Ατομική Διπλωματική Εργασία

ΔΙΑΔΙΚΑΣΤΙΚΗ ΜΟΝΤΕΛΟΠΟΙΗΣΗ ΤΗΣ ΠΑΛΙΑΣ ΛΕΥΚΩΣΙΑΣ ΤΟΥ 19^{ΟΥ} ΑΙΩΝΑ

Σάββας Μιχαήλ

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2009

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Διαδικαστική Μοντελοποίηση της παλιάς Λευκωσίας του 19^{ου} αιώνα

Σάββας Μιχαήλ

Επιβλέπων Καθηγητής
Γιώργος Χρυσάνθου

Η Ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου

Μάιος 2009

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον υπεύθυνο καθηγητή μου κ. Γιώργο Χρυσάνθου για την πολύτιμη βοήθεια του και τις συνεχής συμβουλές καθόλη τη διάρκεια της διπλωματικής μου εργασίας.

Επίσης θα ήθελα να ευχαριστήσω το Nicosia Master Plan, τους διδακτορικούς φοιτητές Δέσποινα Μιχαήλ, Παναγιώτη Χαραλάμπους και τους συμφοιτητές μου Μελίνο Αβερκίου, Μαρία Τζιερκέζου, Ορέστη Σπανό, Παναγιώτα Χαπούπη και Ανδρέα Σιακά για την βοήθεια που μου παρείχαν.

Περίληψη

Το θέμα της Ατομικής Διπλωματική Εργασίας με την οποία ασχολήθηκα είναι η διαδικαστική μοντελοποίηση πόλεων και συγκεκριμένα της παλιάς Λευκωσίας του 19ου αιώνα. Η δημιουργία πόλεων είναι μια πάρα πολύ σημαντική εργασία αφού έχει εφαρμογές στις κινηματογραφικές ταινίες, στα ηλεκτρονικά παιχνίδια, για αρχαιολογικούς σκοπούς, για τον τουρισμό κ.τ.λ.

Στόχος της εργασίας είναι η δημιουργία αξιόπιστων μοντέλων πολύ γρήγορα και σε χαμηλό κόστος, γράφοντας κανόνες (rule based systems) όπως για παράδειγμα τα L-systems για την παραγωγή δένδρων με κανόνες.

Έχοντας τους χάρτες της παλιάς Λευκωσίας σε ηλεκτρονική μορφή, μελετήθηκαν οι ανάλογες πληροφορίες, εξετάστηκαν και τροποποιήθηκαν από ορισμένα προγράμματα, έτσι ώστε να εισάγονται σωστά στο λογισμικό *CityEngine*. Μέσω αυτού του εργαλείου ορίσαμε τους δικούς μας κανόνες για την γραφική αναπαράσταση της παλιάς Λευκωσίας, αφού πρώτα μελετήσαμε την αρχιτεκτονική των σπιτιών της παλιάς Λευκωσίας.

Τα αποτελέσματα τα οποία παράχθηκαν από το λογισμικό *CityEngine*, μπορούμε να τα δούμε από την πιο κάτω εικόνα.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Εισαγωγή Κεφαλαίου

Στόχος του κεφαλαίου αυτού είναι να βοηθήσει τον αναγνώστη να κατανοήσει καλύτερα το θέμα της διπλωματικής εργασίας. Γίνεται μια περιγραφή της εργασίας και μια αναφορά στα L-συστήματα για περισσότερη κατανόηση. Στη συνέχεια περιγράφονται προηγούμενες διπλωματικές εργασίες, παλιών φοιτητών, οι οποίες είχαν παρόμοιο θέμα και εν κατακλείδι αναφέρουμε τις πιθανές χρήσεις των τρισδιάστατων μοντέλων που θα παραχθούν στο τέλος.

1.2 Περιγραφή Εργασίας

Όπως έχω αναφέρει, στόχος μας είναι η αναπαράσταση ως μοντέλο, της παλιάς Λευκωσίας τον $19^{ο}$ αιώνα, με διαδικαστικό τρόπο, χρησιμοποιώντας το λογισμικό CityEngine. Μέσου του λογισμικού αυτού θα γραφτούν κανόνες οι οποίοι αντιπροσωπεύουν αρχιτεκτονικούς κανόνες των σπιτιών της παλιάς Λευκωσίας.

Καταρχήν διαδικαστική μοντελοποίηση (procedural modeling) είναι ένας όρος ο οποίος περιγράφει ένα σύνολο τεχνικών στα γραφικά υπολογιστών για την δημιουργία τρισδιάστατων μοντέλων και textures από ένα σύνολο κανόνων (π.χ. L-systems).

Αυτό το σύνολο κανόνων μπορεί να υλοποιηθεί με μια γραμματική χωρίς συμφραζόμενα (context-free grammar). Υπάρχουν 2 τρόποι μοντελοποίησης της αρχιτεκτονικής με γραμματικές. Ο πρώτος τρόπος είναι χρησιμοποιώντας γραμματικές με ένα σύνολο κανόνων από χαρακτήρες[1]. Οι χαρακτήρες αποτελούν τερματικά και μή-τερματικά σύμβολα. Τα τερματικά σύμβολα θα χρησιμοποιούνται από κάποιο τελικό κανόνα για να αναπαριστά το ανάλογο γεωμετρικό σχήμα. Ο άλλος τρόπος είναι το σύνολο της γραμματικής μας να αποτελείται από σχήματα και ανάλογα με το σχήμα και τον κανόνα να προχωράς σε κάποιο άλλο σχήμα[4]. Η γραμματική η οποία χρησιμοποιείται από το λογισμικό CityEngine είναι η πρώτη και ονομάζεται CGA (Computer Graphic Architecture) shape grammar.

Η CGA shape γραμματική χρησιμοποιεί κανόνες ευανάγνωστους από τους ανθρώπους όπως τα L-systems. Η διαφορά τους είναι ότι τα L-systems είναι παράλληλη γραμματική ενώ η CGA είναι σειριακή. Δηλαδή στα L-systems μπορούν να εκτελούνται πολλοί κανόνες ταυτόχρονα ενώ στη CGA γραμματική εκτελείται ένας κανόνας κάθε φορά. Η γραμματική αυτή αποτελεί ένα συνεπή σχεδιασμό μαζικών μοντέλων και προσόψεων. Επίσης δεν περιορίζεται στην τοποθέτηση κτιρίων έτσι ώστε να είναι ευθυγραμμισμένα στους άξονες. Αυτό μας δίνει την ευκαιρία να μπορούμε να δίνουμε στο πρόγραμμα μας ως δεδομένο εισόδου χάρτες, και από αυτό να δημιουργείται αυτόματα η τρισδιάστατη μοντελοποίηση της περιοχής που έχουμε προσδιορίσει.

Αφού δώσουμε ως δεδομένο εισόδου τους ψηφιακούς χάρτες και μελετήσουμε την αρχιτεκτονική των σπιτιών της παλιάς Λευκωσίας, πρέπει να ορίσουμε τους κατάλληλους κανόνες έτσι ώστε να παράξουμε ρεαλιστικά μοντέλα τα οποία θα αναβιώνουν την παλιά Λευκωσία του 19^{ου} αιώνα.

1.3 L-Systems

Τα L-system[10] (Lindenmayer system) είναι ένα παράλληλο σύστημα κανόνων, δηλαδή μια παραλλαγή μια κανονικής γραμματικής (ένα σύνολο κανόνων και συμβόλων), τα οποία είναι γνωστά για την χρησιμοποίηση τους για την μοντελοποίηση της διαδικασίας ανάπτυξης φυτών (*εικόνα 1.1)*. Είναι όμως επίσης ικανό να μοντελοποιήσει τη μορφολογία ποικίλων οργανισμών. Τα L-systems μπορούν επίσης να χρησιμοποιηθούν για την παραγωγή όμοιων κλασμάτων όπως τα επαναληπτικά συστήματα . Τα L-systems αναπτύχθηκαν από τον Ούγγρο βιολόγο από το πανεπιστήμιο του Utrecht, Aristid Lindenmayer (1925-1989).

Η αναδρομική φύση των κανόνων των L-systems οδηγεί στην ομοιότητα και στην κλασματική μορφή(fractal like form) η οποία είναι εύκολο να περιγραφή με ένα L σύστημα. Τα μοντέλα φυτών και οι φαινομενικά φυσικές οργανικές μορφές μπορούν επίσης να οριστούν εύκολα, αυξάνοντας το αναδρομικό επίπεδο όπου η μορφή αναπτύσσεται αργά και γίνεται ακόμα πιο πολύπλοκη. Τα συστήματα αυτά είναι επίσης γνωστά για την παραγωγή τεχνητής ζωής.

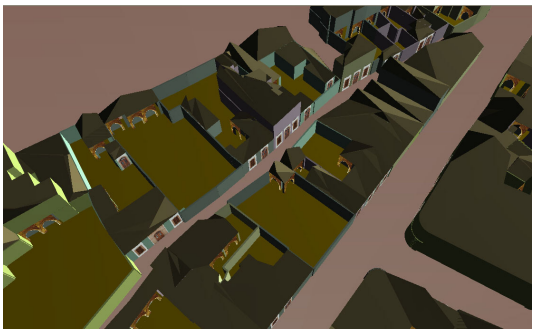
Οι κανόνες των L-systems εφαρμόζονται επαναληπτικά αρχίζοντας από την αρχική κατάσταση. Η διαφορά μεταξύ των L-systems και κανονικών γλωσσών είναι ότι στα L-systems μπορούν να εφαρμοστούν πολλοί κανόνες ταυτόχρονα σε κάθε επανάληψη. Αν οι κανόνες εφαρμόζονταν μόνο ένας κάθε φορά, τότε θα παραγόταν μια γλώσσα και όχι ένα L σύστημα. Επομένως τα L-συστήματα είναι ένα αυστηρά υποσύνολα των γλωσσών.

Εικόνα 1.1: Δέντρα δημιουργημένα χρησιμοποιώντας τα L-systems σε 3D

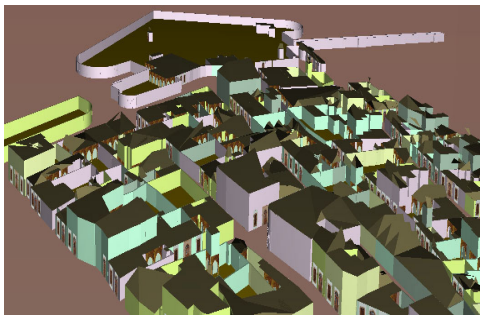
1.4 Προηγούμενες Εργασίες

1.4.1 Προηγούμενη Εργασία I

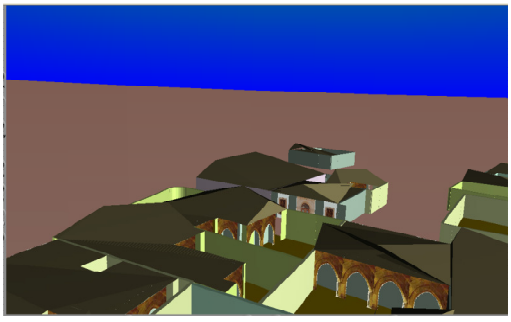
Αρχικά με αυτό το θέμα (Τρισδιάστατη Μοντελοποίηση της παλιάς Λευκωσίας) ασχολήθηκε η Μαριάννα Δικαιάκου το 2002. Μελέτησε την αρχιτεκτονική των κτιρίων της παλιάς Λευκωσίας και όρισε ένα αριθμό κανόνων για την παραγωγή μοντέλων. Ακολούθως δημιούργησε ένα πρόγραμμα χρησιμοποιώντας την γλώσσα προγραμματισμού Java, το οποίο έκτιζε τα σπίτια χωρίς να υπάρχει η στέγη. Επίσης πρόσθεσαν κάποια έτοιμα textures για να κάνουν το μοντέλο πιο ρεαλιστικό. Τα δεδομένα εισόδου ήταν ένας χάρτης της περιοχής ο οποίος αγοράστηκε από το Υπουργείο Εσωτερικών. Τα αποτελέσματα της Μαριάννας Δικαιάκου τα βλέπουμε στις εικόνες 1.2, 1.3, 1.4 και 1.5.



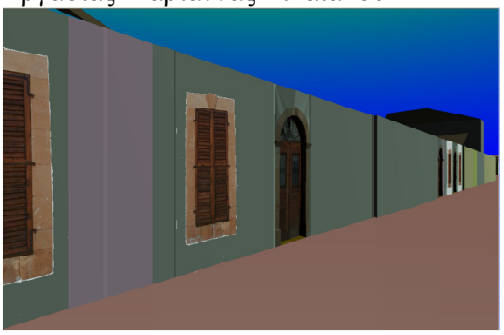
Εικόνα 1.2: Αποτελέσματα Διπλωματικής Εργασίας Μαριάννας Δικαιάκου



Εικόνα 1.3: Αποτελέσματα Διπλωματικής Εργασίας Μαριάννας Δικαιάκου



Εικόνα 1.4: Αποτελέσματα Διπλωματικής Εργασίας Μαριάννας Δικαιάκου

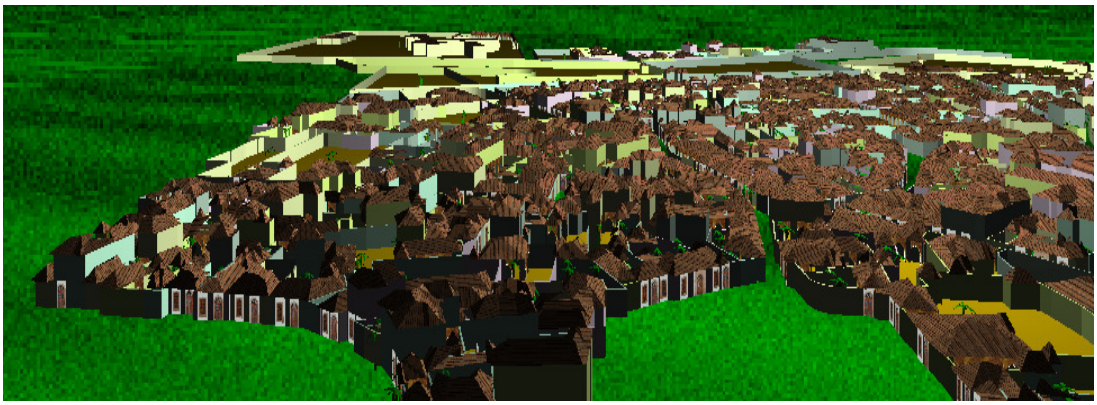


Εικόνα 1.5: Αποτελέσματα Διπλωματικής Εργασίας Μαριάννας Δικαιάκου

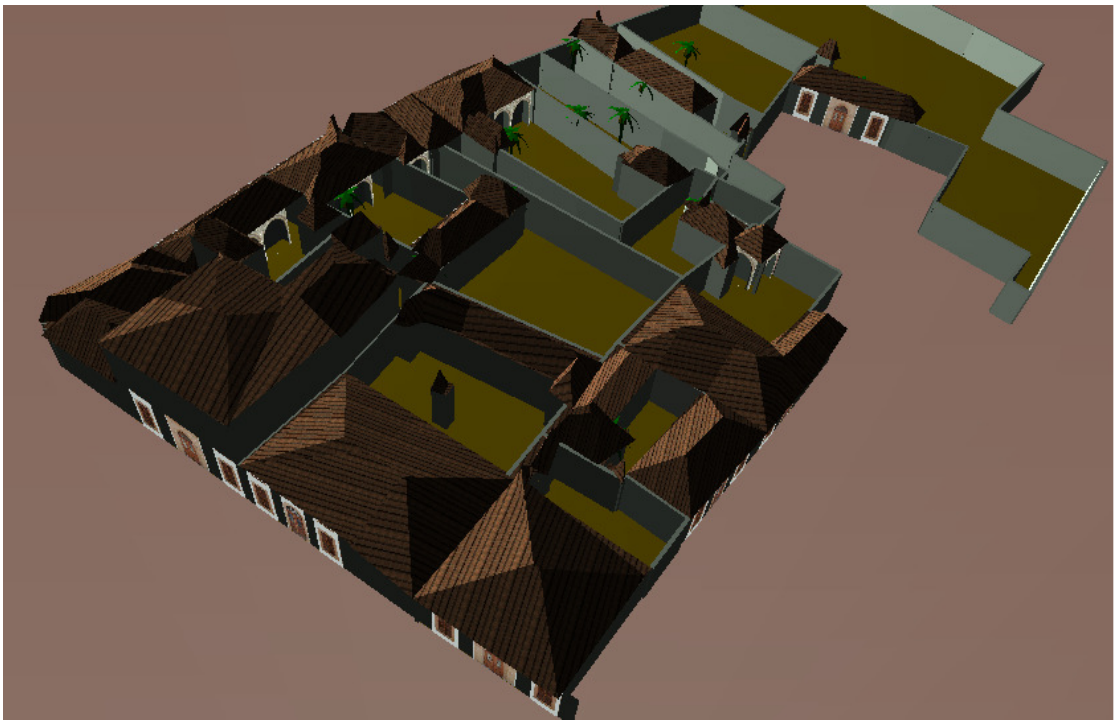
1.4.2 Προηγούμενη Εργασία II

Την προηγούμενη εργασία συνέχισε ο Ανδρέας Ευθυμίου το 2003. Ο Ανδρέας μελέτησε την προηγούμενη δουλειά της Μαριάννας και πρόσθεσε ένα αλγόριθμο για

παραγωγής στεγών στα σπίτια. Ο αλγόριθμος αυτός (straight skeleton) δημιουργεί με βάση τα αρχικά αρχιτεκτονικά σχέδια, ένα νέο σκελετό πολυγώνων, το οποίο μπορεί να χρησιμοποιηθεί για την τρισδιάστατη κατασκευή οροφής ενός κτιρίου. Τα αποτελέσματα του Ανδρέα Ευθυμίου τα βλέπουμε στην *εικόνα 1.6* και *εικόνα 1.7*.



Εικόνα 1.6: Αποτελέσματα Διπλωματικής Εργασίας Ανδρέα Ευθυμίου



Εικόνα 1.7: Αποτελέσματα Διπλωματικής Εργασίας Ανδρέα Ευθυμίου

1.5 Πιθανές χρήσεις τρισδιάστατων μοντέλων

1.5.1 Ταινίες

Μια πολύ σημαντική χρήση των μοντέλων που παράγονται από τέτοιου είδους λογισμικά είναι στις κινηματογραφικές ταινίες. Το CityEngine μπορεί να δημιουργήσει όσα κτίρια και αντικείμενα θέλει ο χρήστης. Με ένα πέρασμα μπορείς να δημιουργήσεις μεγάλες σκηνές με μεσαίο επίπεδο λεπτομέρειας για τα κτίρια. Για σκηνές με μεγάλη λεπτομέρεια στα κτίρια μπορεί ο χρήστης υποβάλει ένα σύνολο εργασιών για την παραγωγή.

1.5.2 Ηλεκτρονικά Παιχνίδια

Αυτά τα μοντέλα θα μπορούσαν ακόμη να χρησιμοποιηθούν και στη βιομηχανία των παιχνιδιών, δίνοντας ρεαλιστικές σκηνές πόλεων σε τρισδιάστατα παιχνίδια. Το CityEngine είναι εύκολο να κατανοηθεί από τους σχεδιαστές παιχνιδιών και τεχνικούς. Μπορούν πολύ εύκολα να εναλλάσσουν κανόνες σχεδίασης και στοιχείων.
Το CityEngine είναι πραγματικά ευέλικτο επιτρέποντας στο χρήστη να κτίση οποιοδήποτε είδος κτιρίου και τρισδιάστατα αντικείμενα από ελάχιστα πολύγωνα σε υψηλής ευκρίνειας γεωμετρία. Το CityEngine προσφέρει ρυθμίσεις για διάφορα επίπεδα λεπτομέρειας για την διαδικασία παραγωγής γεωμετρίας, έτσι ώστε ο χρήστης να έχει πάντα βελτιστοποιημένα στοιχεία δεδομένων.

1.5.3 Αρχαιολογικούς σκοπούς

Αυτά τα μοντέλα θα μπορούσαν να δώσουν μια εικονική πόλη η οποία θα μπορούσε να τοποθετηθεί σε μουσεία και σε άλλα κοινωνικά - πολιτισμικά σημεία του τόπου μας. Θα μπορούσε επίσης να βρίσκεται online στο διαδίκτυο όπου πολλοί χρήστες θα να έχουν πρόσβαση και να έρχονται σε επαφή με άλλους χρήστες. Έτσι ο κόσμος θα παίρνει μια εικόνα για την πολιτισμική κληρονομιά μας.

1.5.4 Τουρισμό

Ένα τέτοιο μοντέλο θα μπορούσε ακόμη να χρησιμοποιηθεί και για τουριστικούς σκοπούς, δίνοντας έτσι τη δυνατότητα σε άτομα να περιηγηθούν σ' ένα κομμάτι μιας πόλης πριν ακόμη την επισκεφθούν.

1.5.5 Προσομοιωτές

Το μοντέλο της πόλης θα μπορεί να χρησιμοποιηθεί για να τρέχουν διάφορες προσομοιώσεις όπως ροής αυτοκινήτων, ροής μάζας ανθρώπων, φυσικών καταστροφών (πλημμύρες, σεισμοί, πυρκαγιές), για τον στρατό και για την αστυνομία. Η προσομοίωση της ροής αυτοκινήτων σε πραγματικό χρόνο θα είναι αρκετά χρήσιμη αφού θα μπορούσε να χρησιμοποιηθεί για την αποσυμφόρηση των δρόμων δίνοντας εναλλακτικές διαδρομές για τους διάφορους προορισμούς. Επίσης θα μπορούσαν να χρησιμοποιηθούν για προσομοιώσεις από τον στρατό λαμβάνοντας διάφορες στρατιωτικές στρατηγικές και από την αστυνομία ελέγχοντας κάθε πιθανά σενάρια για τους διάφορους εγκληματίες.

Κεφάλαιο 2

Προηγούμενη Δουλειά

2.1 Εισαγωγή

Σε αυτό το κεφάλαιο περιγράφουμε σε μεγαλύτερο βάθος το άρθρο «Procedural Modeling of Buildings»[1].

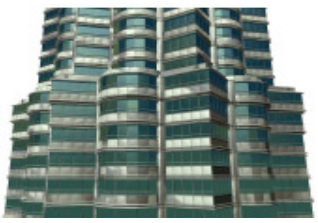
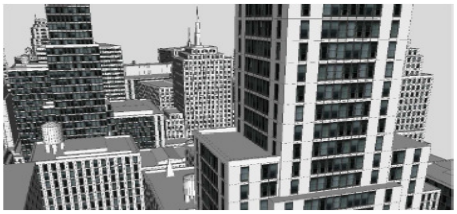
2.2 Διαδικαστική Μοντελοποίηση Πόλεων

Η CGA shape, είναι μια γραμματική σχημάτων για διαδικαστική μοντελοποίηση της CG αρχιτεκτονικής, παράγοντας σπίτια με υψηλή ποιότητα και γεωμετρική λεπτομέρεια. Μπορεί να παράξει εκτεταμένα αρχιτεκτονικά μοντέλα για ηλεκτρονικά παιχνίδια και ταινίες σε χαμηλά κόστη.

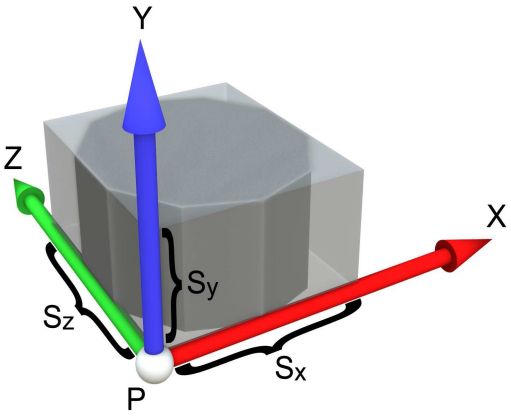
Η δημιουργία συναρπαστικών μοντέλων είναι μια σημαντική εργασία στην δημιουργία επιτυχημένων ταινιών και ηλεκτρονικών μοντέλων. Όμως, η μοντελοποίηση μεγάλων τρισδιάστατων περιβάλλων, όπως για παράδειγμα πόλεις, είναι μια πολύ ακριβείς διαδικασία και μπορεί να χρειαστεί αρκετά ανθρωποχρόνια. Ο τρόπος ο οποίος χρησιμοποιεί το λογισμικό CityEngine είναι η διαδικαστική μοντελοποίηση χρησιμοποιώντας γραμματική σχημάτων, ικανή για την δημιουργία μεγάλων πόλεων με υψηλή γεωμετρική λεπτομέρεια, αποδοτικά και γρήγορα. Η χρησιμοποίηση της CGA shape γραμματικής με κανόνες παραγωγής, εξελίσσουν επαναληπτικά κάποιο σχέδιο, δημιουργώντας σε κάθε επανάληψη, περισσότερη λεπτομέρεια. Οι κανόνες παραγωγής, αρχικά δημιουργούν ένα ακατέργαστο ογκώδες μοντέλο και στη συνέχεια κτίζεται η πρόσοψη. Στο τέλος προσθέτεται περισσότερη λεπτομέρεια όπως για παράδειγμα για τα παράθυρα, τις πόρτες κ.τ.λ. Η πιο σημαντική πληροφορία είναι ότι η επαναχρησιμοποίηση των κανόνων με μικρές διαφοροποιήσεις μπορεί να προκαλέσει τη δημιουργία μεγάλων διαφορετικών αρχιτεκτονικών, φτιάχνοντας έτσι μια ολόκληρη πόλη.

2.3 Μια γραμματική σχημάτων για την CG αρχιτεκτονική

Η CGA shape γραμματική είναι μια επέκταση της γραμματικής συνόλων, η οποία παρουσιάστηκε από τον Peter Wonka. Παρόλο που η ιδέα για τους κανόνες διαχωρισμού, παρουσιάστηκαν σε προηγούμενη δουλειά[2], ο πραγματικός ορισμός των κανόνων διαχωρισμού και επανάληψης είναι συνεισφορά των συγγραφέων αυτού του άρθρου. Ο τρόπος γραφής των κανόνων εμπνεύστηκαν από τους κανόνες των L-systems.


Σχήμα



Η γραμματική δουλεύει με μια σύνθεση σχημάτων. Ένα σχήμα αποτελείται απο ένα σύμβολο (string), τη γεωμετρία (geometric attributes) και άλλα αριθμητικά χαρακτηριστικά. Τα σχήματα αναγνωρίζονται από σύμβολα που ανήκουν είτε στα τερματικά σύμβολα είτε στα μη-τερματικά σύμβολα. Τα πιο σημαντικά γεωμετρικά χαρακτηριστικά είναι η θέση P, τα τρία διανύσματα X, Y, Z, τα οποία περιγράφουν ένα σύστημα συντεταγμένων και ένα διάνυσμα για το μέγεθος, S. Αυτά τα χαρακτηριστικά ορίζουν ένα bounding box το οποίο λέγεται scope.

Διαδικασία Παραγωγής

Μια δομή αποτελείται από ένα πεπερασμένο σύνολο βασικών σχημάτων. Η διαδικασία παραγωγής μπορεί να ξεκινήσει από ένα αυθαίρετο συνδιασμό σχημάτων Α, το οποίο λέγεται αξίωμα και προχωρά ως εξής:

(1) Επίλεξε ένα ενεργό σχήμα με το σύμβολο Β απο το σύνολο

(2) Επέλεξε ένα κανόνα παραγωγής με το Β να βρίσκεται στο αριστερό μέρος του κανόνα, για να υπολογιστεί ο διάδοχος του Β, το οποίο θα είναι ένα νέο σχήμα BNEW.

(3) Σημείωσε το σχήμα Β ως ανενεργό και πρόσθεσε το σχήμα BNEW στη δομή. Συνέχισε με τον κανόνα (1).

Όταν η δομή δεν περιέχει μη-τερματικά σύμβολα, η διαδικασία παραγωγής τερματίζεται.

Γραφή Κανόνων

Ο τρόπος με τον οποίο γράφονται οι κανόνες είναι ο εξής:

id : predecessor : cond → successor:prob

όπου το *id* είναι ένα μοναδικό αναγνωριστικό για τον κανόνα, ο predecessor είναι ένα σύμβολο το οποίο αναπαριστά το σχήμα που πρόκειται να αντικατασταθεί απο τον *successor*. Το *cond* αποτελεί την λογική έκφραση η οποία αν είναι αληθής τότε εκτελείται ο κανόνας. Επίσης ο συγκεκριμένος κανόνας επιλέγεται με πιθανότητα *prob*. Η λογική έκφραση και η πιθανότητα μπορούν να παραληφθούν (είναι προεραιτικά).

Εμβέλεια Κανόνων

Όπως και στα συστήματα L, χρησιμοποιούνται κανόνες για την τροποποίηση των σχημάτων. Το t(tx,ty,tz) είναι ένα διάνυσμα το οποίο μετακινά το σημείο P, όταν προστίθεται σε αυτό. Τα Rx(angle), Ry(angle) και Rz(angle) περιστρέφουν το σχήμα στον αντίστοιχο άξονα με βάση το τοπικό σύστημα συντεταγμένων και τα s(sx, sy, sz) καθορίζουν το μέγεθος του σχήματος. Τα σύμβολα [και] χρησιμοποιούνται για να εισάγουμε και να εξάγουμε το παρόν σχήμα στην στοίβα. Για να εισάγουμε ένα γεωμετρικό αντικείμενο χρησιμοποιούμε τον κανόνα I(object).

Κανόνας Διαχωρισμού (split rule)

Ο κανόνας διαχωρισμού χωρίζει το τρέχον σχήμα με βάση κάποιο άξονα ο οποίος δίνεται ως παράμετρο. Για παράδειγμα δίνεται ο πιο κάτω κανόνας ο οποίος διαχωρίζει την πρόσοψη ενός κτιρίου σε 4 πατώματα και σε ένα περβάζι:

1: fac → Subdiv(“Y”, 3.5,0.3,3,3,3){floor | ledge | floor | floor | floor }

Η πρώτη παράμετρος περιγράφει τον άξονα στον οποίο θα γίνει ο διαχωρισμός και οι υπόλοιπες παράμετροι περιγράφουν το μέγεθος του κάθε κομματιού. Μπορούμε να χρησιμοποιήσουμε επίσης διαχωρισμό ανάμεσα σε πολλαπλούς άξονες όπως “XY”, “XZ”, “YZ”, “XYZ”.

Επανάληψη (repeat)

Για να επιτρέψουμε μεγάλες αλλαγές στον κανόνα διαχωρισμού, για παράδειγμα να κομματιάσουμε σε πολλά μέρη ένα συγκεκριμένο σχήμα, χρησιμοποιούμε τον κανόνα της επανάληψης.

1: floor → Repeat (“X”,2) {B}

Στο πιο πάνω κανόνα, ο όροφος θα διαχωριστεί επαναληπτικά στον άξονα το X κάθε 2 μέτρα, αναθέτοντας στο κάθε κομμάτι τον κανόνα B. Στην περίπτωση που ο όροφος δεν διαχωρίζεται ακριβώς ανά δύο μέτρα, ο τελευταίος όροφος θα έχει ύψος όσο και το υπόλοιπο.

Διαχωρισμός μερών (component split)

Μέχρι στιγμής τα σχήματα μας ήταν τριών διαστάσεων. Με την πιο κάτω εντολή μπορούμε να διαχωρίσουμε ένα σχήμα σε πιο λίγες διαστάσεις:

1: a → comp(type, param) {A | B | … | Z }

Η πρώτη παράμετρος type δείχνει το είδος του διαχωρισμού ακολουθούμενο με παραμέτρου param (αν υπάρχουν). Για παράδειγμα γράφουμε την εντολή comp("faces") { A } για να δημιουργήσουμε ένα (δυσδιάστατο) σχήμα με το σύμβολο A για κάθε όψη του αρχικού τρισδιάστατου σχήματος μας. Ομοίως μπορούμε να διαχωρίσουμε το σχήμα μας σε ακμές (edges) και σε κορυφές (vertices).

2.4 Μαζική Μοντελοποίηση Κτιρίων

Όπως έχουμε εξηγήσει την γραμματική, είναι αρκετά δυνατή, έτσι ώστε να μπορούμε να δημιουργήσουμε πολύπλοκα σχήματα.

Συναρμολόγηση στερεών

Το κτίσιμο μαζικών μοντέλων αποτελείται συνήθως από την ένωση διάφορων ογκωδών σχημάτων. Η πιο απλή κατασκευή χρησιμοποιεί ένα απλό κουτί ως βασικό πρωτόγονο σχήμα. Μπορούμε στη συνέχεια να δημιουργήσουμε διάφορα μοντέλα εφαρμόζοντας τους κανόνες της μεγέθυνσης (scaling), της μεταφοράς (translation) και του διαχωρισμού (split).

Επίσης είναι αναγκαίο να συμπεριλάβουμε στο σύνολο των αρχικών σχημάτων μας και κάποιες βασικές οροφές, όπως βλέπουμε στην πιο κάτω εικόνα.

Έτσι τα μαζικά μοντέλα μπορούν να δημιουργηθούν με δύο τρόπους:

(1) Μας δίδεται ένα σύνολο κτιρίων και χρησιμοποιείται το κτίριο (building lot) ως το αξίωμα της γραμματικής. Στη συνέχεια μπορούμε να παράγουμε μαζικά μοντέλα χρησιμοποιώντας τους κανόνες μεγέθυνσης (scaling), μεταφοράς (translation), περιστροφής (rotation) και διαχωρισμού (split).

(2) Όταν εισάγουμε δεδομένα από κάποια γεωγραφική βάση (ηλεκτρονικό χάρτη), είμαστε πιο περιορισμένοι και μπορούμε να κάνουμε ελάχιστες αλλαγές στη μαζική δημιουργία μοντέλων.

Η δημιουργία μαζικών μοντέλων μπορεί να οδηγήσει στην δημιουργία πολύπλοκων μοντέλων. Ο υπολογισμός των εμφανών επιφανειών δεν είναι εύκολος και επίσης δεν είναι σαφές πως μπορούμε να γράψουμε κανόνες για δημιουργία μόνο των εμφανών επιφανειών.

Απόφραξη (Occlusion)

Γίνεται έλεγχος για να δούμε εάν υπάρχει τομή μεταξύ των σχημάτων μας. Ο πιο απλός έλεγχος, ελέγχει αν το τρέχον σχήμα τέμνεται με οποιοδήποτε άλλο. Το αποτέλεσμα του ελέγχου μπορεί να είναι: no occlusion (“none”), partial occlusion (“part”), full occlusion (“full”). Για παράδειγμα ο πιο κάτω κανόνας ελέγχει αν το μέρος της πρόσοψης, tile, τέμνεται με κάποιο άλλο σχήμα πριν να αντικατασταθεί με την πόρτα:

1: tile: Shape.occ(“all”) = = “none” → door

Στον πιο πάνω κανόνα βλέπουμε το 1 να αναπαριστά το id του κανόνα, το tile το κανόνα που πρόκειται να αντικατασταθεί (predecessor) και στη συνέχεια ακολουθεί η εντολή του CityEngine Shape.occ(“all”) = = “none”. Ο κανόνας αυτός ελεγχει αν το συγκεκριμένο σχήμα τέμνεται με οποιοδήποτε άλλο (για αυτό και η παράμετρος all). Θα μπορούσαμε να ελέξουμε αντί με όλα τα υπόλοιπα αντικείμενα, μόνο με τα active ή

με μόνο ένα αντικείμενο (για παράδειγμα Shape.occ("balcony")). Αφού δεν υπάρχει τομή μπορεί να εφαρμοστεί ο κανόνας door. Αλλιώς δεν εφαρμόζεται ο κανόνας.

2.5 Λογισμικό CityEngine

Το λογισμικό CityEngine διαφέρει σε αρκετά σημεία όσο αφορά το άρθρο που περιγράψαμε πιο πάνω. Συγκεκριμένα:

(1) Οι κανόνες δεν αναγράφονται όπως αναφέραμε, αλλά ως εξής:

Predecessor:

probability:

Condition : Sucessor

Else : Sucessor

Else : Sucessor

(2) Ο κανόνας διαχωρισμού (split rule) αναγράφεται:

facade → split(y) {3.5:floor | 0.3:ledge | 3:floor |3:floor |3:floor}

Ο διαχωρισμός μπορεί να γίνει μόνο στους άξονες X, Y και Z.

(3) Ο κανόνας επανάληψης (repeat rule) δεν χρησιμοποιεί την εντολή repeat. Απλά βάζουμε αστεράκι(*) στον κανόνα όπου θέλουμε να επαναληφθεί. Για παράδειγμα:

Façade → split(y) {3:floor}*

(4) Τέλος δεν υπάρχει εντολή για έλεγχο των σχημάτων αν τέμνονται με άλλα σχήματα.

Κεφάλαιο 3

Ερευνητικό Υπόβαθρο

3.1 Εισαγωγή

Στο τρίτο κεφάλαιο γίνεται μια εκτενέστερη αναφορά στο ερευνητικό υπόβαθρο της Ατομικής Διπλωματικής Εργασίας. Με τον όρο ερευνητικό υπόβαθρο εννοούμε τα επιστημονικά άρθρα καθώς επίσης και οτιδήποτε άλλο χρειάστηκε να το ερευνήσουμε. Για παράδειγμα τους ηλεκτρονικούς χάρτες της παλιάς Λευκωσίας, τα λογισμικά που χρησιμοποιήσαμε και την τυπολογία των σπιτιών.

3.2 Ερευνητικά Έγγραφα

Ένας από τους αρχικούς μας στόχους ήταν να μελετήσουμε διάφορα άρθρα τα οποία ήταν σχετικά με την διαδικαστική μοντελοποίηση πόλεων. Πιο κάτω αναφέρω τα άρθρα τα οποία μελετήθηκαν:

- Procedural Modeling of Building[1]
  Pascal Muller, Peter Wonka, Simon Haegler, Andreas Ulmer, Luc Van Gool
- Urban Design and Procedural Modeling[4]
  Peter Wonka, Pascal Muller, Ben Watson, Andy Fuller

3.3 Ψηφιακά Δεδομένα Εισόδου[5]

Τα αρχεία εισόδου τα οποία περιέχουν τα δεδομένα χωρίζουν το χάρτη σε εφτά επίπεδα, υπάρχει ακόμη ένα επιπλέον αρχείο το οποίο παρουσιάζει και τα εφτά επίπεδα μαζί. Τα εφτά αυτά επίπεδα περιέχουν διαφορετικά σύνολα πληροφοριών. Υπάρχει ένα για τα οικοδομικά διαμερίσματα, ένα για τα οικοδομικά τετράγωνα, ένα για οικόπεδα, ένα για τα σπίτια και τρία για διάφορα τοπολογικά δεδομένα. Με κάθε επίπεδο συσχετίζονται οκτώ αρχεία. Πέντε ήταν σε δεκαεξαδική μορφή και τα οποία παρουσίαζαν γραφικά δεδομένα και τρία ήταν σε μορφή κειμένου. Τα δύο που μπορούσαν να χρησιμοποιηθούν από το πρόγραμμα ήταν τα .mif και .mid τα οποία μπορούσαμε να τα διαβάσουμε με το MapInfo. Αυτά τα δύο είδη αρχείων μαζί χρησιμοποιούνται στους ψηφιακούς χάρτες, το αρχείο .mif περιέχει τα γραφικά δεδομένα και το .mid περιέχει σε μορφή κειμένου τα χαρακτηριστικά τα οποία περιγράφονται. Η επικεφαλίδα του .mif αρχείου παρουσιάζει με λεπτομέρεια τα πεδία και τους τύπους δεδομένων που βρίσκονται μέσα στο .mid αρχείο. Ένας μεγάλος αριθμός πεδίων παρουσιαζόταν σε κάθε επίπεδο, ο αριθμός ήταν μεταξύ του 18 και 30. Μέσα στους χάρτες τους οποίους πήραμε, μόνο το περίγραμμα των σπιτιών παρουσιαζόταν και καμιά άλλη λεπτομέρεια.

3.4 Λογισμικά

3.4.1 3D Studio Max 9.0

Χρησιμοποιήθηκε για την δημιουργία αρχέγονων και πολύ απλών μοντέλων. Αρχέγονα μοντέλα εννοούμε τους κύβους, σφαίρες κτλ. Το 3D Studio Max χρησιμοποιήθηκε επίσης για μερική τροποποίηση του χάρτη. Δηλαδή χρησιμοποιήθηκε για να πάρουμε τον μερικό χάρτη, σβήνοντας ορισμένα σχήματα τα οποία ήταν hidden.

3.4.2 AutoDesk Maya 2008

Με το λογισμικό μοντελοποίησης Maya δημιουργήσαμε τα περιγράμματα των σχημάτων των κτιρίων με όλες τις απαραίτητες πληροφορίες που χρειαζόμασταν από τον σαρωμένο χάρτη. Λόγω όμως της πολυπλοκότητας της διαπροσωπείας του λογισμικού αυτού, στη συνέχεια χρησιμοποιήθηκε το λογισμικό Google Sketcup.

3.4.3 Google Sketchup

Αφού είχαμε κάποια προβλήματα στην ακριβή δημιουργία των περιγραμμάτων των σπιτιών χρησιμοποιώντας το λογισμικό Maya και λόγω του ότι το η διαπροσωπεία του απευθύνεται σε έμπειρους χρήστες του λογισμικού αυτού, χρησιμοποιήθηκε το λογισμικό Google Sketchup το οποίο ήταν πολύ πιο απλό και επίσης επιλύθηκαν και τα προβλήματα ακρίβειας.

3.4.4 MapInfo 9.5

Το συγκεκριμένο λογισμικό χρησιμοποιήθηκε για να διαβάσουμε τις πληροφορίες του χάρτη. Αφού διαβάσαμε τις πληροφορίες αυτές, κάναμε export τον χάρτη σε dfx μορφή. Η dfx μορφή γίνεται αποδεκτή από τα λογισμικά AutoCAD και 3D Studio Max. Κανένα από τα δύο όμως δεν μπορούσε να τροποποιήσει τα δεδομένα και να τα κάνει export σε μορφή obj. Τα συγκεκριμένα λογισμικά έκανα export σε obj αλλά δεν κρατούσαν πληροφορίες για κάθε block(~για κάθε οικόπεδο). Αναγνώριζαν το σχήμα ως ένα ενιαίο χάρτη.

3.4.5 Okino Computer Graphics – PolyTrans

Με το PolyTrans εισάγαμε τα δεδομένα που παράξαμε από το MapInfo και ακολούθως τα κάναμε export σε obj μορφή, αφού επιλέξαμε τις ανάλογες ρυθμίσεις. Το συγκεκριμένο πρόγραμμα αναγνώριζε ξεχωριστά το κάθε block και αντιστρέφοντας τις καθέτους (normal's) για ορισμένα blocks είχαμε τα επιθυμητά αποτελέσματα.

3.4.6 CityEngine

Το λογισμικό CityEngine είναι αυτό που χρησιμοποιήθηκε για την διαδικαστική παραγωγή μοντέλων της παλιάς Λευκωσίας. Αρχικά χρησιμοποιούσαμε το trial version ενώ στη συνέχεια αγοράσαμε την ακαδημαϊκή έκδοση η οποία στοίχιζε 695 δολάρια. Οι ελάχιστες απαιτήσεις του λογισμικού όσο αφορά hardware και software είναι:

Προδιαγραφές Υπολογιστή:

- 2GHz dual core CPU or better (at least Pentium4 compatible Intel/AMD)
- 2 GB of RAM or more
- 500 mb of free disk space or more
- Graphics Card
    - NVIDIA: Geforce 6xxx / Quadro or better NVIDIA graphics card
    - ATI: Radeon X1600 or better ATI graphics card
- Network adapter

Λειτουργικά Συστήματα:

- Windows XP 64bit, Windows Vista 64bit
- Windows XP, Windows Vista
- Mac OS X 10.5 (Leopard) 32bit, Intel only
- Linux x86 (only 32-bit versions supported)
    - >= 2.2.1 of the GTK+ widget toolkit and associated libraries (GLib, Pango)
    - for help system: install xulrunner package and add environment variable "MOZILLA_FIVE_HOME" pointing to the xulrunner

Λογισμικά:

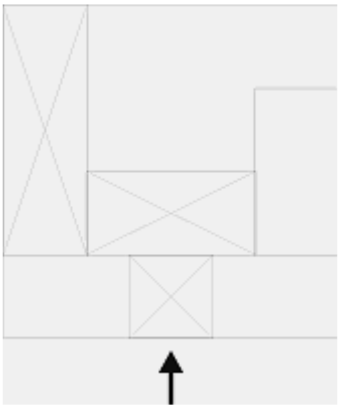
- OpenGL 2.0 driver (e.g. latest NVIDIA or ATI drivers)
- For Autodesk Maya and Max, the FBX plug-in 2009.3 is required

3.5 Τυπολογία σπιτιών[6]

Τα περισσότερα σπίτια στην περιοχή της Χρυσαλινιώτισσας, του κομματιού της παλιάς Λευκωσίας που μοντελοποιούμε, έχουν κρατήσει σε μεγάλο βαθμό με ελάχιστες αλλαγές τη αρχική τους μορφή και στυλ. Για την αναγνώριση της τυπολογία των σπιτιών της Λευκωσίας έχει χρησιμοποιηθεί ένα βιβλίο του Danilo[3]. Στο βιβλίο αυτό αναφέρετε και ο αρχιτεκτονικός ρυθμός που χρησιμοποιείται στην περιοχή. Ακόμη το βιβλίο αναφέρει με μεγάλη λεπτομέρεια για το πως το κάθε στυλ σπιτιού έχει γεννηθεί και τις διάφορες επεκτάσεις και τροποποιήσεις τις οποίες έτυχε. Στο σύνολο έχει τέσσερις κατηγορίες σπιτιών, αλλά μερικά στυλ έχουν αναπτυχθεί με τέτοιο τρόπο που

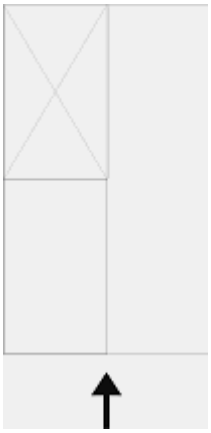
τα συγχύζουμε με άλλα στυλ. Οι κατηγορίες αναφέρονται πιο κάτω, μαζί με μια μικρή περιγραφή των κύριων χαρακτηριστικών του κάθε τύπου:

- *Σπίτι με αυλή:* έχει μεγάλη εσωτερική αυλή και ένα ή περισσότερα κτίρια του σπιτιού να βρίσκονται γύρω του. Η πόρτα άνοιγε σε ένα προθάλαμο ο οποίος οδηγούσε κατευθείαν στη εσωτερική αυλή. Μεγάλες εσωτερικές αυλές ήταν πολύ κοινό χαρακτηριστικό αυτό των σπιτιών (*εικόνα 3.1*).



Εικόνα 3.1: Αρχικό σπίτι με αυλή

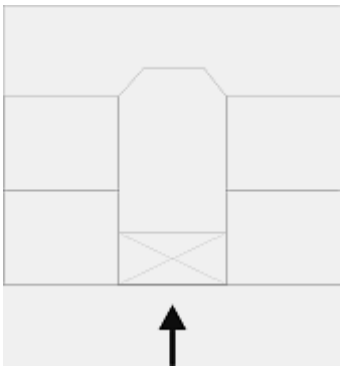
- *Σπίτι με μικρή αυλή:* αυτός ο τύπος μοιάζει αρκετά με την πιο πάνω κατηγορία, αλλά είναι αρκετά πιο σεμνό σε μέγεθος. Η πόρτα μπορούσε να ανοίγει σε προθάλαμο ή απευθείας στην αυλή. Η πλειοψηφία αυτών των σπιτιών έχει ένα όροφο ή ένα πολύ μικρό δεύτερο όροφο (*εικόνα 3.2*).



Εικόνα 3.2: Σπίτι με μικρή αυλή

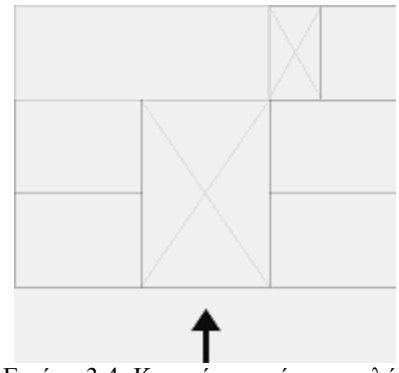
- *Σπίτι σε σειρά:* αυτά τα σπίτια πρωτοπαρουσιάστηκαν γύρω στο τέλος του 19^{ου} αιώνα. Σειρές από σπίτια με το ίδιο αρχιτεκτονικό σχέδιο κτίζονταν σε διάφορα

σημεία της πόλης. Σε αυτό τον τύπο η αυλή έχει μετακινηθεί στο πίσω μέρος του σπιτιού και ταυτόχρονα έχει ελαττωθεί και το μέγεθος της, δεν βρίσκεται πλέον στο κέντρο του σπιτιού. Σε όλες σχεδόν τις περιπτώσεις κανένα δωμάτιο δεν βρίσκεται μέσα στην αυλή ή να εφάπτεται στον πίσω τοίχο (*εικόνα 3.3*).



Εικόνα 3.3: Σπίτι σε σειρά

- *Καινούριο σπίτι με αυλή:* αυτός ο τύπος σπιτιού είναι ο πιο καινούριος, ήταν πολύ δημοφιλές γύρω στο 1930. Βασικά είναι ένας συνδυασμός το δύο προηγούμενων τύπων. Η πόρτα οδηγεί στην αυλή, όπως και στο πρώτο τύπο, αλλά το κτίριο έχει κερδίσει σε μέγεθος και είναι πιο κοντινό στο τρίτο τύπο. Μια πτέρυγα του σπιτιού συνεχίζει κατά μήκος της αυλής, το οποίο δεν το συναντούμε στο τρίτο τύπο (*εικόνα 3.4*).



Εικόνα 3.4: Καινούριο σπίτι με αυλή

Στα πιο πάνω σχεδιαγράμματα οι περιοχές με το σταυρό αντιπροσωπεύουν τους προθάλαμους και τα μικρά τόξα υποδεικνύουν τις εισόδους των σπιτιών.

Οι πιο πάνω περιγραφές αντιπροσωπεύουν τυπικά παραδείγματα του κάθε αρχιτεκτονικού στυλ. Δυστυχώς πολύ λίγα σπίτια τυγχάνει να έχουν αυτά τα πορτραίτα.

Δωμάτια έχουν τοποθετηθεί ή έχουν τύχει αλλαγές αρκετές φορές, τα οικόπεδα έχουν πάρει πολύ παράξενα σχήματα, κάνοντας έτσι πιο δύσκολη την αναγνώριση του κάθε τύπου με σιγουριά.

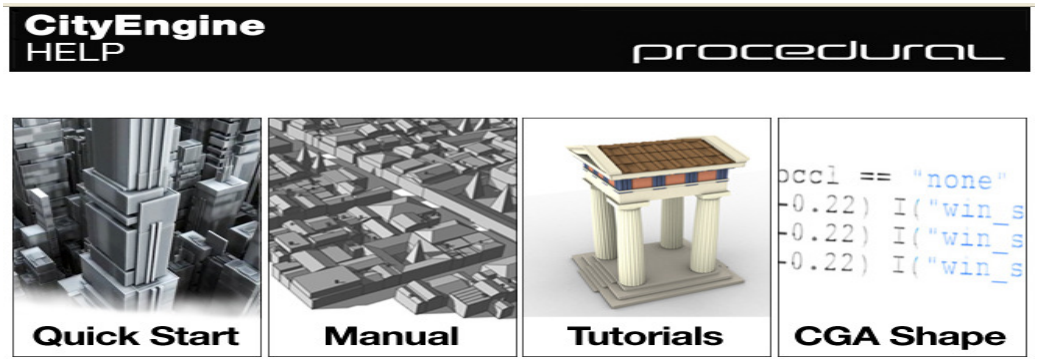
Κεφάλαιο 4

Υλοποίηση

4.1 Εισαγωγή

Το μεγαλύτερο μέρος της Διπλωματικής εργασίας αφορούσε την υλοποίηση. Αρχικός στόχος μας ήταν η εκμάθηση του CityEngine. Δηλαδή η εξοικειώσει με την διεπιφάνεια του λογισμικού και να μάθουμε το τρόπο λειτουργίας των κανόνων. Στην συνέχεια τροποποιήσαμε τα δεδομένα των ψηφιακών χαρτών έτσι ώστε να μπορούν να εισάγονται στο CityEngine. Αφού έγινε αυτό, ο στόχος μας τώρα ήταν να δημιουργήσουμε ένα σπίτι με βάση φωτογραφίες τις παλιάς Λευκωσίας και μετά να χρησιμοποιήσουμε τους ίδιους κανόνες για την δημιουργία σπιτιών σε ένα κομμάτι του χάρτη. Στο τέλος θα εφαρμόζαμε τους κανόνες σε ολόκληρο το χάρτη. Λόγω όμως διαφόρων προβλημάτων η πρώτη μεθοδολογία δεν είχε τα επιθυμητά αποτελέσματα για αυτό επιλέξαμε μια σχετικά διαφορετική μέθοδο. Χρησιμοποιήσαμε κανονικούς χάρτες τις Λευκωσίας, σαρώνοντας τους από ένα σαρωτή και μέσω κάποιων λογισμικών μοντελοποίησης δημιουργήθηκαν τα περιγράμματα των σπιτιών, με περισσότερη λεπτομέρεια. Δηλαδή δημιουργήσαμε το δικό μας ηλεκτρονικό μερικό χάρτη και τον εισάγαμε στο λογισμικό CityEngine.

4.2 Εκμάθηση CityEngine



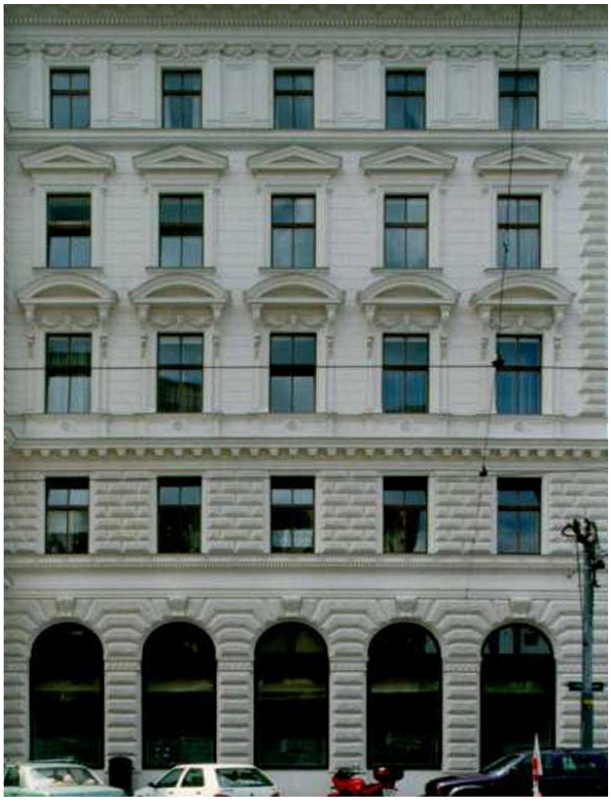
Στην αρχή χρησιμοποιήσαμε το trial version του CityEngine για να δούμε αν το λογισμικό πληροί τις προδιαγραφές για την ανακατασκευή της παλιάς Λευκωσίας. Αφού παρατηρήσαμε ότι μπορούσαμε να κτίζουμε μοντέλα εύκολα και γρήγορα, αγοράσαμε το ακαδημαϊκή έκδοση του CityEngine. Για την καλύτερη εκμάθηση του λογισμικού χρησιμοποιήσαμε το forum της ιστοσελίδας του[8], τα tutorials[9] και το online help Documentation[7]. Ένα από τα πιο σημαντικά tutorials τα οποία βοήθησαν στην κατανόηση των κανόνων αλλά και στην κατασκευή των σπιτιών ήταν το ακόλουθο:

4.2.1 Φροντιστήριο μοντελοποίηση πρόσοψης

Στόχος του φροντιστηρίου αυτού(tutorial 07)[9] ήταν η μοντελοποίηση της πρόσοψης ενός κτιρίου από αληθινή φωτογραφία. Όπως ακριβώς και η περίπτωση μας. Η πρόσοψη η οποία θα μοντελοποιήσουμε φαίνεται στην *εικόνα 4.1*.

Εικόνα 4.1: Αριστερά βλέπομε την φωτογραφία της πρόσοψης του σπιτιου και δεξιά το μοντέλο του σπιτιού που παράχθηκε με κανόνες απο το λογισμικό CityEngine.

Με βάση αυτό το tutorial θα κατανοήσουμε καλύτερα τον τρόπο με τον οποίο γράφονται οι κανόνες για την δημιουργία προσόψεων από πραγματική φωτογραφία. Η πρόσοψη την οποία θα μοντελοποιήσουμε τη βλέπουμε στην *εικόνα 4.2*.



Εικόνα 4.2: Πρόσοψη σπιτιού προς μοντελοποίηση με το λογισμικό CityEngine

Δημιουργώντας το αρχείο κανόνων

- New ... → CityEngine → CGA Grammar File
- Ονομάστε το αρχείο myFacade_01.cga και επιλέξτε Finish

Δημιουργείται ένα νέο αρχείο CGA το οποίο αναγράφει μόνο κάποιες πληροφορίες σε σχόλια.

Όγκος και Πρόσοψη

Αρχικά το μοντέλο δημιουργείται με την λειτουργία extrude. Χρησιμοποιείται κάποιο χαρακτηριστικό (attribute) για το ύψος του κτιρίου το οποίο τοποθετείται στην αρχή του αρχείου κανόνων.

```
attr height                    = 24
```

Στη συνέχεια γράφουμε τον αρχικό κανόνα χρησιμοποιώντας την εντολή extrude και μετά καλούμε τον επόμενο κανόνα, ο οποίος σε αυτή τη περίπτωση είναι ο *Building*.

```
Lot --> extrude(height) Building
```

Σε αυτό το παράδειγμα θα δείξουμε μόνο την πρόσοψη, έτσι χρησιμοποιούμε την εντολή διαχωρισμού comp για να απαλλαχθούμε από όλες τις όψης πλην της πρόσοψης. Ακολούθως καλούμε τον κανόνα *Frontfacade*.

```
Building --> comp(f) { front : Frontfacade}
```

Όροφοι

Εικόνα 4.3: Ανάλυση της πρόσοψης σε ορόφους

Η πρόσοψη χωρίζεται οριζοντίως σε διάφορους ορόφους (*εικόνα 4.3*), όπου κάθε όροφος έχει το ύψος του χαρακτηριστικού *floor_height*. Ο κάθε όροφος ταυτοποιείται με την παράμετρο της εντολής split, την split.index. Αυτή η παράμετρος θα περαστεί στους επόμενους κανόνες για να αποφασιστεί τι στοιχεία θα δημιουργηθούν στον κάθε όροφο. Για τον τελευταίο όροφο, ο δείκτης είναι

τυποποιημένος με τον αριθμό 999. Αυτό μας επιτρέπει να αναγνωρίζουμε το συγκεκριμένο όροφο εύκολα.

Να σημειώσουμε ότι ο κανόνας επανάληψης (repeat) για ενδιάμεσους ορόφους, επιτρέπει στο κτίριο να προσαρμόσει δυναμικά το ύψος των κτιρίων και να συμπληρώσει το υπόλοιπο κάθετο χώρο με ενδιάμεσους ορόφους.

Επιπρόσθετα χαρακτηριστικά τα οποία αφορούν τις διαστάσεις των ορόφων:

```
attr groundfloor_height        = 5.5
attr floor_height              = 4.5
```

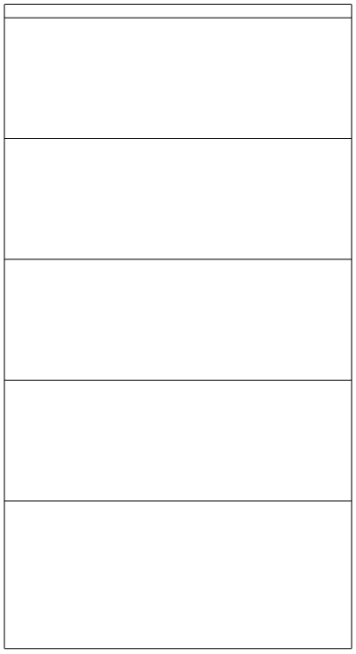
και προσθέτουμε τον κανόνα:

```
Frontfacade -->
split(y){ groundfloor_height : Floor(split.index)     //
Groundfloor
        | floor_height : Floor(split.index)           // First
Floor
        | floor_height : Floor(split.index)           // Second
Floor
        | {~floor_height : Floor(split.index)}*       // Mid
Floors
        | floor_height : Floor(999)     // Top Floor, indexed
with 999
        | 0.5 : s('1,'1,0.3) LedgeAsset} // The top ledge just
below the roof
```

Τώρα θα δημιουργήσουμε την πρόσοψη μας για πρώτη φορά:

- Επιλέξτε το lot στο 3D viewport
- Ανάθεσε το αρχείο κανόνων μέσω των Αρχικών Σχημάτων → Assign Rule File ..., επίλεξε το αρχείο .cga (λόγου χάρη myFacade_01.cga) και επιλέξτε Ok
- Πατήστε το κουμπί generate στο πάνω toolbar (ή Ctrl-g)

Το αποτέλεσμα θα πρέπει να είναι όπως φαίνεται πιο κάτω στην *εικόνα 4.4*.



Εικόνα 4.4: Η πρόσοψη αφού τη διαχωρίσαμε σε ορόφους

Περβάζι Ορόφων

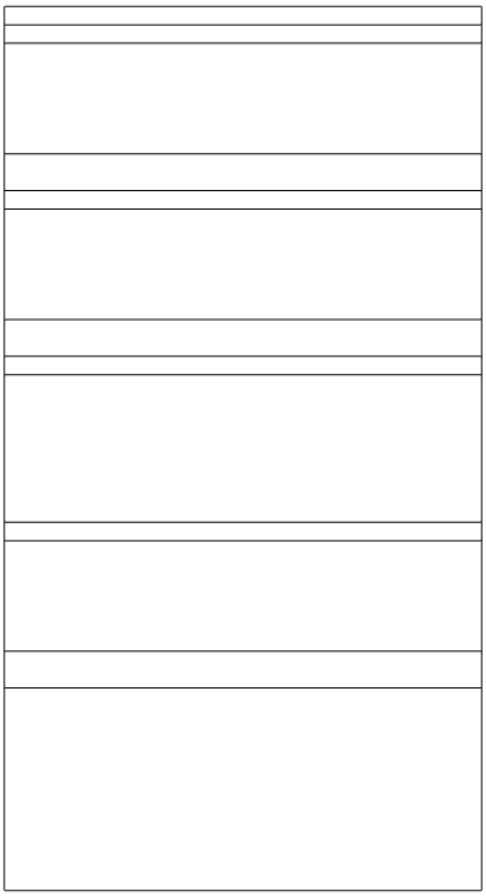
Εικόνα 4.5: Ανάλυση της πρόσοψης σε περβάζια

Τώρα, οι όροφοι θα διαχωριστούν σε περβάζια και tile shapes (*εικόνα 4.5*). Τα κάτω περβάζια (bottom ledges) είναι συγκεκριμένα για κάθε όροφο, έτσι για την δημιουργία τους χρησιμοποιούμε την παράμετρο *floorindex* και για αυτό τον κανόνα.

- Το ισόγειο (floorindex 0) δεν έχει περβάζι για αυτό και καλεί απευθείας τον κανόνα *Tiles*.
- Αφού τα παράθυρα του 2^{ου} ορόφου ξεκινούν από το επίπεδο του ορόφου, τότε δεν έχει κάτω περβάζι ο δεύτερος όροφος. Το μπαλκόνι του ορόφου αυτού θα δημιουργηθεί σε επόμενους κανόνες.
- Όλοι οι όροφοι έχουν κάτω και πάνω περβάζι και ορθογώνιες περιοχές για τα παράθυρα..

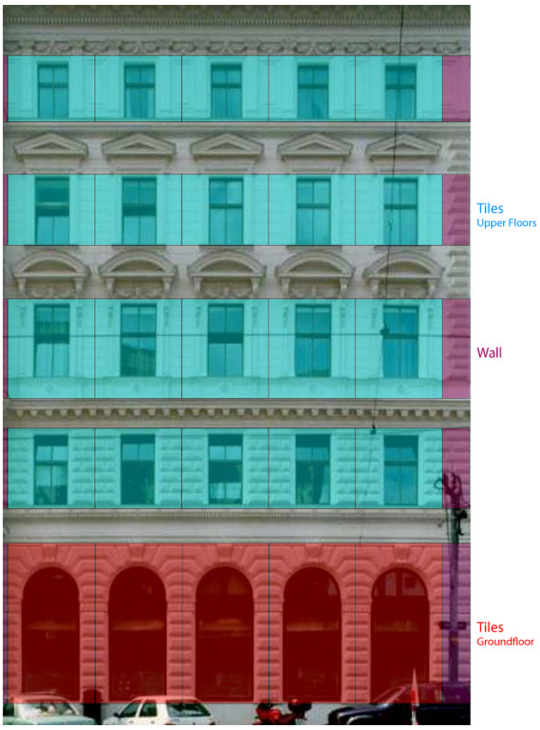
```
Floor(floorindex) -->
     case floorindex == 0 :
        Subfloor(floorindex)
     case floorindex == 2 :
        split(y){~1 : Subfloor(floorindex) | 0.5 : TopLedge}
     else :
        split(y){1 : BottomLedge(floorindex) | ~1 :
                Subfloor(floorindex) | 0.5 : TopLedge}
```

Τα αποτελέσματα της εκτέλεσης των πιο πάνω κανόνων τα βλέπουμε στην *εικόνα 4.6*.



Εικόνα 4.6: Όροφοι διαχωρισμένα με περβάζια

Υπο-όροφοι



Εικόνα 4.7: Ανάλυση υπο-όροφων

Οι υπο-όροφοι αποτελούνται από μικρές περιοχές δεξιά και αριστερά οι οποίες αποτελούν τον τοίχο και επαναλαμβάνονται μεταξύ των παραθύρων που βρίσκονται στο ενδιάμεσο τους (*εικόνα 4.7*).

Προσθέτουμε ακόμα ένα χαρακτηριστικό στην αρχή

```
attr tile_width = 3
```

και προσθέτουμε τον κανόνα

```
Subfloor(floorindex) -->
        split(x){ 0.5 : Wall(1)
                | { ~tile_width : Tile(floorindex) }*
                | 0.5 : Wall(1) }
```

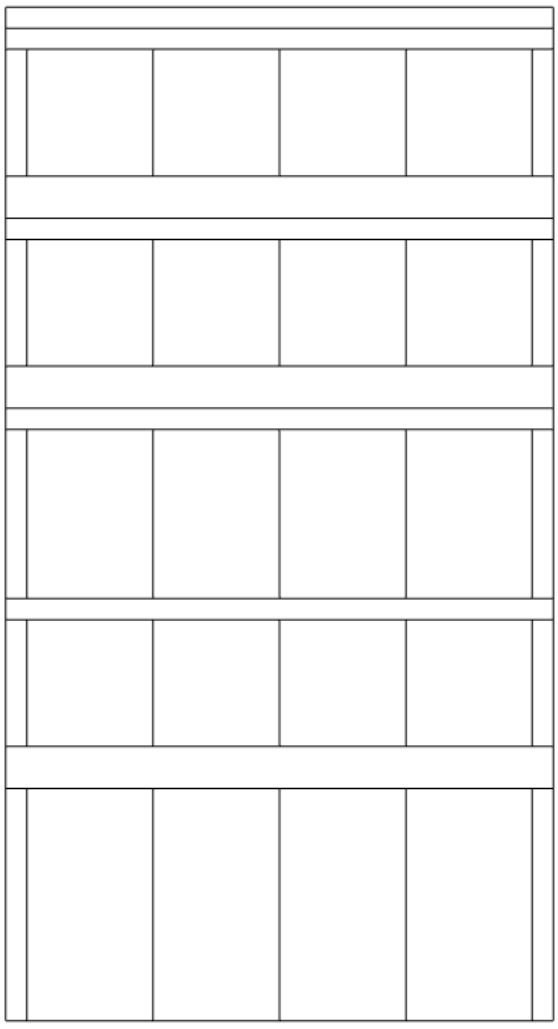
Σε αυτό το σημείο προσθέσαμε ένα κανόνα με παράμετρο, ο οποίος είναι ο *Wall*. Αυτό είναι σημαντικό στο επόμενο βήμα στο οποίο θα βάζουμε textures στην πρόσοψη. Παρατηρώντας την πρόσοψη της φωτογραφίας διακρίνουμε τρία διαφορετικά είδη τοίχων:

- 1: σκούρα τούβλα με texture ακαθαρσίας.
- 2: φωτεινά τούβλα με texture ακαθαρσίας,

- Διαφορετικά: texture ακαθαρσία μόνο.

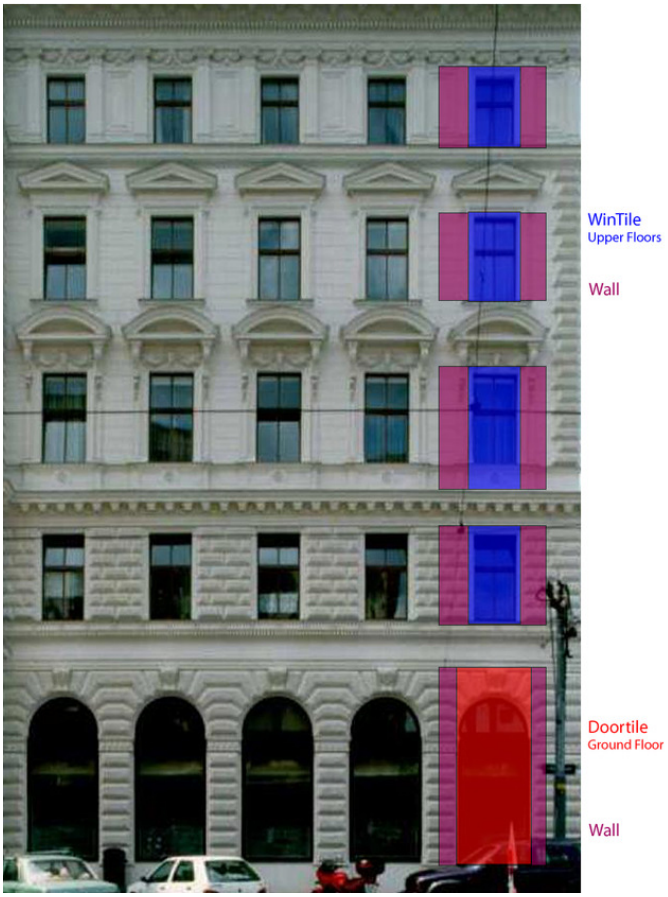
Όπως παρατηρούμε στο πιο κάτω κανόνα, τα είδη των τοίχων, έχουν πανομοιότυπα αποτελέσματα με την φωτογραφία. Στην *εικόνα 4.8* βλέπουμε το αποτέλεσμα εκτέλεσης των κανόνων οι οποίοι διαχωρίζουν την πρόσοψη σε δωμάτια.

```
Wall(style) -->
        // dark bricks with dirt
        case style == 1 :
                color(wallColor)
        // bright bricks with dirt
        case style == 2 :
                color(wallColor)
        // dirt only
        else :
                color(wallColor)
```



Εικόνα 4.8: Όροφοι χωρισμένοι σε ορθογώνια σχήματα (δωμάτια)

Παράθυρα



Εικόνα 4.9: Ανάλυση παραθύρων

Προφανώς, τα τετράγωνα (παράθυρα) σε αυτή την πρόσοψη είναι ομοιογενές (*εικόνα 4.9*). Το μόνα που διαφέρουν είναι αυτά του ισογείου. Χρησιμοποιούμε τα χαρακτηριστικά door_width και window_width για να ορίσουμε τις διάφορες διαστάσεις:

```
attr door_width                 = 2
attr window_width               = 1.4

Tile(floorindex) -->
        case floorindex == 0 :
                split(x){ ~1 : SolidWall
                |  door_width : DoorTile
                | ~1 : SolidWall }
         else :
                split(x){ ~1 : Wall(getStyle(floorindex))
                |  window_width : WindowTile(floorindex)
                | ~1 : Wall(getStyle(floorindex)) }
```

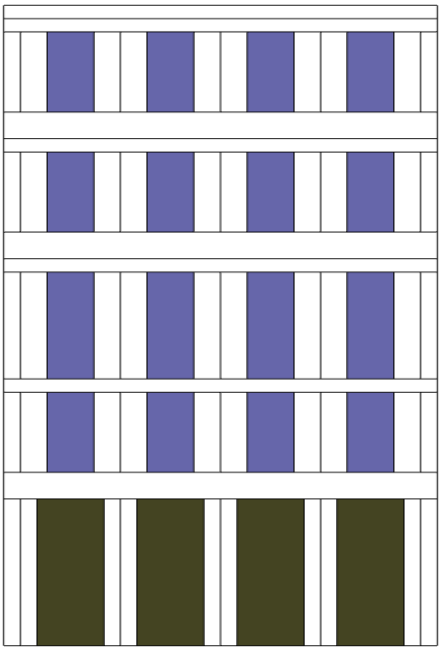
Για τα tiles του ισογείου, προσθέθηκε ο κανόνας SolidWall. Αυτό είναι αναγκαίο, αφού οι πόρτες στο ισόγειο δεν εξέχουν της πρόσοψης. Για να αποφύγουμε τις

τρύπες μεταξύ πόρτας-τοίχου, χρησιμοποιούμε ένα πρωτόγονο αντικείμενο ως τοίχο, εισάγοντας ένα κύβο με κάποιο πάχος. Το μέγεθος(πάχος) του τοίχου το ορίζουμε σαν μεταβλητή με το όνομα wall_inset.

```
wall_inset = -0.4


SolidWall -->
        s('1,'1,wall_inset)
        i("builtin:cube:notex")
        Wall(1)
```

Πιο κάτω ορίζουμε μια συνάρτηση μέσω από την οποία παίρνουμε το είδος του τοίχου με βάση το δείκτη του ορόφου (floorindex). Το πως θέσαμε τα textures για κάθε όροφο, ήταν με βάση την φωτογραφία .

```
getStyle(floorindex) =
        case floorindex == 0 : 1
        case floorindex == 1 : 1
        else : 2
```



Εικόνα 4.10: Πρόσοψη χωρισμένη σε μικρά τετράγωνα

Τα αποτελέσματα της εκτέλεσης των κανόνων για διαχωρισμό της πρόσοψης σε δωμάτια τα βλέπουμε στην *εικόνα 4.10*.

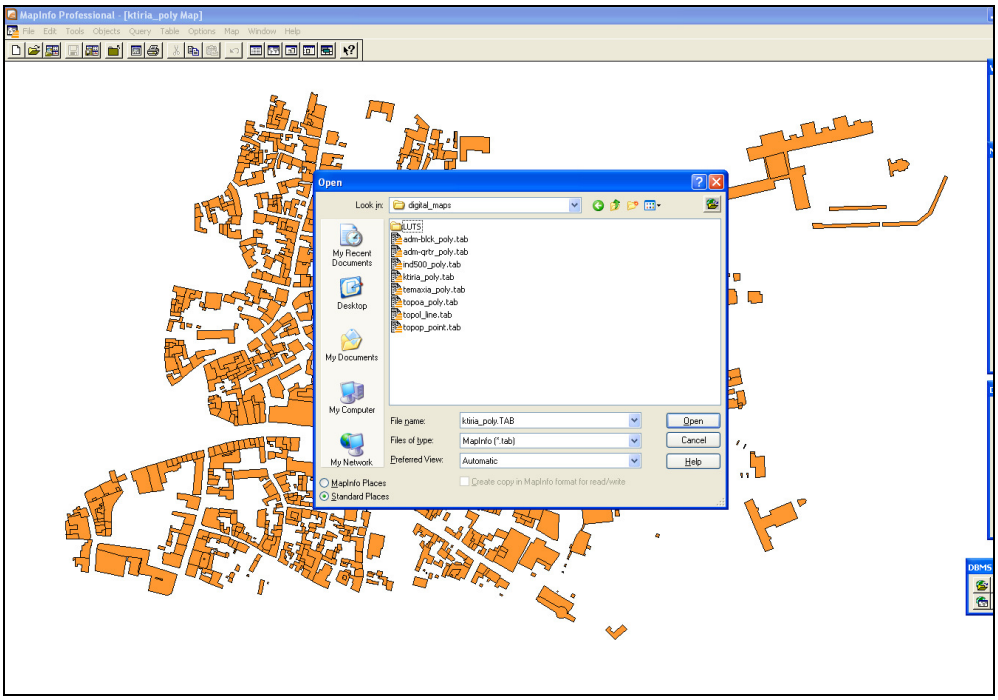
Στα υπόλοιπα 2 μέρη του φροντιστηρίου προσαρμόζονται τα μοντέλα και τα textures. Το τελικό αποτέλεσμα φαίνεται στην *εικόνα 4.11*.

Εικόνα 4.11: Παραγόμενο μοντέλο απο το λογισμικό CityEngine αφού έχουν προσθεθεί τα textures και τα μοντέλα.

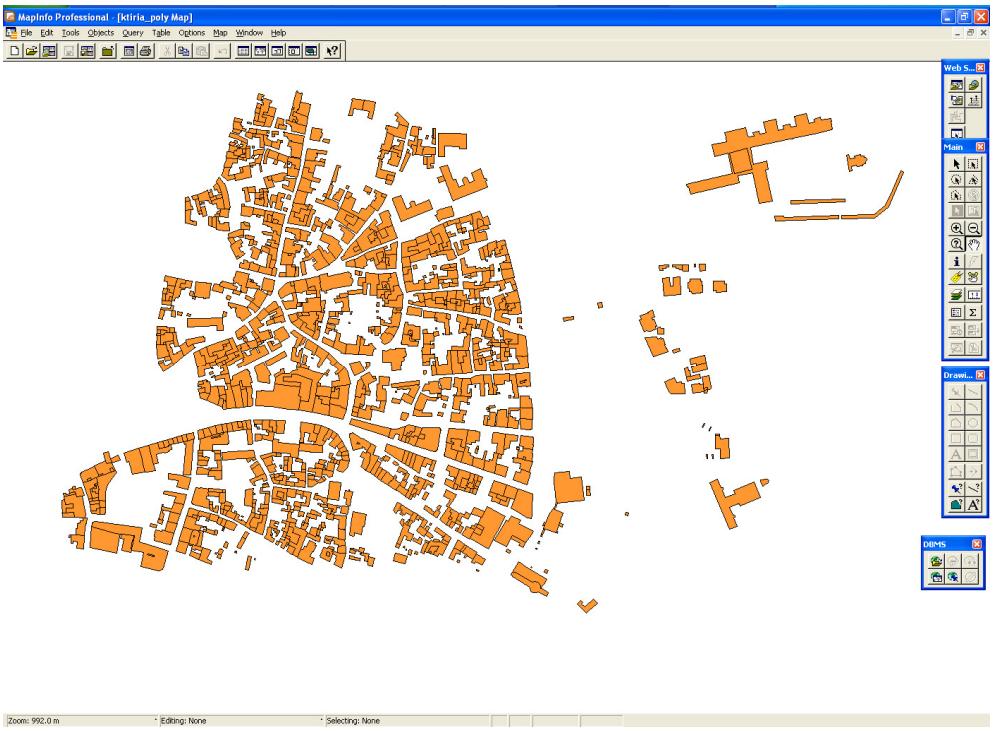
4.3 Τροποποίηση ηλεκτρονικών δεδομένων

Ένα από τα εμπόδια που αντιμετώπισα στη διπλωματική μου εργασία, ήταν η μετατροπή του ηλεκτρονικού χάρτη σε μορφή που να γίνεται αποδεκτή από το λογισμικό ***CityEngine***. Τα αρχεία του ηλεκτρονικού χάρτη ήταν της μορφής mif και tab.

Χρησιμοποιήσαμε το λογισμικό ***MapInfo 9.5*** για να ανοίξουμε τα αρχεία tab. Από τα 7 επίπεδα του χάρτη επέλεξα τα τεμάχια (*εικόνα 4.12*) και τα κτίρια. Στην *εικόνα 4.13* βλέπουμε το χάρτη και το interface του λογισμικού MapInfo.



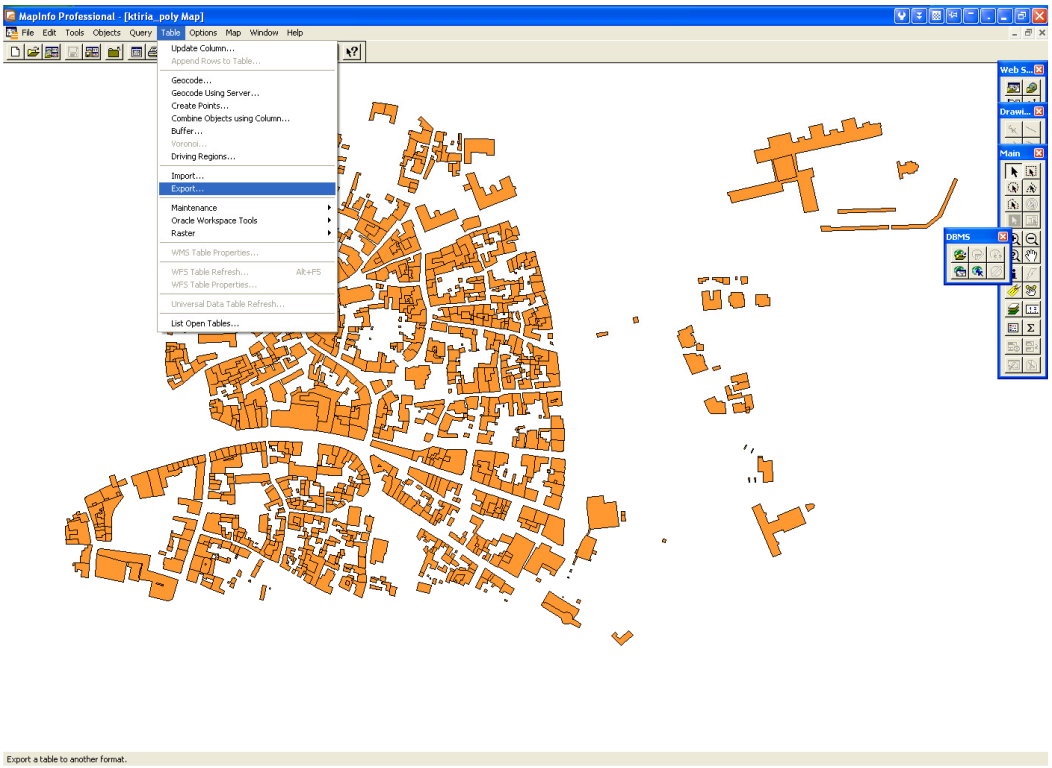
Εικόνα 4.12: Εισαγωγή του αρχείου .tab του ηλεκτρονικού χάρτη στο λογισμικό MapInfo



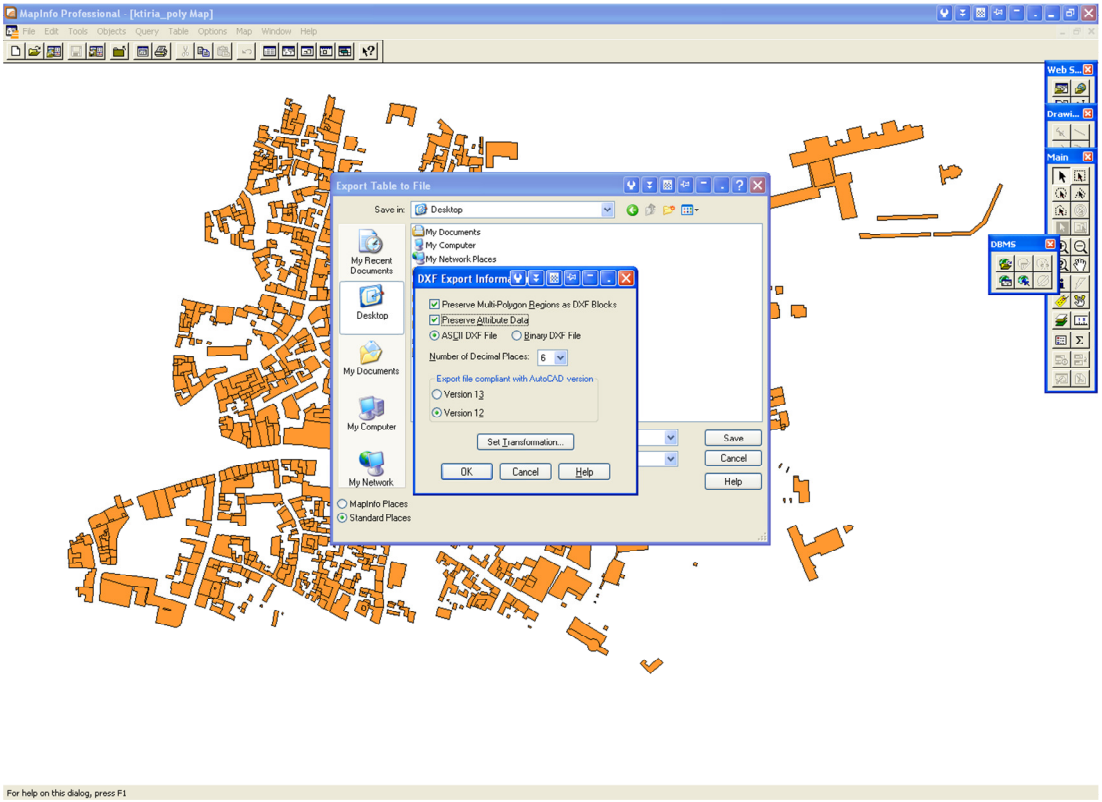
Εικόνα 4.13: Interface του λογισμικού MapInfo.

Αφού εισάξαμε τα δεδομένα στο MapInfo, τα κάναμε export (*εικόνα 4.14* και *εικόνα 4.15*), επιλέγοντας:

- Table → Export



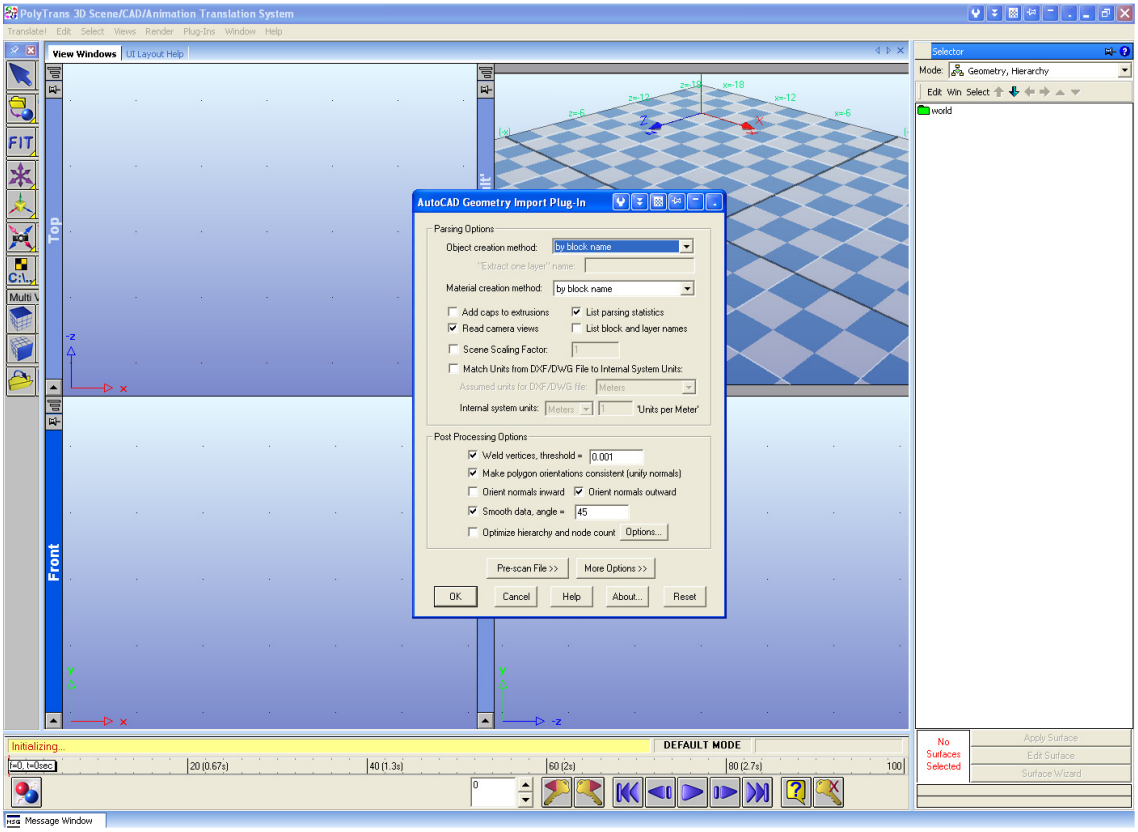
Εικόνα 4.14: Απο το λογισμικό MapInfo εξάγαμε τον χάρτη στη μορφή .dfx έτσι ώστε να τον εισάγουμε στην συνέχεια στο λογισμικό Okino Polytrans.



Εικόνα 4.15: Για τη σωστή εξαγωγή σε αρχείο .dfx παρατηρούμε στην εικόνα τις απαραίτητες ρυθμίσεις.

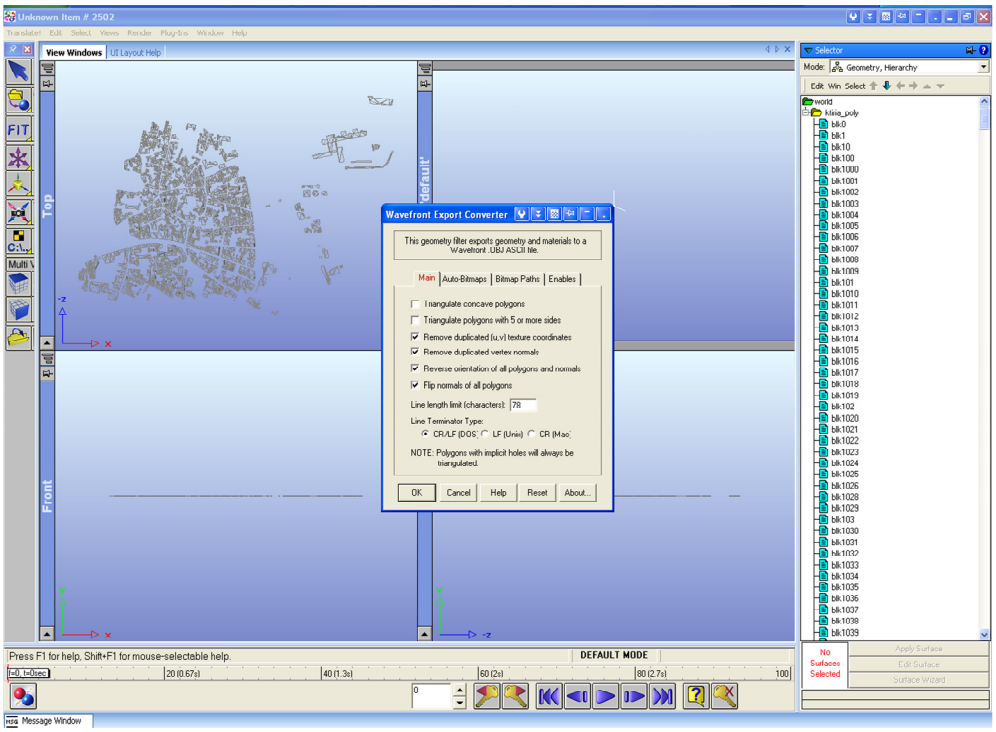
Τα νέα αρχεία που δημιουργήθηκαν ήταν της μορφής *dfx*.

Τα νέα δεδομένα τα εισάγαμε στο λογισμικό ***okino computer graphics-PolyTrans***, επιλέγοντας τις ρυθμίσεις εισαγωγής, όπως φαίνεται στην *εικόνα 4.16*.

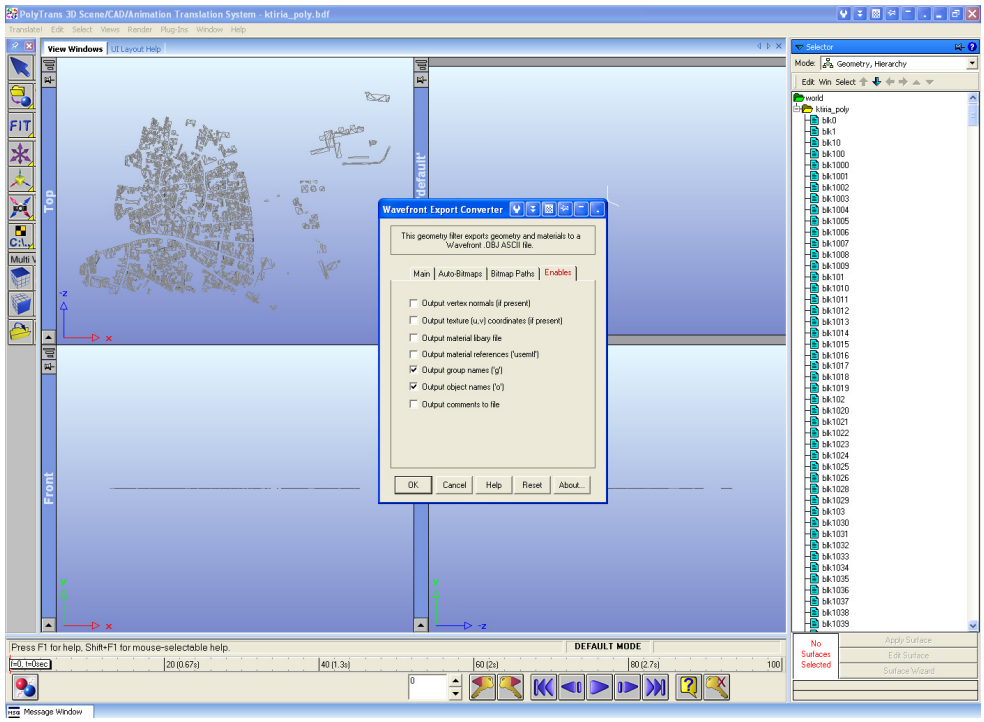


Εικόνα 4.16: Εισαγωγή των .dxf δεδομένων στο λογισμικό Okino Polytrans με τις ανάλογες ρυθμίσεις.

Ακολούθως εξάγαμε τον ηλεκτρονικό χάρτη υπό μορφή obj αρχείου, διαλέγοντας τις ανάλογες ρυθμίσεις έτσι ώστε το κάθε οικόπεδο να αναγνωρίζεται ως μοναδικό. Δηλαδή να μην αναγνωρίζει όλο το χάρτη ως ένα ενιαίο σχήμα. Βλέπουμε περισσότερες λεπτομέρειες όσο αφορά τις ρυθμίσεις για την εξαγωγή των δεδομένων στην *εικόνα 4.17* και στην *εικόνα 4.18*.



Εικόνα 4.17: Εξαγωγή των δεδομένων σε μορφή .obj επιλέγοντας τις ρυθμίσεις Remove duplicated vertex normals και Flip normals of all polygons όσο αφορά το tab, Main

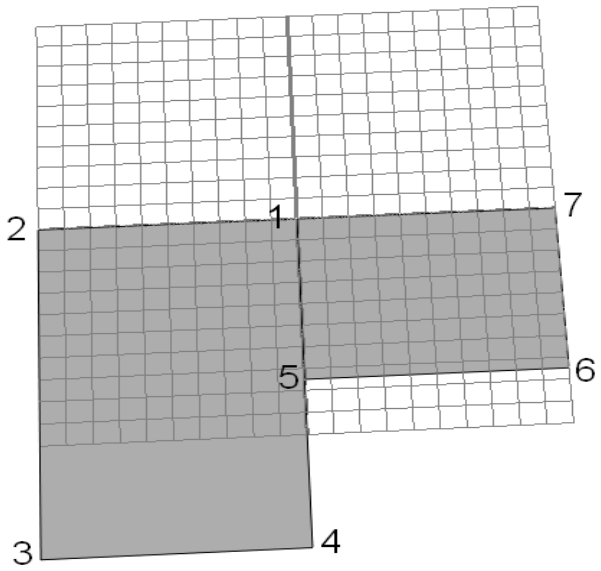


Εικόνα 4.18: Εξαγωγή των δεδομένων σε .obj μορφή επιλέγοντας τις ρυθμίσεις Output group name και output object names όσο αφορά το tab, Enables.

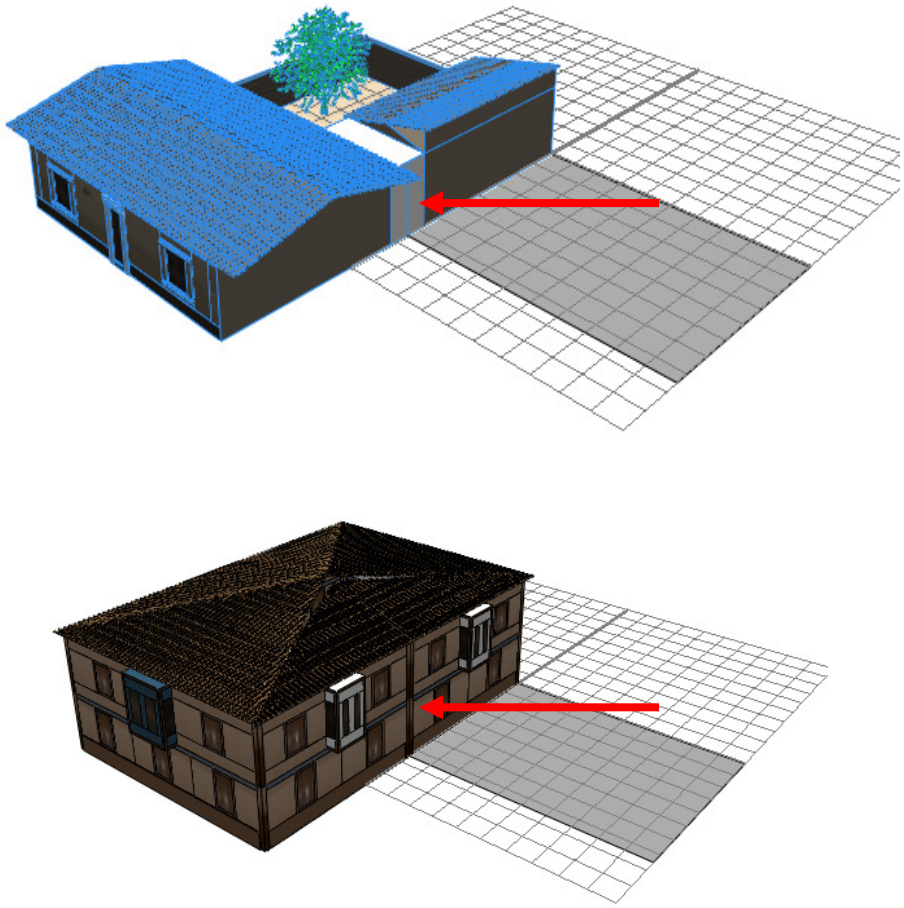
Ο τελικός στόχος ήταν να μετατραπούν τα δεδομένα σε obj μορφή, όπου θα μπορούσαν να εισαχθούν στο λογισμικό CityEngine.

Για να πάρουμε τον μερικό χάρτη, ακολουθήθηκε η ίδια διαδικασία. Το κομμάτι του χάρτη επιλέχθηκε από το λογισμικό MapInfo και όταν έγιναν export σε dfx μορφή, τροποποιήθηκαν από το λογισμικό 3D Studio Max για να διαγραφούν κάποια περιττά σχήματα.

Ένα άλλο πρόβλημα που είχαμε ήταν ότι στο χάρτη της Λευκωσίας το σχήμα του κάθε σπιτιού τις περισσότερες φορές συμπιπτε σε κάποια σημεία με κάποιο άλλο σχήμα. Οπόταν αν για παράδειγμα κάποιο σπίτι είχε 4 σημεία και σύμπιπτε με κάποιο άλλο σχήμα, τότε όταν γινόταν export ο χάρτης λάμβανε υπόψην του αντι τα 4 σημεία για το συγκεκριμένο σχήμα, 5 σημεία (*εικόνα 4.19*). Αυτό δημιουργούσε πρόβλημα στο λογισμικό CityEngine διότι διαχώριζε κάποιο face του σπιτιού σε 2 μέρη και εφαρμοζόταν ο ίδιος κανόνας 2 φορές (*εικόνα 4.20*).

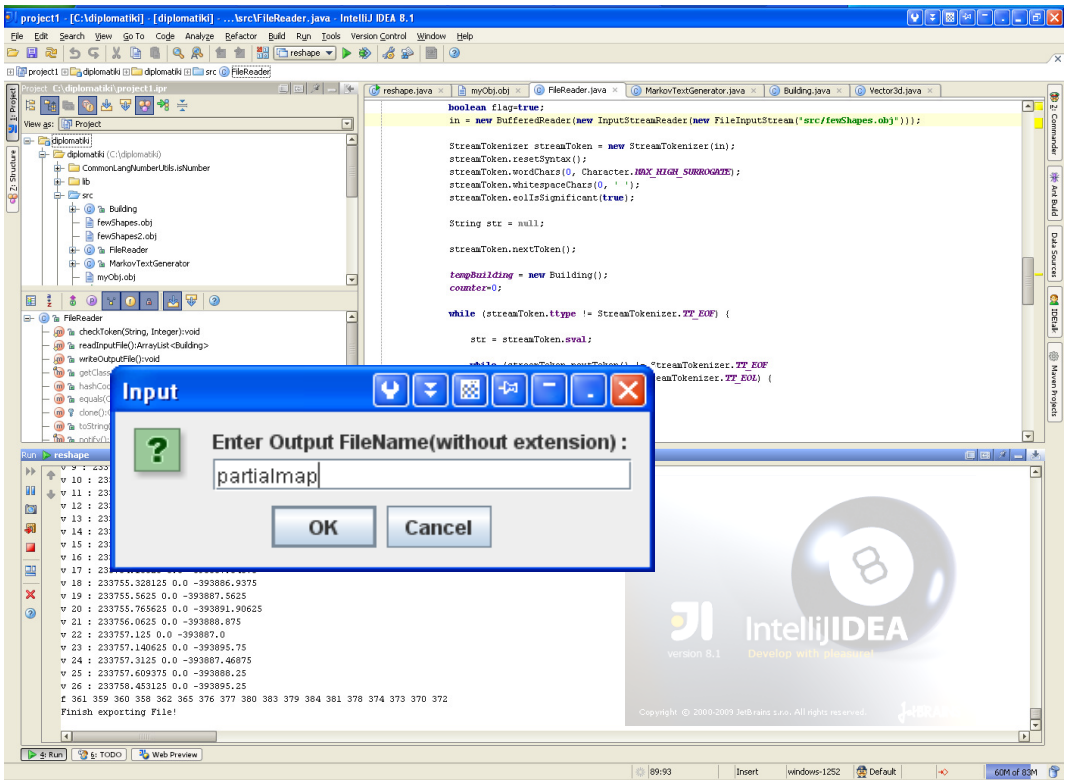


Εικόνα 4.19: Γραφική αναπαράσταση των σχημάτων όπως φαίνονται στο λογισμικό CityEngine



Εικόνα 4.20: Προβλήματα τα οποία παρουσιάζονταν στη δημιουργία των μοντέλων στο λογισμικό CityEngine αφού εφαρμοζόταν ο ίδιος κανόνας δύο φορές σε μια όψη.

Το πρόβλημα αυτό διορθώθηκε με τροποποίηση των τελικών δεδομένων του χάρτη. Δηλαδή αυτών που ήταν σε obj μορφή. Γράφτηκε ένα πρόγραμμα στη γλώσσα προγραμματισμού Java, χρησιμοποιώντας το IDE: IntellijIDEA (*εικόνα 4.21*). Το πρόγραμμα αυτό έλεγχε 3 σημεία κάθε φορά, υπολογίζοντας τη γωνιά που σχηματίζουν. Αν αυτή ξεπερνούσε κάποιο διάστημα (που ορίσαμε εμεις) τότε το μεσαίο σημείο διαγράφετουν. Στόχος μας δηλαδή ήταν να διαγράψουμε τα περιττά σημεία τα οποία βρίσκονταν στη μέση ενός διανύσματος όπου η γωνιά που σχημάτιζαν ήταν σχεδόν 180 μοίρες (σχεδόν ευθεία γραμμή). Ο κώδικας βρίσκεται στο παράρτημα Α.



Εικόνα 4.21: Πρόγραμμα το οποίο γράφτηκε στην γλώσσα προγραμματισμού Java στο IDE, IntellijIDEA, για αφαίρεση των περιττών σημείων που είχε το κάθε σχήμα.

4.4 Μεθοδολογία I

Για την διπλωματική μου εργασία χρησιμοποιήσαμε δύο σχετικά διαφορετικές μεθοδολογίες. Στη πρώτη όπου περιγράφουμε περισσότερο πιο κάτω, χρησιμοποιούμε τους ηλεκτρονικούς μας χάρτες ως δεδομένα εισόδου και προσπαθούμε να φτιάξουμε αρχικά ένα σπίτι της παλιάς Λευκωσίας και στη συνέχεια να εφαρμόσουμε τους κανόνες μας σε ένα μέρος του χάρτη μας. Στο τέλος θα εφαρμόσουμε τους κανόνες σε ολόκληρο το χάρτη.

4.4.1 Δημιουργία ενός σπιτιού

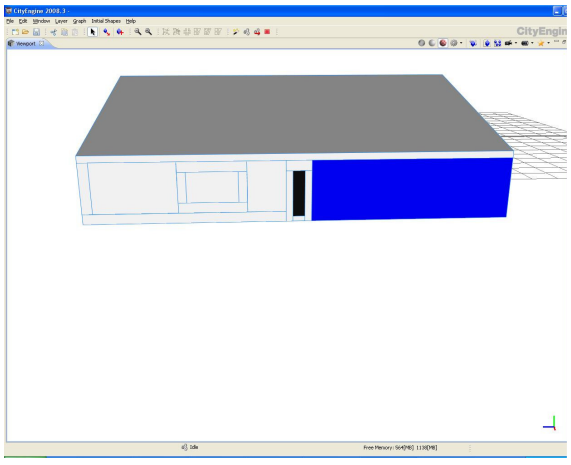
Όπως έχω αναφέρει αρχικός στόχος μας, αφού είχα μάθει να χρησιμοποιώ τους κανόνες του λογισμικού, ήταν η σχεδίαση ενός σπιτιού της παλιάς Λευκωσίας. Τις φωτογραφίες των σπιτιών τις πήρα από τον επιβλέπων καθηγητή μου, κ. Γιώργο Χρυσάνθου. Έχω διαχωρίσει την κατασκευή του σπιτιού σε διάφορα στάδια έτσι ώστε να φαίνεται η πρόοδος που είχαμε κατά διαστήματα.

Να αναφέρω ότι στη δημιουργία του ενός σπιτιού, τα σπίτια κτίζονταν σε σχήματα που είχα ορίζει εγώ. Δηλαδή δημιούργησα απλά τετράγωνα σχήματα (ως αρχή) σε μορφή obj.

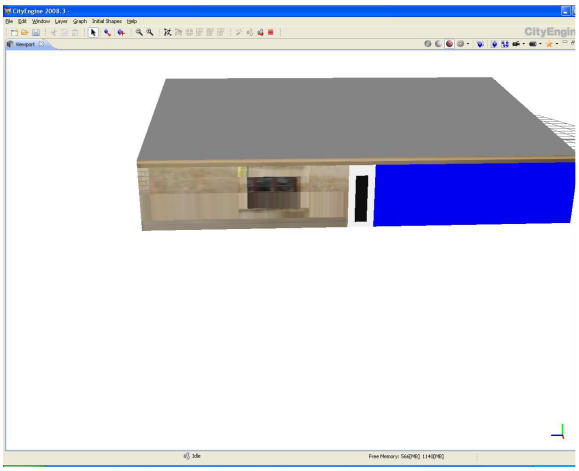
Στάδιο 1

Ως πρώτο στάδιο, μελέτησα την πρόσοψη του σπιτιού (αφού μόνο τις προσόψεις είχαμε) και δημιούργησα μια αρχική όμοια πρόσοψη, χρησιμοποιώντας τον κανόνα διαχωρισμού και εφαρμόζοντας κάποια textures. Τα αποτελέσματα τα βλέπουμε στην *εικόνα 4.23* και στην *εικόνα 4.24*. Στην *εικόνα 4.22* βλέπουμε την φωτογραφία του σπιτιού που μοντελοποιούμε.

Εικόνα 4.22: Αρχική Φωτογραφία Σπιτιού



Εικόνα 4.23: Παραγώμενο μοντέλο απο το λογισμικό CityEngine σε αρχικό στάδιο διαχωρισμένο όπως φαίνεται στην Αρχική Φωτογραφία σπιτιού

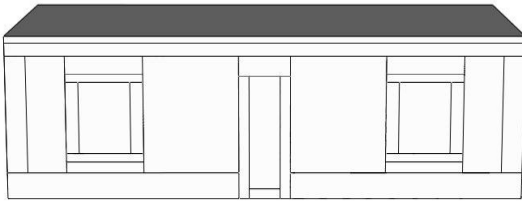


Εικόνα 4.24: Παραγόμενο μοντέλο απο το λογισμικό CityEngine σε αρχικό στάδιο εφαρμόζοντας κάποια textures.

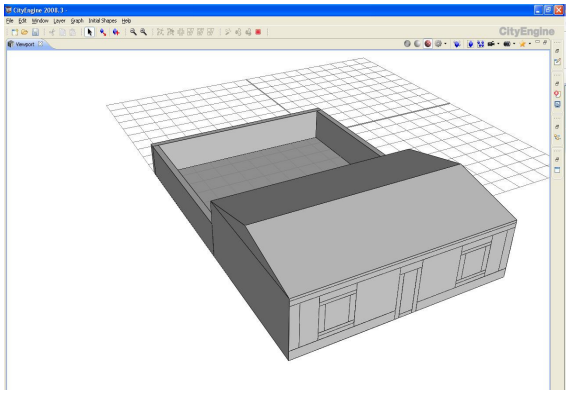
Τα textures πάρθηκαν από τη φωτογραφία, για αυτό και δεν είναι τόσο ρεαλιστικά αφού στη φωτογραφία το κτίριο έχει φωτογραφηθεί με κάποια κλίση. Επίσης ο τρόπος με τον οποίο εισάγαμε τα textures δεν ήταν ο πιο σωστός για αυτό είχαμε τα πιο κάτω αποτελέσματα

Στάδιο 2

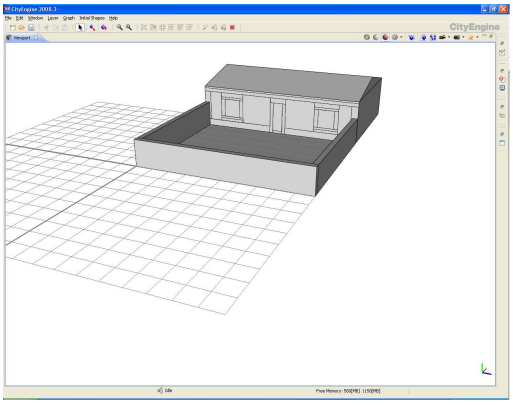
Εικόνα 4.25: Αρχική Φωτογραφία Σπιτιού



Εικόνα 4.26: Παραγόμενο μοντέλο απο το λογισμικό CityEngine διαχωρισμένο στις διάφορες λεπτομέρειες όπως φαίνεται στην Αρχική Φωτογραφία Σπιτιού



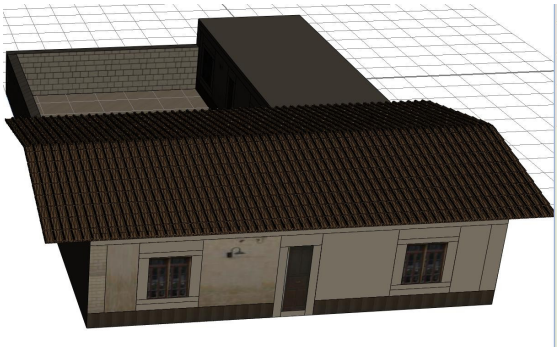
Εικόνα 4.27: Παραγόμενο μοντέλο απο το λογισμικό CityEngine διαχωρισμένο στις διάφορες λεπτομέρειες από διαφορετική κάμερα



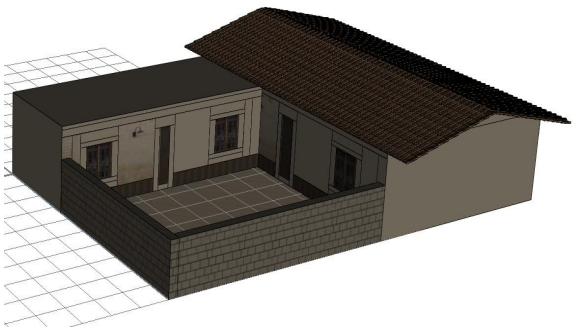
Εικόνα 4.28: Παραγόμενο μοντέλο απο το λογισμικό CityEngine διαχωρισμένο στις διάφορες λεπτομέρειες από διαφορετική κάμερα

Αφού δημιούργησα την πρόσοψη, προχώρησα στην δημιουργία της στέγης, της αυλής και του φράκτη. Με βάση όμως πληροφορίες από τον κ. Χρυσάνθου η στέγη έπρεπε να εξέχει κάποια εκατοστά και να μην είναι ένα με το κτίριο. Τα αποτελέσματα τα βλέπουμε στις *εικόνες 4.26,4.27* και *4.28*.

Στάδιο 3



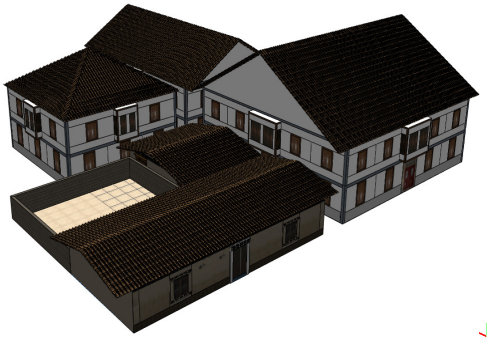
Εικόνα 4.29: Πρόσοψη του μοντέλου, παραγόμενο απο το λογισμικό CityEngine προσθέτοντας textures και μοντέλα



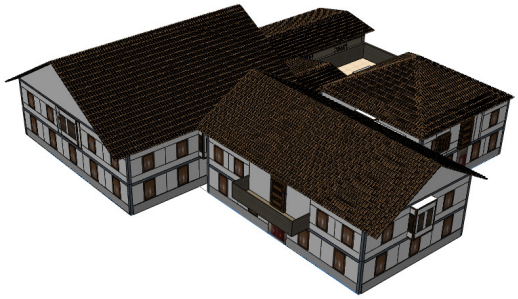
Εικόνα 4.30: Πίσω όψη του μοντέλου, παραγόμενο απο το λογισμικό CityEngine προσθέτοντας textures και μοντέλα.

Όπως έχω αναφέρει σε προηγούμενο κεφάλαιο, συνήθως τα σπίτια είχαν το σχήμα του αγγλικού γράμματος L. Έτσι χρησιμοποιώντας τη φαντασία μας, επεκτείναμε το αρχικό σπίτι, έτσι ώστε να σχηματίζει το γράμμα L. Εφαρμόσαμε κάποια textures τόσο στο σπίτι όσο και στο φράκτη. Επίσης διορθώσαμε την στέγη έτσι ώστε να εξέχει κάποια εκατοστά από το υπόλοιπο κτίριο. Τα αποτελέσματα αυτού του σταδίου τα βλέπουμε στις *εικόνες 4.29* και *4.30*.

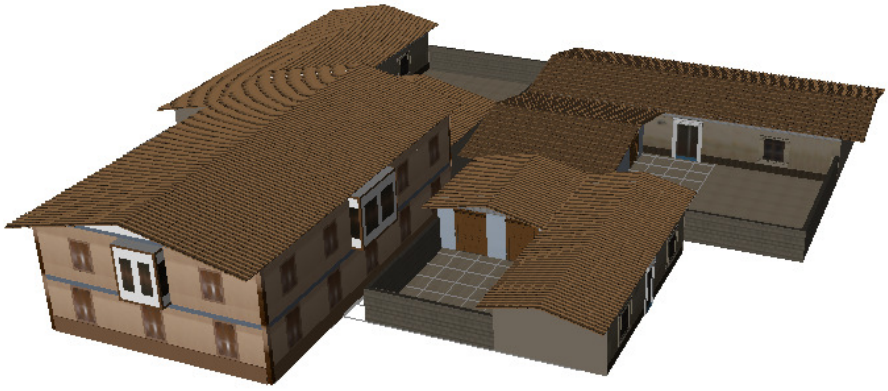
Στάδιο 4



Εικόνα 4.31: Παραγωγή των μοντέλων απο το λογισμικό CityEngine



Εικόνα 4.32: Παραγωγή μοντέλων σπιτιών τύπου Δίπατο από το λογισμικό CityEngine.

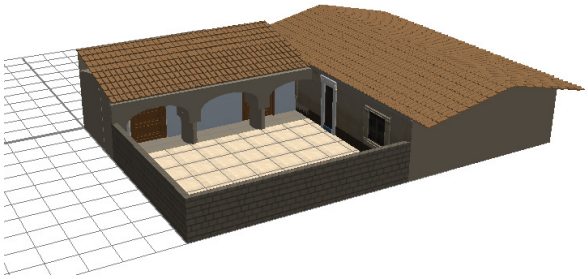


Εικόνα 4.33: Μοντέλα τα οποία παράχθηκαν απο το λογισμικό CityEngine

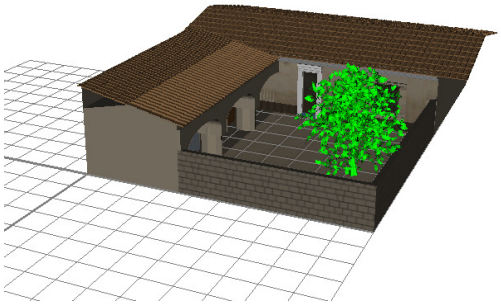
Στη συνέχεια προχωρήσαμε στη δημιουργία και ενός δεύτερου τύπου σπιτιού, το δίπατο. Λόγω έλλειψη πληροφοριών για το που βρίσκεται το σπίτι και που η αυλή στον ηλεκτρονικό χάρτη, για το δίπατο σπίτι δεν βάλαμε αυλή. Στις *εικόνες 4.31* και *4.32* οι τοίχοι των δίπατων σπιτιών δεν έχουν textures σε αντίθεση με την *εικόνα 4.33*.

Επίσης όσο αφορά τα δίπατα σπίτια εφάρμοσα κάποια δυναμικά μοντέλα. Δηλαδή ανάλογα με τυχαίους αριθμούς οι οποίοι παράγονταν σε χρόνο εκτέλεσης, δημιουργούνταν στο δεύτερο όροφο είτε κιόσκια είτε μπαλκόνια. Δυναμική επίσης ήταν και ο τύπος της στέγης. Για παράδειγμα στην πρώτη εικόνα παρατηρούμε (όσο αφορά τα δίπατα σπίτια) δύο διαφορετικά είδη οροφών. Στόχος ήταν να δώσουμε κάποια διαφορετικότητα στην μαζική δημιουργία μοντέλων μας.

Στάδιο 5



Εικόνα 4.34: Ολοκληρομένο μοντέλο σπιτιού τύπου όπως το αγγλικό γράμμα L παραγόμενο απο το λογισμικό CityEngine.



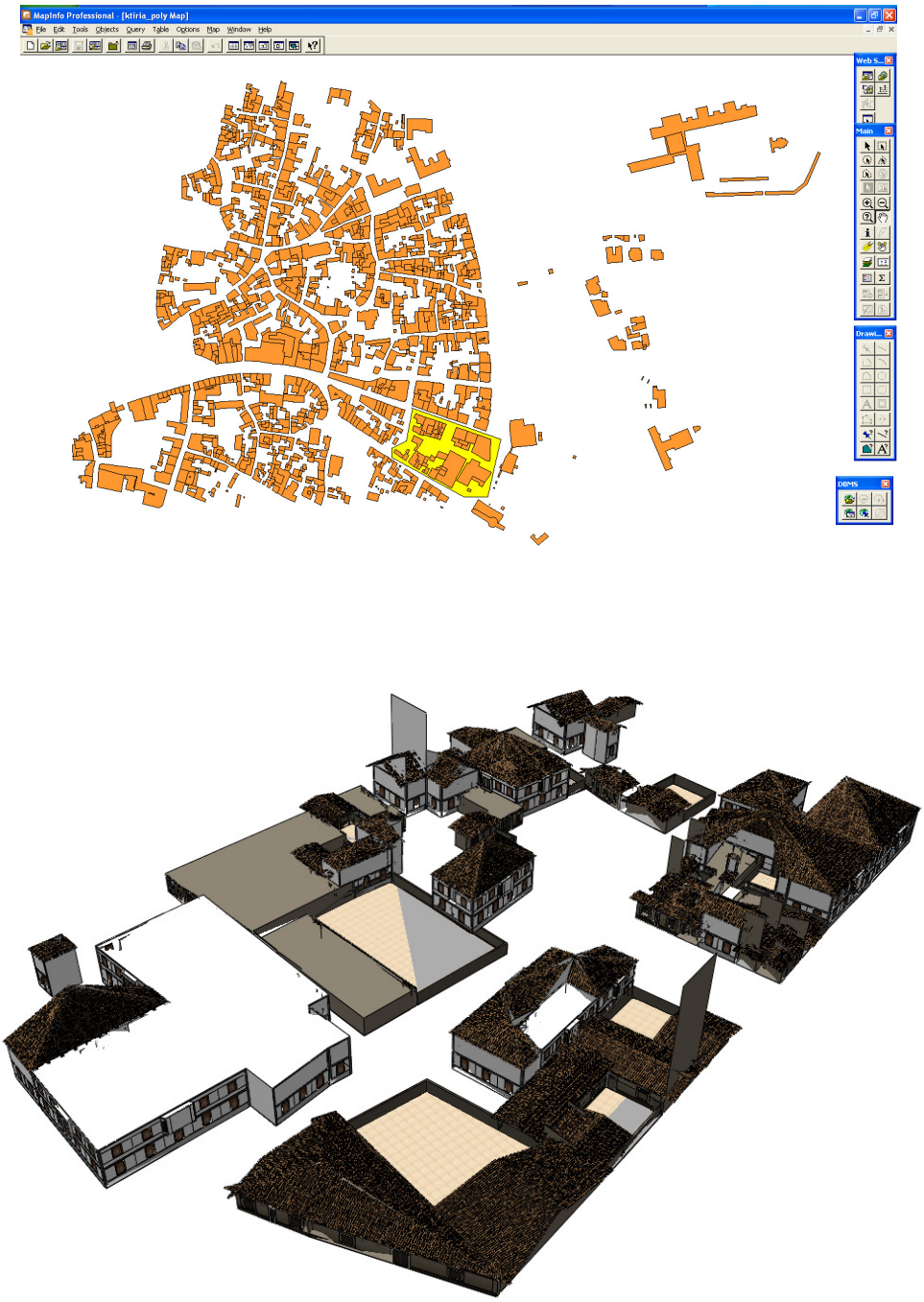
Εικόνα 4.35: Ολοκληρομένο μοντέλο σπιτιού.

Σε αυτό το στάδιο, προσθέσαμε ένα μοντέλων καμάρων στο πίσω σπίτι (*εικόνα 4.34* και *εικόνα 4.35*). Επίσης η στέγη του πίσω σπιτιού ήταν μονή. Αυτό ήταν πρόβλημα, διότι το CityEngine προσφέρει τρία είδη οροφών, όπου η μονή οροφή δεν περιέχεται σε αυτές. Έτσι χρησιμοποίησα τους κανόνες της οροφής όπως φαίνεται στο μπροστά σπίτι και το τροποποίησα όσο γινόταν δυνατό, έτσι ώστε να δημιουργήσω την μονή οροφή. Δηλαδή διαχώρισα στη μέση την οροφή και εμφάνιζα μόνο την μια πλευρά της οροφής. Στη συνέχεια επέκτεινα το μέγεθος της οροφής με την εντολή size του λογισμικού CityEngine, έτσι ώστε να καλύπτει ολόκληρο το σπίτι.

Πρόσθεσα και κάποια μοντέλα δέντρων στην αυλή του σπιτιού, έτσι ώστε να γίνουν πιο ρεαλιστικά τα μοντέλα. Τα μοντέλα των δέντρων παράγονταν και αυτά δυναμικά με βάση την τυχαιομηχανή σε χρόνο εκτέλεσης. Ανάλογα με τον αριθμό που παρήγαγε η τυχαιομηχανή δημιουργούσε ένα δέντρο, πολλά μικρά δεντράκια ή καθόλου δέντρα.

4.4.2 Δημιουργία σπιτιών σε μερικό χάρτη

Με βάση τους κανόνες που είχα γράψει μέχρι στιγμής, τους χρησιμοποίησα στο μερικό χάρτη, όπως φαίνεται στην *εικόνα 4.36*.



Εικόνα 4.36: Στην πάνω εικόνα φαίνεται το κομμάτι του χάρτι στον οποίο εφαρμόστηκαν οι κανόνες και στην κάτω εικόνα φαίνονται τα αποτελέσματα.

Τα αποτελέσματα όμως δεν ήταν ενθαρρυντικά, λόγω των παράξενων σχημάτων του χάρτη. Στο συγκεκριμένο ηλεκτρονικό χάρτη δεν γνωρίζαμε ποια σχήματα είχαν όντως σπίτια, ποια σπίτια είχαν αυλή και αν είχαν αυλή, που βρισκόταν.

4.4.3 Δημιουργία σπιτιών σε ολόκληρο το χάρτη

Τελικός μας στόχος ήταν να εφαρμόσουμε τους κανόνες που είχα είδη γράψει, σε ολόκληρο τον χάρτη. Όταν το προσπάθησα το λογισμικό, συγκεκριμένα η γραμματική έβγαζε κάποιο error λόγω έλλειψης μνήμης. Αυτό δεν ήταν πρόβλημα αφού θα εφαρμόζαμε τους κανόνες σε μέρη του χάρτη κάθε φορά έτσι ώστε να δημιουργήσαμε και να εξάγουμε τα μοντέλα μας.

Δυστυχώς όμως όπως έχω αναφέρει πιο πάνω η μαζική δημιουργία σπιτιών στον ηλεκτρονικό χάρτη, ακόμα και μετά την τροποποίηση του, δεν έβγαζε τα επιθυμητά αποτελέσματα. Πιθανά προβλήματα ήταν το τοπικό σύστημα συντεταγμένων για κάθε σχήμα, έτσι οι καθολικοί κανόνες δεν μπορούσαν να εφαρμοστούν. Να αναφέρω επίσης ότι στο άρθρο «Procedural Modeling of Buildings» οι συγγραφείς σημειώνουν ότι η μαζική δημιουργία σπιτιών χρησιμοποιώντας ηλεκτρονικούς χάρτες δεν έχει τα αναμενόμενα αποτελέσματα και τείνουν να το στηρίξουν περισσότερο αυτό το θέμα στο μέλλον. Έτσι αναγκαστήκαμε να προχωρήσουμε σε μια λίγο διαφορετική μεθοδολογία.

4.4.4 Προβλήματα Υλοποίησης

Τα προβλήματα της υλοποίησης της συγκεκριμένης μεθοδολογίας ήταν αρκετά. Το κυριότερο πρόβλημα ήταν ο χάρτης. Αφιερώσαμε μεγάλο χρονικό διάστημα στην τροποποίηση του χάρτη στην μορφή obj, αφού κανένα από τα γνωστά λογισμικά μοντελοποίησης δεν τροποποιούσε το χάρτη στην επιθυμητή μορφή χωρίς να ενοποιήσει το όλο σχήμα. Αφού τα κατάφερα να τροποποιήσω το χάρτη, ανακαλύψαμε στο λογισμικό CityEngine, ότι δεν μπορούσαμε να χρησιμοποιήσουμε τους καθολικούς και γενικούς μας κανόνες λόγω:

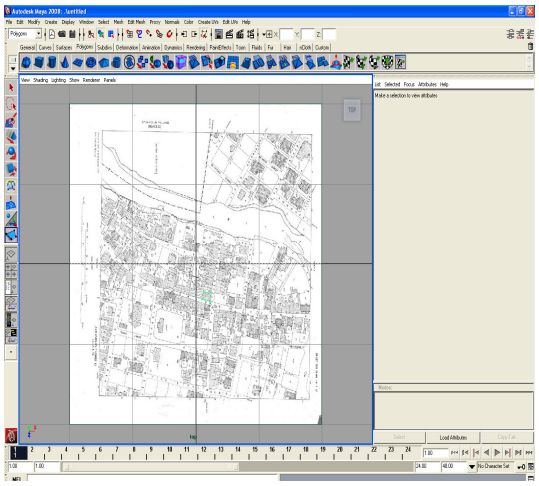
1. της διαφορετικότητας των σχημάτων. Υπήρχαν αρκετά πολύ παράξενα σχήματα για τα οποία δεν γνωρίζαμε καν αν υπήρχε σπίτι εκεί.
2. της έλλειψης πληροφοριών όσο αφορά την τοπολογία και τυπολογία των σπιτιών. Δηλαδή για κάθε σχήμα δεν γνωρίζαμε που βρισκόταν το σπίτι, ποια

ήταν η τυπολογία του σπιτιού, ποιο μέρος του σχήματος ήταν η αυλή και αν είχε πίσω σπίτι(δώματα).

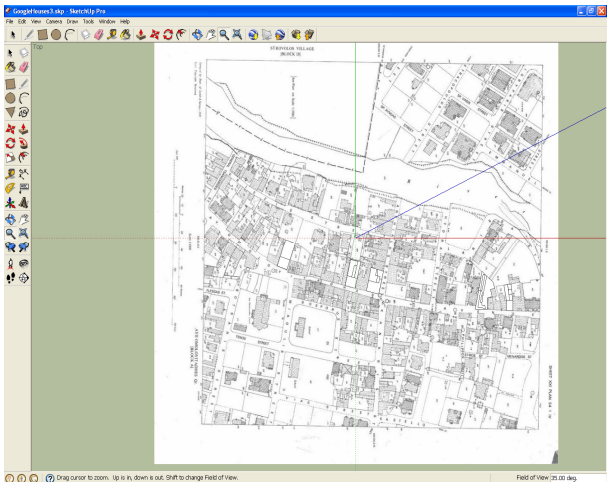
3. του τοπικού συστήματος συντεταγμένων για το κάθε σχήμα.

4.5 Μεθοδολογία ΙΙ

Σε αυτή την μεθοδολογία χρησιμοποιήσαμε κανονικούς χάρτες, τους οποίους σαρώσαμε από ένα σαρωτή(scanner). Ακολουθώντας το φροντιστήριο 5.1 για την δημιουργία και εισαγωγή χειροποίητων σχημάτων, χρησιμποιώντας το χάρτη, δημιούργησα τα δικά μας χειροποίητα σχήματα. Δημιουργήθηκαν αρχικά μέσω του λογισμικού μοντελοποίησης Autodesk Maya, βάζοντας το χάρτη ως background και χρησιμοποιώντας το εργαλείο του Maya, Create Polygon Tool. Αργότερα χρησιμοποιήθηκε το λογισμικό Google Sketchup.



Εικόνα 4.37: Διαπροσωπεία του λογισμικού Maya.

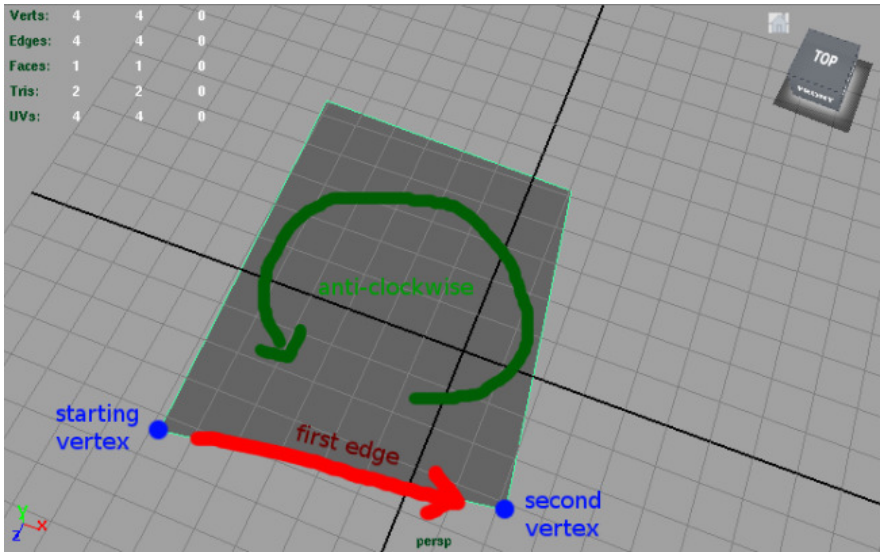


Εικόνα 4.38: Διαπροσωπεία του λογισμικού Google Sketchup.

4.5.1 Δημιουργία των χειροποίητων πολυγώνων

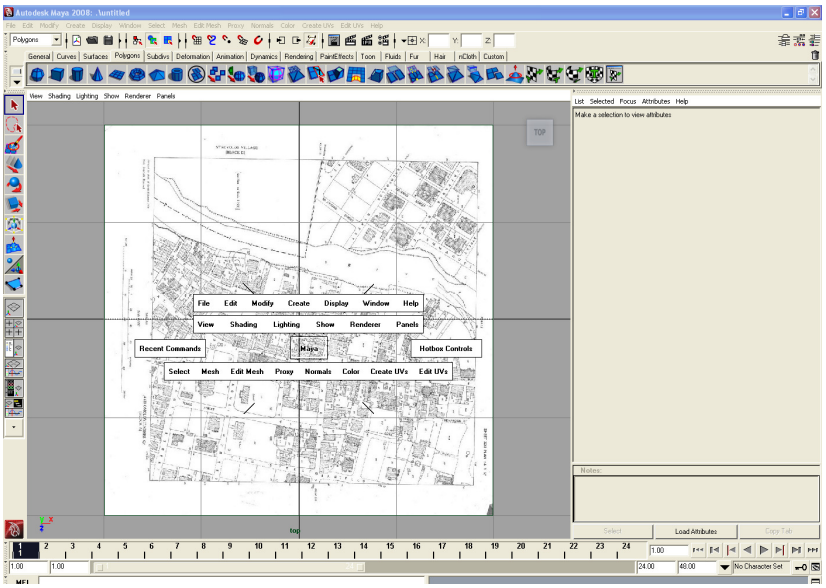
Το CityEngine υποστηρίζει εισαγωγής δεδομένων της μορφής obj. Οπόταν δημιουργήσαμε τα δικά μας πολύγωνα μέσω του λογισμικού Maya, χρησιμοποιώντας το εργαλείο Create Polygon tool. Η δημιουργία των πολυγώνων γινόταν αντίθετα με τη φορά του ρολογιού έτσι ώστε η κάθε όψη του πολυγώνου να βλέπει προς τα έξω (για να

είναι visible το face του πολυγώνου). Επίσης τα δύο πρώτα σημεία που δημιουργούσα αντιπροσωπεύουν την μπροστινή όψη (front face) του πολυγώνου (*εικόνα 4.39*).



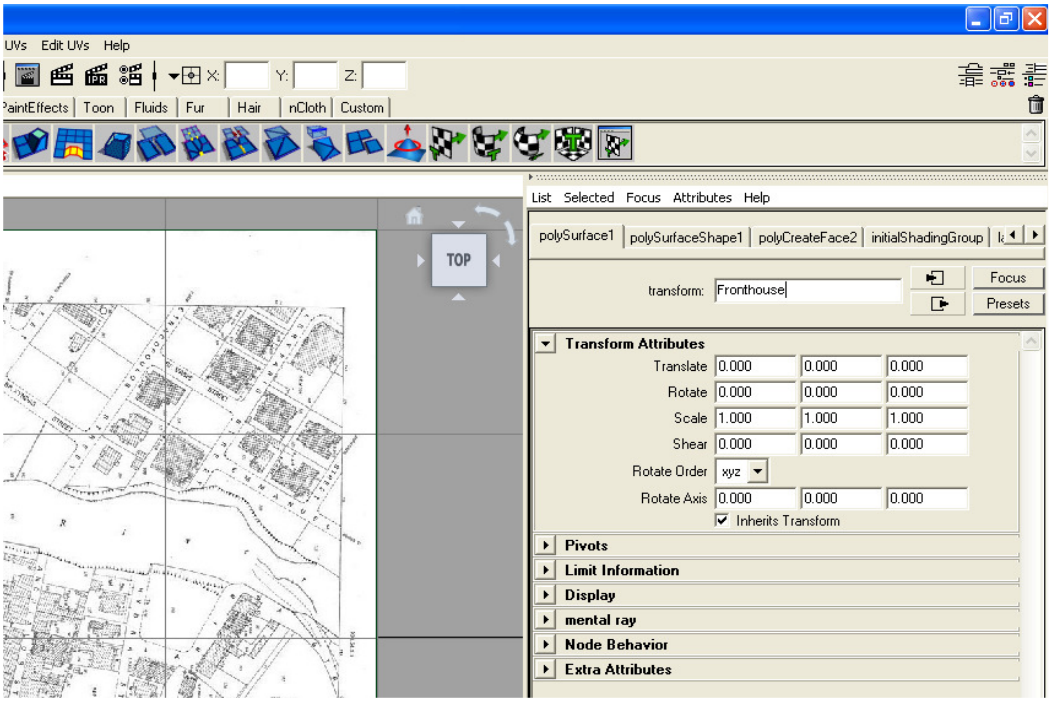
Εικόνα 4.39: Δημιουργία σχημάτων στο Maya χρησιμοποιώντας το εργαλείο Create Polygon Tool.

Εισήγαγα το χάρτη ως background στο λογισμικό Maya (*εικόνα 4.40*) και ακολούθως δημιούργησα τα πολύγωνα επιλέγοντας από το Menu του Maya το Mesh→Create Polygon Tool



Εικόνα 4.40: Δημιουργία των περιγραμμάτων των σπιτιών της παλιάς Λευκωσίας μέσω του λογισμικού Maya έχοντας ως background τον χάρτη της Λευκωσίας.

Με αυτό το εργαλείο σημείωνα τα vertices των σχημάτων των κτιρίων σύμφωνα με τον χάρτη και στο τέλος πατούσα Enter για να το δημιουργήσει. Στη συνέχεια εμφανιζόταν στα δεξιά μια ταπέλα με τα αποτελέσματα του σχήματος που δημιούργησα. Για να αλλάξω το όνομα του σχήματος έτσι ώστε να είναι ίδιο με τον αρχικό κανόνα του CityEngine, επέλεγα την καρτέλα polySurface# (#: ένας αριθμός) και ακολούθως έγραφα το όνομα του σχήματος όπως φαίνεται στην *εικόνα 4.41*.



Εικόνα 4.41: Ανάθεση συμβολικού ονόματος στο σχήμα που δημιουργήθηκε

Όπως έχω αναφέρει προηγουμένως στη συνέχεια χρησιμοποίησα το λογισμικό Google Sketchup, αντί του Maya. Με το Maya όταν έκανα zoom διαμορφωνόταν ανάλογα η κλίμακα των αξόνων, και τα σχήματα έβγαιναν άλλοτε μικρά (όταν το zoom ήταν πολύ μικρό) και άλλοτε πολύ μεγάλα (όταν το zoom ήταν αρκετά μεγάλο). Τα σχήματα έβγαιναν κανονικά όταν έβλεπα όλο το χάρτη στο viewport. Δυστυχώς όμως δεν μπορούσα να δω με ακρίβεια τις διάφορες λεπτομέρειες. Με το Google Sketchup πολύ απλά έκανα import το χάρτη, όρισα το μέγεθος του και στη συνέχεια τον τοποθέτησα στο κέντρο των αξόνων για να ταιριάζουν οι συντεταγμένες τον σχημάτων με το χάρτη ο οποίος ήταν είδη εισαγμένος στο CityEngine. Ακολούθως δημιούργησα τα σχήματα με το εργαλείο Line. Το θετικό του Google Sketchup είναι ότι μπορούσα να κάνω zoom τη κάμερα, χωρίς να αλλάζει η κλίμακα των αξόνων. Δυστυχώς το Google Sketchup χρησιμοποιήθηκε προς το τέλος του δεύτερου εξαμήνου της διπλωματικής εργασίας.

4.5.2 Κανόνες σπιτιών

Στη συγκεκριμένη φάση κανονίσαμε ραντεβού με την αρχιτέκτονα Αθηνά Παπαδοπούλου και την ομάδα της στο αρχιτεκτονικό γραφείο Nicosia Master Plan για να πάρουμε περισσότερες πληροφορίες όσο αφορά τα σπίτια της παλιάς Λευκωσίας.

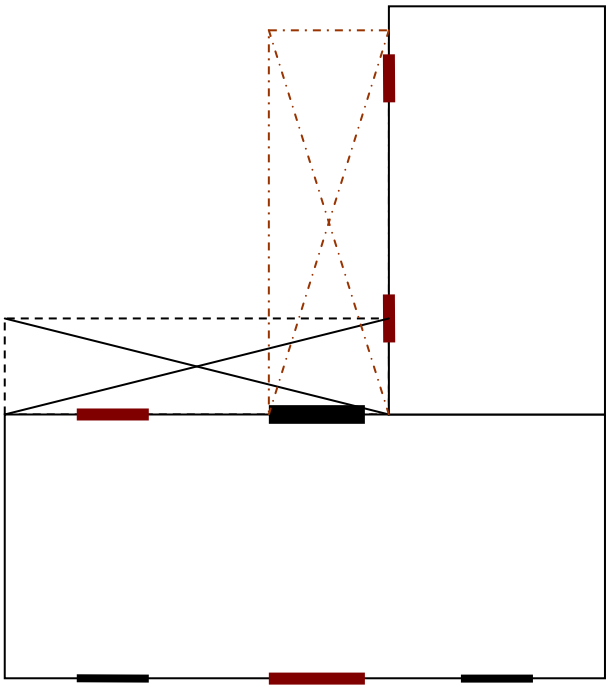
Από την συνέντευξη μας ξεκαθαρίστηκαν αρκετά πράγματα. Ένα κανονικό σπίτι έχει μέγιστο μήκος 15 μέτρα και έχει σχήμα όπως το αγγλικό γράμμα L.Το ύψος του σπιτιού μπροστά ήταν 4 μέτρα. Στη περίπτωση του διώροφου, ήταν 8 μέτρα. Το πίσω σπίτι ήταν συνήθως πιο μικρό από το υπόλοιπο σπίτι. Χρησιμοποιούταν είτε ως βοηθητικός χώρος (κουζίνα, επιπλέον δωμάτιο, κ.τ.λ.) είτε ως χώρος για ζώα. Τα πίσω σπίτια ονομάζονταν, δώματα.

Η στέγη των σπιτιών έχει κλίση 14 μοίρες. Επίσης η στέγη μπορεί να εξέχει του κανονικού σπιτιού μεταξύ 25 και 60 εκατοστών. Στο πίσω σπίτι είτε η στέγη ήταν μονή, είτε δεν είχε καθόλου στέγη.

Το μπροστά σπίτι διαχωρίζεται σε ίσα 3 μέρη. Τα αριστερά και δεξιά μέρη ήταν δωμάτια, ενώ το μεσαίο ήταν ο ηλιακός. Στην έξοδο του ηλιακού υπήρχαν καμάρες. Σε αρκετά σπίτια το μήκος του ηλιακού ήταν πιο μικρό σε σχέση με τα άλλα δύο δωμάτια. Η κυρίως πόρτα έχει μήκος περίπου 1 μέτρο και ύψος περίπου 2.5 με 3 μέτρα. Σε ορισμένα σπίτια υπήρχε και μια πίσω πόρτα η οποία οδηγεί στην αυλή και ήταν πιο μικρή σε σύγκριση με την κυρίως πόρτα. Οι διαστάσεις της πίσω πόρτα είναι 1.10 μέτρα Χ 1.90 μέτρα.

Κάθε δωμάτιο έχει ένα παράθυρο. Παλαιότερα, συγκεκριμένα την $1^{η}$ περίοδο (*βλέπε παράγραφο 4.5.3*) τα παράθυρα βρίσκονταν περίπου 2.5 με 3 μέτρα πάνω από το έδαφος και ήταν σχετικά μικρά. Οι διαστάσεις του είναι 80 cm Χ 60 cm. Στην περίπτωση που του σπίτι ήταν γωνιακό και υπήρχε δρόμος και στο μέρος του πίσω σπιτιού τότε υπήρχαν παράθυρα και σε εκείνα τα δωμάτια όπως αυτά που αναφέραμε. Στο πίσω σπίτι πιθανότατα να ήταν ακόμα πιο μικρά. Για την $2^{η}$ περίοδο (*βλέπε παράγραφο 4.5.3*) τα παράθυρα βρίσκονταν στο κέντρο (όσο αφορα τον άξονα y) και οι διαστάσεις τους είναι 1.10 x 1.90 μέτρα.

Τα κιόσκια ή μπαλκόνια υπάρχουν μόνο στην πρόσοψη του σπιτιού, πάνω από την κύρια είσοδο. Το μήκος τους είναι σχεδόν 2 μέτρα. Τα σπίτια της πρώτης περιόδου (όπως εξηγείται πιο κάτω) είχαν κιόσκια στην πρόσοψη και αυτά της δεύτερης περιόδου είχαν μπαλκόνια.



Εικόνα 4.42: Γραφική αναπαράσταση σπιτιού σχήματος L

Σχήμα	Περιγραφή
	Παράθυρο
	Πόρτα
	Καμάρα
	Καμάρες

Πίνακας 4.1: Επεξήγηση των γραφικών σχημάτων της *εικόνας 4.42*.

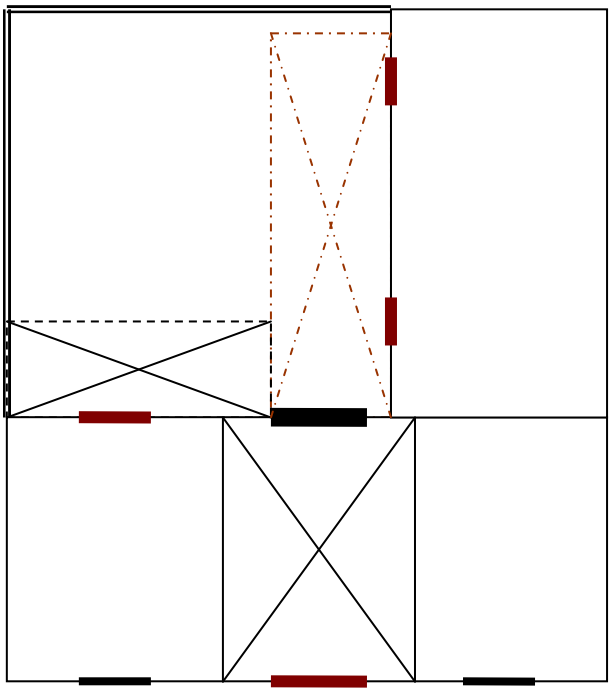
Στην *εικόνα 4.42* βλέπουμε την αναπαράσταση του σπιτιού σχήματος L όπως την περιγράψαμε προηγουμένως. Σε ορισμένα σπίτια υπήρχαν οριζόντιες καμάρες και σε άλλα υπήρχαν κάθετες καμάρες.

4.5.3 Δημιουργία μερικών σπιτιών

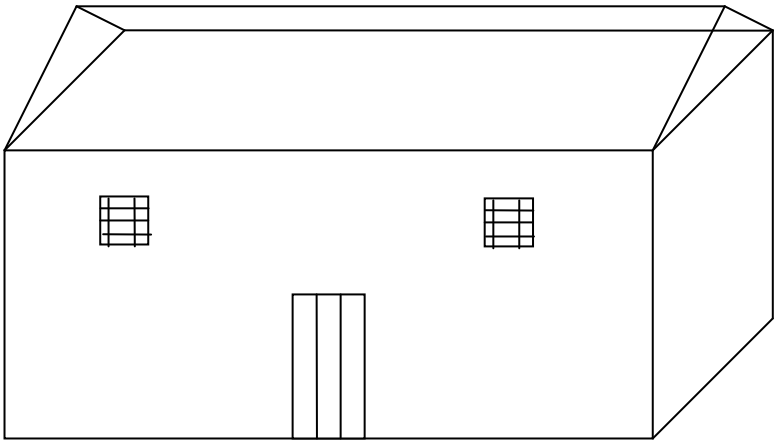
Μετά από συνάντηση με τον κ. Χρυσάνθου, αποφασίσαμε να δημιουργήσουμε ορισμένα σπίτια αφού η διαδικασία δημιουργίας των αποτυπωμάτων των σπιτιών ήταν χειρονακτική. Επομένως ήταν αδύνατο να δημιουργήσουμε όλα τα σχήματα των σπιτιών και να αναπαραστήσουμε ολόκληρη την παλιά Λευκωσία στο χρόνο που μας

έμενε. Ωστόσο τα σχήματα των σπιτιών τα οποία επιλέχθηκαν δημιουργήθηκαν σπίτια 2 περιόδων. Η πρώτη περίοδος αναπαριστά σπίτια από τη Ρωμαϊκή μέχρι Οθωμανική εποχή και η δεύτερη περίοδος από την Οθωμανική μέχρι την Αγγλοκρατία. Χρησιμοποιώντας τις πληροφορίες από το Nicosia Master Plan και τις γνώσεις του κ. Χρυσάνθου καταλήξαμε στα πιο κάτω σχεδιαγράμματα σπιτιών (*εικόνα 4.43* και *εικόνα 4.44*):

Περίοδος 1



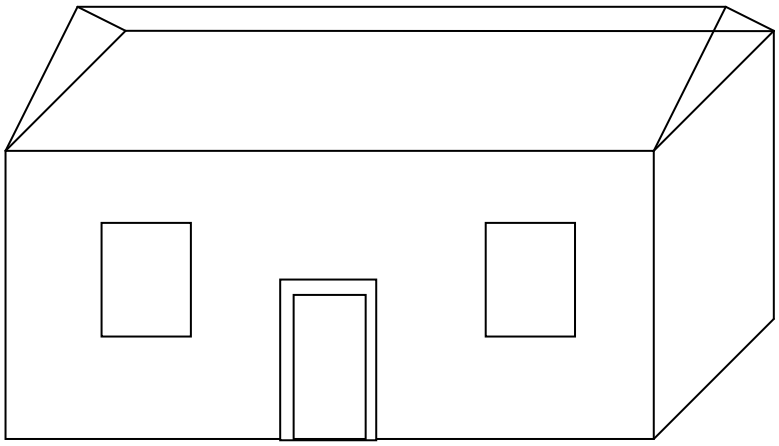
Εικόνα 4.43: Footprints of Courtyard house



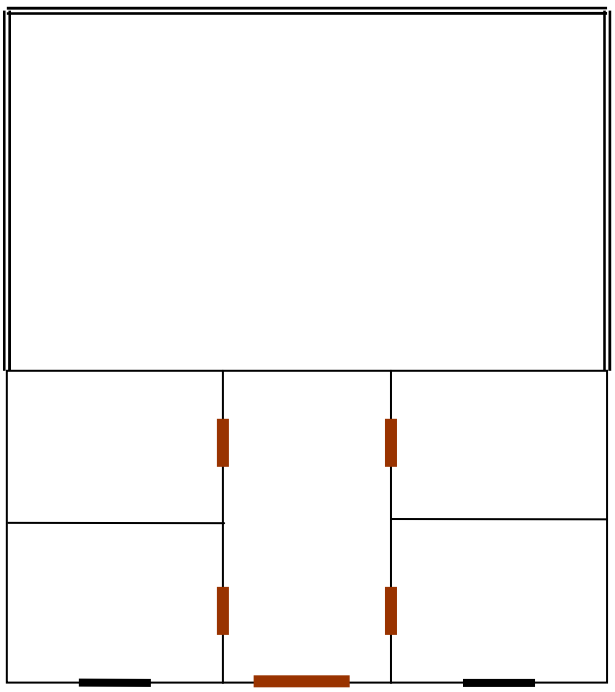
Εικόνα 4.44: Πρόσοψη

Στα πιο πάνω σχεδιαγράμματα παρατηρούμε τα δύο ξεχωριστά δωμάτια και στη μέση τον ηλιακό. Στην *εικόνα 4.43* βλέπουμε το πίσω σπίτι, τις πιθανές καμάρες και την αυλή. Στην *εικόνα 4.44* βλέπουμε την πρόσοψη του σπιτιού. Η πρόσοψη του σπιτιού συνήθως ήταν άσπρη (υπήρχε δηλαδή σουβάς) ή καλυμμένη με πέτρα. Επίσης γύρω από την πόρτα υπήρχε πέτρα.

Περίοδος 2



Εικόνα 4.45: Courtyard house



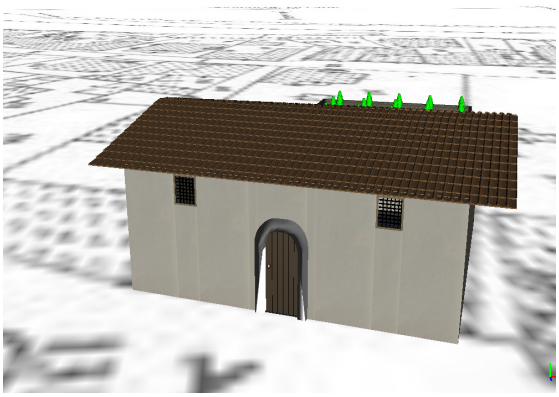
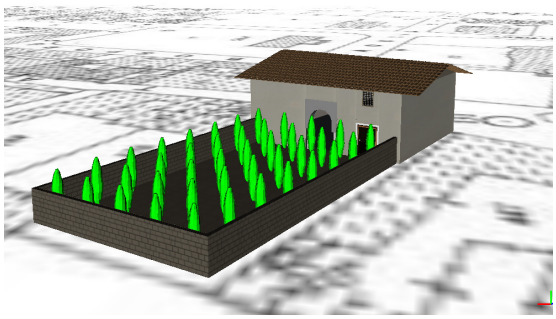
Εικόνα 4.46: Serial house

Στη δεύτερη περίοδο παρατηρούμε ότι τα Courtyard houses (*εικόνα 4.45*) αλλάζουν όσο αφορά την πρόσοψη, τα παράθυρα και η πόρτες. Τα παράθυρα είναι κανονικά (1.9 x 1.1 μέτρα). Επίσης τα πίσω σπίτια είχαν μονή οροφή. Στα Serial houses (*εικόνα 4.46*) παρατηρούμε ότι υπάρχει μια μεγάλη αυλή, ενώ ο ηλιακός είναι κοινός χώρος για τα 4 δωμάτια.

Δημιουργία αρχικών κτιρίων

Χρησιμοποιώντας όλες τις πιο πάνω πληροφορίες γράφτηκαν κανόνες για την γραφική αναπαράσταση ενός αριθμού σπιτιών, για 2 διαφορετικές περιόδους. Πιο κάτω βλέπουμε τα αποτελέσματα.

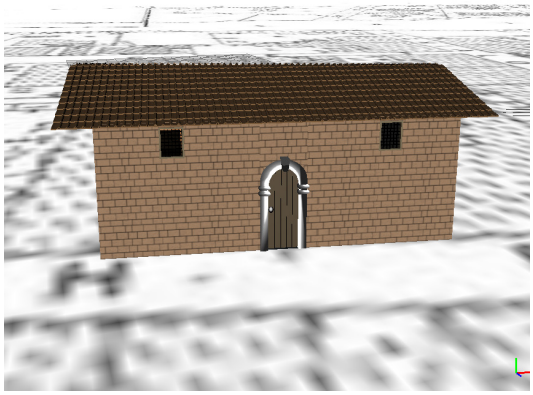
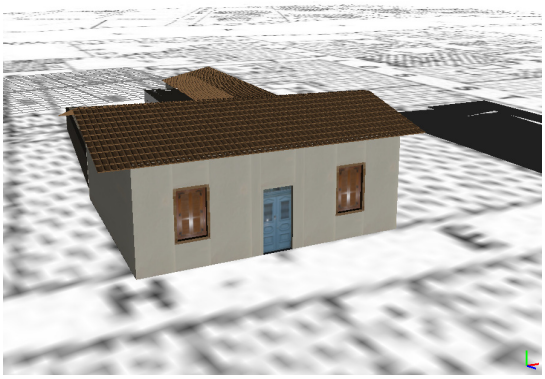
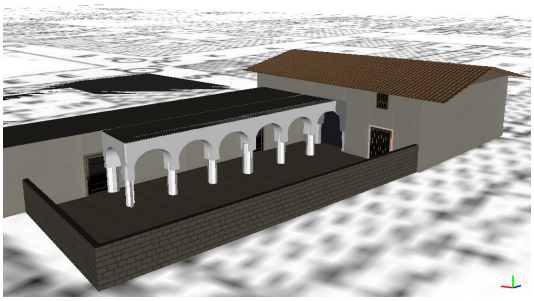
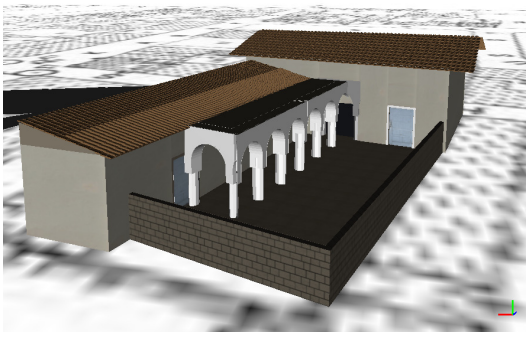
Σπίτι 1

Περίοδος 1	Περίοδος 2
	
	

Πίνακας 4.2: Αρχικά αποτελέσματα μοντέλων του πρώτου σπιτιού, χρησιμοποιώντας την $2^{η}$ μεθοδολογία

Σε απλά σχήματα σπιτιών, δηλαδή αυτά που αποτελούνται από 4 σημεία, η οροφή όπως της έχουμε καθορίσει από τα σχεδιαγράμματα μας, εφαρμόζεται χωρίς κανένα πρόβλημα. Να σημειώσουμε ότι το λογισμικό CityEngine δίνει την δυνατότητα στο χρήστη να εφαρμόσει 3 διαφορετικά είδη οροφών (hip, gable, pyramid). Για απλά σπίτια χρησιμοποίησα την gable και για πολύπλοκα την pyramid. Η δημιουργία των διαφόρων ειδών δέντρων και των διαφόρων παραθύρων (για την περίοδος 2) αποφασίζονται τυχαία, στο χρόνο εκτέλεσης του λογισμικού. Τα αρχικά μοντέλα που παράχθηκαν από το λογισμικό CityEngine τα βλέπουμε στο *πίνακα 4.2*.

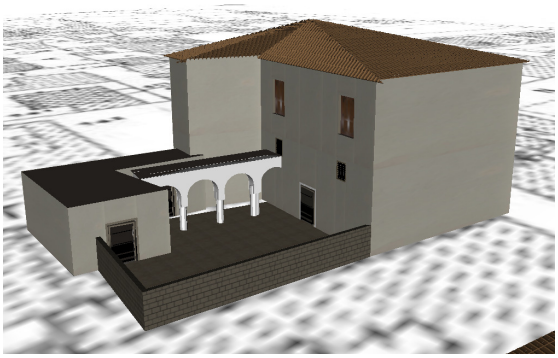
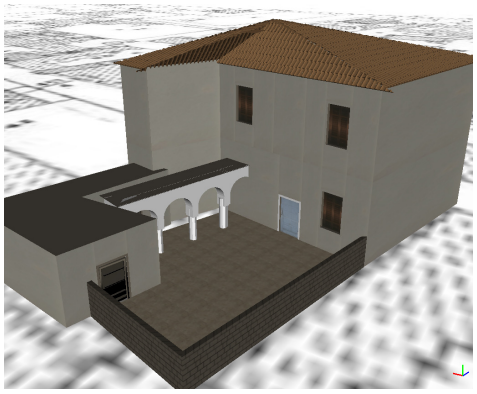
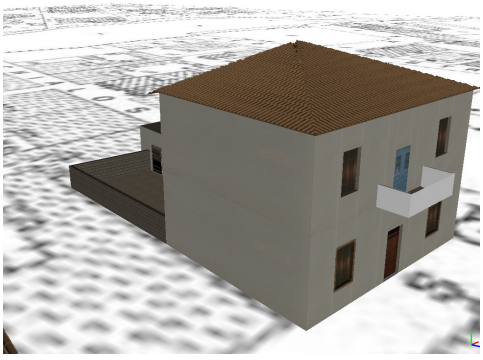
Σπίτι 2

Περίοδος 1	Περίοδος 2
	
	

Πίνακας 4.3: Παραγόμενα μοντέλα δύο διαφορετικών περιόδων(σπίτι 2), χρησιμοποιώντας την $2^{η}$ μεθοδολογία.

Επίσης δυναμικά αποφασίζεται το μοντέλο της πόρτας και τα textures της πρόσοψης. Τα αποτελέσματα του μοντέλου του δεύτερου σπιτιού τα βλέπουμε στο *πίνακα 4.3*.

Σπίτι 3

Περίοδος 1	Περίοδος 2
	
	
	

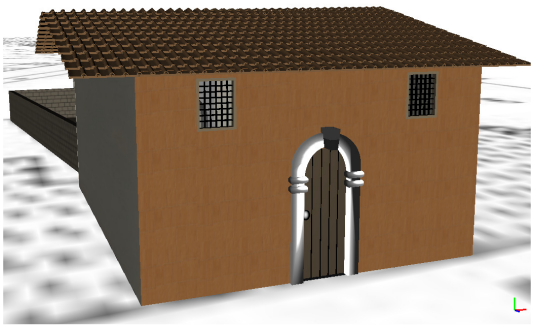
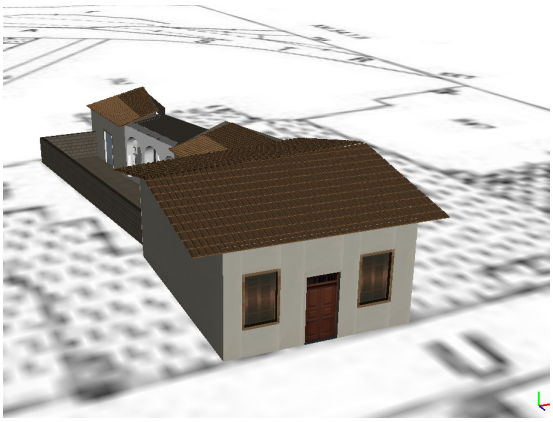
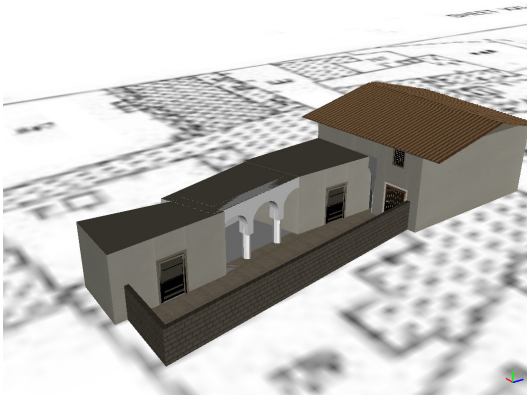
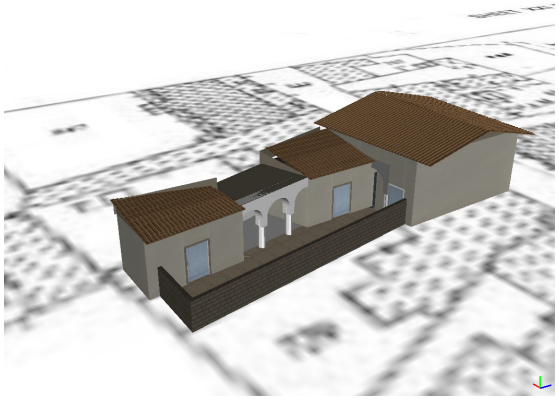
Πίνακας 4.4: Αρχικό μοντέλο σπιτιού τύπου Δίπατο χρησιμοποιώντας τη 2η μεθοδολογία

Για πολύπλοκα σχήματα σπιτιών, όπως το *σπίτι 3(πίνακας 4.4)* η κανονική οροφή δεν εφαρμόζεται στους κανόνες του CityEngine, έστω και αν είναι δηλωμένη ρητά. Για τέτοιου είδους σχήματα, δηλαδή σχήματα τα οποία αποτελούνται από περισσότερα από

4 σημεία, εφάρμοσα την οροφή τύπου πυραμίδας. Σε δύσκολες περιπτώσεις σχημάτων, ακόμη και αυτού του είδους οροφή αλλοιώνεται, με αποτέλεσμα να υπάρχουν κάποια κενά(ανοίγματα) στην κορυφή της πυραμίδας.

Σπίτι 4

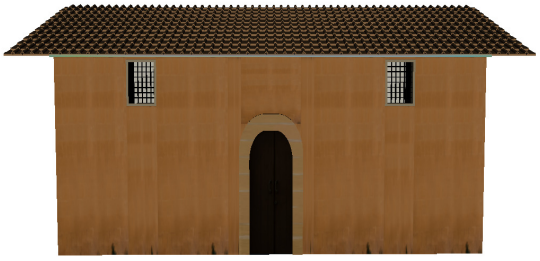
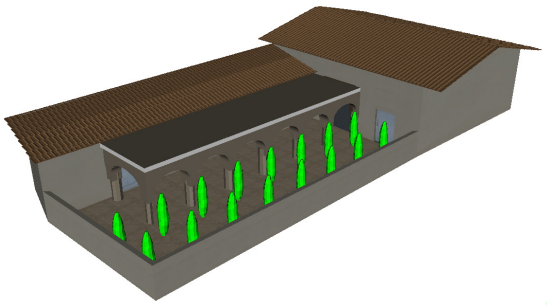
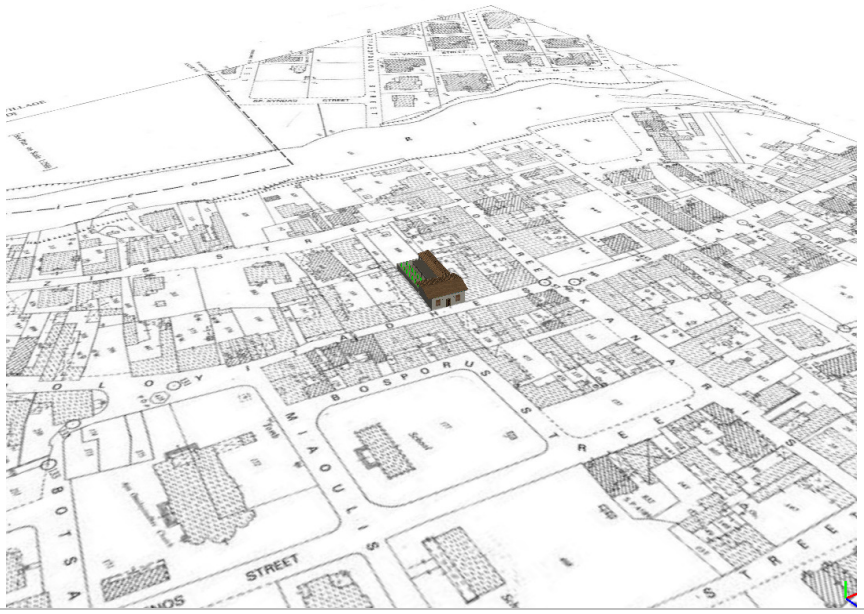
Στο *πίνακα 4.5* βλέπουμε το τελευτάιο διαφορετικό σπίτι το οποίο επιλέχθηκε για να δείξω.

Περίοδος 1	Περίοδος 2
	
	

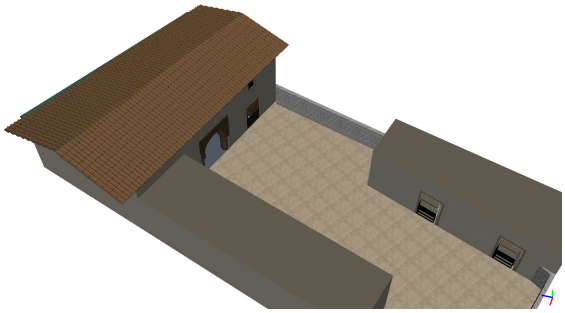
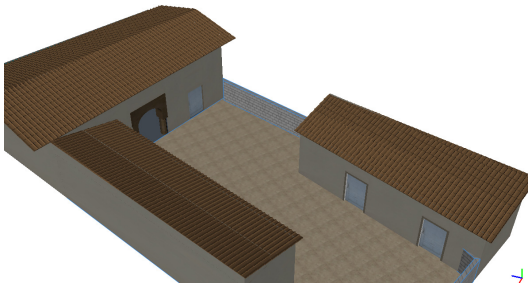
Πίνακας 4.5: Μοντέλα σπιτιού δύο διαφορετικών περιόδων,τα οποία δημιουργήθηκαν χρησιμοποιώντας την $2^{η}$ μεθοδολογία.

Τελικά κτίρια

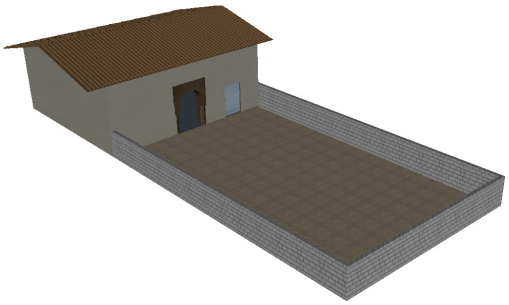
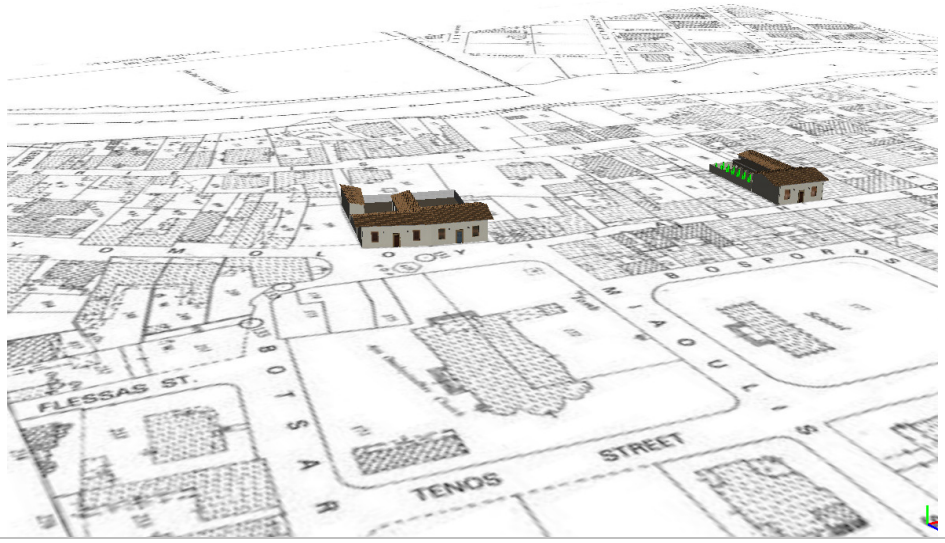
Αφού η χρησιμοποίηση του λογισμικού Maya δεν ήταν εύκολη, μετά από παρότρυνση κάποιου συμφοιτητή μου, χρησιμοποίησα το λογισμικό Google Sketchup για την δημιουργία των σχημάτων των σπιτιών. Επίσης διορθώθηκαν κάποια μικροπροβλήματα τα οποία είναι εμφανές στις πιο πάνω εικόνες. Διορθώθηκαν δηλαδή οι στέγες των κανονικών σπιτιών και οι καμάρες. Όσο αφορά την στέγη του δίπατου σπιτιού παρέμεινε η στέγη τύπου πυραμίδας, αφού η κανονική οροφή όταν το σχήμα του κτιρίου είναι παράξενο (δηλαδή αποτελείται από περισσότερα από τέσσερα σημεία) το CityEngine δεν την εφαρμόζει και παραμένει επίπεδο. Ακόμη όμως και με την οροφή τύπου πυραμίδας, σε πολύπλοκα σχήματα υπάρχουν κάποια κενά στην κορυφή της στέγης λόγω αδυναμίας του λογισμικού. Επιπρόσθετα προστέθηκαν κάποια μοντέλα για να γίνουν πιο ρεαλιστικά τα σπίτια. Έχουν προστεθεί διάφορα μοντέλα πορτών και για τις δύο περιόδους, διαφορετικά textures για τον φράκτη και για την αυλή, και μοντέλα λαμπών και αγωγών του νερού για τα σπίτια της δεύτερης περιόδου. Εν κατακλείδι διαμορφώθηκαν κάποιοι κανόνες και επίσης προσθέσαμε μερικούς νέους. Τα αποτελέσματα των τελικών μοντέλων που παράχθηκαν απεικονίζονται στους *πίνακες 4.7-4.13*. Οι κανόνες που γράφτηκαν βρίσκονται στο Παράρτημα Β.

Περίοδος 1	Περίοδος 2
	
	
	

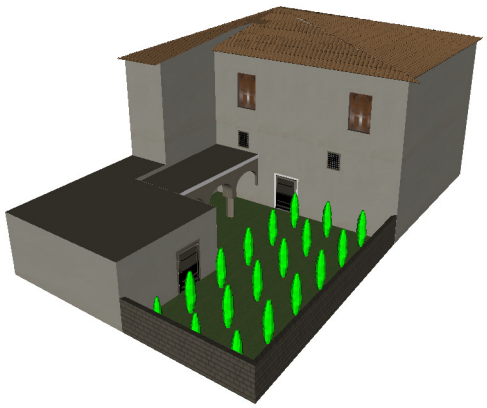
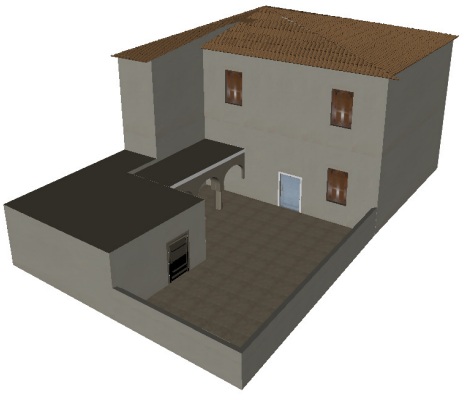
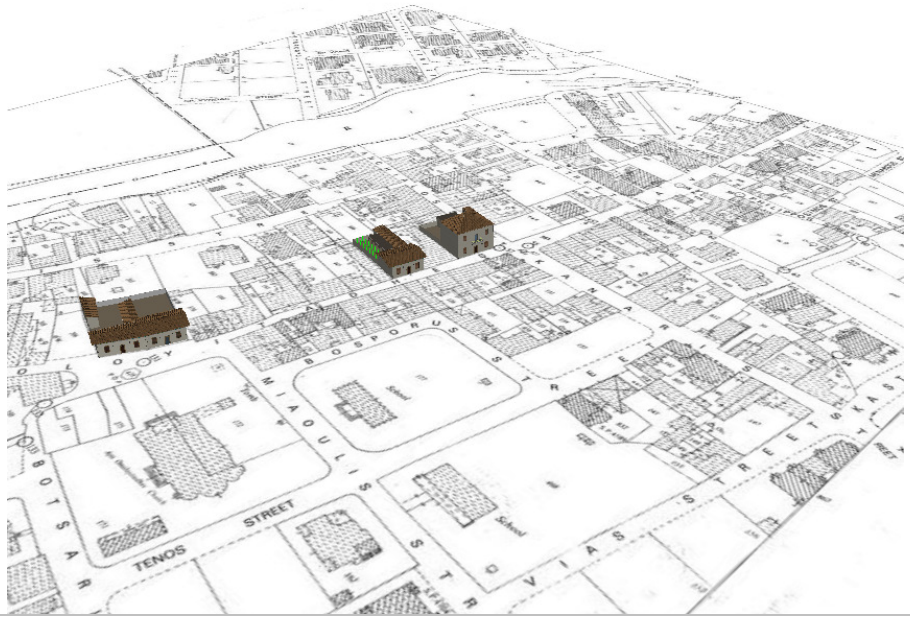
Πίνακας 4.7: Τελικά αποτελέσματα μοντέλων δύο περιόδων παραγόμενα απο το λογισμικό CityEngine. Στην κάτω εικόνα βλέπουμε το παραγόμενο μοντέλο πάνω στο χάρτη.

Περίοδος 1	Περίοδος 2
	
	
	

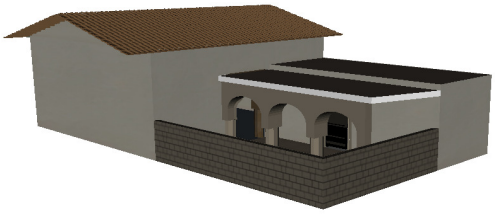
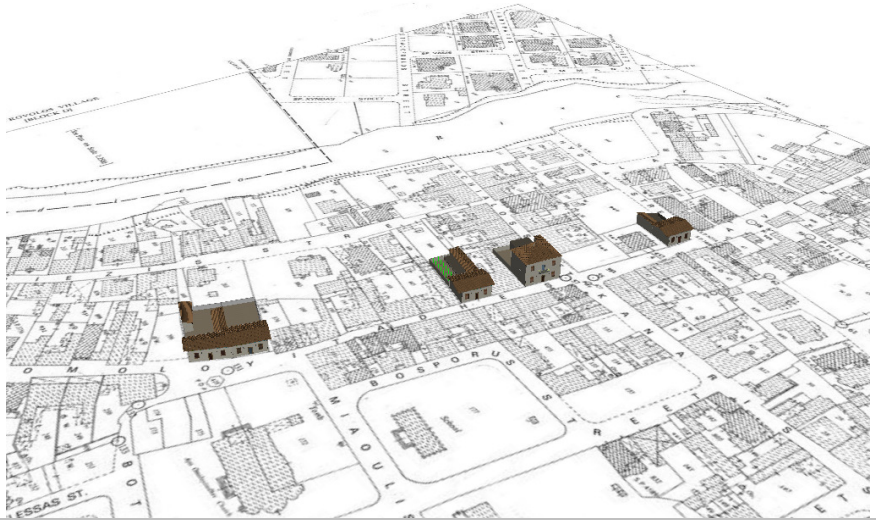
Πίνακας 4.8: Τελικά αποτελέσματα μοντέλων δύο περιόδων παραγόμενα απο το λογισμικό CityEngine. Στην κάτω εικόνα βλέπουμε τα παραγόμενα μοντέλα πάνω στο χάρτη.

Περίοδος 1	Περίοδος 2
	
	
	

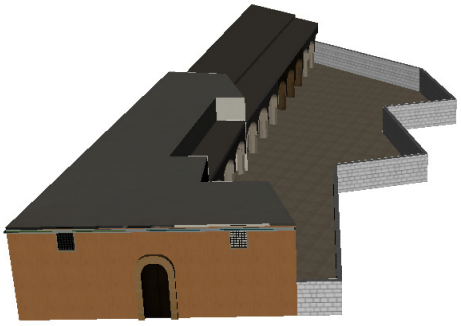
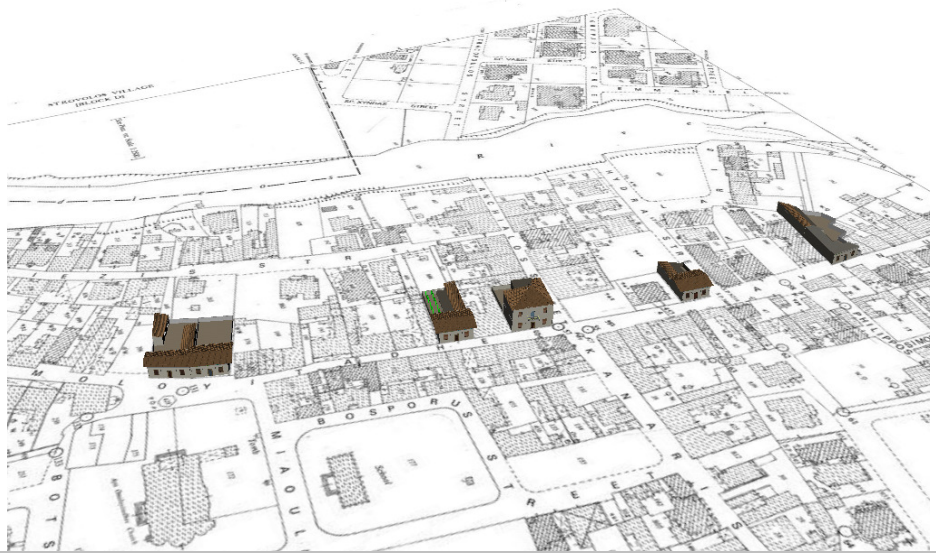
Πίνακας 4.9: Τελικά αποτελέσματα μοντέλων δύο περιόδων παραγόμενα απο το λογισμικό CityEngine. Στην κάτω εικόνα βλέπουμε τα παραγόμενα μοντέλα πάνω στο χάρτη.

Περίοδος 1	Περίοδος 2
	
	
	

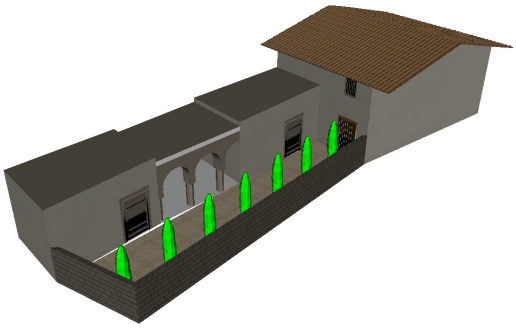
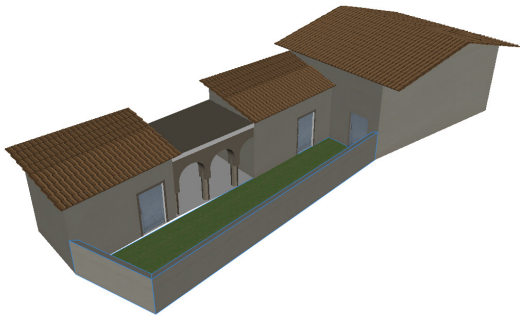
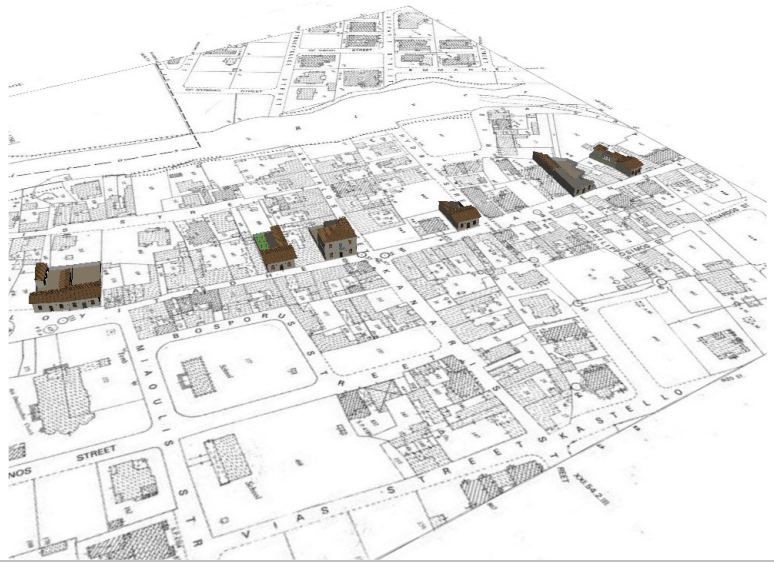
Πίνακας 4.10: Τελικά αποτελέσματα μοντέλων δύο περιόδων παραγόμενα απο το λογισμικό CityEngine. Στην κάτω εικόνα βλέπουμε τα παραγόμενα μοντέλα πάνω στο χάρτη.

Περίοδος 1	Περίοδος 2
	
	
	

Πίνακας 4.11: Τελικά αποτελέσματα μοντέλων δύο περιόδων παραγόμενα απο το λογισμικό CityEngine. Στην κάτω εικόνα βλέπουμε τα παραγόμενα μοντέλα πάνω στο χάρτη.

Περίοδος 1	Περίοδος 2
	
	

Πίνακας 4.12: Τελικά αποτελέσματα μοντέλων δύο περιόδων παραγόμενα απο το λογισμικό CityEngine. Στην κάτω εικόνα βλέπουμε τα παραγόμενα μοντέλα πάνω στο χάρτη.

Περίοδος 1	Περίοδος 2
	
	
	

Πίνακας 4.13: Τελικά αποτελέσματα μοντέλων δύο περιόδων παραγόμενα απο το λογισμικό CityEngine. Στην κάτω εικόνα βλέπουμε τα παραγόμενα μοντέλα πάνω στο χάρτη.

4.5.4 Προβλήματα Υλοποίησης

Ένα από τα προβλήματα αυτής της μεθόδου ήταν η χρησιμοποίηση του λογισμικού μοντελοποίησης Maya. Η διαπροσωπεία (inteface) του λογισμικού δεν είναι φιλικό προς το χρήστη (user friendly). Επίσης τα αποτυπώματα των κτιρίων τα οποία δημιουργήθηκαν μέσω του εργαλείου Create Polygon Tool, δεν ήταν ακριβείς. Σχεδιάστηκαν έχοντας ως background αλλά δεν καταφέραμε να μεγεθύνουμε τον χάρτη και να βλέπουμε περισσότερες λεπτομέρειες χωρίς να τροποποιηθούν οι μονάδες μέτρησης των αξόνων. Δηλαδή μεγεθύνοντας τον χάρτη, άλλαζε η κλίμακα και τα αποτελέσματα των σχημάτων στο χάρτη ήταν πολύ μικρά. Το συγκεκριμένο πρόβλημα λύθηκε χρησιμοποιώντας το λογισμικό Google Sketchup το οποίο είναι πολύ πιο απλό από το Maya. Η διαδικασία δημιουργίας των αποτυπωμάτων των κτιρίων ήταν παρόμοια.

Οι πληροφορίες που πήραμε από το Nicosia Master Plan ήταν συγκεκριμένα για ένα κτίριο (αυτό που είχε σχήμα όπως το αγγλικό γράμμα L), το οποίο θεωρούσαν κοινό σπίτι για τις εποχές εκείνες. Όμως, ακόμη και αυτές η πληροφορίες ήταν ελλειπής, αφού από συναντήσεις που είχα με τον κ Χρυσάνθου (μετά την συνάντηση με το αρχιτεκτονικό γραφείο) μάθαινα καινούργιους κανόνες, τους οποίους ενσωμάτωνα στη συνέχεια.

Κεφάλαιο 5

Συμπεράσματα

5.1 Εισαγωγή

Σε αυτό το κεφάλαιο παραθέτουμε συνοπτικά τα διάφορα στάδια της υλοποίηση της διπλωματικής μου εργασίας, τα αποτελέσματα και τα συμπεράσματα.

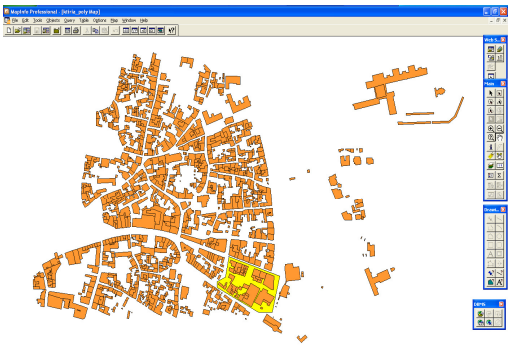
5.2 Αποτελέσματα

Στόχος της διπλωματικής μου εργασίας ήταν η διαδικαστική μοντελοποίηση της παλιάς Λευκωσίας. Ο τρόπος με τον οποίο θα δημιουργούσαμε την παλιά Λευκωσία ήταν γράφοντας κανόνες μέσω του λογισμικού CityEngine. Αρχικά δημιουργήσαμε ένα απλό αρχείο obj, το οποίο αναπαριστούσε το περίγραμμα ενός σπιτιού. Έτσι χρησιμοποιώντας την τεχνική trial and error μάθαινα πως λειτουργούν οι κανόνες στο εργαλείο και παράλληλα προσπαθούσα να κτίσω ένα σπίτι της παλιάς Λευκωσίας παρατηρώντας το από μια εικόνα. Το αρνητικό σε αυτή την περίπτωση ήταν ότι στις φωτογραφίες είχαμε μόνο τις προσόψεις. Πιο κάτω παρατηρούμε τα αρχικά αποτελέσματα στην *εικόνα 5.1* και στην *εικόνα 5.2*.

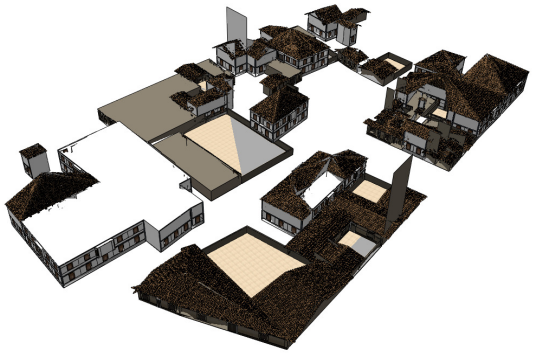
Εικόνα 5.1 – Αρχική Φωτογραφία

Εικόνα 5.2 – Μοντέλο του σπιτιού το οποίο παράχθηκε μέσω του λογισμικού CityEngine

Στη συνέχεια χρησιμοποιήσαμε τους ηλεκτρονικούς χάρτες της παλιάς Λευκωσίας του Nicosia Master Plan. Οι κανόνες οι οποίοι γράφτηκαν προήλθαν από τις φωτογραφίες που δόθηκαν από τον κ. Χρυσάνθου και από τον ίδιο. Η κυριότερη δυσκολία που αντιμετώπισα σε αυτό το στάδιο ήταν η μετατροπή του χάρτη σε μορφή η οποία να γίνεται αποδεκτή από τον λογισμικό CityEngine. Αφού μετέτρεψα τους χάρτες σε μορφή obj, δοκίμασα τους κανόνες σε ένα κομμάτι του χάρτη. Δυστυχώς όμως τα αποτελέσματα δεν ήταν τα επιθυμητά. Τα σχήματα των κτιρίων ήταν παράξενα σε αρκετές περιπτώσεις και δεν γνωρίζαμε περισσότερες λεπτομέρειες όσο αφορά την ύπαρξη κτιρίου σε κάποιο σχήμα, το είδος του κτιρίου, τη θέση του μπροστά κτιρίου, τη θέση του πίσω κτιρίου. Στις πιο κάτω εικόνες (*εικόνα 5.3* και *εικόνα 5.4*) βλέπουμε το μερικό χάρτη και τα αποτελέσματα από τους κανόνες:



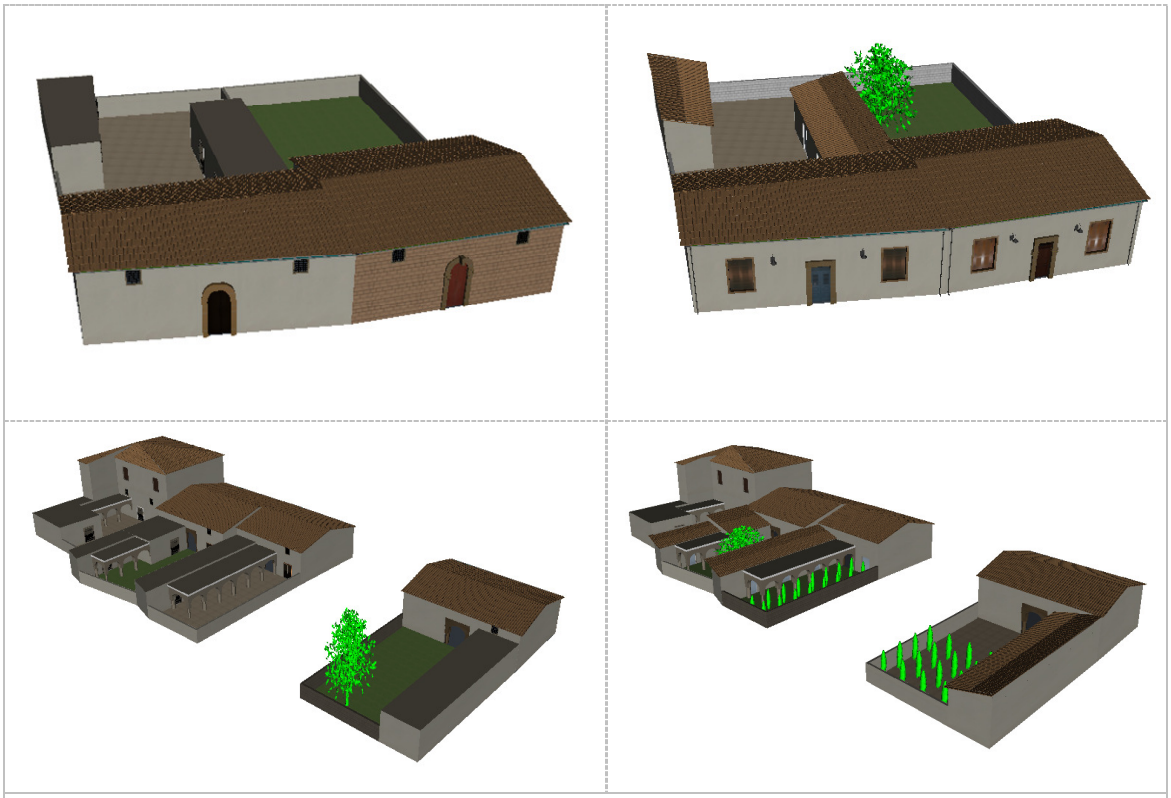
Εικόνα 5.3 – Δημιουργία μερικού χάρτη χρησιμοποιώντας το εργαλείο MapInfo



Εικόνα 5.4 – Μοντέλα του μερικού χάρτη

Αφού τα αποτελέσματα που είχαμε δεν ήταν επιθυμητά, προχωρήσαμε σε μια εναλλακτική λύση. Χρησιμοποιήσαμε κανονικό χάρτη ο οποίος είχε περισσότερες λεπτομέρειες. Δηλαδή μέσω αυτού του χάρτη μπορούσαμε να αναγνωρίσουμε που υπήρχε σπίτι και που ήταν άδειο οικόπεδο, το είδος του σπιτιού (αν ήταν δίπατο ή κανονικό), τη θέση του μπροστινού σπιτιού, την θέση των πίσω σπιτιών (δώματα) και το ακριβές περίγραμμα της αυλής. Σαρώσαμε τον χάρτη και τον μετατρέψαμε σε εικόνα. Ακολούθως, μέσω κάποιου λογισμικού μοντελοποίησης δημιουργήσαμε τα περιγράμματα ορισμένων σχημάτων, αφού η διαδικασία αυτή ήταν χειρονακτική και στο χρόνο που απέμενε ήταν αδύνατο να μοντελοποιήσουμε ολόκληρη την πόλη. Με άλλα λόγια, η δημιουργία των περιγραμμάτων δεν ήταν αυτόματη. Σε αυτό το στάδιο είχαμε συνάντηση με το αρχιτεκτονικό γραφείο Nicosia Master Plan όπου πήραμε περισσότερες πληροφορίες για τα σπίτια της παλιάς Λευκωσίας. Τα αποτελέσματα των κανόνων τα βλέπουμε στη παράγραφο ***Τελικά Κτίρια*** στο τέλος του προηγούμενου κεφαλαίου. Παραθέτω όμως κάποια αποτελέσματα πιο κάτω τα οποία μπορούμε να δούμε στις *εικόνες 5.5* και *5.6*.

Περίοδος 1	Περίοδος 2
	
	



Εικόνα 5.5: Τελικά κτίρια παραγόμενα απο το λογισμικό CityEngine

Για την δέυτερη περίοδο, δημιουργήθηκε μια οδός. Προσθέθηκαν δηλαδή περισσότερα σπίτια και κτίστηκε και δρόμος έτσι ώστε να γίνει πιο έντονος ο ρεαλισμός των μοντέλων. Το αποτελέσμα το βλέπουμε στην *εικόνα 5.6*.

Περίοδος 2

Εικόνα 5.6 – Δημιουργία μιας μικρής οδού για την δεύτερη περίοδο

5.3 Συμπεράσματα

Κατά τη διάρκεια της διπλωματικής μου εργασίας ήρθα σε επαφή με μια πληθώρα λογισμικών, επεκτείνοντας έτσι τις γνώσεις μου στη περιοχή των γραφικών. Το λογισμικό CityEngine καθιστά τη δημιουργία σπιτιών και πόλεων εύκολη και γρήγορη. Οι κανόνες διαχωρίστηκαν σε 8 διαφορετικά αρχεία για να είναι πιο κατανοητοί. Συνολικά χρησιμοποιήθηκαν 216 κανόνες για την αναπαράσταση σπιτιών δύο διαφορετικών περιόδων. Για την πιο ρεαλιστική απεικόνιση και ποικιλία σπιτιών χρησιμοποιήθηκαν αρκετά textures και μοντέλα. Συγκεκριμένα χρησιμοποιήθηκαν 3 διαφορετικά textures για τους τοίχους της πρώτης περιόδου, 2 διαφορετικά textures για τις πίσω πόρτες του μπροστά σπιτιού (μια για κάθε περίοδο), 2 διαφορετικά textures για τις πόρτες του πίσω σπιτιού(μια για κάθε περίοδο), 4(6) πόρτες για την πρόσοψη του μπροστά σπιτιού για την περίοδο 1(περίοδο 2), 4 διαφορετικά είδη παραθύρων, 2

μοντέλα δέντρων, 3 διαφορετικά είδη φρακτών και 2 είδη οροφών. Η δημιουργία 10 σπιτιών εκτελείται σε χρόνο ~1 λεπτού. Συνεπώς η δημιουργία μιας πόλης είναι θέμα μερικών λεπτών (με την προϋπόθεση φυσικά ότι χρησιμοποιείται ένα δυνατός ηλεκτρονικός υπολογιστής).

Συνολικά κτίστηκαν 13 ολοκληρομένα σπίτια δύο περιόδων. Για την απεικόνιση των περιγραμμάτων των σχημάτων χρειάστηκαν 52 πολύγωνα. Για την απεικόνιση των ολοκληρομένων μοντέλων (σπιτιών και δρόμου) που παράχθηκαν από το CityEngine χρειαστηκαν 2928375 πολύγωνα. Δηλαδή για κάθε σπίτι χρείαστηκαν περίπου ~225 000 πολύγωνα.

Η χρησιμοποίηση του λογισμικού CityEngine είναι γενικά απλή. Με βάση όμως τους συγγραφείς του άρθρου η μαζική μοντελοποίηση πόλεων με πολύπλοκα σχήματα πιθανότατα να μην φέρει τα επιθυμητά αποτελέσματα και επίσης να μην είναι αποδοτική. Επίσης τα ρεαλιστικά αποτελέσματα που δείχνουν στο άρθρο προήλθαν μετά από επεξεργασία από διάφορα λογισμικά (όπως το Maya για πρόσθεση του ανάλογου φωτισμού και το Pixar’s RenderMan για το rendering).

Τα σπίτια τα οποία βλέπουμε στην εικόνα 5.5 είναι αποτέλεσμα αυστηρών κανόνων αποκλειστικά από το λογισμικό CityEngine. Για την δημιουργία όμως ολόκληρης της πόλης με ακόμη πιο ρεαλιστικά μοντέλα, κατά τη γνώμη μου χρειάζεται στενή συνεργασία με κάποιο αρχιτέκτονα που να γνωρίζει πολύ καλά την τυπολογία των σπιτιών της παλιάς Λευκωσίας και με κάποιο σχεδιαστή γραφικών (graphic designer) για την ρεαλιστική μοντελοποίηση μοντέλων και textures.

Βιβλιογραφία

[1] P.Muller, P.Wonka, S.Haegler, A.Ulmer and Luc Van Gool, «Procedural Modeling of Buildings», ACM Transactions on Graphics. volume 25. number 3. pages 614-623. 2006. Proceedings of SIGGRAPH 2006.

[2] Peter Wonka, Michael Wimmer, Francois Sillion, and William Ribarsky, «Instant Architecture». *ACM Transactions on Graphics 22*, 3, 669–677

[3] D.Demi, «THE WALLED CITY OF NICOSIA, Typology Study» United Nations Development Program, Nicosia 1997. ISBN 9963-8272-0-9

[4] B. Watson, P. Müller, P. Wonka and A. Fuller, «Urban Design and Procedural Modeling» *Course Notes of the ACM SIGGRAPH 2007*

[5] Α. Ευθυμίου, «Δημιουργία Τρισδιάστατων Εικονικών Μοντέλων Ξεκινώντας από Χάρτες και Φωτογραφίες. Μοντελοποίηση των Εντός των Τειχών Λευκωσίας », Ατομική Διπλωματική Εργασία, 2002.

[6] Μ. Δικαιάκου, «Nicosia 3-D Model», *MSc Thesis, 2003, University College of London*

[7] Procedural Website, http://www.procedural.com:9099/help/index.jsp?topic=/com.procedural.cityengine.help/html/toc.html

[8] Procedural Website, http://www.procedural.com/support/forum.html

[9] Procedural Website, http://www.procedural.com:9099/help/index.jsp?topic=/com.procedural.cityengine.help/html/tutorials/Tutorial_07_Facade_Modeling/toc.html

[10] L-systems,
http://en.wikipedia.org/wiki/L-system

Παράρτημα Α

Σε αυτό το παράρτημα παραθέτω τον κώδικα που γράφτηκε στο Intellij IDEA για την τροποποίηση των ηλεκτρονικών δεδομένων.

```
import java.util.ArrayList;

/**
 * Created by IntelliJ IDEA.
 * User: Savvas Michael
 * File: reshape.java
 * Date: Mar 28, 2009
 * Time: 6:27:11 PM
 * To change this template use File | Settings | File Templates.
 */
public class reshape {

   public static ArrayList<Building> myBuildings;
   public static ArrayList<Vector3d> allVertices;
   public static FileReader readFile;



   /*
    *  copy vertices of each building to the ArrayList allVertices
    */
   public static void copyBuildingsVertices(){

      int i,j;
        allVertices = new ArrayList<Vector3d>();

        for(i=0; i<myBuildings.size(); i++){
            for(j=0; j<myBuildings.get(i).getVerticeSize(); ++j){
```

```
                allVertices.add(myBuildings.get(i).getVertice(j));

        }
    }
    System.out.println("allVertices size: "+allVertices.size());
}

/*
 * removes the extra points from each building
 */
public static void removeExtraPoints(){

    int i,j;
    Vector3d a,b,c,ab,cb;
    int faceSize;
    int buildingSize;
    int removef;
    // check for every building
     buildingSize=myBuildings.size();
    for(i=0; i<buildingSize; ++i){

        //check the faces of each building
        j=0;
        faceSize=myBuildings.get(i).getFaceSize();

        while (j<faceSize){

            if (j==0){
                int tmpf= faceSize-1;

                a = new Vector3d(allVertices.get(myBuildings.get(i).getFace(tmpf)-1));
                b = new Vector3d(allVertices.get(myBuildings.get(i).getFace(j)-1));
                c = new Vector3d(allVertices.get(myBuildings.get(i).getFace(j+1)-1));
```

```
            removef=myBuildings.get(i).getFace(j);
          }
          else if(j==(faceSize-1)){
            a = new Vector3d(allVertices.get(myBuildings.get(i).getFace(faceSize-2)-
1));
            b = new Vector3d(allVertices.get(myBuildings.get(i).getFace(faceSize-1)-
1));
            c = new Vector3d(allVertices.get(myBuildings.get(i).getFace(0)-1));
            removef=myBuildings.get(i).getFace(faceSize-1);
          }else{
             a = new Vector3d(allVertices.get(myBuildings.get(i).getFace(j-1)-1));
             b = new Vector3d(allVertices.get(myBuildings.get(i).getFace(j)-1));
             c = new Vector3d(allVertices.get(myBuildings.get(i).getFace(j+1)-1));
             removef=myBuildings.get(i).getFace(j);
          }

          ab = a.subtract(b);
          cb = c.subtract(b);

          if (ab.angle(cb)> 135.0 && ab.angle(cb)<= 225.0){
             myBuildings.get(i).removeFace(removef);
             faceSize =  myBuildings.get(i).getFaceSize();
          }
          else
             j++;


        }
      }


  }
```

```
/*
 * print the contents of the ArrayList myBuildings
 */
public static void printBuildings(){
   int i, j;

   for(i=0; i<myBuildings.size(); i++){
      System.out.println("\nBuilding: "+i+"\n===============\n");
      for(j=0; j<myBuildings.get(i).getVerticeSize(); ++j){
         System.out.print("v "+j+" : ");
         myBuildings.get(i).printVertice(j);
      }
      System.out.print("f ");

      for(j=0; j<myBuildings.get(i).getFaceSize(); ++j){
         System.out.print(myBuildings.get(i).getFace(j)+ " ");
      }
      System.out.println();
   }
}

/*
 *  main class
 */
public static void main(String[] args){

   Vector3d ab = p1.subtract(p2);
   Vector3d bc = p3.subtract(p2);
   System.out.println("angle = "+ab.angle(bc));

   // read the buildings
   myBuildings = new ArrayList<Building>();
   FileReader.readInputFile();
```

```
        printBuildings();
        copyBuildingsVertices();

        removeExtraPoints();

        printBuildings();

        FileReader.writeOutputFile();

        return ;
    }
}
```

```
import java.io.*;
import java.util.ArrayList;
```

```
import org.apache.commons.lang.math.NumberUtils;

import javax.swing.*;


/**
 * Created by IntelliJ IDEA.
 * User: Savvas Michael
 * File: FileReader.java
 * Date: Mar 30, 2009
 * Time: 5:31:41 PM
 * To change this template use File | Settings | File Templates.
 */
public class FileReader {

    public static boolean flagFace=false;
    public static Building tempBuilding;
    public static double[] tempVector = new double[3];
    public static int counter;
    public static int strType=0;
   /*
    *  Check the current input if it is number or string
    */
   public static void checkToken(String str,Integer newLine){

       if (NumberUtils.isNumber(str)){
            System.out.print(str+" ;");
            if (flagFace){
               tempBuilding.addFace(Integer.parseInt(str));
            }else{
               tempVector[counter]=Double.parseDouble(str);
               counter=counter+1;
```

```
            }
        }else if(str.equals("v")){
            System.out.print("vertice ");
             if (counter!=0){
                Vector3d tmp = new
Vector3d(tempVector[0],tempVector[1],tempVector[2]);
                tempBuilding.addVertice(tmp);
             }
             counter=0;
            strType=0;
        }else if(str.equals("g")){
            System.out.print("group ");
            strType=1;
        }else if(str.equals("o")){
            System.out.print("object ");
             if (counter!=0){
             Vector3d tmp = new
Vector3d(tempVector[0],tempVector[1],tempVector[2]);
             tempBuilding.addVertice(tmp);
             }
             counter=0;
             strType=2;
        }else if(str.equals("s")){
              // to do
            strType=0;
        }else if(str.equals("f")){
            System.out.print("face ");
            flagFace=true;
        }else{
            if (strType==1){
                tempBuilding.setGroupName(str);
            }else if (strType==2){
                tempBuilding.setObjectName(str);
```

```
            }
            System.out.print(str+" ");
        }
      if(newLine==1){
         System.out.print("\n");
         if (flagFace){
            reshape.myBuildings.add(tempBuilding);
            tempBuilding=new Building();
            flagFace=false;
         }
      }
   }

   /*
    *    Reads the input file
    */
   public static ArrayList<Building> readInputFile() {

      BufferedReader in;

      try {
         boolean flag=true;
         in = new BufferedReader(new InputStreamReader(new
FileInputStream("src/fewShapes.obj")));

         StreamTokenizer streamToken = new StreamTokenizer(in);
         streamToken.resetSyntax();
         streamToken.wordChars(0, Character.MAX_HIGH_SURROGATE);
         streamToken.whitespaceChars(0, ' ');
         streamToken.eolIsSignificant(true);

         String str = null;
```

```
        streamToken.nextToken();

        tempBuilding = new Building();
        counter=0;

        while (streamToken.ttype != StreamTokenizer.TT_EOF) {

           str = streamToken.sval;

           while (streamToken.nextToken() != StreamTokenizer.TT_EOF
                 && streamToken.ttype == StreamTokenizer.TT_EOL) {
              //str += "\r\n";

              flag=false;
           }
           if (flag)
              checkToken(str,0);
           else checkToken(str,1);

           flag=true;
        }
        in.close();
     } catch (Exception e) {
        System.out.println("---------- File not Found!!! ----------");
        e.printStackTrace();
     }
     return null;
  }

  /*
   * Write the output obj file
   */
  public static void writeOutputFile(){
```

```
int i,j;
        Writer out;
Writer vert;

        try {
                String fileName = (JOptionPane
                        .showInputDialog("Enter Output
FileName(without extension) : "));
                fileName += ".obj";

                out = new OutputStreamWriter(new FileOutputStream(fileName,
true));

    vert= new OutputStreamWriter(new FileOutputStream("vertices.obj", true));

    for(i=0; i<reshape.myBuildings.size(); ++i){

      for(j=0; j<reshape.myBuildings.get(i).getVerticeSize(); ++j){
            Vector3d tmp = reshape.myBuildings.get(i).getVertice(j);
            out.write("v ");
            out.write(tmp.x+" "+tmp.y+" "+tmp.z+"\n");
      }
      out.write("\no "+reshape.myBuildings.get(i).objectName+"\n");
      out.write("g "+reshape.myBuildings.get(i).groupName+"\n\n");

      out.write("s off\n");

      out.write("f ");
      for(j=0; j<reshape.myBuildings.get(i).getFaceSize(); ++j){
        out.write(reshape.myBuildings.get(i).getFace(j)+ " ");
      }
      out.write("\n");
```

```
        }
        for(i=0; i<reshape.allVertices.size(); ++i){
           Vector3d v = reshape.allVertices.get(i);
           vert.write("v "+v.x+" "+v.y+" "+v.z+"\n");
        }

                        System.out.println("Finish exporting File!");
                        out.close();
        vert.close();
                } catch (IOException e) {
                        e.printStackTrace();
                }
        }

}
```

```
import java.util.ArrayList;
```

```
/**
 * Created by IntelliJ IDEA.
 * User: Savvas Michael
 * File: Building.java
 * Date: Mar 30, 2009
 * Time: 5:32:06 PM
 * To change this template use File | Settings | File Templates.
 */
public class Building {
    public ArrayList<Vector3d> vertices;
    public String groupName;
    public String objectName;
    public ArrayList<Integer> faces;

    public Building(){
        vertices = new ArrayList<Vector3d>();
        faces    = new ArrayList<Integer>();
    }
    public void addVertice(Vector3d x){
        vertices.add(x);
    }

    public Vector3d getVertice(int i){
        return vertices.get(i);
    }
    public void printVertice(int i){
        System.out.println(vertices.get(i).x+" "+vertices.get(i).y+" "+vertices.get(i).z);
    }

    public Integer getVerticeSize(){
        return vertices.size();
    }
```

```
    public void setGroupName(String groupName){
        this.groupName=new String(groupName);
    }

    public void setObjectName(String objectName){
        this.objectName=new String(objectName);
    }

    public void addFace(Integer f){
        faces.add(f);
    }

    public Integer getFaceSize(){
        return faces.size();
    }
    public Integer getFace(Integer index){
        return faces.get(index);
    }
    public void removeFace(Integer index){
         faces.remove(index);
    }
}
```

```
/**
```

```
 * Created by IntelliJ IDEA.
 * User: Savvas Michael
 * File: Vector3d.java
 * Date: Mar 28, 2009
 * Time: 7:06:30 PM
 * To change this template use File | Settings | File Templates.
 */

public class Vector3d {
    double x;
    double y;
    double z;
    public Vector3d(double x, double y, double z)    {
        this.x=x;
        this.y=y;
        this.z=z;
    }
    public Vector3d(Vector3d v){
        this.x = v.x;
        this.y = v.y;
        this.z = v.z;
    }

    public Vector3d getVector3d(){
     return this;
    }

    Vector3d add(Vector3d  v)
    {
        this.x = x + v.x;
        this.y = y + v.y;
        this.z = z + v.z;
        return this;
```

```
    }

    Vector3d subtract(Vector3d B)
    {
       return new Vector3d(x - B.x, y - B.y, z - B.z);

    }

    Vector3d negate()
    {
       return new Vector3d(-x,-y,-z);
    }


    Vector3d scale(double s)
    {
       return new Vector3d(x*s, y*s, z*s);
    }

    double angle(Vector3d v)
    {
       double cosalpha = dotProduct(this, v) / (this.length() * v.length());
       if (cosalpha>1.0) cosalpha=1.0;
      return java.lang.Math.toDegrees(java.lang.Math.acos(cosalpha));

    }

    Vector3d copy()
    {
       return new Vector3d(x,y,z);
    }

// Dot product of two Vector3d's
```

```
    double dotProduct(Vector3d a, Vector3d b)
    {
        return a.x*b.x + a.y*b.y + a.z*b.z;
    }


// Cross product of two Vector3d's
    Vector3d crossProduct(Vector3d a,Vector3d b)
    {
        return new Vector3d(a.y*b.z - a.z*b.y,
                            a.z*b.x - a.x*b.z,
                            a.x*b.y - a.y*b.x);
    }

    double length(){
        return  java.lang.Math.sqrt(  java.lang.Math.pow(x,2)+  java.lang.Math.pow(y,2)  +
java.lang.Math.pow(z,2));
    }









}
```

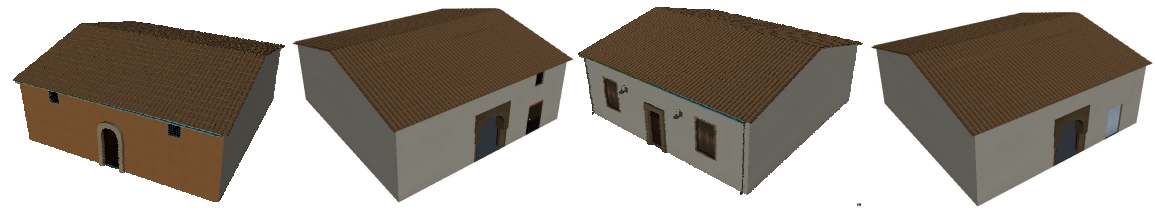
Παράρτημα Β

Σε αυτό το παράρτημα παραθέτω τους κανόνες που γράφτηκαν στο λογισμικό CityEngine για την μοντελοποίηση της Λευκωσίας. Χρησιμοποιήθηκαν 216 κυρίως κανόνες για τα σπίτια, οι οποίοι διαχωρίστηκαν σε 8 αρχεία για καλύτερη ομοιομορφοία. Το μειονέκτημα (του διαχωρισμού σε αρχεία) σε αυτή την περίπτωση ήταν οτι σε ορισμένες περιπτώσεις σε κάποια αρχεία υπήρχαν ίδιοι (περίπου) κανόνες. Για την δεύτερη περίοδο όπου κτίστηκε μια οδός δημιουργήθηκε ακόμη ένα αρχείο για τον δρόμο. Επίσης υπάρχουν 2 αρχεία κανόνων (*mainRules.cga* και *mainRulesPeriod2.cga*) τα οποία καλούν τα υπόλοιπα αρχεία με τις ανάλογες παραμέτρους.

```
/**
 * File:    mainRules.cga
 * Created: 23 Apr 2009 11:03:35 GMT
 * Author:  Savvas Michael
 */

##############################
###   IMPORTS              ###
##############################

import fBuilding : "FrontHouse.cga"
import bBuilding : "BackHouse.cga"
import yard      : "yard.cga"
import arches    : "arches.cga"
import dipato      : "Dipato.cga"



##############################
###   RULES                ###
##############################

Fronthouse--> fBuilding.houseType(1,1)
Fronthouse1--> fBuilding.houseType(1,2)

Backhouse --> bBuilding.secondHouse(1)

Kamara     --> arches.myArch(1)
Kamara4        --> arches.myArch(2)

Yard       --> yard.yard(1)
Yard4    --> yard.yard(2)
Yard5    --> yard.yard(3)

Dipato         --> dipato.Dipato(1)
DipatoBackhouse-->bBuilding.secondHouse(1)
```

```
/**
 * File:    mainRulesPeriod2.cga
 * Created: 2 May 2009 11:14:48 GMT
 * Author:  Savvas Michael
 */


##############################
###   IMPORTS              ###
##############################

import fBuilding : "FrontHouse.cga"
import bBuilding : "BackHouse.cga"
import yard      : "yard.cga"
import arches    : "arches.cga"
import dipato      : "Dipato.cga"



##############################
###   RULES                ###
##############################

Fronthouse--> fBuilding.houseType(2,1)
Fronthouse1--> fBuilding.houseType(2,2)

Backhouse --> bBuilding.secondHouse(2)

Kamara     --> arches.myArch(1)
Kamara4        --> arches.myArch(2)

Yard       --> yard.yard(1)
Yard4    --> yard.yard(2)
Yard5    --> yard.yard(3)

Dipato         --> dipato.Dipato(2)
DipatoBackhouse-->bBuilding.secondHouse(2)
```

Αρχείο κανόνων για κτίσιμο του μπροστά σπιτιού



```
/**
 * File:    FrontHouse.cga
 * Created: 23 Apr 2009 11:07:17 GMT
 * Author:  Savvas Michael
 */

##############################
###   IMPORTS              ###
##############################

import roofType: "roof.cga"
import myText  : "myTextures.cga"
import arches  : "arches.cga"


##############################
###   ATTRIBUTES           ###
##############################

attr diameter           = 1.92
attr doorAsset = "temples/door_parthenon.obj"
attr doorCorniceAsset = "temples/door-cornice_parthenon.obj"
attr topdoorAsset = "temples/topdoor-element_parthenon.obj"

attr wallThickness      = 0.2
attr corniceProjection = diameter*0.1
attr doorFrameW         = 0.1
attr doorFrameH         = 0.2
attr doorH              = 0.5
attr doorD              = 0.1
attr doorWindowH  = doorH*0.15


##############################
###   COLORS               ###
##############################

mov = "#837a6b"
brown="#7D7159"
black="#000000"
grey ="#6b7785"




##############################
###   RULES                ###
##############################


Fronthouse-->fhouse


fhouse      --> 50% : houseType(1,1)
                      else: houseType(2,2)

houseType(period,archBoolean)-->
                      extrude(4)chooseWindow(period,archBoolean)
```

```
chooseWindow(period,archBoolean)    -->
            25% : Building(period,1,archBoolean)
            25% : Building(period,2,archBoolean)
            25% : Building(period,3,archBoolean)
            else: Building(period,4,archBoolean)


Building(period,windowType,archBoolean)    -->
      comp(f){
            top   : roof(gable, 14,1)roofType.myRoof |
            front : FrontFacade(period,windowType)|
            back  : backFacade(period,archBoolean) |
            side  : Wall |
            bottom: color(grey)nil.
            }

FrontFacade(period,windowType)-->
            case scope.sx<5 : myText.wall
            else:
                  case period==1:
                        chooseFrontFacade(period,windowType)
                  else:
                        frontFacade(period,1,windowType,0)



chooseFrontFacade(period,windowType)-->
            25% :
                  frontFacade(period,1,windowType,0)
            25%:
                  frontFacade(period,2,windowType,0)
            25%:
                  frontFacade(period,3,windowType,0)
            else:
                  frontFacade(period,4,windowType,0)


Wall        -->  myText.wall


//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\
////\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//


frontFacade(period,wallType,windowType,houseType) -->
      case houseType==0:
            split(y){
            ~1:   split(x){
                  '0.4 : LRFrontFacade(period,wallType,windowType,1,0)
                  |   // 1:: leftside
                  ~1   : iliakos(period,wallType)
                  |
                  '0.4 : LRFrontFacade(period,wallType,windowType,2,0)
                  // 2:: rightside
                   }
            |0.1:ledge
            }
      else:
```

```
            split(y){
                  ~1: split(x){
                  '0.4 : LRFrontFacade(period,wallType,windowType,1,1)
                  |   // 1:: leftside
                  ~1    : iliakos(period,wallType)
                  |
                  '0.4 : LRFrontFacade(period,wallType,windowType,2,1)
                   // 2:: rightside
                   }

                  | 0.15: myText.wall//DipatoLedge(period)
                  }


ledge-->
      setupUV(0,scope.sx,scope.sy)
      s(0.3,scope.sy/2,scope.sx)
      t(0,0,0.3)
      r(0,89,0)
      i("ledgeModel2.obj")
      bakeUV(0)


ledgeDipato-->split(x){1:HouseUpTex}*
HouseUpTex-->
             setupUV(0,scope.sx,scope.sy)
             set(material.colormap, "HouseUp.jpg")
             bakeUV(0)


chooseWall(wallType)-->
      case wallType==1:
            myText.wall
       case wallType==2:
            myText.wall2
       case wallType==3:
            myText.wall3
       else:
            myText.wall4







iliakos(period,wallType) -->
      split(x){
            ~1:    chooseWall(wallType)|
             1.40: DoorF(period,wallType)          |
            ~1:    chooseWall(wallType)
             }


DoorF(period,wallType)  -->
      split(y) {
            2.6   : DoorFrameX(period,wallType) |
            ~1    :  DoorTop(wallType)
             }
```

```
DoorTop(wallType) --> chooseWall(wallType)


DoorFrameX(period,wallType)-->
      split(x){
            0.1 : DoorFrame(wallType) |
            1.2:DoorFrameY(period,wallType) |
            0.1:DoorFrame(wallType)}


DoorFrameY(period,wallType)-->
      split(y){
            2.5 : Door(period) |
            0.1: DoorFrame(wallType)
            }


DoorFrame(wallType)--> chooseWall(wallType)


Door(period)-->
       case period==1:
            DoorPeriod1
      else :
            DoorPeriod2



LRFrontFacade(period,wallType,windowType,side,houseType)-->
      case
            period==1: period1(wallType)
      else:
            period2(wallType,windowType,side,houseType)


//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\
////\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//


/*
 * creation of windows of period 1
 */


period1(wallType)-->
      split(x) {
             scope.sx/2-0.35: chooseWall(wallType) |
             0.7 : WindowPart(wallType)            |
             ~1  : chooseWall(wallType)
                                              }


WindowPart(wallType)    -->
```

```
      split(y) {
              2.95  : chooseWall(wallType)   |
              0.9: WindowFrameX |
              ~1 : chooseWall(wallType)
            }


WindowFrameX  -->
      split(x){
            0.05:WindowFrame |
            0.6 : WindowFrameY |
            ~1:WindowFrame}

WindowFrameY  -->
      split(y){
            0.05:WindowFrame |
            0.8 : Window       |
            ~1:WindowFrame}


Window        --> WindowAxis

WindowAxis    -->
      case pivot.oz==0:
            extrude(z,-0.1)WindowTile
      else:
            setPivot(yzx, 7)
            alignScopeToAxes(y)
            extrude(z,-0.1)WindowTile


WindowTile    -->
      alignScopeToGeometry(yUp, auto)
      comp(f){
            front: WindowLot |
            back: NIL/*color("#fff000")windowTex.*/ |
            all: color(brown)Sides. }


WindowFrame   --> color(brown)nil.




WindowLot     --> color(black)
                  center(x)
                  s(0.8,'1,-0.05)
                  i("builtin:cube")

                  split(y){0.1 : HBars}*
                  split(x){0.1 : VBar}*


VBar -->
      s(0.02,'1,'1)
      t(0,0,0.1)VBar.
      t(0.1,0,0)VBar.
```

```
HBars -->
      s('1,0.02,-0.05)
      t(0,0.1,0.1) HBar.
      t(0,0.1,0) HBar.


/*
 * creation of the Door of period 1
 */

 DoorPeriod1-->  myText.frontdoor1

 /*
  * Back Door period 1
  */
backDoorPeriod1-->myText.backDoorPeriod1



//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\
////\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\////


/*
 * windows period 2
 */


period2(wallType,windowType,side,houseType) -->
      split(x) {
       scope.sx/2-0.75: /*myText.wall*/  waterexit(side,houseType)|
       1.3 : WindowPart2(windowType)              |
       ~1  : waterexit2(side,houseType)
        }

waterexit(side,houseType)-->
      case side==1:
            split(x){
                  0.2: myText.wall |
                  0.001:  WaterExitObject(houseType)  |
                  ~1: myText.wall}
      else:
            insertLight


insertLight-->
      split(x){
            scope.sx/2: myText.wall |
            0.01: lightY |
            ~1: myText.wall}

lightY-->
      split(y){
            2.7: myText.wall |
            0.01: lightObject |
            ~1:myText.wall}


lightObject-->
```

```
            s(0.5,0.5,0)
            r(0,-90,0)
            t(0,-0.1,-0.1)
            i("models/light/lights.obj")


waterexit2(side,houseType)-->
      case side==2:
            split(x){
                  ~1: myText.wall |
                  0.001:  WaterExitObject(houseType)   |
                  0.2: myText.wall}
      else:
            insertLight


WaterExitObject(houseType)-->
      case houseType==0:
            s(scope.sx+0.2,'1,'1)
            r(0,-90,0)
            i("models/waterexit/waterexit.obj")
      else:
            s(scope.sx+0.2,scope.sy*2+0.25,'1)
            r(0,-90,0)
            i("models/waterexit/waterexit.obj")


WindowPart2(windowType)      -->
      split(y) {
             ~1   : myText.wall    |
              2   : Window2FrameX(windowType)     |
             ~1   : myText.wall
            }

Window2FrameX(windowType)  -->
      split(x){
            0.1:Window2Frame |
            ~1 : Window2FrameY(windowType) |
            0.1:Window2Frame}


Window2Frame   --> myText.windowFrameWall//color(brown)nil.



Window2FrameY(windowType)  -->
      split(y){
            0.1:Window2Frame |
            ~1 : window(windowType)       |
            0.1:Window2Frame}



window(windowType)      -->
      split(x){
            0.0001: color(mov)window2Wall |
            ~1: window2Y(windowType) |
            0.001:color(mov)window2Wall}
```

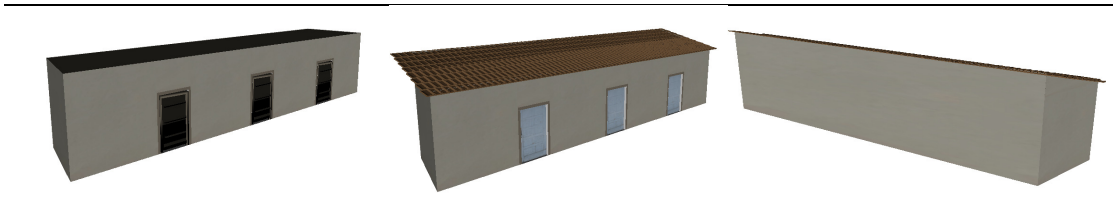
```
window2Y(windowType)-->
      split(y){
            0.0001:window2WallTop |
            ~1 : window2I(windowType)  |
            0.0001:window2WallTop }


window2Wall             -->
      r(0,90,0)
      s(0.15,'1,'1)
      myText.windowFrameWall


window2WallTop          -->
      r(-90,0,0)s('1,0.15,'1)color(mov) myText.windowFrameWall


window2I(windowType)--> t(0,0,-0.15) myText.windows(windowType)


/*
 * Door period 2
 */
DoorPeriod2-->
      split(x){
            0.0001: color(mov)Door2Wall |
            ~1: Door2Y |
            0.001:color(mov)Door2Wall}


Door2Y                    -->
      split(y){
            ~1 : Door2I  |
            0.0001:color(mov)Door2WallTop
            }


Door2Wall-->r(0,90,0) s(0.15,'1,'1)

Door2WallTop-->r(-90,0,0)s('1,0.15,'1)

Door2I -->      t(0,0,-0.15) DoorPeriod2Texture



DoorPeriod2Texture-->
      50%:
            myText.frontdoor3 center(x)
      else:
            myText.frontdoor2 center(x)

DoorPeriod2Model-->myText.frontdoor2
/*
 *
 */
backDoorPeriod2--> t(0,0,-0.16)myText.backDoorPeriod2


//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\
```

```
////\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\

backFacade(type,archBoolean) -->
      case scope.sx<5:
            myText.wall
      else:
            split(x){
                  '0.4 : LBackFacade    |
                  ~1   : singleArchBool(archBoolean)     |
                  '0.4 : RBackFacade(type)
                  }

LBackFacade-->  myText.wall

singleArchBool(archBoolean)-->
      case archBoolean==1:
            singleArch
      else:
            myText.wall


singleArch -->
      split(y) {
            2.78          : arches.kamara2(scope.sx+0.2) |
            ~1        : myText.wall
            }


RBackFacade(type)-->
      split(x){
            scope.sx/2-0.7  : myText.wall     |
            1.40            : smallDoorAreaY(type) |
            ~1              : myText.wall
            }


smallDoorAreaY(type)-->
      split(y) {
            1.9: smallDoorAreaX(type) |
            0.1: DoorFrame  |
            1: myText.wall |
            0.9: backWindow(type) |
            ~1 :myText.wall}


backWindow(type)-->
       case type==1:
            split(x) {
                  ~1  : myText.wall    |
                  0.7: bWindowFrameX |
                  ~1: myText.wall }
       else:
            myText.wall


smallDoorAreaX(type) -->
      split(x) {
            0.1: DoorFrame      |
            1.2: smallDoor(type)  |
```

```
            0.1: DoorFrame}


DoorFrame      --> myText.wall


bWindowFrameX  -->
      split(x){
            0.05:WindowFrame |
            0.6 : bWindowFrameY |
            ~1:WindowFrame
            }


bWindowFrameY  -->
      split(y){
            0.05:WindowFrame |
            0.8 : backFaceWindow       |
            ~1:WindowFrame}


backFaceWindow -->
      extrude(z,0.1)bWindowTile

bWindowTile     -->
      comp(f){
            front: bWindowLot |
            back: NIL/*color("#fff000")windowTex.*/ |
            all: color(brown)Sides. }


bWindowLot     -->
      color(black)
      center(x) s(0.8,'1,-0.05)
      i("builtin:cube")

      split(y){0.1 : bHBars}*
      split(x){0.1 : bVBar}*

bVBar -->
      s(0.02,'1,'1)
      t(0,0,0.05)VBar.
      t(0.1,0,0)VBar.



bHBars -->
      s('1,0.02,-0.05)
      t(0,0.1,0.05) HBar.
      t(0,0.1,0) HBar.



smallDoor(type)-->
      split(y){
            ~1 :smallDoorX(type) |
            0.0001:color(mov)smallDoorWallTop
            }

smallDoorX(type)-->
```

```
	split(x){
		0.0001: color(mov)smallDoorWall |
		~1: sDoorI(type) |
		0.001:color(mov)smallDoorWall
		}


smallDoorWall-->
	r(0,90,0)
	s(0.1,'1,'1)

smallDoorWallTop-->
	r(-90,0,0)
	s('1,0.1,'1)

sDoorI(period) -->
	t(0,0,0.06)
	chooseBackDoor(period)

chooseBackDoor(period)-->
	case period==1:
		backDoorPeriod1
	else:
		backDoorPeriod2
```

Αρχείο κανόνων για κτίσιμο των δώματων

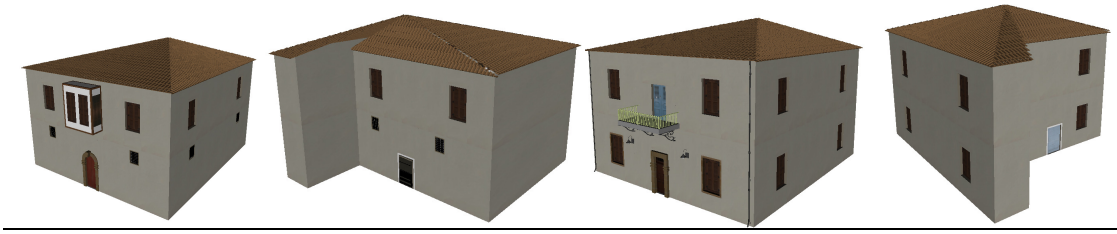


```
/**
 * File:    BackHouse.cga
 * Created: 24 Apr 2009 10:23:56 GMT
 * Author:  Savvas Michael
 */

############################
###   IMPORTS            ###
############################

import myText  : "myTextures.cga"
import roofType: "roof.cga"


############################
###   COLORS             ###
############################

mov = "#837a6b"
```

```
############################
###   RULES               ###
############################

dummySecondHouse-->secondHouse(2)

secondHouse(period) --> extrude(3)Dwmata(period)

Dwmata(period)          -->
      comp(f) {
            top:  color("#837a6b")dwmataTop(period,scope.sy,scope.sx)
            |bottom: dwmataBottom.
            |front : dwmataFront(period)
            |back  : dwmataBack(period)
            |right  : dwmataLeft
            |left : dwmataLeft
            }


dwmataBack(period) -->
      case period==1:
            dwmataLeft
      else:
            s('1,scope.sy+(tan(14)*(scope.sy)),'1)
            dwmataLeft

dwmataLeft-->myText.wall

dwmataFront(period)-->  split(x){4.4:dwmatio(period)}*

dwmatio(period)-->
      case scope.sx >= 4.40 :
            split(x){3:myText.wall | ~1: smallDoorAreaY(period)}
      else: myText.wall


smallDoorAreaY(period)-->
      split(y) {
            1.9: smallDoorAreaX(period) |
            0.1: DoorFrame      |
            ~1 :myText.wall}




smallDoorAreaX(period) -->
      split(x) {
            0.1: DoorFrame      |
            1.2: smallDoor(period)  |
            0.1: DoorFrame}



DoorFrame-->color(mov)


smallDoor(period)-->
      split(y){
            ~1 :smallDoorX(period) |
```

```
            0.0001:smallDoorWallTop
            }


smallDoorX(period)-->
      split(x){
            0.0001: smallDoorWall |
            ~1: sDoorI(period) |
            0.001: smallDoorWall
            }

smallDoorWall-->
      r(0,90,0)
      s(0.1,'1,'1)

smallDoorWallTop-->
      r(-90,0,0)
      s('1,0.1,'1)


sDoorI(period) -->
      t(0,0,-0.1)
      myText.backDoor(period)


dwmataTop(period,scope,mikos)-->
      case period==1:
            nil.
      else:
            roof(gable, 14,1) roofType.singleRoof(scope,mikos)
```

Αρχείο κανόνων για κτίσιμο του δίπατου σπιτιού



```
/**
 * File:    Dipato.cga
 * Created: 25 Apr 2009 15:35:12 GMT
 * Author:  Savvas Michael
 */

############################
###   IMPORTS            ###
############################

import roofType: "roof.cga"
import house   : "FrontHouse.cga"
import textures: "myTextures.cga"
```

```
############################
###   COLORS               ###
############################

brown="#7D7159"
black="#000000"
mov = "#837a6b"
wood="#69472b"
white = "#ffffff"


############################
###   RULES                ###
############################

//dummy rule
DipatoHouse --> Dipato(2)

Dipato(period) --> extrude(8) chooseWindow(period)

chooseWindow(period)    -->
      33% : dipatoBuilding(period,1)
      33% : dipatoBuilding(period,2)
      else: dipatoBuilding(period,3)


dipatoBuilding(period,windowType)    -->
      split(y){
            4 : firstFloor(period,windowType)  |
            ~1: secondFloor(period,windowType)
            }


firstFloor(period,windowType) -->
      case period==1:
            firstFloorPeriod1(windowType)
      else:
            firstFloorPeriod2(windowType)


firstFloorPeriod1(windowType)-->
      comp(f){
            front : house.frontFacade(1,1,windowType,1) |
            right : SideWithWindows(1,windowType)|
            5     : backFaceFirstFloor(1)|
            side  : textures.wall     |
            bottom: nil.
             }


firstFloorPeriod2(windowType)-->
        comp(f){
            front : house.frontFacade(2,1,windowType,1) |
                        //1-front face , 0 - no door
            right : SideWithwindowType2(2,1,0,windowType)|
                        //2-back face  , 1 - door
            5     : SideWithwindowType2(2,2,1,windowType)|
            side  : textures.wall     |
            bottom: nil.
```

```
            }


secondFloor(period,windowType) -->
      comp(f){
            1  : SideWithwindowType2(period,0,0,windowType)   | //0 -
front face
            2  : SideWithwindowType2(period,1,0,windowType)   | //1 -
right face
            5  : SideWithwindowType2(period,2,0,windowType)   | //2 -
back face
            side: textures.wall    |
            top   : roof(pyramid, 14) roofType.myRoof |
            bottom: nil.
            }


//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\
////\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//
/*
 * Rules applied on some sides of the house for creating 2 small
windows
 */

SideWithWindows(type,windowType) -->
      split(x){
            '0.4 : LRFrontFacade(type) |
            ~1   : choosekiosk(1,type,windowType)   |
            '0.4 : LRFrontFacade(type)
             }

choosekiosk(period,type,windowType)-->
      case type==0 && period==1:
            createkiosk(windowType)
      case type==0 && period==2:
            createBalconi
      else : textures.wall


LRFrontFacade(type)-->
      split(x) {
            scope.sx/2-0.35: textures.wall |
            0.7 : WindowPart(type)              |
            ~1  : textures.wall
              }


WindowPart(type)    -->
      split(y) {
              2.95  : textures.wall   |
              0.9   : WindowFrameX(type) |
              ~1    : textures.wall
            }


WindowFrameX(type)  -->
      split(x){
            0.05:WindowFrame |
            0.6 : WindowFrameY(type) |
            ~1:WindowFrame}
```

```
WindowFrameY(type)  -->
      split(y){
            0.05:WindowFrame |
            0.8 : Window(type)       |
            ~1:WindowFrame
            }


Window(type)        -->
      case type==1 || type==0:
            extrude(z,-0.1)WindowTile(type)
      else:
            extrude(z, 0.1)WindowTile(type)


WindowTile(type)    -->
      case type==1 || type==0:
            alignScopeToGeometry(yUp, auto)
            comp(f){
                  front: WindowLot |
                  back: NIL/*color("#fff000")windowTex.*/ |
                  all: color(brown)Sides.
                  }
      else:
            comp(f){
                  front: t(0,0,-0.01)WindowLot |
                  back: NIL/*color("#fff000")windowTex.*/ |
                  all: color(brown)Sides.
                  }


WindowFrame   --> color(brown)nil.
WindowLot     -->
      color(black)
      center(x)
      s(0.8,'1,-0.05)
      i("builtin:cube")


      split(y){0.1 : HBars}*
      split(x){0.1 : VBar}*



VBar -->
      s(0.02,'1,'1)
      t(0,0,0.1)VBar.
      t(0.1,0,0)VBar.

HBars -->
      s('1,0.02,-0.05)
      t(0,0.1,0.1) HBar.
      t(0,0.1,0) HBar.


//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\
///\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//
/*
```

```
 * Rules applied on some sides of the house for creating 2 windows of
type2
 */
SideWithwindowType2(period,type,doorBoolean,windowType)-->
      case doorBoolean==0:
            middleWithOutDoor(period,type,windowType)
      else:
            middleWithDoor(type,windowType)


middleWithOutDoor(period,type,windowType)-->
      split(x){
            '0.4 : LRSideFacade(type,windowType) |
            ~1   : choosekiosk(period,type,windowType)  |
            '0.4 : LRSideFacade(type,windowType)
             }


middleWithDoor(type,windowType)-->
      split(x){
            ~1      : textures.wall |
            1.40  : smallDoorAreaY(2)|
            '0.4  : LRSideFacade(type,windowType)
             }


LRSideFacade(type,windowType)-->
      split(x) {
            scope.sx/2-0.75: textures.wall |
            1.3 : WindowPart2(windowType)  |
            ~1  : textures.wall
            }


WindowPart2(windowType)      -->
      split(y) {
              ~1  : textures.wall   |
               2  : Window2FrameX(windowType)    |
              ~1  : textures.wall
            }







Window2FrameX(windowType)  -->
      split(x){
            0.1:Window2Frame |
            ~1 : Window2FrameY(windowType) |
            0.1:Window2Frame
            }

Window2Frame   --> textures.wall //color(brown) nil.


Window2FrameY(windowType)  -->
      split(y){
            0.1:Window2Frame |
            ~1 : window(windowType)    |
```

```
            0.1:Window2Frame}


window(windowType)      -->
      split(x){
            0.0001: color(mov)window2Wall |
            ~1: window2Y(windowType) |
            0.001:color(mov)window2Wall
            }


window2Y(windowType)-->
      split(y){
            0.0001:window2WallTop |
            ~1 : window2I(windowType)  |
            0.0001:window2WallTop
            }


window2Wall             -->   r(0,90,0) s(0.15,'1,'1)


window2WallTop          -->   r(-90,0,0)s('1,0.15,'1)color(mov)
window2I(windowType)--> t(0,0,-0.15) textures.windows(windowType)

//#########################################################################
//#########################################################################

/*
 * extra rules for the back face of the ground floor of the house:
dipato
 */
backFaceFirstFloor(period)-->
      split(x){
            1.4 : kamara  |
            ~1  : bFace(period)
             }

kamara-->   textures.wall


bFace(period)-->
      split(x){
            0.7 : WindowPart(2) |
            4.5 : bFaceDoor(period) |
            0.7 : WindowPart(2) |
            ~1  : textures.wall
            }


bFaceDoor(period)-->
       split(x){
            scope.sx/2-0.7  : textures.wall     |
            1.40            : smallDoorAreaY(period) |
            ~1              : textures.wall
            }



smallDoorAreaY(period)-->
```

```
      split(y) {
            1.9: smallDoorAreaX(period) |
            0.1: DoorFrame  |
            ~1 :textures.wall
            }

smallDoorAreaX(period) -->
      split(x) {
            0.1: DoorFrame      |
            1.2: smallDoor(period)  |
            0.1: DoorFrame
            }

DoorFrame --> nada. //color(mov)

smallDoor(period)-->
       split(y){
            0.0001:color(mov)smallDoorWallTop |
            ~1 :smallDoorX(period) |
            0.0001:color(mov)smallDoorWallTop
            }


smallDoorX(period)-->
      split(x){
            0.0001: color(mov)smallDoorWall |
            ~1: sDoorI(period) |
            0.001:color(mov) smallDoorWall
            }


smallDoorWall-->
      r(0,90,0)
      s(0.1,'1,'1)


smallDoorWallTop-->
      color(mov)
      r(-90,0,0)
      s('1,0.1,'1)


sDoorI(period) -->
      t(0,0,-0.1)
      textures.backDoor(period)

//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\
////\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//\\//
/*
*  Period 1 : create kiosk
*/

createkiosk(windowsType)-->
      split(y){
            0.6: textures.wall |
            3  : kiosk(windowsType)         |
            ~1 : textures.wall
            }
```

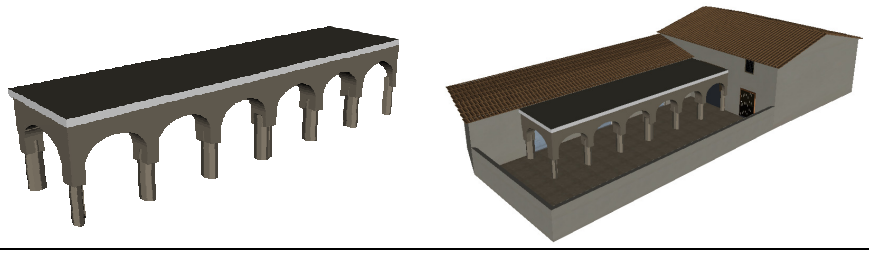
```
kiosk(windowsType)-->
      split(y){
            0.001: kioskBottom |
            ~1 :kioskSides(windowsType) |
            0.001:kioskBottom
            }


//kiosk side windows
kioskBottom-->    r(90,0,0)s('1,0.9,'1)kioskSideMod(1)

kioskSides(windowsType)-->
      split(x){
            0.001: kioskSide |
            ~1:kioskFront(windowsType) |
            0.001:kioskSide
            }


kioskSide-->
      r(0,-90,0)
      s(0.9,'1,'1)
      kioskSideMod(2)


kioskSideMod(type)-->
      split(x){
            0.1: color(wood)nil. |
            ~1: kioskSideSplit(type) |
            0.1:color(wood)nil.
            }


kioskSideSplit(type)-->
      split(y){
            0.1: color(wood)nil. |
            ~1: kioskSideWindow(type)|
            0.1:color(wood)nil.
            }









kioskSideWindow(type)-->
      case type==2:
            split(y){
                  0.4: color(white)nil. |
                  ~1: textures.kioskSideWindowTex |
                  0.4: color(white)nil.
                  }
      else:
            color(white)nil.


//kiosk front windows
kioskFront(windowsType)-->
      t(0,0,0.9) kioskFrontModify(windowsType)
```

```
kioskFrontModify(windowsType)-->
      split(x){
            0.1: color(wood)nil.
           | ~1 : split(y){
                        0.1:color(wood)nil. |
                        ~1:SplitKioskFront(windowsType) |
                        0.1:color(wood)nil.}
            | 0.1: color(wood)nil.
      }


SplitKioskFront(windowsType)-->
   split(y){
      0.4: emptySpace. |
      ~1: KioskFrontWindows(windowsType) |
      0.4:nil.
      }


KioskFrontWindows(windowsType)-->
      split(x){
            0.2:emptySpace. |
            ~1: textures.windows(windowsType) |
            0.3:emptySpace.|
            ~1: textures.windows(windowsType)|
            0.2:emptySpace.
            }


/*
*  Period 2 : create balconi
*/
createBalconi--> 50%:
                        BalconiType1
                  else:
                        BalconiType2

//************************

BalconiType1-->
      split(y){
            1:balconiObject |
            ~1:balconiDoorType1
            }
balconiObject -->
      r(0,-90,0)
      t(0,-0.5,-2.8)
      s(1.5,1.5,'1)
      i("balconi.obj")

balconiDoorType1-->
      split(y){
            0.3:door2Bottom |
            1:DoorType1 |
            ~1: textures.wall
            }
```

```
door2Bottom-->
       t(0,-1,0)textures.wall


DoorType1-->
      s('1,scope.sy+1,'1)
      t(0,-1,0)
      DoorType1Split


DoorType1Split-->
      split(x){
            ~1:textures.wall |
            1.1:doorBalconi |
            ~1:textures.wall
            }

doorBalconi -->
      split(x){
            0.0001: color(mov)balconi2Wall |
            ~1: balconi2Y |
            0.001:color(mov)balconi2Wall
            }


balconi2Y   -->
      split(y){
            0.0001:balconi2WallTop |
            ~1 : balconi2I  |
            0.0001: balconi2WallTop
            }


balconi2Wall-->
      r(0,90,0)
      s(0.15,'1,'1)


balconi2WallTop-->
      r(-90,0,0)
      s('1,0.15,'1)


balconi2I -->
      t(0,0,-0.15)
      textures.balconiDoor
//************************

BalconiType2-->
      split(y){
            0.001:  balconiBottom |
            ~1 :balconiDoorType2
            }


balconiBottom-->
      r(90,0,0)
      s(scope.sx*1.5,1.5,'1)
      t(-scope.sx/6.5,0,0)
      textures.wall balconiFence
```

```
balconiFence-->
      extrude(-1.0)
      comp(f){
            front : balconiFaceFence. |
            right : balconiFaceFence |
            left : balconiFaceFence
            }


balconiFaceFence -->
      s('1,'1,-0.1)
      i("builtin:cube")
      color(white)


balconiDoorType2-->
      split(x){
            scope.sx/2-0.55: textures.wall |
            1.1            : balconiDoorY  |
            ~1             : textures.wall
            }


balconiDoorY-->
      split(y){
            2.2:balconiDoorI |
            ~1 :textures.wall
            }


balconiDoorI-->textures.balconiDoor
```

Αρχείο κανόνων για κτίσιμο των καμάρων



```
/**
 * File:    arches.cga
 * Created: 24 Apr 2009 10:31:18 GMT
 * Author:  Savvas Michael
 */
```

```
############################
###   COLORS             ###
############################

mov = "#837a6b"
white = "#ffffff"



############################
###   RULES              ###
############################

Kamara--> myArch(1)

myArch(type)    --> extrude(2.8) archBuild(type)

archBuild(type) -->
      case type==1:
            arch1
      else:
            arch2


arch1 -->
      comp(f) {
            top    : color(mov)  topFacade.   |
            back   : backFacade  |
            front  : frontFacade |
            right  : rightFacade |
            left   : leftFacade  |
            bottom: NIL
            }


arch2-->
      comp(f) {
            top    : color(mov) topFacade.   |
            back   : backFacade2  |
            front  : frontFacade |
            right  : rightFacade |
            left   : leftFacade  |
            bottom: bottomFacade.
        }


backFacade2 -->
      setupUV(0,scope.sx,scope.sy)
      set(material.colormap, "whiteWall.JPG")
      bakeUV(0)
      color(white)


backFacade  --> NIL


rightFacade --> split(y){~1: NIL | 0.2: ledge.}


leftFacade  --> split(y){~1:kamara1|0.2:ledge.}
```

```
kamara1      -->
      s(0.6,'1,scope.sx-0.1)
      r(0,90,0)
      t(0.05,0,0)
      i("models/Arch/singleArchTexture.obj")
      color(mov)

kamara2(xValue)-->
      s(0.6,scope.sy+0.05,xValue)
      r(0,90,0)
      t(-0.05,0,-0.15)
      i("models/Arch/singleArchTexture.obj")
      color(white)


frontFacade -->
      split(y){
            ~1: frontKamares |
            0.2:ledge.
            }


frontKamares-->
      case scope.sx < 8:
            kamares((scope.sx/10)+0.05)
      else:
            split(x){6.4: kamares(0.7) | ~1:frontKamares}


kamares(size)-->
      s(size,'1,'1)
      r(0,90,0)
      t(0,0,0)
      i("arch.obj")
      color(mov)
```

Αρχείο κανόνων για κτίσιμο των οροφών

```
/**
 * File:    roof.cga
```

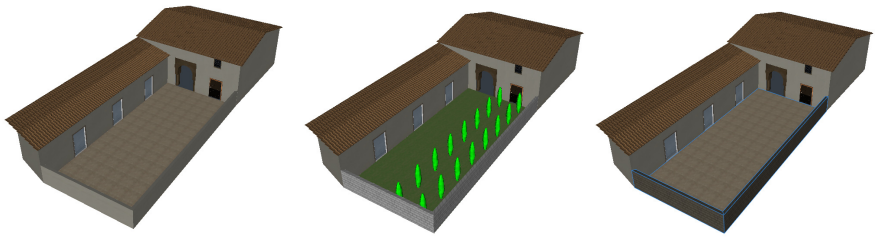
```
 * Created: 23 Apr 2009 12:37:34 GMT
 * Author:  Savvas Michael
 */


############################
###   ATTRIBUTES         ###
############################

attr spacing            = 4.29
attr roofBrickW   = spacing*0.05//0.25
attr roofBrickH   = spacing*0.1//0.35
attr roofbrickC         = "#886644"

attr roofBrickBottomAsset = "temples/roofbrick_bottom.obj"
attr roofBrickRound = "temples/roofbrick_round.obj"


############################
###   COLORS             ###
############################

mov = "#837a6b"
red = "#5c3f40"
white = "#ffffff"



//=========================================================================
//============================   ROOF   ===================================
//=========================================================================

myRoof-->
      set(trim.horizontal,true)
      comp(f){
            bottom : Ceiling |
            vertical : Pediment |
            all : RoofPlane |
            top: color(mov) topRoof. }

Ceiling --> NIL

Pediment -->
      setupUV(0,scope.sx,scope.sy)
      set(material.colormap, "whiteWall.JPG")
      bakeUV(0)
      color(white)


RoofPlane -->
      t(0,-spacing*0.05,0)
      s(scope.sx+spacing*0.1 , scope.sy+spacing*0.05, '1) center(x)
      RoofBricks


RoofBricks -->
      t(0,0,-0.05)
      split(y){ roofBrickH : BottomBrickRow }*
      split(y){
            roofBrickH*0.8 : TopBrickRow |
```

```
            { roofBrickH : TopBrickRow }*
      }


BottomBrickRow -->
      color(roofbrickC)
      split(x){ ~roofBrickW : BottomBrick }*


BottomBrick --> s('1,'1,'1) i(roofBrickBottomAsset)


TopBrickRow -->
      color(roofbrickC)
      s('1,scope.sy+roofBrickH*0.15,'1)
      t(0,0,roofBrickW*0.22)
      split(x){
            ~roofBrickW*0.8 : NIL |
            { ~roofBrickW*0.4 : TopBrick | ~roofBrickW*0.6 : NIL }*
            | ~roofBrickW*0.4 : TopBrick |
            ~roofBrickW*0.8 : NIL }


TopBrick --> set(trim.horizontal,true) i(roofBrickRound)



//=====================================================================
//============================  SINGLE ROOF  ==========================
//=====================================================================

dummySingleRoof--> singleRoof(1,1)

singleRoof(zscope,mikos)-->  //set(trim.horizontal,true)
      comp(f){
            bottom : bCeiling. |
            4 : bPediment1(scope.sx,mikos) |
            7 : bPediment2| left:backRoofPlane |
            1:lbRoofPlane(zscope)  |
            top: topRoof.
            }


bPediment1(pedimentX,mikos) -->
      split(x){
            '0.5: NIL |
            0.001: nada. |
            ~1: Pediment1
            }


bPediment2 -->
      split(x){'0.5: Pediment2 | ~1: NIL}


backPediment(backYSize,mikos)-->
      t(-backYSize,0,0)
      r(0,86.5,0)
      s(mikos*3150,scope.sy*2,'1)
```

```
      t(mikos/2-0.1,0,0)


Pediment1-->
      t(-scope.sx,0,0)
      s(scope.sx*2,scope.sy*2,'1)
      setupUV(0,scope.sx,scope.sy)
      set(material.colormap, "whiteWall.JPG")
      bakeUV(0)
      color(white)


Pediment2-->
      s(scope.sx*2,scope.sy*2,'1)
      setupUV(0,scope.sx,scope.sy)
      set(material.colormap, "whiteWall.JPG")
      bakeUV(0)
      color(white)


backRoofPlane-->color(red)nada.


lbRoofPlane(zscope) -->
      color(red)
      t(-0.2,-0.25,0)
      s(scope.sx+0.5,zscope+0.5,'1)
      bRoofBricks


bRoofBricks -->
      t(0,0,-0.05)
      split(y){ roofBrickH : bBottomBrickRow }*
      split(y){
            roofBrickH*0.8 : bTopBrickRow |
            { roofBrickH : bTopBrickRow }*
            }


bBottomBrickRow -->
      color(roofbrickC)
      split(x){ ~roofBrickW : BottomBrick }*


bBottomBrick --> s('1,'1,'1) i(roofBrickBottomAsset)


bTopBrickRow -->
      color(roofbrickC)
      s('1,scope.sy+roofBrickH*0.15,'1)
      t(0,0,roofBrickW*0.22)

      split(x){
            ~roofBrickW*0.8 : NIL |
            { ~roofBrickW*0.4 : bTopBrick | ~roofBrickW*0.6 : NIL }* |
            ~roofBrickW*0.4 : bTopBrick | ~roofBrickW*0.8 : NIL }


bTopBrick -->
      set(trim.horizontal,true)
```

```
      i(roofBrickRound)
```

Αρχείο κανόνων για κτίσιμο των αυλών



```
/**
 * File:    yard.cga
 * Created: 24 Apr 2009 10:28:59 GMT
 * Author:  Savvas Michael
 */


##############################
###   FUNCTIONS            ###
##############################

getYardType=
      50%:
            1
      else :
            2

##############################
###   COLORS               ###
##############################

mov = "#837a6b"
green="#00ff00"



##############################
###   RULES                ###
##############################

dummyYard--> yard(1)

yard(type)--> chooseFence(type)

chooseFence(type)-->
      33%: extrudeYard(type,1,getYardType)
      33%: extrudeYard(type,2,getYardType)
```

```
      else:extrudeYard(type,3,getYardType)


extrudeYard(type,fenceType,yardType)-->
      case type==1 :
            extrude(1.5)
            comp(f){
                  left : FaceFence(fenceType) |
                  front: FaceFence(fenceType) |
                  bottom: yardOnly(yardType) yardTree(1,yardType)
            }
     case type==3:
            extrude(1.5)
            comp(f){
                  side: FaceFence(fenceType) |
                  bottom: yardOnly(yardType) yardTree(1,yardType)
                  }
      else:
            extrude(1.5)
            comp(f){
                  left : FaceFence(fenceType) |
                  front: FaceFence(fenceType) |
                  back: FaceFence(fenceType)  |
                  bottom: yardOnly(yardType) yardTree(2,yardType)
           }


FaceFence(fenceType)-->
      s('1,'1,-0.2)
      i("builtin:cube")
      yardWallTexture(fenceType)


yardOnly(yardType)-->
      case yardType==1:
            setupUV(0 ,1,1)
            set(material.colormap, "yard.jpg")
            bakeUV(0)
      else:
            setupUV(0 ,1,1)
            set(material.colormap, "grass.png")
            bakeUV(0)



yardTree(size,yardType)-->
      33%:  color(green)
            i("tree2.obj")
            r(-90,0,0)
            s(scope.sx/2,scope.sy*0.6,scope.sz/3)
            t(scope.sx*0.6,0,0)
      33%:  bush(size,yardType)
      else: NIL


bush(size,yardType)    -->
      split(y){
            (scope.sy*size)/2   :
                  split(x){
                        1:yardOnly(yardType) |
```

```
                        ~1:treex1(yardType)
                        }*
      |~1       : yardOnly(yardType)
      }


treex1(yardType)  -->
      split(y){
            1:yardOnly(yardType) | ~1:treez}*

treez   --> color(green)
            center(xy)
            i("tree.obj")
            r(-90,0,0)
            s(scope.sx/2,scope.sy/0.4,scope.sz/2)


yardWall--> extrude(1.5) yardWallTexture

yardWallTexture(fenceType)-->
      case fenceType==1:
            setupUV(0,1.2,1)
            set(material.colormap, "brickwall2.tif")
            bakeUV(0)
            color(mov)
      case fenceType==2:
            setupUV(0,1.2,1)
            set(material.colormap, "brickwall2.tif")
            bakeUV(0)
      else:
            setupUV(0,1.2,1)
            set(material.colormap, "whiteWall.JPG")
            bakeUV(0)
```

Αρχείο κανόνων για τα textures

```
/**
 * File:    myTextures.cga
 * Created: 24 Apr 2009 10:42:25 GMT
 * Author:  Savvas Michael
 */


##############################
###   COLORS               ###
##############################

wallColor = "#caa27f"
white = "#ffffff"
mov = "#837a6b"


##############################
###   RULES                ###
##############################
```

```
wall2 -->
      setupUV(0,1.2,1)
      set(material.colormap, "brickwall2.tif")
      bakeUV(0)
      color(wallColor)

wall3-->
      setupUV(0,1,1)
      set(material.colormap, "NewAssets/brickWall.JPG")
      bakeUV(0)
      color(wallColor)

wall4-->
      setupUV(0,scope.sx,scope.sy)
      set(material.colormap, "NewAssets/brickswall.jpg")
      bakeUV(0)
      color(wallColor)

wall1 -->
      setupUV(0,1.4,0.8)
      set(material.colormap, "brickwall2.tif")
      bakeUV(0)
      color(wallColor)

wall-->
      setupUV(0,scope.sx,scope.sy)
      set(material.colormap, "whiteWall.JPG")
      bakeUV(0)
      color(white)

/*************************************************************/

dummyDoor-->backDoor(1)

backDoor(period)-->
      case period==1:
            backHouseDoorPeriod1
      else:
            backDoorPeriod2


backHouseDoorPeriod1-->
      r(0,90,0)
      s(0.2,scope.sy,scope.sx)
      i("backdoorAll.obj")
      color(mov)
      t(-0.2,0,0)


backDoorPeriod1-->
      r(0,90,0)
      s(0.2,scope.sy,scope.sx)
      i("Backdoor.obj")
      color(mov)


backDoorPeriod2-->
      setupUV(0,1.2,1.9)
      set(material.colormap, "backdoor.png")
```

```
      bakeUV(0)

/*************************************************************/

windowFrameWall-->
      setupUV(0,scope.sx,scope.sy)
      set(material.colormap, "innerWall.jpg")
      bakeUV(0)

/*************************************************************/
balconiDoor-->
      setupUV(0,1.2,2.6)
      set(material.colormap, "frontdoor.PNG")
      bakeUV(0)

frontdoor2-->
      10%:
            setupUV(0,1.2,2.6)
            set(material.colormap, "frontdoor.PNG")
            bakeUV(0)
       40%:
            i("models/door/frontdoor2.obj")
            t(-0.1,0,-0.05)
            s(1.4,'1,'1)
      else:
            i("models/door/frontdoor1.obj")
            t(-0.1,0,-0.05)
            s(1.4,'1,'1)


frontdoor3-->
      10%:
            setupUV(0,scope.sx,scope.sy)
            set(material.colormap, "frontdoor1.jpg")
            bakeUV(0)
      40%:
            i("models/door/frontdoor4.obj")
            t(-0.1,0,-0.05)
            s(1.4,'1,'1)
      else:
            i("models/door/frontdoor3.obj")
            t(-0.1,0,-0.05)
            s(1.4,'1,'1)


frontdoor1-->
      25%:
            i("models/door/door4texture.obj")
            t(-0.15,0,-0.25)
            s(1.5,2.8,'1)
      25%:
            i("models/door/door3texture.obj")
            t(-0.15,0,-0.25)
            s(1.5,2.88,'1)
      25%:
            i("models/door/door4_2texture.obj")
            t(-0.15,0,-0.25)
            s(1.5,2.8,'1)
      else:
            i("models/door/door3_1texture.obj")
```

```
            t(-0.15,0,-0.25)
            s(1.5,2.88,'1)

/*************************************************************/

dummyWindows-->windows(1)

windows(windowType)-->
      case windowType==1:
            setupUV(0,scope.sx,scope.sy)
            set(material.colormap, "window1.jpg")
            bakeUV(0)
      case windowType==2:
            setupUV(0,scope.sx,scope.sy)
            set(material.colormap, "window2.jpg")
            bakeUV(0)
      case windowType==3:
            setupUV(0,scope.sx,scope.sy)
            set(material.colormap, "window3.jpg")
            bakeUV(0)
      else:
            setupUV(0,scope.sx,scope.sy)
            i("windowModel.obj")
            bakeUV(0)


kioskSideWindowTex-->
      setupUV(0,scope.sx,scope.sy)
      set(material.colormap, "window1.jpg")
      bakeUV(0)


frontkioskWindow-->
      setupUV(0,scope.sx,scope.sy)
      set(material.colormap, "windows/window1.jpg")
      bakeUV(0)
      color(white)
```

Αρχείο κανόνων για τους δρόμους (2ης περιόδου)

```
/**
 * File:    streetsp2.cga
 * Created: 8 May 2009 10:14:41 GMT
 * Author:  Savvas Michael
 */
```

```
####################
##### Textures #####
####################

road1_tex = "streets/road1.jpg"
concrete_tex = "streets/pavement3.jpg"
curb_tex = "streets/curb.jpg"


####################
##### Assets ######
####################


lamp_texasset = "streets/modern/lamps/lamp.04.single.lod0.obj"
lampwithsign_texasset =
"streets/modern/lamps/lamp.05.withsign.lod0.obj"

#########################
##### User attributes  ###
#########################


attr sidewalkheight = 0.1
attr avgLanewidth = 5
attr avgLampdist = 5

attr street_color = "#666666"
attr sidewalk_color = "#999999"



########################
##### Functions  ########
########################




// Calculates the (rounded) number of lanes depending on the street
width
calcLanes(streetwidth) =
      streetwidth/ceil(streetwidth/avgLanewidth)


/********************
 *   Rules          *
 ********************/

Street -->
      // Set street color
      color(street_color)
      // A z-up coordinate system is needed to set the texture
      alignScopeToGeometry("zUp", 0)
      // The uv's are set to correctly scale the road texture on the
street polygon.
      // The texture is repeated every 2 meter along the street (x-
axis). The epetions along the y-axis defines the number of lanes
      setupUV(0,2,calcLanes(scope.sy))
      // Define what texture to use
```

```
      set(material.colormap, road1_tex)
      // Apply texture coordinates
      bakeUV(0)


Crossing -->
      color(street_color)
      alignScopeToGeometry("zUp", 0)
      setupUV(0, 3,3, 0)
      set(material.colormap, concrete_tex)
      bakeUV(0)



Sidewalk -->
      // Set sidewalk color
      color(sidewalk_color)
      // To ensure correct extrusion on non-horizontal streets the
current coordinate system is aligned to the global y axis
      alignScopeToAxes("y")
      // Extrude the sidewalk
      extrude(sidewalkheight)
      // No trimming is needed, so we disable both trim dimensions
      set(trim.horizontal, false) set(trim.vertical, false)
      // Get the different faces of the extruded sidewalk cube
      comp(f){ top      :
                  alignScopeToGeometry("zUp", 0)
                  SidewalkTopface // get top face of the sidewalk cube
            and apply lamp and topface rules
                | bottom : NIL // the bottom face is unnecessary and
will be dicarded
                | side       : alignScopeToGeometry("zUp", 0)
SidewalkSideface }      // get side faces of the cube



CrossingSidewalk -->
      color(sidewalk_color)
      alignScopeToAxes("y")
      extrude(sidewalkheight)
      set(trim.horizontal, false) set(trim.vertical, false)
      comp(f){
            top    : SidewalkTopface |
            bottom      : NIL |
            side   : SidewalkSideface }



SidewalkTopface -->
      // Setup texture coordinates to repeat the texture every 3
meters in x and once in y
      setupUV(0,1,scope.sy)
      // Define what texture to use
      set(material.colormap, concrete_tex)
      // Apply texture coordinates
      bakeUV(0)

SidewalkSideface -->
      setupUV(0,1,scope.sy)
      set(material.colormap, curb_tex)
```