ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Διπλωματική Εργασία

ΘΕΡΜΙΚΗ ΠΡΟΣΟΜΟΙΩΣΗ ΚΤΙΡΙΩΝ

ΛΙΟΝΤΗΡΗΣ ΑΝΔΡΕΑΣ

Επιβλέπων Καθηγητής : Καθηγητής Καλαϊτζάκης Κωνσταντίνος
Εξεταστική Επιτροπή: Καθηγητής Καλαϊτζάκης Κωνσταντίνος
Αναπληρωτής Καθηγητής Χαλκιαδάκης Γεώργιος
Επίκουρη Καθηγήτρια Κολοκοτσά Διονυσία

Χανιά, Οκτώβριος 2016

Στην οικογένειά μου…

Ευχαριστίες

Θα ήθελα να ευχαριστήσω

Τον καθηγητή κύριο Καλαϊτζάκη Κωνσταντίνο, για την πολύτιμη βοήθεια και καθοδήγησή του σε όλη τη διάρκεια της εκπόνησης της διπλωματικής μου εργασίας.

Τον κύριο Γομπάκη Κωνσταντίνο για τη συνεισφορά του στην εκπόνηση της εργασίας καθώς και την παροχή των χρήσιμων γνώσεών του .

Τον αναπληρωτή καθηγητή κύριο Χαλκιαδάκη Γεώργιο και την επίκουρη καθηγήτρια κυρία Κολοκοτσά Διονυσία για τη συνεισφορά τους ως μέλη της εξεταστικής επιτροπής.

Τη φίλη μου Καστάνη Ιωάννα για όλη τη δύναμη και το κουράγιο που μου προσέφερε μέχρι το τέλος της εργασίας.

Το φίλο μου Μιχελακάκη Παναγιώτη για τη βοήθειά του και τη φιλοξενία του καθ' όλη τη διάρκεια εκπόνησης της εργασίας μου.

Τη φίλη μου Αλεβυζάκη Νίκη για την αμέριστη βοήθεια της και στήριξή της κατά τη διάρκεια των σπουδών μου.

Την οικογένεια μου και όλους μου τους φίλους για την αγάπη ,τη στήριξη και την εμπιστοσύνη που μου έδειξαν όλα τα χρόνια της φοιτητικής μου ζωής.

Περίληψη

Η ενεργειακή προσομοίωση κτιρίων αποτελεί ένα πολύ σημαντικό εργαλείο το οποίο επιτρέπει την πρόγνωση της ενεργειακής απόδοσης κτιρίων ήδη από το στάδιο της σχεδίασης τους καθώς και την αξιολόγηση παρεμβάσεων που αποσκοπούν στη βελτίωση της ενεργειακής συμπεριφοράς ήδη υφιστάμενων κτιρίων. Τα τελευταία 50 χρόνια έχουν ανακαλυφθεί – αναπτυχθεί και διατεθεί στην αγορά εκατοντάδες λογισμικά τα οποία υλοποιούν ενεργειακές προσομοιώσεις κτιρίων. Στην παρούσα εργασία πραγματοποιείται η συσχέτιση ενός σχεδιαστικού προγράμματος δημιουργίας 3D μοντέλων ευρείας χρήσης (*SketchUp*) και ενός προγράμματος ενεργειακής προσομοίωσης ανοικτού κώδικα (*ESP-r*) με τη δημιουργία και χρήση plug-in το οποίο αποσκοπεί στην διευκόλυνση στο σκέλος της σχεδίασης. Το plug-in το οποίο δημιουργήθηκε χρησιμοποιώντας τη γλώσσα προγραμματισμού Ruby επιτρέπει την εξαγωγή γεωμετρικών στοιχείων κτιριακών μοντέλων από αρχεία εικόνας 3D που σχετίζονται με το SketchUp (.skp) και δημιουργεί αρχεία μορφής GEO τα οποία είναι συμβατά με το ESP-r, παρέχοντάς μας τη δυνατότητα ενεργειακής προσομοίωσης τους μέσω του ενεργειακού προσομοιωτή ESP-r. Ακόμη εξυπηρετεί την αντίστροφη διαδικασία επιτρέποντάς μας την εισαγωγή μοντέλων μορφής GEO στη σκηνή του SketchUp με στόχο την ευκολότερη επεξεργασία των γεωμετρικών τους στοιχείων. Αρχικά παρουσιάζονται οι δομές οι οποίες κατανοήθηκαν για την υλοποίηση του plug-in, στη συνέχεια η μεθοδολογία αντιμετώπισης των ζητημάτων που ανέκυψαν μέχρι την τελική υλοποίηση και τέλος παρατίθεται ένας οδηγός χρήσης του plug-in.

Abstract

The energy simulation of buildings is an important tool for predicting the energy efficiency of buildings in the early stage of their design and evaluating interventions aimed at improving the energy performance of existing buildings. The last 50 years have seen the development and marketing hundreds of software platforms which implement building energy simulations. In this thesis we designed a plug-in which allows the communication between a 3D modeling computer program used for a wide range of drawing applications (*SketchUp*) and an open source program of energy simulation (*ESP-r*) aiming to make easier the part of buildings design. The plug-in which designed in the programming language Ruby exports the geometry of 3D building models designed in SkethUp, to GEO file format which is compatible with ESP-r and allows us to simulate the building models. Furthermore it serves the reverse process allowing us to import GEO files in SketchUp scene for further processing. This thesis describes in length the proper structures we examined, the design of the plug-in and the issues that emerged. Finally it quotes a manual of the plug-in.

Περιεχόμενα

1. Εισαγωγή

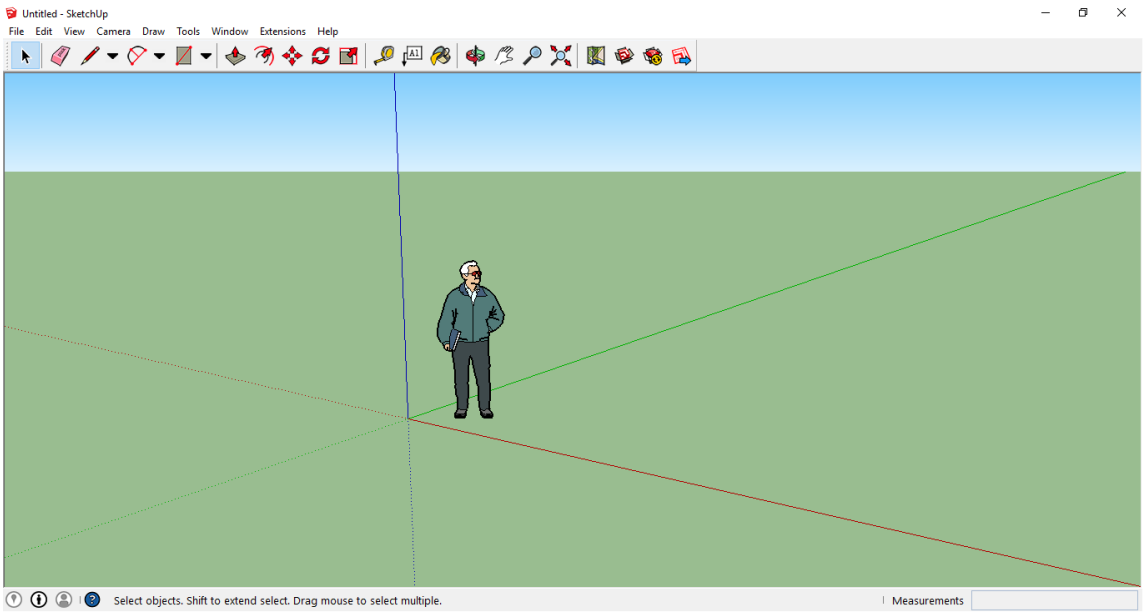
Στα πλαίσια της παρούσας εργασίας δημιουργήθηκε ένα plug-in το οποίο θα έκανε τη διεπαφή του προγράμματος σχεδιασμού SketchUp και του εργαλείου ενεργειακής προσομοίωσης ESP-r ευκολότερη. Η δημιουργία μοντέλων στο ESP-r είναι εξαιρετικά δύσχρηστη και δύστροπη. Αντίθετα, το SketchUp είναι σχεδιασμένο για τη δημιουργία 3D (τρισδιάστατων) μοντέλων παρέχοντας στο χρήστη πληθώρα εργαλείων καθώς και ένα ιδιαίτερα φιλικό γραφικό περιβάλλον. Συνεπώς, ο κύριος σκοπός της δημιουργίας της διεπαφής είναι η δυνατότητα κατασκευής και επεξεργασίας 3D μοντέλων στο SketchUp και η άμεση εισαγωγή τους στο ESP-r ώστε η διαδικασία της κατασκευής και ενεργειακής προσομοίωσης να γίνει ευκολότερη και αποτελεσματικότερη.

2. Περιγραφή Περιβάλλοντος

Γενικά, πριν αρχίσει η ανάπτυξη της διεπαφής αυτής και ακολουθώντας τους κανόνες καλής κατασκευής λογισμικού έγινε μια έρευνα σχετικά με το τι υπάρχει, καθώς και τι εργαλεία θα χρησιμοποιηθούν για την ανάπτυξή της.

2.1 SketchUp

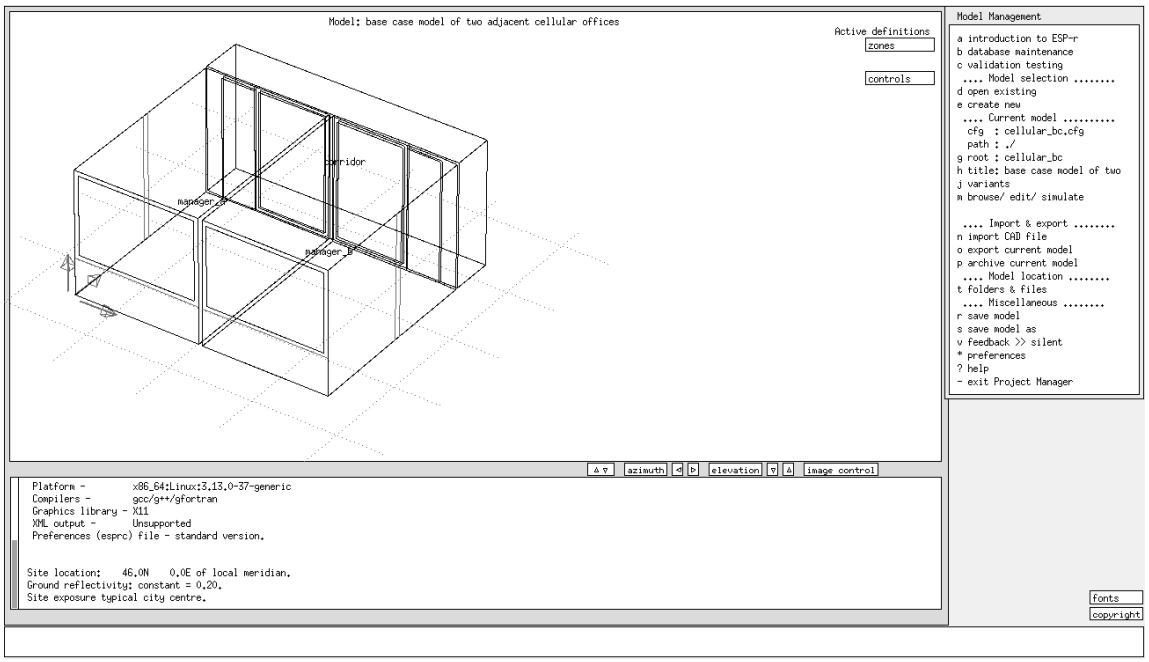
Το πρόγραμμα SketchUp αποτελεί μια ευρέως γνωστή και εύχρηστη εφαρμογή σχεδίασης και δημιουργίας 3D μοντέλων. Διαθέτει ένα πλήρες API (Application Programming Interface) το οποίο παρέχει τη δυνατότητα ανάπτυξης πληθώρας εφαρμογών για το ίδιο. Ο προγραμματισμός για το συγκεκριμένο πακέτο υλοποιείται με τη χρήση μιας σχετικά νέας, interpreted (σσ όχι compiled.) object-oriented γλώσσας η οποία ονομάζεται Ruby. Ουσιαστικά για την ανάπτυξη μιας διεπαφής για το πρόγραμμα αυτό θα πρέπει να δημιουργηθεί ένα script σε γλώσσα Ruby το οποίο θα εκτελείται in-place. Δηλαδή, χωρίς να γίνεται compile αλλά να εκτελείται άμεσα εντός του Ruby interpreter που διαθέτει το SketchUp.[2]



Εικόνα 1. SketchUp Interface

2.2 ESP-r

Το πρόγραμμα προσομοίωσης ESP-r είναι ένας προσομοιωτής για την ενεργειακή απόδοση κτιρίων ο οποίος είναι πλήρης, πολύπλοκος και με πολλές δυνατότητες. Ωστόσο, η υπάρχουσα διεπαφή με το χρήστη είναι αρκετά δύσχρηστη, ειδικά το μέρος που είναι υπεύθυνο για την κατασκευή των μοντέλων για προσομοίωση. Τέλος, από την έρευνα που έγινε καταλήξαμε πως η ανάπτυξη διεπαφής για το ESP-r είναι αρκετά πολύπλοκη και χρονοβόρα.



Εικόνα 2. ESP-r Inteface

2.3 Προηγούμενες Δουλειές

Αναζητώντας στο διαδίκτυο βρήκαμε το su2espr, μια υπάρχουσα διεπαφή για το SketchUp η οποία εξάγει τη γεωμετρία ενός κτιρίου από το SketchUp στο ESP-r με επιτυχία και αποτελεί δημιουργία του Néstor Ubay Pérez Rodríguez. Έτσι σε συνεργασία με τον κύριο Γομπάκη Κωνσταντίνο αποφασίστηκε η επέκταση της παραπάνω διεπαφής καθώς και η ενσωμάτωση μιας δομής που είχε δημιουργήσει ώστε η τελική διεπαφή να εκτελεί τις ζητούμενες λειτουργίες. [3]

2.4 Πληροφορίες Εργαλείων

Στα επόμενα κεφάλαια παρουσιάζονται συνοπτικά τα δύο πιο σημαντικά στοιχεία που κατανοήθηκαν για την εκπόνηση της εργασίας, τα οποία είναι:

- Το GEO file format.
- Το plug-in API του SketchUp.

2.5 Σύνοψη

Ουσιαστικά αυτό που επιτυγχάνεται είναι η κατασκευή αρχείων μοντέλων τα οποία είναι συμβατά με το GEO format του ESP-r καθώς και η δυνατότητα τροποποίησής τους μέσω τουSketchUp.

3. GEO file format

Το ESP-r χρησιμοποιεί ένα κατανεμημένο δένδρο (σσ. distributed tree) αρχείων για να αποθηκεύει τα αρχεία που περιγράφουν ένα project προσομοίωσης. Από την αρχή και για τα πλαίσια της προκείμενης εργασίας, μας ενδιαφέρει το ***πως*** το ESP-r περιγράφει τη γεωμετρία και τα εμπόδια που προκαλούν – άπλες – σκιάσεις/εμπόδια.
Αυτό γίνεται με τη χρήση των .geo αρχείων τα οποία ουσιαστικά αποτελούν μια περιγραφή σε μορφή formatted-text αρχείου των στοιχείων αυτών. Στη συνεχεία περιγράφεται η δομή ενός .geo αρχείου.

3.1 Ορισμός ετικέτας ESP-r (tag)

Για την αναγνώριση κάποιων στοιχείων στο αρχείο χρησιμοποιούνται από το ESP-r κάποιες λέξεις κλειδιά που υποδηλώνουν τι θα κάνει η

συγκεκριμένη γραμμή. Αυτά ορίζονται ως ετικέτες ή γνωστά και ως tags (σσ. από το σημείο αυτό κι έπειτα ετικέτα = tag). Το tag ξεκινάει συνήθως με ένα αστεράκι ”*” και ακολουθεί ο ορισμός του, για παράδειγμα η ημερομηνία παραγωγής του αρχείου γίνεται με το tag date,άρα το ολοκληρωμένο tag είναι: *date.

Τα tags του ESP-r συνήθως έχουν και έναν αριθμό ορισμάτων. Αυτά τα ορίσματα, θα πρέπει να έχουν οπωσδήποτε κάποια τιμή, έστω και κάτι που να δηλώνει την αγνόησή του – γνωστή και ως null τιμή, που συνήθως είναι μια παύλα ”-” η μηδέν ”0”.

3.2 Αρχικά tags

Αρχικά σε κάθε αρχείο υπάρχουν 3 βασικά tags τα οποία είναι κοινά για όλα τα αρχεία .geo. Αυτά είναι τα geometry, date και ένα ειδικό tag χωρίς αστεράκι το Zonedescription.

3.2.1 geometry tag

Το geometry tag μας δείχνει την έκδοση, τον τύπο αρχείου και το όνομα της ζώνης που περιγράφει η γεωμετρία μέσα σε αυτό το αντικείμενο. Στα πλαίσια της εργασίας αυτής πέραν του ονόματος οι υπόλοιπες τιμές θα έχουν τις ίδιες τιμές, άρα το geometry tag δομείται ωςεξής:

***geometry**, έκδοση geo, τύπος geo, όνομα ζώνης. Άρα ένα πραγματικό παράδειγμα είναι:

***geometry**, 1.1, GEN, K2.008

3.2.2 date tag

Το date tag δηλώνει πότε δημιουργήθηκε το εκάστοτε αρχείο και παίρνει ένα όρισμα, την ημερομηνία και την ώρα δημιουργίας. Συνεπώς συντάσσεται ως εξής:

***date**, ημερομηνία και ώρα δημιουργίας.

Άρα ένα πραγματικό παράδειγμα είναι:

***date**, Wed Oct 14 18:47:03 2015

3.2.3 Zone description

Ουσιαστικά εδώ γίνεται γνωστό ότι αρχίζει η δήλωση της γεωμετρίας για την ζώνη του αρχείου. Δεν έχει κάποιο όρισμα και **δε** θέλει αστεράκι κατά τη δήλωση του.

3.3 vertex tag

Το vertex tag περιγράφει τις κορυφές που υπάρχουν στην γεωμετρία του μοντέλου. Όπως γίνεται αντιληπτό πρέπει να δοθούν 3 ορίσματα δηλαδή όσα και οι διαστάσεις που θα οριοθετήσουν την κορυφή αυτή (x, y, z). Άρα το vertextag δομείται ως εξής:

***vertex**, x τιμή, y τιμή, z τιμή

Άρα ένα πραγματικό παράδειγμα είναι:

***vertex**,0,1.00000,2.00000

Σημαντική σημείωση, τα **vertex** είναι οι δομικοί λίθοι (building-blocks) των άκρων (**edges**), άρα πρέπει να δηλωθούν πριν από αυτές. Τέλος κάθε **vertex** διαθέτει και ένα ID το οποίο είναι ο αύξων αριθμός με αρχή τη μονάδα και δίνεται βάσει της σειράς δήλωσης του. Αυτό το ID δε φαίνεται ούτε αποθηκεύεται κάπου στο αρχείο αλλά θεωρείται δεδομένο κατά την ανάγνωση (parsing).

3.4 edges tag

Το edges tag περιγράφει τα άκρα της γεωμετρίας και δίνει τα συσχετισμένα vertices με το συγκεκριμένο άκρο τα οποία εσωτερικά αποθηκεύουμε σαν ένα array. Στο αρχείο το εν λόγω tag παίρνει έναν αυθαίρετο αριθμό ορισμάτων, τόσα όσες είναι οι κορυφές που είναι συνδεδεμένες με το εκάστοτε άκρο. Άρα το edges tag δομείται ως εξής:

***edges**, ID κορυφής 1, ..., ID κορυφής ν

Άρα ένα πραγματικό παράδειγμα είναι:

***edges**, 1, 3, 5, 9 # 4 Κορυφές.

Σημαντική σημείωση, τα **edges** είναι οι δομικοί λίθοι (building-blocks) των επιφανειών (**surfaces**), άρα πρέπει να δηλωθούν πριν από αυτές. Τέλος κάθε **edge** έχει και ένα ID το οποίο είναι ο αύξων αριθμός με αρχή τη μονάδα και δίνεται βάσει της σειράς δήλωσης του. Αυτό το ID δε φαίνεται ούτε αποθηκεύεται κάπου στο αρχείο αλλά θεωρείται δεδομένο κατά την ανάγνωση (parsing).

3.5 surf tag

Το surf tag περιγράφει τις επιφάνειες της γεωμετρίας που βρίσκονται μέσα στο αρχείο. Όπως προαναφέρθηκε πρέπει να έχουν δηλωθεί ήδη τα vertices καθώς και τα edges πριν μπορούμε να βάλουμε αυτό το tag. Γενικά, υποστηρίζει πολλά ορίσματα αλλά μόνο λίγα μας απασχολούν στα πλαίσια της εργασίας αυτής. Άρα το surftag δομείται ως εξής:

***surf**, όνομα επιφάνειας, θέση επιφάνειας[1], είναι παιδί της επιφάνειας (όνομα), -, -, όνομα υλικού, αδιαφάνεια επιφάνειας, εσωτερική/εξωτερική, data 1, data 2[2].

Άρα ένα πραγματικό παράδειγμα είναι:

***surf**, K2.82, VERT, -, -, -, extr_wallGr, OPAQUE, EXTERIOR, 0, 0

3.6 insol tag

Έχει να κάνει με την ηλιακή ακτινοβολία και δε θα μας απασχολήσει στην προκείμενη εργασία, άρα το αγνοούμε.

3.7 insol_calc tag

Όπως και το προηγούμενο tag έχει να κάνει με την ηλιακή ακτινοβολία και δεν θα μας απασχολήσει σε αυτήν την εργασία, άρα το αγνοούμε.

3.8 base_list tag

Το συγκεκριμένο tag παρέχει πληροφορίες για τη βάση (πάτωμα) του μοντέλου. Συγκεκριμένα έχει την εξής σύνταξη:

***base_list**, αριθμός βάσεων μοντέλου (0 αν είναι μοναδική), εμβαδόν βάσεων, μοναδική βάση? (0 ή 1)

Άρα ένα πραγματικό παράδειγμα είναι:

***base_list**, 0, 75.33, 1

[1]Δέχεται μια από τις ακόλουθες τιμές: VERT/CEIL/FLOR/SLOP/UNKN

[2]Τα data 1, 2 είναι συγγενείς επιφάνειες από άλλα αρχεία γεωμετρίας, τα λεγόμενα boundary condition και είναι πάντα 2, μηδενική τιμή σημαίνει ότι δεν συγγενεύει με κάποια επιφάνεια.

3.9 block_start/end & obs tags

Στο σημείο αυτό προς το τέλος του αρχείου δηλώνονται οποιεσδήποτε σκιάσεις διαθέτει το μοντέλο. Αυτές περιέχονται μόνο ανάμεσα στα tags ***block_start** και ***end_block**, δηλαδή είναι κάτι σαν placeholders για τις πιθανές σκιάσεις και μπορούν να περιέχουν από μια έως πολλές διαφορετικές γεωμετρίες σκιάσεων.

3.9.1 obs tag

Οι σκιάσεις μπορούν να έχουν δυο μορφές, μια απλή (περιγράφεται από το obs tag) και μια σύνθετη (περιγράφεται από το obsp tag). Στην παρούσα εργασία ασχοληθήκαμε μόνο με τις άπλες σκιάσεις (obs tag) που ουσιαστικά είναι ένα δοσμένο scaling και rotation κατά μήκος κάθε άξονα (x, y, z) από ένα σημείο ''αρχής''. Συγκεκριμένα έχουν την εξής σύνταξη:

***obs**, x σημείου αρχής, y σημείου αρχής, z σημείου αρχής, φορές scaling x, , φορές scaling y, φορές scaling z, βαθμοί περιστροφής, αδιαφάνεια, όνομα σκιάς, πάντα NONE

Άρα ένα πραγματικό παράδειγμα είναι:

***obs**, 11.000, 11.000, 0.000, 1.000, 1.000, 1.000, 0.000, 1.00, new_blk, NONE

3.10 Επίλογος GEO

Στο σημείο αυτό φτάσαμε στο τέλος της περιγραφής των στοιχείων που περιλαμβάνονται στα GEO αρχεία τα οποία μελετήθηκαν στα πλαίσια της υλοποίησης της παρούσας εργασίας. Παρακάτω περιγράφεται συνοπτικά και ο parser ο οποίος σαρώνει - διαβάζει τα αρχεία GEO και αναγνωρίζει τα tags που παρουσιάζονται παρακάτω:

i) **geometry tag**
ii) **date tag**
iii) **Zone description**
iv) **vertex tag**
v) **edges tag**
vi) **surf tag**
vii) **base_list tag**
viii) **block_start/end_block**
ix) **obs tag**

4. SketchUp API

Στο εν λόγω εδάφιο περιγράφονται συνοπτικά τα στοιχεία και οι δομές που χρησιμοποιήθηκαν από το προγραμματιστικό περιβάλλον ανάπτυξης διεπαφών του SketchUp. Περιγράφουμε συνοπτικά τις κλάσεις (classes) που χρησιμοποιήθηκαν ώστε να είναι ευκολότερη η ανάγνωσή του κώδικα που αναπτύχθηκε.

4.1 Sketchup

Είναι η κυρία κλάση, περιέχει όλες τις πληροφορίες για το υπάρχον μοντέλο καθώς και όλα τα στοιχεία. Από εδώ ξεκινάνε όλα. [4]

4.2 Model

Είναι η κλάση που περιγράφει ένα SketchUp μοντέλο. Από την κλάση Model ξεκινάμε για οποιεσδήποτε ενέργειες θέλουμε να κάνουμε με το μοντέλο μας (μεταβολή γεωμετρίας, υλικό κτλ.).
Συνήθως για το τωρινό μοντέλο πρώτα παίρνουμε το ''τρέχον μοντέλο'' μέσω της κλάσης Sketchup που αναφέραμε προηγουμένως. Συγκεκριμένα θα το πάρουμε καλώντας μέσα στον κώδικα μας την ''Sketchup.active_model'' η οποία επιστρέφει το τρέχον μοντέλο (που φυσικά είναι μια κλάση τύπου Model).[5]

4.3 Color

Είναι η κλάση που παρέχει τη δυνατότητα επεξεργασίας και δημιουργίας χρωμάτων. Χρησιμοποιείται κυρίως για τη δημιουργία υλικών καθώς και την απόδοση χρώματος στα υλικά αυτά (εάν αυτό είναι δυνατό).[6]

4.4 Entity

Αυτή η κλάση περιγράφει τα στοιχεία που μπορούν να περιέχονται σε ένα μοντέλο του SketchUp, όπως Edges, SectionPlanes, Groups κτλ. Γενικά επειδή οι κλάσεις οι οποίες είναι αντικείμενα που μπορούν να μπουν σε ένα μοντέλο (όπως Edge, Face κτλ) είναι παιδιά αυτής της κλάσης οι μέθοδοι αυτής είναι διαθέσιμες και σε αυτές (κλάσεις παιδιά).[7]

4.5 Edge

Κλάση που περιγράφει τις ακμές του μοντέλου.[8]

4.6 Face

Κλάση που περιγράφει τις επιφάνειες του μοντέλου.[9]

4.7 Group

Κλάση στην οποία εισάγονται διάφορα αντικείμενα του μοντέλου προς ομαδοποίηση (πχ δημιουργία ζωνών). [10]

4.8 Transformation

Κλάση που παρέχει τη δυνατότητα πραγματοποίησης γραμμικών μεταβολών στο μοντέλο ή σε κάποιο μεμονωμένο Group του μοντέλου (όπως μετατόπιση, αυξομείωση, περιστροφή κτλ).[11]

5. Υλοποίηση

Στο παρόν κεφάλαιο περιγράφεται η υλοποίηση συνοπτικά καθώς και η λειτουργία των σημαντικότερων συναρτήσεων (functions) του προγράμματος. Γενικά, είναι χωρισμένο σε τρία βασικά μέρη, τα οποία έχουν ως εξής:

- Περιγραφή δικών μας, πρόσθετων δομών δεδομένων
- Συνοπτική περιγραφή modules και δομής
- Περιγραφή εισαγωγής από ESP-r ← SketchUp
- Περιγραφή εξαγωγής από SketchUp → ESP-r

5.1 Πρόσθετες Δομές Δεδομένων

Για την καλύτερη προσαρμογή ακολουθήσαμε (με λίγες αλλαγές) το μοντέλο που είχε δημιουργήσει ο Nestor Perez, αρχικός συγγραφέας της διεπαφής που εξελίσσουμε. Για το λόγο αυτό παραθέτουμε τις ακόλουθες δομές για την καλύτερη περιγραφή κάποιων αντικειμένων εντός του προγράμματος.

- Materials (υλικά) – material_loc.rb
- Vertices (κορυφές) – node.rb
- Obstructions (σκιάσεις) – obstruction.rb
- Surfaces (επιφάνειες) – surface.rb
- Zones (ζώνες) – zone.rb

5.1.1 Material_loc.rb – Εισαγωγή

Ουσιαστικά, αυτές οι επιπλέον δομές συλλέγουν τα στοιχεία που απαιτούνται από τις πιο πλούσιες δομές που διαθέτει το Sketchup, όπως για παράδειγμα η δομή των υλικών κρατάει μόνο 3 από τα πολλά στοιχεία που έχει η κλάση Material του Sketchup – αυτά είναι:

- το όνομα του υλικού.
- αν έχει αδιαφάνεια.
- η τιμή της αδιαφάνειας.

5.1.2 Vertices – node.rb

Όπως και στη δομή των υλικών συλλέγεται μόνο ένα μέρος των δομών του SketchUp, συνεπώς τα στοιχεία που κρατούνται για αυτή τη δομή είναι τα ακόλουθα:

- Το μοναδικό ID του (από το 1 και μετά).
- Τη συντεταγμένη X
- Τη συντεταγμένη Y
- Τη συντεταγμένη Z

5.1.3 Obstruction – obstruction.rb

Για τις σκιάσεις συλλέγονται τα εξής στοιχεία:

- Οι συντεταγμένες του σημείου ''αρχής''[3] (x, y, z)[3 τιμές]
- Φορές που κάνουμε scaling ανά διάσταση (x, y, z) [3 τιμές]
- Τους βαθμούς περιστροφής.
- Την τιμή της αδιαφάνειας του.

5.1.4 Surfaces – surface.rb

Για τις επιφάνειες συλλέγονται τα εξής στοιχεία:

- Το μοναδικό ID τους (από το 1 και μετά).
- Ο τύπος της επιφάνειας.
- Το μοναδικό όνομα της επιφάνειας.
- Το εμβαδόν της επιφάνειας.
- Τα normals της επιφάνειας.[12]
- Τα vertices που την αποτελούν (με βάση το ID τους).
- Το υλικό της επιφάνειας.
- Το γονέα της
- 4 τιμές σχετικές με την οριοθέτηση της στο χώρο (βάσει των normals)

5.1.5 Zones – zone.rb

Στο σημείο αυτό περιγράφονται οι διακριτές ζώνες που υπάρχουν στο μοντέλο. Η κάθε ζώνη αποτυπώνεται στην έξοδο σαν ένα ξεχωριστό GEO αρχείο. Ουσιαστικά εσωτερικά στο Sketchup είναι ένα Group. Για αυτήν την δομή συλλέγονται τα εξής δεδομένα:

- Το μοναδικό ID της (από το 1 και μετά)
- Το μοναδικό όνομά της.
- Τα Sketchup Faces που την αποτελούν
- Τα Vertices που την αποτελούν
- Το group που αφορά σε αυτή τη ζώνη.
- Το group των σκιάσεων που αντιστοιχεί στη ζώνη αυτή.

5.1.6 Σύνοψη για δομές

Γενικά, αυτό γίνεται κυρίως για εργονομικούς λόγους καθώς μπορεί για παράδειγμα να γίνει overload της μεθόδου ”to_s” (to string) της κλάσης ώστε να εκτυπώνεται αυτόματα μια δική μας έκδοση της περιγραφής του αντικείμενου. Αυτό είναι πολύ χρήσιμο όταν γράφουμε τα δεδομένα αυτά σε αρχεία.

5.2 Δομή του προγράμματος - Modules

Ακολουθώντας τις καλές πρακτικές του αντικειμενοστραφούς (object-oriented) προγραμματισμού οι επιμέρους λειτουργίες χωρίστηκαν σε διάφορα αυτόνομα sub-modules. Αυτό έγινε γιατί ως γνωστόν έχει πολλά θετικά στοιχεία και μερικά από αυτά είναι:

- Καλύτερα δομημένος κώδικας
- Ευκολότερος στην κατανόηση
- Καλύτερη συντήρηση του (maintenance). κ.α.

Συγκεκριμένα, ξεκινάμε έχοντας ένα main αρχείο το οποίο καλεί ό,τι είναι απαραίτητο με βάση, κάθε φορά, τη ζητούμενη λειτουργία. Το αρχείο αυτό λοιπόν είναι το: su2espr.rb. Οι λειτουργίες εξαγωγής και εισαγωγής μέσα σε αυτό δηλώνονται σαν items του plug-in menu που διαθέτει το SketchUp. Παραθέτουμε το συγκεκριμένο κώδικα και

αναφέρουμε ακριβώς που γίνεται η κάθε λειτουργία.

```
1  # Access the main View menu
2  view_menu = UI.menu 'Plugins'
3  # Add a separator and a submenu
4  sub_menu = view_menu.add_submenu('ESP-r Plugin')
5  # Add two menu items to the submenu
6  sub_menu.add_item('2d to 3d') {
7      include D2tod3
8      funres = d2tod3
9      UI.messagebox(funres)
10 }
11 sub_menu.add_item('Active Model to .geo') {
12     include Sku2espr
13     # check for shadow material
14     check_shadow_mat
15     sku2espr
16 }
17
18 sub_menu.add_item('Import from ESP-r folder tree') {
19     # check for shadow material
20     include Espr2sku
21     check_shadow_mat
22     espr2sku
23 }
```

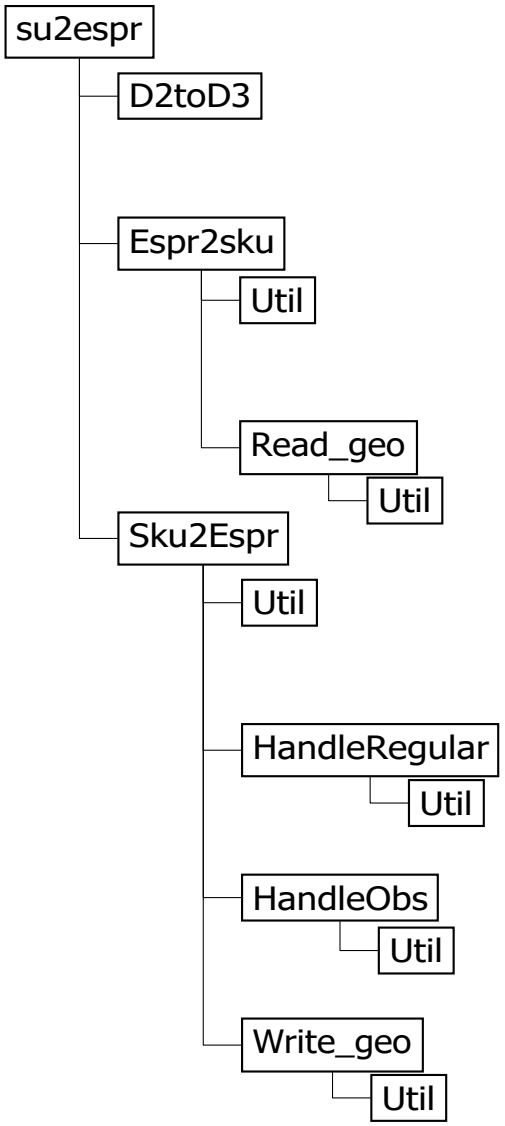
Listing 1: Κώδικας su2espr

Οι επιμέρους λειτουργίες βρίσκονται στα ακόλουθα σημεία:

- Γραμμές 1 − 4: Δηλώνουμε το plug-in μας στο SketchUp
- Γραμμές 6 − 9: Δηλώνουμε το extrude κομμάτι του plug-in (από τον Κ. Γομπάκη)
- Γραμμές 11 − 15: Δηλώνουμε το export σε ESP-r (modules included: Sku2espr)
- Γραμμές 18−22: Δηλώνουμε το import σε SketchUp (modules included: Espr2eku)

Θα πρέπει να σημειωθεί ότι στη Ruby τα modules ≠ classes, τα modules είναι κομμάτια κώδικα τα οποία δηλώνονται έτσι ώστε να μπορούμε να τα χρησιμοποιούμε και να τα φορτώνουμε μέσα σε πολλές κλάσεις – δηλαδή είναι αυτόνομα κομμάτια κώδικα (δηλαδή είναι κάτι σαν βιβλιοθήκες). Οι κλάσεις από την άλλη, είναι – objects

– και εμπεριέχουν μεθόδους που αφορούν μόνο το Object το οποίο περιγράφουν. Συνεπώς οι δομές μας είναι δηλωμένες σαν classes ενώ οι λειτουργικότητες σαν Modules. Μια εικονική δομή των modules και πως αυτά καλούνται μέσα από το πρόγραμμα μας είναι η ακόλουθη:



Σχήμα 1: Module call tree.

5.3 Module util.rb

Αποτελεί ένα πολύ χρήσιμο module το οποίο περιλαμβάνει όλες τις πολύ συχνά χρησιμοποιούμενες συναρτήσεις του προγράμματός μας, καθώς όπως βλέπουμε το τοποθετούμε και το χρησιμοποιούμε σε όλα τα modules που έχει το πρόγραμμα μας. Γενικά, περιγράφεται συνοπτικά τι πραγματοποιεί η κάθε συνάρτηση που περιέχει – ο κώδικας της κάθε συνάρτησης εφόσον είναι μικρός (< 30 γραμμές) θα παρατίθεται στο κείμενο, αλλιώς θα υπάρχει μόνο η περιγραφή της.

5.3.1 create_new_scene

Αυτή η συνάρτηση είναι υπεύθυνη για τη δημιουργία μιας νέας σκηνής (scene) στο Sketchup. Πριν την δημιουργία ρωτάει τον χρήστη εάν επιθυμεί να σώσει την υπάρχουσα σκηνή καθώς η δημιουργία μια νέας απαιτεί το κλείσιμο της ήδη υπάρχουσας.
Αυτός είναι και το σκοπός του if, εάν ο χρήστης πατήσει OK από το dialog box το οποίο εμφανίζεται καλώντας την UI.messagebox τότε δημιουργείται μια νέα σκηνή και διαγράφεται ο Joe (που είναι το ομοίωμα ανθρώπου που εισάγει το Sketchup σε κάθε νέα σκηνή). Αυτό πραγματοποιείται στις γραμμές 8-12 όπου ουσιαστικά στην κενή σκηνή το μόνο Entity που υπάρχει είναι ο Joe, άρα πάμε εισάγονται όλα τα Entities της νέας σκηνής και διαγράφονται την πρώτη (που είναι ο Joe). Τέλος η συνάρτηση επιστρέφει true αν τα καταφέραμε αλλιώς false.

```
1   def create_new_scene
2     choice = UI.messagebox(
3     'To load from ESP-r we have to create a new'+\
4       ' scene, proceed?', MB_OKCANCEL)
5     # check our response
6     if choice == IDOK
7       # create new file
8       Sketchup.file_new
9       # grab entities handle
10      entities = Sketchup.active_model.entities
11      # erase'steve'
12      entities.erase_entities entities[0]
```

```
13          puts ' -- New Scene created'
14          true
15        else
16          puts ' -- User canceled'
17          UI.messagebox('Cannot continue without'+\
18            ' creating a new scene, bye!')
19          false
20        end
21      end
```

Listing 2: Κώδικας create_new_scene

5.3.2 proc_header

Είναι μια απλή συνάρτηση που ουσιαστικά έχει δημιουργηθεί για εξοικονόμηση χώρου και απλά εκτυπώνει στην κονσόλα του SketchUp κάποιες σχετικές πληροφορίες. Αυτές είναι το όνομα αρχείου της παρούσας - νέας σκηνής, το παρόν directory καθώς και ο αριθμός των entities και groups που περιέχονται (σε απόλυτο αριθμό) στη σκηνή. Αυτό εμφανίζεται κατά την διάρκεια του export (SketchUp → ESP-r).

```
1   def proc_header(name, pwd, entities, scene_groups)
2     puts "\nStarting Sketchup to ESP-r export..."
3     puts "Current scene filename: #{name}"
4     puts "Current host directory: #{pwd}"
5     puts "Scene has #{entities.length.to_s} " +\
6     "entities and #{scene_groups.length.to_s} scene groups"
7     puts "
8   end
```

Listing 3: Κώδικας proc_header

5.3.3 proc_header_imp

Είναι μια απλή συνάρτηση που έχει δημιουργηθεί για εξοικονόμηση χώρου, και εκτυπώνει στην κονσόλα του SketchUp κάποιες σχετικές πληροφορίες. Αυτές είναι το παρόν directory καθώς εκεί θα αναζητηθεί πρώτα και πριν ερωτηθεί ο χρήστης για να βρεθεί εάν ήδη υπάρχει η δομή των αρχείων ESP-r. Αυτό εμφανίζεται κατά τη διάρκεια του import (ESP-r → Sketchup).

```
def proc_header_imp
    puts "\nStarting ESP-r to Sketchup import..."
    puts "Searching for "import_from_espr""+\
        " folder in current directory"
    puts "Current host directory: #{Dir.pwd}"
end
```

Listing 4: Κώδικας proc_header_imp

5.3.4 deg_to_rad

Μια απλή συνάρτηση που χρησιμοποιείται για τη μετατροπή των μοιρών σε ακτίνια (radian).

```
# convert degrees to radians
def deg_to_rad(deg)
    deg * 0.0174533
end
```

Listing 5: Κώδικας deg_to_rad

5.3.5 rad_to_deg

Μια απλή συνάρτηση που χρησιμοποιείται για τη μετατροπή των ακτινίων (radian) σε μοίρες.

```
# convert radians to degrees
def rad_to_deg(rad)
    rad / 0.0174533
end
```

Listing 6: Κώδικας rad_to_deg

5.3.6 round_to_sig_bits

Στη συνάρτηση αυτή χρησιμοποιείται η βιβλιοθήκη από το standard library της Ruby BigDecimal, με την οποία γίνεται στρογγυλοποίηση του αριθμού επιτρέποντάς μας να ορίσουμε τον αριθμό των σημαντικών ψηφίων που επιθυμούμε να έχει. Δηλαδή εάν έχουμε τον 4.12335678 και την καλέσουμε βάζοντας σαν όρισμα τον αριθμό και 4 σημαντικά ψηφία τότε θα δοθεί ως έξοδος ο αριθμός 4.1234 καθώς το τελευταίο πριν το

δεκαδικό είναι 5, συνεπώς στρογγυλοποιείται προς τα πάνω και το 3 γίνεται 4.

```
1  def round_to_sig_bits(val, sig_bits)
2    BigDecimal.new(val, sig_bits).to_f
3  end
```

Listing 7: Κώδικας round_to_sig_bits

5.3.7 transform_vertex_to_global

Μια από τις σημαντικότερες συναρτήσεις καθώς μετασχηματίζει μια κορυφή (vertex) της γεωμετρίας από τις τοπικές συντεταγμένες στις ολικές (global), αφού κάθε αντικείμενο ή group έχει το δικό του σύστημα αξόνων το οποίο έχει αρχή τις περισσότερες φορές ένα σημείο διαφορετικό από την αρχή των αξόνων (δηλαδή το (0, 0)).
Ο παραπάνω μετασχηματισμός είναι απαραίτητος καθώς το ESP-r αναγνωρίζει τη γεωμετρία μόνο με βάση την αρχή των αξόνων, συνεπώς και η κάθε κορυφή που εξάγεται πρέπει να έχει εξαχθεί με βάση τις συντεταγμένες από την αρχή των αξόνων.
Επίλυση στο πρόβλημα δόθηκε προσθέτοντας την απόσταση που απέχει το αρχικό κέντρο αναφοράς των συντεταγμένων από την αρχή των αξόνων και σε αυτό προστίθενται οι ήδη υπάρχουσες τιμές.

```
1  # Transform the given vertex to global coordinates
2  def transform_vertex_to_global(sgroup, vert_pos)
3    # x, y, z coordinates of the vertex
4    x_loc = vert_pos.x
5    y_loc = vert_pos.y
6    z_loc = vert_pos.z
7
8    # actual x, y, z axes
9    x_axis = sgroup.transformation.xaxis
10   y_axis = sgroup.transformation.yaxis
11   z_axis = sgroup.transformation.zaxis
12
13   # axis offsets for x, y, z
14   off_x = x_loc * x_axis.x +\
15       y_loc * y_axis.x + z_loc * z_axis.x
16   off_y = x_loc * x_axis.y +\
17       y_loc * y_axis.y + z_loc * z_axis.y
18   off_z = x_loc * x_axis.z +\
19       y_loc * y_axis.z + z_loc * z_axis.z
20
```

```
21      # actual transformations
22      vert_pos.x = sgroup.transformation.origin.x + off_x
23      vert_pos.y = sgroup.transformation.origin.y + off_y
24      vert_pos.z = sgroup.transformation.origin.z + off_z
25
26      vert_pos
27    end
```

Listing 8: Κώδικας transform_vertex_to_global

5.3.8 check_shadow_mat

Η παραπάνω συνάρτηση είναι υπεύθυνη για τη δημιουργία του υλικού των σκιών. Γενικά τα δύο προγράμματα (SketchUp και ESP-r) διαφέρουν ως προς τον ορισμό των materials. Για το λόγο αυτό σε όλες τις σκιές δίνεται ένα προκαθορισμένο υλικό με συγκεκριμένο όνομα και χρώμα. Το όνομα του υλικού αυτού όπως φαίνεται και στον κώδικα παρακάτω είναι 'shadow' και το χρώμα του είναι ένα από τα ήδη υπάρχοντα που διαθέτει η παλέτα του SketchUp και έχει χρωματική ονομασία 'snow'. Ο κώδικας της συνάρτησης ξεκινάει προσπαθώντας να εντοπίσει εάν υπάρχει ένα ήδη υπάρχον υλικό με το ίδιο όνομα, το οποίο πραγματοποιείται στις γραμμές 8-18. Εφόσον βρεθεί υλικό με την παραπάνω ονομασία (shadow) ελέγχεται εάν το χρώμα του είναι το επιθυμητό (snow) ειδάλλως χρωματίζεται κατάλληλα. Όλα τα παραπάνω εκτελούνται στις γραμμές 12-15. Τέλος, αν το εν λόγω υλικό δε βρέθηκε δημιουργείται εξαρχής και στη συνέχεια τοποθετείται στη λίστα με τα υλικά που είναι διαθέσιμα. Προφανώς αφού δημιουργηθεί χορηγούνται σε αυτό και τα κατάλληλα χαρακτηριστικά (όνομα και χρώμα). Τα προαναφερθέντα υλοποιούνται στις γραμμές21-26.

```
1   # check for the shadow material presence
2   def check_shadow_mat
3     # first parse the scene and check if we already have
4     # a material named "shadow", if we do all good, else
5     # addit.
6     shadow_found = false
7     mats = Sketchup.active_model.materials
8     mats.each do |mat|
9       next unless mat.display_name == 'shadow'
10      shadow_found = true
11      puts "\nShadow material found"
12      if mat.color != 'snow'
13        mat.color = 'snow'
14        puts "\n          -- Shadow material color"+\
15          " is not snow, changing it"
```

```
16          end
17          # end the loop
18          break
19        end
20        # if we didn't find it, add the material to avoid
21        # constant checks later on
22        unless shadow_found
23          puts "\nShadow material not present"+\
24          " (default color: snow), adding it..."
25          smat = mats.add('shadow')
26          smat.color = 'snow'
27        end
28      end
```

Listing 9: Κώδικας check_shadow_mat

5.3.9 to_square

Επειδή το SketchUp εσωτερικά χρησιμοποιεί inches (in) για τη μέτρηση των αποστάσεων ενώ το ESP-r μέτρα (m), στο σημείο αυτό γίνεται μετασχηματισμός το εμβαδού από τετραγωνικές inches (in^2) σε τετραγωνικά μέτρα (m^2).

```
1   def to_square(face)
2     face.area*0.0254*0.0254
3   end
```

Listing 10: Κώδικας to_square

5.3.10 convert_path

Η συγκεκριμένη συνάρτηση χρησιμοποιείται για την οριοθέτηση των ”slashes” (δηλαδή τα ”/”, ”\”) βάσει του λειτουργικού. Τα Windows χρησιμοποιούν ”\” ενώ τα Unix-like ”/”. Η Ruby εσωτερικά επεξεργάζεται καλύτερα τα Unix-like filepaths και έτσι γίνεται μετατροπή σύμφωνα με αυτό.
Το if ελέγχει την πλατφόρμα μέσω του Sketchup.platform και εάν είναι :platform_win τότε καλείται η gsub η οποία είναι μέρος της κλάσης Directory (που είναι ο τύπος του path μας). Αυτό που εκτελεί η gsub είναι απλό καθώς όπου εντοπίζει ”\” το αντικαθιστά με”/”.

```
def convert_path path
    if Sketchup.platform == :platform_win
        path.gsub('\\', '/')
    else
        path
    end
end
```

Listing 11: Κώδικας convert_path

5.3.11 check_dir_and_create

Η συνάρτηση αυτή ελέγχει αν υπάρχει ένας φάκελος προορισμού της εξαγωγής και ανδεν υπάρχει δημιουργεί ένα νέο.

```
def check_dir_and_create(path)
    unless File.directory?(path)
        Dir.mkdir(path)
    end
end
```

Listing 12: Κώδικας check_dir_and_create

5.3.12 extract_path

Η συνάρτηση extract_path είναι υπεύθυνη να δώσει το path που αρχικά θα ελεγχθεί πριν ζητηθεί η είσοδός του από το χρήστη. Αυτό το path διαφέρει ανάλογα με τη λειτουργία που θέλει να εκτελέσει ο χρήστης. Για import (δηλ. ESP-r → Sketchup) είναι 'path/import_from_espr/' ενώ για την εξαγωγή (δηλ. Shetchup → ESP-r) 'path/export_to_espr/'. Όπου path είναι ο αρχικός φάκελος στον οποίο έχει δημιουργηθεί το αρχικό αρχείο – μοντέλο. Όπως παρατηρείται ένας ternary operator (expr ? y : n) είναι υπεύθυνος για την επιλογή του τύπου και δίνεται ως ένα boolean όρισμα της συνάρτησης(to_espr).

```
def extract_path(model_path, to_espr)
    path = File.dirname(model_path)
    path = convert_path path
    path + (to_espr ? '/export_to_espr/' :\
        '/import_from_espr/')
end
```

Listing 13: Κώδικας extract_path

5.3.13 remove_espr_comments

Η συνάρτηση αυτή είναι υπεύθυνη για το φιλτράρισμα όλων των σχολίων που ενδέχεται να περιέχουν τα GEO αρχεία. Τα σχόλια σε ένα αρχείο GEO ξεκινούν με μια δίεση ”#” και ότι έπεται μετά το συγκεκριμένο σύμβολο έως το τέλος της γραμμής θεωρείται σχόλιο. Εισάγεται λοιπόν μια γραμμή μέσα στην συνάρτηση και επιστρέφεται το μέρος πριν από την δίεση (αν υπάρχει).
Αυτό γίνεται και πάλι με τη χρήση της συνάρτησης gsub, μόνο που στην περίπτωση αυτή παίρνει σαν όρισμα ένα regular expression όπου το matching ξεκινάει όταν βρεθεί η δίεση ”#” και μετά εντοπιστεί το σύμβολο ”.*” που σημαίνει συμπεριλαμβάνεται οτιδήποτε περιέχεται εν συνεχεία.

```
1  def remove_espr_comments(line)
2    line.gsub(/#.*/, '').strip
3  end
```

Listing 14: Κώδικαςremove_espr_comments

5.3.14 print_puts_ui

Η εν λόγω συνάρτηση είναι υπεύθυνη για την εκτύπωση ενός μηνύματος που δίνεται σαν όρισμα (το msg) και στην κονσόλα του Sketchup καθώς και στο γραφικό περιβάλλον μέσω ενός dialog box.

```
1  def print_puts_ui msg
2    puts msg
3    UI.messagebox msg
4  end
```

Listing 15: Κώδικας print_puts_ui

5.3.15 valid_float?

Συνάρτηση η οποία ελέγχει αν το όρισμα είναι ένας επιτρεπτός αριθμός κινητής υποδιαστολής.

```
1  def valid_float?(str)
2    !!Float(str) rescue false
3  end
```

Listing 16: Κώδικας valid_float?

5.3.16 convert_str_to_float

Πρόκειται για μια συνάρτηση που μετατρέπει με ασφάλεια το δοσμένο όρισμα σε έναν αριθμό κινητής υποδιαστολής αν φυσικά αυτός είναι έγκυρος αλλιώς επιστρέφει σφάλμα.

```
1  def convert_str_to_float(str)
2    if valid_float? str
3      Float(str)
4    else
5      nil
6    end
7  end
```

Listing 17: Κώδικας convert_str_to_float

5.3.17 valid_int?

Συνάρτηση που ελέγχει αν το όρισμα είναι ένας επιτρεπτός ακέραιος αριθμός.

```
1  def valid_int?(str)
2    !!Integer(str) rescue false
3  end
```

Listing 18: Κώδικας valid_int?

5.3.18 convert_str_to_int

Συνάρτηση που μετατρέπει με ασφάλεια το δοσμένο όρισμα σε έναν ακέραιο αριθμό αν αυτός είναι έγκυρος αλλιώς επιστρέφει σφάλμα.

```
1  def convert_str_to_int(str)
2    if valid_int? str
3      Integer(str)
4    else
5      nil
6    end
7  end
```

Listing 19: Κώδικας convert_str_to_int

5.3.19 import_materials_from_file

Αποτελεί μια από τις σημαντικότερες συναρτήσεις καθώς είναι υπεύθυνη να φορτώσει τη λίστα των ήδη exported materials εφόσον αυτή υπάρχει. Συγκεκριμένα, αναμένεται μετά από κάθε επιτυχές export σε ESP-r να υπάρχει και μια λίστα με τα υλικά (materials) που διέθετε η αρχική σκηνή. Γενικά, το format του κάθε material που γίνεται export είναι 6, και πρόκειται για τα ακόλουθα χαρακτηριστικά:

- Μοναδικό όνομα.
- Τιμή κόκκινου(R).
- Τιμή πρασίνου (G)
- Τιμή μπλε (B)
- Τιμή αδιαφάνειας (A).
- Ένταση αδιαφάνειας (%).

Πρακτικά, είναι το όνομα και οι τιμές RGBA που περιγράφουν συνολικά το χρώμα του υλικού. Επειδή ο κώδικας είναι αρκετά μεγάλος παρατίθεται και εξηγείται σε διαφορετικά μέρη τα οποία εξυπηρετούν διαφορετικούς σκοπούς. Το αρχικό μέρος της συνάρτησης είναι υπεύθυνο για τη δημιουργία ενός πίνακα κατακερματισμού (hash table), το οποίο γίνεται στην γραμμή 1.
Ο πίνακας αυτός συγκρατεί μέσα του όλα τα υλικά έχοντας σαν κλειδί εύρεσης το όνομα τους. Το όνομα υλικού είναι εύστοχη επιλογή ως κλειδί αφού είναι μοναδικό βάσει περιορισμού μοναδικότητας. Συνεπώς, για τον έλεγχο ύπαρξης ενός υλικού το μόνο που χρειάζεται είναι μια αναζήτηση στον πίνακα κατακερματισμού καθώς αν δεν υπάρχει θα επιστραφεί ”άδεια” τιμή (null) πολύ γρήγορα καθώς η πολυπλοκότητα αναζήτησης σε hash table είναι $O(1)$.
Στη συνέχεια ελέγχεται αν ήδη υπάρχει φάκελος με τα υλικά και στην περίπτωση που δεν υπάρχει τότε απλά ανατίθεται στα υλικά το προκαθορισμένο (default) υλικό μέχρι ο χρήστης να προβεί σε τροποποιήσεις. Τα προαναφερθέντα πραγματοποιούνται στις γραμμές 4-13 όπου η συνάρτηση επιστρέφει πρόωρα με έναν κενό πινάκα.

```
1  materials_hash = Hash.new
2  filename = filepath + '/model_materials.mat'
3  puts " -- Searching materials in #{filename}"
4  if filename.nil?
5     UI.messagebox('Warning -- no material file provided,'+\
6     ' all geometry will have' +\
7     'default color')
8     return nil
```

```
9   elsif !File.exist?(filename)
10     UI.messagebox('Warning -- material'+\
11       'file provided does not exist')
12     returnnil
13  end
14  materials = Sketchup.active_model.materials
```

Listing 20: Κώδικας

Στο σημείο αυτό φτάνουμε εφόσον έχουμε περάσει επιτυχώς τον παραπάνω έλεγχο και συνεπώς έχει βρεθεί αρχείο με υλικά. Εν συνεχεία ανοίγει το αρχείο για ανάγνωσή του. Ο τρόπος με τον οποίο διαβάζεται το αρχείο είναι γραμμή - γραμμή και επεξεργάζεται η κάθε γραμμή ξεχωριστά (σσ. όπως αναφέρθηκε παραπάνω θεωρούμε ότι υπάρχει μια δήλωση υλικού ανά γραμμή). Για αυτό χρησιμοποιείται ένα for-each τύπου βρόγχος της Ruby όπου έχουμε την κάθε γραμμή του αρχείου ως μεταβλητή f σε κάθε επανάληψη όσο υπάρχουν γραμμές προς προσπέλαση. Αυτό γίνεται στις γραμμές 1-4.

```
1  # parse the file
2  File.open(filename, 'r') do |f|
3  ..
4  end
```

Listing 21: Κώδικας

Παρακάτω γίνεται επεξεργασία της γραμμής, συγκεκριμένα αναμένεται η γραμμή να έχει 6 στοιχεία τα οποία περιγράφηκαν παραπάνω και επιχειρείται η ανάγνωσή τους. Πρώτα η γραμμή που υπάρχει στη μεταβλητή f διαχωρίζεται σε tokens με βάση το ’,’ το οποίο διαχωρίζει τις τιμές αυτές (εξ ορισμού). Στην γραμμή 2 γίνεται αυτός ο διαχωρισμός. Στη συνέχεια γίνεται έλεγχος για τυχόν λάθη (γραμμές 6-11 και 21-24) και τέλος διαβάζονται οι τιμές για τα RGBA καθώς και το ποσοστό εφαρμογής αδιαφάνειας (γραμμές 13-16).

```
1  # parse the file
2  line_tokens = line.strip.split(',')
3  material_found = false
4  puts "Parsing line #{line_no} with material"+\
5  " name #{line_tokens[0]}"
6
7  if line_tokens.length != 6
8     print_puts_ui " -- Warning invalid number of"\
9     " arguments for color "\
```

```
10      +"found at line #{line_no}, skipping"
11      next
12    end
13
14    c_r = convert_str_to_int line_tokens[1]
15    c_g = convert_str_to_int line_tokens[2]
16    c_b = convert_str_to_int line_tokens[3]
17    c_a = convert_str_to_int line_tokens[4]
18
19    m_a = convert_str_to_float line_tokens[5]
20
21    if c_r.nil? or c_g.nil? or c_b.nil? or c_a.nil? or m_a.nil?
22      print_puts_ui " -- Warning invalid color value "\
23      +"found at line #{line_no}, skipping"
24      next
25    end
```

Listing 22: Κώδικας

Στο σημείο αυτό έχει διαβαστεί επιτυχώς μια γραμμή, συνεπώς τώρα πριν δημιουργηθεί ένα νέο υλικό ελέγχεται η λίστα των υλικών που ήδη υπάρχει στο SketchUp για την ύπαρξη κάποιου υλικού με το ίδιο όνομα. Όμως η λίστα των ήδη υπαρχόντων υλικών στο SketchUp δεν είναι σε πίνακα κατακερματισμού, συνεπώς θα πραγματοποιείται μια σειριακή αναζήτηση (γραμμές 2-12). Εάν βρεθεί τέτοιο υλικό τότε διαφοροποιούνται τα χαρακτηριστικά έτσι ώστε να πληροί αυτά του υλικού που μόλις διαβάστηκε (γραμμές 8-10). Τέλος εφόσον βρέθηκε, τότε εισάγεται το τροποποιημένο υλικό στον hash table χωρίς να δημιουργείται άλλο (γραμμή 11).

```
1   # now loop for all of the materials
2   materials.each do |mat|
3     next unless mat.display_name == line_tokens[0]
4     material_found = true
5     puts " -- Material #{line_tokens[0]} already exists,"+\
6       "updating color value"
7
8     mat.color = Sketchup::Color.new(c_r, c_g, c_b, c_a)
9     mat.alpha = m_a
10    # add to the hash
11    materials_hash[mat.display_name] = mat
12  end
```

Listing 23: Κώδικας

Αυτό που πρέπει να γίνει ακόμα είναι να εισαχθεί το υλικό στο hash table που δημιουργούμε. Ελέγχεται αν βρέθηκε ήδη, το οποίο δηλώνεται από τη μεταβλητή material_found. Σε περίπτωση που δε βρέθηκε τότε δημιουργείται ένα νέο υλικό με τα διαβασμένα χαρακτηριστικά (γραμμές 6-8) και τοποθετείται στο hash table μας (γραμμή 10).

```
1   # now we have to add the material
2   unless material_found
3     puts " -- Material #{line_tokens[0]} with"+\
4     " color: (#{c_r}, #{c_g}, #{c_b}," +\
5     " #{c_a}) does not exist, adding it"
6     m = materials.add(line_tokens[0])
7     m.color = Sketchup::Color.new(c_r, c_g, c_b, c_a)
8     m.alpha = m_a
9     # add to the hash
10    materials_hash[m.display_name] = m
11  end
```

Listing 24: Κώδικας

Τελικά, η συνάρτηση επιστρέφει (εφόσον έφτασε μέχρι το τέλος) ένα hash table ο οποίος περιέχει τα υλικά της σκηνής που διαβάζουμε.

5.4 Moduleread_geo.rb

Το επόμενο σημαντικό Module το οποίο εξετάζεται σε αυτό το εδάφιο είναι αυτό με το οποίο πραγματοποιείται η ανάγνωση των Geo αρχείων από το δένδρο φακέλων του ESP-r. Συγκεκριμένα μας ενδιαφέρουν οι φάκελοι που περιέχουν τη *γεωμετρία* του μοντέλου μας, η οποία βρίσκεται εντός του φακέλου **zones**.
Σε αυτό το Module χρησιμοποιείται η βιβλιοθήκη Util.rb, καθώς και οι δομές για τα vertices, sufraces, zones και obstructions τα οποία περιγράφηκαν προηγουμένως. Όπως και στην βιβλιοθήκη Util.rb όλες οι συναρτήσεις που έχουμε δηλώσει περιγράφονται συνοπτικά.

5.4.1 read_geometry_tag

Αυτή η συνάρτηση είναι υπεύθυνη για το διάβασμα του tag της γεωμετρίας όπως περιγράφεται στο GEO file format. Ακόμα ελέγχεται αν αυτό βρέθηκε στο σωστό σημείο. Ως τιμή επιστροφής έχει το όνομα της συγκεκριμένης γεωμετρίας που διαβάζεται (π.χ. ”K2.008”).

```
def read_geometry_tag tokenized_line, line_no
    tag = 'geometry'
    # check position
    unless line_no == 0
        puts "Error, found geometry after the first"\
        " line at line #{line_no.to_s}"
        fail_parse_print tag, line_no,
        'found geometry after the first line',
        true
        returnnil
    end
    # check header length
    unless tokenized_line.length == 3
        fail_parse_print tag, line_no,
        'found line token length for geometry',
        true
        returnnil
    end
    # check format type
    gen_trim = tokenized_line[1].strip
    unless gen_trim == 'GEN'
        fail_parse_print tag, line_no,
        "found invalid file format, we only support"\
        " type GEN got #{gen_trim}",
        true
        returnnil
    end
    # output feedback
    suc_parse_print tag, line_no, false
    # return the name (w/e this is)
    tokenized_line[2].strip
end
```

Listing 25: Κώδικας

5.4.2 parsing_token_error

Πρόκειται για μια βοηθητική συνάρτηση η οποία χρησιμοποιείται για την εκτύπωση ενός αντιπροσωπευτικού μηνύματος στην περίπτωση που διαβαστεί ένα άγνωστο token.

```
# default error for unknown error
```

```
def parsing_token_error(token, line_no)
    fail_parse_print token, line_no,
    "unknown token encountered, token was: #{token}",
    true
end
```

Listing 26: Κώδικας

5.4.3 create_zone

Εξίσου βοηθητική συνάρτηση που χρησιμοποιείται για τη δημιουργία ενός αντικειμένου ζώνης (Zone object) με το όνομα της γεωμετρίας καθώς και έναν αύξοντα αριθμό που είναι ίσος με τον αριθμό των αρχείων που έχουν διαβαστεί μέχρι στιγμής.

```
# create a zone from the parsed data
def create_zone(file_no, line_no, zone, zone_name)
    puts "Creating a new zone data-structure"\
    " with name #{zone_name} and id #{file_no}"
    if zone.nil?
        Zone.new(file_no, zone_name)
    else
        fail_internal_print line_no,
        'Zone already exists... this should not happen...',
        true
    nil
    end
end
```

Listing 27: Κώδικας

5.4.4 token_equal_ignore_case

Βοηθητική συνάρτηση με την οποία ελέγχεται αν τα δυο ορίσματα είναι λεξικογραφικά ισοδύναμα αγνοώντας τα κεφαλαία.

```
# check for token equality ignoring case sensitivity
def token_equal_ignore_case(ftoken, stoken)
    ftoken.casecmp(stoken) == 0
end
```

Listing 28: Κώδικας

5.4.5 delete_after_whitespace

Κατά την εκτέλεση της εν λόγω συνάρτησης χρησιμοποιείται ένα regular expression για τη διαγραφή όλων των χαρακτήρων μετά το *πρώτο* κενό (whitespace).

```
1 # delete trailing characters after first whitespace
2 def delete_after_whitespace token
3     token.gsub(/\s*.*/, '')
4 end
```

Listing 29: Κώδικας

5.4.6 parse_date

Βοηθητική συνάρτηση η οποία χρησιμοποείται για την ανάγνωση του date tag. Επιστρέφεται η τιμή αυτού του tag εφόσον όλα εξελίχθηκαν καλώς.

```
1  # parse the modified date line
2  def parse_date(tokenized_line, line_no)
3      tag = 'date'
4      if line_no != 1
5          fail_parse_print tag,
6          line_no,
7          'encountered a date tag after the second line',
8          true
9          return nil
10     end
11     # rudimentary support
12     suc_parse_print tag, line_no, true
13     tokenized_line
14 end
```

Listing 30: Κώδικας

5.4.7 parse_vertex

Η συνάρτηση **parse_vertex** παρέχει τη δυνατότητα ανάγνωσης και αναγνώρισης των κορυφών (vertices) οι οποίες αποτελούν τη γεωμετρία εκάστοτε αρχείου. Η ανάγνωση γίνεται με τις προδιαγραφές που περιγράφονται στην αναπαράσταση του GEO αρχείου για το Vertex tag. Ακόμα κατά την ανάγνωση εκτελούνται και οι απαραίτητοι έλεγχοι (αριθμός ορισμάτων tag, γραμμές: 14 – 21 και εγκυρότητας τιμών ορισμάτων, γραμμές: 29 – 35) για τις τιμές που διαβάζονται όπως η

αλληλουχία αυτών και το σημείο που διαβάστηκαν. Θα πρέπει να σημειωθεί ότι μπορούμε να διαβάσουμε μόνο μια λίστα από κορυφές ανά αρχείο γεωμετρίας Geo, συνεπώς εάν έχει ήδη διαβαστεί μια λίστα από κορυφές και επιχειρήσουμε να διαβάσουμε μια επόμενη δήλωση λίστας κορυφών από το ίδιο αρχείο θα ήταν λάθος (το οποίο φυσικά ελέγχεται στις γραμμές: 6 – 12). Τέλος εφόσον όλα εξελίχθηκαν ορθώς επιστρέφεται (σε κάθε κλήση της συνάρτησης) ένα νέο αντικείμενο (object) τύπου Node. Συνολικά θα πρέπει να υπάρχουν τόσα Nodes όσα και οι κορυφές που διαβάσαμε (και είναι δηλωμένες).

```
1  # parsing of the vertices that are defined in the geo files
2  def parse_vertex(tokenized_line, line_no, vertex_no,
3      vertice_array, vertices_parsed)
4      tag = 'vertex'
5      args = tokenized_line.length
6      if vertices_parsed
7          fail_parse_print tag, line_no,
8          'found a vertex tag when vertices have'+\
9              ' been already parsed...',
10         true
11         returnnil
12     end
13
14     if args != 4
15         fail_parse_print tag,
16             line_no,
17             "vertex line has invalid number of arguments,"\
18             " needed 4 we got #{args}",
19             true
20             returnnil
21     end
22
23     # try and convert
24     x_cord = convert_str_to_float tokenized_line[1]
25     y_cord = convert_str_to_float tokenized_line[2]
26     z_cord = convert_str_to_float tokenized_line[3]
27
28     # check if we have valid numbers
29     if x_cord.nil? or y_cord.nil? or z_cord.nil?
30         fail_parse_print tag, line_no,
31         'one of the three vertex coordinates'+\
32         ' parsed is not a valid float',
33         true
34         returnnil
35     end
```

```
36      v = Node.new(vertex_no, x_cord, y_cord, z_cord)
37      puts "          -- Creating #{v.to_s}"
38      vertice_array.push(v)
39      suc_parse_print tag, line_no, false
40      v
41  end
```

Listing 31: Κώδικας

5.4.8 parse_insol

Βοηθητική συνάρτηση για την ανάγνωση των insol tags και την αγνόηση τους καθώς στην παρούσα εργασία δε μας απασχολούν.

```
1  # parse insolation -- not needed in sketchup
2  def parse_insol(tokenized_line, line_no)
3      tag = 'insol'
4      suc_parse_print tag, line_no, true
5      tag
6  end
```

Listing 32: Κώδικας

5.4.9 parse_insol_cal

Όπως και η προηγούμενη συνάρτηση χρησιμοποιείται για την ανάγνωση των insol_cal tags και την αγνόηση τους καθώς στην παρούσα εργασία δε μας απασχολούν.

```
1  # parse insolation calculation -- not needed in sketchup
2  def parse_insol_cal(tokenized_line, line_no)
3      tag = 'insol_calc'
4      suc_parse_print tag, line_no, true
5        tag
6  end
```

Listing 33: Κώδικας

5.4.10 parse_shad_calc

Βοηθητική συνάρτηση για την ανάγνωση των shad_calc tags και την αγνόηση τους καθώς στην παρούσα εργασία δε μας απασχολούν.

```
1  # parse shadow calculation -- not needed in sketchup
2  def parse_shad_calc(tokenized_line, line_no)
```

```
3     tag = 'shad_calc'
4     suc_parse_print tag, line_no, true
5     tag
6 end
```

Listing 34: Κώδικας

5.4.11 parse_base_list

Συνάρτηση για την ανάγνωση των base_list και την αγνόηση τους καθώς στην παρούσα εργασία δε μας απασχολούν κατά την εισαγωγή. Ωστόσο τα χρειαζόμαστε κατά την εξαγωγή ενός μοντέλου από το SketchUp στο ESP-r.

```
1 # parse the base list -- not needed in sketchup
2 def parse_base_list(tokenized_line, line_no)
3     tag = 'base_list'
4     suc_parse_print tag, line_no, true
5     tag
6 end
```

Listing 35: Κώδικας

5.4.12 parse_edges

Στη συνάρτηση αυτή διαβάζονται και αναγνωρίζονται από το αρχείο τα edges τα οποία αποτελούν τη γεωμετρία του κτιρίου. Να σημειωθεί ότι κατά την ανάγνωση κάποιου edge list πρέπει να έχει διαβαστεί πρώτα το vertex list, ειδάλλως παρουσιάζεται σφάλμα. Επίσης είναι δυνατό (όμοια με το vertex list) να διαβάσουμε *ένα* edge list ανά αρχείο geo το οποίο το ελέγχουμε ως εξής:

```
1 if edges_parsed
2     fail_parse_print tag, line_no,
3     'found an edge tag when edges have been already parsed...',
4     true
5     return nil
6 end
```

Listing 36: Κώδικας

Στη συνέχεια διαβάζουμε τον αριθμό vertices που απαρτίζουν αυτό το edge και συγκρίνουμε με τον αριθμό που έχουμε διαβάσει. Σε περίπτωση που είναι διαφορετικός τότε παρουσιάζεται σφάλμα.

```
# check the second argument, the edge population
if edge_vert_no.nil?
    fail_parse_print tag, line_no,
    'conversion of second param (edge population) failed',
    true
    return nil
elsif edge_vert_no != (tokenized_line.length-2)
    fail_parse_print tag, line_no,
    "edges to parse (#{edge_vert_no}) mismatch with ones"\
    " supplied: #{tokenized_line.length-2}",
    true
    returnnil
end
```

Listing 37: Κώδικας

Εν συνεχεία δημιουργούμε ένα αντικείμενο επιφάνειας (Surface) (γραμμή: 3) και διαβάζουμε τα vertices στο edge list και τα τοποθετούμε μέσα στην επιφάνειά μας βάσει του index τους (γραμμές: 6 − 21). Τέλος μόλις διαβαστούν όλα ελέγχουμε αν όντως ο αριθμός αυτών ταυτίζεται με αυτόν που περιμένουμε (γραμμές: 23 − 30).

```
# create a surface, the trailing types, area, normal and type are nil
# as they are required by the opposite conversion
surf = Surface.new face_no, nil, nil, nil

# now loop for the number of vertices to read the face points
(2..(tokenized_line.length-1)).each_with_index do |item, idx|
vert_no = convert_str_to_int tokenized_line[item]

    # check for valid conversion
    if vert_no.nil? or vert_no > vertice_array.length or
            vert_no <1
        fail_parse_print tag, line_no,
        "found invalid vertex number or conversion failed at "\
        "pos: #{idx}, parsed: #{tokenized_line[item].to_s}",
        true
        returnnil
    else
        surf.points.push vertice_array[vert_no-1]
        parsed = parsed + 1
    end
```

```
21 end
22
23 # check if we parsed the right amount
24 if parsed != edge_vert_no
25     fail_parse_print tag, line_no,
26     "number of parsed vertices: #{parsed} do not "\
27     "match the expected #{edge_vert_no}",
28     true
29     returnnil
30 end
```

Listing 38: Κώδικας

Τέλος εισάγεται στη λίστα επιφανειών το αντικείμενο επιφάνειας (Surface) που μόλις δημιουργήσαμε (γραμμή: 2) και επιστρέφεται σαν τιμή επιστροφής (γραμμή: 4).

```
1 # finally push the surf
2 face_array.push surf
3     puts 'Parsed face: ' + surf.to_s
4     suc_parse_print tag, line_no, false
5 surf
```

Listing 39: Κώδικας

5.4.13 enable_obs_block

Πρόκειται για μια βοηθητική συνάρτηση η οποία καλείται όταν θέλουμε να δηλώσουμε ότι βρέθηκε ένα block_start tag, το οποίο υποδηλώνει ότι αρχίζουμε να διαβάζουμε obstructions (σκιάσεις). Αν έχει εντοπιστεί ήδη ένα block_start το οποίο δεν έχει κλείσει (μέσω του end_block tag) και επιχειρήσουμε να διαβάσουμε ακόμα ένα, τότε παρουσιάζεται σφάλμα το οποίο και ελέγχεται.

```
1 # enable obstruction block
2 def enable_obs_block(block_flag, line_no)
3     if block_flag
4         fail_internal_print line_no,
5         'Encountered a block_start token while already inside one',
6         true
7         return nil
8     end
9     # enable the block
10    true
11 end
```

Listing 40:Κώδικας

5.4.14 disable_obs_block

Η συνάρτηση *disable_obs_block* καλείται για να δηλώσουμε ότι βρέθηκε ένα end_block tag, το οποίο υποδηλώνει ότι στο σημείο αυτό σταματά η ανάγνωση obstructions. Στην περίπτωση που επιχειρήσουμε να διαβάσουμε ένα end_block tag ενώ δεν έχει εντοπιστεί κάποιο block_start πρώτα, παρουσιάζεται σφάλμα το οποίο ελέγχουμε.

```
1  # enable obstruction block
2  def disable_obs_block(block_flag, line_no)
3      unless block_flag
4          fail_internal_print line_no,
5          "Encountered an end_block token but "\
6          "we already parsed that section",
7          true
8          return nil
9      end
10     # disable the block
11     false
12 end
```

Listing 41: Κώδικας

5.4.15 parse_surf

Μέσω της συνάρτησης *parse_surf* διαβάζουμε τα surf tags τα οποία υποδηλώνουν τα faces τα οποία έχει η γεωμετρία μας. Όπως και με τα προηγούμενα tags που αφορούν στην ουσιαστική γεωμετρία μπορούμε να έχουμε μόνο μια δήλωση λίστας ανά αρχείο Geo και πρέπει να έχουν οριστεί ήδη τα vertice και edge lists τα οποία και ελέγχουμε. Ουσιαστικά, το αντικείμενο **δε** δημιουργείται σε αυτή τη συνάρτηση, αλλά εδώ εμπλουτίζεται με το όνομα του face, το γονέα του καθώς και το υλικό του.

```
1  # parse the face
2  def parse_surf(tokenized_line, line_no, face,
3                 surf_id, surf_parsed)
4      tag = 'surf'
5      args = tokenized_line.length
6      if surf_parsed
7          fail_parse_print tag, line_no,
8          "found a surf tag when surfaces"\
9          " have been already parsed...",
```

```
10        true
11        returnnil
12    end
13
14    if args != 11
15    fail_parse_print tag, line_no,
16        "we parsed surf that does not have"\
17        " 11 arguments, it had: #{args}",
18        true
19        returnnil
20    end
21
22    if surf_id != face.id
23        fail_parse_print tag, line_no,
24        "mismatched id for surf (#{surf_id})"\
25        " and face (#{face.id})", true
26        returnnil
27    end
28    # set up face stuff
29    face.name = tokenized_line[1]
30    face.parent = tokenized_line[3]
31    face.material.push tokenized_line [6]
32
33    # output a successful parse message
34    suc_parse_print tag, line_no, false
35    tag
36 end
```

Listing 42: Κώδικας

5.4.16 parse_zone_description

Η συνάρτηση είναι υπεύθυνη για την ανάγνωση του ”Zone description” tag το οποίο πρέπει να βρίσκεται υποχρεωτικά στην γραμμή no.2 (με index από το 0).

```
1 def parse_zone_description(token, line_no)
2     tag = 'zone description'
3     if line_no != 2
4         fail_parse_print tag,
5         line_no,
6         "encountered a #{tag} tag after the third line",
7         true
8         return nil
9     end
```

```
    # rudimentary support
    suc_parse_print tag, line_no, true
    token
end
```

Listing 43: Κώδικας

5.4.17 parse_obs

Η προκείμενη συνάρτηση είναι υπεύθυνη για την ανάγνωση της γεωμετρίας των obstruction. Καταρχήν για να μπορέσουμε να κάνουμε οποιαδήποτε ανάγνωση obstruction, θα πρέπει να "βρισκόμαστε" μέσα σε ένα obstruction block, στο οποίο εισερχόμαστε όταν εντοπιστεί ένα block_start tag, όπως αναφέρθηκε σε προηγούμενο εδάφιο. Ακόμη ελέγχεται και ο αριθμός των ορισμάτων. Τα παραπάνω ελέγχονται ως εξής:

```
tag = 'obs'
args = tokenized_line.length
if args != 11
    fail_parse_print tag,
        line_no,
        "we parsed obs that has less than"\
        " 11 arguments, it had #{args}",
        true
    return nil
end

unless inside_block
fail_parse_print tag,
    line_no,
    "encountered obs while not being"\
    " inside an obstruction definition block",
    true
    returnnil
end
```

Listing 44: Κώδικας

5.4.18 token_not_supported

Πρόκειται για βοηθητική συνάρτηση η οποία ενημερώνει το χρήστη για την ανάγνωση ενός άγνωστου token μόνο μέσω της κονσόλας (Ruby Console) του SketchUp.

```
def token_not_supported(token, line_no)
    puts " -- Note, although valid token #{token}"\
    " found, we do not support them; skipping them"
    suc_parse_print token, line_no, true
end
```

Listing 45: Κώδικας

5.4.19 suc_parse_print

Βοηθητική συνάρτηση η οποία ενημερώνει το χρήστη για την επιτυχή ανάγνωση κάποιου tag μόνο μέσω της κονσόλας του SketchUp.

```
def suc_parse_print(tag, line_no, skip)
    puts " -- Parsed #{tag} tag at line: #{line_no}"\
    "#{skip ? ', no need for sketchup -- skipping' :"}"
end
```

Listing 46: Κώδικας

5.4.20 fail_parse_print

Συνάρτηση η οποία μας ενημερώνει για τα λάθη μέσω της κονσόλας του SketchUp καθώς και με ένα μήνυμα λάθους το οποίο έχει την μορφή ενός prompt window εντός του SketchUp.

```
def fail_parse_print(tag, line_no, error_msg, ui_print)
    printf " -- Encountered an error while"\
    " parsing a #{tag} tag at line: #{line_no}"
    printf "\n             ** probable error: #{error_msg}"
    if ui_print; UI.messagebox "Parsing error "\
    "at line #{line_no} with message: " + error_msg; end
end
```

Listing 47: Κώδικας

5.4.21 fail_internal_print

Μια βοηθητική συνάρτηση η οποία μας ενημερώνει για τα λάθη μόνο μέσω της κονσόλας του SketchUp.

```
def fail_internal_print(line_no, error_msg, ui_print)
    printf " -- Encountered an internal"\
    " error while parsing line: #{line_no}"
    printf "\n             ** probable error: #{error_msg}"
    if ui_print
```

```
6         UI.messagebox "Internal error while"\
7         " parsing line #{line_no} with message: " + error_msg
8     end
9 end
```

Listing 48: Κώδικας

Στη συνεχεία διαβάζονται τα ορίσματα για το συγκεκριμένο obstruction όπως περιγράφονται στο GEO αρχείο που παρουσιάστηκε στο προηγούμενο εδάφιο.

```
1  # parse obstruction properties
2
3  # position
4  x_pos = convert_str_to_float tokenized_line[1]
5  y_pos = convert_str_to_float tokenized_line[2]
6  z_poz = convert_str_to_float tokenized_line[3]
7
8  # scaling
9  sx = convert_str_to_float tokenized_line[4]
10 sy = convert_str_to_float tokenized_line[5]
11 sz = convert_str_to_float tokenized_line[6]
12
13 # degrees
14 deg = convert_str_to_float tokenized_line[7]
15
16 # geometry opacity
17 opacity = convert_str_to_float tokenized_line[8]
18
19 # obstruction name
20 name = tokenized_line[9]
21
22 # we omit the final argument, which should be NONE
```

Listing 49: Κώδικας

Στη συνεχεία γίνεται ένας έλεγχος τιμών ως εξής:

```
1  # now check if any is nil so that we can fail
2  if x_pos.nil? or y_pos.nil? or z_poz.nil? or
3  sx.nil? or sy.nil? or sz.nil? or deg.nil? or
4  opacity.nil? or name.nil?
5      fail_parse_print tag, line_no,
6      'could not parse one of the obstruction arguments given',
7      true
8      return nil
```

```
end
```

Listing 50: Κώδικας

Στο τέλος δημιουργείται ένα obstruction object το οποίο τοποθετείται εντός της λίστας που περιέχει όλα τα Obstructions του συγκεκριμένου GEO αρχείου.

```
# create the new obstruction instance
# and push it to the obstruction array
obs_array.push Obstruction.new(name,
    x_pos, y_pos, z_poz, sx, sy, sz, deg, opacity)
suc_parse_print tag, line_no, false
tag
```

Listing 53:Κώδικας

5.4.22 place_geo_in_scene

Με την παραπάνω συνάρτηση τοποθετούμε στη νέα σκηνή του SketchUp όλη τη γεωμετρία που διαβάστηκε από το αρχείο GEO.

```
def place_geo_in_scene(filename, face_array,
        obs_array, mat_hash)
    place_faces(face_array, filename, mat_hash)
end
```

Listing 52:Κώδικας

5.4.23 place_faces

Η προκείμενη συνάρτηση εισάγει κάθε face εντός της σκηνής του SketchUp μέσω ενός for-loop. Ακόμη αναθέτει στο κάθε face το material που του αντιστοιχεί, αν αυτό υπάρχει – ειδάλλως τυπώνεται warning.

```
def place_faces(face_array, filename, mat_hash)
    # creating the parenting group that will house of all the
    # faces
    surf_group = Sketchup.active_model.entities.add_group
    # set the group name equal to the filename given as we assume
    # that's the desired name of the group
    surf_group.name = filename

    # loop for each face and add it to the group in question
    face_array.each do |face|
        # add each of the vertices
        vertices = []
```

```
13        face.points.each do |vertex|
14            vertices.push([vertex.x, vertex.y, vertex.z])
15        end
16        # add the face and push it to the group faces
17        scene_face = surf_group.entities.add_face(vertices)
18        if scene_face.nil?
19            puts "Scene face was nil"

20            puts "Vertices are: #{vertices.to_s}"
21            scene_face = Sketchup.active_model.
22                entities.add_face(vertices)
23        end
24        # assign material
25        if !mat_hash.nil? and !mat_hash[face.material[0]].nil?
26            puts " -- Adding material with name "\
27            "#{mat_hash[face.material[0]].display_name}"+\
28            " to face: #{face.name}"
29            scene_face.material = mat_hash[face.material[0]]
30        else
31            puts " -- Invalid hash or material not found for"\
32            " #{face.name} with material "+\
33            "#{mat_hash[face.material[0]]}"
34        end
35    end
36 end
```

Listing 53: Κώδικας

5.5 Module write_geo.rb

Το επόμενο σημαντικό Module το οποίο θα εξετάσουμε είναι αυτό με το οποίο πραγματοποιείται η εγγραφή και εξαγωγή των GEO αρχείων από το SketchUp σε ένα συμβατό με το ESP-r δένδρο φακέλων. Συγκεκριμένα, μιας και εξάγουμε **μόνο** την γεωμετρία δημιουργούμε δυο φακέλους, ένα φάκελο που περιέχει το configuration των ζωνών και έχει ονομασία ”cfg” καθώς και τις ίδιες τις ζώνες, εντός ενός φακέλου με την ονομασία ”zones”. Δυστυχώς, δεν είναι εφικτή η αλλαγή των ονομάτων ή της δομής των φακέλων καθώς αυτή είναι αυστηρά ορισμένη από τις προδιαγραφές του ESP-r.
Στο συγκεκριμένο Module χρησιμοποιείται η βοηθητική βιβλιοθήκη Util.rb. Όπως και στην βιβλιοθήκη Util.rb θα περιγράψουμε όλες τις συναρτήσεις που έχουμε δηλώσει συνοπτικά.

5.5.1 print_date

Βοηθητική συνάρτηση η οποία εκτυπώνει σε ένα αρχείο την ώρα και την ημερομηνία που εκτελείται η συγγραφή του βάσει του format που απαιτεί το ESP-r.

```
# print the date in our format
def print_date
    Time.now.strftime('%a %b %H:%M:%S %Y').to_s
end
```

Listing 54: Κώδικας

5.5.2 write_geo

```
# Function that writes .geo files
def write_geo(zone, path)
    zone_str = 'zones/'
    check_dir_and_create path + zone_str
    geofile = File.new(path + zone_str +
        zone.name + '.geo', 'w')

    # inform the console
    puts "\nWriting geo file for zone with name:#{zone.name}"

    # put the file header
    geofile.puts '*Geometry 1.1, GEN, ' +
        zone.name + ' # tag version, format, zone name'
    geofile.puts '*date ' + print_date +
        '     # latest file modification'
    geofile.puts 'Zone description'

    # write the all important file content.
    write_vertex(geofile, zone.vertex_list)
    write_edges(geofile, zone.faces)
    write_surfaces(geofile, zone.faces)
    write_insolation(geofile)
    write_baselist(geofile, zone.faces)
    write_obstructions(geofile, zone)

    geofile.close

    puts"
end
```

Listing 55: Κώδικας

5.5.3 write_vertex

Η συνάρτηση *write_vertex* είναι υπεύθυνη για την εγγραφή των vertices, που περιέχονται εντός του vertex array, στο αρχείο GEO που μας δείχνει το outfile. Αυτό υλοποιείται με ένα απλό for-each loop μέσα στο οποίο γράφουμε βάσει των προδιαγραφών του vertex tag το κάθε vertex ξεχωριστά και **μόνο ένα** ανά γραμμή.

```
# Write vertices' coordinates
def write_vertex(outfile, vertex_array)
    outfile.puts('# tag, X co-ord, Y co-ord, Z co-ord')
    # loop for each stored vertex
    vertex_array.each do |p|
        outfile.printf("*vertex,%.5f,%.5f,%.5f #                    %1d\n",
        # first convert the values to meters using .to_m
        # function then output them
        p[1].to_m, # x value
        p[2].to_m, # y value
        p[3].to_m, # z value
        p[0])            # vertex id
    end
    outfile.puts('#')
end
```

Listing 56: Κώδικας

5.5.4 write_edges

Αυτή η συνάρτηση είναι υπεύθυνη για την εγγραφή των edges που περιέχονται στο face array στο αρχείο GEO που μας δείχνει το outfile. Αυτό υλοποιείται με ένα απλό for-each loop μέσα στο οποίο γράφουμε βάσει των προδιαγραφών του edge tag το κάθε edge ξεχωριστά και **μόνο ένα** ανά γραμμή. Ακόμα ο αριθμός των edges που γράφονται σημειώνεται ως αύξων αριθμός μετά από τη δήλωση του ως σχόλιο (π.χ. # 1, # 2 κτλ.).

```
# Write zone's edges
def write_edges(outfile, faces)
    outfile.puts("# tag, number of vertices followed"\
    " by list of associated vertices")
    idx =1
    faces.each do |face|
        nv = face.points.length
        str =''
```

```
9         face.points.each do |face_point|
10            str = str + ',' + face_point.id.to_s
11        end
12        outfile.puts('*edges,' +
13            nv.to_s + str + ' # ' + idx.to_s)
14        idx = idx + 1
15    end
16    outfile.puts('#')
17end
```

Listing 57: Κώδικας

5.5.5 write_surfaces

Η συνάρτηση write_surfaces είναι υπεύθυνη για την εγγραφή των surfaces που περιέχονται στο face array, στο αρχείο GEO που μας δείχνει το outfile. Υλοποιείται με ένα απλό for loop μέσα στο οποίο γράφουμε βάσει των προδιαγραφών του surf tag το κάθε surf ξεχωριστά και **μόνο ένα** ανά γραμμή. Ακόμη, ο αριθμός των surf tags που γράφεται σημειώνεται ως αύξων αριθμός μετά από τη δήλωση του ως σχόλιο (π.χ. # 1, # 2 κτλ.).
Σημαντικό σημείο αποτελεί η ιδιότητα facing, η οποία ορίζει προς ποια πλευρά "κοιτάζει" το κάθε face (μπροστά, πίσω κτλ.). Η ιδιότητα αυτή είναι σημαντική για το ESP-r καθώς οι επιφάνειες ανάλογα με το **που** βρίσκονται έχουν διαφορετικές ιδιότητες (όπως μπορούμε να φανταστούμε, ένας εξωτερικός τοίχος θα έχει διαφορετικές ιδιότητες όπως επίσης θα διαφέρει και η θερμική του απόδοση από τον αντίστοιχο εσωτερικό).

```
1 # Write zone's surfaces
2 def write_surfaces(outfile, faces)
3     puts "Writing surfaces to geo file, "\
4         "total surfaces to write: #{faces.length}"
5     for i in 0...faces.length
6         # print the surface
7         puts "        Writing surface with name #{faces[i].name}"
8         check_parent(i, faces)
9         # surface tag
10        outfile.printf '*surf,'
11        # surface name, type and parent surface
12        outfile.printf "#{faces[i].name},"\
13            "#{faces[i].type},#{faces[i].parent}"
14        # surface applied material (assuming uniformity),
15        # and alpha (transparent or not)
16        outfile.printf ",-,-,#{faces[i].material.name},"\
```

```
17          "#{faces[i].material.alpha_type},"
18      # direction (for adjacency)
19      outfile.printf "#{faces[i].facing},"
20      outfile.printf '%02d,%02d, # %d',
21          faces[i].facingzoneid,faces[i].facingsurfaceid, (i+1)
22      if faces[i].facing == 'ANOTHER'
23          outfile.printf " || < #{faces[i].facingsurfacename}:"\
24              "#{faces[i].facingzonename}"
25      elsif faces[i].facing == 'EXTERIOR'
26          outfile.printf ' || < external'
27      end
28      outfile.printf "\n"
29    end
30    outfile.puts('#')
31 end
```

Listing 58: Κώδικας

5.5.6 check_parent

Η συνάρτηση check_parent είναι υπεύθυνη για την εύρεση της επιφάνειας ”γονέα” του κάθε face. Υλοποιείται αναζητώντας τον κοινό αριθμό από vertices που έχουν τα faces μεταξύ τους καθώς και τη φορά των normals τους.

```
1  # Checks whether or not the surface has a parent
2  def check_parent(i, faces)
3      for j in 0...faces.length
4          if j != i and faces[i].normal == faces[j].normal and
5          faces[i].points.length < faces[j].points.length
6          count = 0
7              for pnt1 in faces[i].points
8                  for pnt2 in faces[j].points
9                      if pnt1.id == pnt2.id
10                         count = count + 1
11                     end
12                 end
13             end
14
15             if count < faces[j].points.length and
16                 count >= faces[i].points.length
17                 faces[i].parent = faces[j].name
18             end
19         end
20     end
```

```
21 end
```

Listing 59: Κώδικας

5.5.7 write_insolation

Η παραπάνω συνάρτηση γράφει τα default χαρακτηριστικά (οι τιμές τον οποίων βρέθηκαν από εγχειρίδια του ESP-r) για το insolation καθώς στην παρούσα εργασία δεν ασχολούμαστε με το insolation ωστόσο τα στοιχεία του είναι απαραίτητα για τη δομή των αρχείων GEO.

```
1  # Write data related to the insolation calculation
2  def write_insolation(outfile)
3      outfile.puts("*insol,3,0,0,0,0"\
4          "# default insolation distribution")
5      outfile.puts("#\n# shading directives")
6      outfile.puts("*shad_calc,none # no "\
7          "temporal shading requested")
8      outfile.puts("#\n*insol_calc,none # no"\
9          " insolation requested\n#")
10 end
```

Listing 60: Κώδικας

5.5.8 write_baselist

Η συνάρτηση γράφει το ”base-list” το οποίο είναι μια λίστα με τα πατώματα που έχει το κάθε γεωμετρικό αντικείμενο. Αυτό φαίνεται από το γεγονός ότι το facing property που έχει το κάθε face στα πατώματα είναι τύπου ”FLOR”, συνεπώς μπορούμε να το ελέγξουμε και να το γράψουμε στο αρχείο μας εύκολα.

```
1  # Write base_list line
2  def write_baselist(outfile, faces)
3      area = 0.0
4      list = ''
5      num =0
6
7      faces.each do |face|
8          next unless face.type == 'FLOR'
9          num = num + 1
10         list = list + face.id.to_s + ','
```

```
11          area = area + face.area
12      end
13      if num == 1
14          outfile.puts('*base_list,0,'\
15              +area.to_s+',1')
16      else
17          outfile.puts('*base_list,'\
18              +num.to_s+','+list+area.to_s+' 0')
19      end
20  end
```

Listing 61: Κώδικας

5.5.9 write_obstructions

Η προκείμενη συνάρτηση γράφει όλα τα obstruction που έχει το εκάστοτε GEO αρχείο. Αν υπάρχουν πολλές ζώνες (συνεπώς και πολλά αρχεία GEO) αποφασίζεται σε ποιο αρχείο GEO θα εγγραφεί το κάθε obstruction βάσει της **απόστασης** μεταξύ των κέντρων των αντικείμενων. Τελικά, εισάγεται στο αρχείο GEO που έχει την **μικρότερη διάκεντρη απόσταση** μεταξύ αυτού και του εξεταζόμενου obstruction. Θα πρέπει να σημειωθεί ότι ο έλεγχος αυτός **δε** γίνεται κατά την εγγραφή, αλλά σε άλλο σημείο του προγράμματος μας, ωστόσο αναφέρεται στο σημείο αυτό για πληρότητα.

```
1  def write_obstructions(outfile, zone)
2      if zone.obs_group.length > 0
3          puts "\nLength of obstruction queue for"\
4              " #{zone.name} is #{zone.obs_group.length}"
5          outfile.puts "#\n#block entities:\n# we "\
6              "tag obstructions inside the block"
7          outfile.puts '*block_start, 20 20 # geometric blocks'
8          obs_block = 1
9          # loop for each of the obstructions
10         zone.obs_group.each do |obs|
11             puts "       Writing obstruction with name: #{obs.name}"
12             outfile.write '*obs,'
13             # print (x, y, z) from *origin*
14             outfile.printf '%.5f,%.5f,%.5f,', obs.x, obs.y, obs.z
15             #outfile.write "#{obs.x.to_s},#{obs.y.to_s},#{obs.z},"
16             # print (sx, sy, sz) scale ratios from *group* center
17             outfile.write "#{obs.sx},#{obs.sy},#{obs.sz},"
18             # rotation degrees
19             outfile.write "#{obs.rot_deg},"
```

```
20            # obstruction opacity
21            outfile.write "#{obs.opacity},"
22            # block name, follows the naming
23            # scheme: obstruction_block_{iterno}
24            #outfile.write "obstruction_block_#{obs.name.to_s},"
25            outfile.write "#{obs.name.to_s},"
26            outfile.write "NONE\n"
27            obs_block = obs_block + 1
28          end
29          outfile.puts '*end_block'
30        end
31    end
```

Listing 62: Κώδικας

5.5.10 write_materials

Η συνάρτηση η οποία γράφει τη λίστα των υλικών (materials) καθώς και τις ιδιότητες τους (όπως χρωματικές τιμές, αδιαφάνεια κτλ) σε ένα αρχείο ώστε να μπορέσει να ανακτηθεί η σκηνή ακριβώς όπως ήταν οπτικά. Τονίζεται ότι αυτή η πληροφορία **δεν** είναι απαραίτητη από το ESP-r αλλά είναι επιθυμητό να υπάρχει ώστε να αποδοθούν στις επιφάνειες τα υλικά που είχαν όταν γινόταν η επεξεργασία τους. Για την εξαγωγή χρησιμοποιείται ένα απλό for-each loop και ένα **συγκεκριμένο** και **μοναδικό** όνομα αρχείου (model_materials.mat) το οποίο αποθηκεύεται στον φάκελο των ζωνών (zones).

```
1  # This function takes a hash with all the materials
2  # and writes them
3  def write_materials(model_materials, path)
4      zone_str = 'zones/'
5      check_dir_and_create path + zone_str
6      # open the file
7      mat_outfile = File.new(path + zone_str +\
8          'model_materials.mat', 'w+')
9      # iterate over the materials (since it's a hash they
10     # are unique) and write them to the output file
11     puts "\nExporting materials to"\
12         " file: #{mat_outfile.path}\n\n"
13     model_materials.each_value do |material|
14         puts 'material: ' + material[0].to_s + ', Color: '\
15         + material[1][0].to_s + ', '\
16           + material[1][1].to_s\
17         + ', ' + material[1][2].to_s +\
18           ', ' + material[1][3].to_s\
```

```
        + ', Material Alpha: ' + material[2].to_s

        mat_outfile.puts material[0].to_s + ','\
        + material[1][0].to_s + ','\
material[1][1].to_s + ','\
material[1][2].to_s + ','\
material[1][3].to_s + ','\
        + material[2].to_s
    end
    puts''
    # cleanup.
    mat_outfile.close
end
```

Listing 63: Κώδικας

5.5.11 write_cnn

Η εν λόγω συνάρτηση εξάγει το connectivity για τις ζώνες που εξήχθησαν, το οποίο περιέχει την πληροφορία των ενώσεων μεταξύ των γεωμετρικών αντικειμένων της σκηνής μας. Αποτελεί την καρδιά της συνάρτησης μας, το οποίο είναι ουσιαστικά ένα for-loop για όλα τα GEO αρχεία, ελέγχοντας όλα τα faces που περιέχονται σε κάθε GEO αν "συγγενεύουν" με κάποιο face από άλλη ζώνη.

```
for zone in zones
    for surface in zone.faces
        #see ??? documentation for explanation
        con_file.print('        '+zone.id.to_s+'        '+surface.id.to_s+'        ')
        if surface.facing=='ANOTHER'
        con_file.print('3')
        elsif surface.facing=='EXTERIOR'
        con_file.print('0')
        else
        con_file.print('-1')
        end
        con_file.print('        '+surface.facingzoneid.to_s+\
          '    '+surface.facingsurfaceid.to_s+'        # ')
        con_file.printf('%4d ',surf_num.to_s)
        if surface.facing=='ANOTHER'
        con_file.print(surface.name.to_s+' in zone >|< '\
          +surface.facingsurfacename.to_s+\
          ' in '+surface.facingzonename.to_s)
        elsif surface.facing=='EXTERIOR'
        con_file.print(surface.name.to_s+' in zone is External')
        else
```

```
22            con_file.print(surface.name.to_s+' in zone not yet define')
23        end
24        con_file.puts('')
25        surf_num=surf_num+1
26    end
```

Listing 64: Κώδικας

5.5.12 write_cfg

Η συνάρτηση *write_cfg* εξάγει το configuration για τις ζώνες που εξάγονται. Να σημειωθεί ότι αυτό είναι το configuration **μόνο** για τα GEO files που επεξεργάζονται και αφορά τις ζώνες και όχι το συνολικό configuration για τα υπόλοιπα κομμάτια του ESP-r. Συνεπώς θα πρέπει αν θέλουμε να έχουμε περαιτέρω κομμάτια του ESP-r να κάνουμε ”append” αυτό το αρχείο στο γενικό configuation του ESP-r project μας.

```
1  # Function that writes .cnn file
2  # needs to be placed at the end of the cfg file of the project
3  def write_cfg(zones, path, filename)

4      cfg_str = 'cfg/'
5      check_dir_and_create path + cfg_str
6      # open the configuration file
7      config_file = File.new(path + cfg_str +\
8              filename.to_s + '.cfg1', 'w+')
9      # write the file header
10     config_file.puts('* Building')
11     config_file.puts(filename + ' exported from SketchUp')
12     config_file.puts(zones.length.to_s + '                    # no of zones')
13
14     zones.each do |zone|
15         config_file.puts('*zon ' + zone.id.to_s +\
16           ' # reference for ' + zone.name.to_s)
17         config_file.puts('*opr UNKNOWN # schedules')
18         config_file.puts('*geo ../zones/' + zone.name.to_s +\
19           '.geo # geometry file')
20         config_file.puts('*con UNKNOWN # construction')
21         config_file.puts('*zend # zone end')
22     end
23
24     # pad the connection file
25     config_file.puts('*cnn ' +\
26       filename.to_s + '.cnn              # connections')
27     config_file.puts('0              # do not build flow network')
```

```
28     config_file.close
29 end
```

Listing 65: Κώδικας

5.6 Εισαγωγή στο SketchUp από το ESP-r

Αποτελεί ένα module το οποίο είναι υπεύθυνο για μια από τις δυο βασικές λειτουργίες της παρούσας εργασίας. Χρησιμοποιώντας το μπορούμε να εισάγουμε γεωμετρία από το ESP-r στο Sketchup έτσι ώστε η επεξεργασία της να είναι εύκολη και παραγωγική. Σε αυτό το εδάφιο περιγράφονται αναλυτικά τα περιεχόμενα του αρχείου **espr2sku.rb** το οποίο περιέχει και την υλοποίηση της λειτουργίας αυτής.

5.6.1 Βιβλιοθήκες

Σε αυτή τη λειτουργία γίνεται χρήση των ακολούθων βιβλιοθηκών:

- Util (util.rb)
- Read_Geo (read_geo.rb)

Καθώς και των ακόλουθων δομών δεδομένων:

- Zone (zone.rb)

Όλα τα παραπάνω έχουν αναλυθεί σε προηγούμενα εδάφια και δε θα αναλυθούν περαιτέρω. Να σημειωθεί ότι αυτές είναι οι βιβλιοθήκες και δομές που χρησιμοποιεί **απευθείας** το Module αυτό, **εμμέσως** χρησιμοποιεί και άλλες, το οποίο πραγματοποιείται μέσω κλήσεων συναρτήσεων που υπάγονται στα Util και Read_Geo modules.

5.6.2 Εισαγωγή στο SketchUp

Ουσιαστικά για το κομμάτι αυτό δημιουργήθηκε ένας parser από την αρχή ο οποίος δύναται να υποστηρίζει τα αρχεία .geo και άμεσα να τα μεταφράζει σε δομές δεδομένων οι οποίες είναι κατανοητές από το SketchUp.
Δεδομένου της μορφής των αρχείων GEO και των ιδιοτροπιών τους δημιουργήθηκε ένας top-to-bottom parser ο οποίος ελέγχει και τη δομή του αρχείου μαζί. Λόγω της εύστοχης διάσπασης των συναρτήσεων επετεύχθη η υλοποίηση αρκετά εύκολα και ο κώδικας της βασικής ρουτίνας για αυτή τη λειτουργία είναι πολύ μικρός.

```
1 def espr2sku
2     # print up the tool header
```

```
3     proc_header_imp
4     # get the current path
5     home_path = Dir.pwd
6     # create a new scene
7       unless create_new_scene; return; end
8
9       # extract the correct path based on platform
10      path = extract_path home_path, false
11      if check_for_valid_folder path
12        puts "Path #{path} exists"
13        choice = UI.messagebox('Found an existing'+\
14          ' "import_from_espr" folder, try loading from it?',
15          MB_OKCANCEL)
16        if choice == IDOK
17          puts "Loading from #{path}"
18        else
19          puts 'Probe user to input a valid folder for import'
20        end
21      else
22        puts "Import folder at #{path} does not exist,"+\
23        " probe the user for a valid path"
24        path = probe_for_path
25        unless check_for_valid_folder path
26        puts 'Cannot continue no valid path found'
27        UI.messagebox('Cannot continue no valid path found')
28        end
29      end
30
31      # finally now that we have a path, read the geo files
32      zone_path = path + '/zones'
33      mat_hash = import_materials_from_file zone_path
34      puts "Material hash size #{mat_hash.size}"
35      load_tree_structure zone_path, mat_hash
36  end
```

Listing 66: Κώδικας

Όπως φαίνεται οι περισσότερες γραμμές είναι για έλεγχο έγκυρων παραμέτρων, και η συνάρτηση που ουσιαστικά διαβάζει τη σκηνή από τα αρχεία είναι η load_tree_structure η οποία αναλύεται παρακάτω. Πριν από αυτό όμως ας εστιάσουμε την προσοχή μας στη συνάρτηση check_for_valid.

```
1  def check_for_valid_folder path
2    (!path.nil? and Dir.exists? (path)
3          and validate_folder(path))
```

```
end
```

Listing 67: Κώδικας check_for_valid

Η συνάρτηση *check_for_valid* ελέγχει αν ο φάκελος που ζητάμε υπάρχει καθώς και αν ο φάκελος είναι έγκυρος χρησιμοποιώντας για το τελευταίο τη συνάρτηση *validate_folder* η οποία φαίνεται εν συνεχεία.

```
def validate_folder path
puts 'Checking if the given path'+\
    ' has the required subfolder (zones)'
    (Dir.entries(path).select {
        |entry| File.directory?
            File.join(path, entry) and
    (entry == 'zones')
    }.length > 0)
end
```

Listing 68: Κώδικας validate_folder

```
# load ESP-r tree structure
def load_tree_structure zone_path, mat_hash
    puts "Trying to load geo files from#{zone_path}"
    fnames = Dir[File.join(zone_path, '*.geo')]
    puts fnames.to_s
    unless fnames.length > 0
        puts 'Could not load geo files,'+\
        ' empty zones folder provided'

        nil
    end

    # load the each of the geo files
    fnames.each_with_index do |fname, file_no|
        puts"\n\n"
        load_geo fname, file_no, mat_hash
    end
end
```

Listing 69: Κώδικας load_tree_structure

Ουσιαστικά η *load_tree_structure* είναι ένας wrapper που καλεί τη *load_geo* για να διαβάσει όλα τα αρχεία που περιέχονται στο φάκελο

zone, και η κλήση της *load_geo* είναι ανάλογη των αρχείων .geo που περιέχονται στο φάκελο. Επί της ουσίας η *load_geo* διαβάζει το κάθε αρχείο γραμμή - γραμμή αφαιρώντας τα τυχόν σχόλια που μπορεί να περιέχονται στο αρχείο (Σημ. τα σχόλια στο GEO αρχείο ξεκινούν με τη δίεση ”#”). Τέλος πριν αρχίσει ο διαχωρισμός των tags για να ελεγχθεί ποια συνάρτηση θα κληθεί για το τρέχον tag πραγματοποιείται ένα tokenization στη γραμμή, έχοντας ως διαχωριστή (delimiter) το κόμμα (”,”).

```
1  def load_geo geo_path, file_no, mat_hash
2      zone = nil
3      vertex_id = 1
4      faces_id = 1
5      surf_id = 1
6      vertice_array = []
7      face_array = []
8      obs_array = []
9      vertices_parsed = false
10     edges_parsed = false
11     surf_parsed = false
12     #obs_parsed = false
13     inside_obs_block = false
14     puts "Starting parsing geo file: #{geo_path}\n\n"
15     File.open geo_path, 'r' do |f|
16         f.each_line.with_index do |line, line_no|
17         printf "Parsing line #{line_no}, "
18         # remove comments
19         fline = remove_espr_comments line
20         # check if we have a comment line
21         if fline.empty?
22         printf "comment line found -- skipping\n"
23         next
24         end
25         line_tokens = fline.strip.split(',')
26
27         # basic check here, shown in the next segment.
28
29     end
30     # now try and place the parsed geometry in the scene
31     place_geo_in_scene(File.basename(geo_path), face_array,
32     obs_array, mat_hash)
33 end
```

Listing 70: Κώδικας load_geo

Ο βασικός έλεγχος για το διαχωρισμό του κάθε tag με το δικό του if-case είναι το ακόλουθο το οποίο αν και λίγο μεγάλο κομμάτι κώδικα είναι πραγματικά απλό στην ουσία διότι το μόνο που κάνει είναι η αναγνώριση του τρέχοντος tag και η κλήση της συνάρτησης που το διαχειρίζεται.
Όλες οι συναρτήσεις που είναι στο παρακάτω κομμάτι κώδικα έχουν αναλυθεί στο *Read_Geo Module*, συνεπώς η οποιαδήποτε απορία ως προς το τι υλοποιεί η κάθε μια συνάρτηση απαντάται σε αυτό.

```
# case insensitive search
puts "Tag token: #{filtered_token}"
if token_equal_ignore_case filtered_token, '*geometry'
    zone_name = read_geometry_tag(line_tokens, line_no)
    # check if we found the zone name
    if zone_name.nil?; return; end
    # now try to create the zone
    if create_zone(file_no, line_no,
        zone, zone_name).nil?; return; end
elsif token_equal_ignore_case filtered_token, '*date'
    if parse_date(filtered_token, line_no).nil?; return; end
elsif token_equal_ignore_case filtered_token, 'zone description'
    if parse_zone_description(filtered_token, line_no).nil?;
        return;end
elsif token_equal_ignore_case filtered_token, '*vertex'
    if parse_vertex(line_tokens, line_no,
                vertex_id, vertice_array,
                vertices_parsed).nil?; return; end
        vertex_id = vertex_id + 1
elsif token_equal_ignore_case filtered_token, '*edges'
    vertices_parsed = true
    if parse_edges(line_tokens, line_no, faces_id,
    face_array, vertice_array, edges_parsed).nil?; return; end
    faces_id = faces_id + 1
elsif token_equal_ignore_case filtered_token, '*insol'
    vertices_parsed = true; edges_parsed = true; surf_parsed = true
    if parse_insol(line_tokens, line_no).nil?; return; end
elsif token_equal_ignore_case filtered_token, '*shad_calc'
    vertices_parsed = true; edges_parsed = true; surf_parsed = true
    if parse_shad_calc(line_tokens, line_no).nil?; return; end
elsif token_equal_ignore_case filtered_token, '*insol_calc'
    vertices_parsed = true; edges_parsed = true; surf_parsed = true
    if parse_insol_cal(line_tokens, line_no).nil?; return; end
elsif token_equal_ignore_case filtered_token, '*base_list'
    if parse_base_list(line_tokens, line_no).nil?; return; end
elsif token_equal_ignore_case filtered_token, '*block_start'
    if (inside_obs_block = enable_obs_block(inside_obs_block,
```

```
38          line_no)).nil?
39          return
40      end
41  elsif token_equal_ignore_case filtered_token, '*end_block'
42      if (inside_obs_block = disable_obs_block(inside_obs_block,
43          line_no)).nil?
44          return
45      end
46  elsif token_equal_ignore_case filtered_token, '*obs'
47      if parse_obs(line_tokens, line_no, inside_obs_block,
48          obs_array).nil?; return; end
49  elsif token_equal_ignore_case filtered_token, '*obsp'
50      token_not_supported filtered_token, line_no
51  elsif token_equal_ignore_case filtered_token, '*surf'
52      if parse_surf(line_tokens, line_no, face_array[surf_id-1],
53              surf_id, surf_parsed).nil?
54          return
55      end
56      surf_id = surf_id + 1
57  else
58      parsing_token_error line_tokens[0], line_no; return
59  end
```

Listing 71: Κώδικας βασικού έλεγχου και διαχείρισης tag

5.7 Εξαγωγή από το SketchUp στο ESP-r

Αποτελεί το επόμενο module το οποίο είναι υπεύθυνο για τη δεύτερη βασική λειτουργία της παρούσης εργασίας. Χρησιμοποιώντας το παρέχεται η δυνατότητα εξαγωγής της γεωμετρίας από το SketchUp στο ESP-r έτσι ώστε η προσομοίωση του μοντέλου να καταστεί εύκολη χωρίς να χρειαστεί η σχεδίαση εντός του ESP-r το οποίο είναι μη παραγωγικό και δύσχρηστο. Στο παρόν εδάφιο περιγράφονται αναλυτικά τα περιεχόμενα του αρχείου **sku2espr.rb** το οποίο περιέχει και την υλοποίηση της λειτουργίας αυτής.

5.7.1 Βιβλιοθήκες

Στη λειτουργία γίνεται χρήση των ακόλουθων βιβλιοθηκών:

- Util (util.rb)
- HandleRegular (handle_regular_gem.rb)
- HandleObs (handle_obs.rb)

- Write_Geo (write_geo.rb)

Καθώς και των ακόλουθων δομών δεδομένων:

- Zone (zone.rb)
- Surface (surface.rb)
- Node (node.rb)
- Material (material_loc.rb)

Όλα τα παραπάνω εκτός των *HandleRegular* και *HandleObs* αναλύθηκαν σε προηγούμενα εδάφια συνεπώς δε γίνεται περαιτέρω ανάλυσή τους. Όμως οι δυο βιβλιοθήκες είναι ειδικές για τη λειτουργία αυτή και δημιουργήθηκαν ειδικά για αυτό το εδάφιο, συνεπώς κρίνεται συνετό να εξεταστούν κατά τη διάρκεια αυτής της ενότητας. Τέλος όπως και με το προηγούμενο Module πρέπει να σημειωθεί ότι αυτές είναι οι βιβλιοθήκες και δομές που χρησιμοποιεί **απευθείας** το Module αυτό και **εμμέσως** χρησιμοποιεί και άλλες. Αυτό υλοποιείται μέσω κλήσεων συναρτήσεων που υπάγονται στα *Util*, *Write_Geo*, *HandleRegular* και *HandleObs* modules.

5.7.2 Εξαγωγή

Αρχικά, ορίζονται οι global μεταβλητές μας οι οποίες συλλέγουν τις απαραίτητες πληροφορίες για την εξαγωγή της παρούσης ενεργής σκηνής του SketchUp στο ESP-r. Οι μεταβλητές αυτές είναι οι ακόλουθες:

```
1 model_materials = Hash.new                # material collection
2 scene_groups = []                         # scene objects (grouped)
3 obstruction_groups = []                   # obstruction groups
4 regular_groups = []                       # regular geometry groups
5 model = Sketchup.active_model             # active model
6 entities = model.entities                 # model entities
7 selected_model = model.selection          # currently selected model
8 nhash = Hash.new                          # zone hash
```

Listing 72: Global μεταβλητές

Οι περισσότερες μεταβλητές είναι τύπου array στη Ruby εκτός από τα *material* και *zone hash* τα οποία είναι δηλώνονται ως ένα Hash table λόγω του ότι παρέχεται η δυνατότητα πολύ γρήγορων ελέγχων για την ύπαρξη ή όχι ενός υλικού.
Αρχικά αυτό που πραγματοποιείται είναι ένα φιλτράρισμα όλων των

αντικειμένων τύπου Group κάθε σκηνής του SketchUp, καθώς κάθε αντικείμενο είναι **κατά σύμβαση** μέλος ενός group γεωμετρίας. Έτσι γίνεται ομαδοποίηση κατά έναν όμορφο και απλό τρόπο των GEO files καθώς και των ίδιων των αντικειμένων της σκηνής. Το φιλτράρισμα πραγματοποιείται ως εξής:

```
entities.each do |var|
    if var.is_a?Sketchup::Group
        scene_groups.push(var)
    end
end
```

Listing 73: Φιλτράρισμα Group

Στη συνέχεια αυτό που πραγματοποιείται είναι ένας έλεγχος για τυχόν λάθη στα Group που έχουμε (όπως για παράδειγμα να έχουν το ίδιο όνομα, να μην έχουν όνομα ή όγκο).

Στην περίπτωση που υπάρχει κάποιο λάθος δηλώνεται στο χρήστη. Όλα τα Group τα οποία είναι έγκυρα (valid) εισάγονται σε ένα SketchUp selection. Ο κώδικας που το πραγματοποιεί είναι ο ακόλουθος:

```
selected_model.clear
scene_groups.each do |sgroup|
    # insert to the hash while checking for duplicates
    if nhash[sgroup.name].nil?
    nhash[sgroup.name] = 1
    else
    return 'Found a zone with a duplicate'+\
        ' name, zones must have unique names'
    end
    if sgroup.name == ''
        puts 'no name detected'
        selected_model.add(sgroup)
    elsif !sgroup.name.to_s.ascii_only?
        puts 'weird characters detected in name'
        selected_model.add(sgroup)
    elsif sgroup.volume <= 0.1
        puts "small volume detected in group"+\
            " #{sgroup.name} skipping..."
        selected_model.add(sgroup)
    else
        puts " -- Found valid group with"+\
            " name: #{sgroup.name}"
    end
end
```

Listing 74: Κώδικας έλεγχου εγκυρότητας Group

Στη συνεχεία ελέγχεται αν όντως ευρέθησαν έγκυρα Group, καθώς στην αντίθετη περίπτωση δε είναι δυνατή η συνέχιση της λειτουργίας. Αυτό γίνεται ως εξής:

```
if selected_model.length > 0
    msg = 'Found one or more zone errors... all zones'+\
    ' must be named and that name '\
    +'must be using only ascii characters and their'+\
    ' volume must be > 0.1'
    UI.messagebox msg
return msg
end
```

Listing 75: Κώδικας

Κατόπιν δηλώνονται ακόμα κάποιες global μεταβλητές και τυπώνεται στην κονσόλα ένα μήνυμα το οποίο δηλώνει ότι η εξαγωγή μπορεί να ξεκινήσει χρησιμοποιώντας τη βοηθητική συνάρτηση proc_header.

```
# get the file name in order to
# name the .cnn and .cfg files
model_path = Sketchup.active_model.path
filename = File.basename(model_path, '.*')
scene_zones = []
warning = ''
zone_id = 1

# print header in console
proc_header(filename, File.dirname(model_path),
    entities, scene_groups)
```

Listing 76: proc_header call

Στη συνέχεια διαχωρίζεται η κανονική γεωμετρία από τις σκιάσεις (obstructions). Αυτό κατά σύμβαση γίνεται κατά την διάρκεια του σχεδιασμού. Ο σχεδιαστής πρέπει να ακολουθήσει τουλάχιστον μια σύμβασηέτσιώστεναορίσει μια γεωμετρία ως σκίαση (obstruction), οι οποίες είναι οι ακόλουθες:

- Να ορίσει τη γεωμετρία σε ένα Group το όνομα του οποίου να ξεκινάει με τη συμβολοσειρά obs (υποχρεωτικό)
- Να ορίσει σε όλες τις επιφάνειες (faces) της σκίασης το υλικό με το όνομα shadow. (προαιρετικό)

Στην περίπτωση που ένα Group έχει σωστό όνομα για να αναγνωριστεί ως σκίαση και δεν έχει οριστεί σε αυτό το σωστό υλικό (shadow), κατά την επεξεργασία του για εξαγωγή αλλάζει αυτόματα το υλικό στο σωστό για την ορθή λειτουργία συνεπώς αυτό γίνεται **υποχρεωτικά**.

```
# grab the above geometry which we filtered above
# so we don't loop again the entire scene
scene_groups.each do |sgroup|
    # check if we have an obstruction,
    # if we do then place them in the
    # obstruction array since we must
    # first process the regular geometry
    # first
    if sgroup.name.start_with?('obs')
        puts 'Obstruction group found'
        obstruction_groups.push(sgroup)
    # handle regular groups of geometry
    else
        regular_groups.push(sgroup)
        warning = handle_regular_groups(model_materials,
            scene_zones, sgroup, zone_id)
    zone_id = zone_id + 1
    end
end
```

Listing 77: Κώδικας διαχωρισμού γεωμετρίας

Ακολούθως πραγματοποιείται επεξεργασία των σκιάσεων. Οι συναρτήσεις *handle_regular_groups* και *handle_obstructions* εξετάζονται στη συνέχεια.

```
handle_obstructions(scene_zones, obstruction_groups)
```

Listing 78: Κώδικας επεξεργασίας σκιάσεων

Στη συνέχεια δημιουργούνται οι συνδέσεις μεταξύ των Group (αφορούν το αρχείο cnn) και πραγματοποιείται έλεγχος αν τελικά μας απέμεινε κάποια έγκυρη ζώνη.

```
# to make the connections
scene_zones.each do |zone|
    check_facing(zone, scene_zones)
end

# if we don't have any zones to export,
# stop execution here
if scene_zones.length == 0
    print_puts_ui 'Critical Error: There are'+\
    ' no valid zones to be exported'
return
end
```

Listing 79: Κώδικας δημιουργίας συνδέσεων και ελέγχου εγκυρότητας

Αφού έχουν βρεθεί έγκυρες ζώνες οι οποίες μπορούν να εξαχθούν σε ένα ή περισσότερα GEO αρχεία, ξεκινά η εξαγωγή (export). Δηλαδή δημιουργείται το δένδρο αρχείων και αποθηκεύονται σε αυτό τα έγκυρα Group. Στο τέλος ο αριθμός των GEO αρχείων θα είναι ίσος με τον αριθμό των έγκυρων Group.

```
path = extract_path(model_path, true)

check_dir_and_create path
scene_zones.each do |zone|
    write_geo(zone, path)
end

# create the configuration directory if needed
check_dir_and_create('cfg')

# write the connection file (.cnn)
write_cnn(scene_zones, path, filename)
# write the configuration file (.cfg)
# needs to be placed at the end of the original file of Esp-r
write_cfg(scene_zones, path, filename)
# writes the *unique* materials file
write_materials(model_materials, path)
```

Listing 80: Κώδικας

5.7.3 Χειρισμός κανονικής Γεωμετρίας

Γενικά ο χειρισμός των Group κανονικής γεωμετρίας έχει ως εξής:

- Για κάθε Group δημιουργείται ένα αντικείμενο ζώνης (zone).
- Στη συνέχεια για κάθε Face που περιέχει το προκείμενο Group δημιουργείται και ένα καινούργιο Surface.
- Μετά αποθηκεύεται το material και ανατίθεται σε αυτό το Face.
- Στην συνεχεία για κάθε vertex που ανήκει στο Face δημιουργείται ένα αντικείμενο τύπου Node και αποθηκεύονται τα στοιχεία του vertex σε αυτό.
- Κατόπιν ελέγχεται το orientation του Face μέσω των normals του.
- Στη συνέχεια το Face εντάσσεται στη λίστα των Faces της ζώνης μας.
- Τέλος εισάγεται το Group στις σκηνές που πρόκειται να εξαχθούν.

Ο κώδικας της συνάρτησης χωρίς το βασικό loop το οποίο θα αναλυθεί στη συνέχεια είναι ο ακόλουθος:

```
1  def handle_regular_groups(model_materials,
2      scene_zones, sgroup, zone_id)
3      puts 'Processing geometry group with name: '\
4           + sgroup.name.to_s
5      pnt_id = 1
6      sfc_id = 1
7      debug_msg = ''
8
9      # gran the current zone and wrap it inside a class
10     current_zone = Zone.new(zone_id, sgroup.name)
11
12     # This is the basic loop.
13     sgroup.entities.each do |group_entity|
14     end
15 end
```

Listing 81: Κώδικας handle_regular_groups

Η συνάρτηση *set_type* είναι υπεύθυνη βάσει των normals του κάθε face να δώσει το orientation του, και στη συγκεκριμένη περίπτωση αν πρόκειται για ταβάνι, πάτωμα η κανονικό τοίχο.

```
def set_type(normal)
    if normal[2]==1
        type='CEIL'
    elsif normal[2]==-1
        type='FLOR'
    elsif normal[2] <= 0.001 and
        normal[2] >= -0.001
        type='VERT'
    else
        type='SLOP'
    end
    type
end
```

Listing 82: Κώδικας set_type

Στο σημείο αυτό εξετάζεται το κεντρικό loop, που βρίσκεται στη συνάρτηση *handle_regular_groups* και είναι υπεύθυνο για την κωδικοποίηση της κανονικής γεωμετρίας. Όπως αναφέρθηκε το πρώτο πράγμα που εκτελείται είναι η δημιουργία (μετά από τη ζώνη) ενός Surface αντικειμένου για κάθε Face που διατίθεται. Το οποίο φαίνεται στο ακόλουθο απόσπασμα κώδικα.

```
# create a new surface
face = Surface.new(sfc_id, # surface id
    to_square(group_entity),
    group_entity.normal,
    set_type(group_entity.normal))
# increment the surface id
sfc_id = sfc_id + 1
```

Listing 83: Κώδικας δημιουργίας Surface object

Στην συνέχεια τοποθετείται το material στη λίστα και στο εν λόγω Face, στην περίπτωση που **δεν** έχει κάποιο material ορίζεται σε αυτό το default material που είναι το κόκκινο χρώμα.

```
# check if our current face doesn't have a material assigned
if group_entity.material.nil?
    group_entity.material = 'red'
    debug_msg = debug_msg + "\nSurface was found without a"+\
        " material in group: " +                sgroup.name
```

```
    debug_msg = debug_msg + ' assigning default material: Red'
end

# assign a material to the surface we just created
face.material = Material_loc.new(group_entity.material.name,
                    group_entity.material.alpha)

# push to the global material hash
if model_materials[group_entity.material.name].nil?
    model_materials[group_entity.material.name] = \
                        [group_entity.material.name.to_s,
                        group_entity.material.color.to_a,
                        group_entity.material.alpha.to_f]
end
```

Listing 84: Κώδικας χειρισμού material

Στη συνέχεια δημιουργούνται τα vertices που διαθέτει το κάθε Face και αποθηκεύονται στη vertice list που έχει το συγκεκριμένο Face. Είναι σημαντικό να τονισθεί ότι οι συντεταγμένες των κορυφών εκφράζονται με βάση το κέντρο του **Group** και **όχι** την αρχή των αξόνων όπως απαιτεί το ESP-r. Για την επίλυση αυτού του προβλήματος χρησιμοποιείται η συνάρτηση *transform_vertex_to_global* για μετατροπή των τοπικών συντεταγμένων σε ολικές.

```
# now process the vertices
group_entity.vertices.each do |vertex|
    vertex_exists = false
    vertex_pos = vertex.position
    #puts 'Vertex before: ' + vert_pos.to_s

    # Transform the vertex to the global coordinates
    transform_vertex_to_global(sgroup, vertex_pos)

    # this should not be like this... but it is.
    for p in 0...current_zone.vertex_list.length
    if vertex_pos.x == current_zone.vertex_list[p][1] and
    vertex_pos.y == current_zone.vertex_list[p][2] and
    vertex_pos.z == current_zone.vertex_list[p][3]
    vertex_exists = true
    id2 = current_zone.vertex_list[p][0]
    end
    end

```

```
    # check if the vertex exists already
    unless vertex_exists
        current_zone.vertex_list.push([pnt_id,
            vertex_pos.x, vertex_pos.y, vertex_pos.z])
        id2 =pnt_id
        pnt_id = pnt_id + 1
    end
    # push the vertex in the face vertices list
    face.points.push(Node.new(id2, vertex_pos.x,
                    vertex_pos.y,
                    vertex_pos.z))
end
```

Listing 85: Κώδικας δημιουργίας vertices

Στη συνέχεια ορίζεται ένα αντιπροσωπευτικό όνομα για το συγκεκριμένο Face. Στην περίπτωση που το όνομα αυτό ξεπερνάει σε μέγεθος τους 12 χαρακτήρες κρατώνται οι πρώτοι 12 από αυτούς.

```
if current_zone.name.to_s == ''
    face.name = 'surface_' + sfc_id.to_s
else
    if face.normal[2] == 1
            face.name = current_zone.name + '_' + 'CEIL'
        elsif face.normal[2] == -1
            face.name = current_zone.name + '_' + 'FLOR'
        else
            face.name = current_zone.name + '_' + sfc_id.to_s
        end

    # if we have a larger than 12 characters as the
    # name face hold only the *last* twelve of them
    if face.name.size > 12
        face.name = face.name[(face.name.size - 12),
            face.name.size]
    end
end
```

Listing 86: Κώδικας ονομασίας Faces

Μετά την ονομασία του Face τοποθετείται στη λίστα του Zone και το plug-in προχωρά στην επεξεργασία του επόμενου εγκύρου Face.

```
# push the face in the current zone
current_zone.faces.push(face)
```

Listing 87: Κώδικας

Τέλος, αφού έχει γίνει επεξεργασία όλων των Faces του Group η επεξεργασμένη και γεμάτη πλέον ζώνη (Zone) τοποθετείται στη λίστα μαζί με τις υπόλοιπες που είναι έτοιμες προς εξαγωγή.

```
# store the current group
current_zone.group = sgroup
# push the zone in the zone vector
scene_zones.push(current_zone)
debug_msg
```

Listing 88: Κώδικας

5.7.4 Χειρισμός Σκιάσεων

Ο χειρισμός σκιάσεων (obstructions) είναι λίγο πιο πολύπλοκος από την απλή γεωμετρία καθώς είναι απαραίτητος επιπλέον, ο υπολογισμός των ακόλουθων παραμέτρων:

- Της απόστασης τους από τα κέντρα των Group ώστε να καθοριστεί σε ποιο GEO αρχείο θα ενταχθεί η κάθε σκίαση.
- Την περιστροφή που έχει (ως γωνία)
- Τις συντεταγμένες του κάτω, μπροστά, δεξιά vertex
- Τους λόγους scaling σε X, Y, Z ως προς το προηγούμενο vertex.

Όπως και με την κανονική γεωμετρία όλα πραγματοποιούνται μέσα σε ένα βασικό loop, το οποίο είναι το ακόλουθο:

```
obstruction_groups.each do |obstruction|
        ob_center = obstruction.bounds.center
        puts "\nProcessing Obstruction group "+\
          "with name #{obstruction.name}"
        puts "Group center point is: #{ob_center.to_s}"
        min_distance = ref
        obs_parent_zone = nil
        vertex_near_origin = nil
        vertex_near_origin_pos = nil
        top_surface = nil
        top_surface_vertex = nil
        top_surface_vertex_pos = nil

        # Stuff that will explained...
end
```

Listing 89: Κώδικας βασικού loop για τα obstruction

Κατόπιν για κάθε obstruction αναζητείται το Group που απέχει τη μικρότερη απόσταση από το κέντρο του, γεγονός που πραγματοποιείται με τον παρακάτω τρόπο.

```
scene_zones.each do |scene_zone|
    reg_group = scene_zone.group
    reg_center = reg_group.bounds.center
    dist = reg_center.distance(ob_center)

    # set the min, if needed
    if min_distance == ref or dist < min_distance
        puts "\t            New min distance #{dist.to_s} found"+\
        " setting obstruction group to #{reg_group.name}"
        min_distance = dist
        obs_parent_zone = scene_zone
    else
        puts "\t            No need to adjust the min distance,"+\
            " currently at #{min_distance.to_s}"
    end
end
```

Listing 90: Κώδικας ελέγχου απόστασης obstruction

Στη συνέχεια χρησιμοποιείται το ίδιο σχεδόν loop που χρησιμοποιήθηκε στην κανονική γεωμετρία για το κάθε Face του obstruction με τη διαφορά ότι εδώ ορίζεται το υλικό shadow στο κάθε Face και ότι βάσει της απόστασης από την αρχή των αξόνων πλέον αναζητείται το **παρόν** σημείο που απέχει τη μικρότερη απόσταση από την αρχή των αξόνων και βρίσκεται στο Face του δαπέδου.

```
obstruction.entities.each do |obs_entity|
    next unless obs_entity.is_a? Sketchup::Face
            printf "\n\tFound a valid obstruction face "+\
                "with surface area #{obs_entity.area.to_s}"
            printf "\n\tFace normal is: #{obs_entity.normal.to_s}"
            # assign the shadow material if
            # needed to all faces of the obstruction
                if (obs_entity.material.nil?) or
                    (obs_entity.material != 'shadow')
                    obs_entity.material = 'shadow'
                debug_msg = debug_msg +\
                    "\nSurface was found without a material"
                debug_msg = debug_msg +\
                    " in obstruction group #{obstruction.name}"
```

```
15              debug_msg = debug_msg +\
16                  'assigning default obstruction material: shadow'
17              end
18
19              if obs_entity.normal[2] == 1
20                  top_surface = obs_entity
21                  next
22              elsif obs_entity.normal[2] != -1
23                  puts "\n\tSkipping no need to examine this surface"
24                  next
25              end
26
27              # loop for all the vertices of the obstruction
28              # to find the nearest vertex from the origin as
29              # this is used for a rotation and scaling point
30              min_distance = ref
31              puts "\n\tNeed to examine "+\
32                  "#{obs_entity.vertices.length} vertices"
33              obs_entity.vertices.each do |vertex|
34                  vertex_pos = vertex.position
35                  transform_vertex_to_global(obstruction, vertex_pos)
36                  #puts "\tExamining vertex at: #{vertex_pos.to_s}"
37                  dist = vertex_pos.distance ref_origin
38                  # now compare the distance
39                  if min_distance == ref or dist < min_distance
40                  puts "\t          New min distance #{dist.to_s}"+\
41                      " found setting vertex to #{vertex_pos.to_s}"
42                  min_distance = dist
43                  vertex_near_origin = vertex
44                  else
45                  printf "\t          No need to adjust the min distance,"+\
46                      " currently at #{min_distance.to_s}"
47                  printf " offered #{dist.to_s} from #{vertex_pos.to_s}\n"
48                  end
49              end
```

Listing 91: Κώδικας

Στη συνέχεια εντοπίζονται οι συντεταγμένες της εν λόγω κορυφής σε σχέση με την αρχή των αξόνων καθώς όπως προαναφέρθηκε οι συντεταγμένες των κορυφών είναι υπολογισμένες βάσει του κέντρου του Group στο οποίο ανήκουν.

```
# check if we found a vertex
if vertex_near_origin.nil?
    puts "\nError -- no vertex found from face\n"
else
    # transform to origin coordinates
    vertex_near_origin_pos = vertex_near_origin.position.clone
    transform_vertex_to_global(obstruction,
        vertex_near_origin_pos)
    printf "\n\tSuccess -- Vertex from origin "+\
        "with shortest distance"
    printf " is located at "+\
        "#{vertex_near_origin_pos.to_s}\n\n"
end
```

Listing 92: Κώδικας μετασχηματισμού συντεταγμένων για κορυφή

Σε αυτό το επίπεδο αναζητείται η κορυφή (vertex) στη δεξιά πλευρά αυτού που βρέθηκε προηγουμένως και είναι κοντά στην αρχή των αξόνων έτσι ώστε να υπολογιστεί το διάνυσμα και να βρεθεί η γωνία στρέψης.

```
# now get the vector of the vertex with
# the vertex at its bottom "right-side"
puts "\n"
vertex_after_origin = nil
vertex_after_origin_glob_pos = nil
adjacent_vertices = []
# search surfaces
obstruction.entities.each do |obs_entity|
    next unless (obs_entity.is_a? Sketchup::Face and
        vertex_near_origin.used_by? obs_entity and
        obs_entity.normal[2] == 0)
```

```
12          puts "\t -- Vertex is used by surface"
13          # since we are searching for points we need from the face
14          # only the points that are on the same axis as our box
15          # so they have the same x,y-values but different z values
16          obs_entity.vertices.each do |vertex|
17          if      vertex_near_origin != vertex and
18              vertex_near_origin.position.z == vertex.position.z
19              cur_vpos_glob = transform_vertex_to_global obstruction,
20              vertex.position.clone
21              printf "\t              -- Found vertex with same z "+\
22                  "cords #{vertex.position.to_s},"
23              printf " ref: #{vertex_near_origin.position.to_s}\n"
24              # push the vertex in the adjacency matrix
25              adjacent_vertices.push(vertex)
26          end
27      end
28  end
```

Listing 93: Κώδικας

Εδώ αναζητείται η κορυφή που βρίσκεται ακριβώς πάνω από την κορυφή που βρίσκεται πιο κοντά στους άξονες ώστε να κατασκευαστεί το διάνυσμα και να βρεθεί ο λόγος ύψους (Sy).

```
1  top_surface.vertices.each do |vertex|
2      if vertex_near_origin.position.x == vertex.position.x and
3          vertex_near_origin.position.y ==\
4              vertex.position.y
5          printf "\n\t ** Found a matching vertex"+\
6              " at the top surface, coords: "
7          top_surface_vertex = vertex
8          top_surface_vertex_pos = transform_vertex_to_global\
9              obstruction, vertex.position.clone
10         puts "#{top_surface_vertex_pos.to_s}"
11     end
12 end
13
14 fpos = transform_vertex_to_global obstruction,\
15     adjacent_vertices[0].position.clone
16 spos = transform_vertex_to_global obstruction,\
17     adjacent_vertices[1].position.clone
```

Listing 94: Κώδικας

Εδώ επιλέγεται ποιο από τα δυο διανύσματα θα χρησιμοποιηθεί, καθώς λόγω της φοράς μπορεί να είναι μικρότερο (ως προς το σημείο που βρέθηκε) είτε το δεξί είτε το αριστερό διάνυσμα. Στο σημείο αυτό πραγματοποιείται η σύγκριση για την επιλογή του διανύσματος που έχει τη μικρότερη απόσταση από την αρχή των αξόνων.

```
# we want to create the vector from
# [near_origin --> after_origin] so we
# create the two candidates
dir_vec_f = vertex_near_origin_pos.vector_to fpos
dir_vec_s = vertex_near_origin_pos.vector_to spos

# we also calculate the angle
f_angle = dir_yvec.angle_between dir_vec_f
s_angle = dir_yvec.angle_between dir_vec_s
# assume that we don't have a negative rotation
negative_rotation = false
printf "\n\tFirst vector origin -> fpos #{dir_vec_f.to_s}\n"
printf "\n\tAngle is: #{f_angle.to_s}\n"


printf "\n\tFirst vector origin -> fpos #{dir_vec_s.to_s}\n"
printf "\n\tAngle is: #{s_angle.to_s}"
vec_angle = f_angle
rot_vec = dir_vec_f
rot_vec_l = dir_vec_s
# check if we need the other one
if s_angle < f_angle
    vertex_after_origin = adjacent_vertices[1]
    vec_angle = s_angle
    rot_vec = dir_vec_s
    rot_vec_l = dir_vec_f
end
```

Listing 95: Κώδικας

Στη συνέχεια η διαφορά των διανυσμάτων δίνει τη γωνία ώστε να υπολογιστεί η τελική τιμή της περιστροφής.

```
vertex_after_origin_glob_pos = \
    transform_vertex_to_global obstruction,
            vertex_after_origin.position.clone

if vertex_after_origin_glob_pos.y <
    vertex_near_origin_pos.y
```

```
    puts ' ** We have negative rotation **'
    negative_rotation = true
    vec_angle = -vec_angle
end
printf "\n\t              -- Final after origin vertex position: "
printf "#{vertex_after_origin.nil? ? 'nil' :\
    vertex_after_origin_glob_pos.to_s}\n"
vec_angle_deg = rad_to_deg(vec_angle)
```

Listing 96: Κώδικας ανάθεσης τελικής περιστροφής

Τέλος αυτό που μένει να γίνει είναι η ανάθεση των λόγων scaling σε κάθε κατεύθυνση ανάλογα με τη φορά ώστε να είναι δυνατή η δημιουργία του στο ESP-r.

```
if vertex_after_origin_glob_pos.x <
    vertex_near_origin_pos.x
sx = -rot_vec.length.to_m
else
sx = rot_vec.length.to_m
end
sy = rot_vec_l.length.to_m

sz = vertex_near_origin_pos.distance \
     top_surface_vertex_pos
```

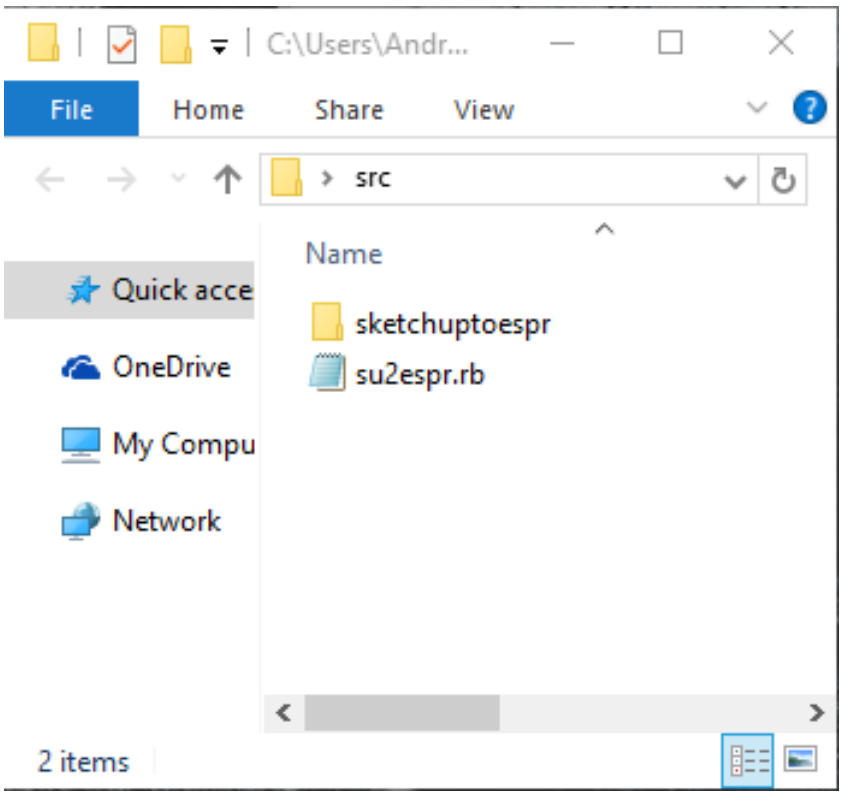
Listing 97: Κώδικας ανάθεσης λόγου scaling

6. Παρουσίαση λειτουργικότητας plug-in

Το παρόν κεφάλαιο αποτελεί εγχειρίδιο χρήσης του plug-in που δημιουργήσαμε και παρουσιάσαμε στα προηγούμενα κεφάλαια. Παρουσιάζονται η εγκατάσταση του plug-in καθώς και οι δύο βασικές λειτουργίες που εκτελεί ανάλογα με την επιλογή του χρήστη. Για την παρακάτω παρουσίαση χρησιμοποιήθηκαν η έκδοση *SketchUp 2015* της εφαρμογής σχεδίασης 3D μοντέλων SketchUp σε περιβάλλον Windows και η open sourced εφαρμογή ενεργειακής προσομοίωσης κτιρίων ESP-r Project Manager release 12.0 σε περιβάλλον Unix.

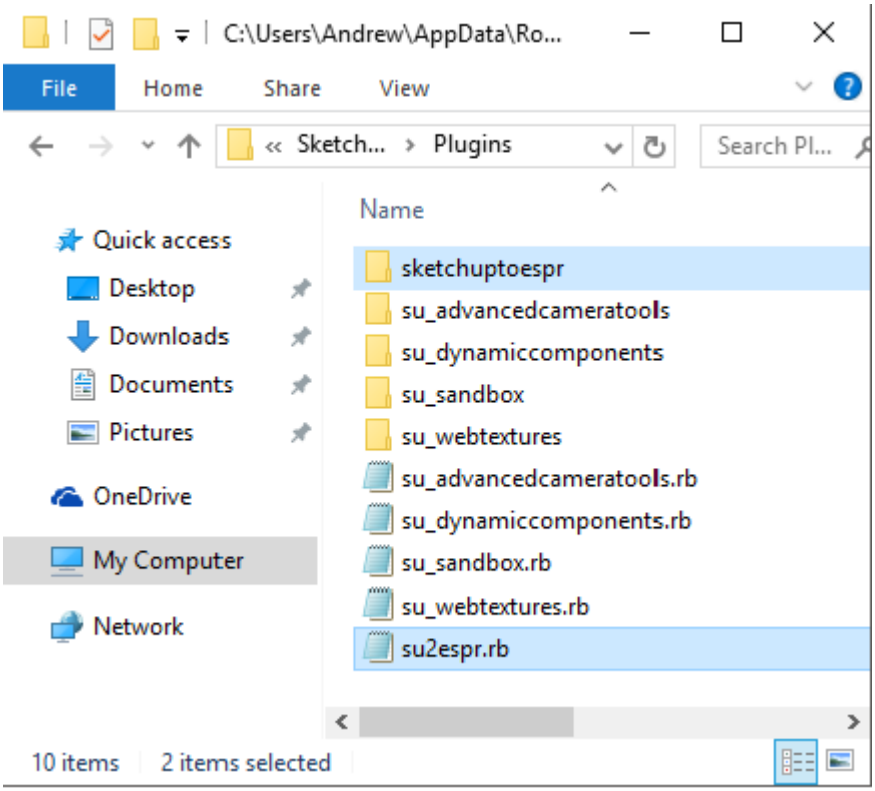
6.1 Εγκατάσταση plug-in

Η εγκατάσταση πραγματοποιείται με την αντιγραφή και τοποθέτηση των αρχείων .rb στο φάκελο προορισμού των plug-in που διαθέτει το SketchUp.



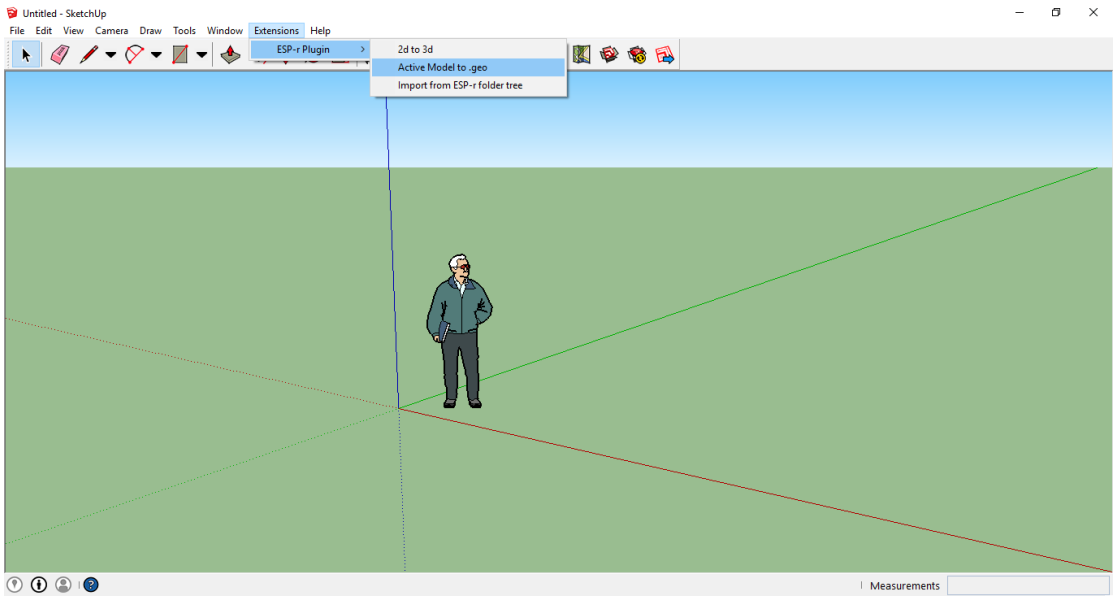
Εικόνα 3. Εικόνα

Τα αρχεία που περιέχει ο φάκελος src τοποθετούνται στην τοποθεσία C:\Users*Username*\AppData\Roaming\SketchUp\SketchUp 2015\SketchUp\Plugins.[4]

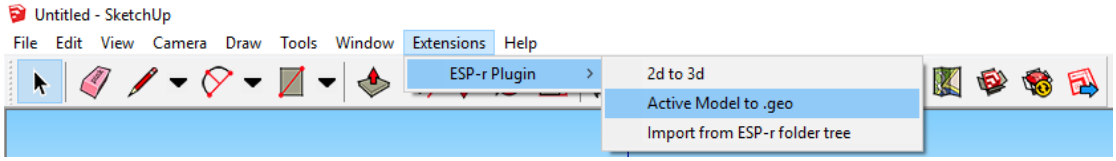


Εικόνα 4. Εικόνα

Το plug-in έχει εγκατασταθεί πλέον στη λίστα με τα plugins που διαθέτει το SketchUp. Εκτελώντας το πρόγραμμα SketchUp παρατηρούμε ότι το plug-in μας περιέχεται στο plug-in menu που διαθέτει το SketchUp.



Εικόνα 5. Εικόνα



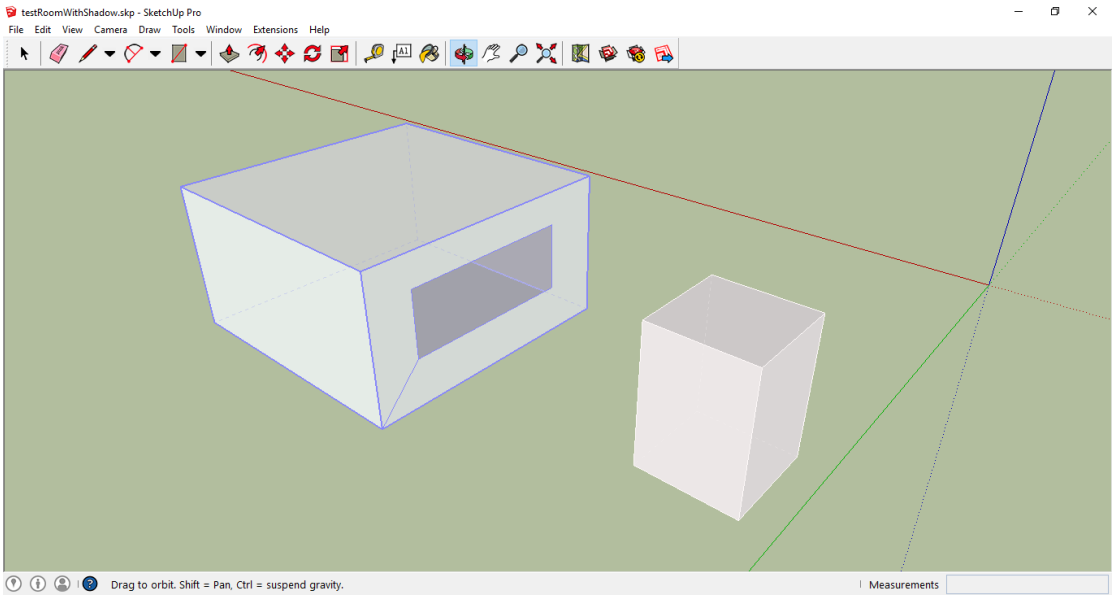
Εικόνα 6. Menu ESP-r Plug-in

6.2 Εξαγωγή αρχείου skp σε μορφή GEO

Στο παρακάτω εδάφιο περιγράφεται η διαδικασία εξαγωγής ενός μοντέλου μορφής .skp (σσ. format αρχείων SketchUp) σε αρχεία .GEO (σσ. format αρχείων ESP-r) με επιτυχία.

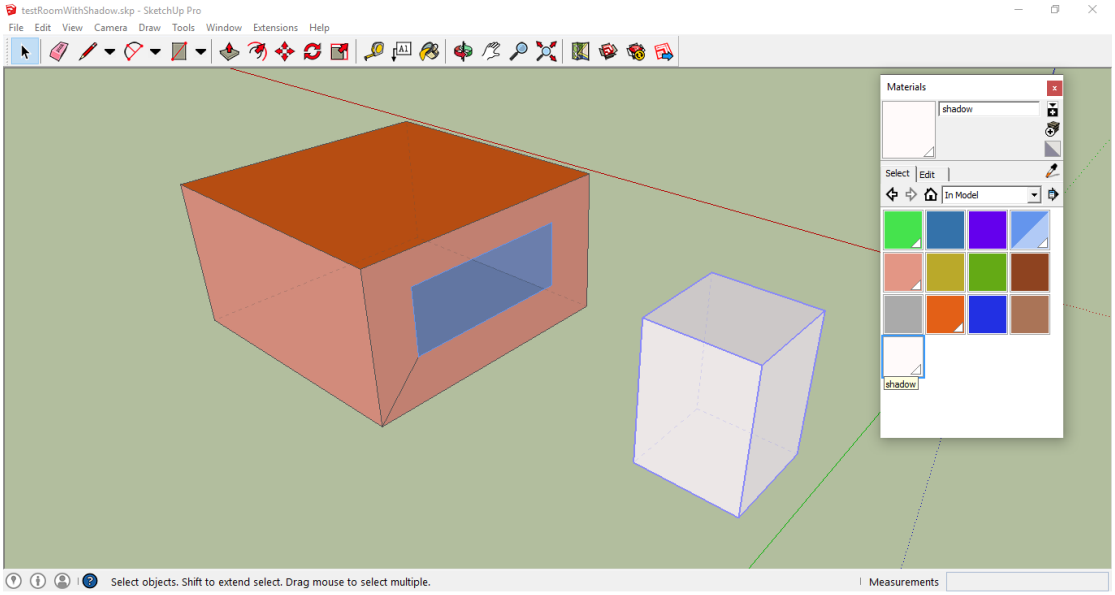
6.2.1 Δημιουργία και διαμόρφωση αρχείων .skp

Όπως αναφέρθηκε, για την ορθή λειτουργία της διεπαφής απαιτείται η τήρηση κάποιων κανόνων κατά τη διαδικασία σχεδίασης ή επεξεργασίας ενός αρχείου στο SketchUp. Στο σημείο αυτό παρουσιάζεται η προτεινόμενη σχεδίαση και επεξεργασία ενός μοντέλου. Αρχικά σχεδιάζουμε ένα μοντέλο είτε εισάγουμε ένα υπάρχον μοντέλο αρχείο .skp. Στο συγκεκριμένο παράδειγμα δημιουργούμε ένα νέο μοντέλο από την αρχή.

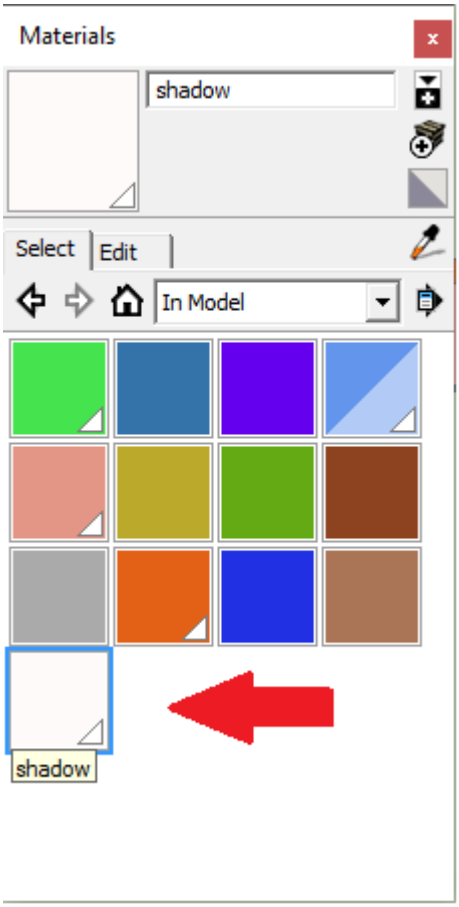


Εικόνα 7. Εικόνα

Ορίζουμε από τι υλικό (material) είναι κατασκευασμένη η κάθε επιφάνεια (surface) που περιέχεται στη σκηνή. Για δική μας ευκολία δημιουργήσαμε ένα υλικό με την ονομασία “shadow” και ορίσαμε μια λευκή απόχρωση με κωδική ονομασία “snow” για να χρωματίσουμε τις επιφάνειες που αφορούν στις σκιάσεις (obstructions).



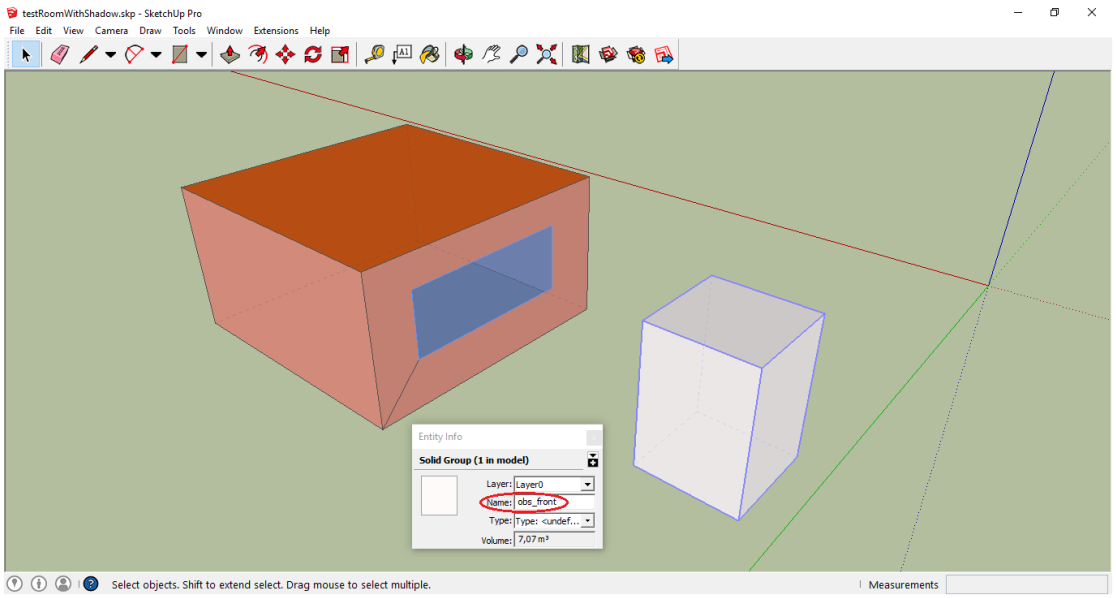
Εικόνα 8. Εικόνα



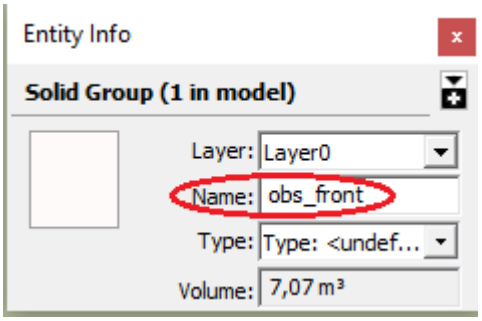
Εικόνα 9. Εικόνα

Εν συνεχεία ορίζουμε τα Groups που υπάρχουν στη σκηνή μας σύμφωνα με τους περιορισμούς που αναφέρθηκαν στο κεφάλαιο 6 και έχουν ως εξής:

- Κάθε Group που βρίσκεται στη σκηνή πρέπει να έχει μοναδικό όνομα.
- Το όνομα των Group που αφορούν σε σκιάσεις πρέπει να ξεκινά υποχρεωτικά με τη λέξη “obs”.



Εικόνα 10. Εικόνα

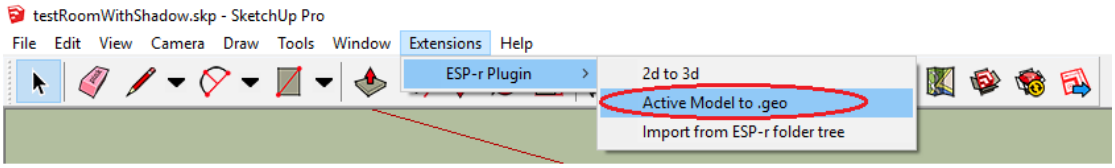


Εικόνα 11. Εικόνα

Έτσι το μοντέλο μας είναι έτοιμο προς εξαγωγή σε μορφή GEO πλέον.

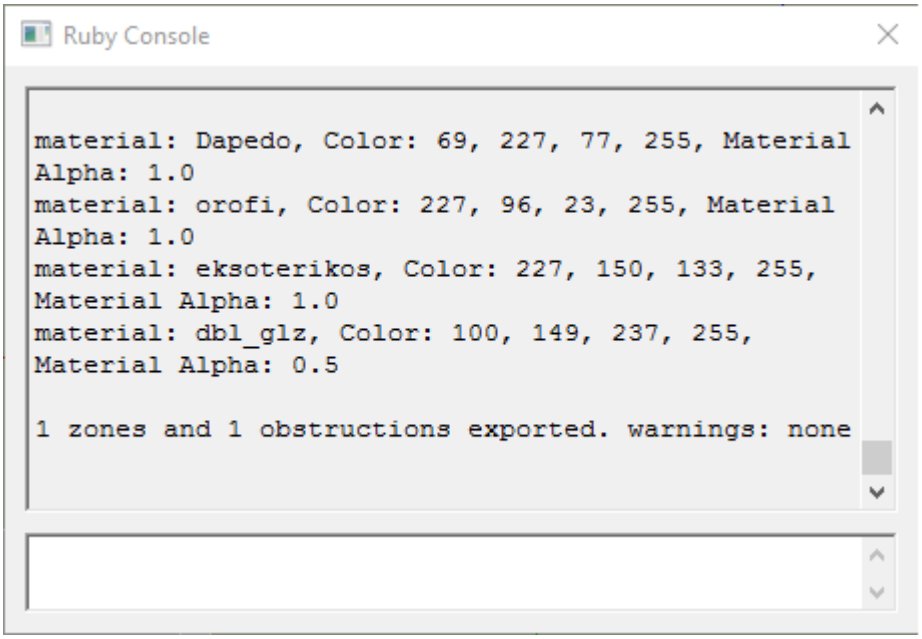
6.2.2 Εξαγωγή σε GEO.

Αφού έχουμε δημιουργήσει το κατάλληλο μοντέλο εκτελούμε την εξαγωγή επιλέγοντας από το plug-in menu του SketchUp την εντολή "*Active Model to .geo*".



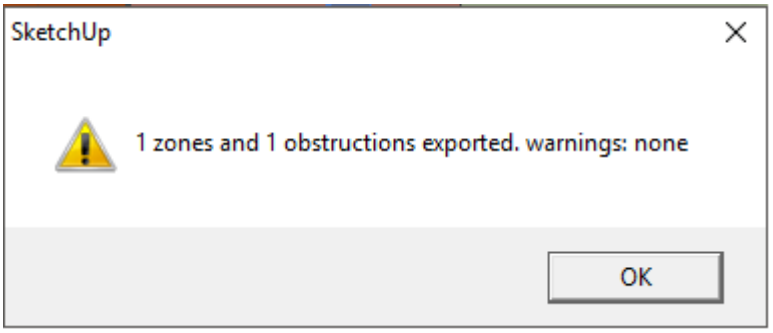
Εικόνα 12. Εικόνα

Η διαδικασία εξαγωγής ξεκινά και στη Ruby Console εμφανίζονται όλες οι διεργασίες που εκτελεί το plug-in για να συλλέξει τα στοιχεία του μοντέλου και να δημιουργήσει τα τελικά GEO αρχεία.



Εικόνα 13. Εικόνα

Εφόσον οι διεργασίες εκτελέστηκαν ορθά το SketchUp εμφανίζει ένα μήνυμα επιτυχίας στο χρήστη στο οποίο αναγράφονται ο αριθμός των ζωνών (zones) και των σκιάσεων (obstructions) καθώς και πιθανά warnings που εντοπίστηκαν κατά την εξαγωγή. Σε περίπτωση αποτυχίας το SketchUp εμφανίζει μήνυμα σφάλματος αναφέροντας σε ποιο σημείο βρέθηκε το σφάλμα.



Εικόνα 14. Εικόνα

Στη θέση πού βρίσκεται το αρχικό αρχείο .skp πλέον έχει δημιουργηθεί ένας φάκελος με την ονομασία "*export_to_espr*". Στο συγκεκριμένο φάκελο περιέχονται τα αρχεία που εξήχθησαν και είναι απαραίτητα για την απεικόνιση του μοντέλου στο ESP-r και είναι τα εξής:

- Αρχεία .geo (εμπεριέχονται στο φάκελο zones και περιγράφουν τη γεωμετρία των Group που εξήχθησαν).
- Αρχείο .mat (εμπεριέχεται στο φάκελο zones και είναι μια λίστα με τα υλικά που έχει το μοντέλο μας – το material shadow δεν περιέχεται στη λίστα αυτή καθώς πλέον η γεωμετρία των σκιάσεων περιγράφεται στα αρχεία .geo και αναγνωρίζεται μέσω του tag obs).
- Αρχείο .cnn (εμπεριέχεται στο φάκελο cfg και περιγράφει τις συνδέσεις ανάμεσα στα Group).
- Αρχείο .cfg1 (εμπεριέχεται στο φάκελο cfg και αποτελεί το configuration file ολόκληρου του αρχείου).

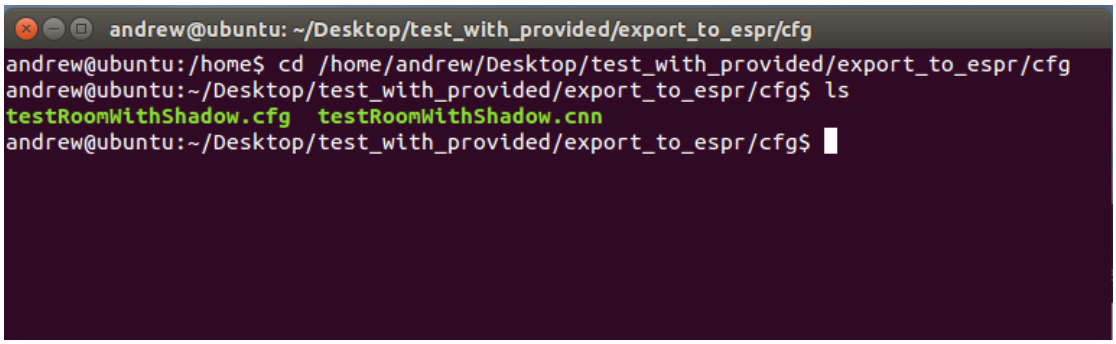
6.2.3 Επεξεργασία αρχείου .cfg

Για λόγους ευκολίας και διατήρησης του αρχείου .cfg σε περίπτωση πολλαπλών εξαγωγών, η επέκταση του αρχείου που εξάγουμε αρχικά παίρνει ως τιμή τη συμβολοσειρά *.cfg1* και για την ορθή λειτουργία απαιτείται απλώς η μετονομασία της σε *.cfg*. Επιπλέον τοποθετούμε το περιεχόμενο του αρχείου cfg_text.txt το οποίο περιέχεται στο φάκελο με το παραδοτέο υλικό, καθώς περιέχει το εισαγωγικό κομμάτι του αρχείου *.cfg* το οποίο διαμορφώνεται κατά τη δημιουργία ενός μοντέλου στο ESP-r και είναι απαραίτητο για την απεικόνιση του μοντέλου μας.

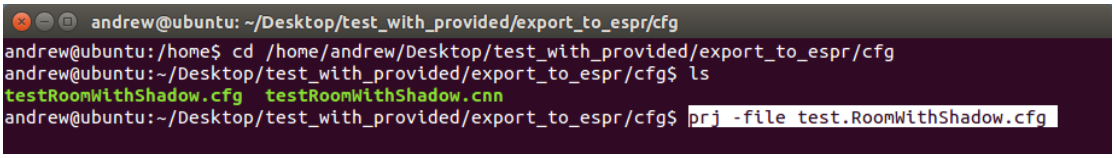
6.2.4 Εμφάνιση μοντέλου στον προσομοιωτή ESP-r.

Εφόσον έχουν ολοκληρωθεί όσα έχουν περιγραφεί στα προηγούμενα εδάφια μπορούμε πλέον να εισάγουμε το κτίριο στον ενεργειακό προσομοιωτή ESP-r. Σε περιβάλλον Unix πλέον και αφού μέσω του terminal έχουμε φτάσει στην τοποθεσία που περιέχεται

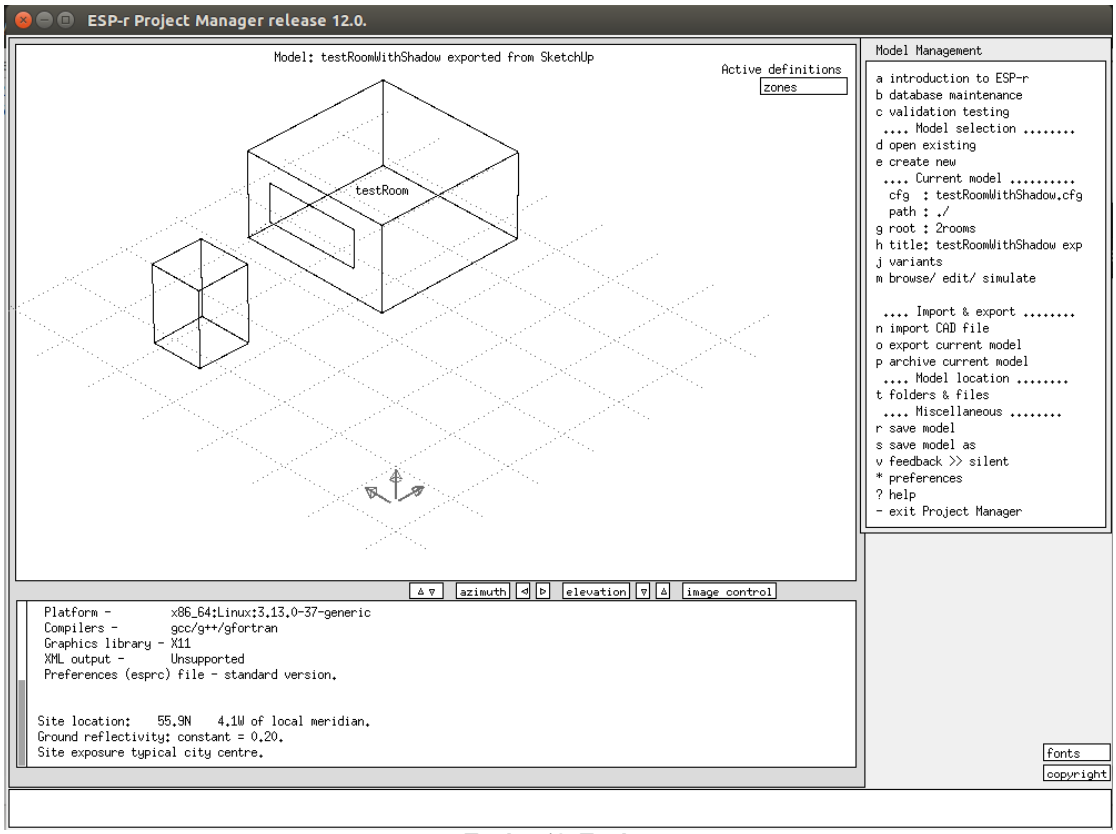
το αρχείο *.cfg* μέσω της εντολής “prj – file *File_name*.cfg” εισάγουμε το αρχείο στο ESP-r.



Εικόνα 13. Εικόνα



Εικόνα 14. Εικόνα

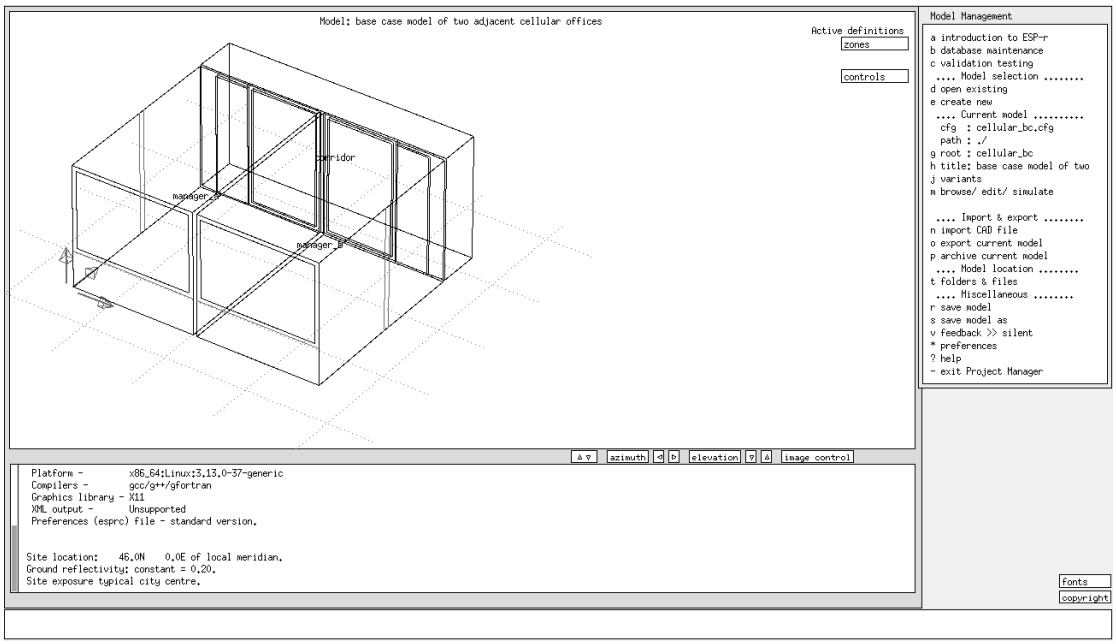


Εικόνα 13. Εικόνα

Το μοντέλο του κτιρίου που δημιουργήθηκε στο SketchUp απεικονίζεται πλέον στη σκηνή του προσομοιωτή ESP-r και παρέχεται στο χρήστη η δυνατότητα ενεργειακής προσομοίωσής του, χρησιμοποιώντας τα εργαλεία που παρέχει το ESP-r. Ακόμη αν το επιθυμεί είναι εφικτή και η επεξεργασία της γεωμετρίας του εκάστοτε exported μοντέλου.

6.3 Εισαγωγή αρχείου GEO στο SketchUp

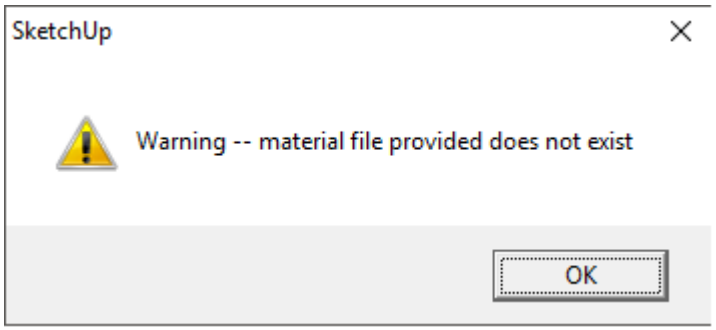
Στο παρόν εδάφιο εξηγείται και παρουσιάζεται η διαδικασία εισαγωγής (import) αρχείων GEO στο πρόγραμμα σχεδιασμού 3D μοντέλων SketchUp. Για την επίδειξη χρησιμοποιήθηκε ένα μοντέλο πρότυπο από τις βιβλιοθήκες του ESP-r με ονομασία "*cellular_bc"*. Στην εικόνα παρακάτω παρουσιάζεται το κτίριο αρχικά όπως απεικονίζεται στο ESP-r.



Εικόνα 14. Εικόνα

6.3.1 Δημιουργία αρχείου υλικών

Στην περίπτωση της εισαγωγής (import) ενός μοντέλου με format GEO στο SketchUp για την ορθή λειτουργία της διεπαφής απαιτείται η δημιουργία του αρχείου που περιέχει τα υλικά των επιφανειών (surfaces) του μοντέλου αν δεν περιέχεται ήδη στο φάκελο zones. Σε περίπτωση που δεν έχει δημιουργηθεί το αρχείο .mat κι ο χρήστης επιχειρήσει να εισάγει ένα μοντέλο, το SketchUp θα εμφανίσει warning προσδιορίζοντας την έλλειψη του αρχείου .mat μέσω μηνύματος και το plug-in σταματά να εκτελείται.



Εικόνα 15. Εικόνα

Όλα τα υλικά εμπεριέχονται στα αρχεία GEO του μοντέλου στο φάκελο zones. Πιο συγκεκριμένα τα υλικά υπάγονται στη δομή των surfaces σύμφωνα με το format των αρχείων GEO που περιγράψαμε στο κεφάλαιο 4, εδάφιο 4.5 και αποτελούν το έκτο στοιχείο που προσδιορίζει την κάθε επιφάνεια (surface). Στο παρακάτω παράδειγμα επισημάνεται με πράσινο χρώμα το στοιχείο που αποτελεί υλικό (material).

***surf**, K2.82, VERT, -, -, -, extr_wallGr, OPAQUE, EXTERIOR, 0, 0

Κατά τη δημιουργία αρκεί ο χρήστης να δημιουργήσει ένα αρχείο .txt και να το αποθηκεύσει σε μορφή .mat στο φάκελο zones. Τα ορίσματα του αρχείου τα οποία διαχωρίζονται με "," είναι το όνομα των υλικών που εντοπίζεται στα αρχεία GEO καθώς και οι τιμές του εκάστοτε χρώματος (τιμές RGB) καθώς και η τιμή και η ένταση της αδιαφάνειας αντίστοιχα. Επίσης υπάρχει η δυνατότητα να δοθεί από το χρήστη ένα μοναδικό χρώμα σε όλα τα υλικά (για λόγους ευκολίας) και στη συνέχεια να διαμορφώσει τα χρώματα μέσω της πλατφόρμας του SketchUp. Συνεπώς ένα αρχείο .mat έχει την εξής μορφή.

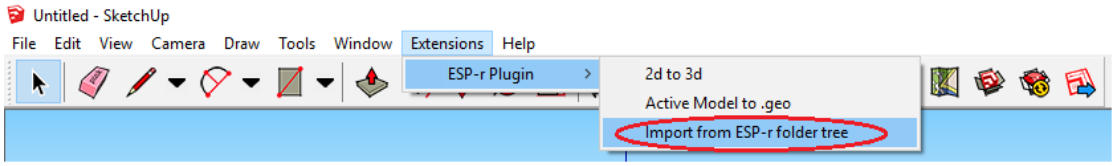
```
gyp_gyp_ptn,255,250,250,100,1.0
insul_frame,255,250,250,100,1.0
door,255,250,250,100,1.0
ceiling,255,250,250,100,1.0
susp_flr_re,255,250,250,100,1.0
dbl_glz,255,250,250,100,1.0
gyp_gyp_ptn,255,250,250,100,1.0
```

Εικόνα 13. Εικόνα

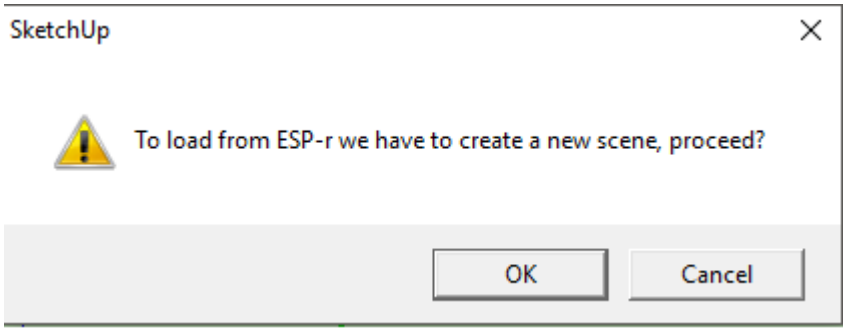
6.3.2 Εισαγωγή GEO στο SketchUp

Εφόσον έχουν ολοκληρωθεί οι απαραίτητες ενέργειες προχωράμε στην εφαρμογή του plug-in με στόχο να εισάγουμε το κτίριο στη σκηνή του SketchUp.

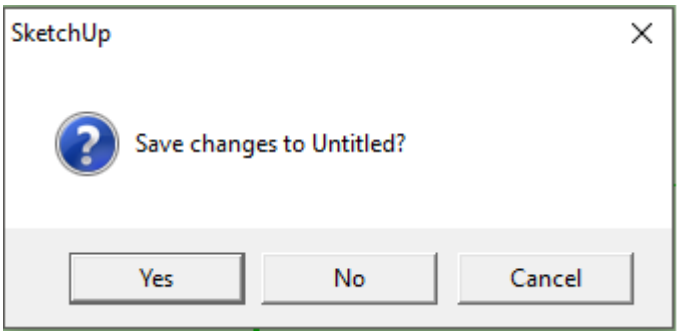
Στο plug-in menu επιλέγουμε την εντολή "*Import from ESP-r tree folder*". Το SketchUp ενημερώνει το χρήστη ότι θα δημιουργηθεί νέα σκηνή και αν επιθυμεί να αποθηκεύσει την ήδη υπάρχουσα.



Εικόνα 17. Εικόνα

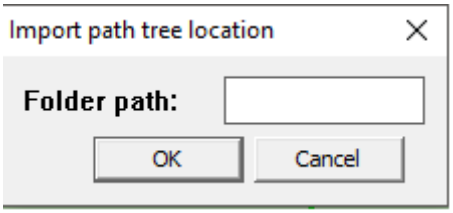


Εικόνα 18. Εικόνα



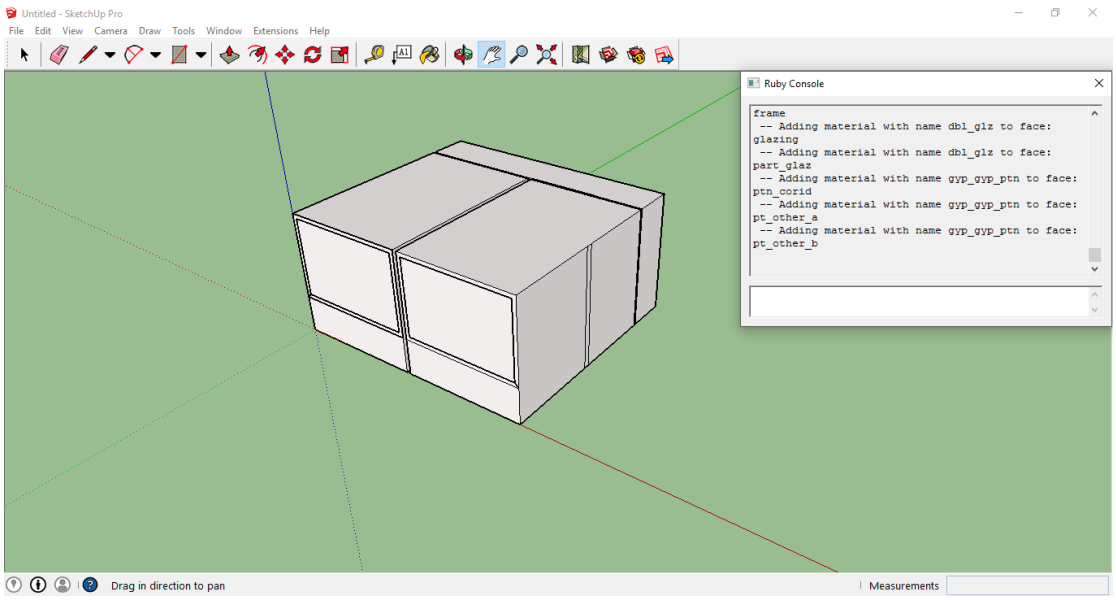
Εικόνα 19. Εικόνα

Στη συνέχεια η διεπαφή ζητά από το χρήστη να εισάγει την τοποθεσία που βρίσκεται ο φάκελος zones.



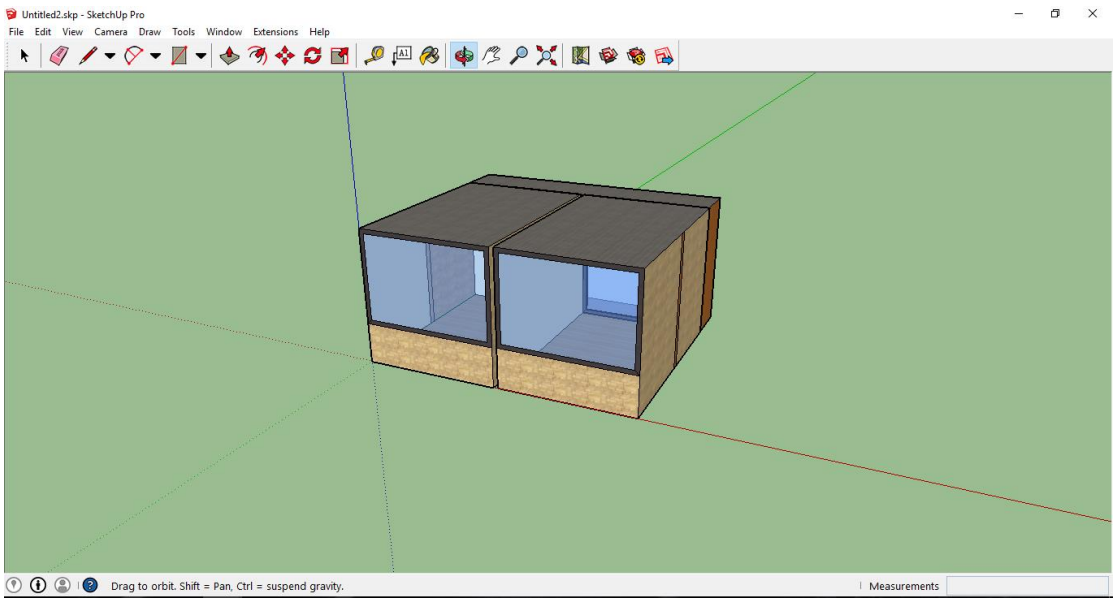
Εικόνα 20. Εικόνα

Αφού ο χρήστης εισάγει το path του φακέλου zones το οποίο για παράδειγμα έχει τη μορφή "*C:\Users\Username\Desktop\cellular_bc*" και πατήσει "*OK*" το plug-in εκτελείται και συλλέγει στοιχεία από τα αρχεία .geo γεγονός το οποίο παρουσιάζεται και στη ruby console. Τελικώς το μοντέλο εμφανίζεται στη σκηνή του SketchUp με τα χρώματα που έχει ορίσει ο χρήστης στο αρχείο .mat.



Εικόνα 21. Εικόνα

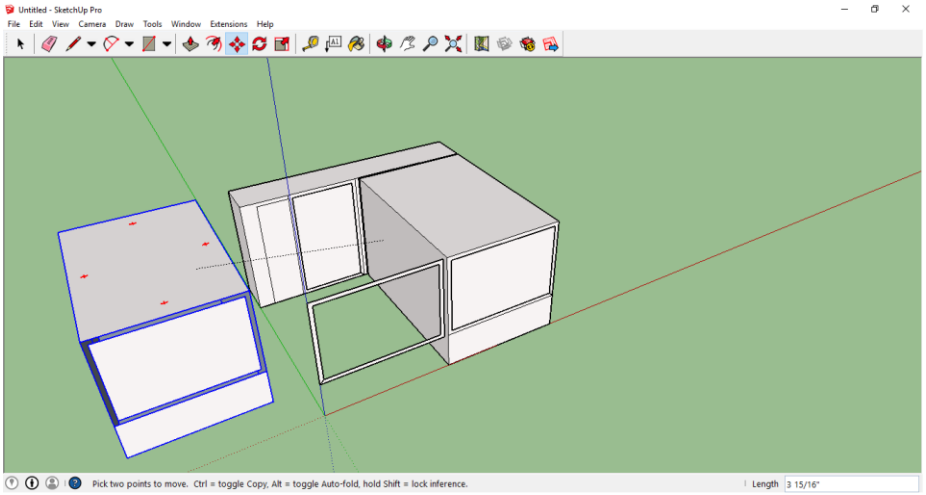
Εφόσον το μοντέλο εισήχθη επιτυχώς στη σκηνή του SketchUp ο χρήστης έχει τη δυνατότητα να το επεξεργαστεί πλήρως χρησιμοποιώντας τα εργαλεία που παρέχει το SketchUp επεμβαίνοντας σε κάθε σημείο του ώστε να το διαμορφώσει όπως ο ίδιος επιθυμεί.



Εικόνα 22. Εικόνα

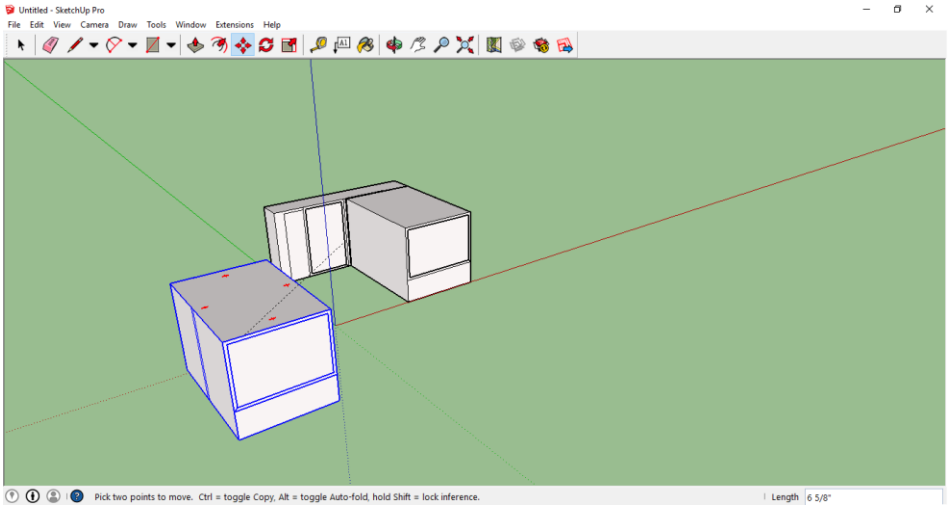
6.4 Παρατηρήσεις

Κατά την εισαγωγή (import) αρχείων GEO στο SketchUp παρατηρήθηκε ένα πρόβλημα (bug) το οποίο δεν επιλύθηκε και αποτρέπει την εμφάνιση των αντικειμένων που δημιουργούν σκιάσεις (obstructions) στη σκηνή. Η γεωμετρία των σκιάσεων αναγνωρίζεται και διαβάζεται ορθώς από το plug-in ωστόσο αποτυγχάνει να ολοκληρώσει την απεικόνιση των σκιάσεων. Το πρόβλημα εντοπίζεται στη δημιουργία Group. Στο μοντέλο που παρουσιάστηκε στο προηγούμενο εδάφιο παρατηρήθηκε ότι εξαιτίας μιας πολύ μικρής απόκλισης συντεταγμένων δεν εξήχθη σωστά το Group ενός zone.



Εικόνα 23. Εικόνα

Παρατηρούμε ότι ένα πλαίσιο παραθύρου δεν έχει ενταχθεί στο υπόλοιπο Group δηλαδή στο zone που ανήκει. Το πρόβλημα οφείλεται στην έλλειψη κάποιων ακμών που ανήκουν στα άκρα του μοντέλου και δημιουργούν κενό στο πλέγμα πιθανότατα λόγω στρογγυλοποιήσεων που πραγματοποιούνται κατά την ανάγνωση του αρχείου .geo. Πρακτικά το ζήτημα επιλύεται δημιουργώντας το Group χειροκίνητα χρησιμοποιώντας τα εργαλεία του SketchUp.



Εικόνα 24. Εικόνα

7. Μελλοντικές επεκτάσεις

Σε επίπεδο επεκτάσεων το plug-in μας σίγουρα επιδέχεται αρκετές. Καταρχάς η επίλυση του ζητήματος της γεωμετρίας που ανέκυψε κατά το import θα κάνει το plug-in αρτίως λειτουργικό. Ακόμη, η αυτοματοποίηση διεργασιών που απαιτούνται να πραγματοποιήσει μέχρι στιγμής ο χρήστης είναι εφικτές και θα καταστήσουν τη διεπαφή ακόμη πιο εύχρηστη. Τέλος υπάρχουν στοιχεία τα οποία αφορούν την πιο πολύπλοκη γεωμετρία και με τα οποία δεν ασχοληθήκαμε και με την εξαγωγή τους σίγουρα θα παρέχουν καλύτερη απεικόνιση.

8. Βιβλιογραφία

1. *THE ESP-r COOKBOOK*, Jon William Hand B.Sc, M.Arch., September 2008.
2. Sketchup API: http://ruby.sketchup.com/
3. Néstor Ubay Pérez Rodríguez source code: http://lists.strath.ac.uk/archives/esp-r/2012/002630.html
4. Class SketchUp: http://www.sketchup.com/intl/en/developer/docs/ourdoc/sketchup
5. Class Model: http://www.sketchup.com/intl/en/developer/docs/ourdoc/model
6. Class Color: http://www.sketchup.com/intl/en/developer/docs/ourdoc/color
7. Class Entity: http://www.sketchup.com/intl/en/developer/docs/ourdoc/entity
8. Class Edge: http://www.sketchup.com/intl/en/developer/docs/ourdoc/edge
9. Class Face: http://www.sketchup.com/intl/en/developer/docs/ourdoc/face
10. Class Group: http://www.sketchup.com/intl/en/developer/docs/ourdoc/group
11. Class Transformation: http://www.sketchup.com/intl/en/developer/docs/ourdoc/transformation
12. Normals: https://en.wikipedia.org/wiki/Normal_(geometry)
13. *The Well-Grounded Rubyist*, David A. Black, Manning Publications, May 2009, ISBN 9781933988658
14. *Beginning Ruby: From Novice to Professional*, Peter Cooper, ISBN 9781430223634

15. *The Well-Grounded Rubyist,* David A. Black, ISBN: 978-1933988658

16. *Eloquent Ruby,* Russ Olsen, ISBN: 978-0321584106

17. *The Ruby Programming Language,* David Flanagan, ISBN: 978-0596516178