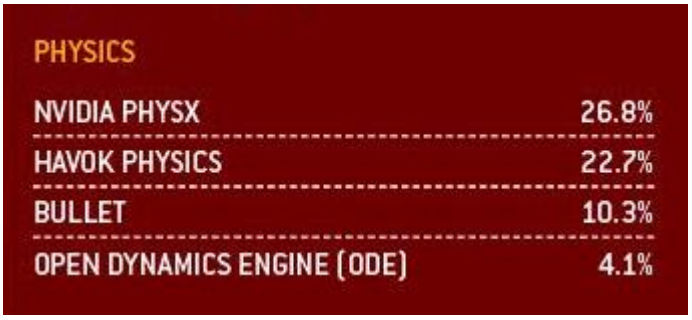
Sio2Engine - SiO2 Engine είναι μια open source 3D μηχανή με ένα εκτεταμένο σύνολο χαρακτηριστικών και χρησιμοποιεί Blender για τη δημιουργία των σκηνών. Έχει χρησιμοποιηθεί σε πολλούς διαφορετικούς τίτλους του iPhone. Είναι δωρεάν με προϋπόθεση την εμφάνιση μιας εικόνας (splash screen) με διαφήμιση της μηχανής κατά την έναρξη του παιχνιδιού. Αλλιώς η τιμή της είναι $ 49.99.

Unity 3D – Η Unity 3D είναι μια μηχανή παιχνιδιού που υπάρχει εδώ και πολλά χρόνια, και έχει γίνει αρκετά δημοφιλές για την ευκολία προγραμματισμού του. Έχει τα δικά της οπτικά εργαλεία για την επεξεργασία σκηνών και για τον προγραμματισμό χρησιμοποιεί την C# και Boo. Η Unity 3D είναι γνωστή για την ευκολία χρήσης της, και έχει χρησιμοποιηθεί σε πολλά διαφορετικά παιχνίδια iPhone. Το κόστος αρχίζει από $ 199 + $ 499.

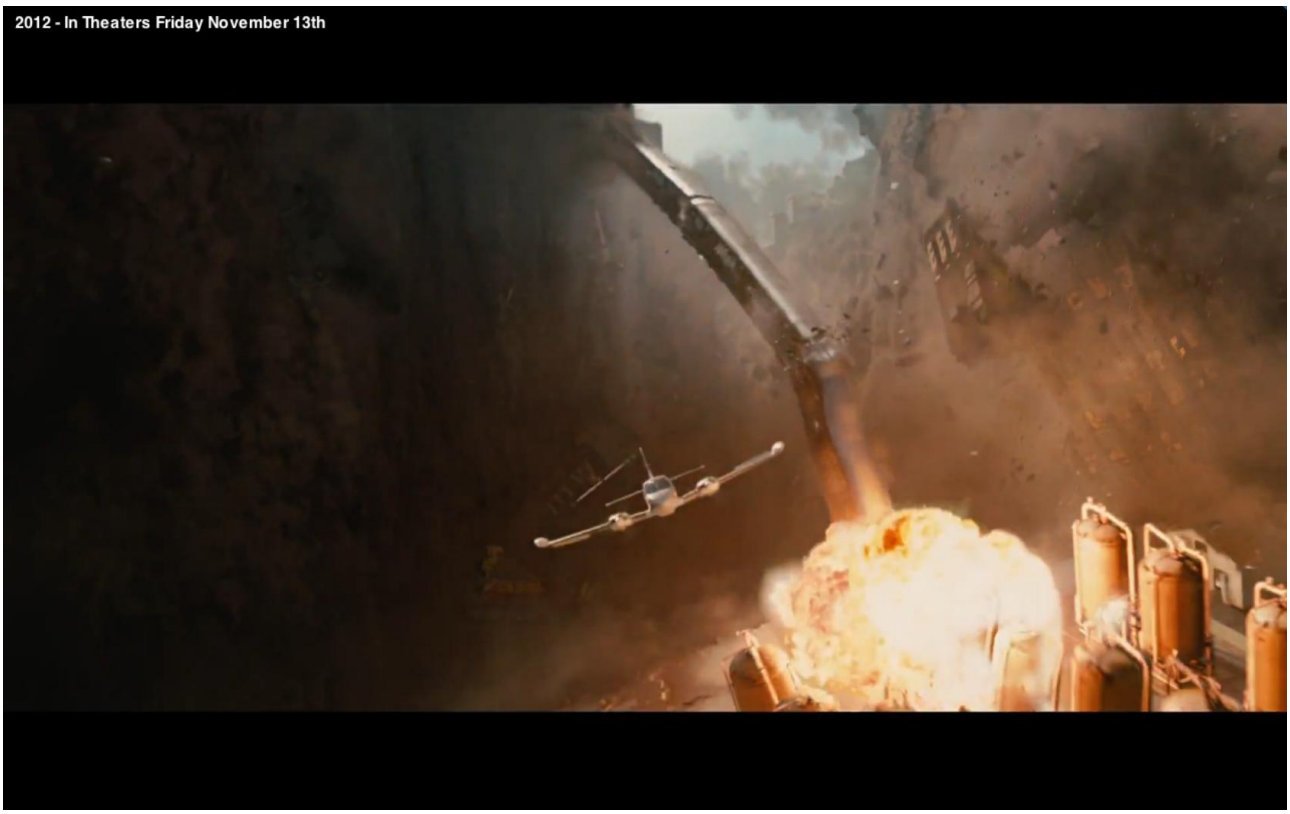
3.2.3 SIO2 Engine:

Μετά από αρκετό ψάξιμο κατέληξα στην **SIO2Engine** μηχανή. Ο βασικός λόγος ήταν ότι είναι ανοικτού κώδικα και για την δημιουργία και επεξεργασία 3d μοντέλων χρησιμοποιεί blender το οποίο είναι και αυτό ανοικτού κώδικα. Είναι πολύ πλούσια σε χαρακτηριστικά και δυνατότητες και δεν χρειάζονται ιδιαίτερες γνώσεις openGL για την χρήση της. Έχει όμως και αρκετές ελλείψεις σε κάποια βασικά χαρακτηριστικά.

Ένα από τα δυνατά σημεία της μηχανής SIO2 είναι ότι ενσωματώνει τη **Bullet Physics Library,** μια πολύ δημοφιλή open source βιβλιοθήκη για real-time υπολογισμό Dynamic και Rigid Body "φυσικής", έλεγχο collision, Particle System και ομίχλης. Η Bullet έχει χρησιμοποιηθεί σε πολλά παιχνίδια Η/Υ, σε κονσόλες όπως PS3, WII κλπ. αλλά και στον κινηματογράφο με χαρακτηριστικό παράδειγμα την ταινία 2012 όπου χρησιμοποιήθηκε η Bullet σε συνδυασμό με Maya για τη δημιουργία των ειδικών εφέ. Αυτή τη στιγμή η Bullet μετά την Nvidia Physics και την Havok Physics κατέχει την τρίτη θέση στην αγορά των βιβλιοθηκών φυσικής.

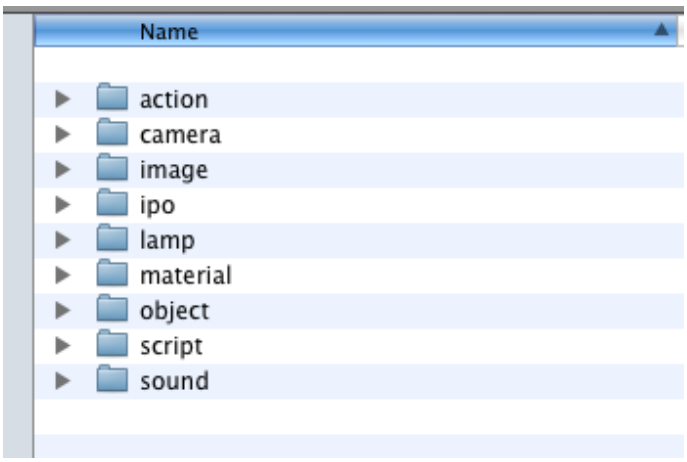


Εικόνα 3.2.1: Κατανομή βιβλιθηκών φυσικής



Εικόνα 3.2.2: Στιγμιότυπο από την ταινία 2012

Η SIO2 Engine σε αντίθεση με άλλες μηχανές παιχνιδιού είναι στην ουσία ένα Project του Xcode το οποίο περιέχει βιβλιοθήκες για την εισαγωγή και απεικόνιση 3D αντικειμένων, ήχου, κίνησης κλπ. Επίσης περιέχει 17 tutorials για να μπορεί ο χρήστης να δει πως να χρησιμοποιήσει τα βασικά χαρακτηριστικά της. Μερικά από τα tutorials συνοδεύονται και με video tutorials στα οποία επεξηγούνται κυρίως τα βήματα που πρέπει να γίνουν στο Blender. Στο πακέτο υπάρχει και ένας exporter γραμμένος σε python ο οποίος αναλαμβάνει την εξαγωγή της σκηνής από το blender σε αρχείο τύπου .sio2. Το αρχείο *.sio2 στην ουσία είναι ένας φάκελος συμπιεσμένος (zip) ο οποίος περιέχει μέσα ξεχωριστούς φακέλους για τα 3d αντικείμενα με τις φυσικές τους ιδιότητες, τις κάμερες, τα texture, τα material, τα φώτα, τους ήχους κα.



Για την εργασία θα ξεκινήσουμε με το Tutorial 06 το οποίο περιέχει το βασικό κώδικα για αρχικοποίηση σκηνής, κάμερας και των αντικειμένων. Επίσης περιέχει μια υλοποίηση φυσικής στα αντικείμενα(Bullet Physics). Από το Tutorial αυτό θα κρατήσουμε μόνο τα πολύ βασικά.

3.3 Blender

3.3.1 Περιγραφή

Για τον 3d σχεδιασμό θα χρησιμοποιήσουμε το Blender. Το Blender είναι ένα πρόγραμμα σχεδίασης τρισδιάστατων γραφικών, είναι ελεύθερο λογισμικό και διανέμεται από την άδεια GNU General Public License.

Χρησιμοποιείται για modeling, UV Mapping, rigging, προσομοιώσεις νερού και καπνού, animation, rendering και για δημιουργία αλληλεπιδραστικών 3d εφαρμογών όπως βιντεοπαιχνίδια και οπτικά εφφέ.

Είναι διαθέσιμο για όλα τα κύρια λειτουργικά συστήματα όπως τα Windows Mac OS και Linux, αλλά υποστηρίζεται και από FreeBSD και Solaris. Το Blender διαθέτει προχωρημένα εργαλεία για animation, διάφορα εργαλεία για σχεδίαση χαρακτήρων και ρούχων για τον χαρακτήρα, εργαλεία για δημιουργία υλικού καθώς επίσης και τη γλώσσα προγραμματισμού Python για εσωτερικό scripting.

3.3.2 Ιστορία:

Το Blender αρχικά αναπτύχθηκε από τα Ολλανδικά animation studio NeoGeo και Not a Number Technologies (NaN). Την ανάπτυξη του Blender είχε αρχίσει ο Ton Roosendaal, ο οποίος προηγουμένως είχε γράψει ένα ray tracer για Amiga το 1989 με όνομα Tracers. Το όνομα "Blender" το εμπνεύστηκε από το ομώνυμο τραγούδι του σουηδικού συγκρότημα Yello από το άλμπουμ τους "Baby".

Ο Roosendaal ίδρυσε την NaN τον Ιούνιο του 1998 για την περαιτέρω ανάπτυξη του προγράμματος. Το Blender διανεμόταν ως shareware μέχρι το 2002 όπου η NaN χρεοκόπησε.

Οι πιστωτές συμφώνησαν για την έκδοση του Blender με την άδεια GNU General Public License με την πληρωμή των €100,000. Στις 18 Ιουλίου 2002 ο Roosendaal ξεκίνησε μια εκστρατεία χρηματοδότησης για την είσπραξη δωρεών, και στις 7 Σεπτεμβρίου 2002 ανακοινώθηκε ότι μαζεύτηκαν αρκετά χρήματα και ο πηγαίος κώδικας θα ήταν ελεύθερος. Το Blender τώρα είναι ελεύθερο λογισμικό και αναπτύσσεται δυναμικά υπό την επίβλεψη της Blender Foundation.

Η Blender Foundation κράτησε το δικαίωμα για την έκδοση του Blender με δύο άδειες, έτσι μαζί με το GNU General Public License το Blender θα μπορούσε να εκδοθεί και με την άδεια "Blender License" το οποίο δεν απαιτεί το κλείσιμο του κώδικα αλλά πληρωμές στην Blender Foundation. Ωστόσο αυτό το δικαίωμα δεν ασκήθηκε ποτέ, και το 2005 αναστάλθηκε επ'αορίστου. Τώρα το Blender διατίθεται εξ ολοκλήρου με την άδεια GNU GPL.

Το Blender έχει σχετικά μικρό μέγεθος και τρέχει σε όλες κύριες πλατφόρμες, συμπεριλαμβανομένου Linux, Mac OS X και Microsoft Windows μαζί με τα FreeBSD, IRIX, OpenBSD, NetBSD και Solaris. Από ανεπίσημες πηγές είναι διαθέσιμο και για AmigaOS 4, BeOS, MorphOS, Pocket PC και SkyOS. Μερικά από τα χαρακτηριστικά του είναι:

Υποστήριξη για δικτυώματα πολυγώνου, γρήγορη υποδιαίρεση επιφάνειας μοντελοποίησης, καμπύλες Bezier, επιφάνειες NURBS, metaball, ψηφιακό sculpting, και ανυσματικές γραμματοσειρές.

Ευέλικτη δυνατότητα εσωτερικού rendering και ενσωμάτωση με το πρόγραμμα YafaRay.

Μη γραμμική επεξεργασία βίντεο/ήχου

Η γλώσσα προγραμματισμού Python για προγραμματισμό λογικής και επιπλέον scripting.

Το Game-Blender για δημιουργία ηλεκτρονικών παιχνιδιών με το Blender

Το Blender έχει την φήμη ότι είναι δύσκολο στην εκμάθηση από χρήστες οι οποίοι είναι συνηθισμένοι σε διαφορετικά προγράμματα επεξεργασίας τρισδιάστατων γραφικών

Εικόνα 3.3.1: Το Blender μπορεί να δημιουργήσει εικόνες πολύ υψηλής ανάλυσης

3.3.3 Elephants Dream (Project Orange)

Τον Σεπτέμβριο του 2005, μερικοί από τους πιο αξιόλογους καλλιτέχνες και προγραμματιστές Blender άρχισαν να εργάζονται σε μια ταινία μικρού μήκους χρησιμοποιώντας κυρίως ελεύθερο λογισμικό, σε μια πρωτοβουλία γνωστή ως The Orange Movie Project. Η προκύπτουσα ταινία, Elephants Dream έκανε πρεμιέρα στις 24 Μαρτίου 2006. Το αποτέλεσμα ήταν εκπληκτικό και γνώρισε μεγάλη επιτυχία.

Εικόνα 3.3.2: Στιγμιότυπο από το Elephants Dream

Εικόνα 3.3.3: Περιβάλλον εργασίας του Blender

3.4 Adobe Photoshop

3.4.1 Περιγραφή

Για την δημιουργία και επεξεργασία εικόνων texture κλπ θα χρησιμοποιήσουμε το Photoshop CS4.

Το Adobe Photoshop είναι ένα πρόγραμμα επεξεργασίας γραφικών που αναπτύχθηκε και κυκλοφόρησε από την Adobe Systems. Αυτή τη στιγμή αποτελεί ηγέτη της αγοράς (market leader) των προγραμμάτων επεξεργασίας εικόνων, και είναι το προϊόν - σήμα κατατεθέν της Adobe Systems. Χαρακτηρίζεται ως "απαραίτητο εργαλείο για τους επαγγελματίες γραφίστες" και θεωρείται πως προώθησε τις αγορές των Macintosh, και στη συνέχεια των Windows.

Η 11η, και πιο πρόσφατη έκδοση του Adobe Photoshop, είναι η Adobe Photoshop CS4, που κυκλοφόρησε τον Οκτώβριο του 2008.

3.4.2 Ιστορία

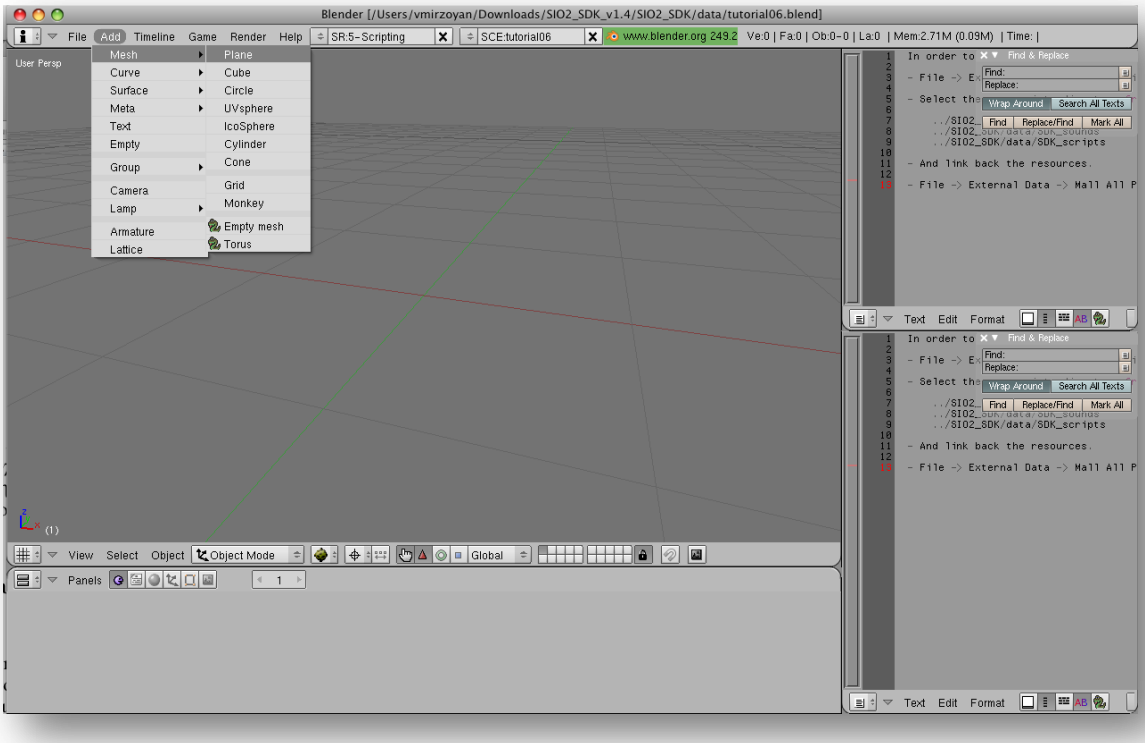
Το 1987 ο Τόμας Κνολ, ένας φοιτητής του Πανεπιστημίου του Μίσιγκαν, ανέπτυξε ένα πρόγραμμα που εμφάνιζε εικόνες grayscale σε μονοχρωματικό περιβάλλον. Αυτό το πρόγραμμα, το οποίο ονόμασε Display, τράβηξε την προσοχή του αδερφού του Τζον Κνολ, ο οποίος πρότεινε στον Τόμας να αναπτύξει ένα πλήρες πρόγραμμα επεξεργασίας εικόνας. Ο Τόμας έκανε διάλειμμα έξι μηνών από τις σπουδές του το 1988 και, σε συνεργασία με τον αδερφό του, ανέπτυξε το πρόγραμμα, το οποίο ονόμασαν ImagePro. Αργότερο το ίδιο έτος, ο Τόμας μετονόμασε το πρόγραμμα του σε Photoshop και έπειτα από συμφωνία με την κατασκευάστρια εταιρία σαρωτών Barneyscan, το πρόγραμμα διανεμήθηκε μαζί με μερικούς σαρωτές. Συνολικά 200 αντίγραφα του προγράμματος διανεμήθηκαν.

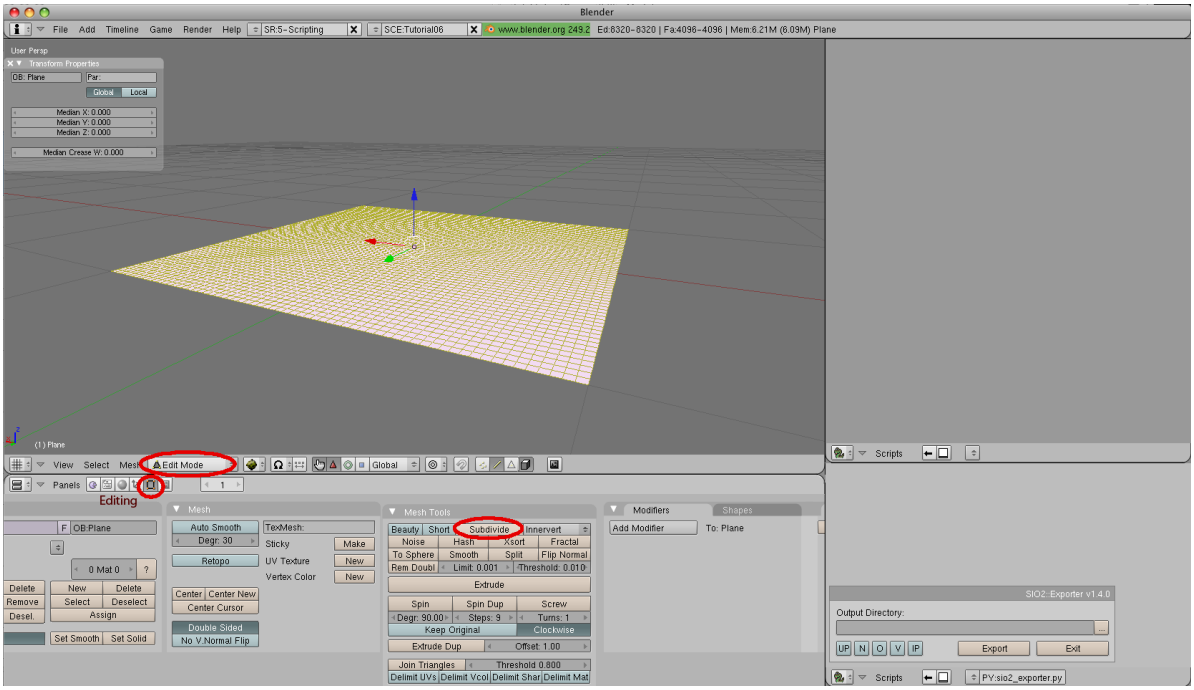
Εν τω μεταξύ, ο Τζον ταξίδεψε στο Σίλικον Βάλεϊ και παρουσίασε το πρόγραμμα του στους μηχανικούς της Apple και στην Adobe. Και οι δύο παρουσιάσεις ήταν επιτυχείς, καθώς η Adobe αποφάσισε να αγοράσει την άδεια να διανείμει το πρόγραμμα τον Σεπτέμβριο του 1988. Η επόμενη έκδοση του προγράμματος, η Photoshop 1.0, κυκλοφόρησε το 1990 αποκλειστικά για συστήματα Macintosh και είχε μέγεθος 1.44 MB.

4 Σχεδίαση 3D χώρου και αντικειμένων

4.1 Δημιουργία και Texturing του Terrain

Στο Blender ανοίγουμε το αντίστοιχο 3d αρχείο tutorial06.blend και σβήνουμε όλα τα περιεχόμενά του. Πατώντας δυο φορές το πλήκτρο A μαρκάρονται όλα τα αντικείμενα που υπάρχουν στη σκηνή και μετά τα σβήνουμε πιέζοντας το πλήκτρο Delete. Για να δημιουργήσουμε το terrain αρχικά φτιάχνουμε ένα plane. Από το μενού add διαλέγουμε την επιλογή mesh και μετά plane. Για να μπορέσουμε να το διαμορφώσουμε πρώτα πρέπει να γίνει subdivide ώστε να έχουμε περισσότερα σημεία ελέγχου(vertices). Για να το πετύχουμε αυτό πατάμε το πλήκτρο Tab για να αλλάξουμε από object mode σε Edit Mode. Μετά από το παράθυρο panels διαλέγουμε την καρτέλα Editing και στη συνέχεια την επιλογή subdivide**.** Κάθε φορά που το πατάμε υποδιαιρεί κάθε πολύγωνο σε 4 μικρότερα. Οπότε για να έχουμε μια ικανοποιητική ποσότητα από πολύγωνα το πατάμε 6 φορές τουλάχιστον. Εδώ θέλει όμως λίγη προσοχή γιατί καθώς αυξάνεται ο αριθμός των πολυγώνων αυξάνεται η λεπτομέρεια, αλλά αυτό έχει ως συνέπεια να μειώνονται οι επιδόσεις. Η SIO2 μπορεί να χειριστεί μέχρι 20.000 πολύγωνα περίπου και να έχει καλές επιδόσεις (50-60 fps).

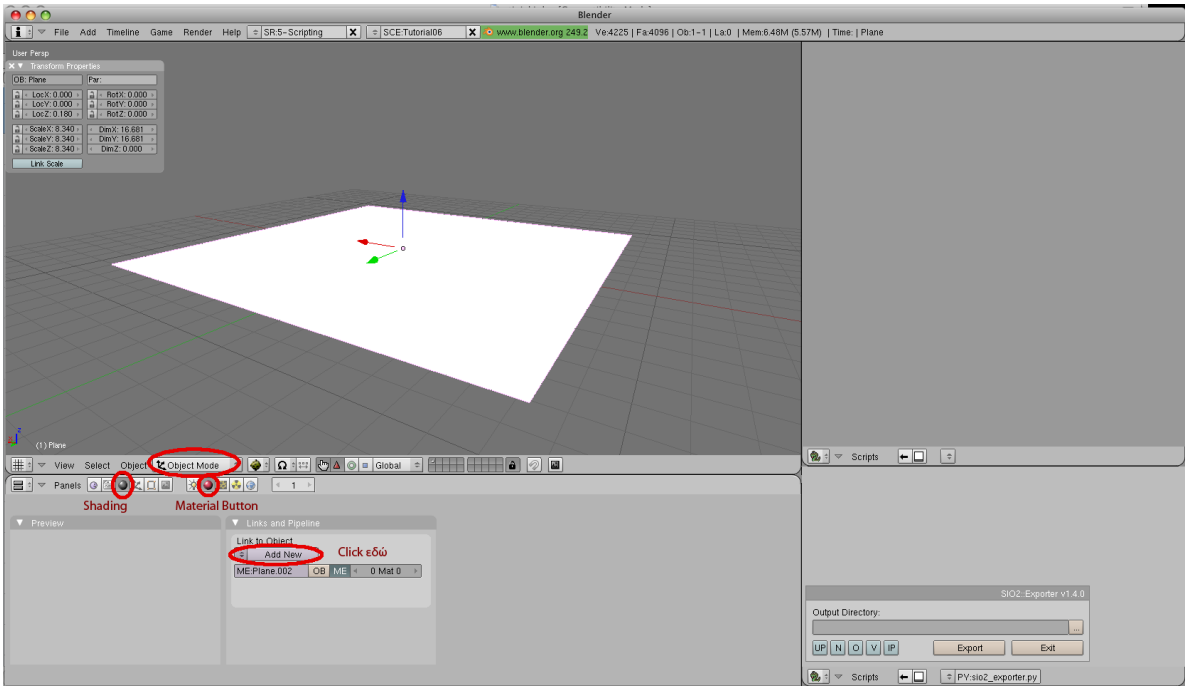




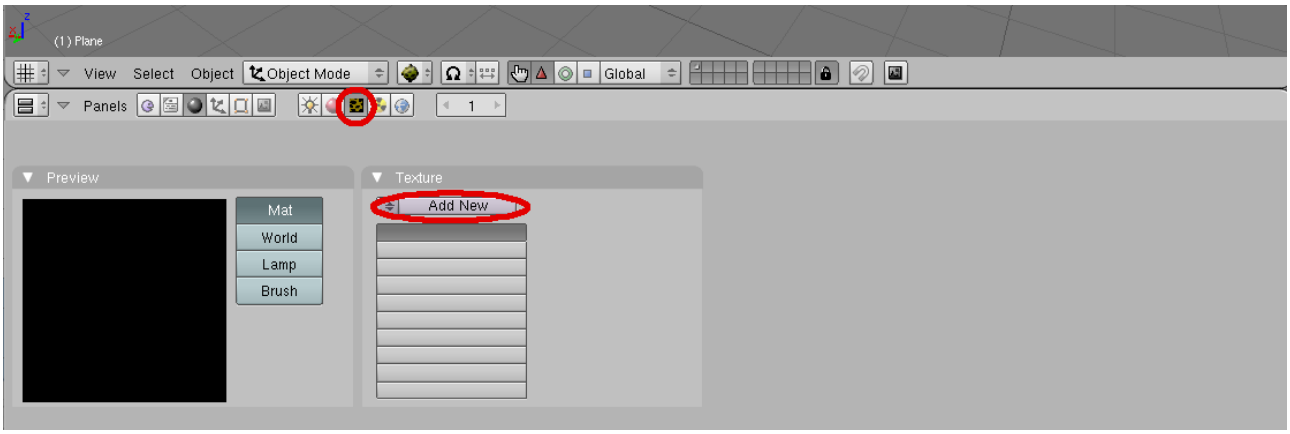
Μετά ξαναπατάμε το πλήκτρο **Tab** για να αλλάξει από Edit Mode σε Object Mode.

Το επόμενο βήμα είναι να προσθέσουμε ένα Texture από terrain στο plane. Για

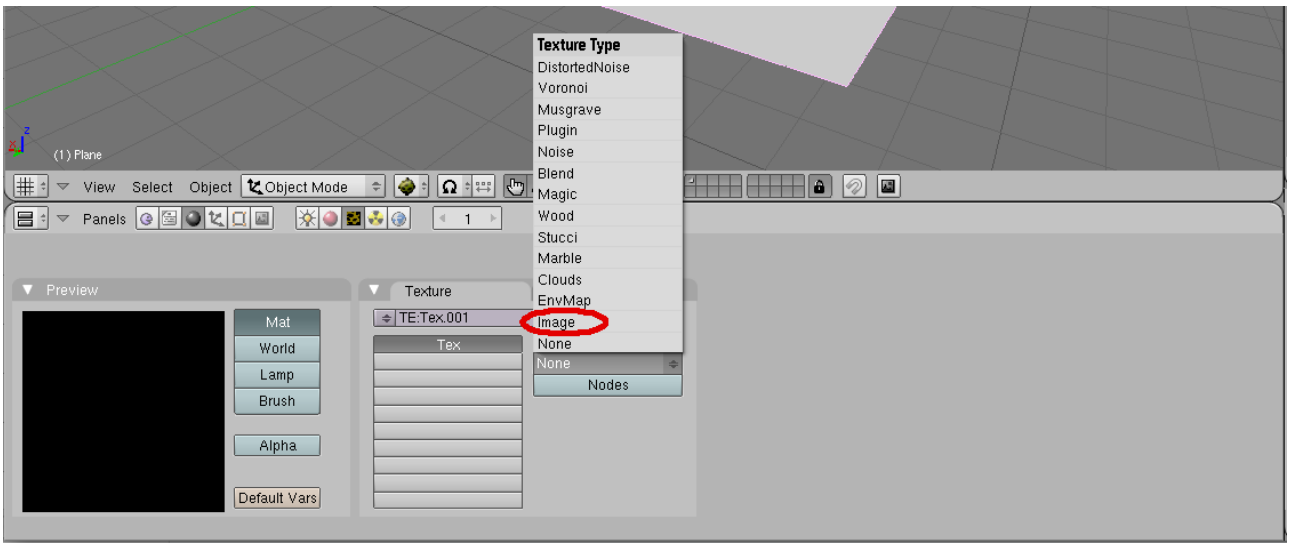
να γίνει αυτό πρέπει πρώτα να προσθέσουμε ένα καινούριο Material. Ενώ είμαστε σε Object Mode επιλέγουμε το Plane, διαλέγουμε την καρτέλα Shading και μετά την επιλογή Material Button**.** Στη συνέχεια **δ**ιαλέγουμε Add New για να δημιουργηθεί ένα νέο Material.



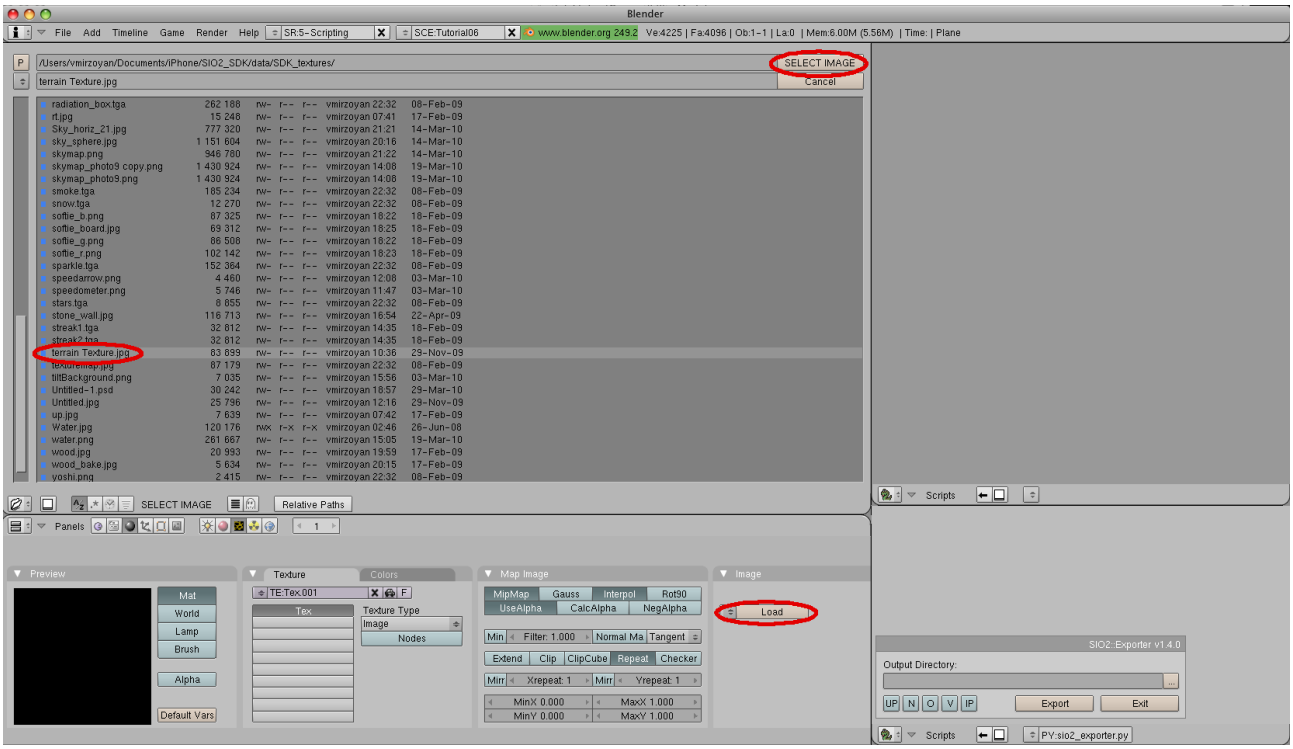
Επόμενο βήμα είναι η προσθήκη του Texture στο Material. Στην καρτέλα Shading διαλέγουμε Texture Button και μετά, την επιλογή Add New.



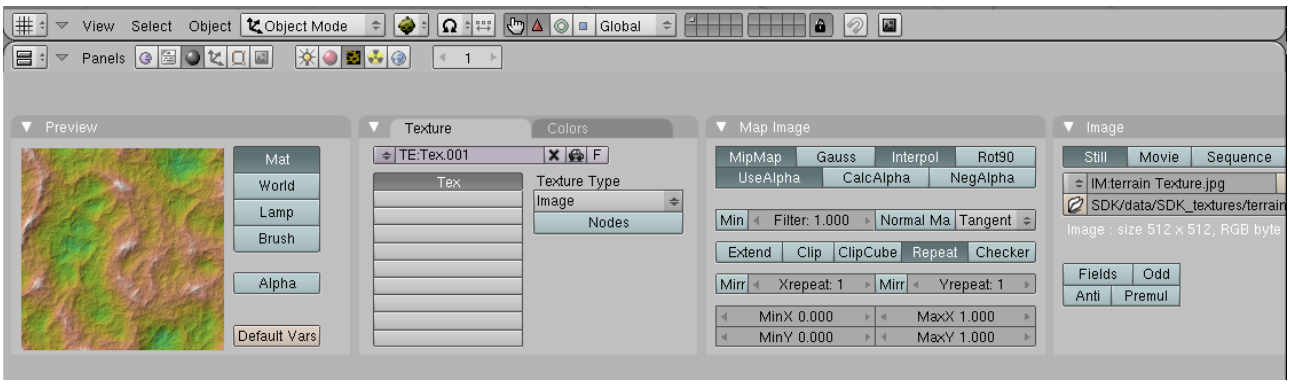
Από τη λίστα Texture Type διαλέγουμε το image**.**



Τώρα επιλέγουμε Load, διαλέγουμε το αρχείο του texture και στη συνέχεια Select Image.

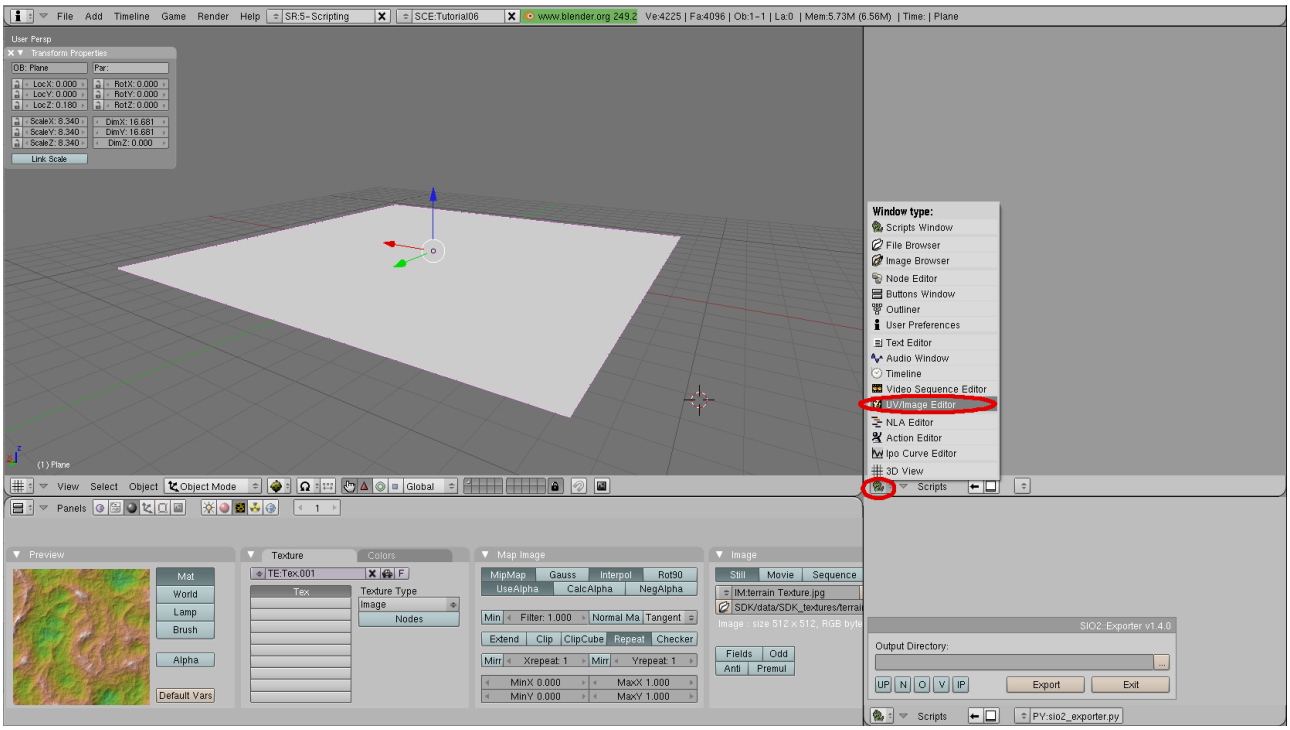


Τώρα το texture έχει προστεθεί στο material, αλλά δεν φαίνεται ακόμα στο plane.



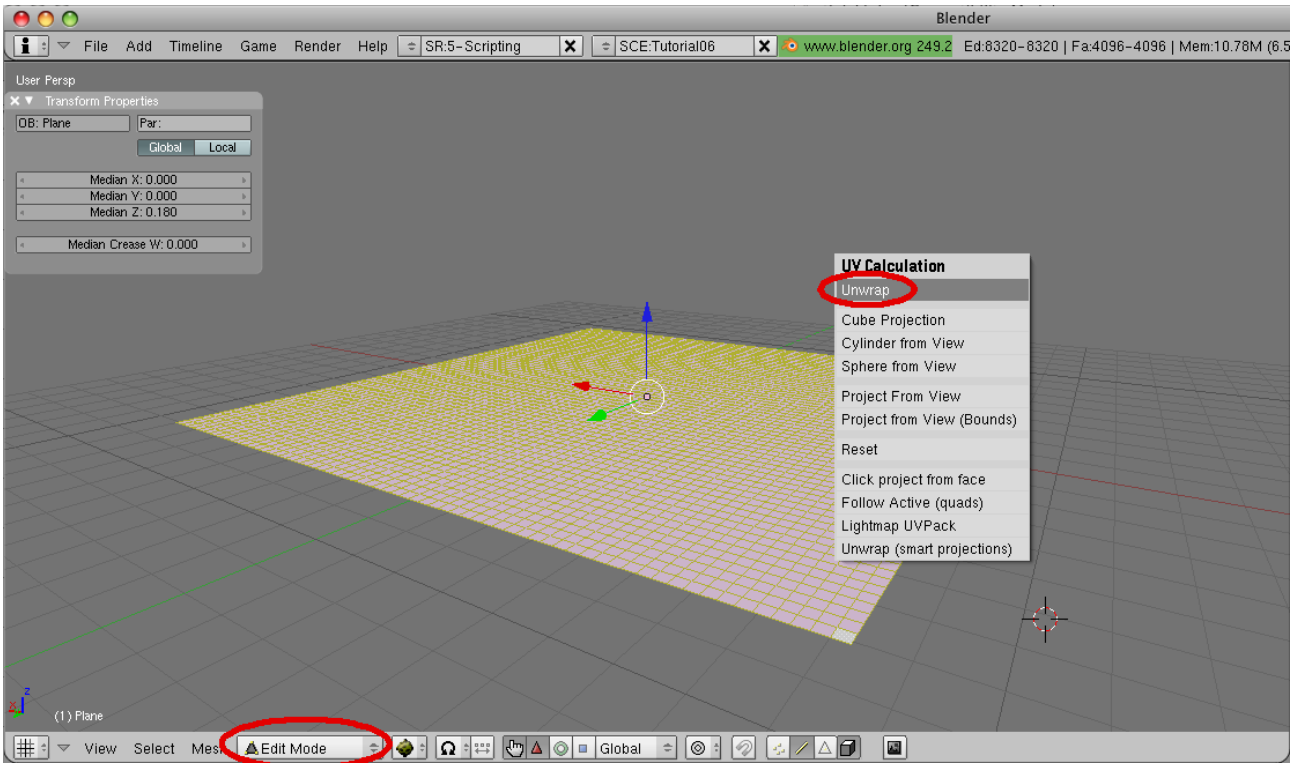
Επόμενο βήμα είναι το UV Mapping. Το UV Mapping είναι η διαδικασία της αντιστοίχησης συγκεκριμένων σημείων ενός Texture (εικόνας) σε συγκεκριμένα σημεία ενός αντικειμένου. Στην περίπτωσή μας, τα σημεία της εικόνας θα αντιστοιχηθούν με τα σημεία του Plane. Για να το πετύχουμε αυτό χρειαζόμαστε τον UV/Image Editor.

Σε οποιοδήποτε από τα παράθυρα του Blender πατάμε το κουμπί που βρίσκεται κάτω αριστερά και από τη λίστα που εμφανίζεται διαλέγουμε το UV/Image Editor.

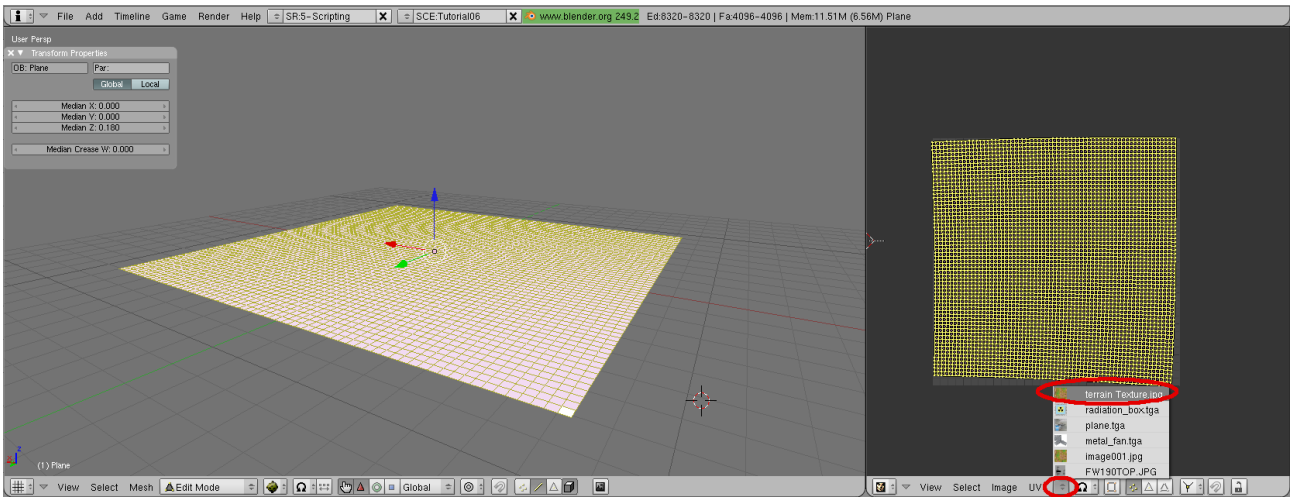


Στη συνέχεια έχοντας επιλεγμένο το plane μεταβαίνουμε σε Edit Mode

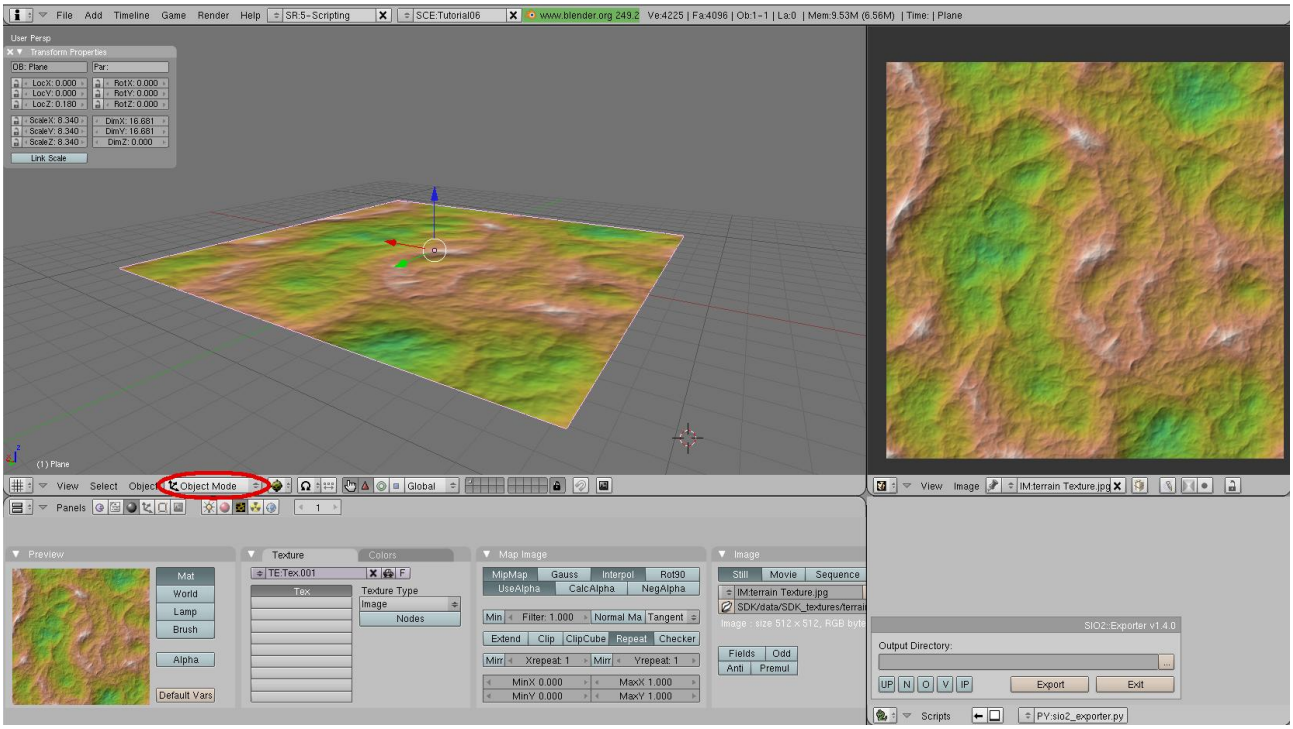
πατώντας το πλήκτρο Tab. Πατώντας το πλήκτρο U στο πληκτρολόγιο εμφανίζεται το μενού UV Calculation και από αυτό διαλέγουμε Unwrap.



Μετά από τη λίστα με τα textures διαλέγουμε το texture του terrain μας.

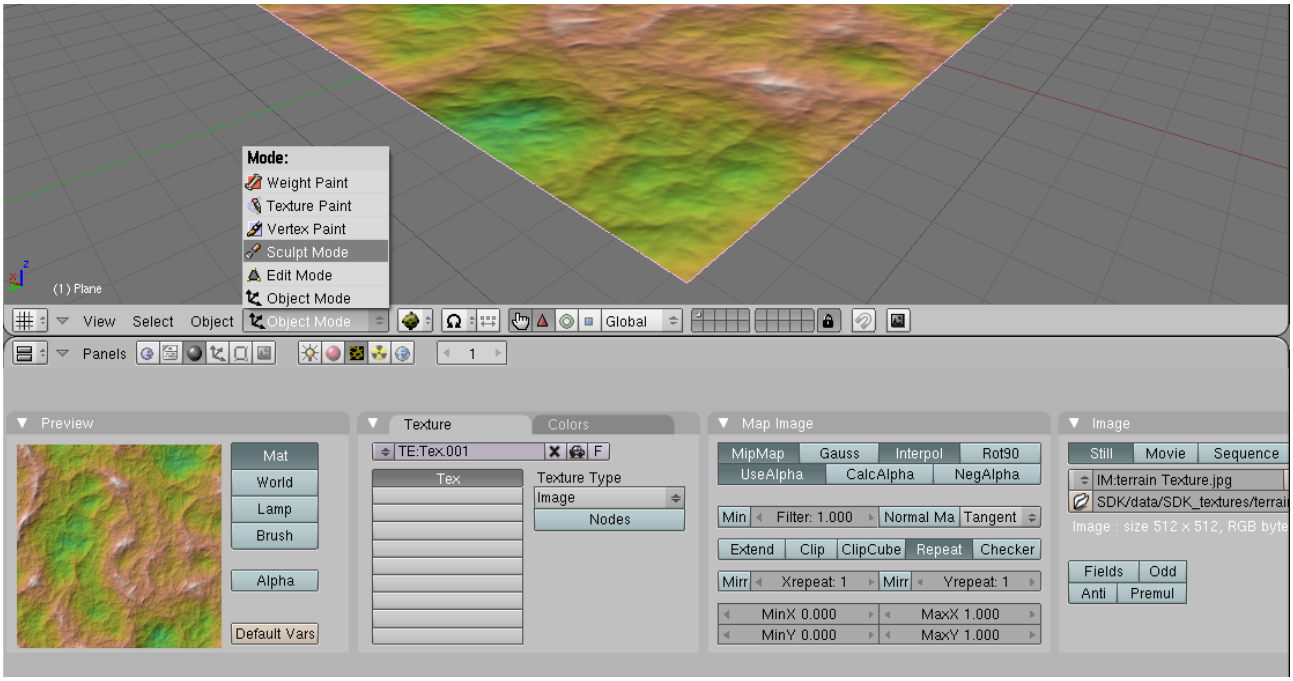


Το αποτέλεσμα φαίνεται παρακάτω.

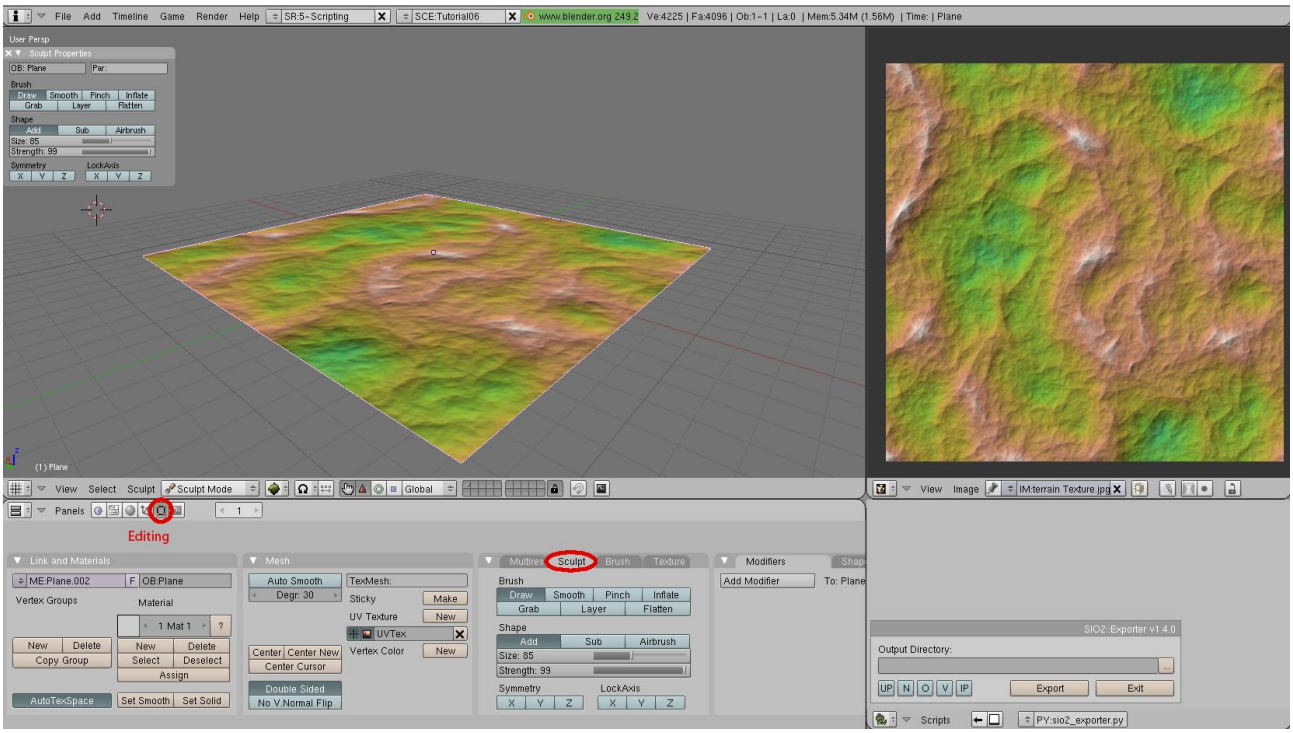


Επόμενο βήμα είναι το Sculpting.

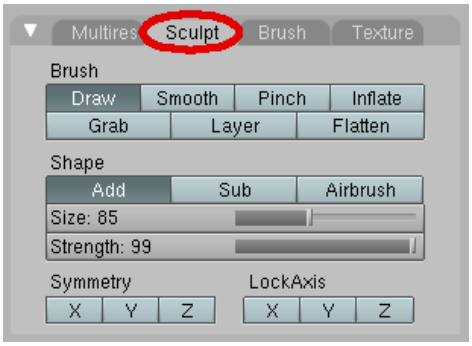
Έχοντας επιλεγμένο το plane, από Object mode μεταβαίνουμε σε Sculpt mode από το μενού Mode που βρίσκεται στη βάση του παραθύρου 3D View.



Στη συνέχεια μέσα από την καρτέλα Editing διαλέγουμε την καρτέλα Sculpt.



Από εδώ παραμετροποιούμε το Brush.



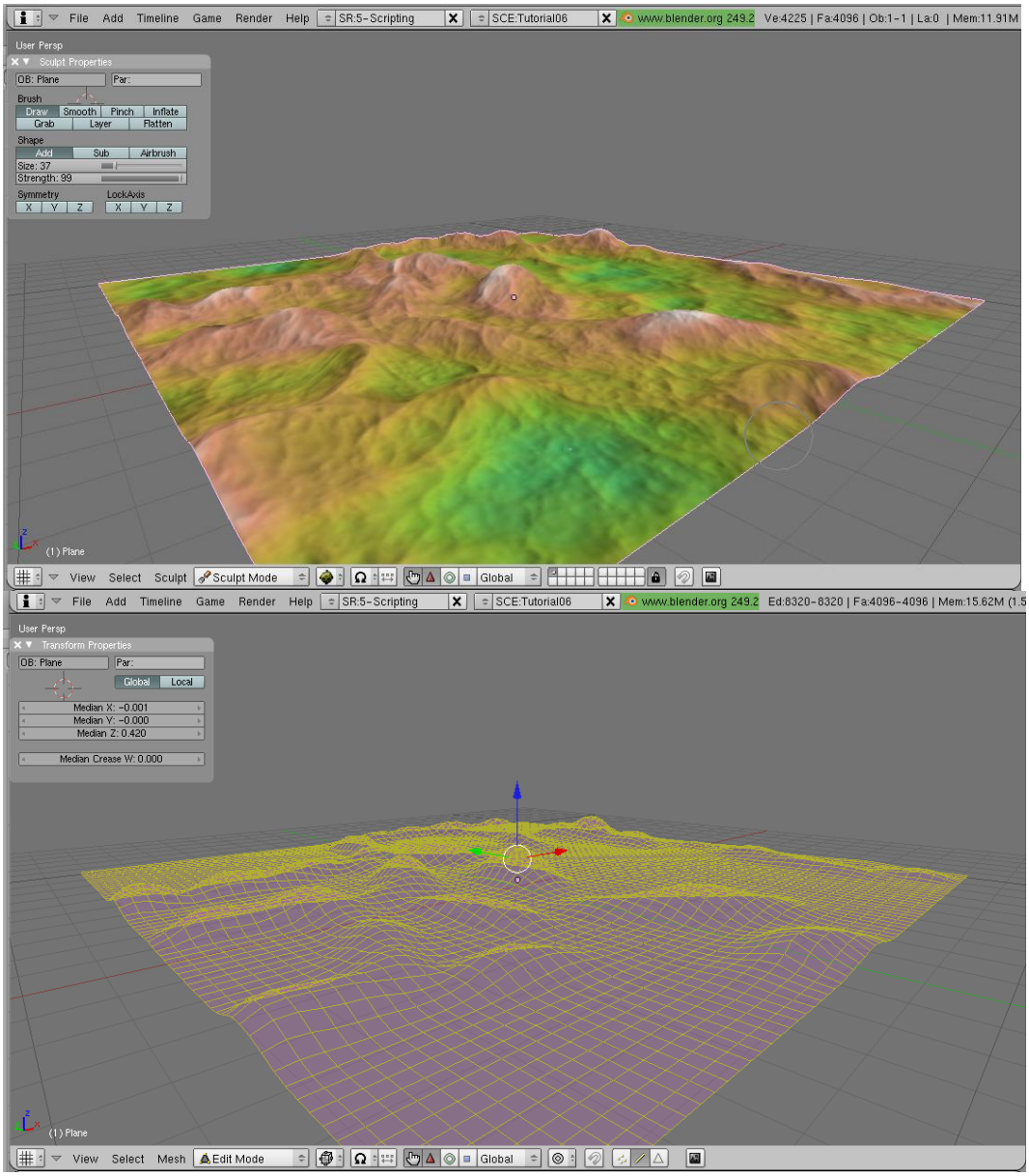
Οι παράμετροι που μας ενδιαφέρουν κυρίως είναι τα Add, Sub, Size και Strength.

Τα Add και Sub καθορίζουν το αν το ύψος των σημείων που επηρεάζουμε θα αυξηθούν ή θα μειωθούν.

Το Size καθορίζει το μέγεθος του Brush και το Strength την ποσότητα της αλλαγής του ύψους των σημείων.

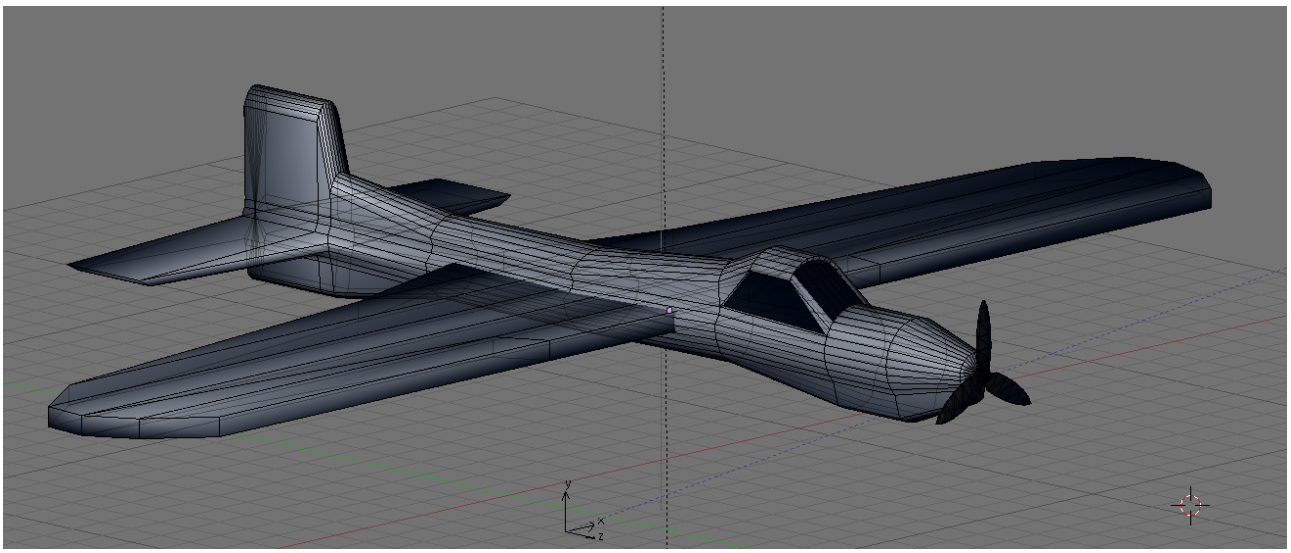
Έχοντας ρυθμίσει το Brush κατάλληλα το χρησιμοποιούμε για να διαμορφώσουμε το ανάγλυφο της επιφάνειας του plane. Συγκεκριμένα αυτό το πετυχαίνουμε τοποθετώντας τον κέρσορα στα σημεία των οποίων το ύψος θέλουμε να αυξήσουμε ή να μειώσουμε και πατώντας το αριστερό πλήκτρο του ποντικιού.

Παρακάτω φαίνεται το τελικό αποτέλεσμα.

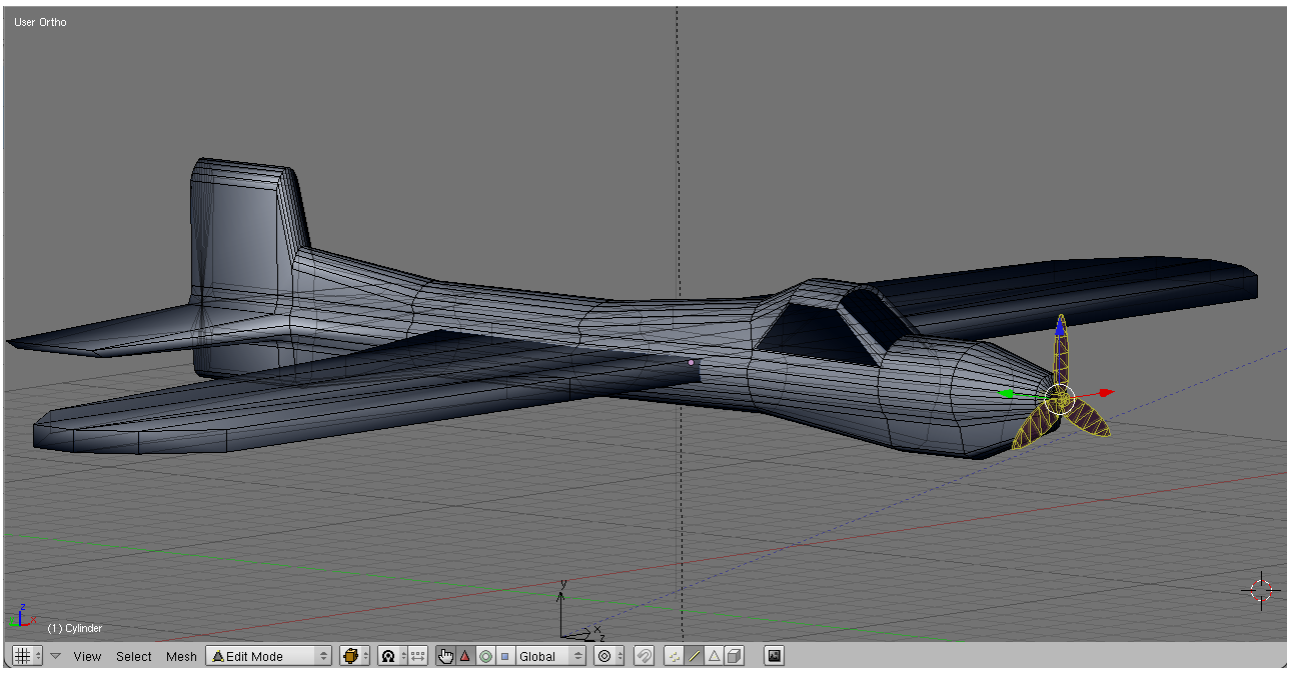


4.2 Δημιουργία και Texturing του Αεροπλάνου

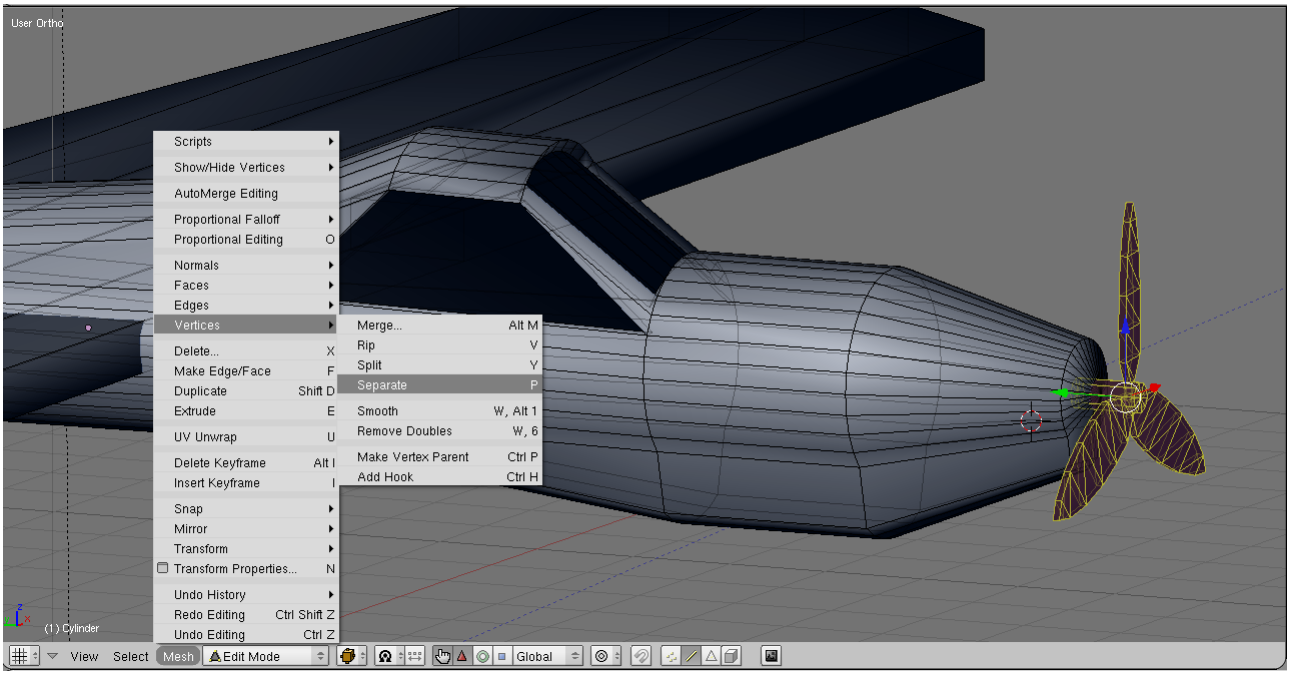
Για λόγους συντομίας θα χρησιμοποιηθεί ένα έτοιμο αεροπλάνο που κατεβάσαμε από το Ίντερνετ, από την τοποθεσία: http://www-roc.inria.fr/gamma/gamma/download/AIRCRAFT/index0.php η οποία περιέχει πολλά διαφορετικά μοντέλα αεροπλάνων δωρεάν και σε πολλές μορφές(formats). Το Texturing όμως θα γίνει στο Blender. Από αυτά επέλεξα αυτό που φαίνεται παρακάτω για περεταίρω επεξεργασία. Η αρχική μορφή του αεροπλάνου είναι η παρακάτω, η οποία δεν έχει Textures. Το επόμενο βήμα λοιπόν θα είναι η δημιουργία Texture για το αεροπλάνο.



Πριν από αυτό όμως θα πρέπει να χωρίσουμε τον έλικα του αεροπλάνου σε ξεχωριστό mesh για να μπορέσει να κινείται ανεξάρτητα. Για να το πετύχουμε αυτό επιλέγουμε το αεροπλάνο και μεταβαίνουμε σε Edit Mode πατώντας το πλήκτρο tab. Στη συνέχεια πατάμε το πλήκτρο “A” ώστε να αποεπιλέξουμε όλα τα πολύγωνα και στη συνέχεια επιλέγουμε μόνο τον έλικα.



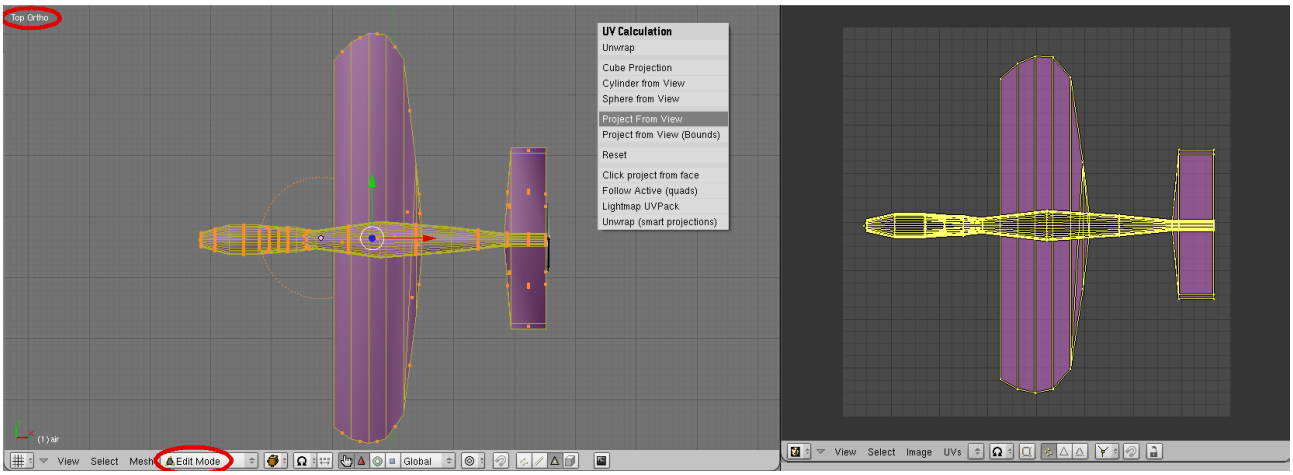
Από το μενού επιλέγουμε Mesh -> Vertices -> Separate. Μ’αυτό τον τρόπο διαχωρίζουμε το αεροπλάνο και τον έλικα σε δύο ξεχωριστά αντικείμενα.



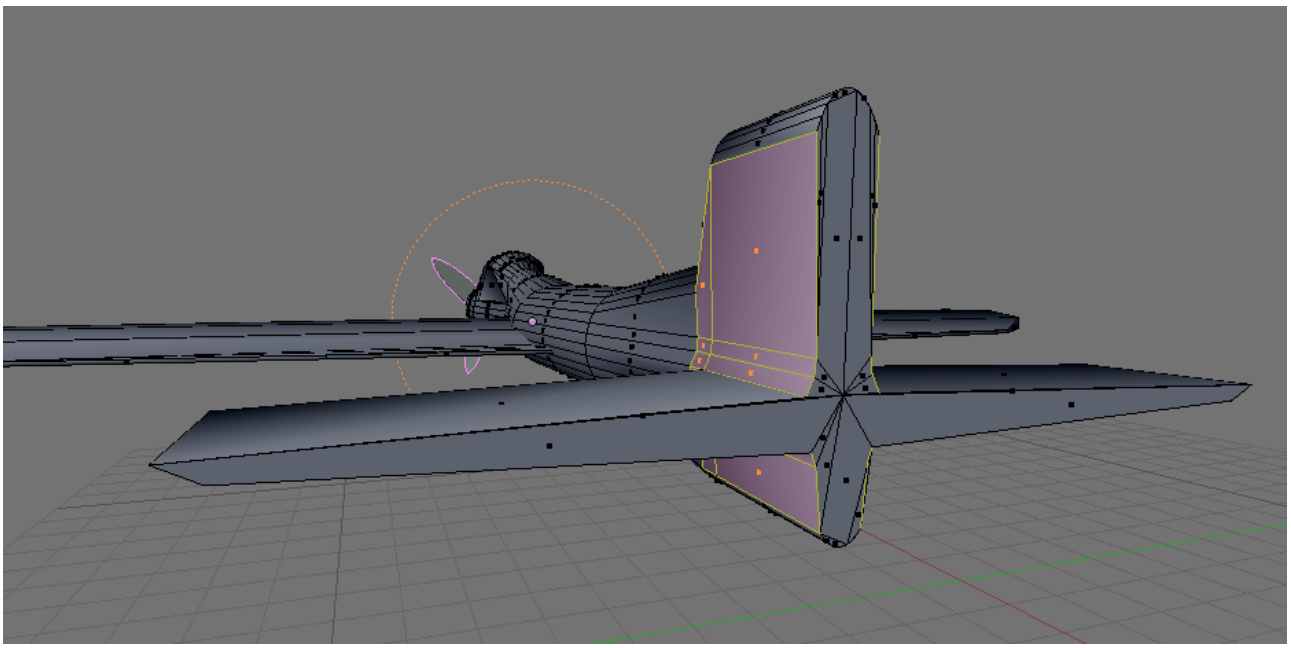
Στη συνέχεια θα φτιάξουμε το Texture για το αεροπλάνο.

Φτιάχνουμε ένα καινούριο Material παρόμοιο με αυτό του Terrain, αλλά δεν προσθέτουμε το Texture για την ώρα. Σε σύγκριση με το Terrain που ήταν απλώς ένα επίπεδο, το αεροπλάνο που έχει πολύπλοκο σχήμα χρειάζεται διαφορετικό UV Mapping.

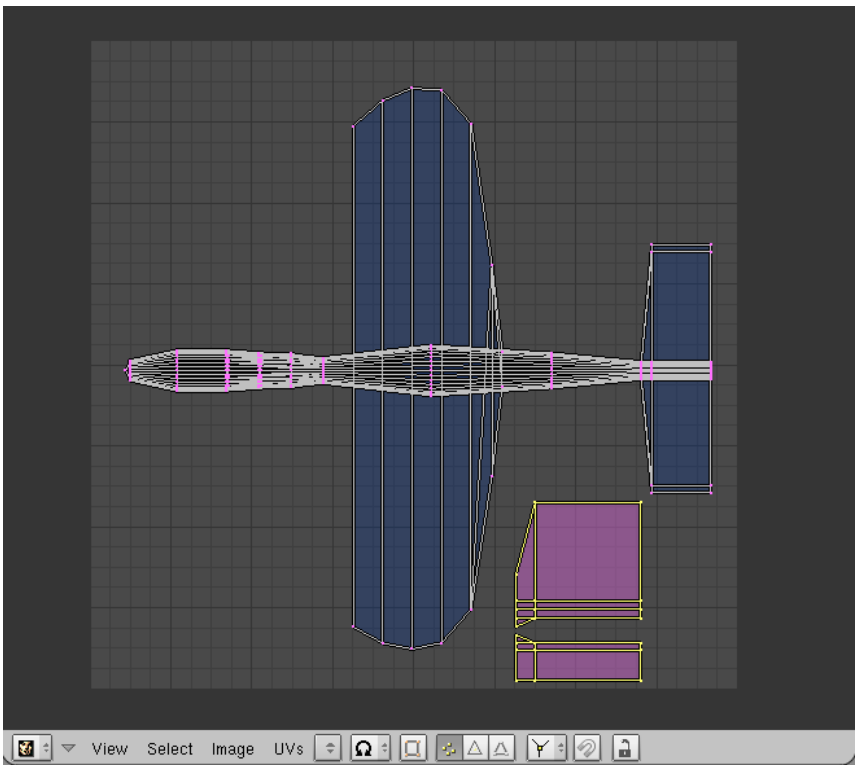
Αρχικά έχοντας επιλεγμένο το αεροπλάνο, το θέτουμε σε Edit Mode. Στη συνέχεια πατάμε το A για να επιλέξουμε όλα τα πολύγωνα, μετά πατάμε το 7 από το NumPad για να μεταβούμε στο Top View και το 5 για να αλλάξουμε σε ορθογραφική προβολή. Τέλος πατάμε το πλήκτρο U για να εμφανίσουμε το UV Calculation μενού και διαλέγουμε Project from View.



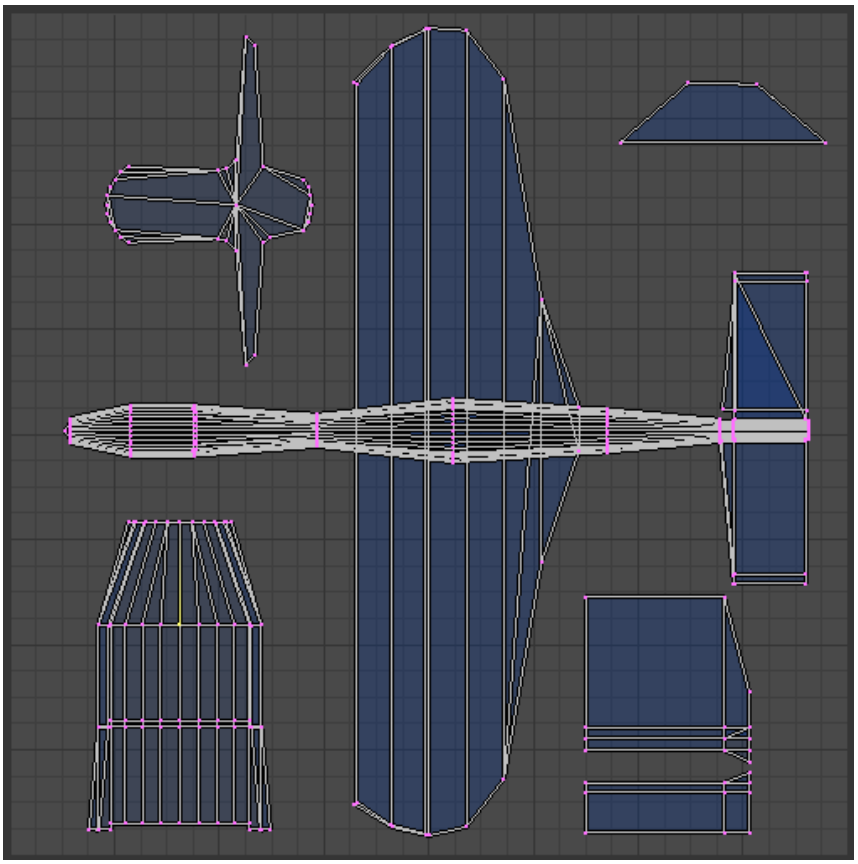
Αυτό μας δίνει ένα αρχικό mapping, αλλά δεν είναι αρκετό. Κάποια πολύγωνα δεν είναι ευδιάκριτα από το Top View, έτσι πρέπει να τα προβάλουμε από άλλη οπτική γωνία. Για παράδειγμα τα πολύγωνα που φαίνονται στην εικόνα θα τα προβάλουμε στο Side View.



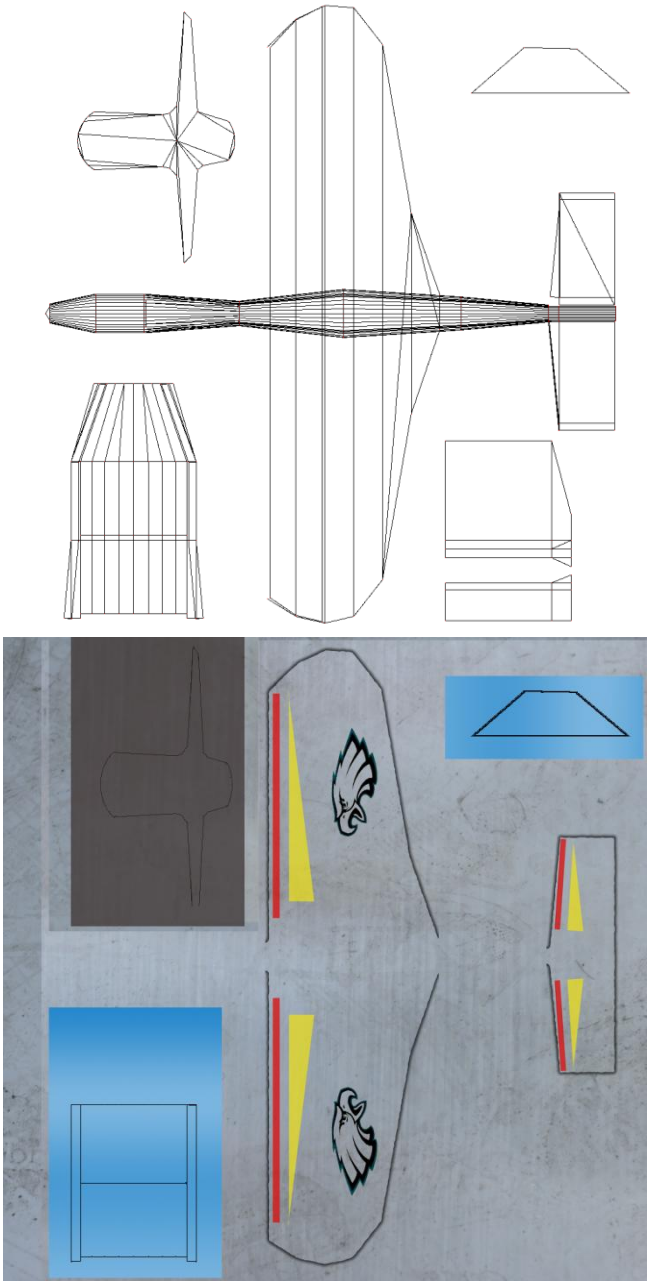
Η διαδικασία είναι σχεδόν η ίδια, απλά αντί να διαλέξουμε όλα τα πολύγωνα διαλέγουμε μόνο αυτά που μας ενδιαφέρουν και αντί για Top διαλέγουμε το View που ταιριάζει καλύτερα. Αφού το προβάλουμε στο UV/Image Editor το τοποθετούμε κάπου ώστε να μην καλύπτει ή καλύπτεται από το άλλο.



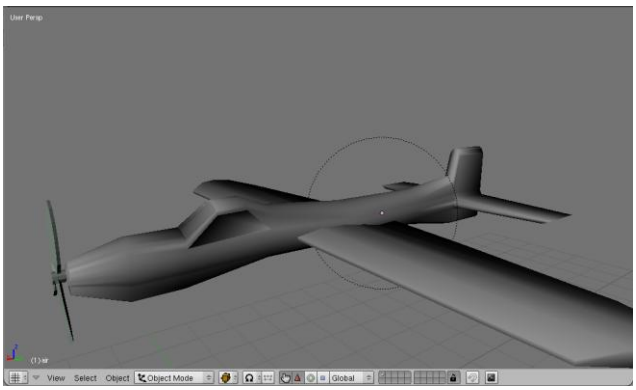
Συνεχίζουμε την ίδια διαδικασία όπου χρειάζεται. Στο τέλος τα στοιχίζουμε όλα μέσα στο πλέγμα ώστε να καταλαμβάνουν όσο το δυνατόν περισσότερο χώρο χωρίς όμως να αλληλεπικαλύπτονται.

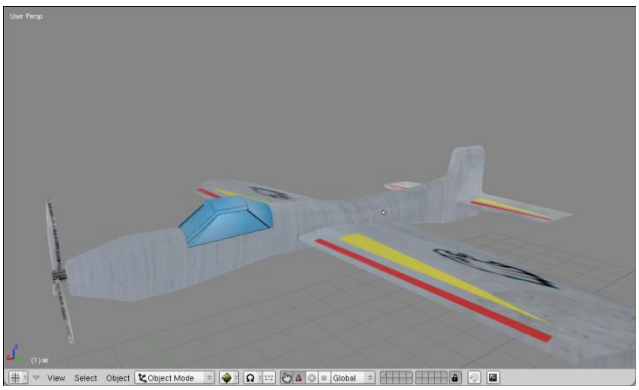


Το UV Layout είναι έτοιμο. Τώρα πρέπει να εξάγουμε το Layout σε μορφή εικόνας ώστε να φτιάξουμε το Texture. Στο UV/Image Editor πατάμε UVs -> Scripts -> Save UV Face Layout. Στο παράθυρο που εμφανίζεται αφήνουμε τις ρυθμίσεις ως έχουν, αλλάζοντας μόνο το size σε 1024 και πατάμε OK. Στο επόμενο παράθυρο ορίζουμε την τοποθεσία του αρχείου και πατάμε Save UV(tga). Στη συνέχεια ανοίγουμε την εικόνα με το Photoshop και αρχίζουμε την ζωγραφική.

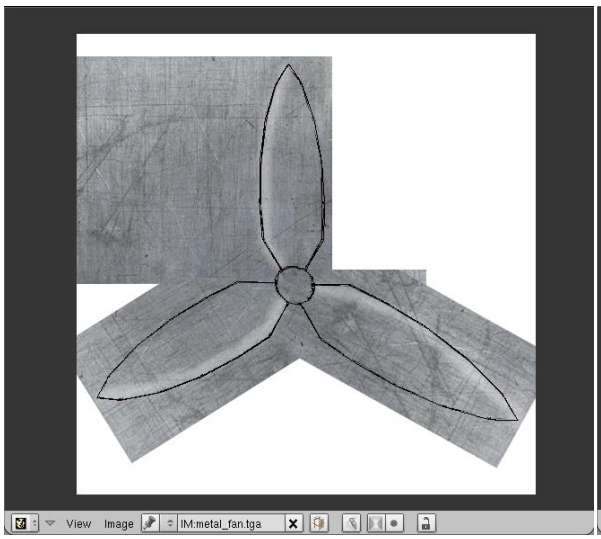


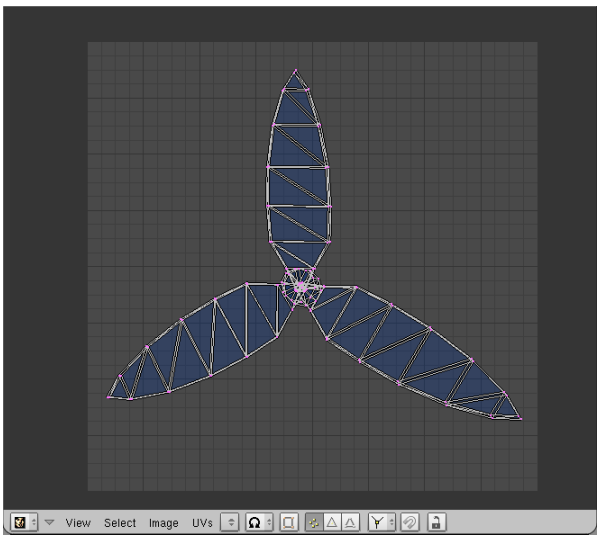
Τώρα πρέπει να εισάγουμε το τελικό Texture στο Blender. Στο Material του αεροπλάνου προσθέτουμε ένα νέο Texture, διαλέγουμε την εικόνα που φτιάξαμε και την εφαρμόζουμε στο Mesh του αεροπλάνου.

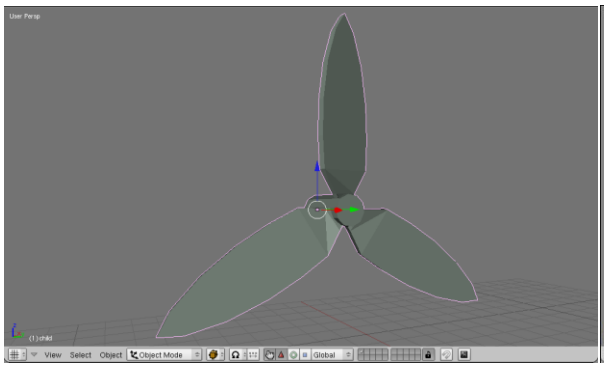


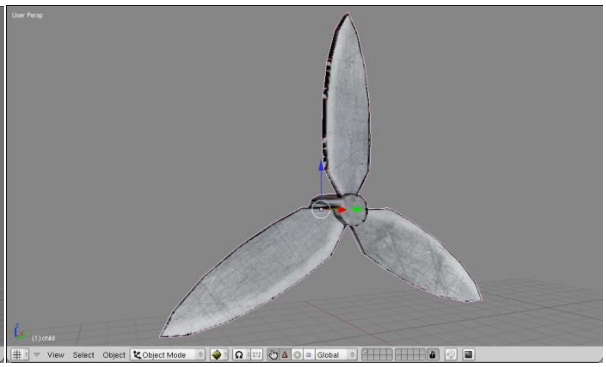


Η ίδια διαδικασία χρειάζεται και για τον έλικα.



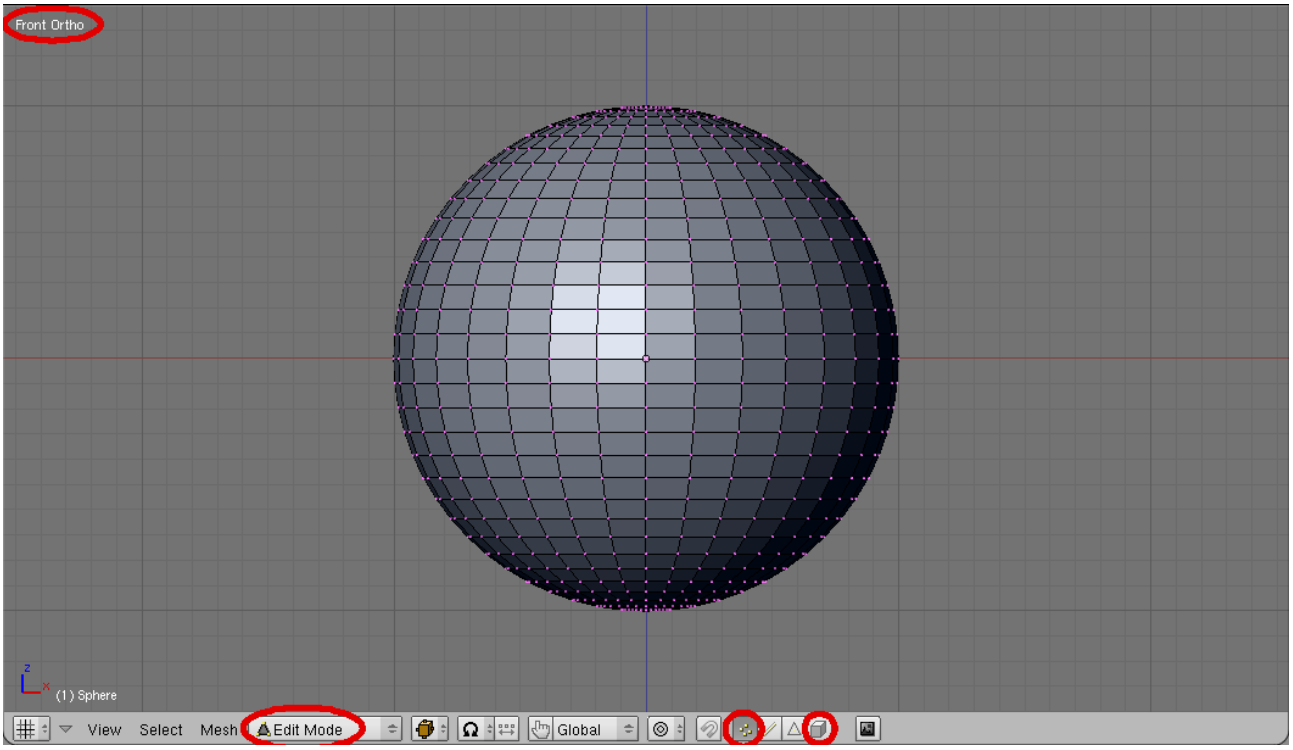




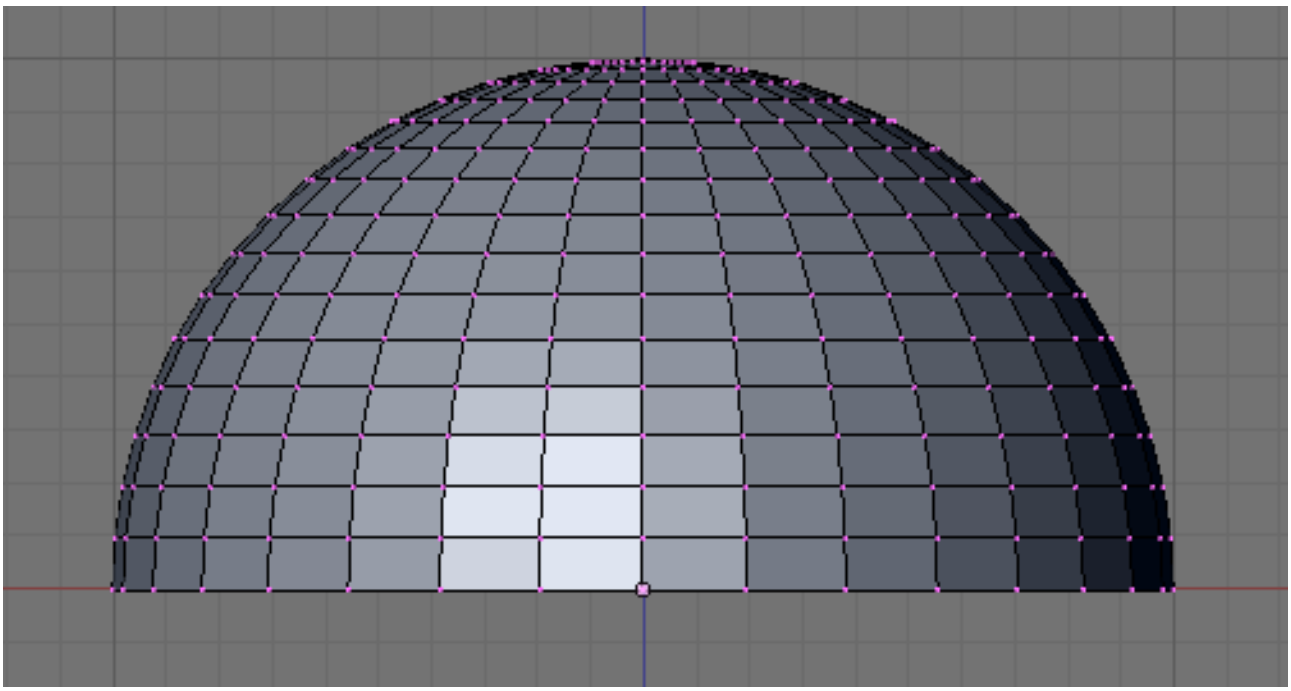


4.3 Δημιουργία του Dome

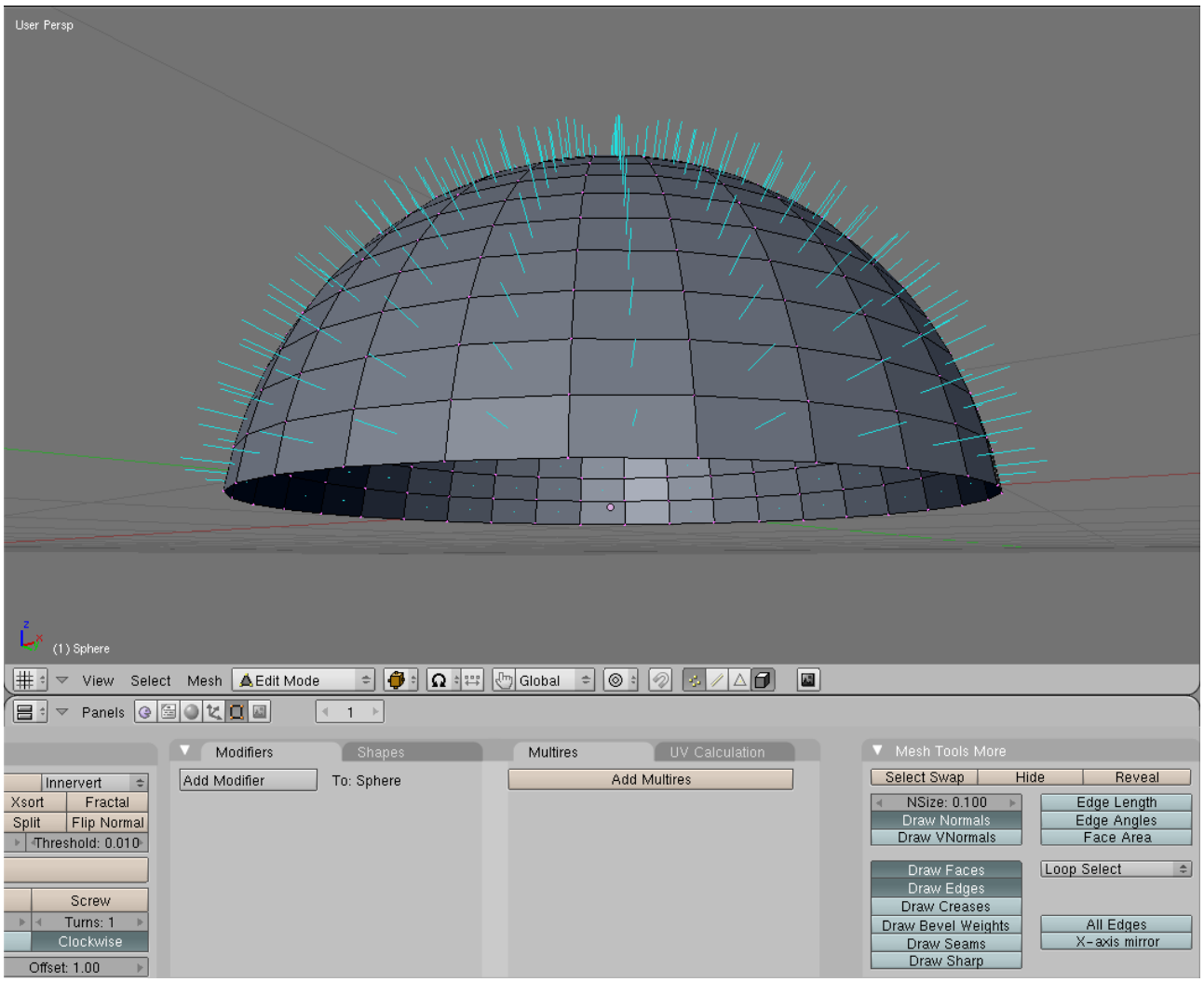
Η δημιουργία του Dome είναι αρκετά απλή διαδικασία εφόσον είναι ουσιαστικά μια σφαίρα χωρισμένη στη μέση. Για να το δημιουργήσουμε, από το μενού Add -> Mesh διαλέγουμε το UVSphere και στο παράθυρο που εμφανίζεται πατάμε OK. Στη συνέχεια πατάμε Tab για να μεταβούμε σε Edit Mode, από τις επιλογές από τη γραμμή μενού διαλέγουμε την επιλογή Vertex Select Mode, απενεργοποιούμε την επιλογή Occlude background geometry και διαλέγουμε μπροστινή ορθογραφική κάμερα(συντομεύσεις NumPad 1, NumPad 5).

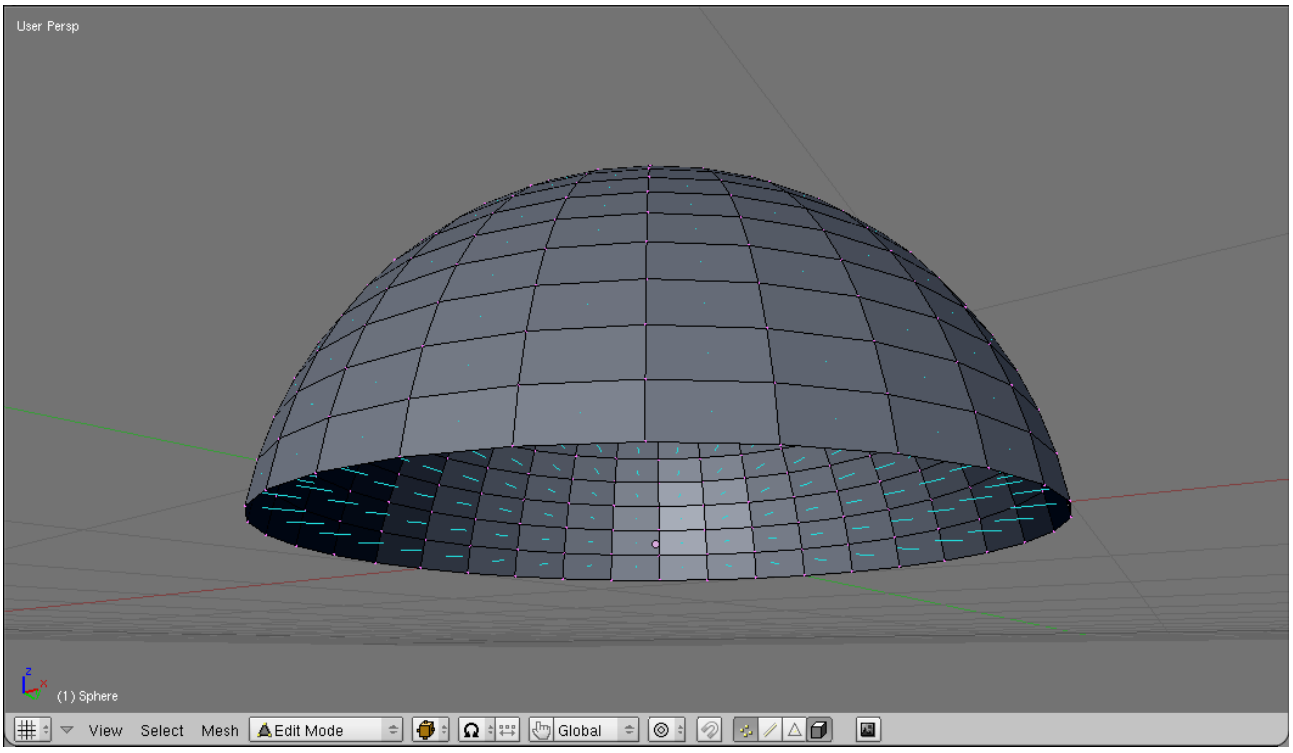


Αν υπάρχουν επιλεγμένα σημεία στην σφαίρα πιέζοντας το A τα αποεπιλέγουμε. Μετά με το πλήκτρο B αλλάζουμε σε Bounding Box selection mode και με drag&drop επιλέγουμε το κάτω μισό των vertices. Πατάμε Delete και από τη λίστα που εμφανίζεται επιλέγουμε Vertices.

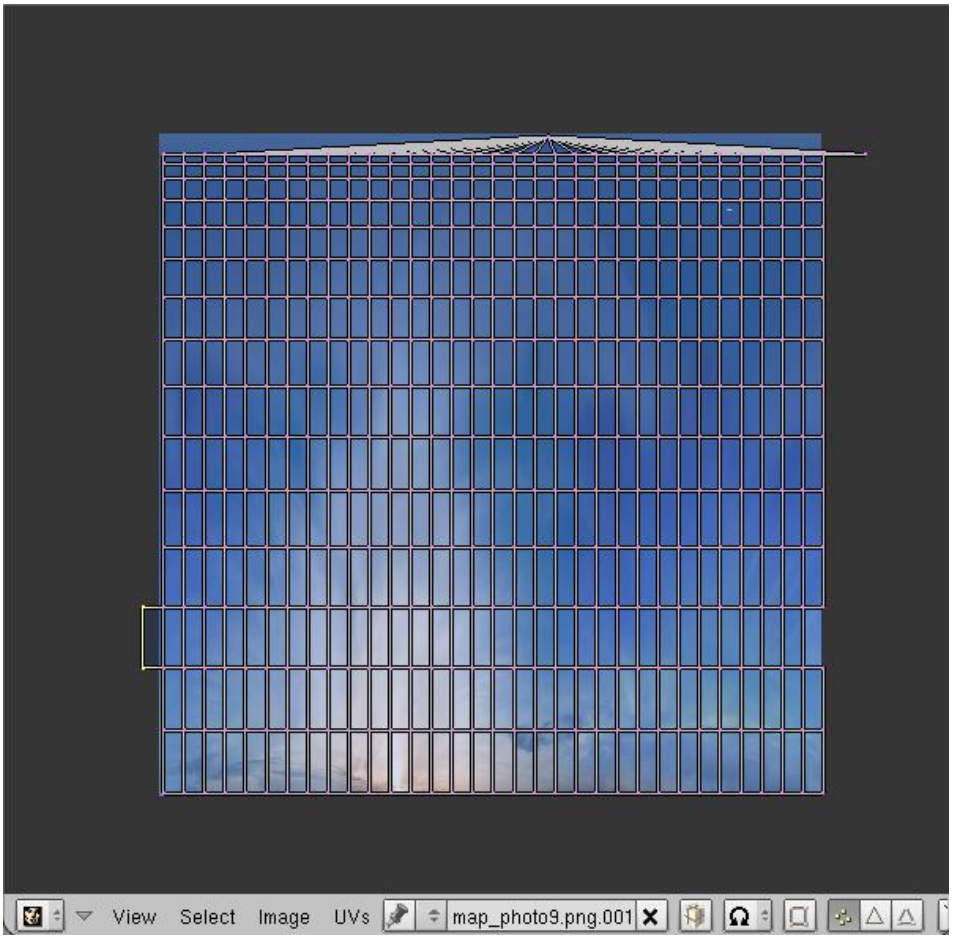


Στο Blender όπως και στα περισσότερα 3d πακέτα τα πολύγωνα είναι Single Sided, δηλαδή είναι ορατά μόνο από τη μια τους πλευρά. Το ποιά θα είναι αυτή η πλευρά ορίζεται από το προς τα πού είναι στραμμένα τα normals. Τα normals εξ'ορισμού είναι αόρατα, αλλά μπορούμε να τα εμφανίσουμε με τα εξής βήματα. Έχοντας επιλεγμένο το αντικείμενο πατάμε Tab για να μεταβούμε σε Edit Mode και από την καρτέλα Editing επιλέγουμε Draw Normals. Όπως φαίνεται και στην παρακάτω εικόνα τα normals του Dome μας είναι στραμμένα προς τα έξω αυτό σημαίνει ότι από τη μέσα μεριά είναι αόρατο. Επειδή τα αντικείμενα και οι κάμερες μας είναι μέσα στο Dome πρέπει να αλλάξουμε την φορά των normals. Πατάμε A για να επιλέξουμε όλα τα πολύγωνα και στη συνέχεια πατάμε Flip Normal. Τώρα τα normals είναι στραμμένα προς τα μέσα και είμαστε έτοιμοι για το επόμενο βήμα, που είναι το Texturing.

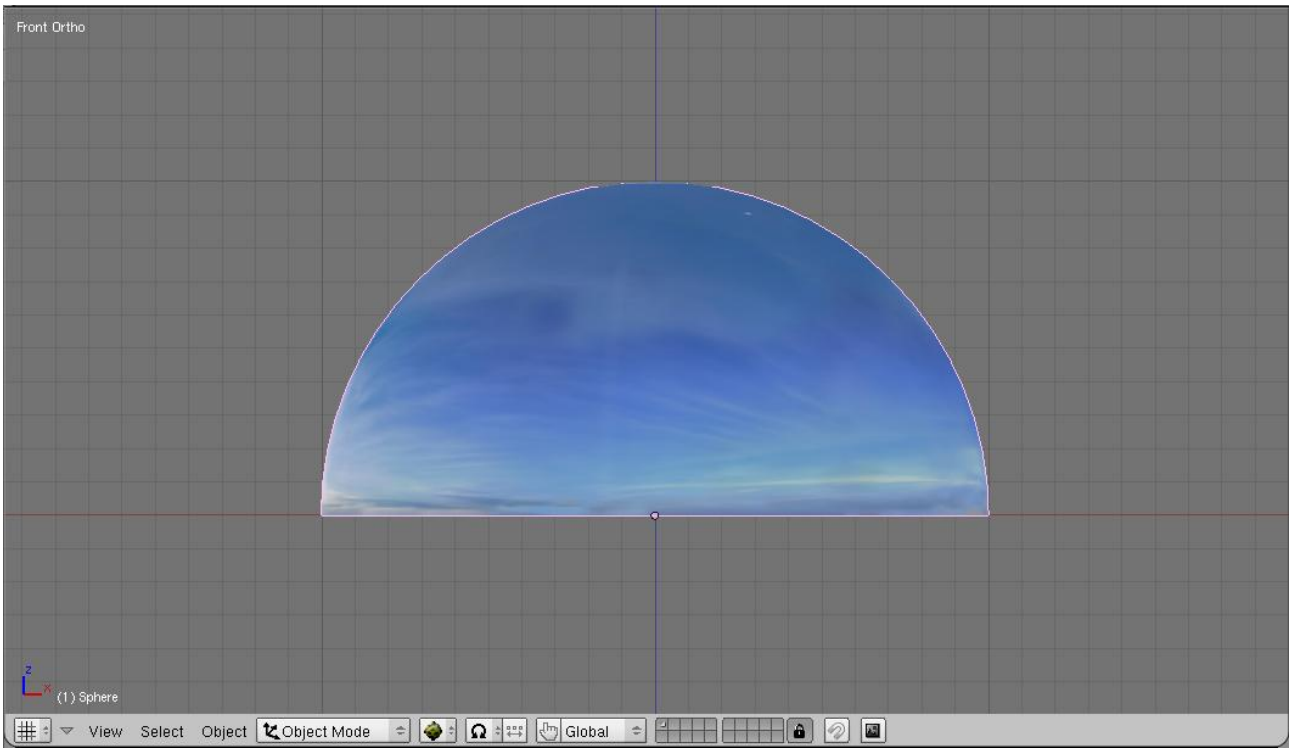

User Persp
(1) Sphere
View Select Mesh Edit Mode Global
Panels 1
Modifiers Shapes Multires UV Calculation
Add Modifier To: Sphere
Add Multires
Innervert
Xsort Fractal
Split Flip Normal
Threshold: 0.010
Screw
Turns: 1
Clockwise
Offset: 1.00
Mesh Tools More
Select Swap Hide Reveal
NSize: 0.100
Draw Normals
Draw VNormals
Edge Length
Edge Angles
Face Area
Draw Faces
Draw Edges
Draw Creases
Draw Bevel Weights
Draw Seams
Draw Sharp
Loop Select
All Edges
X-axis mirror

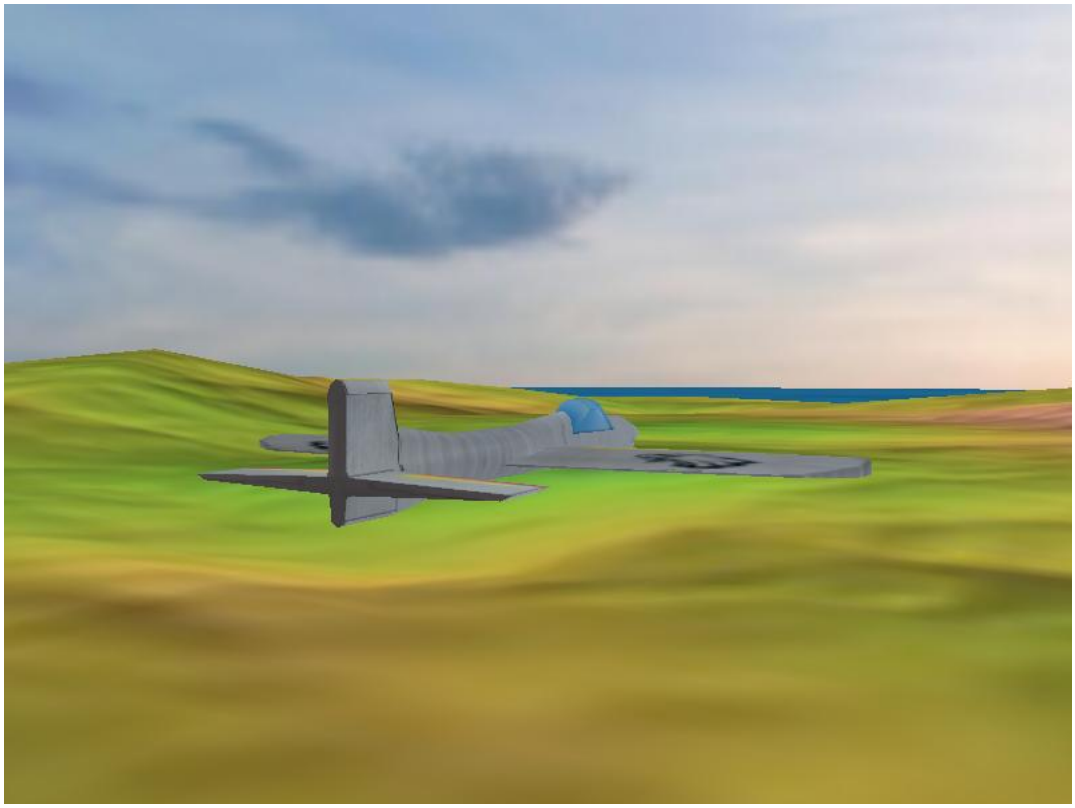

User Persp
(1) Sphere
View Select Mesh Edit Mode Global

Αρχικά φτιάχνουμε ένα Material, παρόμοιο μ'αυτό που φτιάξαμε για το Terrain και προσθέτουμε και κάποιο Texture από περιβάλλον(Environment ή Skymap). Υπάρχουν κάποια παραδείγματα Skymap στην παρακάτω διεύθυνση: Skymaps. Εδώ θα χρησιμοποιήσουμε κυλινδρικό UV Mapping από μπροστινή ορθογραφική προβολή. Αφού διαλέξουμε το Dome, με το πλήκτρο Tab μεταβαίνουμε σε Edit Mode και πατώντας το A διαλέγουμε όλα τα πολύγωνα. Για να αλλάξουμε την κάμερα σε μπροστινή ορθογραφική πατάμε NumPad 1 και μετά NumPad 5. Στη συνέχεια πατάμε U για να εμφανίσουμε το μενού UV Mapping και από τη λίστα που εμφανίζεται διαλέγουμε το Cylinder From View. Στο παράθυρο UV/Image Editor από τη λίστα με τα Texture διαλέγουμε το Skymap που κατεβάσαμε και τοποθετούμε το Map έτσι ώστε να καλύπτει όλη την εικόνα όπως φαίνεται παρακάτω.



Στις παρακάτω εικόνες φαίνεται το τελικό αποτέλεσμα.





5 Προγραμματισμός

5.1 Objective – C και SIO2 Engine

Εφόσον ολοκληρώσαμε το 3D κομμάτι της εργασίας θα περάσουμε στο κομμάτι του προγραμματισμού.

Ο προγραμματισμός του παιχνιδιού θα γίνει εξ'ολοκλήρου σε Objective - C και σε περιβάλλον X-Code. Η Objective – C είναι ένα υπερσύνολο της απλής C δηλαδή οποιοδήποτε πρόγραμμα γραμμένο σε C μπορεί να γίνει compile σε compiler της Objective – C, αλλά και να συμπεριληφθεί σε μια κλάση Objective – C. Η σύνταξη της Objective – C όσον αφορά τα αντικείμενα είναι βασισμένη στην γλώσσα προγραμματισμού Smalltalk. Ενώ η σύνταξη για μη αντικειμενοστραφείς εργασίες όπως εκφράσεις, μεταβλητές, δηλώσεις και κλήσεις συναρτήσεων/μεθόδων κλπ είναι πανομοιότυπη με αυτό της C. Το μοντέλο λειτουργίας της Objective - C αντικειμενοστραφούς προγραμματισμού βασίζεται ανταλλαγή μηνυμάτων με αντικείμενα. Στην Objective – C δεν καλούμε μεθόδους, στέλνουμε μηνύματα.

Σε μια εφαρμογή/παιχνίδι στο SIO2 ο κυρίως κώδικας βρίσκεται στο αρχείο template.mm που βρίσκεται στο φάκελο Other Sources. Στην αρχή του αρχείου βρίσκονται κάποια imports σχετικά με το game engine. Στη συνέχεια κάνουμε τις δηλώσεις των μεταβλητών και των σταθερών μας και έπειτα υλοποιούμε τις τρείς παρακάτω βασικές μεθόδους:

```
void templateLoading( void )
{
    …
}

void templateRender( void )
{
    …
}

void templateShutdown( void )
{
    …
}
```

5.2 Συνάρτηση templateLoading

Η `templateLoading( void )` είναι η πρώτη μέθοδος που καλείται όταν φορτώνει η εφαρμογή. Σε αυτήν τοποθετούμε αρχικοποιήσεις μεταβλητών, διαβάζουμε το αρχείο .sio2 και τοποθετούμε τα αντικείμενα που θα χρειαστούμε από αυτό σε μεταβλητές. Από αυτό επίσης φορτώνουμε τα materials, τα textures καθώς και τις φυσικές ιδιότητες των αντικειμένων. Παρακάτω φαίνεται ο κώδικας για την ανάγνωση ενός αρχείου .sio2:

```
sio2ResourceCreateDictionary( sio2->_SIO2resource );

    sio2ResourceOpen( sio2->_SIO2resource,
                          "filename.sio2", 1 );

    while( i != sio2->_SIO2resource->gi.number_entry )
    {
      sio2ResourceExtract( sio2->_SIO2resource, NULL );
      ++i;
    }

    sio2ResourceClose( sio2->_SIO2resource );

    sio2ResourceBindAllImages( sio2->_SIO2resource );

    sio2ResourceBindAllMaterials( sio2->_SIO2resource );

    sio2ResourceBindAllInstances( sio2->_SIO2resource );
```

```
sio2ResourceBindAllMatrix( sio2->_SIO2resource );

sio2ResourceBindAllPhysicObjects( sio2->_SIO2resource,
                                  sio2->_SIO2physic );

sio2ResourceGenId( sio2->_SIO2resource );

sio2ResetState();
```

Με την εκτέλεση αυτού του κώδικα φορτώνονται όλα τα στοιχεία του αρχείου .sio2 και αποθηκεύονται στο `_SIO2resource` το οποίο είναι δείκτης σε αντικείμενο τύπου `SIO2resource`. Για να διαβάσουμε από το `_SIO2resource` τα περιεχόμενα του χρησιμοποιούμε την μέθοδο `sio2ResourceGetObject( SIO2resource *_SIO2resource, char *_name )`. Σαν πρώτο όρισμα δίνουμε το `_SIO2resource` και σαν δεύτερο τον τύπο και το όνομα του αντικειμένου χωρισμένα με “/” όπως αυτά κατηγοριοποιούνται και τα έχουμε ονομάσει στο Blender. Παρακάτω φαίνονται μερικά παραδείγματα χρήσης της `sio2ResourceGetObject()`.

Φόρτωση αντικειμένων:

```
airplane = ( SIO2object * )sio2ResourceGetObject( sio2->_SIO2resource,
                                                  "object/air" );

child = ( SIO2object * )sio2ResourceGetObject( sio2->_SIO2resource,
                                               "object/fan" );

colObj1 = ( SIO2object * )sio2ResourceGetObject( sio2->_SIO2resource,
                                                 "object/dome" );
```

Τα air, fan και dome είναι τα ονόματα που έχουμε δώσει στα αντικείμενα στο Blender.

Φόρτωση και αρχικοποίηση κάμερας:

```
sio2->_SIO2camera = ( SIO2camera * )sio2ResourceGet( sio2->_SIO2resource,
                                                     SIO2_CAMERA, "camera/Camera" );


sio2Perspective( sio2->_SIO2camera->fov,
                 sio2->_SIO2window->scl->x / sio2->_SIO2window->scl->y,
```

```
                        sio2->_SIO2camera->cstart,
                        sio2->_SIO2camera->cend );

sio2CameraGetProjectionMatrix( sio2->_SIO2camera );
```

Εδώ θα δημιουργήσουμε την Parrent – Child σχέση ανάμεσα στα αεροπλάνο και τον έλικα. Ο λόγος που το κάνουμε είναι για να ακολουθεί ο έλικας όλες τις κινήσεις του αεροπλάνου, αλλά και να μπορεί να περιστρέφεται ανεξάρτητα.

```
makeChildOf(fan->_SIO2transform, airplane->_SIO2transform);
```

Εδώ θα γίνουν επίσης αρχικοποιήσεις για Collision, Physics, Widgets και Particles.

5.3 Συνάρτηση templateRender

Η συνάρτηση `templateRender( void )` καλείται σε κάθε Frame δηλαδή 30 – 60 φορές το δευτερόλεπτο και είναι πολύ βασικό σημείο της εφαρμογής. Από δω καθορίζονται η κινήσεις και οι συμπεριφορές των αντικειμένων που βρίσκονται μέσα στην σκηνή. Εδώ ο κώδικας χωρίζεται σε δύο τμήματα, το 3D και το 2D. Όλες οι εντολές που αφορούν τα 3D αντικείμενα όπως το αεροπλάνο, κάμερες κλπ μπαίνουν ανάμεσα στις εντολές `sio2WindowEnterLandscape3D()` και `sio2WindowLeaveLandscape3D()`. Ενώ όσες αφορούν τα 2D αντικείμενα όπως 2D κείμενο, widgets κλπ μπαίνουν ανάμεσα στις εντολές: `sio2WindowEnter2D()` και `sio2WindowLeave2D()`.

Οι παρακάτω εντολές είναι απαραίτητο να καλούνται σε κάθε frame για την απεικόνιση των 3D περιεχομένων της σκηνής (Render):

```
sio2CameraRender( sio2->_SIO2camera );

sio2PhysicRender( sio2->_SIO2physic,
                        1.0f/60.0f, 1 );

glClear( GL_DEPTH_BUFFER_BIT | GL_COLOR_BUFFER_BIT );

sio2CameraUpdateFrustum( sio2->_SIO2camera );

sio2CameraGetModelviewMatrix( sio2->_SIO2camera );

sio2ResourceCull( sio2->_SIO2resource, sio2->_SIO2camera );

sio2ResourceRender( sio2->_SIO2resource,
                            sio2->_SIO2window,
                            sio2->_SIO2camera,
                            SIO2_RENDER_SOLID_OBJECT |
SIO2_RENDER_TRANSPARENT_OBJECT | SIO2_RENDER_CLIPPED_OBJECT |
SIO2_EVALUATE_SENSOR | SIO2_RENDER_EMITTER);

sio2MaterialReset();
```

5.3.1 Κίνηση του αεροπλάνου.

Η κίνηση του αεροπλάνου γίνεται μέσω των επιταχυνσιόμετρων και της οθόνης αφής της συσκευής. Μέσω της οθόνης αφής κουνώντας το δάκτυλο πάνω ή κάτω θα επιταχύνουμε και θα επιβραδύνουμε το αεροπλάνο και με τα επιταχυνσιόμετρα θα αλλάζουμε την κατεύθυνση της κίνησης. Υπάρχουν τρείς μέθοδοι για την ανάγνωση των τιμών του επιταχυνσιόμετρου και της οθόνης αφής, οι οποίες φαίνονται παρακάτω:

```
void templateScreenTap( void *_ptr, unsigned char _state )
{
    …
}

void templateScreenTouchMove( void *_ptr )
{
    …
}

void templateScreenAccelerometer( void *_ptr )
{
    …
}
```

Η `templateScreenTap` καλείται μόλις ακουμπήσει κάποιος με ένα η περισσότερα δάκτυλα την οθόνη και μας δίνει τις συντεταγμένες απ'όλα τα σημεία επαφής.

Με τον παρακάτω κώδικα μπορούμε να αποθηκεύσουμε το πρώτο σημείο στην μεταβλητή start.

```
start.x = sio2->_SIO2window->touch[ 0 ]->x;
start.y = sio2->_SIO2window->touch[ 0 ]->y;
```

Η `templateScreenTouchMove` εκτελείται μετά την `templateScreenTap` κάθε φορά που κινείται το δάκτυλο πάνω στην οθόνη και περιέχει τις συντεταγμένες της κάθε επαφής κατά τη διάρκεια της κίνησης. Αν αφαιρέσουμε από την τρέχουσα θέση την θέση του start θα δούμε πόσο μεγάλη ή απότομη είναι η κίνηση σε κάθε άξονα.

Ο κώδικας παρακάτω βρίσκει αυτή την διαφορά και ανάλογα αυξάνει η μειώνει τη μεταβλητή xSpeed η οποία καθορίζει την ταχύτητα του αεροπλάνου.

```
float e = sio2->_SIO2window->touch[ 0 ]->y - start.y;

if (xSpeed>=10 and xSpeed<=50) {
    xSpeed += 0.004*e;
}
```

Η `templateScreenAccelerometer` καλείται κάθε φορά που αλλάζει γωνία η συσκευή και μας δίνει τις γωνίες στους άξονες x και y:

sio2->_SIO2window->accel->x

sio2->_SIO2window->accel->y

Οι τιμές αυτές έχουν εύρος από -1.0 μέχρι 1.0, με το 0 να είναι το σημείο ισορροπίας της συσκευής και -1 και 1 όταν η συσκευή είναι στραμμένη -90 και 90 μοίρες αντίστοιχα στον κάθε άξονα. Την ώρα όμως που παίζει κάποιος το παιχνίδι δεν μπορεί να στρίβει 90 μοίρες τη συσκευή γιατί δεν θα μπορεί να βλέπει την οθόνη. Η μέγιστη γωνία της συσκευής σε πραγματικές συνθήκες θα είναι περίπου 40 - 45 μοίρες, δηλαδή τιμές από -0.45 έως 0.45. Πρέπει να μετατρέψουμε το -0.45 – 0.45 σε μοίρες και στρίβει το αεροπλάνο με βάση αυτές. Μετά από κάποιες δοκιμές κατέληξα στους 2 παρακάτω τύπους:

```
xAngle = (xAngle * 0.3) + 0.7 * sio2->_SIO2window->accel->x * -145.0;
yAngle = (yAngle * 0.3) + 0.7 * sio2->_SIO2window->accel->y * 200.0;
```

Το `(xAngle * 0.3) + 0.7 * …` μπροστά είναι ένα είδος φίλτρου για να έχουμε

πιο ομαλή κίνηση. Στην ουσία αυτό που κάνει το φίλτρο είναι να πάρει το 30% της προηγούμενης τιμής και το 70 % της καινούριας για να μην είναι πολύ απότομες οι στροφές και να φαίνεται πιο αληθινή η κίνηση.

Επειδή η συσκευή δεν μπορεί να μας δώσει την τιμή της γωνίας στον Z άξονα θα την δημιουργήσουμε με μαθηματικά:

```
zAngle = zAngle + 3 * sin(xAngle/SIO2_RAD_TO_DEG) *
                        (sin((yAngle-30)/SIO2_RAD_TO_DEG) )) + 0.3 * zAngle;
```

Και προσθέτοντας το φίλτρο έχουμε:

```
zAngle = zAngle * 0.3 * 0.7 * (zAngle + 3 * sin(xAngle/SIO2_RAD_TO_DEG) *
(sin((yAngle-30)/SIO2_RAD_TO_DEG) )) + 0.3 * zAngle;
```

Για να ανανεώσουμε την κλήση του αεροπλάνου σε κάθε frame θα προσθέσουμε στο `templateRender( void )` τις παρακάτω γραμμές:

```
airplane->_SIO2transform->rot->z = zAngle;
airplane->_SIO2transform->rot->x = xAngle;
airplane->_SIO2transform->rot->y = yAngle;
```

Όσον αφορά την κίνηση του αεροπλάνου προς τα μπρος θα την κάνουμε μέσα από τις βιβλιοθήκες φυσικής για να μπορέσουμε να ανιχνεύσουμε και τα collisions. Η μέθοδος που θα χρησιμοποιήσουμε είναι η

`void setLinearVelocity(const btVector3& lin_vel)` η οποία ορίζει την ταχύτητα ενός αντικειμένου σε μια δεδομένη χρονική στιγμή με βάση ένα διάνυσμα (vector) στον τρισδιάστατο χώρο. Το διάνυσμα αυτό θα το δημιουργήσουμε με βάση την κλήση του αεροπλάνου.

```
dirZ = sin( yAngle/SIO2_RAD_TO_DEG ) * cos( xAngle/SIO2_RAD_TO_DEG );
dirY = cos( xAngle/SIO2_RAD_TO_DEG ) * sin( zAngle/SIO2_RAD_TO_DEG );
dirX = cos( zAngle/SIO2_RAD_TO_DEG ) * cos( yAngle/SIO2_RAD_TO_DEG );
```

Τα `dirX, dirY` και `dirZ` ορίζουν το μοναδιαίο διάνυσμα της κατεύθυνσης του αεροπλάνου. Αυτά πρέπει να τα πολλαπλασιαστούν με την ταχύτητα (xSpeed) για να πάρουμε το τελικό διάνυσμα. Οπότε η τελική συνάρτηση για την κίνηση του αεροπλάνου είναι η παρακάτω.

```
airplane->_SIO2objectphysic->_btRigidBody->setLinearVelocity( btVector3( xSpeed
* dirX, xSpeed * dirY, -xSpeed * dirZ));
```

5.3.2 Κίνηση του έλικα.

Η κίνηση που πρέπει να κάνει ο έλικας είναι να περιστρέφεται ανάλογα με την ταχύτητα του αεροπλάνου. Για να το πετύχουμε χρειάζονται οι παρακάτω εντολές στο `templateRender(void)`.

```
sio2TransformGetLocFromMatrix(child->_SIO2transform);
child->_SIO2transform->rot->x += xSpeed;

sio2TransformBindMatrix( child->_SIO2transform );
```

5.3.3 Κίνηση της κάμερας.

Σε εφαρμογές Flight Simulator καταλυτικής σημασίας για την καλύτερη εξομοίωση είναι και η κίνηση της κάμερας. Το πρώτο πράγμα που θα σκέπτονταν κανείς για την υλοποίηση αυτής είναι να δημιουργήσει μια σχέση Parrent – Child ανάμεσα στο αεροπλάνο και την κάμερα. Αυτή η ιδέα δουλεύει αλλά το αποτέλεσμα δεν είναι αυτό που θα περίμενε κανείς και δεν είναι ικανοποιητικό. Δημιουργεί την αίσθηση ότι η κάμερα είναι κολλημένη στο αεροπλάνο και η εικόνα είναι πολύ στατική. Μια πιο σωστή μέθοδος η οποία χρησιμοποιείται και από πολλά Flight Simulator είναι να αποθηκεύσουμε σε ένα πίνακα τις τελευταίες συντεταγμένες του αεροπλάνου και η κάμερα να ακολουθεί αυτές, δηλαδή να κάνει την ίδια διαδρομή με το αεροπλάνο αλλά με λίγη καθυστέρηση. Σε συνδυασμό με αυτό όμως η κάμερα θα εστιάζει πάντα στο αεροπλάνο. Η απόσταση της κάμερας από το αεροπλάνο εξαρτάται από δύο παράγοντες: Το πόσα ‘βήματα’ πίσω βρίσκεται η κάμερα από το αεροπλάνο και πόσο γρήγορα τρέχει το αεροπλάνο. Όσο πιο γρήγορα τρέχει το αεροπλάνο τόσο μεγαλώνει η απόσταση. Οπότε αν θέλουμε να κρατήσουμε σταθερή την απόσταση όταν επιταχύνει το αεροπλάνο πρέπει να λιγοστέψουμε τα βήματα και αντίστροφα.

Παρακάτω φαίνεται ο κώδικας για την αποθήκευση προηγούμενων θέσεων του αεροπλάνου και την μετακίνηση της κάμερας στις θέσεις αυτές. Οι μεταβλητές i και j είναι μετρητές και καθορίζουν το πόσες θέσεις πριν θα βρίσκεται η κάμερα και συνεπώς και την απόσταση της κάμερας από το αεροπλάνο.

```
prevPosiotions[i].x = airplane->_SIO2transform->loc->x;
prevPosiotions[i].y = airplane->_SIO2transform->loc->y;
prevPosiotions[i].z = airplane->_SIO2transform->loc->z;

sio2->_SIO2camera->_SIO2transform->loc->x = prevPosiotions[j].x;
sio2->_SIO2camera->_SIO2transform->loc->y = prevPosiotions[j].y;
sio2->_SIO2camera->_SIO2transform->loc->z = airplane->_SIO2transform->loc->z +
1.0f;
```

Εδώ φαίνεται ο κώδικας για την εστίαση της κάμερας στο αεροπλάνο.

```
airPos.x = _SIO2object->_SIO2transform->loc->x;
airPos.y = _SIO2object->_SIO2transform->loc->y;
airPos.z = _SIO2object->_SIO2transform->loc->z;

sio2Vec3Diff( &objectPos, sio2->_SIO2camera->_SIO2transform->loc,
                          sio2->_SIO2camera->_SIO2transform->dir );

sio2Normalize( sio2->_SIO2camera->_SIO2transform->dir,
                          sio2->_SIO2camera->_SIO2transform->dir );
```

5.4 Widgets.

Τα Widgets είναι δισδιάστατες εικόνες που μπορούν να τοποθετηθούν σε οποιοδήποτε σημείο της οθόνης πάνω από τα 3d γραφικά. Μπορούν να χρησιμοποιηθούν σαν κουμπιά, ενδείξεις ή και τα 2 μαζί. Στο συγκεκριμένο παιχνίδι θα προσθέσουμε 3 widgets. Το πρώτο θα μας δείχνει την κλήση του αεροπλάνου στον Y άξονα και αν πατήσουμε πάνω του θα μηδενίζει, δηλαδή θα θεωρήσει σαν 0 μοίρες την κλήση που έχει η συσκευή εκείνη την στιγμή. Το δεύτερο θα είναι ένα κοντέρ, δηλαδή θα δείχνει την ταχύτητα του αεροπλάνου και το τρίτο κουμπί θα αλλάζει την κάμερα.

Για να προσθέσουμε ένα widget χρειάζονται τα εξής βήματα:

1. Δήλωση των widget στην αρχή του κώδικα.

```
SIO2widget *tiltBackgroundWidget;

SIO2widget *tiltWidget;

SIO2widget *speedWidget;

SIO2widget *arrowWidget;

SIO2widget *cameraWidget;
```

Όπως φαίνεται τα πρώτα 2 widget αποτελούνται από 2 κομμάτια. Ένα φόντο το οποίο μένει σταθερό και ένα αεροπλάνο ή βέλος που γυρίζουν ανάλογα με την κλήση ή την ταχύτητα του αεροπλάνου.

2. Αρχικοποίηση των widget και φόρτωση των εικόνων.

Ο κώδικας για την αρχικοποίηση ενός widget μπαίνει στο `templateLoading(void)`. Εδώ ορίζεται από ποιό αρχείο θα φορτώσει την εικόνα, το μέγεθος και η θέση της, αν θα καλείται κάποια συνάρτηση όταν το πατάμε και ποιά θα είναι αυτή. Παρακάτω φαίνεται η αρχικοποίηση του `tiltWidget:`

```
SIO2stream *yTiltWidgetStream = NULL;
SIO2image *tiltplaneImage = NULL;
SIO2material *tiltWidgetMaterial = NULL;

yTiltWidgetStream = sio2StreamOpen( "planetilt.png", 1 );
if( yTiltWidgetStream )
{
    tiltplaneImage = sio2ImageInit( "planetilt.png" );
    {
      sio2ImageLoad( tiltplaneImage, yTiltWidgetStream );
      sio2ImageGenId( tiltplaneImage, NULL, 0.0f );
    }

    yTiltWidgetStream = sio2StreamClose( yTiltWidgetStream );
```

```
    tiltWidgetMaterial = sio2MaterialInit( "planetilt" );

{
    tiltWidgetMaterial->blend = SIO2_MATERIAL_COLOR;
    tiltWidgetMaterial->_SIO2image[ SIO2_MATERIAL_CHANNEL0 ] = tiltplaneImage;
}

    tiltWidget = sio2WidgetInit( "planetilt" );
    tiltWidget->_SIO2material = tiltWidgetMaterial;
    tiltWidget->_SIO2transform->scl->x = 64.0f;
    tiltWidget->_SIO2transform->scl->y = 64.0f;
    tiltWidget->_SIO2transform->loc->x  = 128.0f;
    tiltWidget->_SIO2transform->loc->y  = 32.0f;

    tiltWidget->area->x = 64;
    tiltWidget->area->y = 64;

    tiltWidget->_SIO2widgettapdown = resetPlaneTilt;

    sio2EnableState( &tiltWidget->flags,
                     SIO2_WIDGET_VISIBLE | SIO2_WIDGET_CENTERED |
                     SIO2_WIDGET_ENABLED );
```

}

Οι τρείς παρακάτω γραμμές χρειάζονται μόνο σε περίπτωση που θέλουμε να συνδέσουμε το Widget με ενέργεια, δηλαδή όταν το πατάμε να καλείται μια συνάρτηση. Οι δύο πρώτες γραμμές ορίζουν την περιοχή που θα δέχεται το πάτημα και η τελευταία γραμμή ορίζει το όνομα της συνάρτησης που θα καλέσει.

```
    tiltWidget->area->x = 64;
    tiltWidget->area->y = 64;

    tiltWidget->_SIO2widgettapdown = resetPlaneTilt;
```

3. Απεικόνιση του widget (Render).

Για την απεικόνιση των widget χρειάζονται οι παρακάτω γραμμές στην

```
void templateRender( void )
{
    sio2WindowEnter2D( sio2->_SIO2window, 0.0f, 1.0f );

     .........

    sio2TransformBindMatrix( tiltWidget->_SIO2transform );
    sio2WidgetRender( tiltWidget, sio2->_SIO2window, 1 );
    sio2WidgetReset();
    sio2MaterialReset();
}
```

Σ’αυτό το σημείο του κώδικα κάνουμε και τα Transformations(μετακίνηση, στροφή κλπ) των widget.

Π.χ. στο δεύτερο Widget (δείκτης ταχύτητας) για να το κάνουμε να στρίβει ανάλογα την ταχύτητα του αεροπλάνου χρειάζονται τα παρακάτω:

```
arrowWidget->_SIO2transform->rot->z = -xSpeed * 4.0f + arc4random() % 7;
sio2TransformBindMatrix( arrowWidget->_SIO2transform );
```

Προσθέτοντας το `arc4random() % 7` στην γωνία περιστροφής δημιουργούμε ένα τυχαίο “τρέμολο” στον δείκτη κάνοντας το να φαίνεται πιο αληθινό.

5.5 Collisions

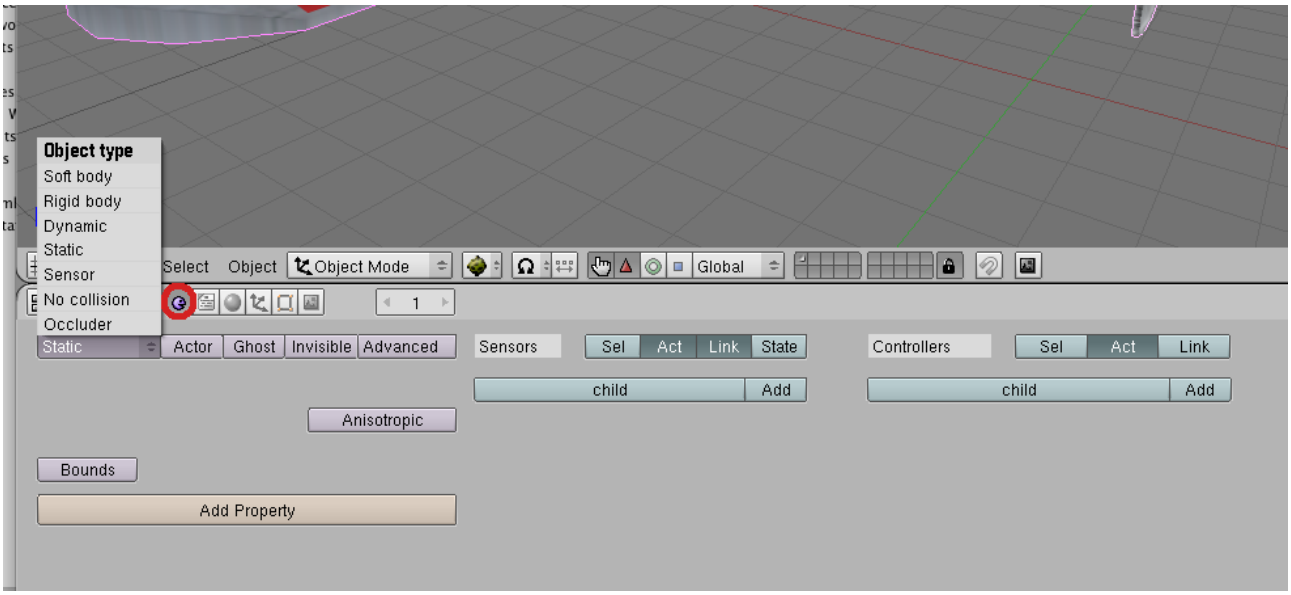
Λέγοντας collision εννοούμε την αντίδραση των διάφορων αντικειμένων στο παιχνίδι όταν συγκρούονται μεταξύ τους. Στην μηχανή SIO2Engine τα collision ανιχνεύονται μόνο μέσω την Bullet και για να γίνει αυτό πρέπει να έχουμε ενεργοποιημένες τις ιδιότητες φυσικής των αντικειμένων στο Blender.

Οι ιδιότητες φυσικής βρίσκονται στην καρτέλα Logic. Η πρώτη επιλογή σ'αυτή την καρτέλα είναι ο τύπος του φυσικού αντικειμένου. Από τις επτά επιλογές που είναι διαθέσιμες μόνο οι τρείς είναι συμβατές με το SIO2. Αυτές είναι:

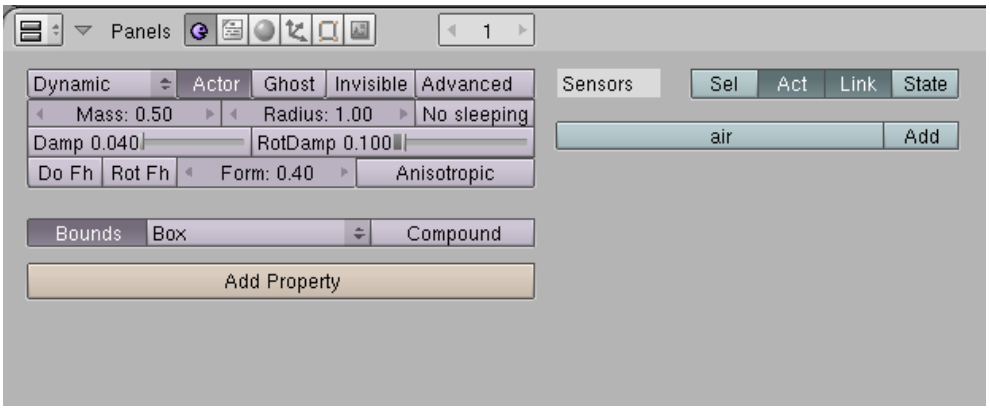
Static: Το αντικείμενο είναι σταθερό σε κάποιο σημείο, όπως ένα κτίριο, έδαφος κλπ.

Rigid body: Το αντικείμενο μπορεί κινείται και σε συγκρούσεις να συμπεριφέρεται σαν ένα κανονικό αντικείμενο.

Dynamic: Το αντικείμενο κινείται, αλλά σε συγκρούσεις επηρεάζεται μόνο η ταχύτητα/επιτάχυνση και όχι η περιστροφή του.



Αφού διαλέξουμε τον τύπο του αντικειμένου πατάμε το Actor για να το ενεργοποιήσουμε. Αν διαλέξουμε Rigid body ή Dynamic εμφανίζονται περισσότερες επιλογές όπως μάζα, ελαστικότητα κλπ.



Για να καταχωρήσουμε ένα collision προσθέτουμε στην `templateLoading(void)` το παρακάτω:

```
sio2SensorInitCollision( "sensor1", colObj1, ariplane, collision1 );
```

Το οποίο ορίζει τα 2 αντικείμενα που θα συγκρουστούν και την συνάρτηση που θα καλείται όταν συγκρουστούν. Η συνάρτηση ορίζεται ως εξής:

```
void collision1(void *_SIO2sensor)
{
    ...
    ...
}
```

Δηλαδή όταν συγκρουστεί το colObj1 με το airplane θα καλείται η συνάρτηση με το όνομα collision1.

ΠΗΓΕΣ

1. http://www.google.com
2. http://el.wikipedia.org/wiki/IPhone
3. http://el.wikipedia.org/wiki/Blender
4. http://en.wikipedia.org/wiki/Xcode
5. http://www.develop-online.net/news/32250/The-top-10-game-engines-revealed
6. http://en.wikipedia.org/wiki/Game_engine
7. http://maniacdev.com/2009/07/a-look-at-8-different-iphone-game-engines-2d-and-3d/
8. http://el.wikipedia.org/wiki/Photoshop
9. http://www-roc.inria.fr/gamma/gamma/download/AIRCRAFT/index0.php
10. http://en.wikipedia.org/wiki/Objective-C
11. http://developer.apple.com/library/ios/#documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/Introduction/Introduction.html#//apple_ref/doc/uid/TP40007898-CH1-SW1