ΑΛΕΞΑΝΔΡΕΙΟ Τ.Ε.Ι. ΘΕΣΣΑΛΟΝΙΚΗΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Procedural Modeling of a Town based on Shape Grammar Rules

<<Εικόνα>>

Του φοιτητή

Νικόλαου Μωραΐτη

Αρ. Μητρώου: 05/2859

Επιβλέπων καθηγητής

…….

Θεσσαλονίκη 2013

ΠΡΟΛΟΓΟΣ

ΠΕΡΙΛΗΨΗ

Αυτή η πτυχιακή εργασία περιγράφει τη διαδικαστική μοντελοποίηση μιας πόλης, και πιο συγκεκριμένα τα παλαιά κτίρια του νησιού της Κέρκυρας ($18^{ο}$-$19^{ο}$ αιώνα), χρησιμοποιώντας κανόνες Shape Grammar Rules (SGR). Επίσης, γίνεται μια εκτενής αναφορά στη χρήση των κανόνων SGR. Αναπόσπαστο κομμάτι αυτής της διατριβής είναι η χρήση του εργαλείου μοντελοποίησης Unity3D που θα μας δώσει αποτελέσματα της τεχνικής SGR.

ABSTRACT

This thesis describes the procedural modeling of a town, more specifically the old buildings of the Ionian Corfu island (18^{th} -19^{th} century), by using Shape Grammar Rules (SGR) and thoroughly explains the use of SGR rules. An integral part of this thesis is the use of modeling tool Unity3D, which provides visual results of the SGR technique.

ΕΥΧΑΡΙΣΤΙΕΣ

Ευχαριστώ θερμά τον καθηγητή κο. Φώτη Λιαροκάπη (Senior Lecturer in Creative Computing Director of Interactive Worlds Applied Research Group Research Fellow, Serious Games Institute) για τη δυνατότητα που μου έδωσε κατά την άσκηση της πρακτικής μου εργασίας και που με σύστησε σε νέες τεχνικές όπως τα Shape grammar rules.

Επίσης, θα ήθελα να ευχαριστήσω οικογένεια και φίλους που με υποστήριξαν και μου συμπαραστάθηκαν για την επιτέλεση της πτυχιακής μου εργασίας και του καλού μου φίλου Martin Frost για την ατέλειωτη υποστήριξή του.

ΠΕΡΙΕΧΟΜΕΝΑ

Ευρετήριο σχημάτων

Ευρετήριο πινάκων

ΕΙΣΑΓΩΓΗ

Η μοντελοποίηση και η οπτική αναπαράσταση έργων που δημιούργησε ο άνθρωπος, όπως οι μεγάλες πόλεις, είναι μια μεγάλη πρόκληση για την επιστήμη των γραφικών υπολογιστών. Οι πόλεις είναι έργα υψηλής λειτουργικότητας και οπτικής πολυπλοκότητας, καθώς αντανακλούν τις ιστορικές, πολιτιστικές, οικονομικές και κοινωνικές αλλαγές κατά την πάροδο του χρόνου σε κάθε πτυχή από όπου κανείς τις παρατηρήσει. Εξετάζοντας αργότερα τις εικόνες του νησιού της Κέρκυρας, καθένας παρατηρεί την μοναδικότητα της αρχιτεκτονικής που παρουσιάζεται στα κτίρια της και που χαρακτηρίζει την βαθύπλουτη ιστορία του νησιού. Η μοντελοποίηση και η οπτική αναπαράσταση μιας μεγάλης περιοχής μιας πόλης χρησιμοποιώντας ηλεκτρονικούς υπολογιστές έχει γίνει εφικτή χάρις τη μεγάλη μνήμη, επεξεργασία και γραφικής ικανότητας του σημερινού υλικού εξοπλισμού υπολογιστών. Οι πιθανές εφαρμογές για μια διαδικαστική υλοποίηση έχουν ένα εύρος από ερευνητικούς και εκπαιδευτικούς σκοπούς όπως ο πολεοδομικός σχεδιασμός και η δημιουργία ενός οπτικού περιβάλλοντος προς προσομοίωση. Ειδικά στην αγορά ψυχαγωγίας όπως η βιομηχανία κινηματογράφου και παιχνιδιών έχουν υψηλή ζήτηση για τη γρήγορη δημιουργία σύνθετων περιβαλλόντων στις εφαρμογές τους.

Η οπτική μοντελοποίηση μεγάλων, πολύπλοκων συστημάτων έχει μια μακρά παράδοση στα γραφικά υπολογιστών. Οι περισσότερες από αυτές τις προσεγγίσεις απευθύνονται στην εμφάνιση φυσικών φαινομένων. Ένα μεγάλο μέρος γοητείας μιας ανάλογης απεικόνισης έγκειται στη δυνατότητα να απεικονίζουν την πολυπλοκότητα συστημάτων μεγάλης κλίμακας, τα οποία αποτελούνται από απλούστερα στοιχεία.

Μερικά από αυτά τα συστήματα περιλαμβάνουν: την προσομοίωση της διάβρωσης (the simulation of erosion) [F.K. Musgrave et. al., 1990], βασιζόμενης σε σωματίδια των δασών (particle based forests) [W.T. Reeves & R. Blau, 1985], και μοντελοποίηση σύννεφων (cloud modeling) [Ken Perlin, 1985].

Η Grammar-based γενιά μοντέλων (κυρίως του μοντέλου L-systems) χρησιμοποιούνται σε γραφικά υπολογιστών, κυρίως για τη δημιουργία γεωμετρίας των φυτών [P. Prusinkiewicz & A. Lindenmayer, 1990] όπως και η περίπτωση των split grammar rules για την απεικόνιση κτιρίων.

Η δημιουργία επιτακτικών μοντέλων είναι ένα κρίσιμο έργω για την ανάπτυξη επιτυχημένων ταινιών κινηματογράφου και ηλεκτρονικών παιχνιδιών. Ωστόσο, μοντελοποιώντας τεράστια τρισδιάστατα περιβάλλοντα, όπως πόλεις, είναι μια πολύ δαπανηρή διαδικασία και μπορεί να απαιτήσει πολλά χρόνια απασχόλησης εργατικού δυναμικού. Σε αυτή την πτυχιακή εργασία θα ασχοληθούμε με τη διαδικαστική μοντελοποίηση κτιρίων χρησιμοποιώντας γραμματικές σχημάτων (shape grammars) ικανά να κατασκευάσουν αποτελεσματικά ένα μικρό μέρος μιας πόλης (και κατά συνέπεια μεγάλες πόλεις) με υψηλή γεωμετρική λεπτομέρεια και παραπάνω από εκατοντάδες πολύγωνα.

Χρησιμοποιούμε μια γραμματική σχημάτων (shape grammars), που ονομάζεται CGA σχήμα (CGA shape, Computer Generated Architecture shape), με τους κανόνες παραγωγής που εξελίσσουν επαναληπτικά ένα σχέδιο δημιουργώντας όλο και περισσότερη λεπτομέρεια στα σχήματα που κατασκευάζονται. Στο πλαίσιο των κτιρίων, οι κανόνες παραγωγής κατασκευάζουν πρώτα ένα ακατέργαστο ογκομετρικό μοντέλο ενός κτιρίου, που ονομάζεται μοντέλο μάζας, και στη συνέχεια συγκροτούμε την πρόσοψη του κτιρίου και τέλος προσθέτουμε όποιες λεπτομέρειες για τα παράθυρα, τις πόρτες, τις σκεπές, τα τόξα και τα κάγκελα. Αργότερα του προσδίδουμε μια απεικόνιση υφής (texture mapping) σε κάθε στοιχείο του κτιρίου ώστε να του αποδοθεί ένας ρεαλιστικός χαρακτήρας. Το κύριο πλεονέκτημα της μεθόδου αυτής είναι ότι η κατασκευή της ιεραρχικής δομής και οι ενδείξεις του μοντέλου καθορίζονται στη διαδικασία μοντελοποίησης. Αυτή η σημασιολογική πληροφορία είναι σημαντικό για την επαναχρησιμοποίηση των κανόνων σχεδιασμού για διαδικαστικές διαφορές (Εικόνα 1) και έτσι δημιουργώντας μια μεγάλη γκάμα αρχιτεκτονικής απορυθμίζοντας μια πόλη, πιο συγκεκριμένα την πόλη της Κέρκυρας.

Εικόνα 1: Απεικόνιση μοντέλων σπιτιών. Αριστερά το House_5 και δεξιά το House_7. Το House_7 προέκυψε από τους κανόνες γραμματικής σχήματος του House_5.

Η ιδέα της μοντελοποίησης αστικών περιβαλλόντων που χρησιμοποιούν γραμματικές σχημάτων ανακαλύφθηκε πρόσφατα από τους [Parish & Müller, 2001] και [Wonka et al., 2003]: Από τη μια πλευρά, οι [Parish & Müller, 2001] έδειξαν πώς να αναπαράγουν μεγάλα αστικά περιβάλλοντα όπου το κάθε κτίριο αποτελεί απλά μαζικά μοντέλα και σκιάσεις για μεγαλύτερη και πιστότερη λεπτομέρεια στις προσόψεις κτιρίων. Ενώ από την άλλη πλευρά, [Wonka et al., 2003] έδειξαν πως μπορεί κανείς να αναπαράγεις γεωμετρικές λεπτομέρειες σε προσόψεις κάθε κτιρίου μοναδικά. Ιδανικά θα θέλαμε να συνδυάσουμε αυτές τις δύο ιδέες και να παράγουμε τεράστια και λεπτομερή αστικά περιβάλλοντα. Ωστόσο υπάρχει μια σημαντική πρόκληση στο πλαίσιο των μαζικών μοντέλων που χρειάζονται να αντιμετωπιστούν και απαιτούν εκτενή αλλαγή και στα δύο μοντέλα. (1) Οι [Parish & Müller, 2001] θα μπορούσαν να παράγουν απλά μοντέλα απλά προσθέτοντας μετακινούμενα και συστρεμμένα κουτιά και οι λεπτομέρειες

προσθέτονταν με σκίαση. Αυτή όμως η στρατηγική δε μπορεί να παράγει επαρκή γεωμετρική λεπτομέρεια και θα υπάρξουν αρκετές ανεπιθύμητες τομές ενός αρχιτεκτονικού στοιχείου (Εικόνα 2).

Εικόνα 2: Το μοντελοποιημένο κτίριο αποτελούμενο από 14 ογκομετρικά αρχέτυπα (κύβους, στέγες) τοποθετημένα από μια στοχαστική γραμματική σχημάτων. Αριστερά: Οι υπάρχουσες μέθοδοι των διαδικαστικών αρχιτεκτονικής μπορεί είτε να τοποθετήσει σκιάσεις σε κάθε όγκο κτιρίου είτε χρησιμοποιώντας κανόνες τμηματοποίησης για διαδικαστική βελτίωση. Και στις δύο περιπτώσεις μερικές ανεπιθύμητες τομές θα 'κόψουν' τα παράθυρα (ή άλλα στοιχεία) με αφύσικο τρόπο, καθώς οι όγκοι δεν αναγνωρίζουν ο ένας τον άλλον. Δεξιά: Η προσέγγιση των κανόνων γραμματικής σχημάτων επιτρέπει την επίλυση αυτών των συγκρούσεων. Επιπλέον, μπορεί να τοποθετηθεί γεωμετρία σε πολύγωνα διαφορετικού προσανατολισμού όπως οι επιφάνειες στέγης. Το παράδειγμα αυτό δημιουργήθηκε μόνο με τη χρήση 6 κανόνων.

(2) οι κανόνες τμηματοποίησης (split rules) που προτείνουν οι [Wonka et al., 2003] επαρκούν μόνο για τα απλά μαζικά μοντέλα. Πολύπλοκα μαζικά μοντέλα θα απαιτούσαν ένα υπερβολικό αριθμό διασπάσεων. Περαιτέρω, το μαζικό μοντέλο δε μπορεί εύκολα να αλλάξει επειδή νέες διαμορφώσεις θα χρειαστούν επιπρόσθετους κανόνες παραγωγής και αντικείμενα αυθαιρέτου προσανατολισμού δε μπορούν να αντιμετωπιστούν εύκολα.

Σε αυτή τη διατριβή θα σας παρουσιαστεί μια grammar-based λύση για να παράγει λεπτομερή οικοδομικά κελύφη που προκύπτουν από περίπλοκα μαζικά μοντέλα. Η προσέγγιση και σκοπός της πτυχιακής εργασίας βασίζεται σε κανόνες γραμματικής σχημάτων (shape grammar rules) κατάλληλα περί αρχιτεκτονικής γραφικών υπολογιστών: Αρχικά θέτουμε και απαντάμε ερωτήματα περί Διαδικαστικής τεχνικής και γιατί επιλέγεται μια τέτοια διαδικασία. Στα επόμενα κεφάλαια θα δούμε να γίνεται μια εκτενής προσέγγιση της διαδικαστικής μοντελοποίησης και ύστερα γίνεται μια περιγραφή στους κανόνες γραμματικής σχημάτων (shape grammar rules), καθώς και των κανόνων τμηματοποίησης (split

grammar rules) που αποτελεί μια υποκατηγορία των πρώτων. Τέλος, θα ακολουθήσει μια εκτεταμένη αναπαράσταση των παλαιών κτιρίων του νησιού της Κέρκυρας χρησιμοποιώντας τους κανόνες γραμματικής σχημάτων παρουσιάζοντας σαφή παραδείγματα και οδηγίες.

ΚΕΦΑΛΑΙΟ 1

Διαδικαστική Τεχνική

ΕΙΣΑΓΩΓΗ

Το κεφάλαιο αυτό μας εισάγει σε μια γενικότερη ιδέα και μας ξεκαθαρίζει το πεδίο γνώσης της εν λόγω πτυχιακής εργασίας μέσα από το βασικό και πρωταρχικό ερώτημα Τι είναι Διαδικαστική τεχνική (procedural technique) και έτσι μετέπειτα ξετυλίγεται το κουβάρι της γνώσης πάνω στο αντικείμενο της Διαδικαστικής Μοντελοποίησης. Ακόμα μας παρουσιάζει τη δύναμη των διαδικαστικών τεχνικών των τελευταίων ετών πάνω στον κλάδο των γραφικών υπολογιστών αλλά και την εξέλιξή τους μέσα από την προηγμένη γεωμετρική μοντελοποίηση.

1.1 Διαδικαστική Τεχνική και Γραφικά Υπολογιστών

Οι διαδικαστικές τεχνικές έχουν χρησιμοποιηθεί κατά τη διάρκεια της ιστορίας των γραφικών υπολογιστών. Αρκετές πρώιμες τεχνικές μοντελοποίησης και υφής συμπεριλαμβανομένων τους διαδικαστικούς ορισμούς της γεωμετρίας και των χρωματισμό επιφανειών. Από αυτά τα πρώτα στάδια, οι τεχνικές μοντελοποίησης έχουν εξελιχθεί σε ένα σημαντικό πρότυπο ισχυρής μοντελοποίησης, υφής και κίνησης. Κατά τα μέσα-τέλη του 1980, οι διαδικαστικές τεχνικές για τη δημιουργία ρεαλιστικής υφής, όπως το μάρμαρο, το ξύλο, η πέτρα και άλλων φυσικών υλών απέκτησαν ευρεία χρήση. Οι τεχνικές αυτές επεκτάθηκαν στις διαδικαστικές τεχνικές, συμπεριλαμβανομένων των μοντέλων του στοιχείου του νερού, του καπνού, της φωτιάς, του ατμού, των πλανητών. Η ανάπτυξη της γλώσσας σκίασης RenderMan [Pixar, 1989] επεκτάθηκε σε μεγάλο βαθμό με τη χρήση των διαδικαστικών τεχνικών. Σήμερα, τα περισσότερα συστήματα rendering και animation παρέχουν ακόμα ένα διαδικαστικό περιβάλλον. Οι διαδικαστικές τεχνικές έχουν γίνει ένα συναρπαστικό προϊόν, ζωτικής σημασίας για τη δημιουργία υπολογιστικής αναπαραγωγής ρεαλιστικών εικόνων και animation. Καθώς το πεδίο συνεχίζει να εξελίσσεται, η σημασία και η σπουδαιότητα των διαδικαστικών τεχνικών θα συνεχίσουν να αυξάνονται. Υπήρξαν δύο πρόσφατες σημαντικές εξελίξεις για διαδικαστικές τεχνικές πραγματικού χρόνου: η αύξηση της ισχύς των CPU και οι ισχυροί προγραμματιζόμενοι επεξεργαστές γραφικών (programmable graphics processors (GPUs)), οι οποίοι είναι διαθέσιμοι σε προσιτούς υπολογιστές

και κονσόλες παιχνιδιών. Αυτό το γεγονός έχει ξεκινήσει μια εποχή όπου μπορούμε να οραματιστούμε και να υλοποιήσουμε πολύπλοκα και διαδραστικά διαδικαστικά μοντέλα και εφέ.

1.2 Τι είναι μια Διαδικαστική Τεχνική;

Οι διαδικαστικές τεχνικές είναι τμήματα κώδικα ή αλγόριθμων που καθορίζουν κάποιο χαρακτηριστικό ενός αναπαραγόμενου μοντέλου ή εφέ υπολογιστή. Για παράδειγμα, μια διαδικαστική υφή μιας μαρμάρινης επιφάνειας δε χρησιμοποιεί μια σαρωμένη εικόνα για να καθορίσει τις τιμές των χρωμάτων που αποτελούν στο σύνολό του την πολυσύνθετη απόχρωση της επιφάνειας του μαρμάρου. Αντί αυτού, χρησιμοποιεί αλγόριθμους και μαθηματικές συναρτήσεις για τον προσδιορισμό των χρωμάτων.

1.3 Η Δύναμη των Διαδικαστικών Τεχνικών

Ένα από τα πιο σημαντικά χαρακτηριστικά των διαδικαστικών τεχνικών είναι η αφαιρετικότητα. Σε μια διαδικαστική προσέγγιση, αντί να προσδιορίζει και να αποθηκεύει όλες τις πολύπλοκες λεπτομέρειες μιας σκηνής ή μιας αλληλουχίας σκηνών, τα ομαδοποιούμε αφαιρετικά σε μια συνάρτηση ή έναν αλγόριθμο (πχ. μια διαδικασία) και αξιολογούμε την εν λόγω διαδικασία όταν και όπου χρειάζεται. Έτσι εξοικονομούμε χώρο αποθήκευσης, καθώς οι λεπτομέρειες δεν καθορίζονται πια παρά υπονοούνται στη διαδικασία, και οι χρονικές απαιτήσεις για τον προσδιορισμό των λεπτομερειών μετατίθενται από τον προγραμματιστή στον υπολογιστή. Αυτό μας επιτρέπει να δημιουργήσουμε εγγενείς μοντέλα πολλαπλών αναλύσεων και υφών που μπορούμε να αξιολογήσουμε στην ανάλυση που χρειάζεται.

Μπορούμε επίσης να αποκτήσουμε τη δύναμη του ελέγχου των παραμέτρων, που μας επιτρέπει να εκχωρήσουμε σε μια παράμετρο μια ουσιαστική έννοια (για παράδειγμα έναν αριθμό που καθιστά τα βουνά ταχιά ή ομαλά). Ο παραμετρικός έλεγχος μας παρέχει επίσης διεύρυνση στην προσπάθεια του modeler/animator: μερικές παράμετροι δίνουν μεγάλες ποσότητες λεπτομέρειας. Ο [A. R. Smith, 1984] αναφέρθηκε σε αυτό ως επέκταση των βάσεων δεδομένων. Αυτός ο παραμετρικός έλεγχος ανακουφίζει τον χρήστη από το χαμηλότερο επίπεδο ελέγχου και προδιαγραφών των λεπτομερειών. Επίσης, αξιοσημείωτο γεγονός είναι όταν συχνά εκπλησσόμαστε ευχάριστα από τις απροσδόκητες συμπεριφορές από τέτοιες διαδικασίες, ιδιαίτερα από πιθανολογικές διαδικασίες.

Τα διαδικαστικά μοντέλα επίσης προσφέρουν ευελιξία. Ο σχεδιαστής των διαδικασιών μπορεί να συλλάβει την ουσία του αντικειμένου, του φαινομένου, ή

της κίνησης δίχως να περιορίζεται από τους σύνθετους νόμους της φυσικής. Οι διαδικαστικές τεχνικές επιτρέπουν την ενσωμάτωση στο μοντέλο από οποιαδήποτε επιθυμητή ποσότητα φυσικής ακρίβειας. Ο σχεδιαστής μπορεί να παράγει ένα ευρύ φάσμα από εφέ, από ακριβή προσομοίωση των φυσικών νόμων μέχρι αμιγώς καλλιτεχνικά εφέ.

1.4 Διαδικαστικές Τεχνικές και Προηγμένη Γεωμετρική Μοντελοποίηση

Οι τεχνικές γεωμετρικής μοντελοποίησης στα γραφικά υπολογιστών έχουν εξελιχθεί σημαντικά, καθώς το πεδίο ωριμάζει και επιχειρεί να απεικονίσει όλο και πιο πολύπλοκα μοντέλα και την πολυπλοκότητα της φύσης. Προηγούμενα γεωμετρικά μοντέλα, όπως πολυγωνικά μοντέλα, σημεία, και γραμμές, δεν επαρκούν για να εκπροσωπήσουν αυτή την αυξημένη πολυπλοκότητα με ένα ελεγχόμενο και εύχρηστο τρόπο. Οι υψηλότερου επιπέδου τεχνικές μοντελοποίησης έχουν αναπτυχθεί για να παρέχουν μια αφαίρεση στο μοντέλο, κωδικοποιώντας τις κλάσεις του αντικειμένου, και επιτρέποντας υψηλού επιπέδου ελέγχου και προδιαγραφών των μοντέλων. Πολλές από αυτές τις προηγμένες γεωμετρικές μεθόδων μοντελοποίησης είναι εγγενώς διαδικαστικές. Grammar-based models [Smith, 1984; Prusinkiewicz & Lindenmayer, 1990], συμπεριλαμβάνοντας των graftals και L-systems, επιτρέπουν τον προσδιορισμό μερικών παραμέτρων για την προσομοίωση πολύπλοκων μοντέλων όπως δέντρων, φυτών, και άλλων φυσικών αντικειμένων. Τα μοντέλα αυτά χρησιμοποιούν τις επίσημες γλώσσες για να καθορίσουμε πολύπλοκους κανόνες ανάπτυξης για τα φυσικά αντικείμενα.

1.5 Χαρακτηριστικά Διαδικαστικών Τεχνικών

Η βασική ιδιότητα της διαδικαστικής παραγωγής (procedural generation), είναι ότι περιγράφει την οντότητα, είτε πρόκειται για γεωμετρία, την υφή ή το εφέ με όρους αλληλουχίας των οδηγιών παραγωγής παρά ως ένα στατικό μπλοκ δεδομένων. Οι οδηγίες μπορούν στη συνέχεια να κληθούν, όταν χρειαστεί να κατασκευαστούν στιγμιότυπα με διαφορετικά χαρακτηριστικά. Ένα τυπικό παράδειγμα αυτής της προσέγγισης θα είναι ο πληθυσμός ενός δάσους με την τεχνική της διαδικαστικής παραγωγής μοναδικών δέντρων [Interactive Data Visualization Inc., 2006].

Οι διαδικαστικές τεχνικές μπορούν συνεπώς να χρησιμοποιηθούν για να παράγουν ποικίλα στοιχεία ενεργητικού. Μια από τις βασικές τεχνικές που μπορούν να χρησιμοποιηθούν είναι τα τρισδιάστατα αρχέτυπα με τυχαίες παραμέτρους, για παράδειγμα ένα ορθογώνιο παραλληλεπίπεδο με τυχαία τιμή ύψους. Απλοί αλγόριθμοι χρησιμοποιούν ψευδείς τυχαίες λειτουργίες που μπορούν να χρησιμοποιηθούν για να παράγουν θόρυβο για χρήση σε υφή και φυσικών σχηματισμών [Ken Perlin, 1985]. Πιο πολύπλοκες αναδρομές

αλγορίθμων όπως fractals ή L-systems μπορούν να χρησιμοποιηθούν για να αναδημιουργήσουν οργανικές δομές που βρίσκονται στη φύση, όπως νιφάδες χιονιού και δέντρα [P. Prusinkiewicz & A. Lindenmayer, 1990]. Πολλές διαδικαστικές τεχνικές έχουν σχεδιαστεί ειδικά για τη μοντελοποίηση αστικών μοντέλων [B. Watson et. al., 2008], [C. Vanegas et. al., 2009]. Όπως πολλές τεχνικές μοντελοποίησης των φυτών, ορισμένες τεχνικές μοντελοποίησης χρησιμοποιούν L-systems. Ο [Ebert et. al., 2003] προσδιορίζει τα παρακάτω ως σημαντικά χαρακτηριστικά των διαδικαστικών τεχνικών:

- **Αφαίρεση:** Δεδομένα γεωμετρικής και υφής δεν προσδιορίζονται με τη συμβατική έννοια του όρου, αντί αυτού οι λεπτομέρειες αντλούνται από έναν αλγόριθμο ή ένα σύνολο διαδικασιών. Αυτές οι διαδικασίες στη συνέχεια τις χειρίζεται ο υπολογιστής και καλούνται όποτε χρειάζονται. Απαιτούνται ελάχιστες λεπτομέρειες και ο χειριστής μπορεί να χειριστεί τα δεδομένα του μοντέλου εύκολα χωρίς να απαιτείται βαθιά γνώση της εφαρμογής.
- **Παραμετρικός έλεγχος:** οι παράμετροι ορίζονται και ρυθμίζονται ώστε να αντιστοιχούν άμεσα σε μια συγκεκριμένη συμπεριφορά στη διαδικαστική παραγωγή. Ο προγραμματιστής μπορεί να ορίσει όσο περισσότερους χρήσιμους ελέγχους απαιτούνται για τους καλλιτέχνες για να λειτουργήσουν αποτελεσματικά. Παράδειγμα παραμέτρων αποτελεί το ύψος των βουνών σε ένα αλγόριθμο εδάφους ή τον αριθμό των τμημάτων σε ένα διαδικαστικό πεδίο.
- **Ευελιξία:** είναι δυνατόν να συλλάβουμε την ουσία μιας οντότητας χωρίς να την οριοθετήσουμε ρητά μέσα στα όρια του πραγματικού κόσμου. Οι παράμετροι μπορούν αργότερα να ποικίλουν για να παράγουν ένα ευρύ φάσμα αποτελεσμάτων οι οποίες δεν περιορίζονται απαραίτητα στους περιορισμούς του αρχικού μοντέλου.

Οι διαδικαστικές τεχνικές έχουν εφαρμοστεί με επιτυχία στην παραγωγή πολλών σύνθετων φαινομένων σε γραφικά υπολογιστών και έχουν αποδειχθεί ευεργετική για πολλούς λόγους .

Υφές, γεωμετρία ή εφέ αφηρημένα μέσα σε διαδικαστικούς αλγορίθμους δε καθορίζονται σε μια ανάλυση μοντέλου ή έναν αριθμό πολυγώνων. Άρα οι διαδικαστικές τεχνικές είναι ως εκ φύσεως πολλαπλής ανάλυσης και μπορούν να διαφέρουν ως προς την πολυπλοκότητα της παραγωγής τους. Αυτή η ικανότητα είναι ιδιαίτερα ενδιαφέρουσα για τα γραφικά υπολογιστών. Για παράδειγμα, το επίπεδο λεπτομέρειας (level of detail, LOD) είναι σημαντικό σε οποιοδήποτε σύστημα απόδοσης και ουσιαστικό για εφαρμογές απόδοσης πραγματικού χρόνου [Tomas Akenine-Möller & Eric Haines (2002]. Η ιδέα πίσω από το LOD είναι να χρησιμοποιούνται πιο απλές μορφές μιας οντότητας εάν συνεισφέρει σε μικρότερο βαθμό με την παρεχόμενη τελική εικόνα. Έτσι, για ένα αντικείμενο που καταλαμβάνει μόνο 4 pixels στην τελική εικόνα, δεν απαιτούνται 10.000 πολύγωνα και η βασική απεικόνιση χρησιμοποιεί 10 πολύγωνα θα μπορούσε να ήταν

επαρκής. Η φύση της πολλαπλής ανάλυσης των διαδικαστικών τεχνικών επιτρέπει τη δυνατότητα αυτόματης δημιουργίας μοντέλων σε πολλαπλά επίπεδα λεπτομέρειας [David S. et. al., 2003].

Συνοπτική περιγραφή των παραγόμενων αντικειμένων είναι δυνατή και συχνά εκφράζονται με όρους μερικών απλών παραμέτρων. Αυτές οι μικρές περιγραφές μπορούν να χρησιμοποιηθούν για να δημιουργήσουν μεγάλες ποσότητες λεπτομερών υφών και γεωμετρίας, αυτό το φαινόμενο είναι γνωστό ως δεδομένα ενίσχυσης (data amplification) [David S. et. al., 2003] και παρέχει στους προγραμματιστές τα μέσα για να κατασκευάσουν έναν ολόκληρο κόσμο παιχνιδιού που είναι εύκολα διανεμητέα σε συνδέσεις χαμηλού εύρους ζώνης του δικτύου. Η περιεκτικότητα των διαδικαστικών τεχνικών γίνεται αντικείμενο εκμετάλλευσης από δημιουργούς Demo Scene που δημιουργούν και διανέμουν σκηνές που είναι πολύπλοκες και πλούσιες σε λεπτομέρεια, υπό τη μορφή μικροσκοπικών εκτελέσιμων αρχείων της τάξης των 2KB [International Scene Organization, 2004].

Η ευελιξία και ο έλεγχος που προσφέρονται από τις διαδικαστικές τεχνικές δίνουν στο σχεδιαστή μια πλατφόρμα για καλλιτεχνική ελευθερία και πειραματισμό. Νέα οπτικά εφέ και πρωτότυπα αντικείμενα μπορούν να δημιουργηθούν από τον πειραματισμό με τις τιμές των παραμέτρων που υπερβαίνουν τα κανονικά όρια [Side Effects Software, 2005].

Τυπικά, οι διαδικαστικοί αλγόριθμοι εφαρμόζονται σε λογισμικό, ωστόσο, πρόσφατες εξελίξεις στο υλικό γραφικών υπολογιστών έχουν δώσει την δυνατότητα εκτέλεσης αυτών απευθείας στην GPU. Για παράδειγμα, πολύπλοκες διαδικαστικές τεχνικές όπως οι ογκομετρικές υφές που προηγουμένως ήταν αδύνατο να τρέξουν σε πραγματικό χρόνο τώρα μπορούν να εφαρμοσθούν με τον τρόπο αυτό [Evan Hart, 2002] [Spitzer John et. al., 2003].

ΕΠΙΛΟΓΟΣ

Μάθαμε πως στο πεδίο των γραφικών υπολογιστών οι εξελίξεις επέρχονται η μία μετά την άλλη με ραγδαίους ρυθμούς όσων αφορά τις διαδικαστικές τεχνικές. Για παράδειγμα αναπτύχθηκαν μέθοδοι που μπορούσαν να απεικονίσουν πιστά επιφάνειες με μεγάλη πολυπλοκότητα στην ανάλυσή τους μέσα από ένα σύνολο κανόνων όπου δε το επέτρεπαν απλές σαρώσεις εικόνων για την απόδοση ρεαλιστικής υφής πάνω σε μοντέλα αντικειμένων τόσο σε απλά όσο και σε πιο σύνθετα μοντέλα αντικειμένων. Η ανάπτυξη αυτών των τεχνικών κατέστησαν απαραίτητη και την μετεξέλιξη στο υλικό κομμάτι των υπολογιστών όπως η αύξηση της ισχύς των CPU. Πέρα από τη μοντελοποίηση απλών υλικών μοντέλων οι διαδικαστικές τεχνικές προχώρησαν και στην επεξεργασία και αναπαραγωγή πιο σύνθετων μοντέλων, όπως το animation. Μπορούμε μέσα από διάφορες διαδικαστικές μεθόδους να καθορίσουμε τη μορφή που θα πάρουν τα μοντέλα μας

περιγράφοντας τα μέσα από παραμέτρους τμημάτων κώδικα ή αλγόριθμους για να μας αποδοθεί ένα επιθυμητό αποτέλεσμα παρέχοντας στον σχεδιαστή μεγάλη ευελιξία προσδίδοντας σε κάθε μοντέλο όποια αφαιρετική παρουσίαση εκείνος επιθυμεί. Η απεικόνιση γραμμών, σημείων και απλών πολυγωνικών μοντέλων αδυνατούσαν να απεικονίσουν πολύπλοκα μοντέλα. Έτσι εφευρέθηκαν νέοι κανόνες γεωμετρικής μοντελοποίησης που μπορούσαν να αποδώσουν πιστά σύνθετα μοντέλα του φυσικού περιβάλλοντος. Αυτοί οι κανόνες γεωμετρικής μοντελοποίησης αποτελούν τον βασικό όρο που ονομάζεται διαδικαστική μοντελοποίηση.

Στο επόμενο κεφάλαιο θα ασχοληθούμε με τη διαδικαστική μοντελοποίηση καθ’ αυτήν καθώς αποτελεί το γενικό όρο, την ομπρέλα, που από κάτω της κρέμονται διάσημες και χρήσιμες τεχνικές διαδικαστικών μεθόδων όπως τα split grammars. Τεχνική που αποτελεί το αντικείμενο αυτής της διατριβής.

ΚΕΦΑΛΑΙΟ 2

Διαδικαστική Μοντελοποίηση

ΕΙΣΑΓΩΓΗ

Καθώς η τεχνολογία εξελίσσεται και η υπολογιστική ισχύς αυξάνεται, η όρεξη των καταναλωτών για περισσότερη λεπτομέρεια και ρεαλισμό συνεχώς αυξάνεται. Η σύγχρονη βιομηχανία των μέσων όπως παιχνίδια, κινηματογράφου, διαφήμισης και τηλεόρασης, αγωνίζονται για να ανταποκριθούν στις προσδοκίες που τέθηκαν από μεγαλύτερα έργα και καθημερινά έξοδα παραγωγής είναι εκτός ελέγχου.

Η παραδοσιακή προσέγγιση για την κάλυψη της ζήτησης των καταναλωτών έχει απλά αυξήσει τον αριθμό των καλλιτεχνών που εργάζονται πάνω σε ένα έργο να παράγουν μεγαλύτερο, πιο λεπτομερή και ρεαλιστικό περιεχόμενο. Εντούτοις, ολοένα και περισσότερο ο καλλιτεχνικός αγωγός δεν παρουσιάζει κλιμάκωση, πράγμα που σημαίνει ότι επιπλέον αριθμός καλλιτεχνών δε δημιουργούν μια ανάλογη απόδοση περιεχομένου. Έτσι οι πρόσθετες δαπάνες προσθέτονται στο ήδη υψηλό κόστος ανάπτυξης που πληρώνονται από τον καταναλωτή. Το αποτέλεσμα αυτού είναι ότι ο χρόνος και τα χρήματα που θα μπορούσαν να έχουν διατεθεί για τη βελτίωση του παιχνιδιού ή την προσθήκη καινοτόμων χαρακτηριστικών έχουν χαθεί με τη δημιουργία του περιεχομένου. Ως συνέπεια του μεγάλου κόστους ανάπτυξης, ένα εμπόδιο εισόδου στην αγορά έχει δημιουργηθεί και νέες αρχάριες επιχειρήσεις θεωρούν ότι είναι δύσκολο να πάρουν κάποια θέση πνίγοντας έτσι την καινοτομία. Μια πιθανή λύση για το πρόβλημα της δημιουργίας περιεχομένου είναι η εφαρμογή των διαδικαστικών τεχνικών. Αυτές οι τεχνικές έχουν χρησιμοποιηθεί για πάνω από 20 έτη στον τομέα των γραφικών του υπολογιστή [David S. et. al., 2003] για ένα ευρύ φάσμα εφαρμογών: προσθήκη θορύβου σε υπάρχουσες υφές [Ken Perlin, 1985], δημιουργώντας τρισδιάστατες υφές των φυσικών υλικών, όπως μάρμαρο και ξύλο [Ken Perlin., 1999], οπτικοποίηση της αληθοφανών μοντέλων διαφόρων ειδών δέντρων και φυτών [P. Prusinkiewicz and A. Lindenmayer, 1990], και της δημιουργίας λεπτομερών κυτταρικών υφών, όπως του δέρματος ή του φλοιού ενός δέντρου [David S. et. al., 2003]. Ολόκληρων διαδικαστικών κόσμων είναι τώρα δυνατά να απεικονιστούν και αυτό αποδεικνύετε μέσα από εφαρμογές όπως το MojoWorld [Ken Musgrave, 2006] και το CityEngine, όπου το κάθε στοιχείο του κόσμου αυτού συμπεριλαμβάνει ρεαλιστικά φυσικά χαρακτηρίστηκα όπως εδάφη, λίμνες, δέντρα, θάμνους όλα κατασκευασμένα χρησιμοποιώντας διαδικαστικές τεχνικές (procedural techniques). Πρόσφατες διαδικαστικές εφαρμογές (procedural applications) έχουν επεκταθεί περαιτέρω για την προσομοίωση ειδικών εφέ, συμπεριλαμβανομένων συστημάτων σωματιδίων (particle systems), νερού, ακόμα

και φυσικές κινήσεις φυσικών στοιχείων [Interactive Data Visualization Inc., 2006]. Πολύπλοκες σκηνές που περιέχουν πολλά διαφορετικά μοντέλα που θα έπρεπε κανονικά να χρειαστούν μήνες για να σχεδιαστούν με το χέρι, τώρα ένα τεράστιο τμήμα αυτών των σκηνών μπορούν να σχεδιαστούν χρησιμοποιώντας ειδικά πακέτα διαδικαστικής παραγωγής (procedural generation) [Side Effects Software, 2005] που μπορούν να παράγουν λεπτομερή και ποικίλα μοντέλα μέσα σε λίγα λεπτά. Η διαδικαστική παραγωγή είναι μια μέθοδος εξοικονόμησης χρόνου γρήγορης και αποτελεσματικής παραγωγής περιεχομένου που μπορεί να ανακουφίσει και, ενδεχομένως, να λύσει τα προβλήματα του αυξημένου κόστους δημιουργίας περιεχομένου.

Οι υπάρχουσες διαδικαστικές λύσεις κατά κύριο λόγο εφαρμόζουν διαδικαστικές τεχνικές για την παραγωγή των φυσικών φαινομένων, αλλά πολλές από τις ίδιες τεχνικές έχουν προφανείς εφαρμογές στην παραγωγή φυσικών φαινομένων. Η δουλειά μας εδώ επικεντρώνεται στη διαδικαστική δημιουργία μιας μικρής πόλης, συγκεκριμένα της πόλης της Κέρκυρας (όπου και θα δούμε αργότερα στα επόμενα κεφάλαια), που αργότερα μπορεί να γίνει η χρήση τους σε παιχνίδια και άλλες γραφικές εφαρμογές που βρίσκονται σε αστικά τοπία.

Το αστικό τοπίο είναι δύσκολο να το μοντελοποιήσεις. Είναι πλούσιο σε οπτική και λειτουργική πολυπλοκότητα και είναι αποτέλεσμα της ανάπτυξης και εξέλιξης επί εκατοντάδες χρόνια υπό την επίδραση αμέτρητων παραγόντων. Μερικούς από τους σημαντικότερους παράγοντες που επηρεάζουν τις πόλεις περιλαμβάνουν τον πληθυσμό, τις συγκοινωνίες, το περιβάλλον, την ανύψωση εδάφους, τη βλάστηση, τη γεωλογία και την πολιτιστική επιρροή. Είναι μια τεράστια πρόκληση για τους ερευνητές και προγραμματιστές να δημιουργήσουν ένα ρεαλιστικό μοντέλο ενός τόσου μεγάλου και πολύπλοκου συστήματος. Στόχος μας είναι να αναπτύξει ένα προσιτό διαδραστικό σύστημα λογισμικού που μπορεί να δημιουργήσει αυτόματα μια ρεαλιστικό, αναλυτικό και ποικιλόμορφο μοντέλο μιας πόλης κατάλληλο για χρήση με απόδοση πραγματικού χρόνου.

2.1 Περί Διαδικαστικής Μοντελοποίησης

Η διαδικαστική μοντελοποίηση (procedural modeling) είναι μια αυξανόμενη δημοφιλής προσέγγιση στα γραφικά υπολογιστών για την ανάπτυξη γρήγορων, τεράστιας έκτασης περιβάλλοντος, συμπεριλαμβάνοντας τόσο τα αστικά όσο και τα υπαίθρια περιβάλλοντα που βασίζονται σε ένα σύνολο προκαθορισμένων κανόνων.

Η διαδικαστική μοντελοποίηση, σε αντίθεση με χειροκίνητη μοντελοποίηση, είναι η διαδικασία της δημιουργίας μοντέλων με τη χρήση των αλγορίθμων που

κωδικοποιούν τις οδηγίες που απαιτούνται για τη δημιουργία ενός μοντέλου. Οι χειροκίνητες διαδικασίες μοντελοποίησης απαιτούν άμεση αλληλεπίδραση με το δημιουργό μοντελιστή (χρησιμοποιώντας τρισδιάστατα πακέτα μοντελοποίησης). Η διαδικαστική μοντελοποίηση, λοιπόν, εξαλείφει την ανάγκη για άμεση ανθρώπινη αλληλεπίδραση. Ο χρήστης καθορίζει από μόνος του τις τιμές των παραμέτρων που επιθυμεί για τους αλγόριθμους, που σε πολλές περιπτώσεις είναι η μόνη μορφή αλληλεπίδρασης που έχει ο χρήστης με το σύστημα. Υπάρχει μια σειρά από οφέλη με τη χρήση της διαδικαστικής μοντελοποίησης:

- Η διαδικαστική μοντελοποίηση είναι παραμετρική, το γεγονός αυτό επιτρέπει ένα σχεδόν απροσδιόριστο αριθμό μεταβολής ενός μοντέλου. Η παραμετροποίηση επίσης επιτρέπει αλλαγές σε ένα συγκεκριμένο μοντέλο χωρίς την ανάγκη να αλλάξει ολόκληρο το σκηνικό [Marshall et al., 1980].
- Η διαδικαστική μοντελοποίηση προωθεί την ενίσχυση της βάσης δεδομένων, η οποία παράγει μια μεγάλη ποσότητα των δεδομένων από ένα μικρό σύνολο εισόδων [Apodaca & Gritz, 2000]. Αυτό είναι πλεονεκτικό αφού μεγάλα, πολύπλοκα μοντέλα μπορούν να αποκτηθούν με ελάχιστες εισροές.
- Οι διαδικασίες έχουν την ικανότητα να αναπαράγουν διεργασίες που συμβαίνουν στα φυσικά συστήματα, και να οδηγήσει στη δημιουργία δομών, οι οποίες αντικατοπτρίζουν τον πραγματικό κόσμο.

Η διαδικαστική μοντελοποίηση της πόλης αναφέρεται στην αυτόνομη κατασκευή τρισδιάστατων πόλεων, όπου μόνο ένας μικρός αριθμός αρχικών παραμέτρων πρέπει να παρέχονται. Τέτοια συστήματα κάνουν χρήση των αλγορίθμων, παρά με χειροκίνητους προσδιορισμούς, για να καθορίσουν οδικά πρότυπα και να τοποθετήσουν δομές σε εδάφη. Οι αρχικές παράμετροι επηρεάζουν την εμφάνιση του τελικού αποτελέσματος, και συχνά εμβολιάζονται τυχαίες τιμές που χρησιμοποιούνται προκειμένου να δημιουργήσουν μια πλούσια και μη ομοιόμορφη εμφάνιση. Ωστόσο, βρίσκοντας το σωστό σύνολο παραμέτρων (και τις αντίστοιχες τιμές), προκειμένου να διαμορφώσει ακριβώς μια υπάρχουσα πόλη είναι σχεδόν αδύνατο, και ως εκ τούτου οι διαδικαστικές πόλεις αποσκοπούν στις προσομοιώσεις, την εικονική πραγματικότητα και τις εφαρμογές ψυχαγωγίας.

Από τότε που η διαδικαστική μοντελοποίηση είναι ένα τέτοιο ενεργό πεδίο έρευνας υπάρχει βέβαια ένας μεγάλος αριθμός δημοσιεύσεων πάνω σε διαφορετικές προσεγγίσεις στο θέμα αυτό. Τα πρωτότυπα L-systems έχουν εντυπωσιακά αποτελέσματα στη μοντελοποίηση των φυτών. Όντας παράλληλων γραμματικών ήταν απόλυτα κατάλληλα λαμβάνοντας υπόψη τον κύριο στόχο της ανάπτυξης κατά την πάροδο του χρόνου [Prusinkiewicz & Lindenmayer, 1990]. Όταν ασχολούμαστε με τις ανθρωπογενείς κατασκευές όπως κτίρια είναι πιο βολικό να σχεδιάσει τη διαδικασία μοντελοποίησης ως μια ακολουθία βημάτων διαχωρισμού, σταδιακά να διυλίσεις το μοντέλο και επιτρέποντας να το δομήσεις. Σε αυτό το πλαίσιο οι γραμματικές τμηματοποίησης (split grammars), οι οποίες βασίζονται στις προαναφερόμενες γραμματικές σχήματος (shape grammars), εισήχθησαν και εφαρμόζονται σε μοντέλα των κτιρίων. Αποτελούνται από μια μεγάλη βάση

δεδομένων κανόνων παραγωγής χρησιμοποιώντας μια στοχαστική διαδικασία για να αποδώσει στα προκύπτοντα κτίρια κάποια τυχαιότητα, ως εκ τούτου, μια πιο ρεαλιστική εμφάνιση [Wonka et al. 2003]. Μία περαιτέρω εξέλιξη προς αυτή την κατεύθυνση είναι CGA (Computer Generated Architecture) σχήμα που είχε χρησιμοποιηθεί για την παραγωγή μεγάλης κλίμακας μοντέλα των πόλεων. Απέδειξε με ένα πολύ εντυπωσιακό τρόπο, ότι η διαδικαστική μοντελοποίηση μπορεί επίσης να είναι μια σωστή προσέγγιση για τη μαζική μοντελοποίηση [Müller et al. 2006]. Πρόσφατα έργα αποδεικνύουν τη χρηστικότητα του σχήματος CGA. Όπως και πριν η χειροκίνητη επεξεργασία των κανόνων υπό μορφή κειμένου απαιτούνταν, ένα διαδραστικό πρόγραμμα επεξεργασίας έχει αναπτυχθεί, επιτρέποντας έτσι την εύκολη δημιουργία μοντέλων, χωρίς να χρειάζεται να αντιμετωπίσει κανείς τη σύνταξη γραμματικής [Lipp, 2007; Lipp et al., 2008]. Επιπλέον, υπάρχουν ήδη τεχνικές στην ανάπτυξη που επιτρέπει να εισαγάγει φυσικούς περιορισμούς σε μοντέλα (ειδικά σε κτίρια) που παράγονται από διαδικαστική μοντελοποίηση. Με βάση τις γραμματικές που επιτρέπουν την παραμετροποίηση των κανόνων παραγωγής, ένας αναλυτής στατικής υπολογίζει αυτές τις παραμέτρους που οδηγούν στο κατάλληλο σχήμα, ικανοποιώντας την επιθυμητή αρχιτεκτονική, καθώς και να είναι δομικά σταθερό, οι οποίες οδηγούν σε ακόμα καλύτερα αποτελέσματα όσον αφορά ρεαλιστικά και ορθά μοντέλα [Whiting et al. 2009].

Μερικές τεχνικές επικεντρώνονται περισσότερο σε δυσδιάστατες διατάξεις πόλεων από ότι σε τρισδιάστατα σχήματα κτιρίων. Οι [Chet et. al., 2008] επιτρέπουν στους χρήστες να επεξεργάζονται τους δρόμους μιας πόλης διαδραστικά με πεδία διάχυσης (tensor fields). Οι [Aliaga et. al., 2008] παράγουν σχεδιαγράμματα δρόμων χρησιμοποιώντας ένα παράδειγμα με βάση τη μέθοδο. Ένα σχετικό πεδίο της έρευνας είναι η αστική προσομοίωσης που προσπαθεί να κατανοήσει πώς οι διάφοροι παράγοντες που επηρεάζουν την ανάπτυξη των πόλεων και την ανάπτυξη στην πάροδο του χρόνου [Torrens P., 2001], [P. Waddell, 2002]. Οι πτυχές της αστικής προσομοίωσης έχουν χρησιμοποιηθεί σε διαδικαστικά μοντέλα για να παράγουν πιο ρεαλιστικά μοντέλα πόλεων [Weber B. et. al., 2009]. Άλλες τεχνικές έχουν σχεδιαστεί για τη μοντελοποίηση μικρότερων δομών. Οι [Legakis et al., 2001] προτείνουν μια μέθοδο για τον αυτόματο τρόπο εξωραϊσμού τρισδιάστατων επιφανειών με διάφορες κυτταρικές υφές όπως τούβλα, πέτρες και κεραμίδια. Οι [Cutler et al., 2002] ανέπτυξαν μια μέθοδο για τη μοντελοποίηση πολυεπίπεδων, στερεών μοντέλων με μια εσωτερική δομή. Μια άλλη μέθοδος έχει αναπτυχθεί για τη μοντελοποίηση δικτυωμάτων βελτιστοποιώντας τις θέσεις και τις δυνάμεις των δοκών και των αρθρώσεων [Smith J. et al. ,2002]. Οι [Pottmann et al., 2007], έχουν αναπτύξει αλγορίθμους που βασίζονται σε Διακριτή Διαφορική Γεωμετρία (Discrete Differential Geometry) που καθορίζουν με ποιο τρόπο γίνεται η οργάνωση δοκών και πάνελ από γυαλί, έτσι ώστε να σχηματίζουν το σχήμα μιας δεδομένης επιφανείας ελεύθερης σχεδίασης και πληρούν διάφορους γεωμετρικούς και φυσικούς περιορισμούς. Ένας άλλος τρόπος για να μοντελοποιήσεις αντικείμενα είναι να συνδυαστούν μαζί μέρη των υφιστάμενων μοντέλων αλληλεπιδραστικά [Funkhouser T. et. al., 2004]. Σε αυτή τη μέθοδο, ο χρήστης μπορεί να αναζητήσει μέσα από μια μεγάλη βάση δεδομένων τρισδιάστατων

μοντέλων για να βρει το επιθυμητό τμήμα, τότε κόβει το τμήμα έξω από το μοντέλο, και ράβει διάφορα τμήματα μαζί για να δημιουργήσουν ένα νέο αντικείμενο.

Η διαδικαστική μοντελοποίηση μπορεί να αντλήσει από ένα ευρύ φάσμα συστημάτων παραγωγής, όπως η Semi-Thue διαδικασίες [Davis et al., 1994], Chomsky γραμματικές [Sipser, 1996], γραμματικές γραφήματος (graph grammars) [Ehrig et al., 1999], γραμματικές σχήματος (shape grammars) [Stiny, 1975], και γραμματικές απόδοσης [Knuth, 1968]. Ωστόσο, η απλή προδιαγραφή ενός συστήματος παραγωγής είναι μόνο η βάση. Πολλά ζητήματα, όπως η γεωμετρική ερμηνεία, συνοπτική γραφή, τον έλεγχο της προέλευσης, καθώς και το σχεδιασμό των πραγματικών μοντέλων, πρέπει ακόμη να αντιμετωπιστούν.

Για τη γεωμετρική μοντελοποίηση των φυτών, [Prusinkiewicz & Lindenmayer, 1990] έδειξαν ότι θαυμάσια αποτελέσματα μπορούν να επιτευχθούν με τη χρήση L-συστημάτων για τη δημιουργία χορδών που ερμηνεύονται με LOGOstyle turtle μηχανισμό [Prusinkiewicz & Lindenmayer, 1990].

Στην αρχιτεκτονική, γραμματικές σχήματος [Stiny, 1975; Stiny, 1980], χρησιμοποιήθηκαν επιτυχώς για την κατασκευή και την ανάλυση του αρχιτεκτονικού σχεδιασμού [Downing & Flemming, 1981; Duarte, 2002; Flemming, 1987; Koning & Eizenberg, 1981; Stiny & Mitchell, 1978]. Η αρχική διατύπωση της γραμματικής σχήματος λειτουργεί άμεσα σε μια ρύθμιση χαρακτηρισμένα από γραμμές και σημεία. Ωστόσο, η παραγωγή είναι εγγενώς πολύπλοκη και συνήθως γίνεται με το χέρι, ή μέσω ηλεκτρονικού υπολογιστή, με έναν άνθρωπο να αποφασίζει για τους κανόνες που ισχύουν. Οι γραμματικές σχήματος είναι δυνατόν να απλοποιηθούν για να οριστούν γραμματικές [Stiny, 1980; Wonka et al., 2003] για να γίνουν πιο επιδεκτικές στην εφαρμογή υπολογιστή. Κυτταρικές υφές [Legakis et al., 2001] μπορούν να χρησιμοποιηθούν για να υπολογιστούν μοτίβα τούβλων και η παραγωγική μοντελοποίηση πλέγματος μπορεί να δημιουργήσει πολύπλοκες επιφάνειες από απλούστερες [Havemann, 2005]. Μόλις το πλαίσιο οριστεί από τη γραμματική, ένα ουσιαστικό μέρος της διαδικαστικής μοντελοποίησης, τότε είναι απαραίτητο να αποσπαστούν οι κανόνες που δημιουργούν αρχιτεκτονικές διαμορφώσεις. Ενώ μια μεγαλύτερη βιβλιοθήκη είναι αναγκαία για το έργο αυτό, προτείνεται να ξεκινήσει κάποιος με τα βιβλία που τονίζουν τη δομή της αρχιτεκτονικής, όπως a visual dictionary [Ching, 1996], "Η λογική της Αρχιτεκτονικής» από [Mitchell,1990], "Space Syntax " [Hillier, 1996], Design patterns [Alexander et al., 1977], studies of symmetry [March & Steadman, 1974; Shubnikov & Koptsik, 1974; Weyl 1952].

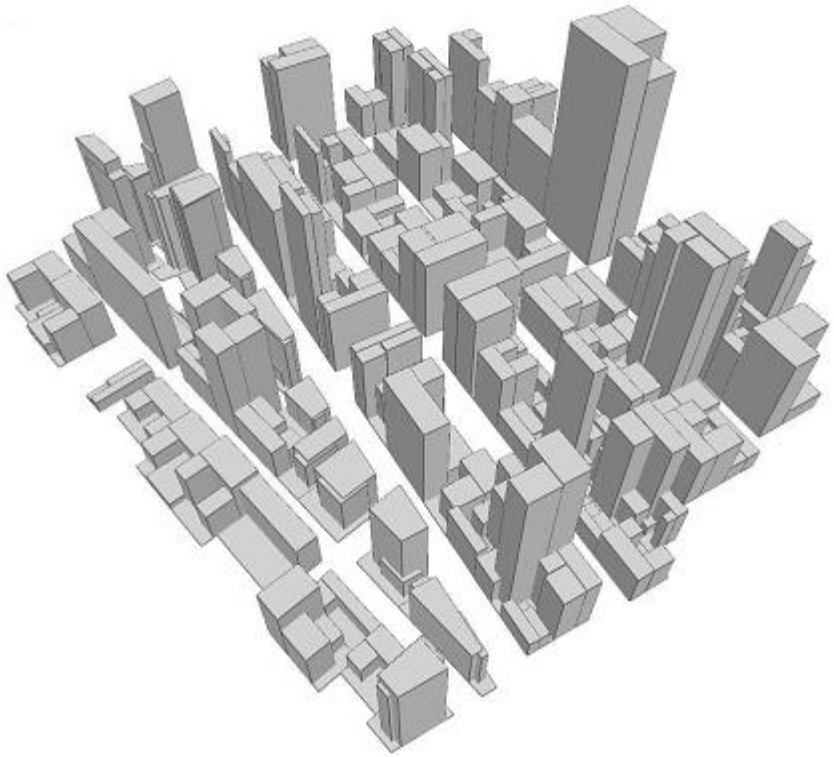
2.2 Το Κίνητρο για Διαδικαστική Μοντελοποίηση

Τα πρότυπα που αναμένονται από καλλιτέχνες μοντέλων σχετικά με την πολυπλοκότητα της εργασίας τους αυξάνονται συνεχώς. Όλο και περισσότερες λεπτομέρειες θα πρέπει να διαμορφωθούν έτσι ώστε να αξιοποιηθεί πλήρως το δυναμικό των σημερινών ηλεκτρονικών υπολογιστών και να παράγουν προηγμένες τεχνολογίες περιεχομένου για ταινίες και παιχνίδια υπολογιστών του αύριο. Εξυπακούεται, ότι κατά τη διάρκεια αυτής της τάσης το κόστος και η προσπάθεια που απαιτείται για να δημιουργηθεί αυτό το περιεχόμενο είναι παρομοίως αυξητικά. Έτσι τεχνικές που επιτρέπουν την αυτοματοποιημένη παραγωγή εξαιρετικά λεπτομερών μοντέλων απαιτούνται για να βοηθήσουν τους καλλιτέχνες για να συμβαδίσουν με την εξέλιξη αυτή. Υπάρχει μια ποικιλία από αυτές τις λεγόμενες διαδικαστικές τεχνικές μοντελοποίησης (π.χ. fractals ή grammars), οι οποίες αποτελούν σήμερα ένα πολύ ενδιαφέρον και συναρπαστικό πεδίο έρευνας.

Ίσως η πιο πολλά υποσχόμενη προσέγγιση είναι η χρήση των γραμματικών σχήματος (shape grammars) [Stiny, 1975], η οποία σε γενικές γραμμές δουλεύει σαν ένα L-system [Lindenmayer, 1968], με τη διαφορά ότι λειτουργεί σε γεωμετρικά σχήματα αντί για σειρές στοιχείων. Χρησιμοποιώντας μόνο ένα μικρό σύνολο των κανόνων παραγωγής, είναι δυνατόν να δημιουργηθούν μοντέλα με μια εξαιρετικά πολύπλοκη γεωμετρία. Έχει δειχθεί ότι η ρεαλιστικά μοντέλα πόλεων, υψηλών λεπτομερειών, μπορούν να παραχθούν από τη χρήση γραμματικών σχήματος (shape grammars) [Müller et. al., 2006]. Ωστόσο, υπάρχουν ορισμένα μειονεκτήματα στην προσέγγιση αυτή:

- Μη δυναμικά μοντέλα: Τα μοντέλα είναι στατικά και δεν είναι δυνατόν να προστεθούν σε αυτά κινούμενα μέρη επιτρέποντάς τα να εμφανίζονται σε διαφορετικές στάσεις.
- Αυτοματοποιημένη διαδικασία: Όσο ισχυρή αυτή η αυτοματοποιημένη μέθοδος αναπαραγωγής μοντέλων μπορεί να είναι, σε ορισμένες περιπτώσεις θέλουμε να επεξεργαστούμε τα αποτελέσματα με χειροκίνητο τρόπο. Φανταστείτε μια εξαιρετικά πολύπλοκη πόλη με χιλιάδες κτίρια, που δημιουργήθηκε από μια γραμματική σχήματος (shape grammar). Ίσως υπήρχαν αρκετοί κανόνες παραγωγής για να καθορίσουν περίπου 30 διαφορετικούς τύπους κτιρίων. Το αποτέλεσμα θα εξακολουθεί να μην είναι τόσο ικανοποιητικό όσο θα ήταν με μια συμβατική μέθοδο μοντελοποίησης. Αυτό που λείπει είναι ιδιαίτερες λεπτομέρειες και χαρακτηριστικά που κάνουν τα κτίρια μοναδικά. Για να δημιουργήσουμε ένα ρεαλιστικό μοντέλο, πιθανότατα θα μας άρεσε να αναμορφώσουμε ορισμένα σημεία εδώ και εκεί για να δώσει στο σύνολό του μια πιο πειστική εμφάνιση. Δυστυχώς μια τέτοια αναμόρφωση δεν είναι δυνατή χωρίς τη συγγραφή ενός μεγάλου αριθμού κανόνων παραγωγής και χάνοντας τα πλεονεκτήματα των διαδικαστικών μοντέλων.

- Δυσκολίες με την τελειοποίηση του μοντέλου: Μπορούμε να συγγράψουμε εύκολα κανόνες παραγωγής για τη δημιουργία μοντέλων υψηλής ευκρίνειας που αποτελείται από απλά αρχέτυπα. Αλλά μόλις θέλουμε να προσθέσουμε πολύ ομαλές λεπτομέρειες για το μοντέλο μας θα γίνει πολύ δύσκολο ή και αδύνατο να τις εκπροσωπήσει χρησιμοποιώντας μόνο τα βασικά αρχέτυπα. Χρειαζόμαστε τη δυνατότητα να εξομαλύνουμε τα μοντέλα με μια σχετικά απλή μέθοδο, αν θέλουμε να διατηρήσουμε τις δυνάμεις των γραμματικών σχήματος.



Εικόνα 3: Μοντέλα μαζικών κτιρίων που δημιουργήθηκαν με μόνο τέσσερις κανόνες (ξεκινώντας από την παρτίδα κτίριο ως αξίωμα). Πηγή: [Mueller et al. 2006]

2.3 Διαδικαστική Μοντελοποίηση Πόλεων

Οι διαδικαστικές μέθοδοι που περιγράφονται στην προηγούμενη ενότητα εφαρμόστηκαν ευρέως στην παραγωγή των φυσικών αντικειμένων και υφών. Μόνο πρόσφατα οι ερευνητές έστρεψαν την προσοχή τους στην εφαρμογή τους

στο πλαίσιο της γενιάς των τεχνητών φαινομένων, όπως μια αστική περιοχή. Η παραγωγή πόλης επιτυγχάνεται μέσω μιας σειράς σταδίων όπου το κάθε ένα χρησιμοποιεί μια σειρά από τεχνικές για τη δημιουργία δρόμων, εκτάσεις εδαφών, κτιριακή δομή και προσόψεις κτιρίου.

Τα οδικά δίκτυα είναι μια βασική πτυχή του χαρακτήρα της πόλης και της ταυτότητας της. Το οδικό δίκτυο είναι δύσκολο να το γενικεύσουμε, δεδομένου ότι αποτελούν συνυφασμένο συστατικό ενός σύνθετου συστήματος. Κατά την προβολή των οδικών δικτύων από ένα χάρτη ή ένα σχέδιο πόλης μπορούν να παρατηρηθούν μια σειρά από σχέδια. Είναι αυτά τα σχέδια που είναι το κλειδί για διαδικαστική παραγωγή που κωδικοποιούν τη δομή του οδικού δικτύου. Υπάρχουν πολλά πρότυπα οδικού δικτύου που έχουν αναπτυχθεί στις πόλεις που κυμαίνονται από το σφιχτά δομημένο σχέδιο δικτύου με κατακόρυφους σε μια κανονική δομή σκακιέρας στο ιεραρχικό δίκτυο με εκτεταμένους δευτερογενής και τριτογενής δρόμους τροφοδοσίας σε οδικές αρτηρίες σε ένα κλάδο παρόμοιου με το σύστημα. Τα πρότυπα που εφαρμόζονται σε μια πόλη είναι αποτέλεσμα πολλών παραγόντων, συμπεριλαμβανομένης της τοποθεσίας, της γεωγραφίας, των πολιτισμικών επιρροών, των τάσεων σχεδιασμού, κλπ. Οι πόλεις μπορούν να ταξινομηθούν από τα οδικά σχέδια που περιέχουν: Οι σύγχρονες πόλεις των ΗΠΑ όπως η Νέα Υόρκη είναι διατεταγμένες σε ένα μοτίβο σκακιέρας ή ψηφιδωτού, ορισμένες ευρωπαϊκές πόλεις όπως το Παρίσι είναι δομημένες με μια ακτινική ή ομόκεντρου μοτίβου. Ωστόσο, οι περισσότερες πόλεις περιέχουν μια σειρά από σχέδια, με διαφορετικά πρότυπα που επικρατούν σε διάφορες περιοχές ή γειτονιές μέσα στην πόλη [Kevin Lynch, 1960] [Alexander C. et. al., 1977].

Τα κτίρια της πόλης είναι δύσκολα αντικείμενα για να δημιουργήσει κάποιος διαδικαστικά λόγω της ατομικότητας τους. Τα κτίρια που υπάρχουν σε μια σύγχρονη πόλη εμφανίζουν ένα ευρύ φάσμα τόσο στην λειτουργικότητα τους όσο και στο στυλ τους. Τα κτίρια ως λειτουργικές μονάδες εξυπηρετούν ένα συγκεκριμένο σκοπό ή ρόλο σε κάθε γειτονιά, δήμο, περιφέρεια και πόλη. Ο αριθμός των ρόλων για τα κτίρια είναι πολλές και σε συνδυασμό με τη γεωγραφική σύσταση μέσα σε μία πόλη το κάνουν ένα εξαιρετικά πολύπλοκο σύστημα. Ένα τέτοιο πολύπλοκο σύστημα είναι δύσκολο να μοντελοποιηθεί αλλά μια απλοποιημένη λύση μπορεί να χρησιμοποιηθεί, παρόμοια με εκείνη που χρησιμοποιείται στην στατιστική ανάλυση, η οποία χρησιμοποιεί κλάσεις ή ομάδες για να διαμορφώσει μια λειτουργία κτιρίου. Ομάδες χρήσης, όπως εμπορική, οικιακή και βιομηχανική μπορούν να χρησιμοποιηθούν ως επιλεγμένες γενικεύσεις για τους πολυάριθμους ρόλους κτιρίου και ένας απλός μηχανισμός για τη λειτουργία μοντελοποίησης εντός των πόλεων. Το ύφος του κτιρίου και συγκεκριμένα η γεωμετρία του και τα υλικά του είναι συχνά το αποτέλεσμα των πολυάριθμων αρχιτεκτονικών και πολιτιστικών επιρροών. Μια τέτοια σύνθετη μορφή είναι δύσκολο να διαμορφωθεί και μία προσέγγιση ή μια ουσιαστική μείωση μοντέλου απαιτείται για να περιορίσει την πολυπλοκότητα του συστήματος παραγωγής.

Για να αξιολογηθούν αποτελεσματικά τα συστήματα παραγωγής έχουμε εντοπίσει έναν αριθμό κριτηρίων [George Kelly & Hugh McCabe, 2006]:

1. **Ρεαλισμός** - Μήπως η παραγόμενη πόλη μοιάζει με μια πραγματική πόλη;
2. **Κλίμακα** - Είναι το αστικό τοπίο στην κλίμακα μιας πόλης;
3. **Διακύμανση** - Μπορεί το σύστημα παραγωγής της πόλης να αναδημιουργήσει τη διακύμανση των οδικών δικτύων και των κτιρίων που βρέθηκαν σε πραγματικές πόλεις ή η έξοδος του συστήματος παραγωγής είναι ομοιογενής (για κάθε παραγόμενο στοιχείο);
4. **Είσοδος** - Ποια είναι τα ελάχιστα στοιχεία εισόδου που απαιτείται για την παραγωγή βασικών στοιχείων εξόδου και ποια δεδομένα εισόδου απαιτούνται για την καλύτερη απόδοση;
5. **Αποτελεσματικότητα** - Πόσο χρόνο παίρνει για να δημιουργήσει τα παραδείγματα που φαίνονται και σε τι υλικό υπολογιστών παράγονται; Πόσο υπολογιστικά αποτελεσματικός είναι ο αλγόριθμος;
6. **Έλεγχος** - Μπορεί ο χρήστης να επηρεάσει την παραγωγή πόλης και να λαμβάνει άμεση ανατροφοδότηση για τις ενέργειές του; Υπάρχει μια χειροκίνητη μέθοδος ελέγχου διαθέσιμη ή ο έλεγχος περιορίζεται; Σε ποιο βαθμό μπορεί ο χρήστης να επηρεάσει τα αποτελέσματα παραγωγής;
7. **Πραγματικό χρόνο** - Μπορεί η παραγόμενη πόλη να προβληθεί σε πραγματικό χρόνο; Υπάρχουν κάποιες τεχνικές βελτιστοποίησης απόδοσης που μπορούν να εφαρμοστούν για να ενεργοποιήσει τον πραγματικό χρόνο εξερεύνησης;

Στα συμπεράσματά μας θα εξετάσουμε και θα αξιολογήσουμε το μοντέλο παραγωγής ενός μικρού τμήματος της πόλης της Κέρκυρας με βάση τα παραπάνω κριτήρια: Ρεαλισμού, Κλίμακας, Διακύμανσης, Εισόδου, Αποτελεσματικότητας, Ελέγχου και Πραγματικού Χρόνου.

2.4 Διαδικαστική Παραγωγή Κτιρίων

Περισσότερα περιβάλλοντα πραγματικού κόσμου περιλαμβάνουν κάποιο είδος από τεχνητές δομές. Σε ένα υπαίθριο περιβάλλον αυτές οι τεχνητέ δομές μπορούν να είναι διάσπαρτα σπίτια που φτιάχνουν μόνο ένα μικρό κομμάτι της διακόσμησης του εδάφους, με τη πλειονότητα των διακοσμήσεων να είναι φυτά, ενώ σε αστικά περιβάλλοντα αυτό θα μπορούσε να αντιστραφεί με κτίρια προσφέροντας μια πλειοψηφία από διακοσμήσεις ενός εικονικού κόσμου. Αν το επίπεδο λεπτομέρειας (LOD) που απαιτείται για κτίρια είναι κατ' αναλογία χαμηλό, καθώς θα μπορούσε να είναι η περίπτωση για ένα έργο προσομοίωσης μιας πτήσης που απεικονίζει έναν εικονικό κόσμο από ένα υψηλό υψόμετρο, τότε απλά γεωμετρικά σώματα μπορούν να δημιουργήσουν επαρκή αποτελέσματα αν συνδυαστούν με κατάλληλους χάρτες υφής που κρύβουν την έλλειψη πραγματικής λεπτομέρειας στη γεωμετρία. Η χρήση των γραμματικών κανόνων

τμηματοποίησης (split grammar rules) [Wonka et. al. 2003] και των γραμματικών κανόνων σχήματος (shape grammar rules) [Müller et. al 2006] για την περιγραφή αρχιτεκτονικών χαρακτηριστικών επιτρέπουν τη χρήση περισσότερων πολύπλοκων σχημάτων και κτιριακών δομών, τα οποία μπορούν να παρουσιάζουν πολυπλοκότητα στη λεπτομέρειά τους [Havemann 2005].

Στο μέγιστο επίπεδο λεπτομέρειας (LOD), ακόμα και το εσωτερικό τμήμα των κτιρίων μπορούν να παραχθούν [Hahn et. al. 2006]. Η τοποθέτηση αυτών των τεχνητών δομών σε έναν εικονικό κόσμο μπορεί να φτάσουν σε μεγάλα επίπεδα πολυπλοκότητας, αν τα κτίρια αυτά συγκροτήσουν μέρος ενός αστικού περιβάλλοντος [Greuter et. al. 2003]. Αυτοί οι πιο πολύπλοκοι διακανονισμοί δημιουργούνται από μια σειρά βημάτων: (α) *πρώτα, παράγεται ένα κατάλληλο οδικό δίκτυο, παρέχοντας αποτελεσματικά χάρτες δρόμων που κατανέμουν το έδαφος και περιορίζουν την τοποθέτηση των κτιρίων.* (β) *Αυτό στη συνέχεια χρησιμοποιείται για να κατευθύνει τη διαίρεση του εδάφους σε τμήματα τα οποία μπορούν να κατανεμηθούν περαιτέρω για να δημιουργήσουν αποτυπώματα κτιρίων,* (γ) *και τα οποία στη συνέχεια χρησιμοποιούνται ως εισροές για την παραγωγή των ίδιων των κτηρίων.*

Λόγω των χρονικών περιορισμών, οι προσπάθειες στην εφαρμογή ενός πιο σύνθετου αστικού συστήματος παραγωγής έπρεπε να απλουστευθεί. Η μέθοδος διανομής κτιρίων ήταν συνεπώς σχεδόν ταυτόσημη με τη μέθοδο κατανομής των φυτών, με μερικές διακριτές αλλαγές. Πρώτα, τα μοντέλα των κτιρίων είχαν προσαρμοστεί ώστε να έχουν «θεμέλια», ή υπόγεια επίπεδα. Ο σκοπός αυτού ήταν να αποφευχθεί η κάτω πλευρά του μοντέλου που παρουσιάζεται, το κτίριο θα πρέπει να είναι τοποθετημένο σε μια κλίση. Δεύτερον, αντί να περιστρέφεται και να διαβαθμίζεται, το οποίο θα ήταν ακατάλληλο για την πλειοψηφία των κτιρίων, οι δομές ορίστηκαν με τυχαίες τιμές ‘ύψους’ και ‘προέκτασης’ που αντέγραψαν τμήματα του μοντέλου του κτιρίου πάνω ή προς την πλευρά του πρωτοτύπου, ο σκοπός είναι να μειωθεί η επανάληψη και να μειωθεί το απομονωμένη αίσθηση που μπορεί να συσχετιστεί με μοναχικά κτίρια.

Τα αποτελέσματα ήταν αποδεκτά, και θα είναι ιδιαίτερα επίκαιρο, αν εφαρμοζόταν σε ένα μικρό οικισμό του μεγέθους ενός χωριού, αλλά και το περιβάλλον στο σύνολό του δεν είχε τη δομή και την πυκνότητα που σχετίζεται με τις αστικές περιοχές. Η λύση στο πρόβλημα αυτό θα ήταν ο επανασχεδιασμός του συστήματος τοποθέτησης, και, ενδεχομένως, του συστήματος παραγωγής εδάφους, από την αρχή, λαμβάνοντας παράλληλα υπόψη τις αρχιτεκτονικές γραμματικές σχήματος και οδικών δικτύων που χρησιμοποιούνται κατά τις προηγούμενες εφαρμογές παραγωγής μιας πόλης.

2.5 Διαδικαστική Παραγωγή και Τοποθέτηση Βλάστησης

Υπάρχουν διαφορετικοί μέθοδοι για τη διαδικαστική μοντελοποίηση της βλάστησης, πολλές εκ των οποίων βασίζονται σε μορφοκλασματικές (fractal) ή απλούστερες βασισμένες σε κανόνες τεχνικές. Μια από τις τελευταίες μεθόδους που έχουν χρησιμοποιηθεί σε πραγματικού χρόνου παραγωγές δασών για εικονικά περιβάλλοντα πραγματικού χρόνου [Di Giacomo et. al. 2001], χρησιμοποιώντας έναν σκελετό τοπολογίας για διαδικαστικά παραγωγίσιμα και κινούμενα δέντρα, που έχει επίσης συνδυαστεί αποτελεσματικά με πραγματικού χρόνου παραγωγίσιμο γρασίδι δημιουργώντας ένα πλούσιο φυσικό τοπίο [Guerraz et. al 2003]. Μια πιο ισχυρότερη, βασισμένη σε κανόνες τεχνική εφαρμόζει μοντελοποίηση βασισμένη σε συστατικά (component-based) είναι η μόνη από [Lintermann B. & Deussen O., 1999], η οποία παρέχει ένα πιο έξυπνο τρόπο για τον έλεγχο της μοντελοποίησης των φυτών από τα γνωστά L-systems [Prusinkiewicz P. & Lindenmayer A., 1990]. Η απόφαση για το ποιο είδος φυτών χρειάζεται να τοποθετηθεί στον εικονικό κόσμο συνήθως εξαρτάται από τον αριθμό των συντελεστών, συμπεριλαμβανομένου του υψόμετρου και της κλίσης του εδάφους, όπως επίσης και τα χαρακτηριστικά της τοπογραφίας όπου υπαγορεύουν την πιθανότητα εμφάνισης ενός συγκεκριμένου είδους φυτού [Wells W.D. 2005]. Μόλις μια θέση στο παραγωγίσιμο έδαφος έχει αποφασιστεί, η παραγωγή των μοντέλων των φυτών μπορούν να ακολουθηθούν με την τοποθέτηση της βλάστησης.

Για την απόδειξη της έννοιας της εφαρμογής, μια απλοποιημένη μέθοδος εφαρμόστηκε που έκανε χρήση της τυχαίας θέσης των μοντέλων βλάστησης, αντί της τυχαίας βλάστησης. Μια σειρά από χαμηλά πολύγωνα δέντρων και θάμνων δημιουργήθηκαν και εξήχθησαν σαν τρισδιάστατα μοντέλα, τα οποία μετά φορτώθηκαν κατά την έναρξη του προγράμματος. Μόλις το έδαφος είχε δημιουργηθεί, η βλάστηση τυχαία ορίστηκε σε διάφορα σημεία στο πλέγμα εδάφους. Ωστόσο, αν ένα συγκεκριμένο μέρος του εδάφους ήταν τόσο ψηλό, χαμηλό, βυθισμένο στο νερό, ή είχε ένα μεγάλο βαθμό κλίσης, τότε αυτή η περιοχή θα απορριπτόταν και θα αγνοούνταν, ο σκοπός είναι να αποφθεχθεί η εμφάνιση της βλάστησης σε μια μη ρεαλιστική περιοχή. Για να ελαττώσουμε το πρόβλημα της επανάληψης, η βλάστηση περιστράφηκε και κλιμακώθηκε από τυχαίες τιμές. Το αποτέλεσμα ήταν εκπληκτικά αποτελεσματικό. Τα φυτά εμφανίζονται να έχουν στοχαστικές ιδιότητες παρόλο που είναι προκαθορισμένα μοντέλα, και αυτό το αποτέλεσμα ήταν εφικτό όταν η πυκνότητα της βλάστησης αυξήθηκε στο επίπεδο του δάσους, όταν η επανάληψη έγινε αξιοσημείωτη.

2.6 Μοντελοποίηση Εμφάνισης και Συμπεριφοράς Αστικών Χώρων

Μοντελοποιώντας την εμφάνιση και τη συμπεριφορά των αστικών χώρων είναι μια μεγάλη πρόκληση. Ένας αστικός χώρος είναι μια σύνθετη συλλογή αρχιτεκτονικών

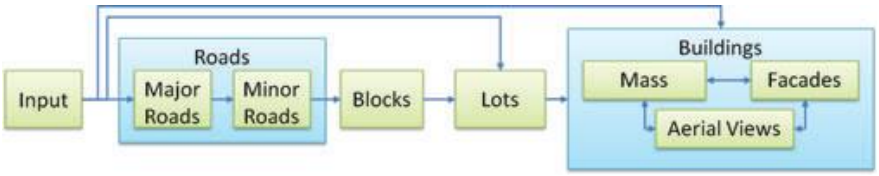
δομών τοποθετημένη σε αγροτεμάχια, κτίρια,, τετράγωνα και γειτονιές που συνδέονται μεταξύ τους με δρόμους. Η κατανόηση, η περιγραφή και την η πρόβλεψη της εμφάνισης (π.χ. δημιουργία 2D/3D γεωμετρικών μοντέλων) και της συμπεριφοράς (π.χ. προσομοίωση αστικής ανάπτυξης στη πάροδο του χρόνου) των πόλεων είναι χρήσιμη σε έναν αυξανόμενο αριθμό εφαρμογών. Παραδοσιακά, η μοντελοποίηση αστικών χώρων υπήρξε μια αρκετά χειρωνακτική εργασία που κατανάλωνε σημαντικές ποσότητες πόρων. Με τις αυξανόμενες απαιτήσεις ποσότητας και ποιότητας στο αστικό περιεχόμενο, υπάρχει επιτακτική ανάγκη για εναλλακτικές λύσεις που επιτρέπουν την γρήγορη, ημιαυτόματη μοντελοποίηση αστικών χώρων. Οι αστικές μέθοδοι μοντελοποίησης είναι σημαντικές σε έναν αυξανόμενο αριθμό εφαρμογών. Ορισμένες από αυτές είναι [Vanegas C. et. al., 2009]:

- **Χαρτογράφηση και οπτικοποίηση:** Ανοικοδόμηση υφιστάμενων αστικών χώρων για τη χαρτογράφηση και για εργαλεία πλοήγησης, οπτικοποίηση ήδη υπαρχόντων πόλεων για τις οποίες υπάρχουν μόνο μερικά δεδομένα και επιτρέπει στους αρχιτέκτονες να απεικονίσουν μια νέα πόλη.
- **Διασκέδαση:** Ταχύς παραγωγή λεπτομερούς ψηφιακού περιεχομένου για τη συμπλήρωση αστικών περιοχών σε βιντεοπαιχνίδια και ταινίες.
- **Αντιμετώπιση καταστάσεων έκτακτης ανάγκης:** Δημιουργία μοντέλων για να εκπαιδεύσει το προσωπικό για την αντιμετώπιση καταστάσεων έκτακτης ανάγκης σε τρέχουσα και υποθετικά αστικά σχεδιαγράμματα, συμπεριλαμβανομένου του σχεδιασμού των διαδρομών εκκένωσης για διάφορες καταστροφές, και προτείνοντας αναπτύξεις έκτακτης ανάγκης των πόρων.
- **Πολεοδομία:** Πρόβλεψη αποτελεσμάτων των πολιτικών χρήσης γης και η επίδρασή τους στις υπάρχουσες γειτονιές, και τη δημιουργία υποθετικών απόψεων ενός αστικού χώρου μετά την εφαρμογή της επεξεργασίας και την ανάπτυξη αλγορίθμων.

Οι αστικοί χώροι είναι δύσκολο να μοντελοποιηθούν, διότι η βασική δομή καθορίζεται από έναν πολύ μεγάλο αριθμό δύσκολο να υπολογισθούν ποσοτικά μεταβλητές, συμπεριλαμβανομένων των πολιτικών της γης, η συμπεριφορά της αγοράς, των υποδομών μεταφορών, των κυβερνητικών σχεδίων και πληθυσμιακών αλλαγών. Επιπλέον, πυκνά αστικά περιβάλλοντα είναι ιδιαιτέρως πολύπλοκο να μοντελοποιηθούν, διότι είναι ταυτόχρονα πυκνά και μεγάλα, που εκτείνονται από λίγα έως εκατοντάδες τετραγωνικά χιλιόμετρα.

Στην παρούσα πτυχιακή εργασία παρουσιάζεται η αναπαράσταση κτιρίων της Κέρκυρας κατά το 18^{ο} και 19^{ο} αιώνα, σε ένα αστικό χώρο μέσα στο κέντρο της πόλης. Η μέθοδος που χρησιμοποιείτε αναπαραστατικά είναι παρόμοια με τη μέθοδο που παρουσιάστηκε σε μια πρόσφατη μελέτη που έγινε πριν μερικά χρόνια [C. A. Vanegas et. al, 2009] για τη δημιουργία μοντέλων αστικών χώρων. Η μελέτη αυτή βασίστηκε (i) στα μέρη που αποτελούν τον αγωγό μοντελοποίησης αστικών χώρων (Εικόνα 4) και (ii) στη διάσταση των γεωμετρικών οντοτήτων που

παράγουν. Ο Πίνακας 1 παρουσιάζει συνοπτικά τις μεθόδους που αναφέρονται στη παρούσα εργασία.



Εικόνα 4: Γενικός αγωγός για τη μοντελοποίηση αστικών χώρων.

Πίνακας 1: Μέθοδοι αστικής μοντελοποίησης με τα αντίστοιχα δεδομένα εισόδου και εξόδου, ομαδοποιούνται ανάλογα με το κύριο αστικό πρόβλημα μοντελοποίησης που θα αντιμετωπίσουμε.

Method	Input	Output
Layout modeling		
Aliaga *et al.* (2008)	Example urban layout and aerial imagery	New road network and imagery with similar style
Building modeling		
Hahn *et al.* (2006)	Building exterior, generation rules	3D geometry of building interior
Liu *et al.* (2006)	Architectural mesh	Quad dominant mesh, offset mesh
Müller *et al.* (2006)	Shape grammar rules	3D building and facade model
Merrell, Manocha (2008)	3D polygonal object	Larger collection of similar objects
Cabral *et al.* (2009)	Mesh and textures, user interaction	Reshaped mesh and textures
Facade modeling		
Legakis *et al.* (2001)	3D polygonal mesh	Detailed 3D texture on the mesh
Wonka *et al.* (2003)	Shape grammar rules	3D facade model
Müller *et al.* (2007)	Facade image, some user interaction	3D facade model, grammar for the facade

Ο αγωγός αστικής μοντελοποίησης αποτελείται περίπου από την παραγωγή ενός οδικού δικτύου, κατανέμοντας τα μπλοκ που προέρχονται από το οδικό δίκτυο σε τμήματα, και δημιουργώντας ένα κτίριο μέσα σε κάθε παρτίδα. Διαφορετικές είσοδοι έχουν χρησιμοποιηθεί σε αυτή τη διαδικασία συμπεριλαμβανομένων των αρχιτεκτονικών σχεδίων στόχου, παράδειγμα 3D μοντέλων ή εικόνων, των κοινωνικοοικονομικών ομάδων δεδομένων και πεδία τανυστών (tensor fields).

2.6.1 Διαδικαστική Μοντελοποίηση

Η μοντελοποίηση των αστικών δομών έχει εκτελεστεί χρησιμοποιώντας διάφορες προσεγγίσεις συμπεριλαμβανομένων των διαδικαστικών μοντέλων, μεθόδων σύνθεσης και άλλων ημι-αυτόματων μηχανισμών δημιουργίας. Όπως είπαμε και παραπάνω, η διαδικαστική μοντελοποίηση έχει χρησιμοποιηθεί για να αυτοματοποιήσει την παραγωγή πολύπλοκων αστικών δομών, συμπεριλαμβανομένων των κτιρίων και σπιτιών, για την παραγωγή ψηφιακού περιεχομένου από ένα σχετικά απλό σύνολο παραμέτρων και κανόνων (Εικόνα 5). Οι μέθοδοι σύνθεσης έχουν επεκτείνει την έννοια της σύνθεσης υφής σε 3D. Επιπλέον, οι διάφορες μορφές της διαδραστικής επεξεργασίας έχουν προταθεί για

να επεκτείνουν περαιτέρω τις παραπάνω προσεγγίσεις και να αυξήσουν την ευκολία χρήσης.

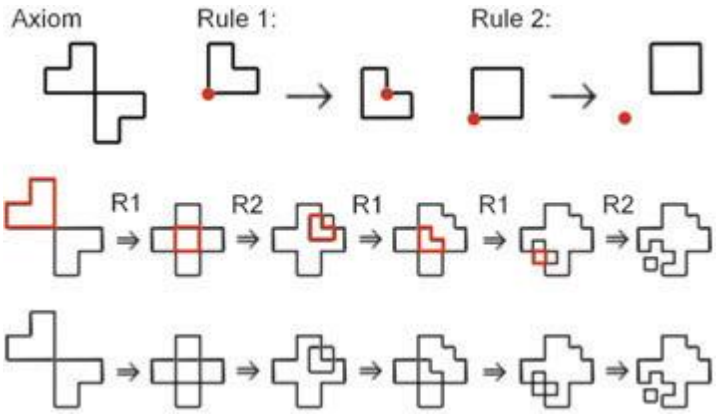
Εικόνα 5: Διαδικαστικά παραγόμενα μοντέλα αστικών χώρων.

Όπως ειπώθηκε και αρχικά σε αυτό το κεφάλαιο, η μοντελοποίηση διαδικαστικών αρχιτεκτονικών μπορεί να χρησιμοποιήσει ένα από τα πολλά συστήματα παραγωγής, όπως η Semi-Thue διαδικασίες [Davis et al., 1994], Chomsky γραμματικές [Sipser, 1996], γραμματικές γραφήματος (graph grammars) [Ehrig et al., 1999], γραμματικές σχήματος (shape grammars) [Stiny, 1975], και γραμματικές απόδοσης [Knuth, 1968]. Μία παράμετρος που καθοδηγεί την επιλογή του συστήματος παραγωγής είναι η εκφραστικότητα. Το πρώτο βασικό ερώτημα είναι λοιπόν: πόσα διαφορετικά κτίρια μπορούν να μοντελοποιηθούν; Αν η εκφραστικότητα είναι το κύριο κριτήριο θα είναι δυνατόν να αρχίσει η μοντελοποίηση μόνο με χρήση της C + + ή μηχανές Turing. Αυτό θα μας επέτρεπε να υπολογίσουμε όλους τους τύπους της αρχιτεκτονικής και η μοντελοποίηση θα ήταν επίσης πολύ ευέλικτη. Ωστόσο, υπάρχει και το ζήτημα της αποτελεσματικότητας: πόσο αποτελεσματικά μπορεί ένας σχεδιαστής έργου να δουλέψει με το πλαίσιο εργασίας; Αυτό το δεύτερο ερώτημα κάνει τις απλοποιήσεις και τους περιορισμούς πιο ελκυστικούς. Σε αντίθετη περίπτωση, η πολυπλοκότητα του διαδικαστικού μοντέλου μπορεί γρήγορα να ξεφύγει του χεριού και ο σχεδιασμός να γίνεται συχνά ασταθής.

Στην αρχιτεκτονική, ο Stiny πρωτοστάτησε στην ιδέα των γραμματικών σχήματος [Stiny,G. (1975); Stiny,G. (1980)]. Αυτές οι γραμματικές σχήματος χρησιμοποιήθηκαν επιτυχώς για την κατασκευή και την ανάλυση του αρχιτεκτονικού σχεδιασμού [Downing, F. & Flemming, U. (1981); Duarte, J. (2002); Flemming, U. (1987); Koning, H. & Eizenberg, J. (1981); Stiny, G. & Mitchell, W. J. (1978)]. Η αρχική διατύπωση της γραμματικής σχήματος λειτουργεί άμεσα σε μια ρύθμιση των επισημασμένων γραμμών και σημείων. Κανόνες παραγωγής μπορούν ουσιαστικά να μοντελοποιηθούν με τη σχεδίαση γραμμών και σημείων

και με την επισήμανση τους. Στην πράξη αυτό οδηγεί σε ένα πρόβλημα παραγωγής, διότι σε κάθε βήμα της επανάληψης υπάρχουν συνήθως πολλοί διαφορετικοί μετασχηματισμοί σύμφωνα με τους οποίους ένας κανόνας μπορεί να εφαρμοστεί και επιπλέον υπάρχουν πολλοί διαφορετικοί κανόνες για να επιλεχθούν. Κλασική γραμματικές σχήματος συνεπώς έχουν ένα κομμάτι που λείπει και που αποτρέπει την αυτόματη παραγωγή: ένας μηχανισμός ελέγχου που επιλέγει ποιός κανόνας να ισχύει και υπό ποιό μετασχηματισμό. Στην Εικόνα 6, δείχνουμε ένα παράδειγμα μιας γραμματικής σχήματος και ένα παράδειγμα παραγωγής.

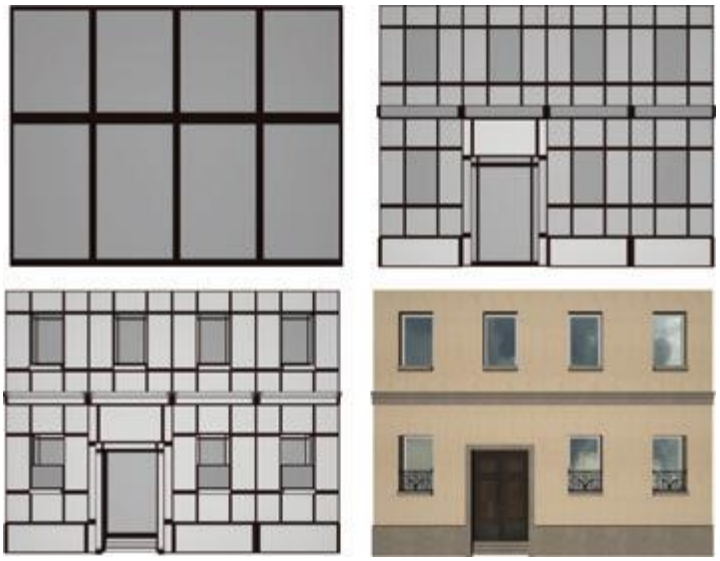
Για να γίνει η έννοια της γραμματικής σχήματος πιο εφαρμόσιμη σε γραφικά ηλεκτρονικών υπολογιστών, Wonka [Wonka et al. 2003] και Müller [Müller et al. 2006] εισήγαγαν ένα πλαίσιο που περιλαμβάνει κανόνες για την αντικατάσταση σχημάτων με μηδέν, ένα ή πολλαπλά άλλα σχήματα, καθώς και μηχανισμούς για να καθορίσουν μια αυτόματη παραγωγή κανόνα. Μόνο η αυτόματη παραγωγή κανόνα επιτρέπει μεγάλης κλίμακας διαδικαστικής μοντελοποίησης. Το πλαίσιο του Müller [Müller et al. 2006] επίσης αναπτύχθηκε περαιτέρω στο εμπορικό λογισμικό CityEngine [Procedural:www.procedural.com, 2008]. Η αρχική γραμματική Wonka [Wonka et al. 2003] εστιάστηκε κυρίως στο μέγεθος ανεξάρτητων κανόνων σχεδιασμού των προσόψεων με πράξεις διάσπασης.



Εικόνα 6: *Γραμματική σχήματος* . Αυτό το σχήμα δείχνει ένα παράδειγμα μιας παραδοσιακής γραμματικής σχήματος. Η γραμματική σχήμα λειτουργεί σχετικά με τις ρυθμίσεις των γραμμών. Η πρώτη γραμμή της εικόνας δείχνει τους δύο κανόνες της γραμματικής σχήμα. Η κόκκινη κουκίδα χρησιμοποιείται για να υποδηλώσει ένα σύστημα συντεταγμένων αναφοράς, έτσι ώστε να οριστεί η μετάθεση των σχημάτων που χρησιμοποιούνται στον κανόνα . Η δεύτερη και η τρίτη σειρά δείχνουν μια παραγωγή ενός νέου σχήματος (δεξιά) ξεκινώντας από ένα αξίωμα (το αρχικό σχήμα) που φαίνεται στα αριστερά. Οι δύο κανόνες που αναφέρονται: κανόνας ένα και κανόνας. Ποιός κανόνας επιλέχθηκε για ένα βήμα παραγωγή είναι γραμμένο πάνω από το βέλος: R1 σημαίνει ένας κανόνας και R2 σημαίνει κανόνας . Οι κόκκινες γραμμές στη μεσαία σειρά τονίζουν το επιμέρους σχήμα που επιλέγεται για αντικατάσταση από το γραμματική.

Η λειτουργία τμηματοποίησης επιτρέπει την διάσπαση στοιχειωδών σχημάτων (όπως κύβοι και κύλινδροι) κόβοντας τα στοιχειώδη σχήματα κατά μήκος των επιπέδων διαχωρισμού . Το μέγεθος ανεξάρτητο από τους κανόνες σχεδιασμού επιτρέπει στο σχεδιαστή να καθορίσει πώς η θέση ενός επιπέδου διαχωρισμού θα πρέπει να αλλάξει όταν το μέγεθος των στοιχειωδών αλλάζει σχήμα και πόσα

επίπεδα διάσπασης θα πρέπει να χρησιμοποιούνται για ένα σχήμα από ένα ορισμένο μέγεθος. Στην Εικόνα 7, δείχνουμε ένα παράδειγμα της αρχικής γραμματικής τμηματοποίησης που προτάθηκε από τον Wonka [Wonka et al., 2003].



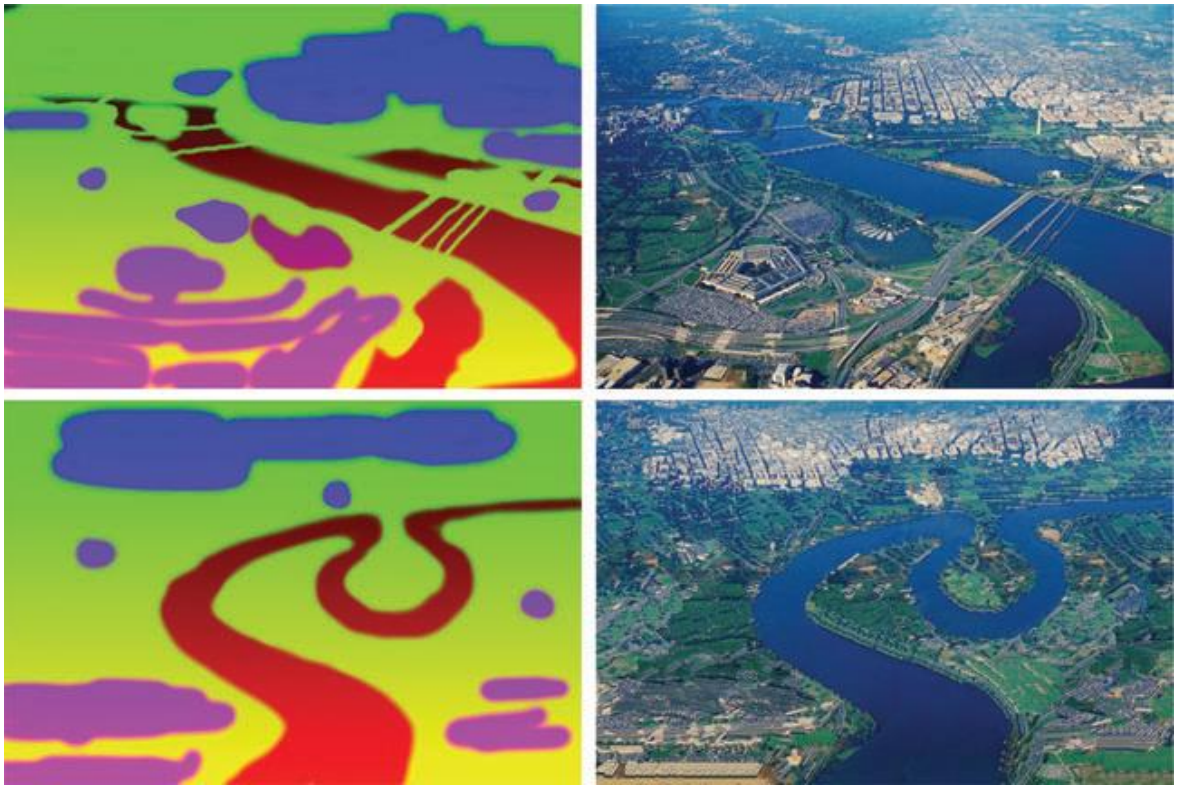
Εικόνα 7: Γραμματική Διάσπασης (split grammar) [Wonka et al., 2003]. Ένα παράδειγμα παραγωγής μιας μικρής πρόσοψης με τους κανόνες διάσπασης. Σημειώστε ότι το τελικό μοντέλο είναι τρισδιάστατο.

2.6.2 Οδικό Δίκτυο και Μοντελοποίηση Διάταξης

Η μοντελοποίηση των δρόμων και η διάταξή τους έχει επικεντρωθεί κυρίως στη δημιουργία εύλογων αεροφωτογραφιών και οδικών δικτύων για αστικούς χώρους. Παρόμοια με τη σύνθεση μοντέλου, η δημιουργία αεροφωτογραφιών στηρίζεται από την έννοια της σύνθεσης υφής (π.χ. [Wei L.-Y, 2009]), αλλά εκτός από την σύνθεση των πίξελ-δεδομένων, συναφή διανυσματικά δεδομένα συντίθεται επίσης. Ο σχεδιασμός των δρόμων προσπαθεί να μιμηθεί το οπτικό στυλ των δικτύων δρόμου σε πραγματικό κόσμο αστικών χώρων.

Ο Hertzmann [Hertzmann et al. 2001] παρουσιάζει ένα διφασικό πλαίσιο σχεδιασμού για την επεξεργασία εικόνας που μπορούν να εφαρμοστούν άμεσα στην σύνθεση υφής και εφαρμόζεται σε αστικές αεροφωτογραφιών. Στην πρώτη φάση, ένα ζευγάρι εικόνων, με μία εικόνα φερόμενη να είναι μια φιλτραρισμένη έκδοση του άλλου, παρουσιάζεται ως δεδομένα εκπαίδευσης. Στη δεύτερη φάση, το φίλτρο εφαρμόζεται σε κάποια νέα εικόνα-στόχο για να δημιουργήσει ένα ανάλογο φιλτραρισμένο αποτέλεσμα Αυτή η μέθοδος υποστηρίζει μια ευρεία ποικιλία της εικόνας εφέ φίλτρου, συμπεριλαμβανομένων των παραδοσιακών φίλτρων εικόνας, της υπερ-ανάλυσης, τη βελτίωση της της σύνθεσης υφής και την υφή-με αριθμούς (texture-bynumbers). Στην τελευταία από αυτές τις εφαρμογές, οι ρεαλιστικές σκηνές αποτελούνται από μια ποικιλία υφών που δημιουργήθηκαν χρησιμοποιώντας μια διεπαφή ζωγραφικής. Νέες εικόνες συντίθεται με την

εφαρμογή των στατιστικών μιας επισημασμένης εικόνας παραδείγματος σε μια προσφάτως επισημασμένη εικόνα. Ένα ενδιαφέρον παράδειγμα αυτών των έργων αποτελείται από τη σύνθεση μιας πανοραμικής θέας της πόλης, όπως δείχθηκε από τους συγγραφείς στην εργασία τους (Εικόνα 8).



Εικόνα 8: Αναλογίες εικόνας [Hertzmann et al. 2001]. Χρήση αναλογίες εικόνας για τη σύνθεση νέων εναέριων απόψεων αστικών χώρων από το παράδειγμα. Συνήθης σύνθεση υφής δεν μπορεί να αναπαράγει το έδαφος στη φωτογραφία, διότι δεν είναι στάσιμο: μακριά στοιχεία είναι διαφορετικά από κοντινά στοιχεία. Οι φωτογραφίες είναι ευγενική προσφορά του Aaron Hertzmann, Πανεπιστήμιο του Τορόντο, στον Καναδά, και τη βάση δεδομένων Corbis.

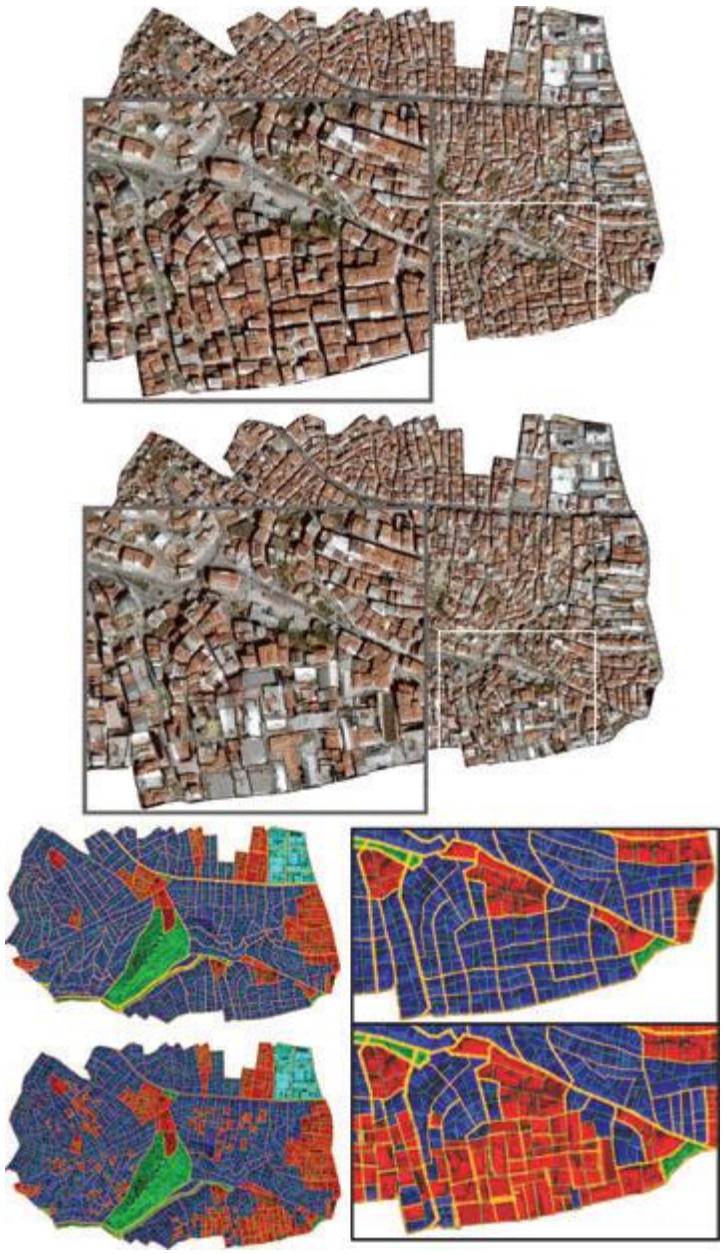
Πιο πρόσφατα, προσεγγίσεις σύνθεσης έχουν προσαρμοστεί να εκμεταλλεύονται την τυπική οργάνωση ενός αστικού χώρου. Παραδοσιακή σύνθεση υφής δεν γνωρίζει τη μοναδική γεωμετρική δομή του αστικού χώρου (π.χ. δρόμων, αγροτεμάχια, με βάση τα χνάρια). Παρά το γεγονός ότι η στρέβλωση υφής μπορεί να μετρηθεί και ίσως να ελαχιστοποιείται (π.χ. [Sander P. et. al. 2001]), η προκύπτουσα εικόνα δεν είναι συνεπής και ισχύει με την έννοια του να διαθέτει ένα εύλογο οδικό δίκτυο και αγροτεμάχια δρόμων, και τα ίχνη κτιρίου.

Οι [Aliaga et al. 2008] προτείνουν μια μέθοδο βάση παραδείγματος (examplebased) σύνθεσης των αστικών διατάξεων που γνωρίζει τη δομή του αστικού χώρου. Η μέθοδός τους χρησιμοποιεί ως είσοδο ένα σύνολο του παραδείγματος αστικών τεμαχίων διάταξης και ταυτόχρονα εκτελεί τόσο μια σύνθεση που βασίζεται στη δομή και μια σύνθεση βασισμένη σε εικόνα για να δημιουργήσει μια πλήρη αστική διάταξη με ένα εύλογο οδικό δίκτυο και με μια εικόνα εναέριας θέας (Εικόνα 9). Δεδομένα δομής και εικόνας από πραγματικού κόσμου πόλεων που χρησιμοποιούνται από τον αλγόριθμο σύνθεσης για να

παρέχουν αρκετά υψηλού επιπέδου λειτουργίες που μπορούν να χρησιμοποιηθούν για να δημιουργήσουν αλληλεπιδραστικά πολύπλοκα

σχεδιαγράμματα από το παράδειγμα. Ο χρήστης μπορεί να δημιουργήσει νέες αστικές διατάξεις από μια σειρά ενεργειών, όπως η ένταξη (join), η επέκταση (expand) και συνδυασμός (blend) χωρίς να ανησυχούν για το χαμηλό επίπεδο κατασκευαστικών λεπτομερειών.

Σε σχετική εργασία, οι ίδιοι συγγραφείς προτείνουν μια μέθοδο για τη διαδραστική αναδιάταξη των αστικών διατάξεων [Aliaga et. al. 2008] (Εικόνα 9). Ειδικότερα, η μέθοδος παίρνει ως είσοδο τα στοιχεία διάνυσμα των μπλοκ, δρόμων, και τα αγροτεμαχίων του αστικού χώρου, μαζί με μια εικόνα εναέριας θέας στον ίδιο χώρο, και θεωρεί τη συνδεσιμότητα και την οριοθέτηση των αγροτεμαχίων και των δρόμων. Υποστηρίζονται διάφορες λειτουργίες επεξεργασίας, όπως η επέκταση, η κλίμακα, η αντικατάσταση και κίνησης υποστηρίζονται. Ο πολεοδομικός σχεδιασμός διασπάται σε μια συλλογή γειτονικών πλακίδια, που χωρίζονται από δρόμο ή όρια του αγροτεμαχίου. Η συγκεκριμένη μετατροπή της διάταξης γίνεται με την κατανομή της σφαιρικής παραμόρφωσης αποτελεσμάτων που προκύπτει ανάμεσα σε όλα τα πλακίδια, επιφυλασσόμενης της συνδεσιμότητας τους και την ελαχιστοποίηση των ατομικών παραμορφώσεών τους παρόμοια με την ελαχιστοποίηση στρεβλώσεων υφής κατά τη διαδικασία χαρτογράφησης υφής για πολύπλοκα αντικείμενα.



Εικόνα 9: Διαδραστική αναδιάρθρωση των αστικών διατάξεων [Aliaga et. al. 2008]]. Δορυφορικές εικόνες από ένα πρωτότυπο και τροποποιημένος πολεοδομικός σχεδιασμός στον οποίο μια κατοικημένη ζώνη έχει μετατραπεί σε βιομηχανική ζώνη (πάνω και μέση). Η διαδικασία αποτελείται από τον επαναυπολογισμό της τοπολογίας της επηρεαζόμενης περιοχής για να φιλοξενήσει αγροτεμάχια ενός νέου τύπου ζωνών, και αντιγραφή επιλεγμένων πλακιδίων από τη βιομηχανική ζώνη (χρώμα κόκκινο) της πόλης σε προηγούμενες κατοικίες (χρώμα μπλε) περιοχής (κάτω).

Ο Chen . [Chen et al. _ 2008] χρησιμοποιεί την έννοια των πεδίων ροής και τομείς τανυστών, όπως αναπτύχθηκαν σε ορισμένα παράδειγμα με βάση τις μεθόδους υφής (π.χ. [Kwatra V. et. al 2005]), αλλά περισσότερο το χρησιμοποιεί για να μοντελοποιηθεί η ρυμοτομία της πόλης (Εικόνα 10). Βασίζονται στην παρατήρηση ότι για πολλά μοτίβα δρόμων υπάρχουν δύο κυρίαρχες κατευθύνσεις, λόγω της ανάγκης για αποτελεσματική χρήση του χώρου. Είναι ενδιαφέρον ότι, οι τομείς τανυστών δημιουργούν δύο σειρές υπερ-βελτιστοποιήσεις: το ένα ακολουθεί το

κύριο ιδιοδιάνυσμα (eigenvector) πεδίο και το άλλο το μικρότερο ιδιοδιανυσματικό πεδίο. Στη δημοσίευσή τους, οι Chen et al. εισάγουν έναν αγωγό μοντελοποίησης που αποτελείται από ένα στάδιο τανυσμού μοντελοποίησης μοντελοποίησης και ένα πεδίο παραγωγής οδικού γραφήματος. Το στάδιο τανυσμού μοντελοποίησης χρησιμοποιεί πολλές εργασίες μοντελοποίησης, συμπεριλαμβανομένης της ιεραρχικής επεξεργασίας, που βασίζονται σε στάδιο τανυσμού θορύβου (noise-based tensor field) τροποποίησης , εξομάλυνσης, μιας διεπαφής πινέλου και τον υπολογισμό των σταδίων τανυσμού από τοπογραφικούς χάρτες. Η παραγωγή οδικού γραφήματος επεκτείνει τους υφισταμένους απλουστευμένους αλγόριθμους εντοπισμού για να δημιουργήσουν ένα γράφημα. Ένα οπτικά πιθανό γράφημα δρόμου έχει κάποιους περιορισμούς για την εγγύτητα των δρόμων και για τον αριθμό των αδιεξόδων.

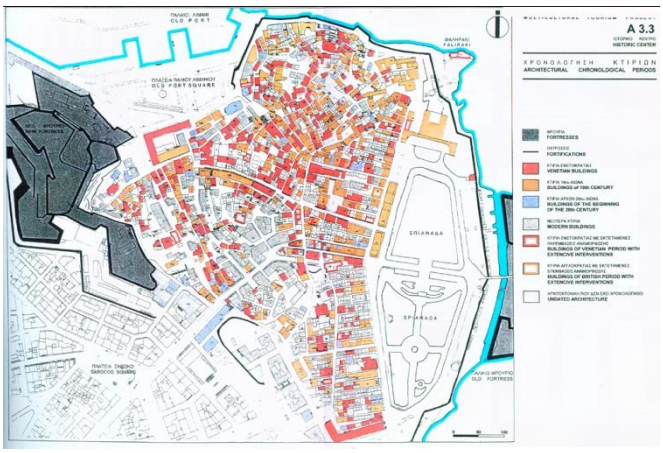
Εικόνα 10: Διαδραστική διαδικαστική μοντελοποίηση δρόμων [Chen et al. _ 2008]. Οι έννοιες των πεδίων ροής και οι τομείς τανυστών χρησιμοποιούνται για να διαμορφώσουν τη ρυμοτομία της πόλης.

Σε μια σχετική προσπάθεια σύνθεσης λεπτομερών γεωμετρικών χαρακτηριστικών του εδάφους, οι [Bruneton E., Neyret F. (*2008*)] αντιμετώπισαν το πρόβλημα σύμφωνα με ένα ανάγλυφο σχήμα σε ένα σύνολο χαρακτηριστικών των διανυσμάτων (vector) συμπεριλαμβανομένων των δρόμων, των ποταμών, των λιμνών και χωραφιών. Η προσέγγισή τους αυτή είναι να συνδυάσει ένα ψηφιακό υψομετρικό μοντέλο με στρωματοποιημένα διανυσματικά δεδομένα GIS για να

πάρουν ακριβή όρια χαρακτηριστικών και να επιβάλει τους περιορισμούς των δεδομένων των διανυσμάτων (vector) στο έδαφος. Φιλικά ως προς τη GPU δομές δεδομένων και αλγορίθμων προτείνονται να επιτρέπουν σε πραγματικό χρόνο την επεξεργασία και την απόδοση των μεγάλων εδαφών (Εικόνα 11).

Εικόνα 11: Σε πραγματικό χρόνο απόδοση και επεξεργασία των εδαφών με βάση τους φορείς (vector-based) [BRUNETON E., NEYRET F., 2008]. Οι μεγάλες εδαφών με βάση τους φορείς (vector-based) είναι γεμάτα με λεπτομερή χαρακτηριστικά, όπως δρόμοι, ποτάμια, λίμνες και τα πεδία σε πραγματικό χρόνο. Οι φωτογραφίες είναι ευγενική προσφορά του Eric Bruneton, INRIA, Γαλλία.

Παρ' όλες τις εκτενής μεθόδους μοντελοποίησης διάταξης οδικών δικτύων που πρωτοαναφέρθηκαν παραπάνω, η μέθοδος που ακολουθήθηκε σε αυτό το κομμάτι της πρακτικής είναι παρόμοια με εκείνης των [Aliaga et. al., 2008]. Επιλέχθηκαν αεροφωτογραφίες (Εικόνα 12) εκείνης της περιόδου (18^{ου} -19^{ου} αιώνα) της πόλης της Κέρκυρας και με βάση το τότε μοντέλο δρόμων και μπλοκ θα κατασκευαστούν στη συνέχεια τα κτίρια. Η μεθοδολογία που ακολουθείται διακρίνονται τα οδικά δίκτυα της περιοχής και κατασκευάζονται διαδικαστικά τα όρια των μπλοκ που σχηματίζουν οι δρόμοι. Από το κομμάτι αυτό έρχεται η σειρά απεικόνισης των κτιρίων εποχής που η τοποθέτησή τους στον αστικό χώρο θα εξαρτηθεί από τα φυσικά τους μεγέθη, όπως οι οικοδομικές τους προεκτάσεις. Τα μπλοκ που δημιουργούνται διαδικαστικά μέσω κώδικα αντιγράφουν όσο πιο πιστά γίνεται τα όρια των μπλοκ που απεικονίζονται στην αεροφωτογραφία και θα συγκολληθούν επάνω στα προσχέδια του χάρτη. Εκτενέστερη ανάλυση αυτού πραγματοποιείται στα επόμενα κεφάλαια που ακολουθούν.

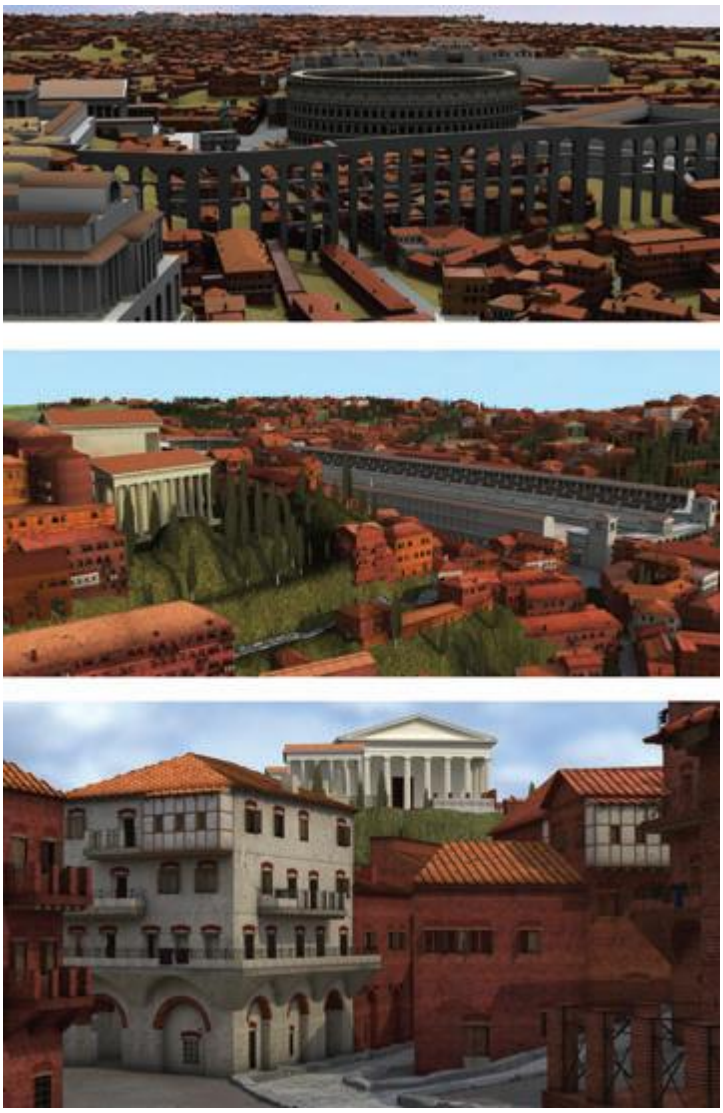


Εικόνα 12: Αεροφωτογραφία χάρτης της Κέρκυρας το 18ο και 19ο αιώνα. Πηγή: Δημόσια Βιβλιοθήκη Κέρκυρας 2012.

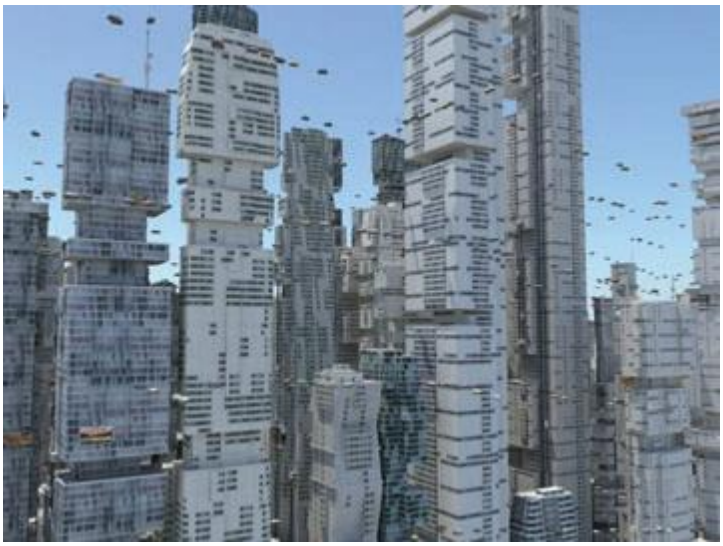
2.6.3 Μοντελοποίηση Κτιρίων

Πολλές εργασίες έχουν ασχοληθεί ειδικά με το πρόβλημα της δημιουργίας 3D μοντέλων κτιρίων. Οι περισσότερες από αυτές τις μεθόδους χρησιμοποιούν γραμματικές σχήματος για τη δημιουργία μάζες κτιρίων και προσόψεων, και προτείνουν μια ποικιλία εργαλείων για την αποτελεσματική αλληλεπίδραση του χρήστη και τη δυνατότητα ελέγχου της γεωμετρίας εξόδου.

Οι [Müller et al., 2006] έχτισαν πάνω σε κανόνες τμηματοποίησης, αλλά πρόσθεσαν διάφορα άλλα στοιχεία. Πρώτον, συμπεριέλαβαν λειτουργίες σχήματος για πολλαπλή μοντελοποίηση (ένα είδος πρόχειρου 3D σκίτσου) με την επέκταση των εντολών turtle που χρησιμοποιούνται σε L-systems [P. Prusinkiewicz & A. Lindenmayer, 1990]. Τα πολλαπλά μοντέλα συνήθως δημιουργούνται από σύνθεση πολλών στοιχειώδη σχημάτων. Η πολλαπλή μοντελοποίηση είναι αρκετά διαισθητικό και αντικατοπτρίζει την πραγματική διαδικασία σχεδιασμού που χρησιμοποιείται στην αρχιτεκτονική. Δεύτερον, εισήγαγαν περιεχόμενα ευαίσθητων κανόνων για τη συντονισμένη παραγωγή των διαφόρων μαζικών κτιρίων. Κατά συνέπεια, μια ευρεία ποικιλία κτιρίων μπορούν να παραχθούν διαδικαστικά. Στην εικόνα 13, δείχνουμε αποδόσεις της εικονικής αναπαράστασης της πόλης της αρχαίας Ρώμης. Η εικόνα 14 δείχνει μια προέκταση της Νέας Υόρκης μετά από 250 χρόνια στο μέλλον, εμπνευσμένο από την ταινία του 1997 *The Fifth Element* (το Πέμπτο Στοιχείο).



Εικόνα 13: Rome Reborn. Δύο απεικονίσεις της ανοικοδόμησης της αρχαίας Ρώμης που αποτελείται από περισσότερα από 7000 διαδικαστικά παραγόμενα οικιακά κτίρια. Τα αξιοθέατα όπως το Κολοσσαίο (επάνω) και το Circus Maximus (κάτω) έχουν διαμορφωθεί με το χέρι. Οι φωτογραφίες είναι ευγενική προσφορά του Bernard Frisher, IATH and Procedural Inc.



Εικόνα 14: New York City 2259. Διαδικαστικά παραγόμενα κτίρια από το μοντέλο της Νέας Υόρκης μετά από 250 χρόνια στο μέλλον. Η εικόνα είναι ευγενική προσφορά της Procedural Inc.

Ο Lipp και η ομάδα του [Lipp et al. 2008] εισήγαγαν την ιδέα της διαδραστικής επεξεργασίας γραμματικής Αντί της γραφής κανόνες με ένα πρόγραμμα επεξεργασίας κειμένου, το πλαίσιο της ομάδας του Οι [Lipp et al. 2008] δίνουν τη δυνατότητα να σχεδιάσει κάποιος και να επεξεργαστεί κανόνες εξ 'ολοκλήρου με μια γραφική διεπαφή χρήστη. Η επέκταση αυτή θα πρέπει να κάνει τη διαδικαστική μοντελοποίηση πιο προσιτή σε ένα ευρύτερο κοινό.

Πρόσφατα, η ιδέα της συγχώνευσης των εννοιών από τον κλάδο της επιστήμης των υπολογιστών με αντικείμενο την τεχνητή όραση και της διαδικαστικής μοντελοποίησης είχε κάποια επιτυχία. Οι [Müller et al. 2007] χρησιμοποιούν την ιδέα των κανόνων τμηματοποίησης για τη δημιουργία ενός άνω προς τα κάτω πλαίσιο για την ανάλυση πρόσοψης, χρησιμοποιώντας μια μόνο ορθογραφική εικόνα ως είσοδο. Πρώτον, εντοπίζονται οι σημαντικές μετατοπίσεις συμμετρίας και τα αντίστοιχα εικονοστοιχεία καταρρέουν το ένα πάνω στο άλλο μέχρις ότου μια μικρή αμείωτη εικόνα πρόσοψης να υπολογίζεται. Σε αυτό χρησιμοποιείται μια ολική βελτιστοποίηση για να βρεθούν τμηματοποίησης γραμμών για τον καθορισμό γραμμών και στηλών των στοιχείων πρόσοψης Περαιτέρω γίνεται υποδιαίρεση παρόμοια με τους κανόνες τμηματοποίησης που εισήγαγαν οι [Müller et al. 2007] και τελικά οι παράμετροι των κανόνων γραμματικής σχήματος μπορούν να εξαχθούν Αυτό καθιστά δυνατό να υπολογίσουμε μεταβολές του ανακατασκευασμένου κτιρίου. Η εικόνα 15 δείχνει ένα παράδειγμα μίας ανακατασκευασμένης πρόσοψης κτιρίου από ένα μόνο εναέρια φωτογραφία.

Εικόνα 15: Τεχνικής βασισμένης σε εικόνα (Image-based) διαδικαστικής μοντελοποίησης των προσόψεων [Müller et al. 2007] Μια εικόνα πρόσοψης κτιρίου που χρησιμοποιείται ως είσοδος στον αλγόριθμο (κορυφή). Ένα γράφημα πλέγματος επικαλύπτεται πάνω από την αρχική εικόνα (μεσαία) Το προκύπτον μοντέλο 3D που παρέχεται με επαναφωτισμό (relighting) και χάρτες σκιάς (shadow maps) (κάτω).

Μια σχετική προσέγγιση από Xiao . [Xiao J. et. al 2008] χρησιμοποιεί επίσης χρησιμοποιεί υποδιαίρεση παρόμοια με το πώς η αρχιτεκτονική διαμορφώνεται με μια τη μοντελοποίηση με βάση τη γραμματική. Σε αντίθεση με τη μοντελοποίηση με βάση τη γραμματική, η ομάδα του Xiao χρησιμοποιεί υποδιαίρεση σε κάθε τμήμα του μοντέλου έτσι ώστε να κάνουν επίσης χρήση των λειτουργιών συγχώνευσης. Το έργο αυτό αφορά τεχνολογία με βάση την όραση για την ανακατασκευή 3D μοντέλων πρόσοψης υψηλής οπτικής ποιότητας εικόνων από

πολλαπλά επίπεδα εδάφους και όψεις οδών (Εικόνα 16). Η μέθοδός τους χρησιμοποιεί εικόνες που έχουν ληφθεί κατά μήκος των δρόμων και στηρίζεται στη δομή από την κίνηση για να ανακτήσει αυτόματα θέσεις της κάμερας και των σημείων σύννεφου (point clouds[1]). Η πρόσοψη είναι αρχικά ως ένα ισοπεδωμένο ορθογώνιο επίπεδο με μια σχετική εικόνα που έχει συνθετηθεί από πολλαπλές φωτογραφίες και αποσυντίθεται περαιτέρω και αυξάνονται χρησιμοποιώντας 3D πληροφορίες σημείο σύννεφου. Ένα βασικό συστατικό για τη δημιουργία των υψηλής ποιότητας μοντέλων είναι τα διαδραστικά εργαλεία που επιτρέπουν στο χρήστη να βελτιώνει το μοντέλο.

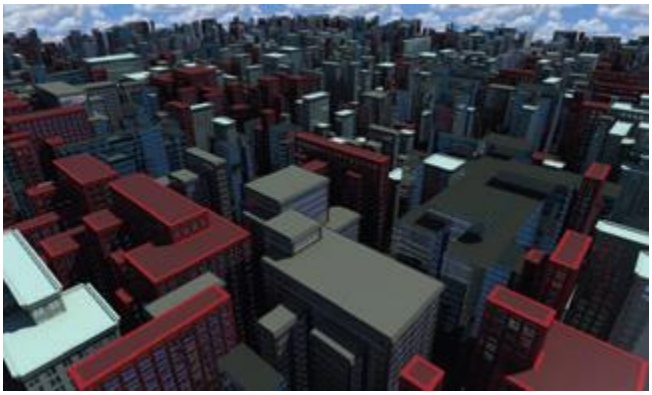
Εικόνα 16: Τεχνικής βασισμένης σε εικόνα (Image-based) πρόσοψης [XIAO J. et. al 2008]. Οι 3D προσόψεις ανακατασκευάστηκαν από εικόνες από πολλαπλά επίπεδα εδάφους και όψεις δρόμων. Οι εικόνες είναι ευγενική προσφορά του Tan Ping, Εθνικό Πανεπιστήμιο της Σιγκαπούρης, Σιγκαπούρη.

Οι [Aliaga et al. 2007] πρότειναν μια μέθοδο για την κατασκευή μιας γραμματικής από φωτογραφημένα και υποδιαιρούμενα κτίρια, που επιτρέπει την ταχεία σκιαγράφηση των νέων αρχιτεκτονικών δομών στο ύφος του πρωτοτύπου. Οι [Aliaga et al. 2007] πρότειναν μια μέθοδο για την κατασκευή μιας γραμματικής από φωτογραφημένα και υποδιαιρούμενα κτιρίων, που επιτρέπει την ταχεία σκιαγράφηση των νέων αρχιτεκτονικών δομών στο ύφος του πρωτοτύπου. Χρησιμοποιώντας δεδομένα από διάφορα μοντέλα που συνέλαβαν, νέα κτιρίων μπορούν να σχεδιαστούν πολύ γρήγορα και τους παρέχεται φωτορεαλισμός ή και καθόλου φωτορεαλισμός (π.χ. πένα και μελάνι) αλλά πάντα σε ένα ύφος συγκρίσιμα με τα πρωτότυπα . Περαιτέρω, η αφαίρεση απόφραξης και αλγόριθμοι εξισορρόπησης χρώματος καθιστούν δυνατή τη χρήση εξαιρετικά αποφραγμένων κτιρίων σε μεταβαλλόμενες συνθήκες φωτισμού (Εικόνα 17).

[1] Ένα σημείο σύννεφου (point cloud) είναι μια περιγραφή ενός συνόλου σημείων, που συνήθως ορίζεται από μια τριπλάδα [x, y, z] που ορίζεται σε ένα ορθογώνιο σύστημα συντεταγμένων αντιπροσωπεύουν ένα τρισδιάστατο χώρο. Μερικές φορές, τα σημεία αυτά αναφέρονται ως κορυφές εάν πρόκειται να χρησιμοποιηθούν ως γωνίες ενός πολυγωνικού πλέγματος, αλλά ένα πλέγμα κατασκευασμένο από ένα σημείο σύννεφου δεν περιλαμβάνει αναγκαστικά αυτά τα σημεία ως κορυφές.

Εικόνα 17: Στυλ γραμματικών για την απεικόνιση της αρχιτεκτονικής [Aliaga et al. 2007]. Ο χρήστης δημιουργεί και χωρίζει ένα αρχικό μοντέλο κτιρίου (πάνω αριστερά). Τα επαναλαμβανόμενα μοτίβα των στοιχείων του κτιρίου αυτόματα εντοπίζονται και μια αντιπροσωπευτική γραμματική κατασκευάζεται (πάνω δεξιά). Ο χρήστης μπορεί να δει στη συνέχεια το μοντέλο (κάτω αριστερά) και να το αλλάξει σε πραγματικό χρόνο παράγοντας νέα μοντέλα (κάτω δεξιά).

Όσον αφορά στην σύνθεση μοντέλου, ο Merrell [Merrell P., 2007] παρουσιάζει μια αντιπροσωπευτική μέθοδο, βασισμένη σε παράδειγμα σύνθεσης τρισδιάστατου μοντέλου. Αυτή η προσέγγιση μπορεί να χρησιμοποιηθεί για τη δημιουργία συμμετρικών μοντέλων, μοντέλα που αλλάζουν με την πάροδο του χρόνου, και τα μοντέλα που ταιριάζουν σε μαλακούς περιορισμούς. Ένας περιορισμός αυτής της πρώτης μεθόδου είναι ότι τα αντικείμενα εισόδου πρέπει να ταιριάζουν σε ένα πλέγμα ευθυγραμμισμένου άξονα . Στη δεύτερη μέθοδο [Merrell P. & Manocha D., 2008], η σύνδεση μεταξύ των χαρακτηριστικών των γειτονικών ορίων του μοντέλου εισόδου αξιοποιείται για να ξεπεραστεί ο προηγούμενος περιορισμός, και μοντέλα με αυθαίρετες κατευθύνσεις υπολογίζονται τα οποία έχουν παρόμοια συνδεδεμένα χαρακτηριστικά και μοιάζουν με τα πρότυπα του δείγματος (Εικόνα 18).



Εικόνα 18: Συνεχής Σύνθεση Μοντέλου (Continuous Model Synthesis) [MERRELL P., MANOCHA D., 2008]. Όλα αυτά τα μοντέλων κτιρίων δημιουργούνται αυτόματα από ένα ενιαίο μοντέλο παράδειγμα. Οι διαφορετικές υφές εφαρμόζονται στα κτίρια, αλλά το σχήμα του κάθε κτιρίου μοιάζει με το σχήμα της εισόδου. Οι εικόνες είναι ευγενική προσφορά του Paul Merrell, Πανεπιστήμιο της Βόρειας Καρολίνας, ΗΠΑ.

Αν και προηγουμένως εστιάσαμε περισσότερο σε υψηλού επιπέδου βασικές έννοιες μοντελοποίησης και για το σχεδιασμό της συνολικής αρχιτεκτονικής δομής, υπάρχουν στρατηγικές που ταιριάζουν καλύτερα για λεπτομέρειες μοντελοποίησης και για την μοντελοποίηση γενικότερων σχεδίων. Ένα αξιοσημείωτο παράδειγμα για τη λεπτομερής μοντελοποίηση είναι οι κυτταρικές υφές [Legakis J. et. al., 2001] που μπορούν να υπολογιστούν για να εκχωρήσουν πρότυπα τούβλων σε επιφάνειες κτιρίων. Ένα μεγάλο παράδειγμα για μια γενική και ισχυρή γλώσσα μοντελοποίησης είναι παραγωγική μοντελοποίησης πλέγματος χρησιμοποιώντας γλώσσα GML που εισήγαγε ο Havemann στη διδακτορική του διατριβή [Havemann, S., 2005]. Η GML επιτρέπει την προδιαγραφή των εντολών που μπορούν να βελτιώσουν και να ορίσουν ένα πλέγμα.

Υπήρξαν πολλές άλλες δημοσιεύσεις που προτείνουν εναλλακτικές διαδικαστικές μεθόδους μοντελοποίησης για την αρχιτεκτονική Η ομάδα του Greuter [Η ομάδα του Greuter 2003] παρουσιάζει μια μέθοδο για την παραγωγή των ψευδό άπειρων πόλεων, στις οποίες όλα τα γεωμετρικά στοιχεία της πόλης παράγονται σε πραγματικό χρόνο, όπως αυτές συναντώνται από τον χρήστη. Η ομάδα του Marvie [Marvie et al., 2005] προτείνει ορισμένες επεκτάσεις L-systems που να καταστούν πιο κατάλληλα για αρχιτεκτονικά μοντέλα Finkenzeller [Finkenzeller D., 2008] και Birch [Birch et al., 2001] εισάγουν ένα διαδραστικό διαδικαστικό πλαίσιο μοντελοποίησης Η ομάδα του Hahn [Hahn et al., 2006] παρουσιάσει μια λύση με επίκεντρο την οικοδόμηση εσωτερικών χώρων. Ενώ η ομάδα του Cabral [Cabral et al., 2009] παρουσιάζει μια προσέγγιση για τη μοντελοποίηση αρχιτεκτονικών σκηνών από την αναδιαμόρφωση και συνδυάζοντας υπάρχοντα μοντέλα υφής, όπου η εκμετάλλευση της γεωμετρίας και η υφή είναι στενά συνδεδεμένες.

Μια άλλη εντελώς διαφορετική μέθοδος υπολογιστικού σχεδιασμού είναι απαραίτητη όταν ο στόχος είναι να υπολογίσουμε ενδιαφέρον μορφές ελεύθερων επιφανειών που είναι δημοφιλείς στη σύγχρονα γυάλινα κτίρια που προσπαθούν

να εντυπωσιάσουν με γεωμετρική πολυπλοκότητα. Η ομάδα του Liu [Liu et al., 2006] χρησιμοποιούν διαδοχικό τετραγωνικό προγραμματισμό (sequential quadratic programming) για να υπολογίσει ένα τετραγωνίδιο, ένα καρέ διάταξης του πίνακα σε μια επιφάνεια. Αυτή η ιδέα επεκτάθηκε από [Pottmann et al., 2007; Pottmann et al., 2008] για να συμπεριλάβει περισσότερες γενικές διατάξεις. Η ομάδα του Liu [Liu et al., 2006] χρησιμοποιούν διαδοχικό τετραγωνικό προγραμματισμό (sequential quadratic programming) για να υπολογίσει ένα τετραγωνίδιο, ένα καρέ διάταξης του πίνακα σε μια επιφάνεια Αυτή η ιδέα επεκτάθηκε από [Pottmann et al., 2007; Pottmann et al., 2008] για να συμπεριλάβει περισσότερες γενικές σχεδιαγραμμάτων δέσμης (beam layouts) Μια άλλη ενδιαφέρουσα εργασία είναι ο υπολογισμός σχεδιαγραμμάτων δέσμης (beam layouts) από τους [Smith et. al., 2002] για κατασκευή δικτυωμάτων.

2.6.4 Οπτικοποίηση Αστικών Χώρων

Η απεικόνιση (visualization) και τα γραφικά υπολογιστών έχουν παίξει αναπόσπαστο ρόλο στην ανάπτυξη και τη χρήση προσομοιώσεων αστικού χώρου διαφόρων τύπων. Μια σειρά έργων έχουν επικεντρωθεί στην ανάπτυξη νέων τεχνικών απεικόνισης για την καλύτερη κατανόηση των αποτελεσμάτων των αστικών μοντέλων προσομοίωσης.

Αρκετές ομάδες του πληθυσμού με διαφορετικά επίπεδα εμπειρίας στο χειρισμό των αστικών στοιχείων προσομοίωσης συνήθως ενδιαφέρονται για αυτά τα αποτελέσματα, συμπεριλαμβανομένων των πολεοδόμων, φορείς χάραξης πολιτικής, το δημόσιο ή ακόμα και των μοντέλων εκτέλεσης της προσομοίωσης. Από τη μία πλευρά, οι παραδοσιακές τεχνικές απεικόνισης πληροφοριών έχουν επικεντρωθεί στο χειρισμό μεγάλων αστικών συνόλων δεδομένων προσομοίωσης και καθιστώντας την ανάλυσή τους πιο διαισθητικό για τους πολεοδόμους. Από την άλλη πλευρά, πρόσφατα ερευνητικά έργα έχουν προτείνει μια διεπιστημονική συνεργασία μεταξύ των γραφικών του υπολογιστή, απεικόνισης και αστικών μοντέλων για την παραγωγή νέων τεχνικών απεικόνισης σύνολο δεδομένων προσομοίωσης αστικού χώρου. Οι τεχνικές αυτές αποσκοπούν στη διευκόλυνση της παρουσίασης και για να αυξηθεί ο αντίκτυπος για προσομοιώσεις δεδομένων αστικού χώρου σε διάφορους τομείς του πληθυσμού [C. Vanegas et. al., 2009]..

Παραδοσιακές προσεγγίσεις απεικόνισης γενικά κάνουν χρήση τεχνικών όπως έγχρωμους (choroplethic) χάρτες που προέκυψε από εξαγωγικά αποτελέσματα της προσομοίωσης, και συνοψίζονται σε μια διζωνική γεωγραφία, σε ένα Γεωγραφικό Σύστημα Πληροφοριών (GIS, Geographical Information System) για την απόδοση τους· άλλες παραλλαγές περιλαμβάνουν animations που δημιουργούνται καθιστώντας μια σειρά τέτοιων δισδιάστατων χαρτών σε έναν βρόχο, που βλέπουν διαφορετικά χρονοτεμάχια ή ποσότητες και 3D απεικονίσεις των αποτελεσμάτων προσομοίωσης με εξώθηση πολυγωνικών μορφών για την

ένδειξη της πυκνότητας, ή με χωρική εξομάλυνση με τη μορφή του περιγράμματος ή χάρτες εδάφους με την ανύψωση που αντιπροσωπεύουν μερικό ενδιαφέρον.

Ο Batty [Batty M. , 1992] για πρώτη φορά σύστησε διάφορες προσεγγίσεις που συνδυάζουν τη μοντελοποίηση αστικών χώρων, με Γεωγραφικά Συστήματα Πληροφοριών (GIS) και γραφικά υπολογιστών. Ο ίδιος συγγραφέας περιέγραψε αργότερα την επίδραση της εικονικής πραγματικότητας και 3D απεικόνισης στο GIS και τα έχει αποδείξει αυτό σε μια ποικιλία σύνθετων παραδειγμάτων [Batty M.,Cole S., 1997]. Πιο πρόσφατα, Ο Batty παρουσίασε μια ολοκληρωμένη άποψη της αστικής δυναμικής στο πλαίσιο της θεωρίας πολυπλοκότητας [Batty M., 2007].

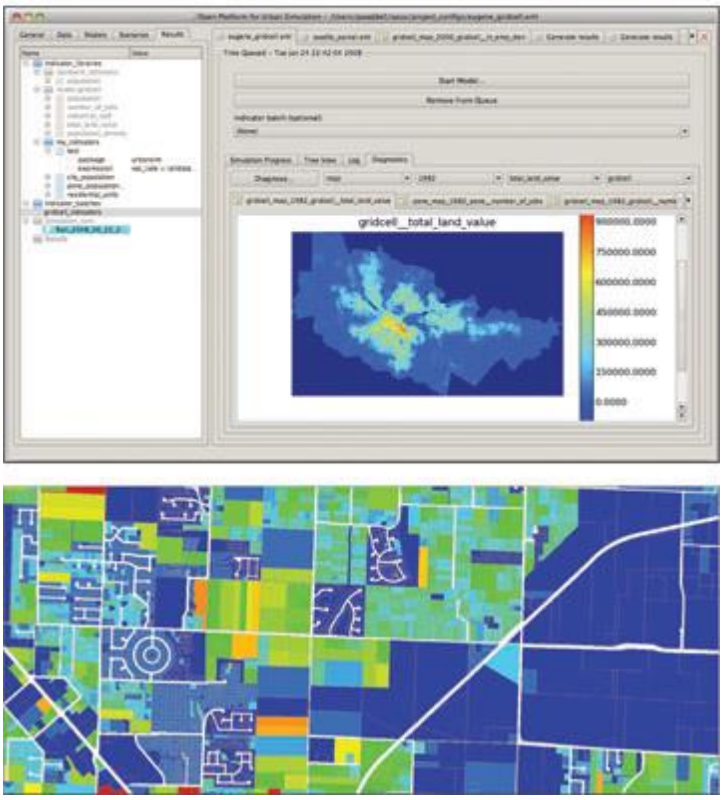
Παρά το γεγονός ότι υπήρξε ένα μεγάλο μέρος εργασίας πάνω στα Γεωγραφικά Συστήματα Πληροφοριών (GIS), πολύ λίγη έρευνα έχει γίνει αξιολογώντας τη χρησιμότητα των άλλων τύπων των απεικονίσεων για αυτόν τον τομέα. Μια μελέτη από την ομάδα του Pinnel [Pinnel et al., 2000] εξετάζει διάφορους τύπους απεικόνισης και προσπαθειών να βρεθούν οι κατάλληλες οπτικές αναπαραστάσεις για τις εργασίες μοντελοποίησης αστικών χώρων. Τα είδη των απεικονίσεων που εξετάστηκαν περιλαμβάνουν γραφήματα, γράφημα πίτας, 2D και 3D χάρτες, διαγράμματα συμβόλων και διαγράμματα φούσκας. Διασταύρωσαν κάθε ένα από αυτούς τους τύπους με τις κωδικοποιήσεις που μπορούν να χρησιμοποιηθούν αποτελεσματικά (π.χ. ένταση του χρώματος, μπάρες, περιοχή / ύψος, το μέγεθος του όγκου, το σχήμα δείκτη). Η μελέτη τους καταλήγει στο συμπέρασμα ότι, για τον αστικό σχεδιασμό και την ανάλυση, απεικονίσεις τύπου χάρτη παρέχουν τις αναγκαίες γεωγραφικές πληροφορίες, παρόλο που για ποσοτικές εργασίες τα γραφήματα και οι περιλήψεις μπορούν να παρουσιάζουν καλύτερα την απαραίτητη πληροφορία.

Μια ευρέως χρησιμοποιούμενη τεχνική απεικόνισης αστικών χώρων είναι τα χαρτογραφήματα που χρησιμοποιούν στρέβλωση χάρτη σχήματος για να απεικονίσουν τις σχέσεις και τις αξίες των αστικών και γεωχωρικών σύνολο δεδομένων (π.χ. [Keim et. al., 2004]). Η βασική ιδέα πίσω από χαρτογραφήματα είναι τα να στρεβλώσει ένα χάρτη με βάση να αλλάξει τα μεγέθη των περιοχών του, σύμφωνα με μια στατιστική παράμετρο, αλλά με έναν τρόπο που κρατά το χάρτη αναγνωρίσιμο. Οι [Chang et al., 2007] προτείνουν μια μέθοδο συνάθροισης που συνδυάζει κτίρια και συγκροτήματα της πόλης σε αναγνώσιμα συμπλέγματα (clusters). Ο στόχος τους είναι να απεικονίσουν ένα αστικό μοντέλο με ένα τρόπο που θέλει να είναι εξαρτημένο στην εστίαση και πολλαπλής ανάλυσης , παρόλο που διατηρεί την αναγνωσιμότητα της πόλης. Στην προσέγγισή τους, η προβολή του 3D μοντέλου και η προβολή δεδομένων έχουν ενσωματωθεί έτσι ώστε οι σχέσεις μεταξύ των γεωχωρικών πληροφοριών του αστικού μοντέλου και τα συναφή δεδομένα για τις αστικούς χώρους (π.χ. πληροφορίες απογραφής) μπορούν να εντοπιστούν διαισθητικά. Αν και η μελέτη χρήστη που διεξήγαγαν έδειξε ότι ορισμένα χαρακτηριστικά που εισάγονται από το σύστημά τους ενίσχυσε την ικανότητα του χρήστη να κατανοήσουν καλύτερα ένα αστικό μοντέλο, σημειώθηκε επίσης ότι η δημιουργία ευανάγνωστων πόλεων για τους χρήστες

όλων των κοινωνικών στρωμάτων δεν είναι ένα ασήμαντο έργο και θα απαιτείται η γνώση από την πλευρά των χρηστών της πόλης προτού δημιουργηθούν τα συμπλέγματα (clusters).

Οι [Roman et al., 2004] παρουσίασαν ένα διαδραστικό σύστημα για την κατασκευή πολλαπλών προοπτικών εικόνων από πλάγιες λήψεις βίντεο που λαμβάνεται από ένα κινούμενο όχημα. Η είσοδος στο σύστημά τους είναι ένα σύνολο καρέ του βίντεο. Το σύστημα τους υπολογίζει αυτόματα μια πρόσθετη εγκάρσιων εγκοπών κάμερας μεταξύ κάθε ζεύγους γειτονικών καμερών καθορισμένο από το χρήστη που οδηγεί σε μια ομαλή παρεμβολή κατά την άποψη της τελικής εικόνας πολλαπλών προοπτικών. Η πολλαπλή προοπτική εικόνας ενός ολόκληρου μπλοκ μιας πόλης μπορεί να δημιουργηθεί μέσα σε λίγα λεπτά. Ο στόχος της εργασίας αυτής είναι να παρακολουθείτε ταυτόχρονα στον πραγματικό κόσμο οι αστικές σκηνές που δεν μπορούν να αποτυπωθούν σε μία και μόνο φωτογραφία, αντί να απεικονίσει τα δεδομένα προσομοίωσης ενός αστικού χώρου. Νέες τεχνικές θα μπορούσαν να διερευνηθούν οι οποίες συνδυάζουν μία προσέγγιση πολλαπλής προοπτικής για την οπτικοποίηση δεδομένων.

Μέχρι σήμερα, τα συστήματα προσομοίωσης, όπως UrbanSim έχουν σχετικά περιορισμένο το πεδίο της απεικόνισης, παρά την παροχή εξελιγμένων και συμπεριφορών μηχανή προσομοίωσης με το μοντέλο του θέση και επιλογές μετακίνησης εκατομμυρίων παραγόντων του συστήματος. Ένα τυπικό σενάριο είναι ότι οι χειροκίνητη μετεπεξεργασία των αποτελεσμάτων της προσομοίωσης πρέπει να γίνει από έναν μοντέλο χρήστη για την εξαγωγή συνοπτικών δεικτών από τα αποτελέσματα, την εξαγωγή τους από το περιβάλλον προσομοίωση σε ένα σύστημα , τη δημιουργία σχεσιακών ενώσεων των δεικτών των υφισταμένων GIS (Γεωγραφικά Συστήματα Πληροφοριών) επιπέδων και στη συνέχεια καθιστώντας χειροκίνητα θεματικούς ή έγχρωμους χάρτες για να καταστήσουν τη χωρική διακύμανση των δεικτών που προκύπτουν (Εικόνα 19). Όπως χρησιμοποιείται στην βιβλιογραφία σχεδιασμού, ένας δείκτης είναι μια μεταβλητή που μεταφέρει πληροφορίες σχετικά με την κατάσταση ή τάσης ενός ή περισσοτέρων χαρακτηριστικών του υπό εξέταση συστήματος. Το έργο των [Schwartzman Y., Borning A., 2007] ανέπτυξε μια web-based σύστημα δεικτών για την UrbanSim και αξιολογούνται τεχνικές σχεδιασμού συμπεριλαμβανομένης της αξίας ευαίσθητου σχεδιασμού, κατασκευής πρωτοτύπων σε χαρτί και συχνές δοκιμές χρήστη.



Εικόνα 19: Παραδοσιακές Αστική Απεικόνιση. (επάνω) Ένα στιγμιότυπο οθόνης από το σύστημα δείκτη που υποστηρίζεται από το UrbanSim [SCHWARTZMAN Y., BORNING A., 2007]. Το πρόγραμμα είναι μια web-based interface για την οπτικοποίηση και την περιήγηση των αποτελεσμάτων των δεικτών, με τη μορφή πινάκων, γραφημάτων, ή χαρτών. Το σχήμα δείχνει μια έγχρωμη απεικόνισης από τιμών του οικοπέδου της πόλης του Γιουτζίν, Όρεγκον. (κάτω) Ένα πρότυπο χωροπληθών χαρτών που παραδοσιακά χρησιμοποιείται για να απεικονίσει τα αποτελέσματα μιας αστικής προσομοίωσης.

Οι απεικονίσεις τεχνικών εικονικής πραγματικότητας εξετάστηκαν επίσης και για την αστική μοντελοποίηση και πολεοδομικό σχεδιασμό. Για παράδειγμα, οι [Drettakis et al., 2007] παρουσιάζουν μία από αυτές τις τεχνικές που εφαρμόζονται σε μια μικρή κλίμακα σεναρίου πραγματικού κόσμου. Η δημοσίευση αυτή καταλήγει στο συμπέρασμα ότι τα κατάλληλα επίπεδα ρεαλισμού, όπως η χωροθέτηση 3D ήχου, η υψηλή λεπτομέρεια βλάστησης και οι σκιές και τα πλήθη, επιτρέπουν την καλύτερη εκτίμηση της συνολικής ατμόσφαιρας των εικονικών περιβαλλόντων, της αντίληψης του χώρου και των φυσικών αντικειμένων, καθώς και η αίσθηση της κλίμακας. Έτσι, χρησιμοποιώντας εικονικά περιβάλλοντα για την οπτική απεικόνιση μεγάλων αστικών χώρων προσομοιώσεων είναι μια επιθυμητή γραμμή των μελλοντικών εργασιών με σημαντικές προκλήσεις που πρέπει να ξεπεραστούν.

2.7 Πρακτικές Εφαρμογών βασισμένες σε Διαδικαστική Μοντελοποίηση

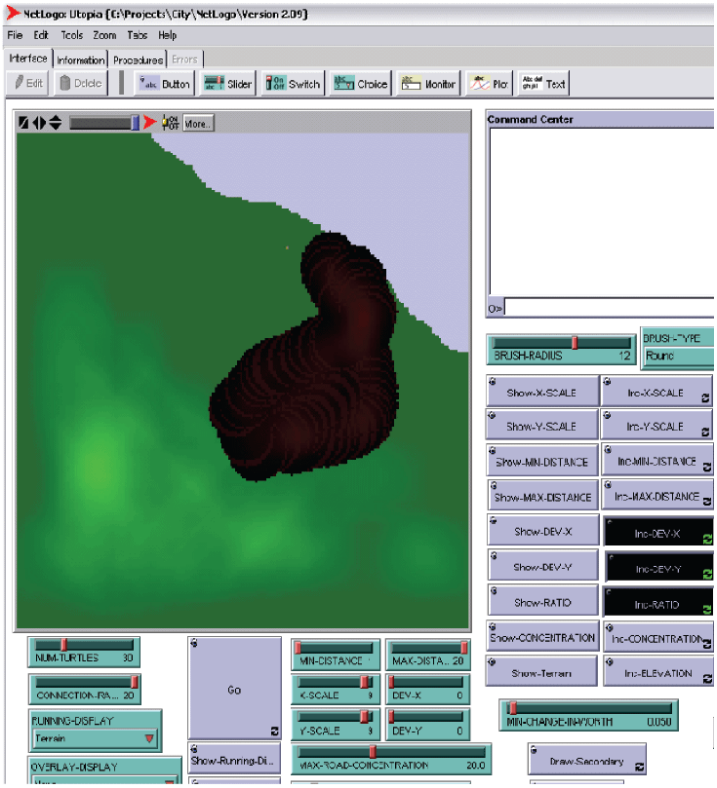
Οι πρακτικές μέσω εφαρμογών που ακολουθούν αφορούν τον σχεδιασμό και απεικόνισης μοντέλων διαδικαστικής μοντελοποίησης αστικών χώρων αλλά και απεικονίσεις βλάστησης, οδικών δικτύων και κτιρίων. Παρακάτω παρουσιάζονται αναλυτικότερα. Μέσα σε αυτές τις πρακτικές παρουσιάζεται και το πρόγραμμα εφαρμογής Unity3D απεικόνισης τρισδιάστατου χώρου και μοντέλων όπου και αποτελεί το βασικό πρόγραμμα εφαρμογής και αναπαράστασης των παλαιών κτιρίων της Κέρκυρας για τη συγκεκριμένη πτυχιακή εργασία.

2.7.1 CityBuilder

Οι [Watson et al., 2003] εφαρμόζουν μια τεχνική με βάση ένα παράγοντα για να δημιουργήσουν πόλεις ως λύση τους τους που λέγεται CityBuilder. Το σύστημα βασίζεται στην πλατφόρμα NetLogoTM που είναι ένα multi-agent προγραμματιζόμενο περιβάλλον μοντελοποίησης με βάση τη γλώσσα προγραμματισμού Logo και έχει σχεδιαστεί για να παρέχει στους χρήστες μια πλατφόρμα για να διερευνήσουν αναδυόμενα φαινόμενα. Η παραγωγή της πόλης υλοποιείται από προσομοίωση πόλεων χρησιμοποιώντας ένα σύνολο από παράγοντες που μπορούν να μοντελοποιήσουν συγκεκριμένες οντότητες της πόλης, όπως προγραμματιστές, οι αρχές σχεδιασμού και κατασκευαστές οδικών έργων. Το σύστημα του CityBuilder δε μοντελοποιεί μόνο το οδικό δίκτυο και τα κτίρια, αλλά επίσης προσομοιώνει την ανάπτυξη και εξέλιξη της πόλης στην πάροδο του χρόνου.

Οι δρόμοι δημιουργούνται από τα τμήματα του δρόμου που συναρμολογούνται σύμφωνα με ένα σχέδιο πλέγματος. Η απόκλιση από το πρότυπο επιτρέπεται και μπορεί να καθοριστεί μέσω μιας παραμέτρου. Μια τιμή απόκλισης από το μηδέν θα οδηγήσει σε ένα αυστηρά ομοιόμορφο πλέγμα (gridded) όπως το οδικό δίκτυο, μια τιμή απόκλισης κοντά στο ένα θα οδηγούσε σε μια οργανική μορφής όπως το δίκτυο. Η διασύνδεση του δικτύου μπορεί επίσης να μεταβληθεί μέσω σταθερών που υπαγορεύουν την οδική πυκνότητα και την απόσταση μεταξύ των οδικών διασταυρώσεων.

Είσοδος υπό τη μορφή ενός χάρτη ύψους εδάφους απαιτείται μαζί με ένα καθορισμένο επίπεδο νερού για να καθορίσει την επιτρεπόμενη περιοχή στην οποία οι δρόμοι και τα κτίρια μπορούν να τοποθετηθούν. Επιπλέον παράμετροι, όπως η οδική πυκνότητα, την απόσταση του πλέγματος και απόκλισης από το δίκτυο μπορεί να ρυθμιστεί χρησιμοποιώντας ολισθητήρες στο περιβάλλον όπως φαίνεται στην Εικόνα 20 για να μεταβάλει τη συμπεριφορά των παραγόντων. Επιπλέον, οι χρήστες μπορούν να προσδιορίζουν ορισμένες τιμές των παραμέτρων για συγκεκριμένες περιοχές, ζωγραφίζοντας στο χάρτη με μια βούρτσα παρόμοια με αυτή σε μια απλή εφαρμογή χρώματος.

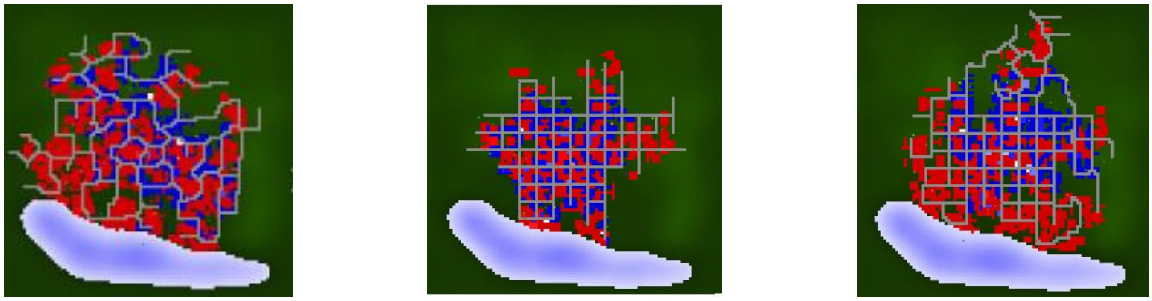


Εικόνα 20: NetLogoTM City Builder Interface

Τα τμήματα δρόμων δημιουργούνται από δύο τύπους παραγόντων - επέκτασης και συνδέσμου:

- Οι επεκτάσεις περιφέρονται γύρω από το έδαφος κοντά στις υπάρχουσες εξελίξεις για την αναζήτηση οικοπέδων οι οποίες δεν εξυπηρετούνται από το οδικό δίκτυο. Μόλις η περιοχή της γης έχει βρεθεί, εκτιμάται ανάλογα με την πυκνότητα δρόμου, κοντά σε υφιστάμενες διασταυρώσεις, και απόκλισης από το σημείο εκκίνησης. Οι δρόμοι ακολουθούν τα όρια του αγροτεμαχίου και προσπαθούν να μην κάνουν μεγάλες αλλαγές στο υψόμετρο του εδάφους.
- Οι σύνδεσμοι περιφέρουν πάνω από το υπάρχον οδικό δίκτυο δειγματοληψίας την απόσταση που διανύει ένα σημείο μέσα σε μια δεδομένη ακτίνα χρησιμοποιώντας μια έρευνα κατά πλάτος πρώτα, από το υψηλότερο επίπεδο προς το κατώτερο, κατά την οποία εξετάζονται όλοι οι κόμβοι ενός επιπέδου, του οδικού δικτύου. Εάν η απόσταση είναι πολύ μεγάλη ο σύνδεσμος θα προτείνει ένα τμήμα του δρόμου μεταξύ των δύο σημείων, ο προτεινόμενος τομέας υπόκειται στους ίδιους ελέγχους όπως και οι επεκτάσεις.

Τα οδικά δίκτυα μπορούν να προβληθούν κατά την εξέλιξή τους σε πραγματικό χρόνο, καθώς και τα παραδείγματα που δημιουργήθηκαν σε 15 έως 30 λεπτά. Η εικόνα 21γ δείχνει ένα από τα βασικά πλεονεκτήματα του συστήματος CityBuilder με την αποτελεσματική ανάμειξη μεταξύ ψηφιδωτού και προαστιακών στυλ δρόμου.



Εικόνα 21: Παράδειγμα εξόδου από διαφορετικές δομές της πόλης: α) πλεγματοποιημένη (gridded), β) Οργανικής (organic) & γ) Μικτή πλεγματοποιημένη και Οργανική (Mixed Gridded and Organic)

Η παραγωγή της χρήσης της γης για κτίρια έχει ολοκληρωθεί μέσω της αλληλεπίδρασης ενός αριθμού παραγόντων, αλλά κατά κύριο λόγο οφείλεται στο έργο των παραγόντων οικοδομικής. Οι παράγοντες οικοδομικής εκτελούν το ρόλος της αστικής ανάπτυξης και έχουν παρόμοιους στόχους: Αγορά γης, να ζητήσει άδεια σχεδιασμού, κατασκευή και πώληση. Ένα ορθογώνιο πλέγμα των κηλίδων (patches) αντιπροσωπεύουν τον κόσμο και κάθε κηλίδα (patch) μπορεί να καταλαμβάνεται από ένα κτίριο ή το δρόμο. Αυτές οι κηλίδες ομαδοποιούνται σε αγροτεμάχια υπό την ιδιοκτησία του παράγοντα κτιρίου. Ο πράκτορας κτιρίου καθορίζει την οριοθέτηση πληροφοριών κάθε αγροτεμαχίου και τα κομμάτια χαρακτηριστικών των κτιρίων.

Οι τρεις διαφορετικοί τύποι έργου ορίζονται ως: οικιακό, εμπορικό και βιομηχανικό. Όλοι οι εργολάβοι επιδιώκουν την αύξηση της αξίας της γης τους και κάθε τύπος έργου αξιολογεί την αξία της γης με διαφορετικό τρόπο και χρησιμοποιεί ένα διαφορετικό σύνολο κανόνων για να ολοκληρώσει τους στόχους του. Για παράδειγμα: εργολάβοι οικιακού τύπου αναζητούν γη κοντά στις λιγότερο πολυσύχναστες περιοχές του οδικού δικτύου σε αντίθεση με τους εμπορικούς που ψάχνουν για τα πιο πολυσύχναστα σημεία του οδικού δικτύου. Το ακίνητο αναθεωρείται και μια τοποθεσία επιλέγεται. Μια πρόταση παρασκευάζεται στη συνέχεια ότι πληροί τις ανάγκες των πελατών και πληροί και τους περιορισμούς της πόλης. Η πρόταση θα πρέπει στη συνέχεια να επανεξεταστεί από την πόλη. Η πρόταση του εργολάβου είναι επιτυχής μόνο αν περάσει τους κανονισμούς της πόλης και κάνει ένα θετικό αντίκτυπο στην κοινότητα με την παροχή μιας υπηρεσίας ή την αύξηση της αξίας της γης. Αφού ολοκληρωθεί αυτή η διαδικασία ο εργολάβος ξεκινά και πάλι να ψάχνει για περισσότερα ακίνητα. Στην εικόνα 22 φαίνονται τρεις εικόνες στιγμιότυπα της εξέλιξης μιας μικρής πόλης από τα αριστερά προς τα δεξιά.

Εικόνα 22: Ακολουθία Ανάπτυξης. Με το κίτρινο είναι το οικιακό κομμάτι, με το κόκκινο είναι το εμπορικό, και με το μπλε είναι βιομηχανικό κομμάτι. Οι δρόμοι είναι χρώματος γκρι.

Το σύστημα CityBuilder δημιουργεί ένα οδικό δίκτυο και καθορίζει τη χρήση της γης που χρησιμοποιείται στη συνέχεια για τον προσδιορισμό των τύπων κτιρίου, αλλά δεν παράγει πραγματική γεωμετρία κτιρίου και υφές. Η απεικόνιση των κτιρίων της πόλης δεν είναι ένα χαρακτηριστικό του συστήματος αυτού, αλλά λαμβάνει χώρα εξωτερικά του μοναδική μηχανή απεικόνισης παιχνιδιού SimCity.

Πεδία συζήτησης

Ανάλυση [George Kelly & Hugh McCabe, 2006]:

1. **Ρεαλισμός**: Το οδικό δίκτυο φαίνεται ρεαλιστικός και έχει την ικανότητα για την αποτελεσματική μετάβαση μεταξύ των διαφόρων μοντέλων δρόμου, ιδιαίτερα η μετάβαση από τις κεντρικές αστικές περιοχές σε λιγότερο πυκνά προαστιακές περιοχές. Δε δημιουργούνται κτίρια , αλλά ο χάρτης χρήσης γης φαίνεται ρεαλιστικός που μοιάζει με πραγματικά στατιστικά στοιχεία.
2. **Κλίμακα**: Η έξοδος που δημιουργήθηκε από το σύστημα και το παράδειγμα φαίνεται στην Εικόνα 23 είναι περιορισμένο σε κλίμακα και είναι ένα συγκρίσιμο με εκείνο της κλίμακας ενός χωριού ή μιας μικρής πόλης και όχι μία πόλης πρωτεύουσας.
3. **Διαφορετικότητα**: Οι διάφορες ζώνες που υποστηρίζονται με τις εμπορικές ζώνες με άκαμπτο μπλοκ όπως δομές δρόμων και οικιστικών περιοχών με εκτεταμένο οδικό δίκτυο. Τρεις διαφορετικές χρήσεις γης και τους τύπους κτιρίων που ορίζονται σε εμπορική, οικιακή και βιομηχανική. Είναι αδύνατο να κρίνουμε τη διαφορά που επιτυγχάνεται με αυτές τις κατηγορίες, όπως η απεικόνιση γίνεται από τον μηχανή SimCity η οποία είναι έξω από το σύστημα.
4. **Είσοδος**: Η είσοδος του χάρτη ύψους εδάφους και της στάθμης του νερού που απαιτούνται για τον προσδιορισμό των επιτρεπόμενων περιοχών στις οποίες δημιουργούνται κτίρια μπορούν να τοποθετηθούν. Άλλες εισόδοι μπορούν να καθορίζονται από το χρήστη μέσω της διαδραστικής εφαρμογής.
5. **Αποδοτικότητα**: Το CityBuilder μοντελοποιεί όχι μόνο τη δομή μιας πόλης, αλλά και την εξέλιξή της και, ως αποτέλεσμα της προστιθέμενης πολυπλοκότητας ο αλγόριθμος είναι υπολογιστικά εντατικός και

χρονοβόρος. Μια πόλη μόνο περιορισμένης κλίμακας παρόμοιας με ένα χωριό μπορεί να να παραχθεί σε μια χρονική περίοδο περίπου 15 λεπτών μη συμπεριλαμβανομένης της δημιουργίας οποιασδήποτε γεωμετρίας κτιρίου ή υφών.

6. **Έλεγχος**: Ένα καινοτόμο χαρακτηριστικό είναι διαθέσιμο σε μορφή εργαλείου χρωμάτων που μπορούν να χρησιμοποιηθούν για την βαφή τιμών των παραμέτρων στο χάρτη. Αριθμητικές παραμέτρους όπως οδικής συμπύκνωσης , η απόκλισης και κλίμακας μπορούν να καθοριστούν μέσω μιας διαδραστικής εφαρμογής χρησιμοποιώντας τα διάφορα ρυθμιστικά και γραφικά στοιχεία του γραφικού περιβάλλοντος χρήστη.
7. **Σε πραγματικό χρόνο**: Δεν υπάρχουν πραγματικού χρόνου εκτιμήσεις ή ακόμα και ένα τρισδιάστατο μοντέλο της πόλης. Η απεικόνιση παρέχεται μέσω ενός εξωτερικού συστήματος, η μηχανή SimCity, που χρησιμοποιεί ένα επίπεδο πλακιδίου εικόνας bitmap προβολής παιχνιδιού.

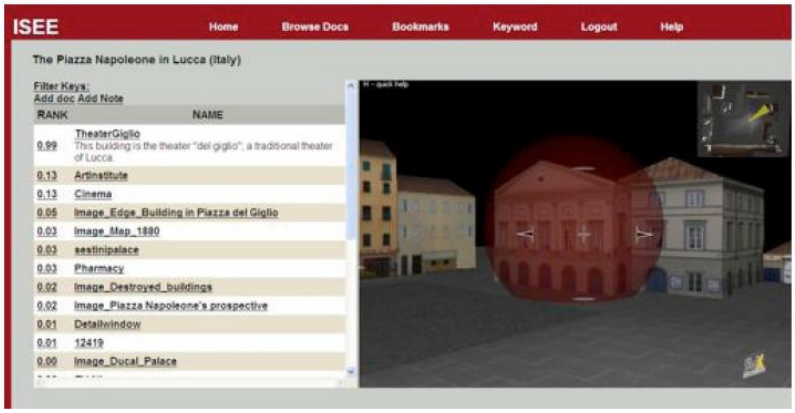
Το σύστημα αυτό θα μπορούσε να επεκταθεί εύκολα, αλλά με έναν αλγόριθμο υψηλής υπολογιστικής πολυπλοκότητας, επίσης δεν είναι κατάλληλη για διαδικαστική παραγωγή και θα μπορούσε να είναι πιο κατάλληλη για εφαρμογές προσομοίωσης.

2.7.2 XVR Framework

Το πλαίσιο XVR (XVR framework) [Carrozzino et. Al., 2005], που αναπτύχθηκε από κοινού από την PERCRO και VRMedia s.r.l., είναι ένα ολοκληρωμένο περιβάλλον που διατίθενται για την ανάπτυξη των εφαρμογών εικονικής πραγματικότητας, βασισμένο σε μια γλώσσα προγραμματισμού προσανατολισμένη σε εικονική πραγματικότητα (VR oriented scripting language) ειδικά για 3D γραφικά, ήχο 3D και, σε γενικές γραμμές, πολλά άλλα τυπικά συστατικά εικονικής πραγματικότητας. Το πλαίσιο XVR στην πραγματικότητα χωρίζεται σε δύο κύριες ενότητες: η ενότητα ελέγχου ActiveX, το οποίο φιλοξενεί τη διεπαφή για web browsers, και την XVR Virtual Machine (VM) ενότητα, η οποία περιέχει τον πυρήνα της τεχνολογίας, όπως τα 3D γραφικά, ήχο και φυσικές μηχανές, multimedia μηχανές και όλα τα στοιχεία λογισμικού διαχείρισης των άλλων ενσωματωμένων δυνατοτήτων του πλαισίου XVR. Είναι επίσης δυνατό να φορτώσει επιπλέον μονάδες που προσφέρουν προηγμένες λειτουργίες και που δεν είναι άμεσα διαθέσιμες. Η XVR scripting γλώσσα επιτρέπει τον καθορισμό της συμπεριφοράς της εφαρμογής, παρέχοντας τις βασικές λειτουργίες της γλώσσας και σχετικές μεθόδους εικονικής πραγματικότητας, διαθέσιμες όπως συναρτήσεις ή κλάσεις. Το σενάριο στη συνέχεια καταρτίζεται σε ένα bytecode που επεξεργάζεται και εκτελείται από την XVR-VM. Η ολοκληρωμένη 3D μηχανή, χτισμένη στην κορυφή του OpenGL, επιτρέπει τη διαχείριση της οπτικής εξόδου, όχι μόνο σε ένα πρότυπο παράθυρο γραφικού περιβάλλοντος

(είτε web ή τοπικά), αλλά και σε πιο προηγμένες συσκευές όπως Στερεοφωνικά Συστήματα Προβολής (Strereo Projection Systems) και Οθόνες προσαρμοσμένες στο κεφάλι (HMDs - Head mounted displays). Η μηχανή χρησιμοποιεί state-of-the-art αλγόριθμους επιλογής, απλούστευσης, κανονικής χαρτογράφησης και εικόνας προσωρινής αποθήκευσης για την επίτευξη καλών αναπαραστάσεων πραγματικού χρόνου , ακόμη και με υψηλής πολυπλοκότητας μοντέλα.

Πολλές εφαρμογές που έχουν δημιουργηθεί με το πλαίσιο XVR κάνουν χρήση 3D μοντέλων αστικών περιοχών. Αυτές μπορεί να περιλαμβάνουν προσομοιωτές οχημάτων, προσομοιωτές συμπεριφορά του πλήθους και, τον τελευταίο καιρό, και μια καινοτόμο μεθοδολογία (ISEE) που ασχολούνται με πρόσβαση σε πληροφορίες που σχετίζονται με την Πολιτιστική Κληρονομιά [Pecchioli et. al., 2008] (Εικόνα 23).



Εικόνα 23: Εφαρμογή ISEE.

Στην εργασία αυτή, τα διαδραστικά 3D μοντέλων αναπαράγουν τα κύρια χαρακτηριστικά των αντίστοιχων πραγματικών περιβαλλόντων που χρησιμοποιούνται για τη χαρτογράφηση χωρικών ζωνών σε «κομμάτια» πληροφοριών. Το καινοτόμο στοιχείο έγκειται στη χρήση της μεθόδου Gauss σε αναπαράσταση 3D (3D Gaussians) τόσο για την χαρτογράφηση των πληροφοριών που σχετίζονται με τις ζώνες όσο και την τρέχουσα προβολή του χρήστη· Αυτό αποδίδει στο να αποκτήσουν αυτόματα ένα μέτρο της «χωρικής συνάφειας» της πληροφορίας, όπως ορίζεται με βάση τη θέση της στον κόσμο όσο και σχετικά με τη θέση / προσανατολισμό του χρήστη στον 3D χώρο. Αυτή η εφαρμογή αντιπροσωπεύει μία τυπική περίπτωση όπου η οπτική πιστότητα δεν είναι το κύριο ζήτημα: μάλλον, η δυνατότητα ταχείας παραγωγής ακόμη που αδρών ορισμένων 3D μοντέλων, με επαρκή γεωγραφική αντιστοιχία με τις πραγματικές θέσεις, αποτελεί βασικό στοιχείο προκειμένου να πρωτοτυπήσουν γρήγορα τις βάσεις δεδομένων για να έχουν πρόσβαση με τη μεθοδολογία αυτή.

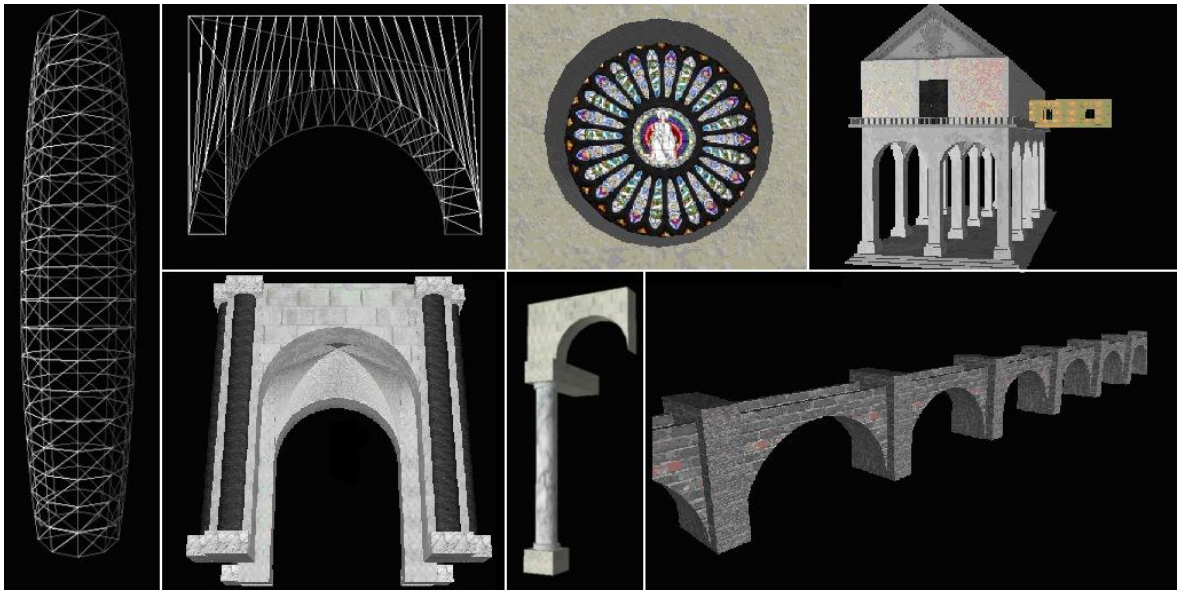
2.7.3 City Modeling Procedural Engine (CMPE)

Η Μηχανή Διαδικαστικής Μοντελοποίησης Πόλης (CMPE, City Modelling Procedural Engine) είναι ένα εργαλείο για την ημι-αυτόματη δημιουργία των 3D αστικών περιβαλλόντων, με σκοπό να στηρίξει όσες εισόδους είναι διαθέσιμες, όπως χάρτες vector, raster χάρτες, DTMs, αεροφωτογραφίες, περιγραφές κειμένου, προσπαθώντας να παρέχει την πιο συνεκτική έξοδο με την παρεχόμενη είσοδο. Παρά το γεγονός ότι ολόκληρη η ροή δεδομένων μπορεί αυτομάτως να προχωρήσει από την αρχή μέχρι το τέλος, σε κάθε στάδιο, η χειροκίνητη παρέμβαση του χρήστη είναι δυνατή να διορθώσει τα λάθη που προκαλούνται από την αυτόματη διαδικασία, για τη βελτιστοποίηση των αποτελεσμάτων, ή για την εισαγωγή περισσότερων λεπτομερειών [Carrozzino M. et. al., 2009].

Ο κύριος στόχος του CMPE είναι να επιτρέψει την δημιουργία τεράστιων 3D αστικών συνόλων δεδομένων, κατάλληλα για απόδοση σε πραγματικό χρόνο, τα οποία μπορούν να:

- ληφθούν γρήγορα με περιορισμένες παρεμβάσεις από το χρήστη, έτσι ώστε να επιταχυνθεί η διαδικασία δημιουργίας (είτε να οριστικοποιηθεί στο πρωτότυπο ή στο τελικό μοντέλο).
- αποθηκεύονται σε ένα περιορισμένο ποσό της μνήμης, έτσι ώστε να επιτρέπει τη μετάδοση μέσω του δικτύου σχετικών δεδομένων ακόμα και σε συνθήκες χαμηλού εύρους ζώνης.

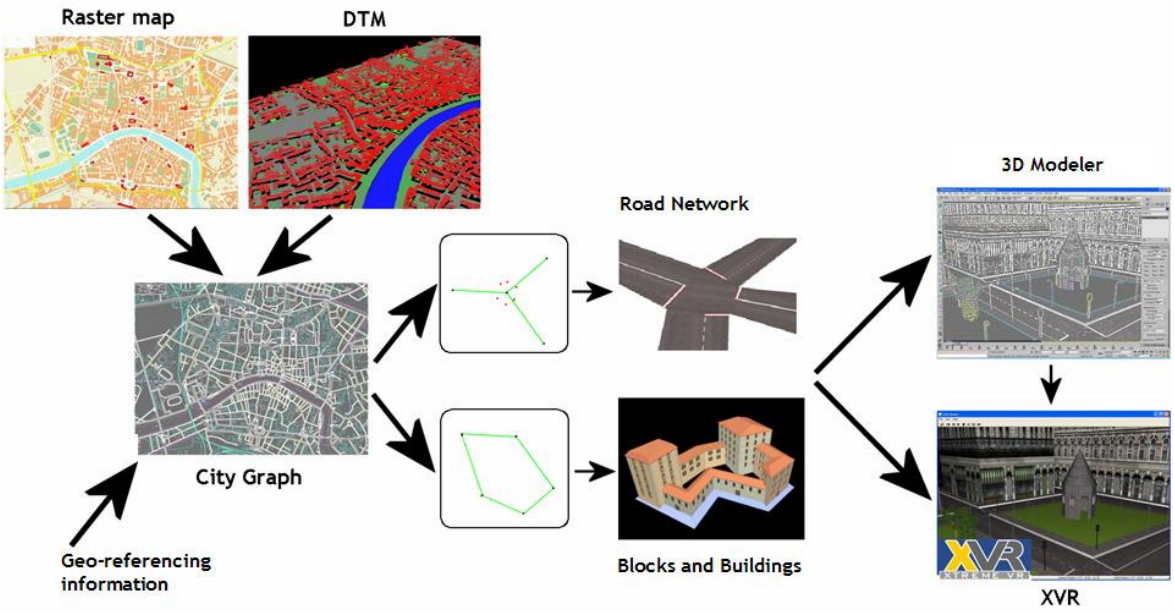
Μια πρόσθετη βιβλιοθήκη λογισμικού (PVRlib) υλοποιεί τις λειτουργίες που σχετίζονται με τη διαδικαστική παραγωγή απλών αρχιτεκτονικών οντοτήτων (ξεκινώντας από τα βασικά γεωμετρικά σχήματα, όπως κουτιά, κυλίνδρους και σφαίρες, σε πιο πολύπλοκα σχήματα, όπως καμάρες, κίονες, κιονόκρανα), τα οποία μπορούν να χρησιμοποιηθούν για να προστεθούν εύκολα μικρές βελτιώσεις στο τελικό μοντέλο ή, να ενσωματωθούν σε μια XVR γλώσσα προγραμματισμού, για να μοντελοποιήσουμε πιο περίπλοκες αρχιτεκτονικές οντότητες (όπως ναοί εκκλησίες, μνημεία) δεν είναι άμεσα χαρακτηρίσιμες μέσω απλών παραμέτρων (Εικόνα 24).



Εικόνα 24: Παράδειγμα της βιβλιοθήκης αντικειμένων PVRlib.

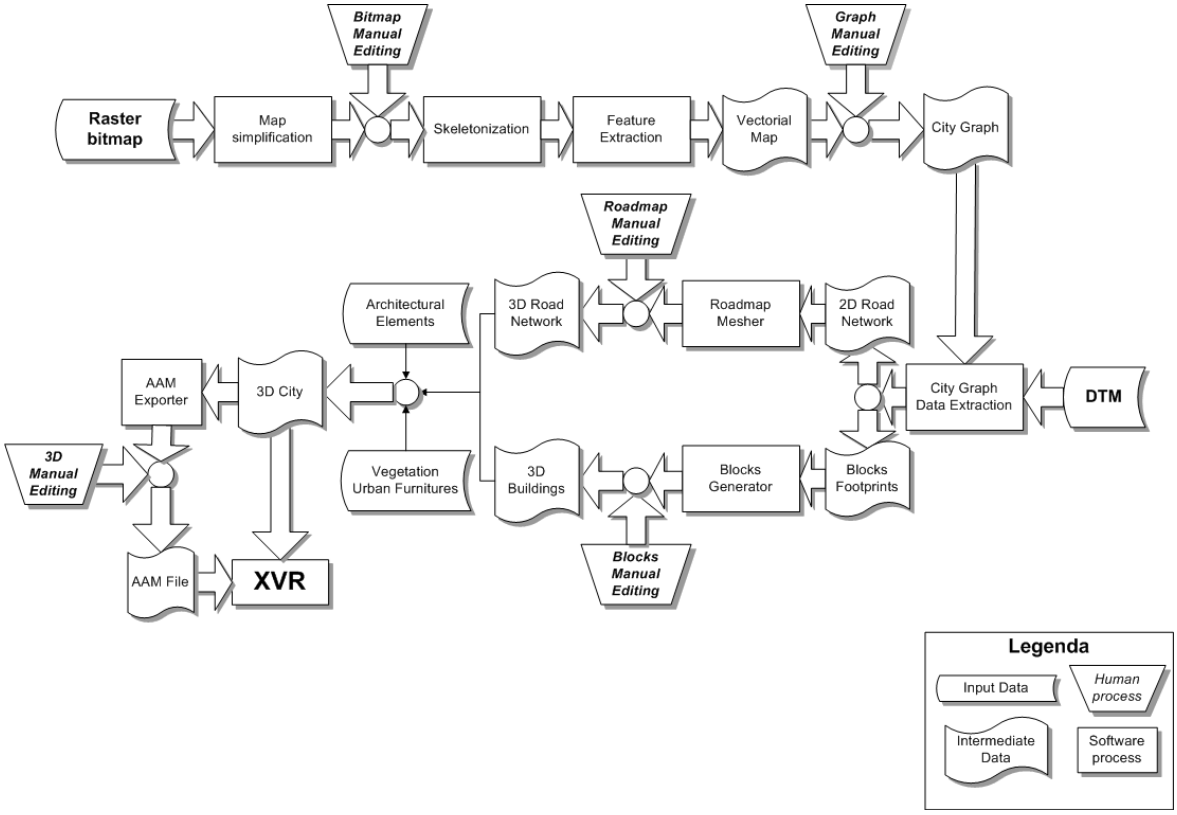
Ως μια γενική επισκόπηση, της ροής εργασιών CMPE (Εικόνα 25) μπορεί να σκιαγραφηθεί ως εξής [Carrozzino M. et. al., 2009]:

- αυτόματη εξαγωγή από raster χάρτες των σχετικών χαρακτηριστικών που σχετίζονται με το οδικό δίκτυο ή με τα ίχνη των μπλοκ (το στάδιο αυτό δεν μπορεί να γίνει εάν τα δεδομένα είναι ήδη σε διανυσματική μορφή).
- Η παραγωγή των δομών δεδομένων για τη γενική διαχείριση.
- Η παραγωγή του δισδιάστατου οδικού δικτύου (περιλαμβάνει, αν είναι δυνατόν, τους σιδηροδρόμους και τα ποτάμια).
- Η παραγωγή του τρισδιάστατου οδικού δικτύου.
- Προσδιορισμός των παραμέτρων για τα επόμενα στάδια (τετράγωνα και κτίρια).
- Παραγωγή των 3D μπλοκ και των 3D κτιρίων.
- Αποθήκευση του συνολικού 3D μοντέλου σε AAM μορφή (AAM format), είτε για να τροφοδοτήσει άμεσα τη μηχανή φωτορεαλισμού XVR σε πραγματικό χρόνο, ή να τελειοποιηθεί χειροκίνητα εντός 3D προγραμμάτων δημιουργίας μοντέλων όπως το 3D Studio Max.
- εγχειρίδιο μοντελοποίησης των στοιχείων (είτε με το λογισμικό μοντελοποίησης 3D ή απευθείας μέσω της μηχανής XVR, μέσω της βιβλιοθήκης PVRLib) και την εξαγωγή τους από το πρόγραμμα ως αρχεία AAM.
- Φωτοαπόδοση XVR.



Εικόνα 25: Ένα υψηλού επιπέδου σύστημα ροής εργασιών CMPE.

Ένα λεπτομερές σχηματικό διάγραμμα CMPE παρουσιάζεται παρακάτω στην Εικόνα 26.



Εικόνα 26: Σχηματικό διάγραμμα CMPE.

Συμπερασματικά, το CMPE είναι ένα εργαλείο για την ταχεία αυτόματη παραγωγή πολύ σύνθετων εικονικών αστικών περιβαλλόντων, κατάλληλα για απόδοσης πραγματικού χρόνου και απαιτώντας ελάχιστες παρεμβάσεις από το χρήστη.

2.7.4 CityGML

Το CityGML είναι ένα κοινό μοντέλο πληροφοριών για την αναπαράσταση τρισδιάστατων αστικών αντικειμένων. Προσδιορίζει τις κλάσεις και τις σχέσεις για τα πιο σχετικά τοπογραφικά αντικείμενα σε πόλεις και περιφερειακά πρότυπα όσον αφορά τις γεωμετρικές, τοπολογικές, σημασιολογικές ιδιότητες και των ιδιοτήτων εμφάνισης. Το πρόθεμα "City" έχει ευρεία έννοια και περιλαμβάνει όχι μόνο των κτισμάτων, αλλά και ανύψωσης εδάφους, βλάστησης, υδατικών συστημάτων, "αντικείμενα πεζοδρομίου» και πολλά άλλα. Περιλαμβάνονται οι ιεραρχίες γενίκευσης μεταξύ των θεματικών κατηγοριών, συναθροίσεις, τις σχέσεις μεταξύ των αντικειμένων και των χωρικών ιδιοτήτων [Thomas H. Kolbe et. al. 2005].

Αυτές οι θεματικές τύποι πληροφοριών υπερβαίνουν μορφές γραφικών ανταλλαγής και επιτρέπουν στους χρήστες να χρησιμοποιούν εικονικά μοντέλα 3D πόλεων για εξεζητημένες εργασίες ανάλυσης σε διαφορετικούς τομείς εφαρμογών όπως η προσομοίωση, η αστική εξόρυξη δεδομένων, τις εγκαταστάσεις διαχείρισης, υποστήριξης αποφάσεων και θεματικές έρευνες [Thomas H. Kolbe et. al. 2005].

CityGML είναι ένα ανοικτό μοντέλο δεδομένων και XML-based μορφής για την αποθήκευση και την ανταλλαγή εικονικών 3D μοντέλων πόλης. Υλοποιείται ως ένα σχήμα εφαρμογής της Γλώσσας Σήμανσης Γεωγραφίας τύπου 3 (GML3, Geography Markup Language 3), το επεκτάσιμο διεθνές πρότυπο για την ανταλλαγή χωρικών δεδομένων που αναπτύχθηκε στο πλαίσιο του Open Geospatial Consortium (OGC) και ISO TC211 [Thomas H. Kolbe et. al. 2005].

Το GML3, χρησιμοποιείται με άλλα πρότυπα OGC - κυρίως προδιαγραφών OpenGIS Web Feature Service (WFS) - παρέχει ένα πλαίσιο για την ανταλλαγή των απλών και πολύπλοκων μοντέλων 3D. Ωστόσο, WFS και GML3 καθορίζουν μόνο τη συντακτική διαλειτουργικότητα. Ένα έγγραφο GML3 πρέπει να διαρθρώνεται με τον ορισμό ενός σχήματος εφαρμογής που είναι προσαρμοσμένο σε ένα συγκεκριμένο πεδίο εφαρμογής. Στην περίπτωση αυτή, το πεδίο εφαρμογής είναι η τρισδιάστατη μοντελοποίηση μιας πόλης και το GML3 σχήμα εφαρμογής (ή προφίλ) είναι το CityGML [Thomas H. Kolbe et. al. 2005].

Το CityGML εκμεταλλεύεται άλλα ανοικτά πρότυπα και η ανάπτυξη του προχώρησε σε προσεκτική συνεργασία με άλλες ομάδες. Για παράδειγμα, η γραφική απόδοση των δεδομένων κωδικοποιημένα σε CityGML μπορεί να

επιτευχθεί με τη χρήση τυποποιημένων μορφών δεδομένων γραφικά υπολογιστών, όπως VRML, GeoVRML, X3D ή Universal 3D (U3D). Για να διασφαλιστεί η ευρωπαϊκή αποδοχή, οι προγραμματιστές CityGML έχουν συντονιστεί με το EuroSDR, μια χωρική οργάνωση ερευνητικών δεδομένων που αποτελούνται από εκπροσώπους των φορέων παραγωγής γεωγραφικών πληροφοριών και ερευνητικά κέντρα από 18 κράτη μέλη της Ευρώπης, μαζί με τους συμμετέχοντες από τη βιομηχανία και τον εμπορικό τομέα [Thomas H. Kolbe et. al. 2005].

Εντός του OGC, η συζήτηση CityGML πραγματοποιείται στο πλαίσιο της διαλειτουργικότητας OGC CAD/GIS/3D Working Group. Το CityGML θα υποβληθεί στη OGC Τεχνική Επιτροπή, που συναντήθηκε στο Εδιμβούργο, στη Σκωτία κατά τη διάρκεια της εβδομάδας του Ιουνίου 26-29, ως υποψήφιος OpenGIS προδιαγραφών για την έγκριση επανεξέτασης και πιθανής υιοθέτησής του από τα μέλη της επιτροπής. Η πρόθεση είναι να γίνει ένα ανοιχτό πρότυπο και, συνεπώς, να χρησιμοποιηθεί δωρεάν [Thomas H. Kolbe et. al. 2005].

Η ανάπτυξη του CityGML ξεκίνησε το 2002 από τα μέλη της Ομάδα Ειδικού Ενδιαφέροντος 3D (SIG 3D) της πρωτοβουλίας υποδομής χωρικών δεδομένων Βόρειας Ρηνανίας-Βεστφαλίας (NRW GDI) στη Γερμανία. Ο GDI NRW έχει δραστηριοποιηθεί στην OGC από το 1999. Ο SIG 3D είναι μια ανοιχτή ομάδα που αποτελείται από περισσότερες από 70 εταιρείες, δήμους, και ερευνητικά ιδρύματα που εργάζονται για την ανάπτυξη και την εμπορική εκμετάλλευση διαλειτουργικών μοντέλα 3D και απεικόνισης. Ένα άλλο προϊόν δραστηριοτήτων SIG 3D είναι μια προτεινόμενη 3D Web Service (W3DS) πρότυπο, μια υπηρεσία 3D απεικόνισης που έχει σήμερα την ιδιότητα ενός εγγράφου προβληματισμού στο OGC (OGC Doc. No 05-002) [Thomas H. Kolbe et. al. 2005].

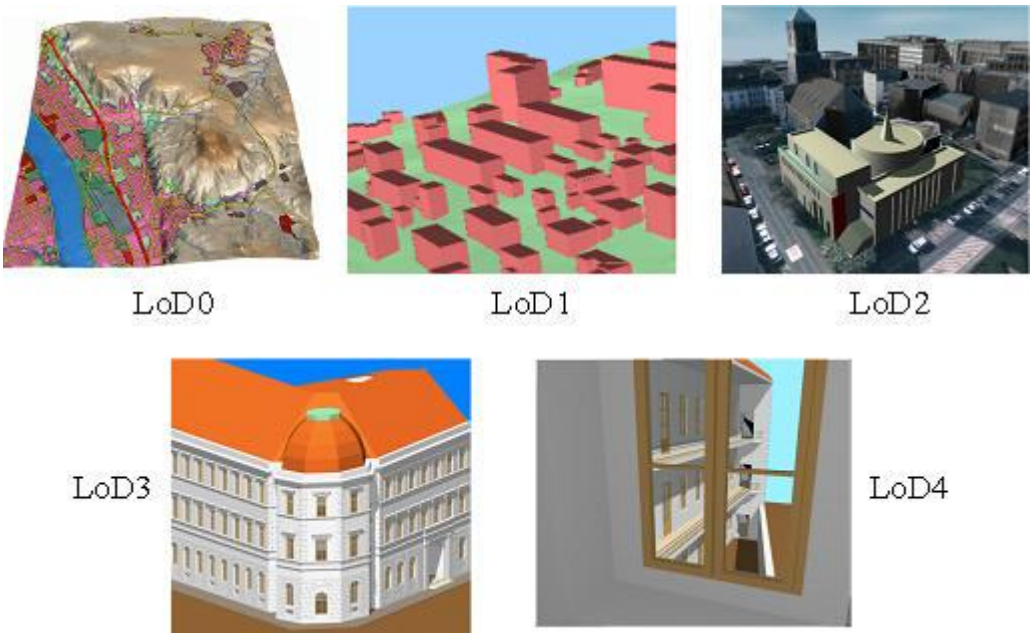
Μερικοί από τους φορείς που εργάζονται και υποστηρίζουν το CityGML είναι οι δήμοι του Βερολίνου, Αμβούργο, Κολωνία, το Ντίσελντορφ, Recklinghausen και Λεβερκούζεν, η Επίσημη Χαρτογράφηση της Μεγάλης Βρετανίας, ο Οργανισμός Χαρτογράφηση του κράτους της Βόρειας Ρηνανίας-Βεστφαλίας, εταιρείες συμπεριλαμβανομένης της T-Mobile, Rheinmetall Defence Electronics, Snowflake, CPA Geo-Information, GIStec και 3D Geo και ερευνητικά ιδρύματα όπως τα πανεπιστήμια της Βόννης, Πότσνταμ, Ντόρτμουντ, Εφαρμοσμένων Επιστημών της Στουτγάρδης, η Helmholtz ερευνητικό Κέντρο Καρλσρούης, και το Ινστιτούτο Fraunhofer για Έρευνες Γραφικών στο Darmstadt [Thomas H. Kolbe et. al. 2005].

Το CityGML αντιπροσωπεύει τη γραφική εμφάνιση των μοντέλων πόλεων, αλλά και τις σημασιολογικές ιδιότητες ή θεματικές ιδιότητες, ταξινομήσεις και συναθροίσεις ψηφιακών μοντέλων εδάφους, περιοχές (συμπεριλαμβανομένων των κτιρίων, γεφυρών και σηράγγων), βλάστησης, υδατικών συστημάτων, εγκαταστάσεις μεταφορών, κλπ. Το βασικό μοντέλο διαφοροποιεί πέντε συναπτά επίπεδα λεπτομέρειας (LOD), όπου τα αντικείμενα γίνονται πιο λεπτομερείς με την αύξηση LoD, τόσο στη γεωμετρία και στη θεματική διαφοροποίηση. Τα CityGML

αρχεία μπορούν - αλλά δεν χρειάζεται να - περιέχουν πολλαπλές αναπαραστάσεις για κάθε αντικείμενο σε διαφορετικά LoD ταυτόχρονα [Thomas H. Kolbe et. al. 2005].

Το CityGML υποστηρίζει διαφορετικά επίπεδα λεπτομέρειας (LOD), τα οποία ενδέχεται να προκύψουν από ανεξάρτητες διαδικασίες συλλογής δεδομένων και χρησιμοποιούνται για την αποτελεσματική και αποδοτική οπτικοποίηση ανάλυσης των δεδομένων. Σε ένα CityGML σύνολο δεδομένων, το ίδιο αντικείμενο μπορεί να εκπροσωπείται σε διαφορετικές LoD ταυτόχρονα, επιτρέπει την ανάλυση και την οπτικοποίηση του ίδιου αντικειμένου σε σχέση με διαφορετικούς βαθμούς ανάλυσης. Επιπλέον, τα δύο σύνολα δεδομένων CityGML που περιέχουν το ίδιο αντικείμενο σε διαφορετικές LoD μπορούν να συνδυαστούν και να ενσωματωθούν [Thomas H. Kolbe et. al. 2005].

Το CityGML παρέχει πέντε διαφορετικά επίπεδα LoD [Thomas H. Kolbe et. al. 2005], τα οποία απεικονίζονται στην Εικόνα 27. Ο αδρότερο LoD0 επίπεδο είναι ουσιαστικά δύο και μισή διαστάσεων Ψηφιακό Μοντέλο Εδάφους, με το οποίο μοντέλο μπορεί να είναι ντυμένο με μια αεροφωτογραφία ή ένας χάρτης Το επίπεδο LoD1 είναι το γνωστό μοντέλο μπλοκ, χωρίς οποιεσδήποτε δομές στέγης ή υφών. Σε αντίθεση, ένα κτίριο στο επίπεδο LoD2 έχει διαφοροποιήσει τις δομές οροφής και υφών. Αντικείμενα βλάστησης μπορούν επίσης να παρουσιαστούν. Το επίπεδο LoD3 δηλώνει αρχιτεκτονικά μοντέλα με λεπτομερείς δομές τοίχου και οροφής , βεράντες, όρμους και προβολές. Υψηλής ανάλυσης υφές μπορούν να χαρτογραφηθούν σε αυτές τις δομές. Επιπροσθέτως, λεπτομερής βλάστηση και αντικείμενα μεταφοράς είναι συστατικά ενός LoD3 μοντέλου. Το επίπεδο LoD4 ολοκληρώνει ένα μοντέλο επιπέδου LoD3 με την προσθήκη εσωτερικών δομών, όπως τα δωμάτια, εσωτερικές πόρτες, σκάλες και έπιπλα.



Εικόνα 27: Τα πέντε επίπεδα-λεπτομέρειας (LOD) που ορίζονται από το CityGML.

Τα διαφορετικά επίπεδα LoD χαρακτηρίζονται επίσης από ακρίβειες και ελάχιστες διαστάσεις των αντικειμένων [Thomas H. Kolbe et. al. 2005]. Στο επίπεδο LoD1, το ύψος θέσεως και ακρίβειας των σημείων μπορεί να είναι 5m ή λιγότερο, ενώ όλα τα αντικείμενα με ένα ίχνος τουλάχιστον 6m ανά 6m πρέπει να εξεταστούν. Η ακρίβεια της θέσης του επιπέδου LoD2 είναι 2m, ενώ η ακρίβεια ύψους είναι 1m. Σε αυτό το επίπεδο LoD, όλα τα αντικείμενα έχουν ίχνος τουλάχιστον 4m ανά 4m πρέπει να εξεταστούν. Και οι δύο τύποι ακρίβειας στο επίπεδο LoD3 είναι 0,5 m, και το ελάχιστο ίχνος είναι 2m ανά 2m. Τέλος, το ύψος θέσεως και ακρίβειας του επιπέδου LoD4 πρέπει να είναι 0,2 m ή λιγότερο. Με τη βοήθεια αυτών των στοιχείων, η ταξινόμηση σε πέντε επίπεδα LoD μπορεί να χρησιμοποιηθεί για την αξιολόγηση της ποιότητας ενός συνόλου 3D μοντέλου δεδομένων πόλεων . Επιπλέον, η κατηγορία των επιπέδων LoD κάνει τα σύνολα δεδομένων συγκρίσιμα και έτσι υποστηρίζει τη διαδικασία ένταξης αυτών των συνόλων.

Περαιτέρω εξελίξεις πέραν της έκδοσης 1.0 μπορεί να αντιμετωπίσει δυναμικά χαρακτηριστικά (π.χ. κινούμενα αντικείμενα, παλίρροιες νερού), το χρόνο και την ιστορία, και, ενδεχομένως, την ενσωμάτωση της αρχής της γεωμετρικής μοντελοποίησης της Εποικοδομητικής Στερεάς Γεωμετρίας (Constructive Solid Geometry).

Συμπερασματικά, το εργαλείο CityGML είναι το αποτέλεσμα μιας καλά οργανωμένης προσπάθειας να φέρει σε επαφή τα σχετικά πρότυπα με προσεκτικό συντονισμό με τις διάφορες ομάδες των προτύπων και των ομάδων χρηστών.

2.7.5 CityEngine

Ένα αξιόλογο έργο στον τομέα της διαδικαστικής παραγωγής πόλεων ονομάζεται CityEngine, η εξέλιξη της δουλειάς προτάθηκε από τους [Parish Y. I. H., Müller P. 2001]. Η μηχανή χρησιμοποιεί μια διαδικαστική προσέγγιση που βασίζεται σε L-systems για την μοντελοποίηση των πόλεων . Από διάφορους χάρτες απεικόνισης που δίνονται ως είσοδος στο σύστημα, όπως τα σύνορα μεταξύ θάλασσας και ξηράς, και η πυκνότητα του πληθυσμού, το CityEngine δημιουργεί ένα σύστημα εθνικών οδών και δρόμων, χωρίζει τη γη σε τμήματα, και δημιουργεί την κατάλληλη γεωμετρία για τα κτίρια στις αντίστοιχες διανομές. Ένα σύστημα υφής με βάση τα στοιχεία υφής και τις διαδικαστικές μεθόδους επιτρέπει την προσθήκη οπτικού ορισμού στα κτίρια.

Το Esri CityEngine είναι μια τρισδιάστατη (3D) εφαρμογή λογισμικού μοντελοποίησης που αναπτύχθηκε από την Esri R & D Center Zurich και ειδικεύεται στην παραγωγή 3D αστικών περιβαλλόντων. Με τη προσέγγιση διαδικαστικής μοντελοποίησης, το CityEngine επιτρέπει την αποτελεσματική

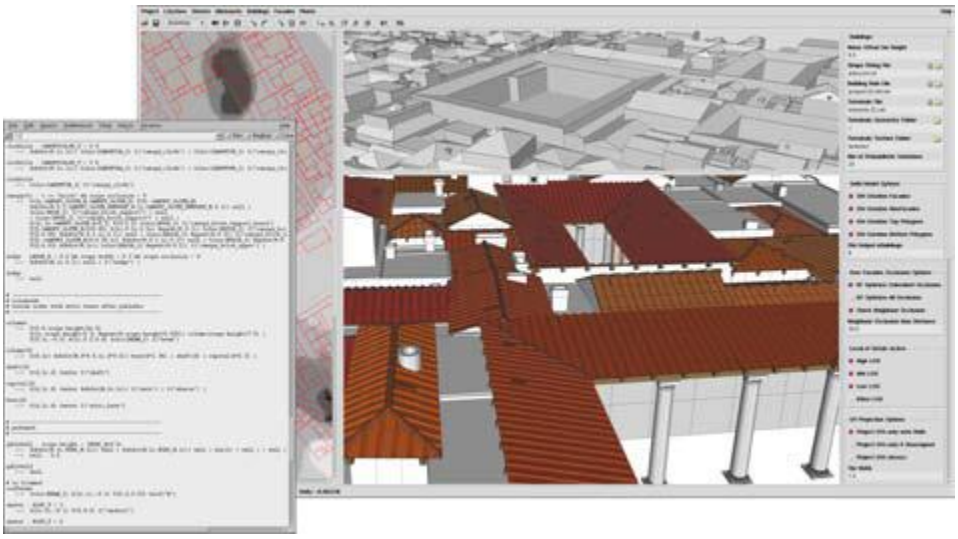
δημιουργία λεπτομερών μεγάλης κλίμακας 3D πόλεων με μόνο μερικά κλικ του ποντικιού, αντί του εξαντλητικού χρόνου και της εντατικής εργασίας δημιουργίας αντικειμένου και χειροκίνητης τοποθέτησης. Το CityEngine συνεργάζεται με την αρχιτεκτονική τοποθέτηση αντικειμένων και διάταξης με τον ίδιο τρόπο που η Εικονογραφημένη Αντίληψη Περιβάλλοντος (VUE, Visual Understanding Environment) διαχειρίζεται το έδαφος, τα οικοσυστήματα και τη χαρτογράφηση της ατμόσφαιρας και είναι εξίσου διαφορετικά όσον αφορά την ικανότητα του χειρισμού αντικειμένων και περιβαλλοντικής πιστότητας και αρμονίας, σε σχέση με το VUE . Η πρόσφατη απόκτηση του CityEngine από την Esri έχει ως στόχο να προωθήσει τις καινοτομίες σε 3D GIS και της τεχνολογίας geodesign.

Τα χαρακτηριστικά του CityEngine είναι τα εξής [Parish Y. I. H., Müller P. 2001]:

- EGIS / CAD Δεδομένα: Υποστήριξη για πρότυπα στάνταρ βιομηχανίας , όπως αρχείων της ESRI, της AutoCAD (CAD format), geoDatabase αρχείων και OpenStreetMap που επιτρέπουν την εισαγωγή / εξαγωγή κάθε γεωχωρικών / διανυσματικών δεδομένων.Το CityEngine δημιουργεί αστικά περιβάλλοντα από το μηδέν, τα οποία βασίζονται σε μια ιεραρχική σειρά ευνόητων κανόνων που μπορούν να επεκταθούν ανάλογα με τις ανάγκες των χρηστών.
- Πυρήνα Μοντελοποίησης βασισμένο σε κανόνες (Rule-based Modeling Core): διαδικαστική μοντελοποίηση με βάση CGA κανόνων που επιτρέπουν να ελέγχουν μάζες, γεωγραφικά στοιχεία, αναλογίες, ή υφές κτηρίων ή δρόμους σε μια πόλη με μια ευρεία κλίμακα.
- Παραμετρική Μοντελοποίηση Διεπαφής (Parametric Modeling Interface) : Μια εύχρηστη διεπαφή για τον αλληλεπιδραστικό έλεγχο συγκεκριμένων οδικών ή κτιριακών παραμέτρων, όπως το ύψος ή η ηλικία (ορίζεται από τους κανόνες CGA).
- Δυναμικά σχεδιαγράμματα πόλης (Dynamic City Layouts): διαδραστικός σχεδιασμός, επεξεργασία και τροποποίηση αστικών σχεδιαγραμμάτων που αποτελούνται από μπλοκ, δρόμους, και αγροτεμάχια.
- Μοντελοποίηση Πόλης Ελεγχόμενου Χάρτη (Map-Controlled City Modeling): Οποιαδήποτε παράμετρος των κτιρίων και των δρόμων μπορεί να ελεγχθεί καθολικά μέσω των χαρτών εικόνας (για παράδειγμα, τα ύψη των κτιρίων).
- Πρότυπα οδικών Δικτύων (Street Networks Patterns) : Μοναδικά εργαλεία ανάπτυξης δρόμων για γρήγορο σχεδιασμό και κατασκευής αστικών διατάξεων.
- Οδηγός Πρόσοψης (Facade Wizard): Γρήγορη δημιουργία κανόνων από μια εικόνα ή μια υφή μαζικών μοντέλων με αυτό το απλό και εύκολο στη χρήση οπτικό εργαλείο συγγραφής πρόσοψης.
- Τρισδιάστατες Μορφοποιήσεις Βιομηχανικών Προτύπων (Industry-Standard 3D Formats) : το CityEngine υποστηρίζει Collada, Autodesk FBX, 3DS, OBJ Wavefront, RenderMan RIB, mental ray MI και e-on software's Vue, τα οποία επιτρέπουν άψογη ανταλλαγή δεδομένων.

- Πληροφόρησης (BIM για τις Πόλεις): Προσαρμοσμένες εκθέσεις βασισμένες σε κανόνες μπορούν να παραχθούν για να αναλύσουν τον αστικό σχεδιασμό π.χ. υπολογίζει αυτόματα τις ποσότητες, όπως GFA, FAR, κλπ.
- Python: Ειδικές εργασίες αγωγού με την ενσωματωμένη διεπαφή δέσμης ενεργειών της Python(Python scripting interface).
- Διαθέσιμο για όλες τις πλατφόρμες: διαθέσιμη για τα Windows (32/64bit), Mac OSX (64bit) και Linux (32/64bit).

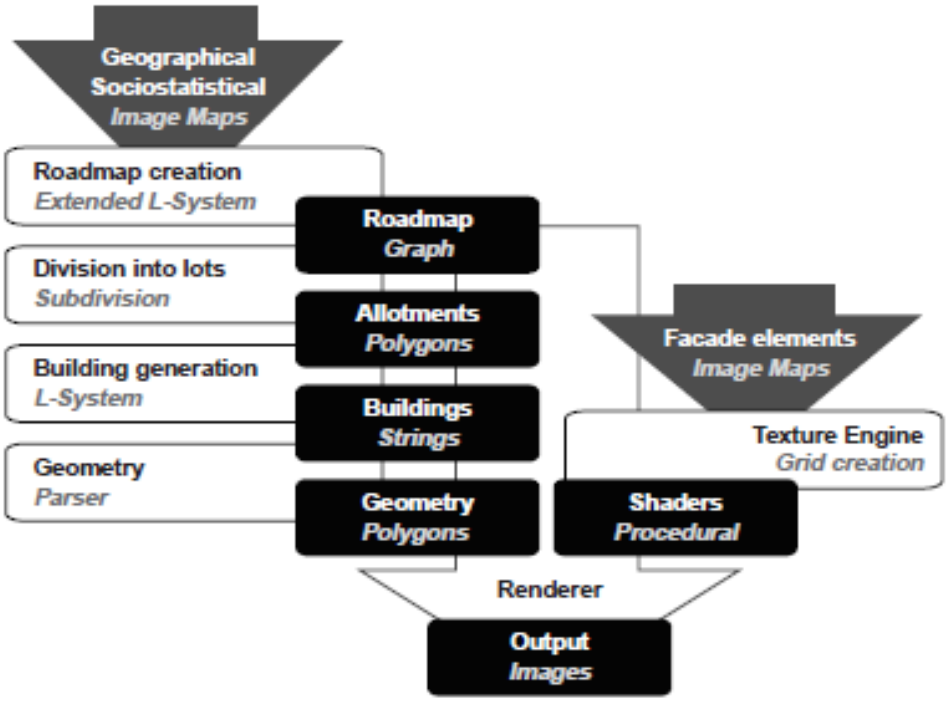
Στην Εικόνα 28 εμφανίζεται ένα στιγμιότυπο της διεπαφής χρήστη CityEngine. Μπορούμε να εισάγουμε περισσότερες μορφές δεδομένων GIS, συμπεριλαμβανομένων ράστερ χάρτες και KML format του Google Earth για την κατασκευή μαζικών μοντέλων . Παρόμοια με άλλες εφαρμογές μοντελοποίησης , το CityEngine βασίζεται σε πολλές διαφορετικές οπτικές του μοντέλου που οδηγούν σε μια επαναληπτική διαδικασία σχεδιασμού. Οι πιο συχνά χρησιμοποιούμενες προβολές είναι: (1) Η λειτουργία επισκόπησης για να δείξει ίχνη κτιρίων, δρόμων και όρια του αγροτεμαχίου. (2) Μία τρισδιάστατη όψη της επιμέρους παραγωγή της γραμματικής π.χ. μέχρι την απεικόνιση μαζικών μοντέλων 3D. (3) τρόπους προεπισκόπησης της τελικής γεωμετρίας των επιλεγμένων υποτμημάτων. (4) Διάφορα εργαλεία για να απεικονίσει τις ρυθμίσεις σχήματος, την προοπτική των σχημάτων, τα τελειώματα επιφανειών, snap lines και την τοπολογία των σχημάτων παραγωγής δέντρων για τον οπτικό εντοπισμό σφαλμάτων. (5) Ένας επεξεργαστής κανόνων (rule editor) για τη γραμματική σχήματος.



Εικόνα 28: Στιγμιότυπο από το CityEngine, το περιβάλλον CGA σχήματος μοντελοποίησης . Στην αριστερή πλευρά του κύριου παραθύρου είναι ένα GIS-like viewer για να εμφανίζει τη διάταξη της πόλης και στα δεξιά ένα OpenGL προεπισκόπηση για να δείτε επιλεγμένα τμήματα της παραγόμενης γεωμετρίας. Το παράθυρο στο μπροστινό μέρος περιέχει τον επεξεργαστή κανόνα (rule editor).

Το σύστημα CityEngine αποτελείται από πολλά διαφορετικά εργαλεία τα οποία σχηματίζουν τον αγωγό που δείχνει η Εικόνα 29. Στο πρώτο βήμα, τα δεδομένα

εισόδου τροφοδοτούνται στο σύστημα οδικής παραγωγής, χρησιμοποιώντας ένα εκτεταμένο L-συστήματος. Οι περιοχές μεταξύ των οδών στη συνέχεια υποδιαιρούνται για τον καθορισμό των μεριδίων όπου τα κτίρια τοποθετούνται σε αυτά. Σε ένα τρίτο βήμα, εφαρμόζοντας ένα άλλο L-σύστημα, τα κτίρια που δημιουργούνται ως αναπαράσταση συμβολοσειράς Boolean λειτουργίες σε απλά στερεά σχήματα. Τέλος, ένα πρόγραμμα ανάλυσης ερμηνεύει όλα τα αποτελέσματα για το λογισμικό απεικόνισης. Το λογισμικό απεικόνισης θα πρέπει να είναι σε θέση να επεξεργαστεί πολυγωνική γεωμετρία και χάρτες υφής. Αυτή είναι η περίπτωση για σχεδόν οποιαδήποτε 3D φωτοαποδοσης. Επιπλέον, οι περισσότεροι επεξεργαστές φωτοαπόδοσης scanline (scanline renderers [2]) υποστηρίζουν διαδικαστικές υφές, έτσι ώστε ο προτεινόμενος μηχανισμός για να δημιουργήσει προσόψεις των κτιρίων μπορούν να ενσωματώνονται μέσα στον αγωγό.



Εικόνα 29: Ο αγωγός του εργαλείου δημιουργίας της πόλης. Τα σκούρα κουτιά καταγράφουν τα αποτελέσματα και τα δεδομένα των δομών των επιμέρους εργαλείων στα λευκά τετράγωνα.

Το CityEngine χρησιμοποιεί μια προσέγγιση διαδικαστικής μοντελοποίησης που σημαίνει ότι δημιουργεί αυτόματα μοντέλα μέσα από ένα προκαθορισμένο σύνολο κανόνων. Οι κανόνες ορίζονται μέσω ενός συστήματος γραμματικής σχήματος CGA που επιτρέπει τη δημιουργία πολύπλοκων παραμετρικών μοντέλων. Ένας

[2] Scanline rendering είναι ένας αλγόριθμος για τον προσδιορισμό ορατής επιφάνειας, σε 3D γραφικά υπολογιστών, που λειτουργεί με βάση γραμμή προς γραμμή και όχι βάση πολύγωνο-ανά-πολύγωνο ή pixel-ανά-pixel . Όλα τα πολύγονα πρέπει να καθίστανται πρώτα ταξινόμηση κατά την κορυφή y συντεταγμένη στην οποία εμφανίζονται για πρώτη φορά, μετά κάθε γραμμή ή σάρωση γραμμής της εικόνας υπολογίζεται χρησιμοποιώντας την τομή της γραμμής σάρωσης με τα πολύγωνα στο μπροστινό μέρος της ταξινομημένης λίστας, ενώ η ταξινομημένη λίστα ενημερώνεται για να απορρίψει τα όχι πλέον ορατά πολύγωνα, όπως η ενεργή γραμμή σάρωσης προωθείται κάτω από την εικόνα.

χρήστης μπορεί να αλλάξει ή να προσθέσει τη γραμματική σχήματος όσο χρειάζεται παρέχοντας χώρο για νέες δυνατότητες σχεδιασμού.

Μοντελοποιώντας ένα αστικό περιβάλλον μέσα στο CityEngine συνήθως ξεκινά με τη δημιουργία ενός δικτύου δρόμου είτε με το εργαλείο σχεδίασης δρόμων ή μέσω των χαρτών που εισάγονται από σελίδες όπως το openstreetmap.org. Το επόμενο βήμα είναι να υποδιαιρεθούν όλα τα τμήματα όσες φορές ορίζονται και προκύπτουν σε ένα αστικό χάρτη των τμημάτων και δρόμων (βλ. New York 2259 παράδειγμα). Τώρα, επιλέγοντας το σύνολο ή μέρος των τμημάτων αυτών το CityEngine μπορεί να διαταχθεί ξεκινώντας να παράγει κτίρια. Λόγω της υποκείμενης τεχνολογίας διαδικαστικής μοντελοποίησης όλα τα κτίρια μπορούν να δημιουργηθούν για να διαφέρουν το ένα από το άλλο για να επιτευχθεί μια αισθητική αστικού περιβάλλοντος. Σε αυτό το σημείο το μοντέλο πόλεων μπορεί να επανασχεδιαστεί και να ρυθμιστεί με αλλαγή των παραμέτρων ή την ίδια γραμματική σχήματος.

2.7.6 Unity3D

Το Unity είναι ένα ολοκληρωμένο εργαλείο δημιουργίας για τη δημιουργία 3D παιχνίδια βίντεο ή άλλου περιεχόμενου, όπως αρχιτεκτονικών απεικονίσεων ή real-time 3D animations.

Το Unity είναι ένα cross-platform μηχανή παιχνιδιού με ενσωματωμένο IDE που αναπτύχθηκε από τη Unity Technologies. Χρησιμοποιείται για την ανάπτυξη παιχνιδιών βίντεο για τις πρόσθετες λειτουργείες δικτύου (web plugins) , desktop πλατφόρμες, κονσόλες και φορητές συσκευές, και χρησιμοποιείται από πάνω από ένα εκατομμύριο προγραμματιστές. Το μερίδιο αυτό αυξήθηκε από OS X εργαλείο ανάπτυξης παιχνιδιών που υποστηρίζονται το 2005 σε μια μηχανή παιχνιδιού multi-platform [Sue Blackman, 2011].

Η νεότερη ενημερωμένη έκδοση είναι το, Unity 4.2.2, που κυκλοφόρησε τον Οκτώβριο του 2013. Υποστηρίζει σήμερα την ανάπτυξη για iOS, Android, Windows, Blackberry 10, OS X, Linux, προγράμματα περιήγησης στο Web, Flash, PlayStation 3, Xbox 360, Windows Phone 8, και το Wii U [Sue Blackman, 2011]. Δύο εκδόσεις της μηχανής παιχνιδιού είναι διαθέσιμα για download, Unity και Unity Pro.

Η μηχανή γραφικών χρησιμοποιεί Direct3D (Windows, Xbox 360), OpenGL (Mac, Windows, Linux, PS3), OpenGL ES (Android, iOS), και ιδιαίτερες APIs (Wii). Υπάρχει υποστήριξη για τη χαρτογράφηση πρόσκρουσης (bump mapping), χαρτογράφηση αντανάκλασης, parallax χαρτογράφηση, οθόνη χώρου του περιβάλλοντος σύγκλεισης (SSAO, Screen Space Ambient Occlusion), δυναμικές

σκιές που χρησιμοποιούν χάρτες σκιάς, render-to-texture και πλήρους οθόνης μετά την επεξεργασία εφέ [Sue Blackman, 2011].

Το Unity υποστηρίζει στοιχεία ενεργητικού τέχνης και μορφές αρχείων από το 3ds Max, Maya, Softimage, Blender, modo, ZBrush, Cinema 4D, Cheetah3D, Adobe Photoshop, Adobe Fireworks και Allegorithmic Substance. Αυτά τα στοιχεία ενεργητικού μπορούν να προστεθούν στο έργο παιχνιδιού , και να διαχειρίζεται μέσω γραφικής διεπαφής χρήστη Unity [Sue Blackman, 2011].

Το scripting της μηχανής παιχνιδιού είναι χτισμένο σε Mono 2.6, του ανοικτού κώδικα εφαρμογής πλαισίου .NET. Οι προγραμματιστές μπορούν να χρησιμοποιήσουν UnityScript (μια προσαρμοσμένη γλώσσα με ECMAScript εμπνευσμένη από σύνταξη, που αναφέρεται ως JavaScript από το λογισμικό), C #, ή Boo (η οποία η σύνταξή της γλώσσας αυτής είναι εμπνευσμένη από τη σύνταξη της Python) [Sue Blackman, 2011]. Ξεκινώντας με την έκδοση 3.0, το Unity έχει προσαρμοσμένη την έκδοση του MonoDevelop για σενάρια αποσφαλμάτωσης [Sue Blackman, 2011].

Το Unity3D δεν είναι αποκλειστικά ένα εργαλείο διαδικαστικής μοντελοποίησης, όπως είδαμε, αλλά ένα γενικό εργαλείο αναπαράστασης τρισδιάστατων αντικειμένων και αλληλεπίδρασης με αυτών. Στην εν λόγω πτυχιακή χρησιμοποιείται σαν εργαλείο αναπαράστασης των κτιρίων της Κέρκυρας προσφέροντας υψηλή ακρίβεια και ευκρίνεια στα απεικονίζοντα μοντέλα. Στην Εικόνα 30, βλέπουμε το GUI της μηχανής παιχνιδιού Unity.

Εικόνα 30: Αναπαράσταση μιας πόλης με κυβικά αντικείμενα με τη βοήθεια χάρτη του Google Maps (επάνω), αναπαράσταση μιας πόλης με εργαλεία του Unity (κάτω).

Παρακάτω, στα επόμενα κεφάλαια, θα φανούν πτυχές της εφαρμογής με μεγάλη λεπτομέρεια πάνω στο αντικείμενο του έργου που παρουσιάζεται.

ΕΠΙΛΟΓΟΣ

Στο Κεφάλαιο αυτό αναδείξαμε τον τρόπο αναπαράστασης διαδικαστικής παραγωγής αστικού περιβάλλοντος και εμβαθύναμε στον τρόπο απεικόνισης οδικών δικτύων και κτιρίων αλλά και βλάστησης όπως παρατηρούνται και στο φυσικό μας περιβάλλον.

Είδαμε μερικά από τα εργαλεία οπτικοποίησης που χρησιμοποιούνται στην αγορά για την επίτευξη δυσδιάστατων αλλά και τρισδιάστατων μοντέλων απεικόνισης με τη χρήση πλατφόρμων λογισμικού μέσω γλωσσών προγραμματισμού και πλαισίων. Μερικά από αυτά παρέχουν στους χρήστες τους ένα δυναμικό σχεδιασμό μοντέλων όπως το CityEngine και Unity3D, ενώ κάποια άλλα που ασχολούνται αποκλειστικά με την επίδειξη δισδιάστατων αστικών χώρων για την ανάδειξη πληθυσμού κτιρίων και εκτάσεων πόλης, χαμηλής οπτικής λεπτομέρειας, όπως το CityBuilder. Εργαλεία που βοήθησαν και εξακολουθούν να βοηθούν όχι μόνο προγραμματιστές και σχεδιαστές χώρων αλλά και ανθρώπους στον κλάδο της αρχιτεκτονικής.

ΚΕΦΑΛΑΙΟ 3

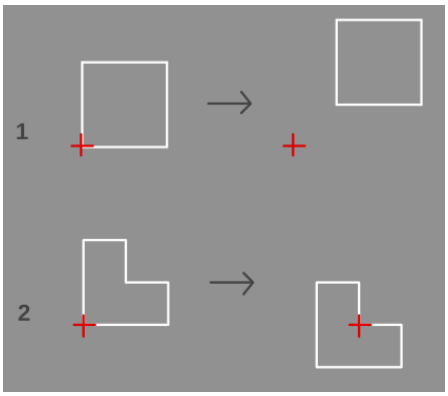
Κανόνες Γραμματικής Σχήματος

ΕΙΣΑΓΩΓΗ

Μια γραμματική σχήματος (Shape grammar) είναι ένας φυσικός τρόπος για να εκπροσωπείται ένα αρχιτεκτονικό στυλ έτσι ώστε να μπορεί να αναπαραχθεί σε κτίρια διαφόρων μεγεθών. Οι γραμματικές σχήματος εισήχθησαν για πρώτη φορά από τον George Stiny και James Gips σε δημοσίευσή τους "Shape Grammars and the Generative Specification of Painting and Sculpture " [Stiny & Gips, 1971]. Επίσης, μια γραμματική σχήματος είναι ένα σύνολο κανόνων που εφαρμόζονται με τρόπο βήμα-βήμα για να δημιουργήσει ένα σύνολο, ή γλώσσας, σχεδίων. Οι γραμματικές σχήματος είναι τόσο περιγραφικές όσο και παραγωγικές. Οι κανόνες της γραμματικής σχήματος παράγουν είτε υπολογίζουν σχέδια, και οι ίδιοι οι κανόνες είναι οι περιγραφές των μορφών των παραγόμενων σχεδίων [Knight TW, 1992].

Γενικότερα, οι γραμματικές σχήματος στον υπολογισμό μοντέλων αποτελούν μια ειδική κατηγορία συστημάτων παραγωγής που δημιουργούν γεωμετρικά σχήματα. Τυπικά, τα σχήματα είναι 2 - ή 3-διαστάσεων, αφού οι γραμματικές το σχήματος είναι ένας τρόπος για τη μελέτη 2 - και 3-διαστάσεων γλώσσες.

Μια γραμματική σχήματος αποτελείται από τους κανόνες σχήματος και μια μηχανή παραγωγής που επιλέγει και επεξεργάζεται κανόνες. Ένας κανόνας σχήματος καθορίζει με ποιόν τρόπο ένα υπάρχον (ή μέρος αυτού) σχήμα μπορεί να μεταμορφωθεί. Ένας κανόνας σχήματος αποτελείται από δύο μέρη που χωρίζονται από ένα βέλος με κατεύθυνση από αριστερά προς τα δεξιά. Το μέρος αριστερά του βέλους ονομάζεται η αριστερή πλευρά (LHS, Left-Hand Side). Απεικονίζει μια κατάσταση από την άποψη ενός σχήματος και ενός δείκτη. Το δεξί μέρος του βέλους ονομάζεται η δεξιά πλευρά (RHS, Right-Hand Side). Απεικονίζει πώς το σχήμα LHS πρέπει να μετατραπεί και που ο δείκτης είναι τοποθετημένος. Ο δείκτης βοηθά να εντοπιστεί και να προσανατολίσει το νέο σχήμα (Εικόνα 31).



Εικόνα 31: Το μέρος αριστερά του κάθε βέλους ονομάζεται η αριστερή πλευρά (LHS). Απεικονίζει μια κατάσταση από την άποψη ενός σχήματος και ενός δείκτη. Το δεξί μέρος του κάθε βέλους ονομάζεται

η δεξιά πλευρά (RHS). Απεικονίζει πώς το σχήμα LHS πρέπει να μετατραπεί και που ο δείκτης είναι τοποθετημένος.

Μια γραμματική σχήματος αποτελείται ελάχιστα από τρεις κανόνες σχήματος: κανόνα έναρξης, τουλάχιστον έναν κανόνα μετασχηματισμού, και κανόνα τερματισμού. Ο κανόνας έναρξης είναι απαραίτητος για να ξεκινήσει η διαδικασία παραγωγής σχήματος. Ο κανόνας τερματισμού είναι απαραίτητος ώστε η διαδιακασία παραγωγής σχήματος να σταματήσει. Ο απλούστερος τρόπος για να σταματήσει η διαδικασία είναι κατά κανόνα σχήματος που αφαιρεί το δείκτη. Οι γραμματικές σχήματος διαφέρουν από τις γραμματικές του Chomsky [Sipser, 1996] από μια σημαντική άποψη: οι κανόνες παραγωγής μπορούν να εφαρμόζονται σειριακά (όπως οι γραμματικές Chomsky) ή παράλληλα (δεν επιτρέπεται στις γραμματικές Chomsky), παρόμοια με τον τρόπο "παραγωγής" που εφαρμόζονται στα L-systems.

Ένα σύστημα γραμματικής σχήματος έχει επιπλέον ένα χώρο εργασίας όπου η δημιουργημένη γεωμετρία προβάλλεται. Η μηχανή παραγωγής ελέγχει την υπάρχουσα γεωμετρία, που συχνά αναφέρεται ως Σχήμα Παρούσας Εργασίας (Σχήμα Παρούσας Εργασίας), για τις συνθήκες που ταιριάζουν με το LHS, Left-Hand Side των κανόνων σχήματος. Κανόνες σχήματος με το ταίριασμα LHS, Left-Hand Side είναι επιλέξιμα για χρήση. Εάν περισσότεροι από ένας κανόνας ισχύει, η μηχανή παραγωγής έχει να επιλέξει ποιο κανόνα να εφαρμόσει. Στο εναλλακτικό σενάριο, η μηχανή επιλέγει έναν από τους κανόνες γραμματικής και στη συνέχεια προσπαθεί να βρει όλους τους αγώνες της LHS, Left-Hand Side του κανόνα αυτού στο Σχήμα Παρούσας Εργασίας. Εάν υπάρχουν πολλές αντιστοιχίες, η μηχανή μπορεί (ανάλογα με τη διαμόρφωση / εφαρμογή της) [Sipser, 1996]:

- να εφαρμόσει τον κανόνα σε όλες τις αντιστοιχίσεις, παράλληλα,
- να εφαρμόσει τον κανόνα για όλες τις αντιστοιχίσεις σειριακά (το οποίο θα μπορούσε να οδηγήσει σε ασυνέπειες) ή να επιλέξει μια από τις εντοπισμένες αντιστοιχίες και να εφαρμόσει τον κανόνα μόνο σε αυτή την αντιστοίχιση.

Οι γραμματικές σχήματος είναι πιο χρήσιμες όταν περιορίζονται σε ένα μικρό, καλά καθορισμένο πρόβλημα παραγωγής, όπως σχεδιαγράμματα κατοικιών και βελτίωση δομής. Επειδή οι κανόνες σχήματος συνήθως ορίζονται σε μικρά σχήματα, μια γραμματική σχήματος μπορεί να περιέχει γρήγορα πολλούς κανόνες. Η γραμματική σχήματος Palladian villas που παρουσιάζονται από τον [Mitchell W., 1990] για παράδειγμα, περιέχει 69 κανόνες, που εφαρμόζονται σε ολόκληρα τα οκτώ στάδια.

Παρά τη δημοτικότητά τους και τη δυνατότητα εφαρμογής τους σε ακαδημαϊκούς κύκλους, οι γραμματικές δεν έχουν βρει ευρεία χρήση σε γενικές με τη βοήθεια υπολογιστή εφαρμογές σχεδιασμού.

3.1 Γραμματική Σχήματος

Οι γραμματικές σχήματος (shape grammar) περιέχουν κανόνες σχετικά με το γεωμετρικό χειρισμό και το μετασχηματισμό, ενώ το μοντέλο που βασίζεται στη γνώση (knowledge-based model) περιέχει πληροφορίες για τις χωρικές χρήσεις, την οργάνωση, τα στοιχεία και συναφείς πληροφορίες. Τα κτίρια αναλύονται και σχεδιαγράμματα δημιουργούνται μέσα από την επικοινωνία και την αλληλεπίδραση μεταξύ αυτών των δύο συστημάτων.

Η οντολογία περιέχει πληροφορίες σχετικά με σχέδια οικοδόμησης, θέσης, χρήσης, προσανατολισμού και μεγέθους, αλλά δεν αποτυπώνει μορφή σε κτίρια με γεωμετρική έννοια. Πρόκειται για αναπαράσταση γνώσης σχετικά με ένα θέμα, και περιγράφει το καθένα ξεχωριστά ως βασικά αντικείμενα, κλάσεις, όπως συλλογές ή τύπους ιδιοτήτων αντικειμένων, καθώς και τα χαρακτηριστικά, και τις σχέσεις μεταξύ των αντικειμένων. Εδώ, η οντολογία χρησιμοποιείται για να συλλάβει τη γνώση σχετικά με αρχιτεκτονικές αρχές σχεδιασμού, την οικοδομική την ανατομία, τη δομή και τα συστήματα. Υπάρχουν ορισμένες ομοιότητες μεταξύ της γραμματικής σχήματος και βάση γνώσης του μοντέλου και κυρίως ότι και οι δύο περιέχουν κανόνες σχεδιασμού, ωστόσο, η φύση του κανόνα ποικίλλει.

Οι γραμματικές σχήματος είναι υπολογιστικοί κανόνες για την παραγωγή γεωμετρικών σχημάτων, και υπάρχουν δύο τύποι, αναλυτικές και πρωτότυπες. Οι αναλυτικές γραμματικές αναπτύχθηκαν για να περιγράψουν και να αναλύσουν ιστορικά στυλ ή σχέδια από συγκεκριμένους αρχιτέκτονες [Stiny G & Mitchell WJ, 1978], [Chiou SC & Krishnamurti R, 1995], [Cagdas G, 1996], [Duarte JP, 2005]. Οι αναλυτικές γραμματικές χρησιμοποιούν σύνολα υφιστάμενων σχεδίων, το σώμα, για την ανάπτυξη της γλώσσας, και να συναγάγει τους κανόνες. Οι γραμματικές δοκιμάζονται χρησιμοποιώντας τους κανόνες τόσο για τη δημιουργία σχεδίων στο σώμα, όσο και νέων σχεδίων. Οι πρωτότυπες γραμματικές βασίζονται σε γενικευμένους κανόνες και έχουν ως στόχο να δημιουργήσουν στιγμιότυπα του αρχικού στυλ των σχεδίων. Αυτοί οι τύποι γραμματικών δεν έχουν ευρέως αντιμετωπιστεί ως αναλυτικές, λόγω της δυσκολίας «μετάφρασης της αφηρημένης, πειραματικής μορφής εντός αρχιτεκτονικών σχεδίων που ταιριάζουν με συγκεκριμένα πλαίσια σχεδιασμού ή προγράμματα" [Knight TW, 1992].

Η συνδυαστική φύση των κανόνων σχεδιασμού συλλαμβάνονται από γραμματικές σχήματος και η οντολογία προσφέρει τη δυνατότητα ότι η γνώση του σχεδιασμού μπορεί άμεσα να εκπροσωπείται, διατηρείται και επεξεργάζεται. Από την άποψη αυτή, [Nelson CB & Fenves SJ, 1990] παρατηρούν: Σε ένα σενάριο μηχανικής, η ανάλυση και ο σχεδιασμός είναι οι παρόμοιες διαδικασίες χειρισμού συνεχούς γνώσης. Όταν οι μηχανικοί χρησιμοποιούν γνώσης για τη δημιουργία νέων προϊόντων, η διαδικασία αυτή ονομάζεται σχεδιασμού. Η διαδικασία δημιουργίας νέων αντικειμένων θα ονομάζονται παραγωγή. Η διαδικασία ελέγχου των αντικειμένων θα ονομάζεται ανάλυση. Ως εκ τούτου, εάν η γνώση της μηχανικής μπορεί να συλληφθεί (π.χ., πόσο τα μηχανικά αντικείμενα ταιριάζουν μεταξύ τους καθώς επίσης και η λειτουργία του κάθε αντικειμένου) και να εκπροσωπείται

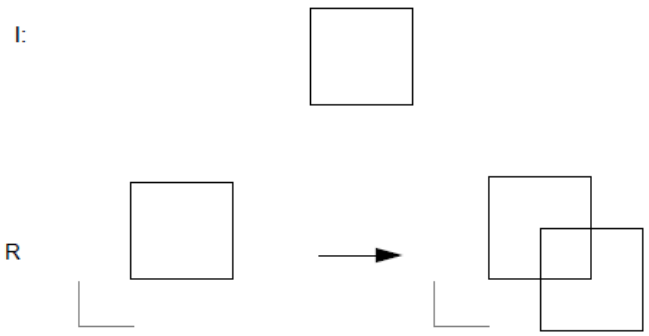
χρησιμοποιώντας μια γραμματική, τόσο ο σχεδιασμός όσο και η ανάλυση μπορούν να εκτελεστούν.

Ένας παραλληλισμός μπορεί να γίνει μεταξύ των μηχανικών και των προϊόντων, και των αρχιτεκτόνων με τα κτίρια. Η αρχιτεκτονική γνώση του σχεδιασμού αποτυπώνεται στην οντολογία, και επεξεργάζονται από τη γραμματική σχήματος, επιτρέποντας έτσι την παραγωγή και την ανάλυση. Με αυτή την έννοια, οποιαδήποτε διαφορά μεταξύ των αναλυτικών και πρωτότυπων γραμματικών σχήματος ελαττώνεται, ως γενικοί κανόνες μπορούν να είναι προσαρμόσιμοι ανάλογα με το περιεχόμενο, το μέγεθος του κτιρίου, το ύφος ή τη λειτουργία.

3.2 Παραγωγή Γραμματικής Σχήματος

Αυτό το απλό σχέδιο γραμματικής σχήματος που παρουσιάζεται στην Εικόνα 33 δημιουργεί σχέδια για ένα ορθογώνιο πλέγμα. Το αρχικό σχήμα είναι ένα τετράγωνο (I). Ο κανόνας (R) καθορίζει μια πιθανή επέμβαση αντικατάστασης σχήματος που παρουσιάζεται στην Εικόνα 32, που ενώνει δύο σχήματα: ένα μοτίβο, ένα τετράγωνο, και την αντικατάστασή του, το ίδιο τετράγωνο με την προσθήκη ενός μετακινούμενου αντιγράφου που έχει οριστεί.

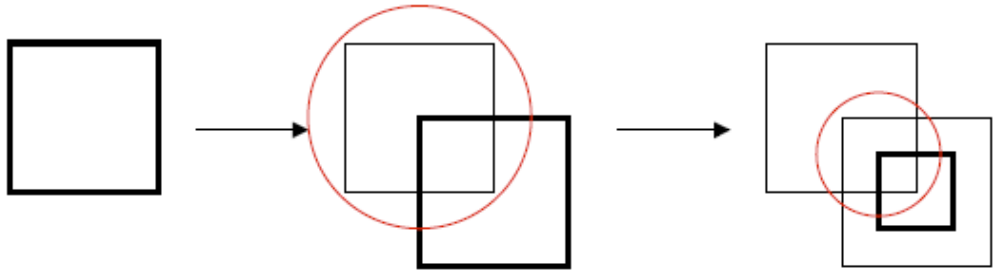
Ένας κανόνας ισχύει σε ένα σχήμα κάθε φορά που μια παρόμοιο μετασχηματισμό κάνει το μοτίβο μέρος , π.χ. κάθε τετραγώνου, επιτρέποντας το μετασχηματισμένο μοτίβο να αντικατασταθεί από το σχήμα αντικατάστασης που καταλλήλως μετασχηματίστηκε [Knight TW, 1992].



Εικόνα 32: Απλό σχήμα Γραμματικής Σχήματος.

Εφαρμόζοντας τον κανόνα στο αρχικό σχήμα δίδει ένα σχήμα που περιέχει τρία τετράγωνα, δύο στο ίδιο μέγεθος με το αρχικό σχήμα και το ένα στο μισό μέγεθος, που προκύπτει από τα δύο. Η εφαρμογή του κανόνα στο μικρότερο τετράγωνο

αποδίδει τετράγωνα σε τρία μεγέθη: δύο με μέγεθος ίσο με το αρχικό τετράγωνο, δύο με το μισό μέγεθος, και ένα ένα τέταρτο του αρχικού μεγέθους (Εικόνα 33).



Εικόνα 33: Αποτέλεσμα εφαρμογή γραμματικών κανόνων Εικόνας 32.

Η διαδικασία ανάπτυξης και χρήση μιας γραμματικής σχήματος μπορεί να διαιρεθεί σε διάφορες λογικές φάσεις [Knight TW, 1992].

1. Δημιουργία και τροποποίηση της γραμματικής σχήματος. Ο σχεδιαστής δημιουργεί τους κανόνες και ένα αρχικό σχήμα, και επαληθεύει ή αλλάζει τους χωρικούς και λογικούς περιορισμούς.

2. Μεταγλώττιση της γραμματικής. Ενώ η γίνεται η μετατροπή της γραμματικής στην εσωτερική μορφή, το σύστημα ελέγχει ότι κάθε κανόνας ισχύει πάντα μόνο ενός πεπερασμένου αριθμού τρόπων.

3. Εξερευνώντας τη γλώσσα των σχεδίων που ορίζονται από τη γραμματική. Ο σχεδιαστής εξερευνά τη γλώσσα των σχεδίων, δημιουργεί σχέδια, επιβάλλει πρόσθετους περιορισμούς, σταματά τη διαδικασία παραγωγής, οπισθοχωρεί σε ένα προηγούμενο σχέδιο, ή αποθηκεύει την τρέχουσα κατάσταση. Ο σχεδιαστής μπορεί να ερμηνεύσει τα προκύπτοντα σχέδια σε ένα καμπυλόγραμμο κόσμο και να τα χρησιμοποιεί ως βάση για ένα σχέδιο.

Μία διεργασία παραγωγής σχήματος, το οποίο απεικονίζεται στην Εικόνα 33, σπέρνεται με το σχήμα (I) και στη συνέχεια ακολουθείται από μια αναδρομική κλήση κανόνων σχήματος από το (R). Η εφαρμογή κανόνα αποτελείται από τα ακόλουθα βήματα (Εικόνα 31) [Knight TW, 1992]:

- Εύρεση ένα μέρος του σχήματος που είναι παρόμοιο με την αριστερή πλευρά του κάθε κανόνα.
- Βρείτε γεωμετρικούς μετασχηματισμούς (π.χ. μεγέθους, μετακίνησης, περιστροφής) για να κάνει την αριστερή πλευρά του κανόνα και το τμήμα του εν λόγω σχήματος να ταιριάζει απόλυτα.

- Εφαρμογή των μετασχηματισμών από το προηγούμενο στάδιο στη δεξιά πλευρά του κανόνα και στη συνέχεια να το αντικαταστήσει για το αντιστοιχισμένο τμήμα της εικόνας.

ΕΠΙΛΟΓΟΣ

Παρουσιάστηκαν οι γραμματικές σχήματος και η διαφορά τους με την οντολογία καθώς και η κατηγοριοποίησή τους σε αναλυτικές γραμματικές και πρωτότυπες. Ακόμη δόθηκε με σαφήνεια και παραδείγματα με εικόνες ο τρόπος χρήσης αυτών των γραμματικών κανόνων σχήματος. Οι παραπάνω κανόνες γραμματικής σχήματος, που εφαρμόζονται σε μοντέλα και επεξηγήθηκαν με λεπτομέρεια, αποτελούν προοίμιο για την περιγραφή μιας εξίσου σημαντικής ενότητας και υποκατηγορίας των γραμματικών σχήματος, των γραμματικών σχήματος για Παραγόμενη Αρχιτεκτονική Υπολογιστή (CGA grammars) που ακολουθεί στο επόμενο κεφάλαιο.

ΚΕΦΑΛΑΙΟ 4

Γραμματική Σχήματος για Παραγόμενη Αρχιτεκτονική Υπολογιστή

ΕΙΣΑΓΩΓΗ

Σε αυτό το κεφάλαιο θα παρουσιάσουμε τους βασικούς κανόνες, κανόνες γραμματικής για Παραγόμενη Αρχιτεκτονική Υπολογιστή (CGA shape grammar) και κανόνες τμηματοποίησης (split grammar rules), που στο σύνολό τους συνθέτουν το τεχνικό κομμάτι αυτής της πτυχιακής άσκησης.

Ιστορικά, Ο Peter Wonka, στη δημοσίευσή του "Instant Architecture», προτείνει μια αυτοματοποιημένη προσέγγιση για τις γραμματικές σχήματος. Ο Wonka αντιμετωπίζει τα σχήματα, όπως τα συμβολικά αντικείμενα [Wonka et al. 2003]. Ο ίδιος επεκτείνει το έργο του Stiny [Stiny, G. N., 1975] με την προσθήκη της έννοιας των ιδιοτήτων, όπως το πλάτος ή το ύψος του σχήματος, για κάθε σχήμα στο αλφάβητο. Αυτά τα χαρακτηριστικά μπορούν να περάσουν ως ένα πλαίσιο για τους κανόνες κατά τη διάρκεια της αξιολόγησης. Το πλαίσιο ενός κανόνα διαδίδεται στα παιδιά του, κατανέμοντάς τα ομαλά όσον το δυνατόν γίνεται για να χωρέσουν τους περιορισμούς που ορίζονται από τα χαρακτηριστικά των παιδιών. Για τα (i) μη-τερματικά σχήματα (τα σχήματα που πρέπει ακόμα να αξιολογηθούν), τα χαρακτηριστικά μπορεί να είναι είτε απόλυτα ή σχετικά με αυτά του πλαισίου αξιολόγησης του κανόνα. Για τα (ii) τερματικά σχήματα , τα χαρακτηριστικά τους πρέπει να είναι απόλυτα, όμως το μέγεθος ενός τερματικού σχήματος μπορεί να είναι μικρότερο από το πλαίσιο κανόνων που δίνεται κατά τη διάρκεια της αξιολόγησης. Με τον τρόπο αυτό, η γραμματική του Wonka [Wonka et al. 2003] υπαγορεύει την ατομική, προκατασκευασμένο κομμάτι που θα πρέπει να εισαχθεί σε κάθε δεδομένη θέση επί του κτιρίου. Κωδικοποιώντας την χωρική πληροφορία των σχημάτων στην γραμματική, ο Wonka γενικεύει γραμματικές σχήματος του Stiny, γεγονός που τις καθιστά πιο κατάλληλες για μια εφαρμογή υπολογιστή.

Οι Pascal Müller και Peter Wonka περιγράφουν μια εφαρμογή της προηγούμενης δουλειάς Wonka στο "Procedural Modeling of Buildings", όπως η Παραγόμενη Αρχιτεκτονική Υπολογιστή (CGA, Computer Generated Architecture) γραμματικής σχήματος [Müller et al. 2006]. Εισήγαγαν τις έννοιες των οι κανόνες τμηματοποίησης (split grammar rules) και οι κανόνες επανάληψης (repeat rules). Όπως ορίστηκαν από τους Müller και Wonka οι κανόνες τμηματοποίησης χωρίζουν το πλαίσιο του κανόνα μεταξύ του ακριβούς συνόλου των αποτελεσμάτων παραγωγής του κανόνα. Οι κανόνες επανάληψης είναι μια επέκταση των κανόνων τμηματοποίησης οι οποίοι επιτρέπουν σε ένα σύνολο αποτελεσμάτων παραγωγής να επαναληφθούν όσες φορές χωράει στο πλαίσιο του κανόνα κατά τη διάρκεια της αξιολόγησης.

Πριν όμως αναλύσουμε περαιτέρω, ερχόμαστε να απαντήσουμε στο πρωταρχικό ερώτημα όλων που αναλύθηκε και στα πιο πρώτα κεφάλαια. *Γιατί διαδικαστική μοντελοποίηση;*

Σαν πρώτη απάντηση είναι το γεγονός ότι (α) τα συμβατικά μοντέλα είναι χρονοβόρα: Ας αρχίσουμε με το πιο βασικό και προφανές μειονέκτημα. Όπως έχει ήδη αναφερθεί, ο χειροκίνητος τρόπος μοντελοποίησης ρεαλιστικών μοντέλων είναι εξαιρετικά χρονοβόρος. Τη μεγαλύτερη ανακούφιση που μπορεί να προσφέρει το CGA σχήμα στη μοντελοποίηση είναι η αυτοματοποίηση της διαδικασίας. Μετά έρχεται σαν απάντηση (β) η επανάληψη: Κατά το σχεδιασμό ενός μοντέλου που περιέχει το ίδιο τμήμα αρκετές φορές, είναι απαραίτητο να εκτελέσει πολλές περιττές εργασίες. Να εξεταστεί η πρόσοψη ενός κτιρίου, όπου ο ίδιος τύπος παραθύρου πρέπει να τοποθετηθεί πάνω σε αυτό με ένα σταθερό διάστημα ανάμεσα σε κάθε παράθυρο. Μπορεί να είναι αρκετά κουραστικό να φροντίσει για την ακριβή τοποθέτηση χειροκίνητα. Μια πιο βολική και κομψή προσέγγιση είναι ο ορισμός ορισμένων κανόνων παραγωγής, που παράγουν την επιθυμητή χωρική κατανομή των αντικειμένων. Και (γ) η μετέπειτα επεξεργασία: Αν ο καλλιτέχνης, μετά την ολοκλήρωση του μοντέλου, θέλει να αλλάξει κάτι βασικό, π.χ. το μέγεθος ή το σχήμα, μια ανακατανομή του συνόλου των συνιστωσών του μοντέλου είναι απαραίτητη. Χρησιμοποιώντας μια γραμματική, η προσέγγιση θα είναι πολύ πιο εύκολη: Μια βασική αλλαγή είναι πάντα εύκολο να επιτευχθεί, δεδομένου ότι οι κανόνες παραγωγής στη συνέχεια θα παράγουν και θα διανέμουν στοιχεία του μοντέλου με βάση το νέο σχήμα του.

Αφού συζητήθηκαν διεξοδικά τα οφέλη του σχήματος CGA, μπορούμε τώρα να εξηγήσουμε τις βασικές του έννοιες.

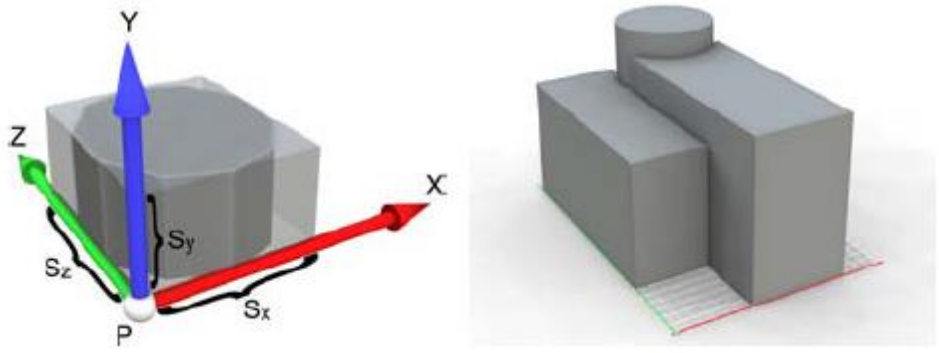
4.1 Βασικές Αρχές CGA σχήματος

Όπως αναφέρθηκε το σχήμα CGA είναι σε γενικές γραμμές μια γραμματική τμηματοποίησης (split grammar). Λειτουργεί σε μια διαδοχική σειρά, διότι αυτό επιτρέπει την καλύτερη αντιμετώπιση δόμησης του μοντέλου. Παρ 'όλα αυτά ο συμβολισμός των γενικών κανόνων παραγωγής είναι παρόμοια με εκείνη των L-systems [Prusinkiewicz & Lindenmayer, 1990].

Το CGA σχήμα είναι μια επέκταση ενός συνόλου γραμματικής , το οποίο εισήχθη από τους [Wonka et al., 2003]. Αν και η ιδέα για τον κανόνα τμηματοποίησης παρουσιάστηκε σε προηγούμενο έργο, ο πραγματικός ορισμός της τμηματοποίησης, συμπεριλαμβανομένης της επανάληψης τμηματοποίησης και η κλιμάκωση των κανόνων είναι η συνεισφορά των Müller και Wonka [Müller et. al, 2006]. Περαιτέρω εισήγαγαν με τη σειρά τους ένα συστατικό τμηματοποίησης, η βάση για τη μοντελοποίηση με μονοδιάστατα, δισδιάστατα και τρισδιάστατα σχήματα. Ο σημειογραφία της γραμματικής και των γενικών κανόνων για την προσθήκη, την κλίμακα, τη μετακίνηση και τη περιστροφή σχημάτων είναι εμπνευσμένα από τα L-systems [Prusinkiewicz & Lindenmayer, 1990], αλλά παρατείνονται για τη μοντελοποίηση της αρχιτεκτονικής. Ενώ οι παράλληλες γραμματικές σχήματος όπως τα L-systems είναι κατάλληλα για να συλλάβουν την

ανάπτυξη στην πάροδο του χρόνου, η διαδοχική εφαρμογή των κανόνων επιτρέπει τον χαρακτηρισμό της δομής π.χ. της γεωγραφικής κατανομής των χαρακτηριστικών και των συστατικών [Prusinkiewicz et al. 2001]. Ως εκ τούτου, το CGA σχήμα είναι μια διαδοχική γραμματική (παρόμοιο με Chomsky γραμματικές σχήματος).

Σχήμα: Η γραμματική λειτουργεί με διαμόρφωση σχημάτων [Müller et al. 2006]: ένα σχήμα αποτελείται από ένα σύμβολο (string), γεωμετρία (γεωμετρικά χαρακτηριστικά) και αριθμητικά χαρακτηριστικά. Τα σχήματα αναγνωρίζονται από τα σύμβολα τους, τα οποία είναι είτε ένα τερματικό σύμβολο ∈ Σ, ή ένα μη τερματικό σύμβολο ∈ V. Τα αντίστοιχα σχήματα ονομάζονται τερματικά σχήματα και μη τερματικά σχήματα. Τα πιο σημαντικά γεωμετρικά χαρακτηριστικά είναι η θέση P, τα τρία ορθογώνια διανύσματα X, Y, και Z, που περιγράφουν ένα σύστημα συντεταγμένων, και ένα μέγεθος διανύσματος S. Αυτά τα χαρακτηριστικά ορίζουν ένα προσανατολισμένο πλαίσιο οριοθέτησης στο διάστημα που ονομάζεται πεδίο εφαρμογής (scope) (βλέπε Εικόνα 34).



Εικόνα 34: Αριστερά: Το πεδίο εφαρμογής του σχήματος. Το σημείο P, μαζί με τους τρείς άξονες X, Y και Z και ένα μέγεθος S ορίζουν ένα κουτί στο χώρο που περιέχει το σχήμα. Δεξιά: Ένα απλό μοντέλο μαζικό κτίριο που αποτελείται από τρία αρχέτυπα σχήματα.

Παραγωγική διαδικασία: Μια ρύθμιση παραμέτρων είναι ένα πεπερασμένο σύνολο βασικών σχημάτων [Müller et al. 2006]. Η παραγωγική διαδικασία μπορεί να ξεκινήσει με μια αυθαίρετη διαμόρφωση των σχημάτων A, που ονομάζεται αξίωμα (axiom), και έχει ως ακολούθως: (1) Επιλέγουμε ένα ενεργό σχήμα με το σύμβολο B στο σύνολο (2) επιλέγουμε έναν κανόνα παραγωγής με το B στην αριστερή πλευρά για να υπολογίσει έναν διάδοχο για το B, ένα νέο σύνολο σχημάτων BNEW (3) σηματοδοτούμε το B σχήμα ως ανενεργό και προσθέτουμε τα σχήματα BNEW της διαμόρφωσης και συνεχίζουμε με το βήμα (1). Όταν η ρύθμιση δεν περιέχει άλλα μη τερματικά σχήματα, η παραγωγική διαδικασία τερματίζεται. Ανάλογα με τον αλγόριθμο επιλογής στο στάδιο ένα, το δέντρο παραγωγής [Sipser 1996] μπορεί να διερευνηθεί είτε πρώτα κατά βάθος (depth-first) ή πρώτα κατά πλάτος (breadth-first). Ωστόσο, και οι δύο από αυτές τις έννοιες δεν επιτρέπουν επαρκή έλεγχο πάνω στη παραγωγή. Ως εκ τούτου, μπορούμε να αναθέσουμε μια προτεραιότητα σε όλους τους κανόνες, σύμφωνα με

τη λεπτομέρεια που αντιπροσωπεύεται από το σχήμα για να αποκτήσουμε μία (τροποποιημένη) παραγωγή πρώτα κατά πλάτος (breadth-first) : μπορούμε απλά να επιλέξουμε το σχήμα με τον κανόνα της υψηλότερης προτεραιότητας στο πρώτο βήμα. Η στρατηγική αυτή εγγυάται ότι η παραγωγή προχωρά από χαμηλή λεπτομέρεια σε υψηλή λεπτομέρεια με έναν ελεγχόμενο τρόπο. Παρακαλώ σημειώστε, ότι δεν μπορούμε να διαγράψουμε τα σχήματα, αλλά τα "σημειώνουμε" ως ανενεργά, αφού έχουν αντικατασταθεί. Αυτό μας δίνει τη δυνατότητα να θέσουμε υπό αμφισβήτηση την ιεραρχία σχήματος, και όχι μόνο την ενεργή ρύθμιση παραμέτρων.

Συμβολισμός: οι κανόνες παραγωγής ορίζονται στην παρακάτω φόρμα [Müller et al. 2006]:

id: predecessor : cond ~ successor : prob

όπου id είναι ένα μοναδικό αναγνωριστικό για τον κανόνα, ο προκάτοχος (predecessor) $\in V$ είναι ένα σύμβολο που προσδιορίζει ένα σχήμα που πρόκειται να αντικατασταθεί με το διάδοχο του, και η συνθήκη (condition ή cond) είναι ένας φρουρός (λογική έκφραση), που έχει να αξιολογηθεί ως αληθής για τον κανόνα που πρέπει να εφαρμοστεί . Ο κανόνας επιλέγεται με πιθανότητα prob. Για παράδειγμα, ο κανόνας

1: fac(h) : h > 9 ~ floor(h/3) floor(h/3) floor(h/3)

αντικαθιστά το σχήμα πρόσοψη (fac, facade) με τρία σχήματα ορόφου, αν η παράμετρος ύψος (h, height) είναι μεγαλύτερο από 9. Για να καθορίσουμε τα σχήματα διαδοχής (successor shapes) χρησιμοποιούμε διαφορετικές μορφές κανόνων που εξηγούνται σε αυτό το κεφάλαιο.

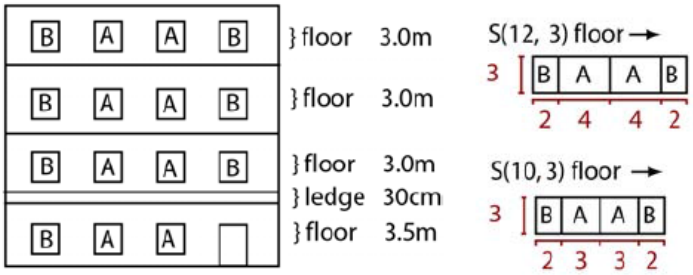
Κανόνες πεδίου εφαρμογής: Παρόμοια με τα L-systems χρησιμοποιούμε τους γενικούς κανόνες για την τροποποίηση σχημάτων [Müller et al. 2006]: *T (tx, ty, tz)* είναι ένα διάνυσμα μετακίνησης (translation vector) που προστίθεται στη θέση πεδίου εφαρμογής (scope) *P*, *Rx* (γωνία), *Ry* (γωνία), και της *Rz* (γωνία), περιστρέφει (rotate) τον αντίστοιχο άξονα του συστήματος συντεταγμένων, και *S (sx, sy, sz)* καθορίζει το μέγεθος (size) του πεδίου εφαρμογής. Χρησιμοποιούμε για να ωθήσουμε (push) και να αποσπάσουμε (pop) το τρέχον πεδίο εφαρμογής σε μια στοίβα. Κάθε μη τερματικό σύμβολο $\in V$ στον κανόνα θα δημιουργηθεί με το τρέχον πεδίο εφαρμογής. Ομοίως, η εντολή *I (objId)* προσθέτει ένα στιγμιότυπο ενός γεωμετρικού πρωτόγονου σχήματος (primitive) με το αναγνωριστικό *objId* . Τυπικά αντικείμενα περιλαμβάνουν έναν κύβο, ένα τετράπλευρο σχήμα, και ένα κύλινδρο, αλλά οποιοδήποτε τρισδιάστατο μοντέλο μπορεί να χρησιμοποιηθεί εξίσου. Το παράδειγμα που ακολουθεί απεικονίζει τη σχεδίαση του μαζικού μοντέλου που απεικονίζεται στην Εικόνα 34, δεξιά [Müller et al. 2006]:

1: A ~ [T(0,0,6) S(8,10,18) I("cube")]
T(6,0,0) S(7,13,18) I("cube") T(0,0,16) S(8,15,8) I("cylinder")

Βασικός κανόνας τμηματοποίησης: Ο βασικός κανόνας τμηματοποίησης χωρίζει το τρέχον πεδίο εφαρμογής κατά μήκος ενός άξονα [Müller et al. 2006]. Για παράδειγμα, σκεφτείτε τον κανόνα για να τμηματοποιήσουμε τη πρόσοψη της Εικόνας 35 αριστερά σε τέσσερις ορόφους και ένα περβάζι:

1: fac ~ Subdiv("Y",3.5,0.3,3,3,3){ floor | ledge | floor | floor | floor }

Η πρώτη παράμετρος περιγράφει τη τμηματοποίηση του άξονα ("X", "Y" ή "Z") και οι υπόλοιπες παράμετροι περιγράφουν τα μεγέθη τμηματοποίησης . Μεταξύ της οριοθέτησης { και } δίνεται μια λίστα σχημάτων, χωρίζοντάς τα με |. Χρησιμοποιούμε επίσης παρόμοιους κανόνες τμηματοποίησης για να χωρίσουμε κατά μήκος πολλαπλών αξόνων ("XY", "XZ", "YZ" ή "XYZ"), εμφωλευμένες τμηματοποιήσεις, ή εμφωλευμένους συνδυασμούς τμηματοποίησης και L -system κανόνων.



Εικόνα 35: Αριστερά: Ένας βασικός σχεδιασμός πρόσοψης. Δεξιά: Μια απλή τμηματοποίηση που θα μπορούσε να χρησιμοποιηθεί για τους ανώτερους τρεις ορόφους.

Κλιμάκωση κανόνων: Από το προηγούμενο παράδειγμα μπορούμε να δούμε την πρώτη πρόκληση. Η τμηματοποίηση είναι προσδιορισμένη για να λειτουργήσει καλά με ένα πεδίο εφαρμογής μεγέθους y = 12,8, αλλά και για άλλα πεδία εφαρμογής ο κανόνας πρέπει να κλιμακωθεί. Από την εμπειρία μας, δεν είναι όλα τα τμήματα αρχιτεκτονικής κλίμακας εξίσου καλά, και είναι σημαντικό να έχει τη δυνατότητα να γίνει διάκριση μεταξύ απόλυτων τιμών (τιμές που δεν κλιμακώνονται) και των σχετικών τιμών (τιμές που κλιμακώνονται) [Müller et al. 2006]. Οι τιμές θεωρούνται απόλυτες από προεπιλογή και θα χρησιμοποιήσουμε το γράμμα R για να δηλώσουμε σχετικές τιμές, π.χ.

1: floor ~ Subdiv("X",2,1r,1r,2){ B | A | A | B }

όπου οι σχετικές r_i τιμές αντικαθίστανται ως r_i * *(Scope.sx* - $\sum abs_i$*)* / r_i (Scope.sx

αντιπροσωπεύει το μέγεθος του x μήκους του τρέχοντος πεδίου εφαρμογής). Στην Εικόνα 35 παρουσιάζει σωστά την εφαρμογή του κανόνα πάνω σε δύο διαφορετικού μεγέθους ορόφους (με x-μήκους 12 και 10).

Επανάληψη: Για να καταστεί δυνατή μεγαλύτερης κλίμακας αλλαγές στους κανόνες τμηματοποίησης, συχνά θέλουμε να επιστρώσουμε ένα συγκεκριμένο στοιχείο [Müller et al. 2006]. Για παράδειγμα:

1: floor ~ Repeat(”X”,2){ B }

Το δάπεδο θα πρέπει να επενδυθεί σε όσα στοιχεία του τύπου Β κατά μήκος του άξονα x του πεδίου εφαρμογής, καθώς υπάρχει χώρος για αυτό. Ο αριθμός των επαναλήψεων υπολογίζεται ως *επαναλήψεις = (Scope.sx / 2)* και μπορούμε να προσαρμόσουμε το πραγματικό μέγεθος του στοιχείου ανάλογα.

Συστατικό τμηματοποίησης: Μέχρι αυτό το σημείο όλα τα σχήματα (πεδία εφαρμογής) υπήρξαν τρισδιάστατα. Η ακόλουθη εντολή μας επιτρέπει να χωρίσουμε σε σχήματα μικρότερων διαστάσεων [Müller et al. 2006]:

1: a ~ Comp(type, param){ A | B | ... | Z }

Όπου ο τύπος προσδιορίζει τον τύπο της τμηματοποίησης του συστατικού που σχετίζεται με *param* παραμέτρους (αν υπάρχουν). Για παράδειγμα, γράφουμε *Comp («face») {A}* για να δημιουργήσουμε ένα σχήμα με το σύμβολο A για κάθε πρόσοψη του αρχικού τρισδιάστατου σχήματος. Ομοίως χρησιμοποιούμε *Comp («edges») {B}* και *Comp ("vertices") {C}* για να χωριστεί σε άκρα και κορυφές αντίστοιχα. Για να αποκτήσουμε πρόσβαση μόνο σε επιλεγμένα συστατικά μπορούμε να χρησιμοποιήσουμε εντολές όπως η *Comp («edges», 3) {A}* για να δημιουργήσουμε ένα σχήμα A ευθυγραμμισμένο με το τρίτο άκρο του μοντέλου ή *Comp («side face ») {B}* για να αποκτήσουμε πρόσβαση στην πλευρά πρόσοψης ενός κύβου ή πολυγωνικού κυλίνδρου. Για να κωδικοποιήσουμε τα σχήματα μικρότερης διάστασης χρησιμοποιούμε πεδία εφαρμογής όπου ένας ή περισσότεροι άξονες έχουν μηδενικό μέγεθος. Για να επιστρέψουμε σε υψηλότερες διαστάσεις μπορούμε απλά να χρησιμοποιήσουμε την *S* εντολή μεγέθους με μη μηδενική τιμή στην αντίστοιχη διάσταση (π.χ. να εξωθηθεί ένα σχήμα της πρόσοψης κατά τα συνηθισμένα και ως εκ τούτου, το μετασχηματισμό του σε ογκομετρικό σχήμα).

4.2 Κανόνες Τμηματοποίησης στην Πράξη

Με βάση όσα ισχυρίστηκαν πιο πάνω, περνάμε στο πεδίο εφαρμογής των κανόνων που περιγράψαμε ως τώρα.

Η φιλοσοφία που ακολουθούν οι κανόνες τμηματοποίησης (split grammar rules) είναι στην πράξη –αλλά και στη θεωρία- απλή και ακολουθεί το ρητό *«διαίρει και βασίλευε».*

Αρχικά, για να μπορεί να απεικονιστεί έστω και ένα κτίριο σε ένα χώρο που επιθυμείται κάποιο επίπεδο ρεαλισμού, πρέπει να μελετηθεί η ιστορία του εν λόγω κτιρίου ώστε να αποδοθεί σε αυτό ο ανάλογος ρεαλισμός. Από έγκυρες πηγές της δημόσιας βιβλιοθήκης της Κέρκυρας, ανακαλύψαμε για τα παρουσιαζόμενα κτίρια μας τα εξής σημαντικά σημεία που αξίζει να αναφέρουμε εδώ.

Από ιστορικές αναφορές παίρνουμε στοιχεία για τη μορφολογία της παλαιάς πόλης της Κέρκυρας: *«Ο τύπος των κατοικιών έχει καθοριστεί από τη στενότητα του χώρου και την πυκνότητα του πληθυσμού», «Απογραφή του 1800: 2006 σπίτια στην παλαιά πόλη», «Απογραφή του 1940: 1700 σπίτια με μέση κάλυψη 84τμ., ελάχιστη κάλυψη 30τμ.», «οι ελεύθεροι χώροι ελάχιστοι», «απουσία αυλών και κήπων», «οι δρόμοι στενοί 3μ», «απουσία υπογείων», «τοξοστοιχίες στοών», «Το 19ο αιώνα υπήρξαν κτιριακές αλλαγές και αύξηση ύψους των κτιρίων», «μέχρι και το 1864 γινόντουσαν κατασκευαστικές εργασίες στο ναό του Αγίου Σπυρίδωνα».*

Όλες αυτές οι αναφορές ήταν ο προάγγελος για την πιστή αναπαράσταση των κτιρίων στην παλαιά πόλη της Κέρκυρας. Από το πρώτο κτίριο που θα κατασκευαζόταν μέχρι και το σύνολό τους ως συγκρότημα κτιρίων.

Στη συνέχεια βλέπουμε ένα από τα σπίτια που βασίστηκαν στις παραπάνω ιστορικές περιγραφές (Εικόνα 36).

Εικόνα 36: Διώροφο παραδοσιακό σπίτι της παλαιάς πόλης της Κέρκυρας. Αριστερά: 3D μοντέλο κατασκευασμένο στο Unity3D. Δεξιά: Το συντηρημένο κτίριο που αποδόθηκε με το Unity3D.

Σαν επόμενο βήμα μας ήταν να πάρουμε ένα πραγματικό αντικείμενο του φυσικού περιβάλλοντος της πόλης, για παράδειγμα μια φωτογραφία ενός κτιρίου που διατηρείται μέχρι και σήμερα από τον 18ο ή 19ο αιώνα (Εικόνα 36, δεξιά), και να του αποδώσουμε ζωή στον γραφικό κόσμο υπολογιστών μέσω εργαλείων 3D απεικόνισης που μας προσέφερε το Unity3D (Εικόνα 36, αριστερά).

Ο αλγόριθμος που ακολούθησε με βάση της φιλοσοφίας «διαίρει και βασίλευε» ή διαφορετικά κανόνες τμηματοποίησης ήταν οι εξής: Κατασκευή των βάσεων και

των στύλων του ισογείου με επανάληψη. Έπειτα κατασκευάζεται το ισόγειο τμήμα (κύβος) και οι λοιποί όροφοι (με σειρά 1ος , 2ος) με επανάληψη. Στη συνέχεια, δημιουργούμε τις πόρτες και τα παράθυρα με επαναληπτική μέθοδο και οι στύλοι του 1ου ορόφου. Τέλος, η οροφή. Αφού υπολογίσουμε το ύψος των στύλων του ισογείου και τη βάση του πρώτου ορόφου, τότε είμαστε σε θέση να τοποθετήσουμε και τα τόξα με επαναληπτική μέθοδο κατά μήκος του κτιρίου. Η ίδια μέθοδος ακολουθείται και για τα τόξα του 2ου ορόφου. Στο σημείο αυτό να πούμε πως το κτίριο αποτελεί το πρωτεύον αντικείμενο. Τα επιμέρους στοιχεία όπως στύλοι, τόξα, πόρτες και παράθυρα αποτελούν τα «παιδιά» του πρωταρχικού μας μοντέλου ενώ, για παράδειγμα, το πάνω μέρος του στύλου αποτελεί το «παιδί» του κυρίους σώματος του στύλου το οποίο με τη σειρά του αποτελεί το «παιδί» του κτιρίου, δηλαδή του πρωταρχικού μας μοντέλου. Έτσι, ανάμεσα σε κάθε αντικείμενα και κάθε επιμέρους αντικείμενα των αντικειμένων δημιουργείται μια σχέση «πατέρα-παιδιού». Ο Πίνακας 2 και η Εικόνα 37, παρουσιάζουν τον παραπάνω αλγόριθμο.

Εικόνα 37: Το "σπάσιμο" του 3D μοντέλου της εικόνας 37.

Πίνακας 2: Η γραμματική τμηματοποίησης του κτιρίου της Εικόνας 37.

Start Symbol =	***House_3***
House_3 ->	Arch
	vert
	Arch second floor
	Arch second floor vert
	Attic
	Balcony front wall
	Repeated Balcony side wall
	Repeated Column
	Repeated Column second floor

	Base left
	Base middle
	Base right
	Repeated Eaves 1st floor
	Repeated Eaves 2nd floor
	Repeated Doors
	Repeated Windows
	Repeated Stairs
	Roof
Column ->	Column base
	Column top
Column second floor ->	Column base
	Column top

Σχήμα: Το κύριο μοντέλο του κτιρίου (House_3) βασίστηκε σε ένα αντικείμενο Κύβου που αργότερα όλα τα «παιδιά» ή τα επιμέρους στοιχεία του «συγκολλήθηκαν» επάνω του. Μέσω της γλώσσας javascript δίνουμε τις απαραίτητες συντεταγμένες για να οριστεί η θέση του αντικειμένου στο χώρο και το μέγεθος του:

oBuilding3 = GameObject.CreatePrimitive(PrimitiveType.Cube);

oBuilding3.name = "House_3";

oBuilding3.transform.position = Vector3(this.RandX, 3, this.RandZ);

oBuilding3.transform.localScale.x = 7;

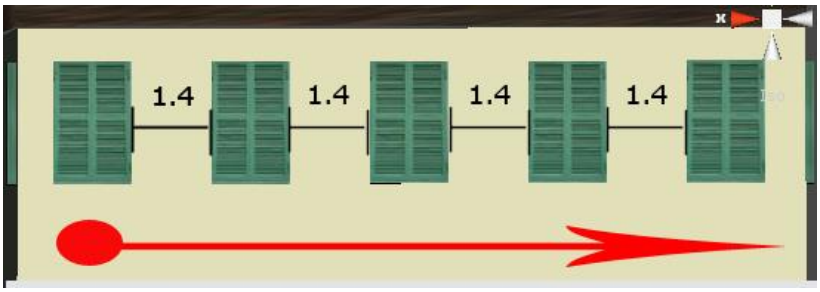
oBuilding3.transform.localScale.y = 4.5;

oBuilding3.transform.localScale.z = 5;

Αρχικά δημιουργούμε ένα αντικείμενο *oBuilding3* με τύπο μορφής Κύβου και με όνομα αντικειμένου House_3. Ύστερα του αποδίδουμε τη θέση στο χώρο χρησιμοποιώντας τις συντεταγμένες X, Y, Z καλώντας τη μέθοδο Vector3(). Οι μεταβλητές που αντιστοιχούν στις συντεταγμένες X και Z θέτουν το αντικείμενο τυχαία στο χώρο σε αυτές τις συντεταγμένες και μόνο. Αυτό εξυπηρετεί αργότερα την απεικόνιση γενικότερα των κτιρίων στο χώρο σε τυχαίες θέσεις. Μετά θέτουμε το μέγεθος του αντικειμένου με τις συγκεκριμένες τιμές. Οι τιμές αυτές είναι τύπου localScale, σύμφωνα με το Unity3D, διότι αφορά τις τοπικές τιμές μεγέθους του αντικειμένου σε σχέση με το χώρο απεικόνισης του.

Παραγωγική διαδικασία: Εδώ σε αυτό το στάδιο, και συγκεκριμένα στην περίπτωση των παραθύρων του 2ου ορόφου της Εικόνας 37, εφαρμόζεται η παραγωγική διαδικασία κατασκευής αντιγράφων του σχήματος του παραθύρου, που όπως αναφέρθηκε στον Πίνακα 2 αποτελεί και «παιδί» του βασικού σχήματος

του κτιρίου *House_3*. Έτσι ξεκινώντας από το δεξί άκρο του ορόφου προς τα αριστερά (ή και αντίστροφα) δημιουργούνται τα επιθυμητά αντίγραφα του παραθύρου όπως φαίνεται στην Εικόνα 38.



Εικόνα 38: Διαδικαστική παραγωγή παραθύρων από τα δεξιά προς τα αριστερά. Η απόσταση μεταξύ των παραγόμενων στοιχείων είναι ίσα.

Συμβολισμός και κανόνες πεδίου εφαρμογής::

```
for(i=0; i<=3; i++){

if(oWindow){

var cloneWind_attic = Instantiate(oWindow, Vector3(nextPOSX_wind,nWind_Y, nWind_Z), Quaternion.identity);

cloneWind_attic.transform.parent = oBuilding3.transform; //parenting

cloneWind_attic.name = "Window attic " + (++c);

cloneWind_attic.transform.localScale.x =0.1;

cloneWind_attic.transform.localScale.y = 0.2402316;

cloneWind_attic.transform.localScale.z = 0.0163088;

aCollectionObj_3.Add(cloneWind_attic);

nextPOSX_wind -= 1.4;

}//end if

}//end for
```

Από τον παραπάνω κώδικα javascript θα σταθούμε σε μόνο μερικά σημεία. Οι κανόνες πεδίου εφαρμογής αποτελεί η συνθήκη *if* που ελέγχει αν το αντικείμενο που κατασκευάζουμε είναι τύπου παραθύρου. Αν είναι αληθής, τότε οι κανόνες μετακίνησης (*Vector3(nextPOSX_wind,nWind_Y,nWind_Z)*) και μεγέθους (*localScale.x, localScale.y, localScale.z*) εφαρμόζουν τις ανάλογες ρυθμίσεις τιμών για τα αντίγραφα αντικείμενα παραθύρου που δόθηκαν από τον κώδικα. Άλλος κανόνας που εφαρμόζεται είναι η σχέση μεταξύ των αντικειμένων. Για κάθε

αντίγραφο παραθύρου που δημιουργείται «συγκολλείται» κάτω από το αντικείμενο «πατέρα» όλων, το *oBuilding3.*

Κλιμάκωση κανόνων: *oWindow ~ cloneWind_attic ("X",0.1, 0.1, 0.1, 0.1){W1 | W2 | W3 | W4}.* Όπου η τιμή 0.1 αποτελεί το μήκος του κάθε παραθύρου-αντίγραφου και το W1 αποτελεί το παράθυρο-αντίγραφο 1, W2 αποτελεί το παράθυρο-αντίγραφο 2 κ.ο.κ.

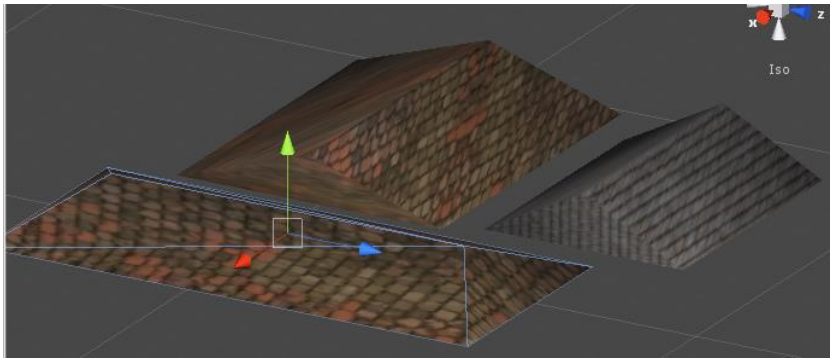
Επανάληψη: Η επανάληψη αφορά το πεδίο ορισμού της *for{}.* Συγκεκριμένος αριθμός αντιγράφων των παραθύρων επαναλαμβάνονται κατά μήκος του άξονα X, από τα δεξιά προς τα αριστερά, ανά ίση απόσταση αναπαραγωγής (Εικόνα 39).

4.2.1 Σκεπές

Οι σκεπές αποτελούν διαφορετική περίπτωση από τη δημιουργία κτιρίων. Αποτελεί μια διαφορετική ενότητα αλλά συνάμα είναι μέρος των κανόνων τμηματοποίησης που συνθέτουν ένα κτίριο. Η σκεπή, όπως και τα τόξα στη συνέχεια αποτελούν ένα σύνολο από πλέγματα (meshes) για την εφαρμογή Unity3D.

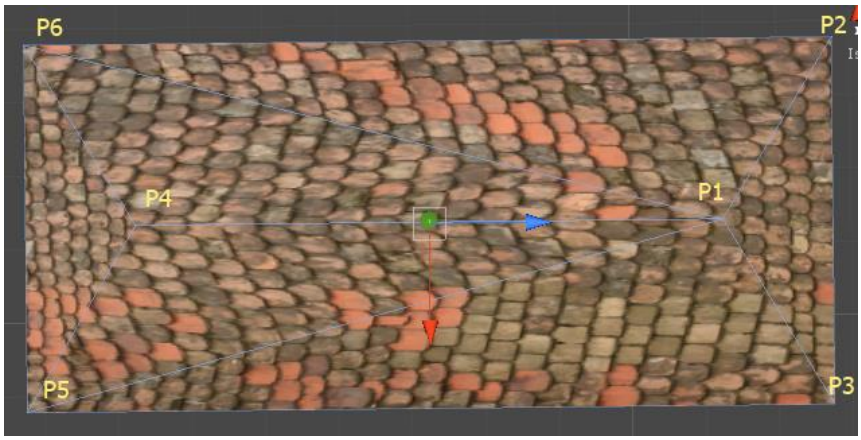
Ενώ εξωτερικά δεν διαφέρουν από τα πρωτόγονα αντικείμενα, πλέγματα αποθηκεύονται αρκετά διαφορετικά. Αντί να αποθηκεύονται ως μια λίστα παραμέτρων που χρησιμοποιούνται για τη δημιουργία ενός 3D αντικειμένου, τα πλέγματα αποθηκεύονται ως μια συλλογή από σημεία στο χώρο. Ως εκ τούτου, μπορούν να αναλάβουν μια ατελείωτη ποικιλία σχημάτων, αλλά τείνουν να καταλαμβάνουν πολύ χώρο στο σκληρό δίσκο. Στη μνήμη, και οι δύο τύποι αντικειμένων καταλαμβάνουν την ίδια ποσότητα χώρου.

Στο πρακτικό κομμάτι της απεικόνισης των κτιρίων συναντάμε ένα βασικό είδος για κάθε κτίριο, εκείνο των έξι άκρων (Εικόνα 39).



Εικόνα 39: Οροφές με δύο κορυφές.

Η βάση της οροφής είναι ουσιαστικά μια επέκταση του περιβάλλοντος ενός κιβωτίου του κάθε κτιρίου και αποτελείται από οκτώ σημεία. Τα τέσσερα κάτω σημεία είναι τα ίδια με τα υψηλότερα σημεία του περιβάλλοντος κιβωτίου του κτιρίου. Τα τέσσερα πάνω σημεία προκύπτουν με την πρόσθεση ενός σταθερού αριθμού στην συνιστώσα *y* στο καθένα από τα κάτω σημεία. Για να υπολογιστεί και στη συνέχεια να απεικονιστεί σωστά μια σκεπή, υπολογίζουμε τα σημεία μας με τη φορά του ρολογιού κοιτάζοντας τη βάση οροφής από πάνω. Υπόψη ότι υπολογίζουμε αρχικά το διάνυσμα κατεύθυνσης της διχοτόμου της κάθε εσωτερικής γωνίας του τετράπλευρου που σχηματίζουν τα τέσσερα αρχικά σημεία της οροφής (Εικόνα 40).



Εικόνα 40: Στο σχήμα της οροφής έχουμε 6 άκρα: 4 γωνίες με μικρότερο συντελεστή y (σημεία P2,P3,P5,P6) και 2 γωνίες με μεγαλύτερο συντελεστή y (σημεία P1,P4). Ο υπολογισμός τους γίνεται με τη φορά του ρολογιού.

*var tri: int[] = new int[18]; //3 vertices * 6 triangles =18*

//vertices positioning

verts[0] = new Vector3(this.RandX - 3.5, 8.5, this.RandZ + 0.5); //P1

verts[1] = new Vector3(this.RandX - 3.5, 7.5, this.RandZ - 2.6); //P2

verts[2] = new Vector3(this.RandX - 3.5, 7.5, this.RandZ + 4); //P3

verts[3] = new Vector3(this.RandX + 3.5, 8.5, this.RandZ + 0.5); //P4

verts[4] = new Vector3(this.RandX + 3.5, 7.5, this.RandZ + 4); //P5

verts[5] = new Vector3(this.RandX + 3.5, 7.5, this.RandZ - 2.6); //P6

Ο πάνω κώδικας σε javascript υπολογίζει τα τρίγωνα που προκύπτουν από το σύνολο των άκρων που περιγράφονται από τις τρεις συντεταγμένες του χώρου μας. Για να κατασκευάσουμε τα άκρα των τριγώνων, υπολογίζουμε τα έξι τρίγωνα που προκύπτουν από το σχήμα επί τα τρία σημεία τριγώνου (*var tri: int[] = new int[18]; //3 vertices * 6 triangles =18*). Τα άκρα P1 και P4 έχουν Y=8.5 που σημαίνει ότι αφορά τα δύο υπερυψωμένα σημεία της σκεπής.

//triangles

tri[0] = 0; //refer to vertices: verts[0]

tri[1] = 1; //refer to vertices: verts[1]

tri[2] = 2; //refer to vertices: verts[2]

tri[3] = 0;

tri[4] = 2;

tri[5] = 4;

tri[6] = 3;

tri[7] = 0;

tri[8] = 4;

tri[9] = 3;

tri[10] = 4;

tri[11] = 5;

tri[12] = 3;

tri[13] = 5;

tri[14] = 0;

tri[15] = 0;

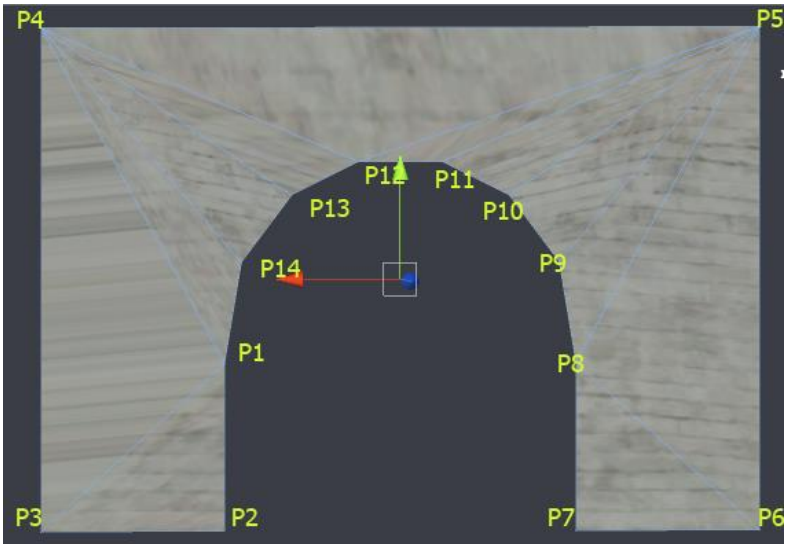
tri[16] = 5;

tri[17] = 1;

Για κάθε ένα σύνολο σημείων verts[i], δημιουργούν και ένα τρίγωνο. Έτσι για την κατασκευή μιας σκεπής έξι σημείων χρειαζόμαστε απαραίτητα το σχηματισμό δεκαοκτώ τριγώνων.

4.2.2 Τόξα

Ένα άλλο κομμάτι αρκετά μεγάλης δυσκολίας ήταν η απεικόνιση των τόξων που απεικονίζονται στις προσόψεις των κτιρίων της παλαιάς Κέρκυρας. Όπως είπαμε και πιο πριν, τόσο τα τόξα όσο και οι σκεπές κατασκευάζονται από σύνολα πλεγμάτων (meshes) στο Unity3D για να πάρουν μια πραγματική υπόσταση στο τρισδιάστατο χώρο (Εικόνα 41). Ο τρόπος εφαρμογής των τόξων δε διαφέρει καθόλου από εκείνης της σκεπής.



Εικόνα 41: Απεικόνιση τόξου από ένα σύνολο 12 τριγώνων και 14 σημείων (P1...P14).

Όπως κι οι σκεπές έτσι και τα τόξα αποτελούνται από έναν αριθμό τριγωνικών πλεγμάτων. Όσο πιο πολλά τα πλεγματικά τρίγωνα τόση λεπτομέρεια προκύπτει στο μοντέλο του τόξου.

var verts: Vector3[] = new Vector3[14];

*var tri: int[] = new int[72]; //3 vertices * 12 triangles = 36 * 2 (duplicate triangles for the back face) =72*

Σε συνδυασμό της Εικόνας 41 με τον κώδικα στο πάνω μέρος της παραγράφου, βλέπουμε πως πρέπει να κατασκευάσουμε δεκατέσσερα σημεία. Αυτά τα σημεία (P1…P14) αντιστοιχούνται σε ένα πίνακα τύπου *Vector3[],* αποτελώντας τις γωνίες σε ένα πίνακα συντεταγμένων x,y,z. Αργότερα, σχηματίζουμε έναν πίνακα τριγώνων μεγέθους 72. Αυτό γιατί θέλουμε να αναπαραστήσουμε 12 τρίγωνα που συνθέτουν το μοντέλο πλέγματος του τόξου μας επί των τριών συντεταγμένων του χώρου μας επί 2 γιατί θέλουμε το πλέγμα του τόξου μας να φαίνεται και από την πίσω πλευρά του κτιρίου μας (αν επιθυμούμε κάτι τέτοιο φυσικά) που πρακτικά αυτό σημαίνει ότι θέλουμε την αναπαράσταση άλλων 12 τριγώνων επί των τριών συντεταγμένων του χώρου.

Για κάθε αναπαράσταση τριγώνων έχουμε τα εξής:

//vertices positioning

verts[0] = new Vector3(this.RandX - 1.9, 2.5, this.RandZ + 3.777); //P1

verts[1] = new Vector3(this.RandX - 1.9, 2.35, this.RandZ + 3.777); //P2

verts[2] = new Vector3(this.RandX - 1.72, 2.35, this.RandZ + 3.777); //P3

verts[3] = new Vector3(this.RandX - 1.72, 3.3, this.RandZ + 3.777); //P4

verts[4] = new Vector3(this.RandX - 3.2, 3.3, this.RandZ + 3.777); //P5

verts[5] = new Vector3(this.RandX - 3.2, 2.35, this.RandZ + 3.777); //P6

verts[6] = new Vector3(this.RandX - 3.02, 2.35, this.RandZ + 3.777); //P7

verts[7] = new Vector3(this.RandX - 3.02, 2.5, this.RandZ + 3.777); //P8

verts[8] = new Vector3(this.RandX - 2.96, 2.7, this.RandZ + 3.777); //P9

verts[9] = new Vector3(this.RandX - 2.83, 2.9, this.RandZ + 3.777); //P10

verts[10] = new Vector3(this.RandX - 2.62, 3, this.RandZ + 3.777); //P11

verts[11] = new Vector3(this.RandX - 2.3, 3, this.RandZ + 3.777); //P12

verts[12] = new Vector3(this.RandX - 2.09, 2.9, this.RandZ + 3.777); //P13

verts[13] = new Vector3(this.RandX - 1.96, 2.7, this.RandZ + 3.777); //P14

Αν προσέξουμε στο πεδίο z της μεθόδου *Vector3()* έχουμε την ίδια τιμή κι αυτό γιατί η κατασκευή του πλέγματος των τριγώνων έχουν το ίδιο βάθος. Ο υπολογισμός των τριγώνων γίνεται με τον ίδιο ακριβώς τρόπο που έγινε και με την δημιουργία της σκεπής, δηλαδή με τη φορά του ρολογιού (βλέπε Εικόνα 41 και πάνω κώδικα).

//triangles

tri[0] = 0; //refer to vertices: verts[0]

tri[1] = 1; //refer to vertices: verts[1]

tri[2] = 2; //refer to vertices: verts[2]

tri[3] = 0;

tri[4] = 2;

tri[5] = 3;

……

tri[69] = 7;

tri[70] = 6;

tri[71] = 5;

Παρομοίως και εδώ, για κάθε ένα σύνολο σημείων verts[i], κατασκευάζουμε και ένα τρίγωνο. Έτσι για την κατασκευή ενός τόξου δεκατεσσάρων σημείων χρειαζόμαστε απαραίτητα το σχηματισμό εβδομήντα δύο τριγώνων (36 τριγώνων μπροστά και 36 τριγώνων πίσω).

4.2.3 Φινίρισμα

Όπως και στον φυσικό κόσμο, έτσι και στον τρισδιάστατο χώρο, για να έχουμε μια ρεαλιστική απεικόνιση των σχεδιασμένων 3D μοντέλων των οικοδομημάτων μας πρέπει να περάσουν από ένα τελικό φινίρισμα. Μια διαδικασία μέσα από μεθόδους που μας παρέχει το εργαλείο τρισδιάστατης απεικόνισης μοντέλων Unity3D για να αντικατοπτρίζουν τα αντικείμενά μας το φυσικό μας κόσμο.

Παρακάτω παρουσιάζουμε κάποιες τεχνικές χρώματος και υφής που ακολουθήθηκαν για να φινίρουμε τα μοντέλα των κτιρίων της παλαιάς Κέρκυρας, όπως πρωτοεμφανίστηκαν το 18ο και 19ο αιώνα.

4.2.3.1 Απόδοση Χρώματος

Καθώς τα βασικά χρώματα των κτιρίων της παλαιάς πόλης που κατασκευάστηκαν στην Κέρκυρα μεταξύ 18ου-19ου αιώνα αποτελούσαν της οικογένειας χρωμάτων του κίτρινου, του μπεζ, του κόκκινου και του λευκού (κατά πλειονότητα) με τις κεραμιδί οροφές, κρίθηκε λοιπόν εύλογο- και συνάμα εύκολο- να χρησιμοποιηθούν έτοιμοι μέθοδοι χρωμάτων από την εφαρμογή Unity3D. Αυτή και μόνο η πρακτική μπορούσε να καλύψει πλήρως τις απαιτήσεις για το χρωματισμό όλων των οικιών.

Για παράδειγμα η μέθοδος χρώματος που χρησιμοποιήσαμε σε ένα από τα σπίτια ήταν του τύπου*: oBase.renderer.material.color = Color(1, 0.980, 0.804, 0);*

Εδώ φαίνεται η απόδοση χρώματος στη βάση ενός κτιρίου. Η μέθοδος *Color()* εδώ καθορίζεται με τέσσερις τιμές τύπου: r (red), g (green), b (blue), a (transparency). Και οι τέσσερις τιμές του κόκκινου, πράσινου, μπλέ υπόβαθρων και υπόβαθρο διαφάνειας παίρνουν τιμές από 0 έως 1 (στο σύστημα τιμών κινητής υποδιαστολής, floating point value), όπως με άλλο τρόπο θα λέγαμε από 0 έως 255 στο δεκαδικό σύστημα. Για παράδειγμα η τιμή του υποβάθρου του πράσινου στο δεύτερο πεδίο της μεθόδου χρώματος, μας υποδηλώνει την ένταση χρώματος του πράσινου με τιμή 0.980 ή, προσεγγιστικά στο δεκαδικό σύστημα, ίσο με 250. Η τιμή της διαφάνειας ορίζεται επίσης από το 0 έως το 1 και καθορίζει την τιμή διαφάνειας χρώματος που θα έχει το τρισδιάστατο μοντέλο μας. Αν έχουμε τιμή a=0 τότε έχουμε διαφάνεια, στην αντίθετη περίπτωση με a=1 έχουμε εντελώς αδιαφανή απεικόνιση χρώματος.

4.2.3.2 Απόδοση Υφής

Ενώ η γέμιση χρώματος είναι μια πιο απλή μέθοδος φινιρίσματος, η απόδοση υφής σε ένα μοντέλο χώρου είναι μια πιο περίπλοκη διαδικασία.

Αντικείμενα όπως ένα παράθυρο, μια πόρτα, μια σκεπή ήταν αδύνατο να αποδοθούν με ένα απλό γέμισμα χρώματος, καθώς δε θα παρουσίαζαν καμιά

ρεαλιστική υπόσταση στο περιβάλλον. Υπόψη μας πως όσο πιο ρεαλιστικά απεικονιστούν τα μοντέλα των οικοδομημάτων μας τόσο δίνουμε μια πιο φυσική αίσθηση στο χώρο που παρατηρεί και αλληλεπιδρά ο χρήστης. Για την επίτευξη μιας πιο φυσικής εικόνας «ντύσαμε» τα επιμέρους αντικείμενα των 3D μοντέλων μας με τις κατάλληλες υφές (Εικόνα 42).

Εικόνα 42: Φωτογραφίες από παράθυρα της Κέρκυρας που χρησιμοποιήθηκαν για το project της Πτυχιακής εργασίας.

Ένας υψηλός βαθμός λεπτομέρειας σκηνής και πολυπλοκότητας μπορεί να επιτευχθεί μέσω της χρήσης των λεπτομερών υφών των κτιρίων [Parish Y. I. H. and Müller P., 2001].

Αναφορικά, ο Catmull [Catmull, E. E., 1974] δημιούργησε τις πρώτες υφές εικόνων γραφικών του υπολογιστή. Οι επιφάνειες του Catmull εκπροσωπήθηκαν ως παραμετρικές κηλίδες. Κάθε σημείο της 3D επιφάνεια μιας παραμετρικής κηλίδας αντιστοιχεί σε ένα συγκεκριμένο σημείο 2D (u,v) στο χώρο των παραμέτρων. Αυτή η 2D-3D αντιστοιχία συνεπάγεται ότι μια δισδιάστατη εικόνα υφής μπορεί εύκολα να χαρτογραφηθεί πάνω σε μια επιφάνεια 3D. Οι (u,v) παράμετροι οποιουδήποτε σημείου της κηλίδας μπορεί να χρησιμοποιηθεί για να υπολογίσει μια αντίστοιχη θέση εικονοστοιχείου στην εικόνα υφής.

Ένας χάρτης υφής (texture mapping) εφαρμόζεται (αντιστοιχίζεται) προς την επιφάνεια ενός σχήματος ή ενός πολυγώνου [Jon Radoff, 2007]. Αυτή η διαδικασία είναι παρόμοια με την εφαρμογή ενός μοτίβου χαρτιού σε ένα απλό λευκό κουτί . Κάθε κορυφή σε ένα πολύγωνο αποδίδει συντεταγμένες υφής (η οποία στην περίπτωση των δισδιάστατων μοντέλων είναι επίσης γνωστό ως μια συντεταγμένη UV) είτε μέσω σαφής εκχώρησης ή με διαδικαστικό ορισμό. Οι θέσεις δειγματοληψίας εικόνας, στη συνέχεια παρεμβάλλονται κατά μήκος της πρόσοψης ενός πολυγώνου για να παράγει ένα οπτικό αποτέλεσμα που φαίνεται να παρουσιάζει περισσότερο πλούτο από ό, τι θα μπορούσε διαφορετικά να επιτευχθεί με ένα περιορισμένο αριθμό πολυγώνων.

Για να επιτύχουμε κάτι τέτοιο στο Unity3D χρειάζεται μια απλή κλήση της μεθόδου Load() από την κλάση Resources όπως φαίνεται παρακάτω στον κώδικα:

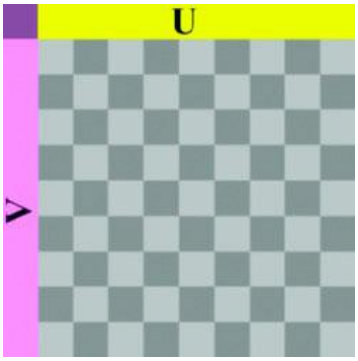
oWindow.renderer.material.mainTexture = Resources.Load ("window_corfu2");

Στο συγκεκριμένο παράδειγμα φορτώνουμε μια εικόνα μορφής jpeg από το φάκελο Resources του έργου μας και το «συγκολλάμε» πάνω στο αντικείμενο του παραθύρου μας. Το Unity3D είναι υπεύθυνο για την εκτέλεση των διαδικασιών του χάρτη υφής. Στην Εικόνα 43 βλέπουμε ένα τέτοιο παράδειγμα.

Εικόνα 43: Αριστερά: ένα πρότυπο κτιριακό μοντέλο της Κέρκυρας. Μέση: η μορφή του παραθύρου που παρουσιάζεται στο φυσικό μοντέλο αριστερά. Δεξιά: Το παραγόμενο 3D μοντέλο που διαθέτει το χάρτη υφής του παραθύρου στη μέση, αποκτώντας ρεαλισμό.

Παρόλα αυτά όταν ερχόμαστε αντιμέτωποι με πιο ιδιαίτερα και πολυπλοκότερα πλέγματα (meshes) που με τη σειρά τους χρειάζονται να τα ντύσουμε με υφές, όπως οι σκεπές και τα τόξα, ερχόμαστε αντιμέτωποι κανόνες που πρέπει εμείς οι ίδιοι να τους προσαρμόσουμε στα γούστα μας για το καλύτερο δυνατό αποτέλεσμα. Εφαρμόζοντας δισδιάστατους χάρτες υφής (u,v) σε τρισδιάστατα επίπεδα μοντέλα είναι κάτι εύκολο και δεν παρουσίασε καμιά πολυπλοκότητα στα απλά σχήματα της πόρτας και των παραθύρων των οικοδομημάτων μας. Ωστόσο, μοντέλα όπως οι σκεπές που αποτελούν ένα σύνολο πλεγμάτων με διαφορετική θέση και μέγεθος στο χώρο απαιτούν προσαρμογή από πλευράς χρήστη.

Σφαιρικά, κάθε φορά που ένα αντικείμενο χρησιμοποιεί μια υφή ή εικόνα ως μέρος του υλικού του, χρειάζεται συντεταγμένες χαρτογράφησης να πει στη μηχανή φωταπόδοσης πώς να εφαρμόσει την εικόνα στη πρόσοψη του αντικειμένου. Στην απλούστερη μορφή του, η χαρτογράφηση (mapping) μπορεί να είναι τόσο απλή όσο μια επίπεδη προβολή ή τόσο περίπλοκη όπως πολλαπλά UV "ξετυλίγματα"[Sue Blackman, 2011]. Οι συντεταγμένες χαρτογράφησης αναφέρονται ως U, V και W, όπου το U παριστά μία πλευρά του χάρτη και το V παριστά το άλλο άκρο, όπως φαίνεται στην Εικόνα 44. W είναι ο άξονας μιας κανονικής πρόσοψης και είναι σε χρήση, όταν περιστρέφεται ο χάρτης σχετικά με τον εν λόγω άξονα.



Εικόνα 44: UVW συντεταγμένες.

Η χαρτογράφηση UV προβλέπει ένα χάρτη υφής σε ένα αντικείμενο 3D. Τα γράμματα "U" και "V" υποδηλώνουν τους άξονες της 2D υφής (όταν χρησιμοποιούμε τετραδικό σύστημα Χάμιλτον (το οποίο είναι πρότυπο), το W χρησιμοποιείται επίσης) επειδή "X", "Y" και "Z" χρησιμοποιούνται ήδη για να υποδηλώσουν τους άξονες του 3D αντικειμένου στο χώρο μοντέλου.

Η UV υφή επιτρέπει πολύγωνα που συνθέτουν ένα 3D αντικείμενο να είναι βαμμένο με χρώμα από μια εικόνα. Η εικόνα ονομάζεται UV χάρτης υφής, [Mullen, T., 2009] αλλά είναι απλά μια συνηθισμένη εικόνα. Η UV διαδικασία χαρτογράφησης περιλαμβάνει την ανάθεση των pixels στην εικόνα για να επιστρώνονται στην επιφάνεια του πολυγώνου, που συνήθως γίνεται από "προγραμματιστική" αντιγραφή ενός διαμορφωμένο κομμάτι τριγώνου του χάρτη εικόνας και επικόλληση του σε ένα τρίγωνο πάνω στο αντικείμενο. [Murdock, K.L.,2008] Το UV είναι η εναλλακτική λύση XY, που μόνο αντιστοιχίζει σε ένα χώρο υφής και όχι μέσα στο γεωμετρικό χώρο του αντικειμένου. Ωστόσο, ο υπολογισμός απόδοσης χρησιμοποιεί UV συντεταγμένες για να καθορίσει πώς να ζωγραφίσει την τρισδιάστατη επιφάνεια.

Όταν ένα μοντέλο που έχει δημιουργηθεί ως ένα πολύγωνο πλέγμα χρησιμοποιώντας ένα 3D μοντελοποιητή, οι uv συντεταγμένες μπορούν να παραχθούν για κάθε κορυφή στο πλέγμα. Ένας τρόπος είναι για τον 3D μοντελοποιητή για να ξεδιπλώσει το ξετυλιγμένο πλέγμα τριγώνου στις ραφές, αυτόματα τακτοποιεί τα τρίγωνα σε μια επίπεδη σελίδα. Εάν το πλέγμα είναι μια UV σφαίρα, για παράδειγμα, μόλις το μοντέλο είναι ξετυλιγμένο , ο καλλιτέχνης μπορεί να ζωγραφίσει μια υφή σε κάθε τρίγωνο ξεχωριστά, χρησιμοποιώντας το ξετυλιγμένο πλέγμα ως πρότυπο. Όταν η σκηνή καθίσταται, κάθε τριγώνου θα χαρτογραφήθεί με την κατάλληλη υφή από το "αυτοκόλλητο φύλλο».

Ένας UV χάρτης μπορεί είτε να παράγεται αυτόματα από την εφαρμογή λογισμικού, είτε χειροκίνητα από τον καλλιτέχνη, ή κάποιο συνδυασμό και των δύο. Συχνά ένας uv χάρτης που θα δημιουργηθεί, στη συνέχεια ο καλλιτέχνης θα το προσαρμόσει και θα το βελτιστοποιήσει για την ελαχιστοποίηση ραφών και επικαλύψεων. Εάν το μοντέλο είναι συμμετρικό, ο καλλιτέχνης μπορεί να

επικαλύπτει αντίθετα τρίγωνα για να επιτρέψει το χρωματισμό και τις δύο πλευρές ταυτόχρονα.

Οι UV συντεταγμένες εφαρμόζονται ανά πρόσοψη, [Murdock, K.L.,2008], όχι ανά κορυφή. Αυτό σημαίνει ότι μια κοινή κορυφή μπορεί να έχει διαφορετικές συντεταγμένες uv σε κάθε ένα από τα τρίγωνα του, έτσι γειτονικά τρίγωνα μπορούν να κοπούν, και να τοποθετηθούν σε διαφορετικές περιοχές του χάρτη υφής (κάτι που συμβαίνει στην περίπτωση των σκεπών και των τόξων).

Η UV διαδικασία χαρτογράφησης στην απλούστερη μορφή της, απαιτεί τρία βήματα: ξετύλιγμα του πλέγματος, δημιουργώντας την υφή, εφαρμόζοντας την υφή [Mullen, T., 2009].

Στον κώδικα που ακολουθεί εξετάζουμε την περίπτωση της σκεπής ενός κτιρίου:

var uv: Vector2[] = new Vector2[6];

uv[0] = new Vector2(0,0.5);

uv[1] = new Vector2(1,1);

uv[2] = new Vector2(1,0);

uv[3] = new Vector2(1,0.5);

uv[4] = new Vector2(0,0);

uv[5] = new Vector2(0,1);

Αρχικά δηλώνουμε πως το σχήμα UV αποτελεί ένα πίνακα που καταχωρεί τις έξι προσόψεις τριγώνου που σχηματίζουν το αντικείμενο της σκεπής μας (Εικόνα 40). Στη συνέχεια, για κάθε πρόσοψη τριγώνου UV θέτουμε και τις ανάλογες τιμές.

Κάτι ανάλογο κάνουμε και για τα τόξα των κτιρίων μας (Εικόνα 41).

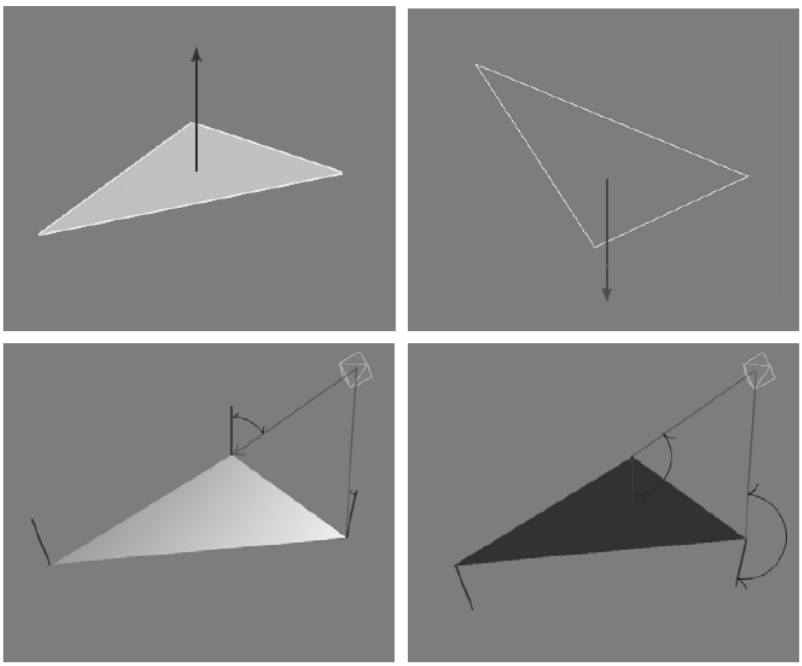
Πέρα από το UV χάρτη, σειρά έχει και η ομαλότητα (normal) των προσόψεων ή τριγώνων των πλεγμάτων στην εφαρμογή μοντελοποίησης Unity3D.

Η πρόσοψη, που αναφέρεται επίσης ως ένα τρίγωνο, είναι το μικρότερο αποδόσιμο μέρος ενός πλέγματος [Sue Blackman, 2011]. Αυτό ορίζεται από τρεις κορυφές, τις ακμές που τις συνδέουν, και η επιφάνεια μεταξύ τους. Μία πρόσοψη που έχει επίσης αυτό που ονομάζεται ομαλότητα (normal) για να πει στη μηχανή σε ποια πλευρά να καταστήσει την πρόσοψη. Εκτός και αν χρησιμοποιούμε μια σκίαση (shader) που ρητά αναφέρει ότι μια πρόσοψη θα πρέπει να τίθεται και στις δύο πλευρές, μια πρόσοψη συντάσσεται μόνο στη μία πλευρά.

Η ομαλότητα πρόσοψης δεν πρέπει να συγχέεται με η ομαλότητα κορυφής . Οι ομαλότητες κορυφής οι ομαλότητες (normals) χρησιμοποιούνται για τον υπολογισμό του φωτός που θα λάβει μια πρόσοψη . Ένας φορέας (vector) ανιχνεύεται από του φωτός που θα λάβει προς η ομαλότητα κορυφής , και η γωνία

πρόσπτωσης, η γωνία μεταξύ των δύο, χρησιμοποιείται για τον υπολογισμό της ποσότητας φωτός που λαμβάνεται από εκείνη τη κορυφή. Όσο μικρότερη είναι η γωνία πρόσπτωσης, τοσο μεγαλύτερο είναι το ποσοστό του φωτός που λαμβάνει. Η κορυφή με ένα φως απευθείας πάνω από την ομαλότητά του θα έχει γωνία πρόσπτωσης 0 βαθμούς και θα λάβουν όλο το ποσό του φωτός.

Περιστασιακά, μπορούμε να δούμε προσόψεις είτε η ομαλότητα πρόσοψης γυρνάει ή η ομαλότητα κορυφής οι ομαλότητες (normals) γυρνάει, δίνοντας πολύ περίεργα αποτελέσματα, όπως φαίνεται στην Εικόνα 45. CAD δεδομένα που χρησιμοποιήθηκαν για πλέγματα συχνά δεν έχουν καθόλου πληροφορίες ομαλότητα πρόσοψης, δεδομένου ότι τείνουν να σύρονται πάντα διπλής όψης, όταν οι ομαλότητες (normals) δημιουργούνται , συχνά συγχέονται.



Εικόνα 45: Προσόψεις και η ομαλότητα πρόσοψης (face normals) τους και η ομαλότητα κορυφής (vertex normals).

Επάνω σειρά: Η ομαλότητα πρόσοψης καθορίζει ποια πλευρά θα συρθεί η πρόσοψη.

- Αριστερά: η ομαλότητα πρόσοψης.
- Δεξιά: η ομαλότητα πρόσοψης είναι στραμμένη προς τα κάτω (το πρόσωπο δεν έχει συρθεί).

Κάτω σειρά: Η ομαλότητα κορυφής και γωνίες πρόσπτωσης. Σημειώστε ότι η μεγάλη γωνία πρόσπτωσης στην πρόσοψη με την ομαλότητα κορυφής δείχνει προς τη λάθος κατεύθυνση.

- Αριστερά: η ομαλότητα πρόσοψης και ομαλότητα κορυφής στραμμένη προς τα επάνω.
- Δεξιά: η ομαλότητα πρόσοψης στραμμένη προς τα επάνω, αλλά η ομαλότητα κορυφής είναι στραμμένη προς τα κάτω.

Είναι μια τεχνική που χρησιμοποιείται για την απομίμηση του φωτισμού των προσκρούσεων και βαθουλωμάτων - μια υλοποίηση της χαρτογράφησης Bump [Wikipedia, 2013]. Χρησιμοποιείται για να προσθέσει λεπτομέρειες χωρίς τη χρήση περισσότερων πολυγώνων. Μια κοινή χρήση αυτής της τεχνικής είναι να ενισχύσει σε μεγάλο βαθμό την εμφάνιση και τις λεπτομέρειες ενός χαμηλού μοντέλου πολυγώνου δημιουργώντας μια ομαλότητα από ένα υψηλό πρότυπο πολυγώνου ή χάρτη ύψους.

Οι χάρτες ομαλότητας συνήθως αποθηκεύονται ως κανονικό RGB , όπου η RGB συνιστώσες αντιστοιχεί στο X, Y, και Z συντεταγμένες, αντίστοιχα, της επιφάνειας.

ΕΠΙΛΟΓΟΣ

Είδαμε τις βασικές αρχές που θεμελιώνουν το CGA σχήμα μέσα από βασικά σχήματα, παραγωγικές διαδικασίες, συμβολισμούς και κανόνες πεδίου εφαρμογής. Επίσης, είδαμε βασικούς κανόνες τμηματοποίησης που βασίζονται στη φιλοσοφία «διαίρει και βασίλευε», και την επανάληψη ορισμένων κανόνων για να συνδέσουν με τη σειρά τους το νήμα της διαδικαστικής μοντελοποίησης, όπου και αφορά αυτή η διαδικασία κανόνων τμηματοποίησης.

Στη συνέχεια, περάσαμε στην πράξη όλων των παραπάνω κανόνων και εντοπίσαμε αυτά τα σημεία σε τμήματα του κώδικα javascript που χρησιμοποιούμε στην εφαρμογή μοντελοποίησης τρισδιάστατου χώρου Unity3D.

Σταθήκαμε πάνω στην πολυπλοκότητα μοντελοποίησης των σκεπών και των τόξων που απαρτίζουν τα κτίρια της παλαιάς Κέρκυρας του 18^{ου}-19^{ου} αιώνα, ώστε να αποτυπωθεί μια πιο ρεαλιστική αίσθηση στα οικοδομήματα.

Ξεχωριστό υποκεφάλαιο αφιερώθηκε στο φινίρισμα των μοντέλων μας καθώς ο χρωματισμός και η απόδοση υφής αφορούν ένα σημαντικό και ξεχωριστής σημασίας κεφάλαιο πάνω στην πτυχιακή εργασία.

Στο επόμενο και τελευταίο κεφάλαιο παρουσιάζονται όλα τα συμπεράσματα της πτυχιακής εργασίας με βάση όσων αναλύθηκαν εκτενέστερα στα προηγούμενα κεφάλαια.

ΣΥΜΠΕΡΑΣΜΑΤΑ

Συμπεράσματα ή και προτάσεις (αποτελεί αυτοτελές τμήμα της εργασίας).

Στην εργασία αυτή παρουσιάσαμε ένα πρόγραμμα που συνδυάζει την παραγωγική δύναμη των γραμματικών σχήματος με το ισχυρό περιβάλλον μοντελοποίησης του Unity3D. Έχουμε λάβει μια σύνθετη, ακαδημαϊκή ιδέα και το φτιάξαμε έτσι ώστε να μπορεί να εφαρμοστεί στο περιβάλλον παραγωγής από τεχνικούς καλλιτέχνες. Γράφοντας κανόνες για γραμματικές σχήματος παίρνει κάποιο χρόνο· ωστόσο εξακολουθεί να είναι ευκολότερη διαδικασία και λιγότερο χρονοβόρα από την χειρονακτική διαδικασία.

Αν και η γενική ιδέα σε αυτή τη διατριβή ήταν η παρουσίαση μιας πόλης μέσω της διαδικαστικής παραγωγής, με βάση τους κανόνες τμηματοποίησης. Η επιλογή της Κέρκυρας ήταν μια ίσως τυχαία επιλογή αν όχι μέρος του στόχου της πτυχιακής εργασίας. Κατά την πορεία της εργασίας αυτής επιθυμήσαμε να προσδώσουμε έναν χαρακτήρα και μια ταυτότητα στην πόλη ώστε να μην έχουμε τη βαρετή συνήθως εμφάνιση γκρίζων, άμορφων κουτιών ή κύβων του προγράμματος μοντελοποίησης Unity3D. Έτσι μετά την απόφαση της ταυτότητας της πόλης περάσαμε στο στάδιο της αρχιτεκτονικής της, αφού η αρχιτεκτονική μιας περιοχής αντιπροσωπεύει την μοναδικότητά της σε σχέση με άλλες πόλεις. Βάση των κανόνων και χαρακτηριστικών της αρχιτεκτονικής του νησιού κατά τον 18ο-19ο αιώνα προσδώσαμε ιδιαίτερη εμφάνιση και λεπτομέρεια στην απεικόνιση των τρισδιάστατων μοντέλων των κτιρίων μας.

Τώρα, με βάση την τεχνική διαδικαστικής παραγωγής, κανόνες τμηματοποίησης (split grammar rules), που ελέχθησαν για την κατασκευή των κτιριακών μοντέλων εξετάζονται με τα παρακάτω κριτήρια αξιολόγησης [George Kelly & Hugh McCabe, 2006]:

1. **Ρεαλισμός (realism):** Η τεχνική της γραμματικής τμηματοποίησης παράγει πολύ ρεαλιστικά κτίρια, ακόμη και φτάνοντας μέχρι το σημείο να αναδημιουργήσει αποτελεσματικά διαφορετικά στυλ αρχιτεκτονικής.
2. **Κλίμακα (scale):** Το παράδειγμα που εμφανίζεται στην εργασία είναι περιορισμένης κλίμακας, αποδεικνύει όμως τα πλεονεκτήματα του συστήματος με τη δημιουργία μιας μικρής ομάδας κτιρίων σε μια πλατεία της πόλης ή του κέντρου. Ένα υψηλό επίπεδο μεταβλητότητας φαίνεται στο παράδειγμα, αλλά ο αριθμός των κτιρίων είναι περιορισμένος και δεν είναι απόλυτα ίδιος με εκείνη της κλίμακας της πόλης.
3. **Μεταβολή (variation)**: Το στυλ των κτιρίων ποικίλλει σε μεγάλο βαθμό συμβάλλοντας έτσι στην παραγωγή πολύ ρεαλιστικών αποτελεσμάτων, όμως δεν είναι σαφές πόσα διαφορετικά είδη κτιρίων μπορούν να παραχθούν.
4. **Είσοδος (input)**: Το σύστημα απαιτεί ουσιαστική αρχική είσοδο με δείγματα, όπως αυτά που απεικονίζονται στο έργο, που απαιτεί μια βάση δεδομένων που περιέχει περίπου εκατοντάδες κανόνες και ιδιότητες, και

πήρε περίπου αρκετούς μήνες να συγκεντρωθούν. Από αυτή τη βάση δεδομένων μια ποικιλία κτιρίων από διαφορετικά στυλ μπορούσαν να δημιουργηθούν και τα δεδομένα θα μπορούσαν να διανέμονται με το σύστημα χωρίς να χρειάζεται ο χρήστης να συγκεντρώσει το δικό του σύνολο δεδομένων.

5. **Αποτελεσματικότητα (efficiency)**: Ο αλγόριθμος παρόλο πολύπλοκος είναι αρκετά αποτελεσματική η δημιουργία κτιρίων της 40.3k τρίγωνα και 73.2k κορυφές σε περίπου 1 δευτερόλεπτο σε έναν επεξεργαστή Intel Pentium Dual στα 1.46Ghz 1.47Ghz.
6. **Έλεγχος (control)**: Υπάρχει ένα διαδραστικό πρόγραμμα επεξεργασίας GUI που δίνει στο χρήστη τη δυνατότητα να περιηγείται στο χώρο με τη χρήση καμερών. Επίσης, στο χρήστη παρέχεται και ένα κουμπί εκτέλεσης της τυχαίας κάθε φορά παραγωγής κτιρίων στον τρισδιάστατο χώρο. Όμως οι κανόνες γραμματικής τμηματοποίησης μπορούν να επεξεργαστούν από πλευράς κώδικα με μη αυτόματο τρόπο. Αυτή η διαδικασία περιγράφεται ως μη τετριμμένη και απαιτεί ένα επίπεδο τεχνογνωσίας και εμπειριών με τις γραμματικές τμηματοποίησης. Θα μπορούσε κάλλιστα να είναι ένα εμπόδιο για την επέκταση του συστήματος. Μπορεί επίσης να υπάρχουν περιορισμοί σχετικά με το μέγεθος του συστήματος και τον αριθμό των κανόνων που μπορεί το σύστημα να διαχειριστεί με επιφύλαξη.
7. **Σε πραγματικό χρόνο (real-time):** Οι λεπτομερείς στα κτίρια που το σύστημα παράγει μπορούν να διερευνηθούν σε πραγματικό χρόνο, ωστόσο, ο αριθμός κτιρίων στην οθόνη κάθε φορά είναι περιορισμένος για αυτό και δίνεται στο χρήστη η δυνατότητα ελέγχου και παρακολούθησης του χώρου μέσα από άλλες οπτικές γωνίες. Είναι σαφώς ένα όριο του συστήματος με μια τέτοια υψηλή καταμέτρηση πολυγώνων. Το επίπεδο στήριξης λεπτομέρειας θα είναι απαραίτητο αν χρησιμοποιηθεί το σύστημα για εφαρμογές πραγματικού χρόνου.

Το έργο αυτό πραγματοποιήθηκε σε Λειτουργικό Σύστημα Windows 7. Με επεξεργαστή Intel Pentium Dual στα 1.46Ghz 1.47Ghz, με RAM 2GB και κάρτας γραφικών Mobile Intel(R) 965 Express Chipset Family.

Σημαντικό επίσης να αναφέρουμε εδώ πως το έργο αυτό έχει δυνατότητα ανάπτυξης στο μέλλον. Αργότερα με στόχο την ανάπτυξή του έργου σε μεγαλύτερα γεωγραφικά όρια από αυτά που παρουσιάζονται εδώ μπορεί να διατεθεί σε μουσεία της πόλης της Κέρκυρας, αποσκοπώντας στον τουρισμό και την εκπαίδευση του νησιού. Μια επίσης καλή πρακτική θα ήταν για τουριστικούς αλλά και λόγους διαφήμισης του νησιού προς την υπόλοιπη Ελλάδα αλλά και στο εξωτερικό να φιλοξενηθεί σε κάποιο ιστότοπο που προβάλει την ιστορία του νησιού της Κέρκυρας. Επίσης θα μπορούσε να παρουσιαστεί η ίδια η εφαρμογή και ίσως η πιο ανεπτυγμένη μορφή της στο μέλλον σε σχολικούς ή εκπαιδευτικούς χώρους που θα μαθαίνουν για την ιστορία και την αρχιτεκτονική της πόλης των Φαιάκων υπό μορφή σεμιναρίων όπου το έργο θα συμπαρουσιάζεται με κάποιον

φιλόλογο ή ιστορικό. Παράλληλα θα μπορούσε να αποτελέσει και πλατφόρμα εκπαιδευτικού παιχνιδιού.

Τέλος, η διαδικαστική μοντελοποίηση αποτελεί μεγάλο πεδίο έρευνας που ακόμα και σήμερα δεν έχει σταματήσει να απασχολεί.

ΑΝΑΦΟΡΕΣ

Akenine-Möller Tomas, Eric Haines (2002). Real-Time Rendering.A K Peters,Ltd. 2002.

Alexander C, Ishikawa S, Silverstein M (1977); A Pattern Language: Towns,Buildings, Construction. Oxford University Press.

Alexander, C., Ishikawa, S., and Silverstein, M. (1977). A Pattern Language: Towns, Buildings, Construction. Oxford University Press, New York.

Aliaga D. G., Beneˇs B., Vanegas C. A., Andrysco N. (2008): Interactive reconfiguration of urban layouts. IEEE Computer Graphics & Applications 28, 3, pp.38–47.

Aliaga D. G., C. A. Vanegas, and B. Beneˇs, (2008) “Interactive example-based urban layout synthesis,” ACM Trans. Graph., vol. 27, no. 5, pp.1–10.

Apodaca, A. A., and Gritz, L. (2000). Advanced Renderman: Creating CGI for Motion Pictures. Academic Press, San Diego, California.

Batty M. (1992):Urban modeling in computer-graphic and geographic information system environments. Environment and Planning B: Planning and Design 19, 6, pp.663–688.

Batty M. (2007): Cities and Complexity: Understanding Cities with Cellular Automata, Agent-Based Models, and Fractals. The MIT Press, September 2007.

Batty M.,Cole S. (1997): Time and space: Geographic perspectives on the future. Futures 29, 4–5, 277–289.

Birch P., Browne S., Jennings V., Day A., Arnold D. (2001): Rapid Procedural-Modeling of Architectural Structures. In Virtual Reality, Archaeology and Cultural Heritage (VAST), pp.187–196.

Blackman Sue (2011). Beginning 3D Game Development with Unity: The World’s Most Widely Used Multiplatform Game Engine Copyright © 2011. ISBN-13 (electronic): 978-1-4302-3423-4. Mapping: pg.83-92.

Bruneton E., Neyret F. (2008): Real-time rendering and editing of vector-based terrains. In Eurographics ’08, April, (Hersonissos, Gr`ece, 2008), Drettakis G., Scopigno R., (Eds.), vol. 27.

Cabral M., Lefebvre S., Dachsbacher C., Drettakis G. (2009): Structure preserving reshape for textured architectural scenes. Computer GraphicsForum (Proceedings of the Eurographics Conference).

Cagdas G (1996) A shape grammar: the language of traditional Turkish houses. Environment and Planning B: Planning and Design 23(4):443-464

Carrozzino M., Tecchia F., Bacinelli S., Cappelletti C., Bergamasco M. (2005) Lowering the development time of multimodal interactive application: the real-life experience of the XVR project, in Proceedings of ACM SIGCHI International Conference on Advances in Computer Entertainment Technology, ACE 2005, June 2005.

Carrozzino M., Tecchia F., Bergamasco M. (2009). “urban procedural modeling for real-time rendering”. Proceedings of the 3rd ISPRS International Workshop 3D-ARCH.

Catmull, E. E. (1974). A subdivision algorithm for computer display of curved surfaces. Ph.D. thesis, Department of Computer Science, University of Utah.

Chen G., G. Esch, P. Wonka, P. Müller, and E. Zhang, (2008) “Interactive procedural street modeling,” ACM Trans. Graph., vol. 27, no. 3.

Ching, F. D. K. (1996).A Visual Dictionary of Architecture.Wiley.

Chiou SC, Krishnamurti R (1995) The fortunate dimensions of Taiwanese traditional architecture. Environment and Planning B: Planning and Design 22: 547-562

Cutler B., J. Dorsey, L. McMillan, M. Müller, and R. Jagnow, (2002) “A procedural approach to authoring solid models,” ACM Trans. Graph., vol. 21, no. 3, pp.302–311.

Davis, M., Sigal, R., Weyuker, E. J., and Davis, M. D. (1994). Computability, Complexity, and Languages: Fundamentals of Theoretical Computer Science.Academic Press.

Di Giacomo, T., Capo, S. and Faure, F. (2001). An interactive forest. Proc. of the 2001 Eurographics Workshop on Computer Animation and Simulation, pp.65-74.

Downing, F., and Flemming, U. (1981). The bungalows of buffalo.Environment and Planning B 8, pp. 269–293.

Drettakis G., Roussou M., Reche A., Tsingos N. (2007): Design and evaluation of a real-world virtual environment for architecture and urban planning. Presence: Teleoper. Virtual Environ.16, 3, pp.318–332.

Duarte JP (2005) Towards the mass customization of housing: the grammar of Siza’s houses at Malagueira. Environment and Planning B: Planning and Design32: pp.347-380.

Duarte, J. (2002). Malagueira Grammar – towards a tool for customizing Alvaro Siza’s mass houses at Malagueira. PhD thesis, MIT School of Architecture and Planning.

Ebert David S., F. Kenton Musgrave, Darwyn Peachey, Ken Perlin, and Steven Worley (2003) Texturing & Modeling. A Procedural Approach, Third Edition - ISBN 1558608486 -721s –LRN.

Ehrig, H., Engels, G., Kreowski, H.-J., and Rozenberg, G. (1999).Handbook of Graph Grammars and Computing by Graph Transformation: Applications, Languages and Tools.World Scientific Publishing Company.

Finkenzeller D. (2008): Detailed building facades. IEEE Computer Graphics & Applications 28, 3, pp.58–66.

Flemming, U. (1987). More than the sum of its parts: the grammar of queen anne houses. Environment and Planning B 14, pp.323–350.

Funkhouser T., Kazhdan M., Shilane P., Min P., Kiefer W., Tal A., Rusinkiewicz S., and D. Dobkin, (2004) “Modeling by example,” SIGGRAPH ’04.

Greuter, S., Parker, J., Stewart, N., Leach, G., (2003). Real-time Procedural Generation of ‘Pseudo Infinite’ Cities. Proc. of GRAPHITE 2003, ACM SIGGRAPH, pp.87-94.

Guerraz, S., Perbet, F., Raulo, D., Faure, F., and Cani, M-P., (2003). A Procedural Approach to Animate Interactive Natural Sceneries, Proc. of the 16th Int’1Conference on Computer Animation and Social Agents (CASA 2003), pp.73-78.

Hahn, E. ,Bose, P., Whitehead, A. (2006), Persistent Realtime Building Interior Generation Proc. of Sandbox Symposium 2006, pp.179-186.

Hart Evan (2002), ATI Research; 3D Textures and Pixel Shaders, ShaderX vertex and pixel tips and tricks.

Havemann, S. (2005). Generative Mesh Modeling. PhD thesis, TU Braunschweig.

Hertzmann A., Jacobs C. E., Oliver N., Curless B., Salesin D. H. (2001): Image analogies. In Proceedings of ACM SIGGRAPH, pp.327–340.

Hillier B., A. Penn, J. Hanson, Grajewski and J. Xu (1993). Natural Movement: or, Configuration and Attraction in Urban Pedestrian Movement. Environment and Planning B, Vol. 20, pp.29-66.

Hillier, B. (1996). Space Is The Machine: A Configurational Theory Of Architecture. Cambridge University Press.

Interactive Data Visualization Inc.., (2006) SpeedTree RT.http://www.speedtree.com

International Scene Organization (2004). Scene Awards 2004.http://scene.org/awards.php?year=2004.

Keim D. A., North S. C., Panse C.: Cartodraw (2004): A fast algorithm for generating contiguous cartograms. IEEE Transactions on Visualization and Computer Graphics 10,1, pp.95–110.

Kelly George, Hugh McCabe (2006), ITB Journal «A Survey of Procedural Techniques for City Generation», School of Informatics and Engineering, Institute of Technology, Blanchardstown, Dublin 15, Ireland

Knight TW (1992) Designing with grammars. in Hatch (ed), Computer-Aided Architectural Design. Van Nostrand-Reinhold, New York.

Knuth, D. (1968). Semantics of context-free languages. Mathematical Systems Theory 2, 2, pp.127–145.

Koning, H., and Eizenberg, J. (1981). The language of the prairie: Frank lloyd wrights prairie houses. Environment and Planning B 8, pp.295– 323.

Kwatra V., Essa I., Bobick A., Kwatra N. (2005): Texture optimization for example-based synthesis. ACM Transactions on Graphics 24, 3, pp.795–802.

Legakis J., Dorsey J., and Gortler S., (2001) “Feature-based cellular texturing for architectural models,” in Proc. Of ACM SIGGRAPH ’01, pp.309–316.

Lindenmayer, A. (1968). Mathematical models for cellular interaction in development: Parts i and ii. Journal of Theoretical Biology 18.

Lintermann, B. and Deussen, O. (1999), Interactive Modeling of Plants. IEEE Computer Graphics and Applications 19(1): pp.56-65.

Lipp, M. (2007). Interactive Computer Generated Architecture. Master’s thesis, Technical University of Vienna.

Lipp, M., Wonka, P., and Wimmer, M. (2008). Interactive visual editing of grammars for procedural architecture. ACM Transactions on Graphics 27, No.3, pp.10.

Liu Y., Pottmann H., Wallner J., Yang Y.-L., Wang W. (2006): Geometric modeling with conical meshes and developable surfaces. ACM Transactions on Graphics 25, 3, pp.681–689.

Lynch Kevin (1960) ‘The Image of the City’.Cambridge:MIT Press.

March, L., and Steadman, P. (1974). The Geometry of Environment. MIT Press.

Marshall, R., Wilson, R., and Carlson, W. (1980). Procedure models for generating three-dimensional terrain. In SIGGRAPH, ACM Press, pp.154–162.

Marvie J. E., Perret J., Bouatouch K. (2005): The FLsystem: A functional L-system for procedural geometric modeling. The Visual Computer 21,5, pp.329–339.

Merrell P. (2007): Example-based model synthesis. In ACM Symposium on Interactive 3D Graphics and Games, pp.105–112.

Merrell P., Manocha D. (2008): Continuous model synthesis. ACM Transactions on Graphics 27, 5, pp.1–7.

Mitchell, W. J. (1990). The Logic of Architecture: Design, Computation, and Cognition. MIT Press.

Mullen, T (2009). Mastering Blender. 1st ed. Indianapolis, Indiana: Wiley Publishing, Inc.

Müller, P., Wonka, P., Haegler, S., Ulmer, A., and Van Gool, L., (2006). Procedural modeling of buildings. ACM Transactions on Graphics 25, 3, pp.614–623.

Murdock, K.L. (2008). 3ds Max 2009 Bible. 1st ed. Indianapolis, Indiana: Wiley Publishing, Inc.

Musgrave Ken (2006), Pandromeda. Mojo World Applications.http://www.pandromeda.com/products/

MusgraveF.K.,C.E.KolbandR.S.Mace (1990),The Synthesis and Rendering of Eroded Fractal Terrains,In SIGGRAPH 89 Proceedings, pp.41-50.

Nelson CB, Fenves SJ (1990) Manipulating shape and its function. Journal of Computing in Civil Engineering 4(3): pp.221-238

Parish Y. I. H., Müller P. (2001). Procedural Modeling of Cities, In Proceedings of ACM SIGGRAPH 2001, ACM Press / ACM SIGGRAPH, New York, Annual Conference Series, ACM, pp.301-308.

Pecchioli L. Carrozzino M., Mohamed F. (2008) ISEE: Accessing Relevant Information By Navigating 3d Interactive Virtual Environments, in Proceedings of the 14th International Conference on Virtual Systems and Multimedia, IEEE VSMM 2008, pages 326-331, Limassol, Cyprus.

Peponis J., C. Zimring and Y.K. Choi (1990). Finding the Building in Wayfinding. In Environment and Behavior, Vol. 22, pp. 555- 590.

Perlin Ken (1985).An Image Synthesizer, in Proc ACM SIGGRAPH pp.287-296.

Perlin Ken (1999) Making Noise,http://www.noisemachine.com/talk1/index.html.

Pinnel L. D., Dockrey M., Brush A. J. B., Borning A. (2000): Design of visualizations for urban modelling. In VisSym: Joint Eurographics and IEEE TCVC Symposium on Visualization.

Pixar. (1989).The RenderMan interface:Version 3.1.San Rafael,CA:Pixar

Pottmann H., Liu Y., Wallner J., Bobenko A., and Wang W., (2007) “Geometry of multi-layer freeform structures for architecture,” Proc.Of ACM SIGGRAPH’07.

Pottmann H., Schiftner A., Bo P., Schmiedhofer H., Wang W., Baldassini N., Wallner J. (2008): Freeform surfaces from single curved panels. ACM Transactions on Graphics 27,3, pp.1–10.

Procedural:www.procedural.com, 2008

Prusinkiewicz P. and A. Lindenmayer (1990).The algorithmic beauty of plants,Springer-Verlag New York,Inc.

Radoff Jon (2007). Anatomy of an MMORPG,http://radoff.com/blog/2008/08/22/anatomy-of-an-mmorpg/,Originally published March 27, 2007 on PlayerVox.

Reeves W.T. and R. Blau (1985),Approximate and Probabilistic Algorithms for Shading and Rendering Structured Particle Systems.Computer Graphics (SIGGRAPH 85 Proceedings),19(3): pp.313-322.

Roman A., Garg G., Levoy M. (2004): Interactive design of multi-perspective images for visualizing urban landscapes. In IEEE Visualization, pp.537–544.

Sander P. V., Snyder J., Gortler S. J., Hoppe H. (2001): Texture mapping progressive meshes. In SIGGRAPH ’01: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (New York, NY, USA, 2001), ACM, pp.409–416.

Schwartzman Y., Borning A. (2007): The indicator browser: A web-based interface for visualizing urbanism simulation results. Hawaii International Conference on System Sciences, 92a.

Shubnikov, A. V., and Koptsik, V. A. (1974). Symmetry in Science and Art. Plenum Press, New York.

Side Effects Software (2005).Manufacturer of Houdini,http://www.sidefx.com.

Sipser, M. (1996).Introduction to the Theory of Computation. Course Technology,Boston.

Smith J., Hodgins J., Oppenheim I., and Witkin A., (2002) “Creating models of truss structures with optimization,” ACM Trans. Graph., vol. 21, no. 3, pp.295–301.

Smith, A. R. (1984).Plants, fractals and formal languages.In H. Christiansen, ed.,Computer Graphics (SIGGRAPH ’84 Proceedings),18: pp.1–10.

Spitzer John, Green Simon and NVIDIA Corporation; (2003). Noise and Procedural Techniques. In Proceedings of Game Developers Conference 2003,GDC.

Stiny G, Mitchell WJ (1978) The Palladian grammar. Environment and Planning B: Planning and Design 5: pp.5-18

Stiny, G. (1980). Introduction to shape and shape grammars. Environment and Planning B 7, pp.343–361.

Stiny, G. N. (1975). Pictorial and formal aspects of shape and shape grammars and aesthetic systems. PhD thesis, University of California, Los Angeles.

Stiny, G., and Gips, J. (1971). Shape grammars and the generative specification of painting and sculpture. In IFIP Congress (2), pp.1460–1465.

Stiny, G., and Mitchell, W. J. (1978). The palladian grammar. Environment and Planning B 5, pp.5–18.

Thomas H. Kolbe, Gerhard Gröger, Lutz Plümer (2005). CityGML – Interoperable Access to 3D City Models. Article published in Oosterom, Zlatanova, Fendel (Eds.): Proceedings of the Int. Symposium on Geo-information for Disaster Management on 21.-23. March 2005 in Delft, Springer Verlag, pp. 4-5.

Torrens P., David, and Sullivan, (2001) “Cellular automata and urban simulation: where do we go from here?” Environment and Planning B: Planning and Design March 2001, vol. 28, no. 2, pp.163–168.

Vanegas C. A., D. G. Aliaga, P. Wonka, P. Müller, P. Waddell and B. Watson (2009), “Modelling the Appearance and Behaviour of Urban Spaces”, Journal compilation c_ 2009 The Eurographics Association and Blackwell Publishing Ltd. Published by Blackwell Publishing, 9600 Garsington Road, Oxford OX4 2DQ, UK and 350 Main Street, Malden, MA 02148,USA, vol. 29, no. 1, pp.25-42..

Waddell P., (2002) “Urbansim: Modeling urban development for land use, transportation and environmental planning,” Journal of the American Planning Association, vol. 68, pp.297–314.

Watson B., P. Müller, O. Veryovka, A. Fuller, P. Wonka, and C. Sexton, (2008) “Procedural urban modeling in practice,” IEEE Comput. Graph. Appl., vol. 28, no. 3, pp.18–26.

Watson Ben, Thomas Lechner, Uri Wilensky, Martin Felsen (2003). Procedural City Modeling.

Weber B., Müller P., Wonka P., and Gross M., (2009) “Interactive geometric simulation of 4d cities,” Computer Graphics Forum,April 2009.

Wei L.-Y., Lefebvre S., Kwatra V., Turk G. (2009): Eurographics STAR, State of the Art in Example-based texture synthesis.

Wells W.D., (2005). Generating Enhanced Natural Environments and Terrain for Interactive Combat Simulations. Doctoral Dissertation, Naval Post Graduate School, Monterey (CA).

Weyl, H. (1952).Symmetry.Princeton University Press.

Whiting, E., Ochsendorf, J., and Durand, F. (2009). Procedural modeling of structurally-sound masonry buildings. In siggraph Asia ’09: acm siggraph Asia 2009 papers, ACM, New York, NY, USA, pp.1–9.

Wonka, P., Wimmer, M., Sillion, F., and Ribarsky, W. (2003). Instant architecture. In siggraph ’03: acm siggraph (2003) Papers, ACM, New York, NY, USA, pp.669–677.

Xiao J., Fang T., Tan P., Zhao P., Ofek E., Quan L. (2008): Image-based facade modeling. ACM Transactions on Graphics 27,5, pp.1–10.

ΠΑΡΑΡΤΗΜΑΤΑ

//Main_GUI.js

```
public var pln1a: Pl_lvl_1a;

public var pln1b: Pl_lvl_1b;

public var pln1c: Pl_lvl_1c;

public var pln1d: Pl_lvl_1d;

public var pln2a: Pl_lvl_2a;

public var pln2b: Pl_lvl_2b;

public var pln2c: Pl_lvl_2c;

public var pln2d: Pl_lvl_2d;

public var pln2e: Pl_lvl_2e;

public var pln3a: Pl_lvl_3a;

public var pln3b: Pl_lvl_3b;

public var pln3c: Pl_lvl_3c;

public var pln3d: Pl_lvl_3d;

public var pln3e: Pl_lvl_3e;

public var pln3f: Pl_lvl_3f;

public var pln3g: Pl_lvl_3g;

public var pln3h: Pl_lvl_3h;

public var pln4: Pl_lvl_4;

public var pln5a: Pl_lvl_5a;

public var pln5b: Pl_lvl_5b;

public var pln5c: Pl_lvl_5c;

public var pln5d: Pl_lvl_5d;

public var pln5e: Pl_lvl_5e;

public var pln5f: Pl_lvl_5f;
```

```
public var pln5g: Pl_lvl_5g;

public var pln5h: Pl_lvl_5h;

public var onoff: boolean = false;

function OnGUI () {

	//create background boxes and buttons, available on GUI

	GUI.Box(Rect(0,0,120,65), "Top-left Camera"); //GUI background Box for Top-left Camera

	GUI.Box(Rect(Screen.width - 700,0,120,65), "Front Camera");//GUI background Box for Front Camera

	GUI.Box(Rect(Screen.width - 120,0,120,65), "Top-right Camera"); //GUI background Box for Top-right Camera

	GUI.Box (Rect (0,Screen.height - 60,120,65), "Bottom-left Camera");//GUI background Box for Bottom-left Camera

	GUI.Box (Rect (Screen.width - 130,Screen.height - 60,130,65), "Bottom-right Camera");//GUI background Box for Bottom-left Camera


	//---------------------------------------create buttons---------------------------------------

	if(GUI.Button(Rect(40,25,37,37), Resources.Load("left_cam"))){ //if you press the button with rectangle size and image from resources then...

		Instantiate(GameObject.Find("top_left_Camera")); // finds the gameobject by tag name top_left_Camera and creates a clone of that camera gameobject

	}//end if


	if(GUI.Button(Rect(Screen.width - 660,25,37,37), Resources.Load("front cam"))){

		Instantiate(GameObject.Find("Main Camera"));

	}//end if


	if(GUI.Button(Rect(Screen.width-80,25,37,37), Resources.Load("right_cam"))){
```

```
            Instantiate(GameObject.Find("top_right_Camera"));

        }//end if

        if(GUI.Button(Rect(40,Screen.height-35,37,37),
Resources.Load("left_cam"))){

            Instantiate(GameObject.Find("bottom_left_Camera"));

        }//end if

        if(GUI.Button(Rect(Screen.width-80,Screen.height-35,37,37),
Resources.Load("right_cam"))){

            Instantiate(GameObject.Find("bottom_right_Camera"));

        }//end if

        if(GUI.Button(Rect(Screen.width-660,Screen.height-
45,50,50),Resources.Load("town"))){

            //generate town

            if(onoff == false ){

                pln1a.Plane_lvl_1a();

                pln1b.Plane_lvl_1b();

                pln1c.Plane_lvl_1c();

                pln1d.Plane_lvl_1d();

                pln2a.Plane_lvl_2a();

                pln2b.Plane_lvl_2b();

                pln2c.Plane_lvl_2c();

                pln2d.Plane_lvl_2d();

                pln2e.Plane_lvl_2e();

                pln3a.Plane_lvl_3a();

                pln3b.Plane_lvl_3b();

                pln3c.Plane_lvl_3c();

                pln3d.Plane_lvl_3d();

                pln3e.Plane_lvl_3e();
```

```
            pln3f.Plane_lvl_3f();
            pln3g.Plane_lvl_3g();
            pln3h.Plane_lvl_3h();
            pln4.Plane_lvl_4();
            pln5a.Plane_lvl_5a();
            pln5b.Plane_lvl_5b();
            pln5c.Plane_lvl_5c();
            pln5d.Plane_lvl_5d();
            pln5e.Plane_lvl_5e();
            pln5f.Plane_lvl_5f();
            pln5g.Plane_lvl_5g();
            pln5h.Plane_lvl_5h();
            GameObject.Find("Smokes").particleEmitter.emit= false;
            onoff=true;
}
else if (onoff == true){ //demolish town
pln1a.Demolish();
pln1b.Demolish();
pln1c.Demolish();
pln1d.Demolish();
pln2a.Demolish();
pln2b.Demolish();
pln2c.Demolish();
pln2d.Demolish();
pln2e.Demolish();
pln3a.Demolish();
pln3b.Demolish();
```

```
                pln3c.Demolish();
                pln3d.Demolish();
                pln3e.Demolish();
                pln3f.Demolish();
                pln3g.Demolish();
                pln3h.Demolish();
                pln4.Demolish();
                pln5a.Demolish();
                pln5b.Demolish();
                pln5c.Demolish();
                pln5d.Demolish();
                pln5e.Demolish();
                pln5f.Demolish();
                pln5g.Demolish();
                pln5h.Demolish();
                GameObject.Find("Smokes").particleEmitter.emit  = true;
                onoff = false;
                }
        }//end if
}//end function onGUI
```

//Church.js

```
//create objects
var oChurch : GameObject;
var oWindow : GameObject;
var oCube : GameObject;
var oColumn: GameObject;
```

```
var tmpobj: GameObject;

var oCylinder: GameObject;

var oArch: GameObject;

var oUpperEave_: GameObject;

var oCupola: GameObject;

//-----------------------start public variables---------------------------------

public var i: int = 0;

public var j: int = 0;

public var c: int = 0;


//Random ranges between X, Y coordinates

public var RandX:float;

public var RandZ:float;


public var aCollectionObj_6 = new Array(); // an Array - a collection- for the objects of Church

//-----------------------end public variables-----------------------------------



//MAIN function

public function mainDO() {


        //RandX = Random.Range(10,247);

        //RandZ = Random.Range(10,247);


        MainBuilding(RandX, RandZ);

        BellTower(RandX, RandZ);
```

```
        FrontPrt(RandX, RandZ);
        RightPrt(RandX, RandZ);
        Windows(RandX, RandZ);
        Doors(RandX, RandZ);
        Stairs(RandX, RandZ);
        Eaves(RandX, RandZ);
        MainWind(RandX, RandZ);
        Do_ArchWind(RandX, RandZ);
        MainRoof(RandX, RandZ);
        RightRoof(RandX, RandZ);
        BackRoof(RandX, RandZ);
        Do_ArchDoor(RandX, RandZ);
        Do_Arch(RandX, RandZ);
        Do_DoorRoof(RandX, RandZ);
        Do_BellTower_Arch(RandX, RandZ);
}

public function Destroy_(){// destroy the house

DestroyImmediate(oChurch);

}

//----------------------Start Main Building-----------------------------------------------------------
--
public function MainBuilding(RandX, RandZ){
```

```
        oChurch =  GameObject.CreatePrimitive(PrimitiveType.Cube);
        oChurch.transform.position = Vector3(this.RandX, 3.985427, this.RandZ);
        oChurch.name = "Church";
        oChurch.renderer.material.color = Color(0.996, 0.9411, 0.8392, 0);
        oChurch.transform.localScale.x = 12.90037;
        oChurch.transform.localScale.y = 7.680003;
        oChurch.transform.localScale.z = 6.561959;
        aCollectionObj_6.Add(oChurch);
        //}//end if
}//end function MainBuilding(Rand_X,Rand_Z)
//----------------------End Main Building-----------------------------------------------------------
-



//-----------------------Start  BellTower-------------------------------------------------------------
--
public function BellTower(RandX, RandZ){
        oCube = GameObject.CreatePrimitive(PrimitiveType.Cube);
        Destroy(oCube);
        oCylinder = GameObject.CreatePrimitive(PrimitiveType.Cylinder );
        Destroy(oCylinder);


        if(oCube){
                var oBellTower = Instantiate(oCube, Vector3(this.RandX + 8.43,
8.4548, this.RandZ + 3.509899), Quaternion.identity);
                oBellTower.transform.parent = oChurch.transform; //parenting
                oBellTower.name = "Bell Tower";
                oBellTower.renderer.material.color = Color(0.996, 0.9411, 0.8392,
0);
```

```
                oBellTower.transform.localScale.x = 0.3027877;
                oBellTower.transform.localScale.y = 2.178845;
                oBellTower.transform.localScale.z = 0.5168234;
                aCollectionObj_6.Add(oBellTower);
        }//end if

        var BTposX:float = this.RandX + 8.43;
        var BTposY:float = 16.9;
        var BTposZ:float = this.RandZ + 3.509899;

        //::::::::::::start
eaves::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::::::::::::::::::::
        for(i=0; i<=2;i++){
                if(oCube){
                        var oBellTowerEaves = Instantiate(oCube, Vector3(BTposX,
BTposY, BTposZ), Quaternion.identity);
                        oBellTowerEaves.transform.parent = oChurch.transform;
//parenting
                        oBellTowerEaves.name = "Bell Tower eaves " + (++c);
                        oBellTowerEaves.renderer.material.color = Color.white;
                        if(i==1){
                                oBellTowerEaves.transform.localScale.x = 0.3141422;
                                oBellTowerEaves.transform.localScale.z = 0.5426651;
                        }//end if
                        else{
                                oBellTowerEaves.transform.localScale.x = 0.3027877;
                                oBellTowerEaves.transform.localScale.z = 0.5168234;
                        }//end else
```

```
                oBellTowerEaves.transform.localScale.y = 0.02257257;
                aCollectionObj_6.Add(oBellTowerEaves);
            }//end if
            BTposY += 0.15;
        }//end for

        BTposY = 20.06;
        for(i=0;i<=5;i++){
            if(oCube){
                var oBellTowerEaves_upper = Instantiate(oCube,
Vector3(BTposX, BTposY, BTposZ), Quaternion.identity);
                oBellTowerEaves_upper.transform.parent =
oChurch.transform; //parenting
                oBellTowerEaves_upper.name = "Bell Tower  eaves " + (++c);
                oBellTowerEaves_upper.renderer.material.color =
Color.white;
                if(i==1 || i==4){
                    oBellTowerEaves_upper.transform.localScale.x =
0.3141422;
                    oBellTowerEaves_upper.transform.localScale.z =
0.5426651;
                }//end if
                else if(i==2 || i==5){
                    oBellTowerEaves_upper.transform.localScale.x =
0.329285;
                    oBellTowerEaves_upper.transform.localScale.z =
0.5685066;
                }//end else if
                else{
```

```
                    oBellTowerEaves_upper.transform.localScale.x          =
0.3027877;
                    oBellTowerEaves_upper.transform.localScale.z          =
0.5168234;
                }//end else
                oBellTowerEaves_upper.transform.localScale.y               =
0.02257257;
                aCollectionObj_6.Add(oBellTowerEaves_upper);
            }//end if
            if( i<2 || i>=3)
                BTposY += 0.15;
            else
                BTposY += 2.3;
        }//end for
        c = 0;
        //::::::::::::end
eaves:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::::::::::::::::::::


        //:::::::::::::start attic::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
        BTposY = 21.5;
        if(oCube){
            var oBellTowerAttic = Instantiate(oCube, Vector3(BTposX, BTposY,
BTposZ), Quaternion.identity);
            oBellTowerAttic.transform.parent = oChurch.transform; //parenting
            oBellTowerAttic.name = "Bell Tower attic ";
            oBellTowerAttic.renderer.material.color    =    Color(0.996,    0.9411,
0.8392, 0);
            oBellTowerAttic.transform.localScale.x = 0.3027877;
            oBellTowerAttic.transform.localScale.y = 0.2899312;
```

```
oBellTowerAttic.transform.localScale.z = 0.5168234;
aCollectionObj_6.Add(oBellTowerAttic);
}//end if
//::::::::::::end attic::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::


//:::::::::::::start
columns:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
:::::::::::::::::::
BTposX = this.RandX + 6.7729;
BTposY = 18.2;
BTposZ = this.RandZ + 4.9214;


for(i=0;i<=1;i++){
for(j=0; j<=1; j++){
if(oCube){
var oBellTowerColumns = Instantiate(oCube,
Vector3(BTposX, BTposY, BTposZ), Quaternion.identity);
oBellTowerColumns.transform.parent =
oChurch.transform; //parenting
oBellTowerColumns.name = "Bell Tower column " +
(++c);
oBellTowerColumns.renderer.material.color =
Color.white;
oBellTowerColumns.transform.localScale.x =
0.04627316;
oBellTowerColumns.transform.localScale.y =
0.2422188;
oBellTowerColumns.transform.localScale.z =
0.08771555;
aCollectionObj_6.Add(oBellTowerColumns);
}//end if
```

```
                BTposX += 3.31;
            }//end for
            BTposX = this.RandX + 6.7729;
            BTposZ -= 2.8154;
        }//end for
        c = 0;
        //:::::::::::::end columns::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::


        //:::::::::::::start middle columns:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
        BTposX = this.RandX + 6.7729;
        BTposY = 18.2;
        BTposZ = this.RandZ + 3.52166;
        for(i=0; i<=1; i++){
            if(oCylinder){
                var oBellTowerMdlClmns_H = Instantiate(oCylinder, Vector3(BTposX, BTposY, BTposZ), Quaternion.identity);
                oBellTowerMdlClmns_H.transform.parent = oChurch.transform; //parenting
                oBellTowerMdlClmns_H.name = "Bell Tower middle horizontal column " + (++c);
                oBellTowerMdlClmns_H.renderer.material.color = Color.white;
                oBellTowerMdlClmns_H.transform.localScale.x = 0.03294696;
                oBellTowerMdlClmns_H.transform.localScale.y = 0.1176622;
                oBellTowerMdlClmns_H.transform.localScale.z = 0.05811055;
                aCollectionObj_6.Add(oBellTowerMdlClmns_H);
            }//end if
            BTposX += 3.31;
```

```
}//end for

c = 0;


BTposX = this.RandX + 8.3671;

BTposY = 18.2;

BTposZ = this.RandZ + 2.106;

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

    if(oCylinder){

        var oBellTowerMdlClmns_V = Instantiate(oCylinder, Vector3(BTposX, BTposY, BTposZ), Quaternion.identity);

        oBellTowerMdlClmns_V.transform.parent = oChurch.transform; //parenting

        oBellTowerMdlClmns_V.name = "Bell Tower middle vertical column " + (++c);

        oBellTowerMdlClmns_V.renderer.material.color = Color.white;

        oBellTowerMdlClmns_V.transform.localScale.x = 0.03294696;

        oBellTowerMdlClmns_V.transform.localScale.y = 0.1176622;

        oBellTowerMdlClmns_V.transform.localScale.z = 0.05811055;

        aCollectionObj_6.Add(oBellTowerMdlClmns_V);

    }//end if

    BTposZ += 2.8154;

}//end for

c = 0;

//::::::::::::end middle columns::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::


//::::::::::::start rails::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

BTposX = this.RandX + 10.5047;

BTposY = 23.5;
```

```
BTposZ = this.RandZ + 5.3276;


for(i=0; i<=19;i++){
    if(oCube){
        var oBellTowerRail_H = Instantiate(oCube, Vector3(BTposX, BTposY, BTposZ), Quaternion.identity);
        oBellTowerRail_H.transform.parent = oChurch.transform; //parenting
        oBellTowerRail_H.name = "Bell Tower horizontal rail " + (++c);
        oBellTowerRail_H.renderer.material.color = Color.black;
        oBellTowerRail_H.transform.localScale.x = 0.005183983;
        oBellTowerRail_H.transform.localScale.y = 0.1046447;
        oBellTowerRail_H.transform.localScale.z = 0.01092465;
        aCollectionObj_6.Add(oBellTowerRail_H);
    }//end if
BTposX -= 0.217;
}//end for



BTposX = this.RandX + 10.5047;
BTposZ = this.RandZ + 1.71639;
for(i=0; i<=19;i++){
    if(oCube){
        var oBellTowerRail_H_ = Instantiate(oCube, Vector3(BTposX, BTposY, BTposZ), Quaternion.identity);
        oBellTowerRail_H_.transform.parent = oChurch.transform; //parenting
        oBellTowerRail_H_.name = "Bell Tower horizontal rail " + (++c);
```

```
                oBellTowerRail_H_.renderer.material.color = Color.black;
                oBellTowerRail_H_.transform.localScale.x = 0.005183983;
                oBellTowerRail_H_.transform.localScale.y = 0.1046447;
                oBellTowerRail_H_.transform.localScale.z = 0.01092465;
                aCollectionObj_6.Add(oBellTowerRail_H_);
            }//end if
            BTposX -= 0.217;
        }//end for
        c = 0;

        BTposX = this.RandX + 10.5047;
        BTposZ = this.RandZ + 5.3276;
        for(i=0; i<=16; i++){
            if(oCube){
                var oBellTowerRail_V = Instantiate(oCube, Vector3(BTposX, BTposY, BTposZ), Quaternion.identity);
                oBellTowerRail_V.transform.parent = oChurch.transform; //parenting
                oBellTowerRail_V.name = "Bell Tower vertical rail " + (++c);
                oBellTowerRail_V.renderer.material.color = Color.black;
                oBellTowerRail_V.transform.localScale.x = 0.005183983;
                oBellTowerRail_V.transform.localScale.y = 0.1046447;
                oBellTowerRail_V.transform.localScale.z = 0.01092465;
                aCollectionObj_6.Add(oBellTowerRail_V);
            }//end if
            BTposZ -= 0.217;
        }//end for
```

```
BTposX = this.RandX + 6.3817;
BTposZ = this.RandZ + 5.3276;
for(i=0; i<=16; i++){
    if(oCube){
        var oBellTowerRail_V_ = Instantiate(oCube, Vector3(BTposX, BTposY, BTposZ), Quaternion.identity);
        oBellTowerRail_V_.transform.parent = oChurch.transform; //parenting
        oBellTowerRail_V_.name = "Bell Tower vertical rail " + (++c);
        oBellTowerRail_V_.renderer.material.color = Color.black;
        oBellTowerRail_V_.transform.localScale.x = 0.005183983;
        oBellTowerRail_V_.transform.localScale.y = 0.1046447;
        oBellTowerRail_V_.transform.localScale.z = 0.01092465;
        aCollectionObj_6.Add(oBellTowerRail_V_);
    }//end if
    BTposZ -= 0.217;
}//end for
c = 0;

BTposX = this.RandX + 8.4368;
BTposY = 23.92;
BTposZ = this.RandZ + 5.3288;
for(i=0; i<=1; i++){
    for(j=0; j<=1; j++){
        if(oCube){
            var oBellTowerMainRail_H= Instantiate(oCube, Vector3(BTposX, BTposY, BTposZ), Quaternion.identity);
            oBellTowerMainRail_H.transform.parent = oChurch.transform; //parenting
```

```
                        oBellTowerMainRail_H.name = "Bell Tower main horizontal rail " + (++c);
                        oBellTowerMainRail_H.renderer.material.color = Color.black;
                        oBellTowerMainRail_H.transform.localScale.x = 0.005183983;
                        oBellTowerMainRail_H.transform.localScale.y = 0.3257105;
                        oBellTowerMainRail_H.transform.localScale.z = 0.01092465;
                        oBellTowerMainRail_H.transform.eulerAngles.z = 90;
                        aCollectionObj_6.Add(oBellTowerMainRail_H);
                    }//end if
                    BTposY -= 0.82;
                }//end for
                BTposY = 23.92;
                BTposZ -= 3.6107;
            }//end for
            c = 0;

            BTposX = this.RandX + 10.5047;
            BTposY = 23.92;
            BTposZ = this.RandZ + 3.51628;
            for(i=0; i<=1; i++){
                for(j=0; j<=1; j++){
                    if(oCube){
                        var oBellTowerMainRail_V= Instantiate(oCube, Vector3(BTposX, BTposY, BTposZ), Quaternion.identity);
                        oBellTowerMainRail_V.transform.parent = oChurch.transform; //parenting
```

```
                    oBellTowerMainRail_V.name = "Bell Tower main vertical rail " + (++c);
                    oBellTowerMainRail_V.renderer.material.color = Color.black;
                    oBellTowerMainRail_V.transform.localScale.x = 0.005183983;
                    oBellTowerMainRail_V.transform.localScale.y = 0.560019;
                    oBellTowerMainRail_V.transform.localScale.z = 0.01092465;
                    oBellTowerMainRail_V.transform.eulerAngles.x = 90;
                    aCollectionObj_6.Add(oBellTowerMainRail_V);
                }//end if
                BTposY -= 0.82;
            }//end for
            BTposY = 23.92;
            BTposX -= 4.1267;
        }//end for
        c = 0;
        //:::::::::::end rails::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

        //::::::::start cupola:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
        oCupola = GameObject.CreatePrimitive(PrimitiveType.Sphere);
        Destroy(oCupola);

        BTposX = this.RandX + 8.43;
        BTposY = 23.26;
        BTposZ = this.RandZ + 3.509899;
```

```
        for(i=0; i<=1; i++){
            if(oCupola){
                var oBellTowerMainRailCpl= Instantiate(oCupola, Vector3(BTposX, BTposY, BTposZ), Quaternion.identity);
                oBellTowerMainRailCpl.transform.parent = oChurch.transform; //parenting
                oBellTowerMainRailCpl.name = "Bell Tower cupola";
                oBellTowerMainRailCpl.renderer.material.color = Color(0.59607, 0.08235, 0.00392, 0);
                if( i==1 ){
                    oBellTowerMainRailCpl.transform.localScale.x = 0.1458287;
                    oBellTowerMainRailCpl.transform.localScale.y = 0.3024061;
                    oBellTowerMainRailCpl.transform.localScale.z = 0.2857603;
                }//end if
                else{
                    oBellTowerMainRailCpl.transform.localScale.x = 0.270826;
                    oBellTowerMainRailCpl.transform.localScale.y = 0.8189769;
                    oBellTowerMainRailCpl.transform.localScale.z = 0.4915887;
                }//end else
                aCollectionObj_6.Add(oBellTowerMainRailCpl);
            }//end if
            BTposY += 2.6;
        }//end for
        //::::::::end cupola::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
```

```
}//end function BellTower(RandX, RandZ)

//----------------------End BellTower-----------------------------------------------------------------------
--

//---------------Start Front Part-------------------------------------------------------------------------
---

public function FrontPrt(RandX, RandZ){

        oCube = GameObject.CreatePrimitive(PrimitiveType.Cube);

        Destroy(oCube);

        var oBack = Instantiate(oCube, Vector3(this.RandX + 4.50393, 2.4548,
this.RandZ + 4.15462), Quaternion.identity);

        oBack.transform.parent = oChurch.transform; //parenting

        oBack.name = "Back Part";

        oBack.renderer.material.color = Color(0.996, 0.9411, 0.8392, 0);

        oBack.transform.localScale.x = 0.300649;

        oBack.transform.localScale.y = 0.6070714;

        oBack.transform.localScale.z = 0.3997754;

        aCollectionObj_6.Add(oBack);

}// end function FrontPrt(RandX, RandZ)

//---------------End Front Part---------------------------------------------------------------------------
--

//-----------------Start Right Part-------------------------------------------------------------------------
---

public function RightPrt(RandX, RandZ){

        oCube = GameObject.CreatePrimitive(PrimitiveType.Cube);

        Destroy(oCube);
```

```
        var oRight = Instantiate(oCube, Vector3(this.RandX - 9.17795, 2.4548, this.RandZ + 0.012966), Quaternion.identity);

        oRight.transform.parent = oChurch.transform; //parenting

        oRight.name = "Right Part";

        oRight.renderer.material.color = Color(0.996, 0.9411, 0.8392, 0);

        oRight.transform.localScale.x = 0.4120472;

        oRight.transform.localScale.y = 0.6070714;

        oRight.transform.localScale.z = 0.9969666;

        aCollectionObj_6.Add(oRight);

}// end function RightPrt(RandX, RandZ)

//-----------------End Right Part-------------------------------------------------------------------------------


//--------------------------Start Windows-----------------------------------------------------------------------


public function Windows(RandX, RandZ){

        oWindow = GameObject.CreatePrimitive(PrimitiveType.Cube);


        //transparent windows

        //oWindow.transform.renderer.material.color = Color.clear;

        oWindow.renderer.material.mainTexture = Resources.Load("window_church1"); //import texture for windows


        var nWind_X:float = this.RandX + 5.8255;

        var nWind_Y:float = 4.0296;

        var nWind_Z:float = this.RandZ + 5.50;
```

```
//:::::Start Back Part Windows:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

oWindow.transform.position = Vector3(nWind_X, nWind_Y, nWind_Z);
oWindow.transform.parent = oChurch.transform; //parenting
oWindow.name = "Back Window 0";
oWindow.transform.localScale.x = 0.05189104;
oWindow.transform.localScale.y = 0.1531791;
oWindow.transform.localScale.z = 0.0163088;
aCollectionObj_6.Add(oWindow);


for(i=0; i<=1; i++){
    for(j=0; j<=1; j++){
        if(oWindow){
            var oBackWind = Instantiate(oWindow, Vector3(nWind_X - 1.3, nWind_Y, nWind_Z), Quaternion.identity);
            oBackWind.transform.parent = oChurch.transform; //parenting
            oBackWind.name = "Back Window " + (++c);
            oBackWind.transform.localScale.x = 0.05189104;
            oBackWind.transform.localScale.y = 0.1531791;
            oBackWind.transform.localScale.z = 0.0163088;
            nWind_Y -= 2;
            aCollectionObj_6.Add(oBackWind);
        }//end if
    }//end for
    nWind_Y = 4.0296;
    nWind_X -= 1.3;
}//end for
```

```
        c = 0;



        nWind_X = this.RandX + 2.6062;

        nWind_Y = 4.0296;

        nWind_Z = this.RandZ + 4.3227;


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

                if(oWindow){

                        var oBackSideWind = Instantiate(oWindow, Vector3(nWind_X,
nWind_Y, nWind_Z), Quaternion.identity);

                        oBackSideWind.transform.parent = oChurch.transform;
//parenting

                        oBackSideWind.name = "Back side Window " + (++c);

                        oBackSideWind.transform.localScale.x = 0.1261232;

                        oBackSideWind.transform.localScale.y = 0.1531791;

                        oBackSideWind.transform.localScale.z = 0.0163088;

                        oBackSideWind.transform.eulerAngles.y = 90;

                        nWind_Y -= 2;

                        aCollectionObj_6.Add(oBackSideWind);

                }//end if

        }//end for

        c = 0;

        //:::::End Back Part
Windows::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::


        //:::::Start Right Side
Windows::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

        nWind_X = this.RandX - 6.9751;
```

```
nWind_Y = 4.0296;
nWind_Z = this.RandZ + 3.3014;

for(i=0; i<=3; i++){
    if(oWindow){
        var oRghtSideWind = Instantiate(oWindow, Vector3(nWind_X, nWind_Y, nWind_Z), Quaternion.identity);
        oRghtSideWind.transform.parent = oChurch.transform; //parenting
        oRghtSideWind.name = "Right side back window " + (++c);
        oRghtSideWind.transform.localScale.x = 0.05189104;
        oRghtSideWind.transform.localScale.y = 0.1531791;
        oRghtSideWind.transform.localScale.z = 0.0163088;
        aCollectionObj_6.Add(oRghtSideWind);

        if(i ==1)
            nWind_X -= 1.8;
        else
            nWind_X -= 1.3;
    }//end if
}//end for
c = 0;

nWind_X = this.RandX - 8.2751;
if(oWindow){
    var oRghtSideDnWind = Instantiate(oWindow, Vector3(nWind_X, nWind_Y - 2, nWind_Z), Quaternion.identity);
    oRghtSideDnWind.transform.parent = oChurch.transform; //parenting
```

```
            oRghtSideDnWind.name = "Right side back window 5";
            oRghtSideDnWind.transform.localScale.x = 0.05189104;
            oRghtSideDnWind.transform.localScale.y = 0.1531791;
            oRghtSideDnWind.transform.localScale.z = 0.0163088;
            aCollectionObj_6.Add(oRghtSideDnWind);
        }//end if


        nWind_X = this.RandX - 6.9751;
        nWind_Y = 4.0296;
        nWind_Z = this.RandZ - 3.2817;

        for(i=0; i<=3; i++){
            if(oWindow){
                var oRghtSideFrontWind = Instantiate(oWindow,
Vector3(nWind_X, nWind_Y, nWind_Z), Quaternion.identity);
                oRghtSideFrontWind.transform.parent = oChurch.transform;
//parenting
                oRghtSideFrontWind.name = "Right side front window " +
(++c);
                oRghtSideFrontWind.transform.localScale.x = 0.05189104;
                oRghtSideFrontWind.transform.localScale.y = 0.1531791;
                oRghtSideFrontWind.transform.localScale.z = 0.0163088;
                aCollectionObj_6.Add(oRghtSideFrontWind);
                if(i ==1)
                    nWind_X -= 1.8;
                else
                    nWind_X -= 1.3;
```

```
            }//end if
        }//end for
        c = 0;

        nWind_X = this.RandX - 8.2751;
        if(oWindow){
            var oRghtSideDnFrontWind = Instantiate(oWindow, Vector3(nWind_X, nWind_Y - 2, nWind_Z), Quaternion.identity);
            oRghtSideDnFrontWind.transform.parent = oChurch.transform; //parenting
            oRghtSideDnFrontWind.name = "Right side front window 5";
            oRghtSideDnFrontWind.transform.localScale.x = 0.05189104;
            oRghtSideDnFrontWind.transform.localScale.y = 0.1531791;
            oRghtSideDnFrontWind.transform.localScale.z = 0.0163088;
            aCollectionObj_6.Add(oRghtSideDnFrontWind);
        }//end if

        //:::::End Right Side Windows::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

}//end function Windows(RandX, RandZ)

//--------------------------End  Windows-----------------------------------------------------------------

//----------------------------Start Doors-----------------------------------------------------------------

public function Doors(RandX, RandZ){
```

```
        oWindow = GameObject.CreatePrimitive(PrimitiveType.Cube);
        Destroy(oWindow);
        //oWindow.transform.renderer.material.color = Color.clear;
        oWindow.renderer.material.mainTexture = Resources.Load("door_corfu2");
//import texture for door

        if(oWindow){
                var oBckPrtDoor = Instantiate(oWindow, Vector3(this.RandX + 5.8255, 1.503612, this.RandZ + 5.48), Quaternion.identity);
                oBckPrtDoor.name = "Back part door";
                oBckPrtDoor.transform.parent = oChurch.transform;
                oBckPrtDoor.transform.localScale.x = 0.0600062;
                oBckPrtDoor.transform.localScale.y = 0.291657;
                oBckPrtDoor.transform.localScale.z = 0.019;
                aCollectionObj_6.Add(oBckPrtDoor);
        }//end if

        if(oWindow){
                var oRightPrtBackDoor = Instantiate(oWindow, Vector3(this.RandX - 10.7118, 1.253612, this.RandZ + 3.3214), Quaternion.identity);
                oRightPrtBackDoor.name = "Right part back door";
                oRightPrtBackDoor.transform.parent = oChurch.transform;
                oRightPrtBackDoor.transform.localScale.x = 0.08101162;
                oRightPrtBackDoor.transform.localScale.y = 0.291657;
                oRightPrtBackDoor.transform.localScale.z = 0.019;
                aCollectionObj_6.Add(oRightPrtBackDoor);
        }//end if
```

```
        if(oWindow){
                var oRightPrtFrontDoor = Instantiate(oWindow, Vector3(this.RandX - 10.7118, 1.253612, this.RandZ - 3.3017), Quaternion.identity);
                oRightPrtFrontDoor.name = "Right part front door";
                oRightPrtFrontDoor.transform.parent = oChurch.transform;
                oRightPrtFrontDoor.transform.localScale.x = 0.08101162;
                oRightPrtFrontDoor.transform.localScale.y = 0.291657;
                oRightPrtFrontDoor.transform.localScale.z = 0.019;
                aCollectionObj_6.Add(oRightPrtFrontDoor);
        }//end if


oCube = GameObject.CreatePrimitive(PrimitiveType.Cube);
Destroy(oCube);
 //Color(0.9490, 0.8901, 0.7686, 0);


var posX: float = this.RandX - 1.28;
var posY: float = 3.2;
var posZ: float = this.RandZ + 3.3;
if(oCube){
        var oDoorEaves = Instantiate(oCube, Vector3(posX, posY, posZ), Quaternion.identity);
        oDoorEaves.transform.parent = oChurch.transform;
        oDoorEaves.name = "door eaves";
        oDoorEaves.renderer.material.mainTexture = Resources.Load("DoorEaves");
        oDoorEaves.transform.localScale.x = 0.1762082;
        oDoorEaves.transform.localScale.y = 0.04129031;
        oDoorEaves.transform.localScale.z = 0.121307;
```

```
        aCollectionObj_6.Add(oDoorEaves);
}//end if


posX = this.RandX - 0.28;
posY = 3.01;
posZ = this.RandZ + 3.5;
for(i=0; i<=1; i++){
        for(j=0; j<=1; j++){
                if(oCube){
                        var oDoorUpprClmn = Instantiate(oCube, Vector3(posX, posY, posZ), Quaternion.identity);
                        oDoorUpprClmn.transform.parent = oDoorEaves.transform;
                        oDoorUpprClmn.name = "door upper column part " + (++c);
                        oDoorUpprClmn.renderer.material.color = Color(0.4588, 0.3647, 0.2431, 0);
                        if(j==0){
                                oDoorUpprClmn.transform.localScale.x = 0.1762082;
                                oDoorUpprClmn.transform.localScale.y = 0.170057;
                                oDoorUpprClmn.transform.localScale.z = 0.7051173;
                        }//end if
                        else{
                                oDoorUpprClmn.transform.localScale.x = 0.154843;
                                oDoorUpprClmn.transform.localScale.y = 0.2416804;
                                oDoorUpprClmn.transform.localScale.z = 0.581718;
                        }//else
                        aCollectionObj_6.Add(oDoorUpprClmn);
                }//end if
                posY -= 0.05;
```

```
        }//end for
        posY = 3.01;
        posX -= 2;
}//end for
c = 0;

oCylinder = GameObject.CreatePrimitive(PrimitiveType.Cylinder);
Destroy(oCylinder);

posX = this.RandX - 0.28;
posY = 2.90;
for(i=0; i<=1; i++){
        for(j=0; j<=1; j++){
                if(oCylinder){
                        var oDoorUpprClmnCldr = Instantiate(oCylinder, Vector3(posX, posY, posZ), Quaternion.identity);
                        if(j==0){
                                oDoorUpprClmnCldr.name = "door upper column part cylinder " + (++c);
                                oDoorUpprClmnCldr.renderer.material.color = Color(0.9490, 0.8901, 0.7686, 0);
                                oDoorUpprClmnCldr.transform.parent = oDoorEaves.transform;
                                oDoorUpprClmnCldr.transform.localScale.x = 0.1329714;
                                oDoorUpprClmnCldr.transform.localScale.y = 0.1246673;
                                oDoorUpprClmnCldr.transform.localScale.z = 0.3851586;
                        }//end if
```

```
                else{
                        oDoorUpprClmnCldr.name = "door column " + (++c);
                        oDoorUpprClmnCldr.renderer.material.color =
Color.white;
                        oDoorUpprClmnCldr.transform.parent =
oDoorEaves.transform;
                        oDoorUpprClmnCldr.transform.localScale.x =
0.120656;
                        oDoorUpprClmnCldr.transform.localScale.y =
3.634116;
                        oDoorUpprClmnCldr.transform.localScale.z =
0.3494862;
                }//else
                aCollectionObj_6.Add(oDoorUpprClmnCldr);
        }//end if
        posY -= 1.18;
    }//end for
    posY = 2.90;
    posX -= 2;
}//end for
c = 0;

posX = this.RandX - 0.28;
posY = 0.52;
posZ = this.RandZ + 3.5;

for(i=0; i<=1; i++){
    for(j=0; j<=2; j++){
        if(oCube){
```

```
                    var oDoorlwrClmn = Instantiate(oCube, Vector3(posX, posY, posZ), Quaternion.identity);
                    oDoorlwrClmn.transform.parent = oDoorEaves.transform;
                    oDoorlwrClmn.name = "door lower column part " + (++c);
                    oDoorlwrClmn.renderer.material.color = Color(0.4588, 0.3647, 0.2431, 0);
                    if(j==0){
                        oDoorlwrClmn.transform.localScale.x = 0.154843;
                        oDoorlwrClmn.transform.localScale.y = 0.3658382;
                        oDoorlwrClmn.transform.localScale.z = 0.581718;
                    }//end if
                    if(j==1){
                        oDoorlwrClmn.transform.localScale.x = 0.2026358;
                        oDoorlwrClmn.transform.localScale.y = 0.4652826;
                        oDoorlwrClmn.transform.localScale.z = 0.7051173;
                    }//end if
                    else if(j==2){
                        oDoorlwrClmn.transform.localScale.x = 0.2489888;
                        oDoorlwrClmn.transform.localScale.y = 0.355633;
                        oDoorlwrClmn.transform.localScale.z = 0.7787634;
                    }//else if
                    aCollectionObj_6.Add(oDoorlwrClmn);
                }//end if
                posY -= 0.13;
            }//end for
            posY = 0.52;
            posX -= 2;
}//end for
```

```
//Create a door face clone at the back face of the church
posX = this.RandX - 1.28;
posY = 3.2;
posZ = this.RandZ - 3.3;

if(oDoorEaves){
	var ocloneDoorBck = Instantiate(oDoorEaves, Vector3(posX, posY, posZ), Quaternion.identity);
	ocloneDoorBck.transform.parent = oChurch.transform;
	ocloneDoorBck.name = "door back face";
	ocloneDoorBck.transform.localScale.x = 0.1762082;
	ocloneDoorBck.transform.localScale.y = 0.04129031;
	ocloneDoorBck.transform.localScale.z = 0.121307;
	ocloneDoorBck.transform.eulerAngles.y = 180;
	aCollectionObj_6.Add(ocloneDoorBck);
}//end if

}//end function Doors(RandX, RandZ)
//----------------------------End Doors-------------------------------------------------------------


//-------------------------------Start Stairs-----------------------------------------------------
public function Stairs(RandX, RandZ){

	oCube = GameObject.CreatePrimitive(PrimitiveType.Cube);
```

```
Destroy(oCube);


var stairsX:float = this.RandX + 5.8255;
var stairsY:float = 0.353612;
var stairsZ:float = this.RandZ + 5.5689;
for(i=0; i<=2; i++){
	if(oCube){
		var oFrntPrtStairs = Instantiate(oCube, Vector3(stairsX, stairsY, stairsZ), Quaternion.identity);
		oFrntPrtStairs.name = "back part stairs " + (++c);
		oFrntPrtStairs.transform.parent = oChurch.transform;
		oFrntPrtStairs.transform.localScale.x = 0.0600062;
		oFrntPrtStairs.transform.localScale.y = 0.01337241;
		if(i==1)
			oFrntPrtStairs.transform.localScale.z = 0.058875;
		else if(i==2)
			oFrntPrtStairs.transform.localScale.z = 0.08348349;
		else
			oFrntPrtStairs.transform.localScale.z = 0.03248049;
		aCollectionObj_6.Add(oFrntPrtStairs);
		stairsY -= 0.10;
		stairsZ += 0.087;
	}//end if
}//end for
c = 0;


stairsX = this.RandX - 1.2758;
```

```
stairsY = 0.253612;
stairsZ = this.RandZ + 3.4055;
for(i=0; i<=1; i++){
    if(oCube){
        var oMainBackStair = Instantiate(oCube, Vector3(stairsX, stairsY, stairsZ), Quaternion.identity);
        oMainBackStair.name = "main stairs " + (++c);
        oMainBackStair.transform.parent = oChurch.transform;
        oMainBackStair.transform.localScale.x = 0.1102614;
        oMainBackStair.transform.localScale.y = 0.01337241;
        oMainBackStair.transform.localScale.z = 0.1215557;
    }//end if
    stairsZ -= 6.779146;
}//end for
c = 0;

stairsX = this.RandX - 4.6078;
stairsY = 0.153612;
stairsZ = this.RandZ + 0.6055836;
if(oCube){
var oMainBackStairDn = Instantiate(oCube, Vector3(stairsX, stairsY, stairsZ), Quaternion.identity);
oMainBackStairDn.name = "main back stair down";
oMainBackStairDn.transform.parent = oChurch.transform;
oMainBackStairDn.transform.localScale.x = 1.126423;
oMainBackStairDn.transform.localScale.y = 0.01337241;
oMainBackStairDn.transform.localScale.z = 1.484616;
}//end if
```

```
}//end function Stairs(RandX, RandZ)
//--------------------------------End Stairs-----------------------------------------------------------------


//--------------------------------Start Eaves----------------------------------------------------------------
public function Eaves(RandX, RandZ){

	oCube = GameObject.CreatePrimitive(PrimitiveType.Cube);
	Destroy(oCube);

	var eavesX:float = this.RandX + 5.8255;
	var eavesY:float = 2.6296;
	var eavesZ:float = this.RandZ + 5.52;

	for(i=0; i<=2; i++){
		if(oCube){
			var oBckPrtEaves = Instantiate(oCube, Vector3(eavesX, eavesY, eavesZ), Quaternion.identity);
			oBckPrtEaves.name = "back part eaves " + (++c);
			oBckPrtEaves.transform.parent = oChurch.transform;
			if(i>0){
				oBckPrtEaves.transform.localScale.x = 0.05501559;
				oBckPrtEaves.transform.localScale.y = 0.01000328;
			}
			else{
				oBckPrtEaves.transform.localScale.x = 0.06241104;
```

```
                    oBckPrtEaves.transform.localScale.y = 0.01000328;
                }
                oBckPrtEaves.transform.localScale.z = 0.04600906;
                aCollectionObj_6.Add(oBckPrtEaves);
                eavesX -= 1.3;
            }//end if
        }//end for
        c = 0;

        eavesX = this.RandX + 2.4730;
        eavesY = 4.6296;
        eavesZ = this.RandZ + 4.3227;

        for(i=0; i<=1; i++){
            if(oCube){
                var oBckPrtSideEaves = Instantiate(oCube, Vector3(eavesX,
eavesY, eavesZ), Quaternion.identity);
                oBckPrtSideEaves.name = "Back part side eave " + (++c);
                oBckPrtSideEaves.transform.parent = oChurch.transform;
                oBckPrtSideEaves.transform.localScale.x = 0.130525;
                oBckPrtSideEaves.transform.localScale.y = 0.01000328;
                oBckPrtSideEaves.transform.localScale.z = 0.01852396;
                oBckPrtSideEaves.transform.eulerAngles.y = 90;
                aCollectionObj_6.Add(oBckPrtSideEaves);
                eavesY -= 2;
            }//end if
        }//end for
```

```
c = 0;



eavesX = this.RandX - 10.7118;

eavesY = 2.453612;

eavesZ = this.RandZ + 3.3821;


if(oCube){

	var oBckPrtRghSideEave = Instantiate(oCube, Vector3(eavesX, eavesY, eavesZ), Quaternion.identity);

	oBckPrtRghSideEave.name = "Back part right side eave";

	oBckPrtRghSideEave.transform.parent = oChurch.transform;

	oBckPrtRghSideEave.transform.localScale.x = 0.08191628;

	oBckPrtRghSideEave.transform.localScale.y = 0.01000328;

	oBckPrtRghSideEave.transform.localScale.z = 0.04600906;

	aCollectionObj_6.Add(oBckPrtRghSideEave);

}//end if


eavesZ = this.RandZ - 3.33347;


if(oCube){

	var oFrntPrtRghSideEave = Instantiate(oCube, Vector3(eavesX, eavesY, eavesZ), Quaternion.identity);

	oFrntPrtRghSideEave.name = "Front part right side eave";

	oFrntPrtRghSideEave.transform.parent = oChurch.transform;

	oFrntPrtRghSideEave.transform.localScale.x = 0.08191628;

	oFrntPrtRghSideEave.transform.localScale.y = 0.01000328;

	oFrntPrtRghSideEave.transform.localScale.z = 0.04600906;
```

```
            aCollectionObj_6.Add(oFrntPrtRghSideEave);
        }//end if

        eavesX = this.RandX;
        eavesY = 7.6;
        eavesZ = this.RandZ;

        for(i=0; i<=2; i++){
            if(oCube){
                var oMainChurchEave = Instantiate(oCube, Vector3(eavesX,
eavesY, eavesZ), Quaternion.identity);
                oMainChurchEave.name = "main Church eave " + (++c);
                oMainChurchEave.transform.parent = oChurch.transform;
                if( i==1 ){
                    oMainChurchEave.transform.localScale.x = 1.018288;
                    oMainChurchEave.transform.localScale.z = 1.023838;
                }//end if
                else if( i==2){
                    oMainChurchEave.transform.localScale.x = 1.005431;
                    oMainChurchEave.transform.localScale.z = 1.011043;
                }//end else if
                else{
                    oMainChurchEave.transform.localScale.x = 1.031216;
                    oMainChurchEave.transform.localScale.z = 1.036801;
                }//end else
                oMainChurchEave.transform.localScale.y = 0.02880077;
                aCollectionObj_6.Add(oMainChurchEave);
```

```
            }//end if
            eavesY -= 0.20;
        }//end for
        c = 0;

}//end function Eaves(RandX, RandZ)
//--------------------------------End Eaves-----------------------------------------------------------


//----------------------------Start Roofs-----------------------------------------------------------
//::::::Start Main Roof::::::::::::::::::::::::::::::::::::::::::::::::::::::::
public function MainRoof(RandX, RandZ){
        var tmpobj: GameObject = new GameObject();
        var mf: MeshFilter = tmpobj.AddComponent(MeshFilter);

//      tmpobj.AddComponent(MeshFilter);
        tmpobj.AddComponent(MeshRenderer);
        tmpobj.name = "Main Roof";
        tmpobj.transform.parent = oChurch.transform;

        var verts: Vector3[] = new Vector3[6];
        var normals: Vector3[] = new Vector3[6];
        var uv: Vector2[] = new Vector2[6];
        var tri: int[] = new int[18]; //3 vertices * 6 triangles =18


        //vertices positioning
```

```
verts[0] = new Vector3(this.RandX - 9, 7.8, this.RandZ + 3.15); //P1
verts[1] = new Vector3(this.RandX - 9, 9.1, this.RandZ + 6.575); //P2
verts[2] = new Vector3(this.RandX - 9, 7.8, this.RandZ + 10); //P3

verts[3] = new Vector3(this.RandX + 4, 7.8, this.RandZ + 10); //P4
verts[4] = new Vector3(this.RandX + 4, 9.1, this.RandZ + 6.575); //P5
verts[5] = new Vector3(this.RandX + 4, 7.8, this.RandZ + 3.15); //P6



uv[0] = new Vector2(1, 1);
uv[1] = new Vector2(0, 0.5);
uv[2] = new Vector2(1, 0);

uv[3] = new Vector2(0, 0);
uv[4] = new Vector2(1, 0.5);
uv[5] = new Vector2(0, 1);



//triangles
tri[0] = 1;
tri[1] = 0;
tri[2] = 2;

tri[3] = 1;
tri[4] = 2;
tri[5] = 3;
```

```
        tri[6] = 1;
        tri[7] = 3;
        tri[8] = 4;

        tri[9] = 4;
        tri[10] = 3;
        tri[11] = 5;

        tri[12] = 4;
        tri[13] = 5;
        tri[14] = 1;

        tri[15] = 1;
        tri[16] = 5;
        tri[17] = 0;

        var mesh: Mesh = new Mesh();
        mesh.name = "Main Roof";
        mesh.vertices = verts;
        mesh.triangles = tri;
        mesh.uv = uv;
        mesh.normals = normals;
        mesh.RecalculateNormals();
        mf.mesh = mesh;
        tmpobj.renderer.material.mainTexture                    =
Resources.Load("tiled_roof_red_grey"); //import texture
}//end function MainRoof(RandX, RandZ)
```

```
//::::::end Main Roof::::::::::::::::::::::::::::::::::::::::::::::::::::::::

//::::::start Right Roof::::::::::::::::::::::::::::::::::::::::::::::::::::::
public function RightRoof(RandX, RandZ){
        var tmpobj: GameObject = new GameObject();
        var mf: MeshFilter = tmpobj.AddComponent(MeshFilter);

//      tmpobj.AddComponent(MeshFilter);
        tmpobj.AddComponent(MeshRenderer);
        tmpobj.name = "Right Roof";
        tmpobj.transform.parent = oChurch.transform;

        var verts: Vector3[] = new Vector3[6];
        var normals: Vector3[] = new Vector3[6];
        var uv: Vector2[] = new Vector2[6];
        var tri: int[] = new int[18]; //3 vertices * 6 triangles =18


        //vertices positioning
        verts[0] = new Vector3(this.RandX - 14.5502, 4.8, this.RandZ + 3.15); //P1
        verts[1] = new Vector3(this.RandX - 14.5502, 6.1, this.RandZ + 6.575); //P2
        verts[2] = new Vector3(this.RandX - 14.5502, 4.8, this.RandZ + 10); //P3

        verts[3] = new Vector3(this.RandX - 8.9751, 4.8, this.RandZ + 10); //P4
        verts[4] = new Vector3(this.RandX - 8.9751, 6.1, this.RandZ + 6.575); //P5
        verts[5] = new Vector3(this.RandX - 8.9751, 4.8, this.RandZ + 3.15); //P6
```

```
uv[0] = new Vector2(1, 1);
uv[1] = new Vector2(0, 0.5);
uv[2] = new Vector2(1, 0);

uv[3] = new Vector2(0, 0);
uv[4] = new Vector2(1, 0.5);
uv[5] = new Vector2(0, 1);



//triangles
tri[0] = 1;
tri[1] = 0;
tri[2] = 2;

tri[3] = 1;
tri[4] = 2;
tri[5] = 3;

tri[6] = 1;
tri[7] = 3;
tri[8] = 4;

tri[9] = 4;
tri[10] = 3;
tri[11] = 5;
```

```
        tri[12] = 4;
        tri[13] = 5;
        tri[14] = 1;

        tri[15] = 1;
        tri[16] = 5;
        tri[17] = 0;

        var mesh: Mesh = new Mesh();
        mesh.name = "Right Roof";
        mesh.vertices = verts;
        mesh.triangles = tri;
        mesh.uv = uv;
        mesh.normals = normals;
        mesh.RecalculateNormals();
        mf.mesh = mesh;
        tmpobj.renderer.material.mainTexture =
Resources.Load("tiled_roof_red_grey"); //import texture
}//end function RightRoof(RandX, RandZ)
//::::::end Right Roof::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::


//:::::::::::::::Start Back Roof::::::::::::::::::::::::::::::::::::::::::::::::
public function BackRoof(RandX, RandZ){
        var tmpobj: GameObject = new GameObject();
        var mf: MeshFilter = tmpobj.AddComponent(MeshFilter);
```

```
//      tmpobj.AddComponent(MeshFilter);
        tmpobj.AddComponent(MeshRenderer);
        tmpobj.name = "Back Roof";
        tmpobj.transform.parent = oChurch.transform;

        var verts: Vector3[] = new Vector3[6];
        var normals: Vector3[] = new Vector3[6];
        var uv: Vector2[] = new Vector2[6];
        var tri: int[] = new int[12]; //3 vertices * 4 triangles


        //vertices positioning
        verts[0] = new Vector3(this.RandX - 0.05, 6.1, this.RandZ + 9.85); //P1
        verts[1] = new Vector3(this.RandX - 0.05, 4.8, this.RandZ + 9.85); //P2
        verts[2] = new Vector3(this.RandX - 0.05, 4.8, this.RandZ + 12.2); //P3

        verts[3] = new Vector3(this.RandX + 3.85, 4.8, this.RandZ + 12.2); //P4
        verts[4] = new Vector3(this.RandX + 3.85, 4.8, this.RandZ + 9.85); //P5
        verts[5] = new Vector3(this.RandX + 3.85, 6.1, this.RandZ + 9.85); //P6


        uv[0] = new Vector2(1, 1);
        uv[1] = new Vector2(0, 0.5);
        uv[2] = new Vector2(1, 0);

        uv[3] = new Vector2(0, 0);
        uv[4] = new Vector2(1, 0.5);
```

```
uv[5] = new Vector2(0, 1);



//triangles
tri[0] = 0;
tri[1] = 1;
tri[2] = 2;

tri[3] = 0;
tri[4] = 2;
tri[5] = 3;

tri[6] = 0;
tri[7] = 3;
tri[8] = 5;

tri[9] = 3;
tri[10] = 4;
tri[11] = 5;

var mesh: Mesh = new Mesh();
mesh.name = "Back Roof";
mesh.vertices = verts;
mesh.triangles = tri;
mesh.uv = uv;
mesh.normals = normals;
mesh.RecalculateNormals();
```

```
        mf.mesh = mesh;
        tmpobj.renderer.material.mainTexture                                    =
Resources.Load("tiled_roof_red_grey"); //import texture
}//end function RightRoof(RandX, RandZ)

//:::::::::::::end Back Roof::::::::::::::::::::::::::::::::::::::::::::::::::::::::

//-----------------------------End Roofs-------------------------------------------------------------------


//------------------------------------- Start Main Windows-----------------------------------------
-----
public function MainWind(RandX, RandZ){
//Arch Windows Upper eaves
        oCube = GameObject.CreatePrimitive(PrimitiveType.Cube);
        Destroy(oCube);
        oCube.renderer.material.color = Color(0.9490, 0.8901, 0.7686, 0);

        var oUpperEave_ = GameObject.CreatePrimitive(PrimitiveType.Cube);
        oUpperEave_.renderer.material.color = Color(0.9490, 0.8901, 0.7686, 0);
        oUpperEave_.transform.position = Vector3(this.RandX + 1.31, 6.46,
this.RandZ + 3.3);
        oUpperEave_.transform.parent = oChurch.transform;
        oUpperEave_.name = "Upper window eave 0";
        oUpperEave_.transform.localScale.x = 0.1257709;
        oUpperEave_.transform.localScale.y = 0.02279421;
        oUpperEave_.transform.localScale.z = 0.06467526;
        aCollectionObj_6.Add(oUpperEave_);
```

```
var posX:float = this.RandX + 1.31;
var posY:float = 6.64;
var posZ:float = this.RandZ + 3.3;
if(oCube){
		var oUpperEave = Instantiate(oCube, Vector3(posX, posY, posZ), Quaternion.identity);
		oUpperEave.name = "upper window eave 2";
		oUpperEave.transform.parent = oUpperEave_.transform;
		oUpperEave.transform.localScale.x = 1.154025;
		oUpperEave.transform.localScale.y = 1.040318;
		oUpperEave.transform.localScale.z = 1.1768;
		aCollectionObj_6.Add(oUpperEave);
}//end if


//Arch Window Columns
posX = this.RandX + 2;
posY = 5.65;
posZ = this.RandZ + 3.3;

for(i=0; i<=1; i++){
	for(j=0; j<=1; j++){
		if(oCube){
			var oUpperPrtClmn = Instantiate(oCube, Vector3(posX, posY, posZ), Quaternion.identity);
			oUpperPrtClmn.name = "upper part column " + (++c);
```

```
                        oUpperPrtClmn.transform.parent = oUpperEave_.transform;
                        if(j==1){
                            oUpperPrtClmn.transform.localScale.x = 0.2343267;
                            oUpperPrtClmn.transform.localScale.z = 0.7816373;
                        }//end if
                        else{
                            oUpperPrtClmn.transform.localScale.x = 0.2915291;
                            oUpperPrtClmn.transform.localScale.z = 0.7816373;
                        }//end else
                        oUpperPrtClmn.transform.localScale.y = 1.188682;
                        aCollectionObj_6.Add(oUpperPrtClmn);
                    }//end if
                    posY -= 0.1;
                }//end for
                posY = 5.65;
                posX -= 1.35;
            }//end for
            c = 0;

            posX = this.RandX + 2;
            posY = 5.06;
            posZ = this.RandZ + 3.3;

            for(i=0; i<=1; i++){
```

```
            if(oCube){
                var oColumnWindArch = Instantiate(oCube, Vector3(posX, posY, posZ), Quaternion.identity);
                oColumnWindArch.name = "Arch window column " + (++c);
                oColumnWindArch.transform.parent = oUpperEave_.transform;
                oColumnWindArch.transform.localScale.x = 0.1794632;
                oColumnWindArch.transform.localScale.y = 5.113039;
                oColumnWindArch.transform.localScale.z = 0.587822;
                aCollectionObj_6.Add(oColumnWindArch);
            }//end if
            posX -= 1.35;
        }//end for
        c = 0;

        posX = this.RandX + 2;
        posY = 4.57;
        posZ = this.RandZ + 3.3;

        for(i=0; i<=1; i++){
            if(oCube){
                var oLowerPrtClmn = Instantiate(oCube, Vector3(posX, posY, posZ), Quaternion.identity);
                oLowerPrtClmn.name = "lower part column " + (++c);
                oLowerPrtClmn.transform.parent = oUpperEave_.transform;
                oLowerPrtClmn.transform.localScale.x = 0.2381323;
                oLowerPrtClmn.transform.localScale.y = 0.825889;
```

```
                    oLowerPrtClmn.transform.localScale.z = 0.7770827;
                    aCollectionObj_6.Add(oLowerPrtClmn);
            }//end if
            posX -= 1.35;
        }//end for
        c = 0;

        //Arch Windows Lower eaves
        posX = this.RandX + 1.33;
        posY = 4.36;
        posZ = this.RandZ + 3.3;
        for(i=0; i<=1; i++){
            if(oCube){
                var oLowerEave = Instantiate(oCube, Vector3(posX, posY, posZ), Quaternion.identity);
                oLowerEave.name = "lower window eave " + (++c);
                oLowerEave.transform.parent = oUpperEave_.transform;
                if(i==1){
                    oLowerEave.transform.localScale.x = 1.317234;
                    oLowerEave.transform.localScale.z = 0.9266781;
                }//end if
                else{
                    oLowerEave.transform.localScale.x = 1.122489;
                    oLowerEave.transform.localScale.z = 0.7780409;
                }//end else
                oLowerEave.transform.localScale.y = 0.483218;
                aCollectionObj_6.Add(oLowerEave);
```

```
            }//end if
            posY += 0.09;
        }//end for
        c = 0;

        //Create Clones (front side)
        posX = this.RandX + 1.31;
        posY = 6.46;
        for(i=0; i<=2; i++){
            if(oUpperEave_){
                var ocloneMainWind_frnt = Instantiate(oUpperEave_,
Vector3(posX, posY, posZ), Quaternion.identity);
                ocloneMainWind_frnt.name = "Upper window eave " + (++c);
                ocloneMainWind_frnt.transform.parent = oChurch.transform;
                ocloneMainWind_frnt.transform.localScale.x = 0.1257709;
                ocloneMainWind_frnt.transform.localScale.y = 0.02279421;
                ocloneMainWind_frnt.transform.localScale.z = 0.06467526;
                aCollectionObj_6.Add(ocloneMainWind_frnt);
                posX -= 2.6;
            }//end if
        }//end for
        c = 0;

        //Create Clones (back side)
        posX = this.RandX - 3.91;
        posY = 6.46;
        posZ = this.RandZ - 3.3;
```

```
	for(i=0; i<=2; i++){
		if(oUpperEave_){
			var ocloneMainWind_bck = Instantiate(oUpperEave_, Vector3(posX, posY, posZ), Quaternion.identity);
			ocloneMainWind_bck.name = "Upper window eave " + (++c);
			ocloneMainWind_bck.transform.parent = oChurch.transform;
			ocloneMainWind_bck.transform.localScale.x = 0.1257709;
			ocloneMainWind_bck.transform.localScale.y = 0.02279421;
			ocloneMainWind_bck.transform.localScale.z = 0.06467526;
			aCollectionObj_6.Add(ocloneMainWind_bck);
			posX += 2.6;
		}//end if
	}//end for
	c = 0;

}//end function MainWind(RandX, RandZ)
//------------------------------------ End Main Windows-----------------------------------------
--

//-----------------------------------Start Triangle Eaves-----------------------------------------
--
private function TriangleEaves(RandX, RandZ, tmpobj){
	tmpobj = new GameObject();
	var mf: MeshFilter = tmpobj.AddComponent(MeshFilter);

	tmpobj.AddComponent(MeshRenderer);
```

```
tmpobj.name = "Door Roof";
//tmpobj.renderer.material.color = Color(0.9490, 0.8901, 0.7686, 0);
tmpobj.transform.parent = oChurch.transform;

var verts: Vector3[] = new Vector3[6];
var uv: Vector2[] = new Vector2[6];
var normals: Vector3[] = new Vector3[6];
var tri: int[] = new int[24]; //( 3 vertices * 8 triangles )

//vertices positioning
verts[0] = new Vector3(this.RandX + 0.02, 3.36, this.RandZ + 3.66); //P1
verts[1] = new Vector3(this.RandX + 0.02, 3.36, this.RandZ + 2.9); //P2
verts[2] = new Vector3(this.RandX - 1.28, 3.7, this.RandZ + 3.66); //P3
verts[3] = new Vector3(this.RandX - 1.28, 3.7, this.RandZ + 2.9); //P4
verts[4] = new Vector3(this.RandX - 2.58, 3.36, this.RandZ + 2.9); //P5
verts[5] = new Vector3(this.RandX - 2.58, 3.36, this.RandZ + 3.66); //P6

//normals
normals[0] = new Vector3(0, 0, 1);
normals[1] = new Vector3(0, 0, 1);
normals[2] = new Vector3(0, 0, 1);
normals[3] = new Vector3(0, 0, 1);
normals[4] = new Vector3(0, 0, 1);
normals[5] = new Vector3(0, 0, 1);

//uv mapping (textures)
uv[0] = new Vector2(0, 0);
```

```
uv[1] = new Vector2(0, 1);

uv[2] = new Vector2(0.5, 0);

uv[3] = new Vector2(0.5, 1);

uv[4] = new Vector2(1, 1);

uv[5] = new Vector2(1, 0);


//triangles

tri[0] = 2;

tri[1] = 0;

tri[2] = 3;


tri[3] = 3;

tri[4] = 0;

tri[5] = 1;


tri[6] = 1;

tri[7] = 4;

tri[8] = 3;


tri[9] = 4;

tri[10] = 5;

tri[11] = 3;


tri[12] = 5;

tri[13] = 2;

tri[14] = 3;
```

```
        tri[15] = 2;
        tri[16] = 5;
        tri[17] = 0;

        tri[18] = 0;
        tri[19] = 4;
        tri[20] = 1;

        tri[21] = 4;
        tri[22] = 0;
        tri[23] = 5;

        var mesh: Mesh = new Mesh();
        mesh.name = "Door Roof";
        mesh.vertices = verts;
        mesh.triangles = tri;
        mesh.uv = uv;
        mesh.normals = normals;
        //mesh.RecalculateNormals();
        tmpobj.renderer.material.mainTexture = Resources.Load("door_Roof_2");
        mf.mesh = mesh;
}//end function TriangleEaves(RandX, RandZ, tmpobj)
//-----------------------------------End Triangle Eaves-----------------------------------------
-
```

```
//-------------------------Start Arch-------------------------------------------------------------------
private function Arch(RandX, RandZ, oArch) {
        oArch = new GameObject();
        var mf: MeshFilter = oArch.AddComponent(MeshFilter);

        oArch.AddComponent(MeshRenderer);
        oArch.name = "Arch";
        oArch.renderer.material.color = Color(0.9490, 0.8901, 0.7686, 1);
        oArch.transform.parent = oChurch.transform;

        var verts: Vector3[] = new Vector3[12];
        var uv: Vector2[] = new Vector2[12];
        var normals: Vector3[] = new Vector3[12];
        var tri: int[] = new int[60]; //( 3 vertices * 10 triangles ) * 2 (two face arch mesh)

        //vertices positioning
        verts[0] = new Vector3(this.RandX + 1.82, 5.95, this.RandZ + 3.3); //P1
        verts[1] = new Vector3(this.RandX + 1.82, 5.65, this.RandZ + 3.3); //P2
        verts[2] = new Vector3(this.RandX + 2.12, 5.65, this.RandZ + 3.3); //P3
        verts[3] = new Vector3(this.RandX + 2.12, 6.45, this.RandZ + 3.3); //P4
        verts[4] = new Vector3(this.RandX + 0.52, 6.45, this.RandZ + 3.3); //P5
        verts[5] = new Vector3(this.RandX + 0.52, 5.65, this.RandZ + 3.3); //P6
        verts[6] = new Vector3(this.RandX + 0.82, 5.65, this.RandZ + 3.3); //P7
        verts[7] = new Vector3(this.RandX + 0.82, 5.95, this.RandZ + 3.3); //P8
        verts[8] = new Vector3(this.RandX + 0.92, 6.1, this.RandZ + 3.3); //P9
        verts[9] = new Vector3(this.RandX + 1.12, 6.2, this.RandZ + 3.3); //P10
```

```
verts[10] = new Vector3(this.RandX + 1.52, 6.2, this.RandZ + 3.3); //P11
verts[11] = new Vector3(this.RandX + 1.72, 6.1, this.RandZ + 3.3); //P12

//normals
normals[0] = new Vector3(1, 0, 1);
normals[1] = new Vector3(1, 0, 1);
normals[2] = new Vector3(1, 0, 1);
normals[3] = new Vector3(1, 0, 1);
normals[4] = new Vector3(1, 0, 1);
normals[5] = new Vector3(1, 0, 1);
normals[6] = new Vector3(1, 0, 1);
normals[7] = new Vector3(1, 0, 1);
normals[8] = new Vector3(1, 0, 1);
normals[9] = new Vector3(1, 0, 1);
normals[10] = new Vector3(1, 0, 1);
normals[11] = new Vector3(1, 0, 1);

//triangles
tri[0] = 0;
tri[1] = 1;
tri[2] = 2;

tri[3] = 0;
tri[4] = 2;
tri[5] = 3;

tri[6] = 0;
```

```
tri[7] = 3;
tri[8] = 11;

tri[9] = 11;
tri[10] = 3;
tri[11] = 10;

tri[12] = 10;
tri[13] = 3;
tri[14] = 4;

tri[15] = 10;
tri[16] = 4;
tri[17] = 9;

tri[18] = 9;
tri[19] = 4;
tri[20] = 8;

tri[21] = 8;
tri[22] = 4;
tri[23] = 7;

tri[24] = 7;
tri[25] = 4;
tri[26] = 5;
```

```
tri[27] = 7;
tri[28] = 5;
tri[29] = 6;

tri[30] = 0;
tri[31] = 2;
tri[32] = 1;

tri[33] = 0;
tri[34] = 3;
tri[35] = 2;

tri[36] = 11;
tri[37] = 3;
tri[38] = 0;

tri[39] = 10;
tri[40] = 3;
tri[41] = 11;

tri[42] = 10;
tri[43] = 4;
tri[44] = 3;

tri[45] = 9;
tri[46] = 4;
tri[47] = 10;
```

```
        tri[48] = 8;
        tri[49] = 4;
        tri[50] = 9;

        tri[51] = 7;
        tri[52] = 4;
        tri[53] = 8;

        tri[54] = 7;
        tri[55] = 5;
        tri[56] = 4;

        tri[57] = 7;
        tri[58] = 6;
        tri[59] = 5;

        var mesh: Mesh = new Mesh();
        mesh.name = "Arch";
        mesh.vertices = verts;
        mesh.triangles = tri;
        mesh.uv = uv;
        mesh.normals = normals;
        //mesh.RecalculateNormals();
        mf.mesh = mesh;
}//end function Arch(Rand_X, Rand_Z)
//-------------------------End Arch-----------------------------------------------------------------
```

```
//-------------------------Start Bell Tower Arch-----------------------------------------------------------
private function Bell_Tower_Arch(RandX, RandZ, oArch) {
	oArch = new GameObject();
	var mf: MeshFilter = oArch.AddComponent(MeshFilter);

	oArch.AddComponent(MeshRenderer);
	oArch.name = "Bell Tower Arch";
	oArch.renderer.material.color = Color(0.9490, 0.8901, 0.7686, 0);
	oArch.transform.parent = oChurch.transform;

	var verts: Vector3[] = new Vector3[12];
	var uv: Vector2[] = new Vector2[12];
	var normals: Vector3[] = new Vector3[12];
	var tri: int[] = new int[60]; //( 3 vertices * 10 triangles ) * 2 (two face arch mesh)

	//vertices positioning
	verts[0] = new Vector3(this.RandX + 15.9, 19.55, this.RandZ + 9.3); //P1
	verts[1] = new Vector3(this.RandX + 15.9, 19.05, this.RandZ + 9.3); //P2
	verts[2] = new Vector3(this.RandX + 16.2, 19.05, this.RandZ + 9.3); //P3
	verts[3] = new Vector3(this.RandX + 16.2, 20.05, this.RandZ + 9.3); //P4
	verts[4] = new Vector3(this.RandX + 14.6, 20.05, this.RandZ + 9.3); //P5
	verts[5] = new Vector3(this.RandX + 14.6, 19.05, this.RandZ + 9.3); //P6
	verts[6] = new Vector3(this.RandX + 14.9, 19.05, this.RandZ + 9.3); //P7
	verts[7] = new Vector3(this.RandX + 14.9, 19.55, this.RandZ + 9.3); //P8
```

```
verts[8] = new Vector3(this.RandX + 15, 19.7, this.RandZ + 9.3); //P9
verts[9] = new Vector3(this.RandX + 15.2, 19.8, this.RandZ + 9.3); //P10
verts[10] = new Vector3(this.RandX + 15.6, 19.8, this.RandZ + 9.3); //P11
verts[11] = new Vector3(this.RandX + 15.8, 19.7, this.RandZ + 9.3); //P12

//normals
normals[0] = new Vector3(0, 0, 1);
normals[1] = new Vector3(0, 0, 1);
normals[2] = new Vector3(0, 0, 1);
normals[3] = new Vector3(0, 0, 1);
normals[4] = new Vector3(0, 0, 1);
normals[5] = new Vector3(0, 0, 1);
normals[6] = new Vector3(0, 0, 1);
normals[7] = new Vector3(0, 0, 1);
normals[8] = new Vector3(0, 0, 1);
normals[9] = new Vector3(0, 0, 1);
normals[10] = new Vector3(0, 0, 1);
normals[11] = new Vector3(0, 0, 1);

//triangles
tri[0] = 0;
tri[1] = 1;
tri[2] = 2;

tri[3] = 0;
tri[4] = 2;
tri[5] = 3;
```

```
tri[6] = 0;
tri[7] = 3;
tri[8] = 11;

tri[9] = 11;
tri[10] = 3;
tri[11] = 10;

tri[12] = 10;
tri[13] = 3;
tri[14] = 4;

tri[15] = 10;
tri[16] = 4;
tri[17] = 9;

tri[18] = 9;
tri[19] = 4;
tri[20] = 8;

tri[21] = 8;
tri[22] = 4;
tri[23] = 7;

tri[24] = 7;
tri[25] = 4;
```

```
tri[26] = 5;


tri[27] = 7;
tri[28] = 5;
tri[29] = 6;


tri[30] = 0;
tri[31] = 2;
tri[32] = 1;


tri[33] = 0;
tri[34] = 3;
tri[35] = 2;


tri[36] = 11;
tri[37] = 3;
tri[38] = 0;


tri[39] = 10;
tri[40] = 3;
tri[41] = 11;


tri[42] = 10;
tri[43] = 4;
tri[44] = 3;


tri[45] = 9;
```

```
tri[46] = 4;
tri[47] = 10;

tri[48] = 8;
tri[49] = 4;
tri[50] = 9;

tri[51] = 7;
tri[52] = 4;
tri[53] = 8;

tri[54] = 7;
tri[55] = 5;
tri[56] = 4;

tri[57] = 7;
tri[58] = 6;
tri[59] = 5;

var mesh: Mesh = new Mesh();
mesh.name = "Bell Tower Arch";
mesh.vertices = verts;
mesh.triangles = tri;
mesh.uv = uv;
mesh.normals = normals;
//mesh.RecalculateNormals();
mf.mesh = mesh;
```

```
}//end function Bell_Tower_Arch(Rand_X, Rand_Z)

//-------------------------End Bell Tower Arch-----------------------------------------------------------------



//-------------------------Start Bell Tower Arch-----------------------------------------------------------------
private function Bell_Tower_Vertical_Arch(RandX, RandZ, oArch) {
        oArch = new GameObject();
        var mf: MeshFilter = oArch.AddComponent(MeshFilter);

        oArch.AddComponent(MeshRenderer);
        oArch.name = "Bell Tower Vertical Arch";
        oArch.renderer.material.color = Color(0.9490, 0.8901, 0.7686, 0);
        oArch.transform.parent = oChurch.transform;

        var verts: Vector3[] = new Vector3[12];
        var uv: Vector2[] = new Vector2[12];
        var normals: Vector3[] = new Vector3[12];
        var tri: int[] = new int[60]; //( 3 vertices * 10 triangles ) * 2 (two face arch mesh)

        //vertices positioning
        verts[0] = new Vector3(this.RandX + 9.3, 19.55, this.RandZ + 15.9); //P1
        verts[1] = new Vector3(this.RandX + 9.3, 19.05, this.RandZ + 15.9); //P2
        verts[2] = new Vector3(this.RandX + 9.3, 19.05, this.RandZ + 16.2); //P3
        verts[3] = new Vector3(this.RandX + 9.3, 20.05, this.RandZ + 16.2); //P4
        verts[4] = new Vector3(this.RandX + 9.3, 20.05, this.RandZ + 14.6); //P5
```

```
verts[5] = new Vector3(this.RandX + 9.3, 19.05, this.RandZ + 14.6); //P6
verts[6] = new Vector3(this.RandX + 9.3, 19.05, this.RandZ + 14.9); //P7
verts[7] = new Vector3(this.RandX + 9.3, 19.55, this.RandZ + 14.9); //P8
verts[8] = new Vector3(this.RandX + 9.3, 19.7, this.RandZ + 15); //P9
verts[9] = new Vector3(this.RandX + 9.3, 19.8, this.RandZ + 15.2); //P10
verts[10] = new Vector3(this.RandX + 9.3, 19.8, this.RandZ + 15.6); //P11
verts[11] = new Vector3(this.RandX + 9.3, 19.7, this.RandZ + 15.8); //P12

//normals
normals[0] = new Vector3(0, 0, 1);
normals[1] = new Vector3(0, 0, 1);
normals[2] = new Vector3(0, 0, 1);
normals[3] = new Vector3(0, 0, 1);
normals[4] = new Vector3(0, 0, 1);
normals[5] = new Vector3(0, 0, 1);
normals[6] = new Vector3(0, 0, 1);
normals[7] = new Vector3(0, 0, 1);
normals[8] = new Vector3(0, 0, 1);
normals[9] = new Vector3(0, 0, 1);
normals[10] = new Vector3(0, 0, 1);
normals[11] = new Vector3(0, 0, 1);

//triangles
tri[0] = 0;
tri[1] = 1;
tri[2] = 2;
```

```
tri[3] = 0;

tri[4] = 2;

tri[5] = 3;


tri[6] = 0;

tri[7] = 3;

tri[8] = 11;


tri[9] = 11;

tri[10] = 3;

tri[11] = 10;


tri[12] = 10;

tri[13] = 3;

tri[14] = 4;


tri[15] = 10;

tri[16] = 4;

tri[17] = 9;


tri[18] = 9;

tri[19] = 4;

tri[20] = 8;


tri[21] = 8;

tri[22] = 4;

tri[23] = 7;
```

```
tri[24] = 7;
tri[25] = 4;
tri[26] = 5;

tri[27] = 7;
tri[28] = 5;
tri[29] = 6;

tri[30] = 0;
tri[31] = 2;
tri[32] = 1;

tri[33] = 0;
tri[34] = 3;
tri[35] = 2;

tri[36] = 11;
tri[37] = 3;
tri[38] = 0;

tri[39] = 10;
tri[40] = 3;
tri[41] = 11;

tri[42] = 10;
tri[43] = 4;
```

```
tri[44] = 3;

tri[45] = 9;
tri[46] = 4;
tri[47] = 10;

tri[48] = 8;
tri[49] = 4;
tri[50] = 9;

tri[51] = 7;
tri[52] = 4;
tri[53] = 8;

tri[54] = 7;
tri[55] = 5;
tri[56] = 4;

tri[57] = 7;
tri[58] = 6;
tri[59] = 5;

var mesh: Mesh = new Mesh();
mesh.name = "Bell Tower Vertical Arch";
mesh.vertices = verts;
mesh.triangles = tri;
mesh.uv = uv;
```

```
        mesh.normals = normals;
        //mesh.RecalculateNormals();
        mf.mesh = mesh;
}//end function Bell_Tower_Vertical_Arch(Rand_X, Rand_Z)
//-------------------------End Bell Tower Vertical Arch-----------------------------------------
----------------------



//--------------Start Arch Windows-------------------------------------------------------------
-
private function Arch_Wind(RandX, RandZ, tmpobj){
        tmpobj = new GameObject();
        var mf: MeshFilter = tmpobj.AddComponent(MeshFilter);

        tmpobj.AddComponent(MeshRenderer);
        tmpobj.name = "Arch Window";
        tmpobj.transform.parent = oChurch.transform;

        var verts: Vector3[] = new Vector3[11];
        var uv: Vector2[] = new Vector2[11];
        var normals: Vector3[] = new Vector3[11];
        var tri: int[] = new int[54]; //( 3 vertices * 9 triangles ) * 2 (for the back face)

        //vertices positioning
        verts[0] = new Vector3(this.RandX + 1.82, 4.5, this.RandZ + 3.3); //P1
        verts[1] = new Vector3(this.RandX + 1.82, 5.65, this.RandZ + 3.3); //P2
        verts[2] = new Vector3(this.RandX + 1.82, 5.95, this.RandZ + 3.3); //P3
```

```
verts[3] = new Vector3(this.RandX + 1.72, 6.1, this.RandZ + 3.3); //P4
verts[4] = new Vector3(this.RandX + 1.52, 6.2, this.RandZ + 3.3); //P5
verts[5] = new Vector3(this.RandX + 1.12, 6.2, this.RandZ + 3.3); //P6
verts[6] = new Vector3(this.RandX + 0.92, 6.1, this.RandZ + 3.3); //P7
verts[7] = new Vector3(this.RandX + 0.82, 5.95, this.RandZ + 3.3); //P8
verts[8] = new Vector3(this.RandX + 0.82, 5.65, this.RandZ + 3.3); //P9
verts[9] = new Vector3(this.RandX + 0.82, 4.5, this.RandZ + 3.3); //P10
verts[10] = new Vector3(this.RandX + 1.32, 5.65, this.RandZ + 3.3); //P11


//uv mapping (textures)
uv[0] = new Vector2(0, 0);
uv[1] = new Vector2(0, 0.6);
uv[2] = new Vector2(0.2, 0.8);
uv[3] = new Vector2(0.3, 0.9);
uv[4] = new Vector2(0.4, 0.85);
uv[5] = new Vector2(0.6, 0.85);
uv[6] = new Vector2(0.8, 0.9);
uv[7] = new Vector2(0.8, 0.8);
uv[8] = new Vector2(1, 0.6);
uv[9] = new Vector2(1, 0);
uv[10] = new Vector2(0.5, 0.6);

//triangles
tri[0] = 8;
tri[1] = 0;
tri[2] = 1;
```

```
tri[3] = 10;
tri[4] = 1;
tri[5] = 2;

tri[6] = 10;
tri[7] = 2;
tri[8] = 3;

tri[9] = 10;
tri[10] = 3;
tri[11] = 4;

tri[12] = 10;
tri[13] = 4;
tri[14] = 5;

tri[15] = 10;
tri[16] = 5;
tri[17] = 6;

tri[18] = 10;
tri[19] = 6;
tri[20] = 7;

tri[21] = 10;
tri[22] = 7;
```

```
tri[23] = 8;

tri[24] = 8;
tri[25] = 9;
tri[26] = 0;

tri[27] = 8;
tri[28] = 1;
tri[29] = 0;

tri[30] = 10;
tri[31] = 2;
tri[32] = 1;

tri[33] = 10;
tri[34] = 3;
tri[35] = 2;

tri[36] = 10;
tri[37] = 4;
tri[38] = 3;

tri[39] = 10;
tri[40] = 5;
tri[41] = 4;

tri[42] = 10;
```

```
        tri[43] = 6;
        tri[44] = 5;

        tri[45] = 10;
        tri[46] = 7;
        tri[47] = 6;

        tri[48] = 10;
        tri[49] = 8;
        tri[50] = 7;

        tri[51] = 8;
        tri[52] = 0;
        tri[53] = 9;

        var mesh: Mesh = new Mesh();
        mesh.name = "Arch Window";
        mesh.vertices = verts;
        mesh.triangles = tri;
        mesh.uv = uv;
        mesh.normals = normals;
        //mesh.RecalculateNormals();
        tmpobj.renderer.material.mainTexture                                    =
Resources.Load("window_church2");
        mf.mesh = mesh;
}//end function Arch_Wind(RandX, RandZ, tmpobj)
//--------------End Arch Windows-----------------------------------------------------------------
```

```
//-----------------Start Arch Doors-------------------------------------------------------------------
private function Arch_Door(RandX, RandZ, tmpobj){
	tmpobj = new GameObject();
	var mf: MeshFilter = tmpobj.AddComponent(MeshFilter);

	tmpobj.AddComponent(MeshRenderer);
	tmpobj.name = "Arch Door";
	//tmpobj.renderer.material.color = Color.clear;
	tmpobj.transform.parent = oChurch.transform;

	var verts: Vector3[] = new Vector3[11];
	var uv: Vector2[] = new Vector2[11];
	var normals: Vector3[] = new Vector3[11];
	var tri: int[] = new int[54]; //( 3 vertices * 9 triangles ) * 2 (for the back face)

	//vertices positioning
	verts[0] = new Vector3(this.RandX + 4.62, 0.3, this.RandZ + 3.3); //P1
	verts[1] = new Vector3(this.RandX + 4.62, 2.35, this.RandZ + 3.3); //P2
	verts[2] = new Vector3(this.RandX + 4.62, 2.55, this.RandZ + 3.3); //P3
	verts[3] = new Vector3(this.RandX + 4.42, 2.9, this.RandZ + 3.3); //P4
	verts[4] = new Vector3(this.RandX + 4.12, 3, this.RandZ + 3.3); //P5
	verts[5] = new Vector3(this.RandX + 3.72, 3, this.RandZ + 3.3); //P6
	verts[6] = new Vector3(this.RandX + 3.42, 2.9, this.RandZ + 3.3); //P7
	verts[7] = new Vector3(this.RandX + 3.22, 2.55, this.RandZ + 3.3); //P8
```

```
verts[8] = new Vector3(this.RandX + 3.22, 2.35, this.RandZ + 3.3); //P9
verts[9] = new Vector3(this.RandX + 3.22, 0.3, this.RandZ + 3.3); //P10
verts[10] = new Vector3(this.RandX + 3.92, 2.35, this.RandZ + 3.3); //P11


//uv mapping (textures)
uv[0] = new Vector2(0, 0);
uv[1] = new Vector2(0, 0.7);
uv[2] = new Vector2(0, 0);
uv[3] = new Vector2(0, 0);
uv[4] = new Vector2(0, 0);
uv[5] = new Vector2(0, 0);
uv[6] = new Vector2(0, 0);
uv[7] = new Vector2(0, 0);
uv[8] = new Vector2(1, 0.7);
uv[9] = new Vector2(1, 0);
uv[10] = new Vector2(0.5, 0.6);


//triangles
tri[0] = 8;
tri[1] = 0;
tri[2] = 1;

tri[3] = 10;
tri[4] = 1;
tri[5] = 2;
```

```
tri[6] = 10;
tri[7] = 2;
tri[8] = 3;

tri[9] = 10;
tri[10] = 3;
tri[11] = 4;

tri[12] = 10;
tri[13] = 4;
tri[14] = 5;

tri[15] = 10;
tri[16] = 5;
tri[17] = 6;

tri[18] = 10;
tri[19] = 6;
tri[20] = 7;

tri[21] = 10;
tri[22] = 7;
tri[23] = 8;

tri[24] = 8;
tri[25] = 9;
```

```
tri[26] = 0;

tri[27] = 8;
tri[28] = 1;
tri[29] = 0;

tri[30] = 10;
tri[31] = 2;
tri[32] = 1;

tri[33] = 10;
tri[34] = 3;
tri[35] = 2;

tri[36] = 10;
tri[37] = 4;
tri[38] = 3;

tri[39] = 10;
tri[40] = 5;
tri[41] = 4;

tri[42] = 10;
tri[43] = 6;
tri[44] = 5;

tri[45] = 10;
```

```
        tri[46] = 7;
        tri[47] = 6;

        tri[48] = 10;
        tri[49] = 8;
        tri[50] = 7;

        tri[51] = 8;
        tri[52] = 0;
        tri[53] = 9;

        var mesh: Mesh = new Mesh();
        mesh.name = "Arch Door";
        mesh.vertices = verts;
        mesh.triangles = tri;
        mesh.uv = uv;
        mesh.normals = normals;
        mesh.RecalculateNormals();
        tmpobj.renderer.material.mainTexture = Resources.Load("door_corfu2");
//import texture for door
        mf.mesh = mesh;
}//end function Arch_Door(RandX, RandZ, tmpobj)
//-------------------End Arch Doors-------------------------------------------------------------------
--

//---------------------------Start DO Arched Windows & Doors--------------------------------
--------
public function Do_ArchWind(RandX, RandZ){
```

```
        tmpobj = new GameObject();
        Destroy(tmpobj);

        //front main windows
        for(i=0; i<=2; i++){
                if(tmpobj){
                        Arch_Wind(this.RandX, this.RandZ, tmpobj);
                        this.RandX -= 2.6;
                }//end if
        }//end for

        //back main windows
        this.RandX += 2.6;
        this.RandZ -= 6.6;
        for(i=0; i<=2; i++){
                if(tmpobj){
                        Arch_Wind(this.RandX, this.RandZ, tmpobj);
                        this.RandX += 2.6;
                }//end if
        }//end for
}//end function Do_ArchWind(RandX, RandZ)



public function Do_ArchDoor(RandX, RandZ){
        tmpobj = new GameObject();
```

```
        Destroy(tmpobj);

        this.RandX -= 7.8;
        this.RandZ += 6.6;
        if(tmpobj){
                Arch_Door(this.RandX, this.RandZ, tmpobj);
        }//end if

        this.RandZ -= 6.6;
        if(tmpobj){
                Arch_Door(this.RandX, this.RandZ, tmpobj);
        }//end if
}//end function Do_ArchDoor(RandX, RandZ)

public function Do_Arch(RandX, RandZ){
        oArch = new GameObject();
        Destroy(oArch);

        oCube = GameObject.CreatePrimitive(PrimitiveType.Cube);
        Destroy(oCube);

        //front window arches
        this.RandX += 5.2;
        this.RandZ += 6.7;
        for(i=0; i<=2; i++){
                if(oArch){
                        Arch(this.RandX, this.RandZ, oArch);
```

```
                this.RandX -= 2.6;
            }//end if
        }//end for

        //back window arches
        this.RandX += 2.6;
        this.RandZ -= 6.75;
        for(i=0; i<=2; i++){
            if(oArch){
                Arch(this.RandX, this.RandZ, oArch);
                this.RandX += 2.6;
            }//end if
        }//end for
}//end function Do_Arch(RandX, RandZ)
//----------------------------End DO Arched Windows & Doors---------------------------------
-------



//------------------Start DO Door Roof ----------------------------------------------------------
--
public function Do_DoorRoof(RandX, RandZ){
        tmpobj = new GameObject();
        Destroy(tmpobj);

        //front side
        this.RandX -= 2.6;
        this.RandZ += 6.7;
        if(tmpobj){
```

```
            TriangleEaves(this.RandX, this.RandZ, tmpobj);
        }//end if

        //back side
        this.RandZ -= 6.75;
        if(tmpobj){
            TriangleEaves(this.RandX, this.RandZ, tmpobj);
        }//end if
}//end function Do_DoorRoof(RandX, RandZ)
//------------------End DO Door Roof ----------------------------------------------------------------
-


//------------------Start BellTower Arches-----------------------------------
public function Do_BellTower_Arch(RandX, RandZ){
        oArch = new GameObject();
        Destroy(oArch);

        this.RandX -= 6.2;
        this.RandZ += 2.5;
        for(i=0; i<=1; i++){
            for(j=0; j<=1; j++){
                if(oArch){
                    Bell_Tower_Arch(this.RandX, this.RandZ, oArch);
                    this.RandX -= 1.6;
                }//end if
            }//end for
```

```
            this.RandX += 3.2;
            this.RandZ -= 3.2;
        }//end for

        //verical Arches
        this.RandX += 6.95;
        this.RandZ -= 0.5;
        for(i=0; i<=1; i++){
            for(j=0; j<=1; j++){
                if(oArch){
                    Bell_Tower_Vertical_Arch(this.RandX, this.RandZ,
oArch);
                    this.RandZ -= 1.6;
                }//end if
            }//end for
            this.RandZ += 3.2;
            this.RandX -= 3.2;
        }//end for
}//end function Do_BellTower_Arch(RandX, RandZ)
//-------------------------End BellTower Arches-----------------------------
```

//House_1.js

```
//--Main Building Objects-----------------------------------------------------
//--Main Building Objects-----------------------------------------------------
var oBuilding1 : GameObject;
var oWindow : GameObject;
var oCube : GameObject;
```

```
var oRails: GameObject;

//-------------------start public variables---------------------------------------------------------

public var aCollectionObj_1 = new Array(); // an Array - a collection- for the objects of House_1

public var i: int = 0;

public var j: int = 0;

public var c: int = 0; //counter


public var wind_X : float;

public var wind_Y : float;

public var wind_Z : float;


public var RandX: float;

public var RandZ: float;

//-------------------end public variables-----------------------------------------------------------



//MAIN function

public function mainDO() {



        MainBuilding(RandX, RandZ);

        Windows(RandX, RandZ);

        Door(RandX, RandZ);

        Eaves(RandX, RandZ);

        Balcony(RandX, RandZ);

        Roof(RandX, RandZ);
```

```
}

public function Destroy_(){// destroy the house

DestroyImmediate(oBuilding1);

}

//----Start oBuilding1----------------------------------------
public function MainBuilding(RandX, RandZ){
        //create object oBuilding1
        oBuilding1 = GameObject.CreatePrimitive(PrimitiveType.Cube);
        //name the object oBuilding1
        oBuilding1.name = "House_1";
        //set color(r,g,b,a) to the object oBuilding1
        oBuilding1.renderer.material.color = Color(Random.Range(0.98039, 0.88039), Random.Range(0.819607,0.719607), Random.Range(0.843137,0.743137), 0);  //Color(1, 0.714, 0.757, 0);
        //Positioning the oBuilding1 object
        oBuilding1.transform.position = Vector3(this.RandX, 4.153612, this.RandZ);
        //Scaling the oBuilding1 object
        oBuilding1.transform.localScale.x = 5.622439;
        oBuilding1.transform.localScale.y = 8.275616;
        oBuilding1.transform.localScale.z = 4.411408;
        aCollectionObj_1.Add(oBuilding1);
}//end function MainBuilding(RandX, RandZ)
//----End oBuilding1----------------------------------------
```

```
//--------------------------START WINDOWS----------------------------------------------------------
------------

public function Windows(RandX, RandZ){


        oWindow = GameObject.CreatePrimitive(PrimitiveType.Cube);


        oWindow.renderer.material.mainTexture                                    =
Resources.Load("window_corfu3"); //import texture for windows


        wind_X = this.RandX + 1.7556326;

        wind_Y = 6.8821215;

        wind_Z = this.RandZ + 2.2319577;


        //Positioning the oWindow object

        oWindow.transform.position = Vector3(wind_X,wind_Y,wind_Z);

        oWindow.name = "topWindow 0";

        oWindow.transform.parent = oBuilding1.transform;

        //Scaling the oWindow object

        oWindow.transform.localScale.x = 0.1237501;

        oWindow.transform.localScale.y = 0.1284585;

        oWindow.transform.localScale.z = 0.019;


//::::::::::::::::::::::::::::::::::;Start                                    Top
Windows::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

        for(i=0; i<=2; i++){

                if(oWindow != null) {//if the object oWindow exists
```

```
                var cloneTopWindows = Instantiate(oWindow, Vector3(wind_X - 1.2, wind_Y, wind_Z), Quaternion.identity); //make 3 more clones of the oWindow object and place them near to it

                cloneTopWindows.transform.parent = oBuilding1.transform; //parenting

                cloneTopWindows.name = "topWindow " + (++c);

                cloneTopWindows.transform.localScale.x = 0.1237501;

                cloneTopWindows.transform.localScale.y = 0.1284585;

                cloneTopWindows.transform.localScale.z = 0.019;

                aCollectionObj_1.Add(cloneTopWindows);

                wind_X -= 1.2;

            } //end if

        } //end for

        c = 0;


        //left side windows--------------------------

        var nWind_X:float = this.RandX + 2.8090137;

        var nWind_Y:float = 6.8821215;

        var nWind_Z:float = this.RandZ - 1.0237426;

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

            for(j=0; j<=1; j++){

                if(oWindow != null){

                    var cloneLeftWindows = Instantiate(oWindow, Vector3(nWind_X, nWind_Y, nWind_Z), Quaternion.identity);

                    cloneLeftWindows.transform.parent = oBuilding1.transform; //parenting

                    cloneLeftWindows.name = "top left window " + (++c);

                    cloneLeftWindows.transform.localScale.x = 0.1608783;

                    cloneLeftWindows.transform.localScale.y = 0.1284585;
```

```
                        cloneLeftWindows.transform.localScale.z = 0.019;
                        cloneLeftWindows.transform.eulerAngles.y = 90;
//rotates y angle 90 degrees
                        aCollectionObj_1.Add(cloneLeftWindows);
                    }//end if
                    nWind_Y -= 2.5;
                }//end for
                nWind_Y = 6.8821215;
                nWind_Z += 2.0;
            }//end for
            c = 0;

            //right side windows--------------------------
            nWind_X = this.RandX - 2.8090137;
            nWind_Y = 6.8821215;
            nWind_Z = this.RandZ - 1.0237426;
            for(i=0; i<=1; i++){
                for(j=0; j<=1; j++){
                    if(oWindow != null){
                        var cloneRightWindows = Instantiate(oWindow,
Vector3(nWind_X, nWind_Y, nWind_Z), Quaternion.identity);
                        cloneRightWindows.transform.parent =
oBuilding1.transform; //parenting
                        cloneRightWindows.name = "top right window " +
(++c);
                        cloneRightWindows.transform.localScale.x =
0.1608783;
                        cloneRightWindows.transform.localScale.y =
0.1284585;
```

```
                        cloneRightWindows.transform.localScale.z = 0.019;
                        cloneRightWindows.transform.eulerAngles.y = 90;
//rotates y angle 90 degrees
                        aCollectionObj_1.Add(cloneRightWindows);
                    }//end if
                    nWind_Y -= 2.5;
                }//end for
                nWind_Y = 6.8821215;
                nWind_Z += 2.0;
            }//end for
            c = 0;
//:::::::::::::::::::::::::::::::::;End Top
Windows::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::


//:::::::::::::::::::::::::::::::::;Start Middle
Windows::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

            wind_X = this.RandX + 2.8973138;
            wind_Y = 4.074193;

            for(i=0; i<=3; i++){
                if(oWindow != null){
                    var cloneMiddleWindows = Instantiate(oWindow,
Vector3(wind_X - 1.2, wind_Y, wind_Z), Quaternion.identity);
                    cloneMiddleWindows.transform.parent = oBuilding1.transform;
//parenting
                    cloneMiddleWindows.name = "middleWindow " + (++c);
                    cloneMiddleWindows.transform.localScale.x = 0.1288288;
```

```
                    cloneMiddleWindows.transform.localScale.y = 0.2147541;
                    cloneMiddleWindows.transform.localScale.z = 0.019;
                    aCollectionObj_1.Add(cloneMiddleWindows);
                    wind_X -= 1.2;
               } //end  if
          } //end for
          c = 0;
//::::::::::::::::::::::::::::::::;End Middle Windows:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::


//::::::::::::::::::::::::::::::::;Start Down Windows:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

          wind_X = this.RandX + 1.5556326;
          wind_Y = 1.00569;

          for(i=0; i<=1; i++){
               if(oWindow != null) {//if the object oWindow exists
                    var cloneDownWindows = Instantiate(oWindow, Vector3(wind_X,wind_Y,wind_Z), Quaternion.identity);
                    cloneDownWindows.transform.parent = oBuilding1.transform; //parenting
                    cloneDownWindows.name = "downWindow " + (++c);
                    cloneDownWindows.transform.localScale.x = 0.1818855;
                    cloneDownWindows.transform.localScale.y = 0.1157715;
                    cloneDownWindows.transform.localScale.z = 0.0155889;
                    aCollectionObj_1.Add(cloneDownWindows);
                    wind_X -= 3.154466;
```

```
            } //end if
        } //end for
        c = 0;
//::::::::::::::::::::::::::::::::::;End Down Windows::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
}//end function Windows(RandX, RandZ)
//-------------------------END  WINDOWS-----------------------------------------------------------------
```

```
//------------------------------START  DOOR-----------------------------------------------------------
public function Door(RandX, RandZ){

        oWindow = GameObject.CreatePrimitive(PrimitiveType.Cube);
        Destroy(oWindow);
        //oWindow.transform.renderer.material.color = Color.clear;

        oWindow.renderer.material.mainTexture = Resources.Load("door_corfu3"); //import texture for door

        var oDoor = Instantiate(oWindow, Vector3(this.RandX, 0.953612, this.RandZ + 2.2319577), Quaternion.identity);
        oDoor.name = "door";
        oDoor.transform.parent = oBuilding1.transform;

        oDoor.transform.localScale.x = 0.1656405;
        oDoor.transform.localScale.y = 0.2321697;
        oDoor.transform.localScale.z = 0.019;
```

```
        aCollectionObj_1.Add(oDoor);
}//end function Door(RandX, RandZ)
//------------------------------END DOOR-----------------------------------------------------------
----



//----------------------------------START  EAVES-------------------------------------------------
-------
public function Eaves(RandX, RandZ){

        oCube = GameObject.CreatePrimitive(PrimitiveType.Cube);
        Destroy(oCube);

        //eaves facade
        var eaves_X:float = this.RandX + 1.7556326;
        var eaves_Y:float = 7.4821215;
        var eaves_Z:float = this.RandZ + 2.2519577;

        for(i=0; i<=3; i++){
                for(j=0; j<=1; j++){
                        if(oCube != null){
                                var cloneEaves_facade = Instantiate(oCube,
Vector3(eaves_X, eaves_Y, eaves_Z), Quaternion.identity);
                                cloneEaves_facade.transform.parent =
oBuilding1.transform;
                                cloneEaves_facade.name = "eaves " + (++c);
                                cloneEaves_facade.transform.localScale.x =
0.1912387;
                                cloneEaves_facade.transform.localScale.y = 0.0125;
```

```
                        cloneEaves_facade.transform.localScale.z = 0.025;
                        aCollectionObj_1.Add(cloneEaves_facade);
                    }//end if
                    eaves_Y -= 2.3;
                }//end for
                eaves_Y = 7.4821215;
                eaves_X -= 1.2;
            }//end for
            c = 0;

            //left side eaves----------------------------------
            eaves_X = this.RandX + 2.83;
            eaves_Y = 7.4821215;
            eaves_Z = this.RandZ - 1.0237426;
            for(i=0; i<=1; i++){
                for(j=0; j<=1; j++){
                    if(oWindow != null){
                        var cloneEaves_lftSd = Instantiate(oCube,
Vector3(eaves_X, eaves_Y, eaves_Z), Quaternion.identity);
                        cloneEaves_lftSd.transform.parent =
oBuilding1.transform; //parenting
                        cloneEaves_lftSd.name = "left side eave " + (++c);
                        cloneEaves_lftSd.transform.localScale.x = 0.1919583;
                        cloneEaves_lftSd.transform.localScale.y = 0.0125;
                        cloneEaves_lftSd.transform.localScale.z = 0.025;
                        cloneEaves_lftSd.transform.eulerAngles.y = 90;
//rotates y angle 90 degrees
                        aCollectionObj_1.Add(cloneEaves_lftSd);
```

```
                }//end if
                eaves_Y -= 2.5;
            }//end for
            eaves_Y = 7.4821215;
            eaves_Z += 2.0;
        }//end for
        c = 0;



        //right side eaves----------------------------------
        eaves_X = this.RandX - 2.83;
        eaves_Y = 7.4821215;
        eaves_Z = this.RandZ - 1.0237426;
        for(i=0; i<=1; i++){
            for(j=0; j<=1; j++){
                if(oWindow != null){
                    var cloneEaves_rghtSd = Instantiate(oCube, Vector3(eaves_X, eaves_Y, eaves_Z), Quaternion.identity);
                    cloneEaves_rghtSd.transform.parent = oBuilding1.transform; //parenting
                    cloneEaves_rghtSd.name = "right side eave " + (++c);
                    cloneEaves_rghtSd.transform.localScale.x = 0.1919583;
                    cloneEaves_rghtSd.transform.localScale.y = 0.0125;
                    cloneEaves_rghtSd.transform.localScale.z = 0.025;
                    cloneEaves_rghtSd.transform.eulerAngles.y = 90; //rotates y angle 90 degrees
                    aCollectionObj_1.Add(cloneEaves_rghtSd);
                }//end if
```

```
                eaves_Y -= 2.5;
            }//end for
            eaves_Y = 7.4821215;
            eaves_Z += 2.0;
        }//end for
        c = 0;

}//end function Eaves(RandX, RandZ)
//----------------------------------END EAVES-------------------------------------------------------
---




//-------------------------------------START BALCONY-----------------------------------------------
------
public function Balcony(RandX, RandZ){

        oRails = GameObject.CreatePrimitive(PrimitiveType.Cube);
        Destroy(oRails);
        oCube = GameObject.CreatePrimitive(PrimitiveType.Cube);
        Destroy(oCube);


        oRails.renderer.material.color = Color(0.4118, 0.4118, 0.4118, 0);


        var base_ = Instantiate(oCube, Vector3(this.RandX, 3.1, this.RandZ +
2.4719577), Quaternion.identity);
        base_.transform.parent = oBuilding1.transform; //parenting
        base_.name = "base balcony";
```

```
base_.transform.localScale.x = 0.947154;
base_.transform.localScale.y = 0.0125;
base_.transform.localScale.z = 0.1373834;
aCollectionObj_1.Add(base_);
c = 0;

//Rails Facade----------------------------------------
var rail_X:float = this.RandX;
var rail_Y:float = 3.7;
var rail_Z:float = this.RandZ + 2.67;

for(i=0; i<=1; i++){
    if(oCube != null){
        var rail_ = Instantiate(oRails,Vector3(rail_X, rail_Y, rail_Z), Quaternion.identity);
        rail_.transform.parent = oBuilding1.transform; //parenting
        rail_.name = " main rail " + (++c);
        rail_.transform.localScale.x = 0.9464975;
        rail_.transform.localScale.y = 0.006083317;
        rail_.transform.localScale.z = 0.02218747;
        aCollectionObj_1.Add(rail_);
        rail_Y -= 0.55;
    }//end if
}//end for
c = 0;
```

```
rail_X = this.RandX + 2.6259075;
rail_Y = 3.42;
rail_Z = this.RandZ + 2.67;

for(i=0; i<=26; i++){
    if(oCube != null){
        var oRails_ = Instantiate(oRails,Vector3(rail_X, rail_Y, rail_Z),
Quaternion.identity);
        oRails_.transform.parent = oBuilding1.transform; //parenting
        oRails_.name = "rails " + (++c);
        oRails_.transform.localScale.x = 0.01005142;
        oRails_.transform.localScale.y = 0.05990216;
        oRails_.transform.localScale.z = 0.01005142;
        aCollectionObj_1.Add(oRails_);
        rail_X -= 0.2;
    }//end if
}//end for
c = 0;


//Main Side Rails ---------------------------------------
rail_X = this.RandX + 2.6259075;
rail_Y = 3.7;
rail_Z = this.RandZ + 2.46;

for(i=0; i<=1; i++){
    for(j=0; j<=1; j++){
```

```
                    if(oCube != null){
                        var oSideRails = Instantiate(oRails,Vector3(rail_X, rail_Y, rail_Z), Quaternion.identity);
                        oSideRails.transform.parent = oBuilding1.transform; //parenting
                        oSideRails.name = "main side rail " + (++c);
                        oSideRails.transform.localScale.x = 0.02218747;
                        oSideRails.transform.localScale.y = 0.006083317;
                        oSideRails.transform.localScale.z = 0.1151402;
                        aCollectionObj_1.Add(oSideRails);
                    }//end if
                    rail_Y -= 0.55;
                }//end for
                rail_Y = 3.7;
                rail_X -= 5.251815;
            }//end for
            c = 0;


            //Side Rails ----------------------------------------
            rail_X = this.RandX + 2.6259075;
            rail_Y = 3.42;
            rail_Z = this.RandZ + 2.67;

            for(i=0; i<=2; i++){
                for(j=0; j<=1; j++){
                if(oCube != null){
```

```
                    var oRails_sd = Instantiate(oRails,Vector3(rail_X, rail_Y, rail_Z), Quaternion.identity);
                    oRails_sd.transform.parent = oBuilding1.transform; //parenting
                    oRails_sd.name = "side rails " + (++c);
                    oRails_sd.transform.localScale.x = 0.01005142;
                    oRails_sd.transform.localScale.y = 0.05990216;
                    oRails_sd.transform.localScale.z = 0.01005142;
                    aCollectionObj_1.Add(oRails_sd);
                }//end if
                rail_X -= 5.251815;
        }//end for
        rail_X = this.RandX + 2.6259075;
        rail_Z -= 0.2;
        }//end for
        c = 0;
}//end function Balcony(RandX, RandZ)
//-----------------------------------END  BALCONY-----------------------------------------------
---



//-------------------------------------START  ROOF-------------------------------------------
-----
public function Roof(RandX, RandZ){
        var tmpobj: GameObject = new GameObject();
        var mf: MeshFilter = tmpobj.AddComponent(MeshFilter);

//      tmpobj.AddComponent(MeshFilter);
        tmpobj.AddComponent(MeshRenderer);
```

```
tmpobj.name = "Roof";
tmpobj.transform.parent = oBuilding1.transform;

var verts: Vector3[] = new Vector3[6];
var normals: Vector3[] = new Vector3[6];
var uv: Vector2[] = new Vector2[6];
var tri: int[] = new int[18]; //3 vertices * 6 triangles =18


//vertices positioning
verts[0] = new Vector3(this.RandX - 2.8, 9.5, this.RandZ ); //P1
verts[1] = new Vector3(this.RandX - 2.8, 8.3, this.RandZ -2.25); //P2
verts[2] = new Vector3(this.RandX - 2.8, 8.3, this.RandZ +2.25); //P3

verts[3] = new Vector3(this.RandX +2.8, 9.5, this.RandZ ); //P4
verts[4] = new Vector3(this.RandX +2.8, 8.3, this.RandZ +2.25); //P5
verts[5] = new Vector3(this.RandX +2.8, 8.3, this.RandZ -2.25); //P6


uv[0] = new Vector2(0, 0.5);
uv[1] = new Vector2(1, 1);
uv[2] = new Vector2(1, 0);

uv[3] = new Vector2(1, 0.5);
uv[4] = new Vector2(0, 0);
uv[5] = new Vector2(0, 1);
```

```
//triangles
tri[0] = 0; //refer to vertices: verts[0]
tri[1] = 1; //refer to vertices: verts[1]
tri[2] = 2; //refer to vertices: verts[2]

tri[3] = 0;
tri[4] = 2;
tri[5] = 4;

tri[6] = 3;
tri[7] = 0;
tri[8] = 4;

tri[9] = 3;
tri[10] = 4;
tri[11] = 5;

tri[12] = 3;
tri[13] = 5;
tri[14] = 0;

tri[15] = 0;
tri[16] = 5;
tri[17] = 1;

var mesh: Mesh = new Mesh();
```

```
        mesh.name = "Roof";
        mesh.vertices = verts;
        mesh.triangles = tri;
        mesh.uv = uv;
        mesh.normals = normals;
        mesh.RecalculateNormals();
        mf.mesh = mesh;
        tmpobj.renderer.material.mainTexture = Resources.Load("RoofTileWood");
//import texture
}//end function Roof(RandX, RandZ)
//--------------------------------------END ROOF-------------------------------------------------
```

//House_2.js

```
//--Main Building Objects-----------------------------------------------------------
var oBuilding2 : GameObject;
var oWindow : GameObject;
var oBalconyWind : GameObject;
var oDoor : GameObject;
var oAttic : GameObject;
var oRails : GameObject;
//var oCube : GameObject; //= GameObject.CreatePrimitive(PrimitiveType.Cube);



//------------------start public variables-----------------------------------------------------
public var aCollectionObj_2 = new Array(); // an Array - a collection- for the objects of House_2
public var i: int = 0;
public var j: int = 0;
```

```
public var c: int = 0;


public var wind_X : float;
public var wind_Y : float;
public var wind_Z : float;


public var RandX: float;
public var RandZ: float;
//------------------end public variables-------------------------------------------------------


//MAIN function
public function mainDO() {


        //RandX = Random.Range(10,247);
        //RandZ = Random.Range(10,247);


        MainBuilding(RandX, RandZ);
        Windows(RandX, RandZ);
        Attic(RandX, RandZ);
        Balcony(RandX, RandZ);
        Doors(RandX, RandZ);
        MainRoof(RandX, RandZ);
        AtticRoof(RandX, RandZ);
}//end function Start ()


public function Destroy_(){
```

```
DestroyImmediate(oBuilding2);

}

//----Start MainBuilding----------------------------------------
public function MainBuilding(RandX, RandZ){
        //create object oBuilding2
        oBuilding2 = GameObject.CreatePrimitive(PrimitiveType.Cube);
        //Positioning the oBuilding2 object
        oBuilding2.transform.position = Vector3(this.RandX, 6.35569, this.RandZ);
        //name the object oBuilding2
         oBuilding2.name = "House_2";
         //oBuilding2.renderer.material.mainTexture = Resources.Load("wall_");
         oBuilding2.renderer.material.color = Color(Random.Range(1,0.9),
Random.Range(0.980,0.880), Random.Range(0.804,0.704), 0); //set color
        //Random ranges between X, Y coordinates
        //Scaling the oBuilding2 object
        oBuilding2.transform.localScale.x = 7.0;
        oBuilding2.transform.localScale.y = 12.0;
        oBuilding2.transform.localScale.z = 7.0;
        aCollectionObj_2.Add(oBuilding2);
}//end MainBuilding(RandX, RandZ)
//----End function MainBuilding----------------------------------------
```

```
//-------------------------START WINDOWS-----------------------------------------------------------

public function Windows(RandX, RandZ){

//::::::::::::::::::::::::::::::::::::::::::::Start FACADE::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

	oWindow = GameObject.CreatePrimitive(PrimitiveType.Cube);

	//transparent windows

	//oWindow.transform.renderer.material.color = Color.clear;

	oWindow.renderer.material.mainTexture = Resources.Load("window_corfu2");  //import texture for windows

	//Positioning the oWindow object

	wind_X = this.RandX + 2.82102;

	wind_Y = 11.474193;

	wind_Z = this.RandZ + 3.5051424;

	oWindow.transform.position = Vector3(wind_X,wind_Y,wind_Z);

	oWindow.name = "window0";

	oWindow.transform.parent = oBuilding2.transform;

	//Scaling the oWindow object

	oWindow.transform.localScale.x =0.1;

	oWindow.transform.localScale.y = 0.116753;

	oWindow.transform.localScale.z = 0.019;



	var nPosY_: float = wind_Y - 2.5;


	for(i=0; i<=2; i++){

		if(oWindow){ //if the oWindow object exists

			var cloneWindow_leftColumn = Instantiate(oWindow, Vector3(wind_X, nPosY_, wind_Z), Quaternion.identity);
```

```
                cloneWindow_leftColumn.transform.parent = oBuilding2.transform; //parenting

                cloneWindow_leftColumn.name = "window" + (++c);

                cloneWindow_leftColumn.transform.localScale.x =0.1;

                cloneWindow_leftColumn.transform.localScale.y = 0.116753;

                cloneWindow_leftColumn.transform.localScale.z = 0.019;

                aCollectionObj_2.Add(cloneWindow_leftColumn);

                nPosY_-=2.5;

            }//end if

        }//end for

        c = 0;


        for(i=0; i<=2; i++){

            if(oWindow){

                var cloneWindow_3rd_flr = Instantiate(oWindow, Vector3(wind_X - 1.45, wind_Y - 2.5, wind_Z), Quaternion.identity);

                cloneWindow_3rd_flr.transform.parent = oBuilding2.transform; //parenting

                cloneWindow_3rd_flr.name = "3rd flr window" + (++c);

                cloneWindow_3rd_flr.transform.localScale.x =0.1;

                cloneWindow_3rd_flr.transform.localScale.y = 0.116753;

                cloneWindow_3rd_flr.transform.localScale.z = 0.019;

                aCollectionObj_2.Add(cloneWindow_3rd_flr);

                wind_X -= 1.45;

            }//end if

        }//end for

        c = 0;
```

```
var tmpWind_X = this.RandX + 2.82102;
for(i=0; i<=2; i++){
    if(oWindow){
        var cloneWindow_1st_flr = Instantiate(oWindow, Vector3(tmpWind_X - 1.45, wind_Y - 7.5, wind_Z), Quaternion.identity);
        cloneWindow_1st_flr.transform.parent = oBuilding2.transform; //parenting
        cloneWindow_1st_flr.name = "1st flr window" + (++c);
        cloneWindow_1st_flr.transform.localScale.x = 0.1;
        cloneWindow_1st_flr.transform.localScale.y = 0.116753;
        cloneWindow_1st_flr.transform.localScale.z = 0.019;
        aCollectionObj_2.Add(cloneWindow_1st_flr);
        tmpWind_X -= 1.45;
    }//end if
}//end for
c = 0;

var tmp_Wind_Y = 11.474193;
var tmp_Wind_X = this.RandX + 2.82102;
for(i=0; i<=3; i++){
    if(oWindow){
        var cloneWindow_rightColumn = Instantiate(oWindow, Vector3( tmp_Wind_X - 5.7, tmp_Wind_Y, wind_Z), Quaternion.identity);
        cloneWindow_rightColumn.transform.parent = oBuilding2.transform; //parenting
        cloneWindow_rightColumn.name = "window" + (++c);
        cloneWindow_rightColumn.transform.localScale.x = 0.1;
        cloneWindow_rightColumn.transform.localScale.y = 0.116753;
        cloneWindow_rightColumn.transform.localScale.z = 0.019;
```

```
                aCollectionObj_2.Add(cloneWindow_rightColumn);
                tmp_Wind_Y-= 2.5;
            }//end if
        }//end for
        c = 0;

        tmpWind_X = this.RandX + 2.82102;
        for(i=0; i<=1; i++){
            if(oWindow){
                var cloneGND_wind = Instantiate(oWindow,
Vector3(tmpWind_X - 1.45, 1.35569, wind_Z), Quaternion.identity);
                cloneGND_wind.transform.parent = oBuilding2.transform;
//parenting
                cloneGND_wind.name = "gnd flr window" + (++c);
                cloneGND_wind.transform.localScale.x = 0.1;
                cloneGND_wind.transform.localScale.y = 0.116753;
                cloneGND_wind.transform.localScale.z = 0.019;
                aCollectionObj_2.Add(cloneGND_wind);
                tmpWind_X -= 2.9;
            }//end if
        }//end for
        c = 0;
        //::::::::::::::::::::::::::::::::::::::::::::End
FACADE::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

        //::::::::::::::::::::::::::::::::::::::::::::Start                    Left
Side::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
        /* var wind_X = RandX + 2.82102;
        var wind_Y = 11.474193;
```

```
var wind_Z = RandZ + 3.4551424;*/


wind_X = this.RandX + 3.50;
wind_Y = 11.474193;
wind_Z = this.RandZ - 1.82102;
for(i=0; i<=2; i++){
        for(j=0; j<=4; j++){
                if(oWindow){
                var cloneSideLeftWindow = Instantiate(oWindow, Vector3(wind_X, wind_Y, wind_Z), Quaternion.identity);
                cloneSideLeftWindow.transform.parent = oBuilding2.transform; //parenting
                cloneSideLeftWindow.name = "left side window " + (++c);
                cloneSideLeftWindow.transform.localScale.x = 0.1;
                cloneSideLeftWindow.transform.localScale.y = 0.116753;
                cloneSideLeftWindow.transform.localScale.z = 0.019;
                cloneSideLeftWindow.transform.eulerAngles.y = 90; //rotates y angle 90 degrees
                aCollectionObj_2.Add(cloneSideLeftWindow);
                wind_Y-=2.5;
                if( (i==1) && (j==4) )
                        cloneSideLeftWindow.active = false;
                }//end if
        }//end for
        wind_Y = 11.474193;
        wind_Z += 1.45;
}//end for
c = 0;
```

```
//:::::::::::::::::::::::::::::::::::::::::End Left Side:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::


//:::::::::::::::::::::::::::::::::::::::::Start Right Side:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

wind_X = this.RandX - 3.50;

wind_Y = 11.474193;

wind_Z = this.RandZ - 1.82102;

for(i=0; i<=2; i++){

	for(j=0; j<=4; j++){

		if(oWindow){

		var cloneSideRightWindow = Instantiate(oWindow, Vector3(wind_X, wind_Y, wind_Z), Quaternion.identity);

		cloneSideRightWindow.transform.parent = oBuilding2.transform; //parenting

		cloneSideRightWindow.name = "right side window " + (++c);

		cloneSideRightWindow.transform.localScale.x =0.1;

		cloneSideRightWindow.transform.localScale.y = 0.116753;

		cloneSideRightWindow.transform.localScale.z = 0.019;

		cloneSideRightWindow.transform.eulerAngles.y = 90; //rotates y angle 90 degrees

		aCollectionObj_2.Add(cloneSideRightWindow);

		wind_Y-=2.5;

		if( (i==1) && (j==4))

		cloneSideRightWindow.active = false;

		}//end if

	}//end for

	wind_Y = 11.474193;

	wind_Z += 1.45;
```

```
        }//end for

        c = 0;

        //::::::::::::::::::::::::::::::::::::::::::::End                                    Right
Side::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

}//end function Windows(RandX, RandZ)

//--------------------------END  WINDOWS-----------------------------------------------------
---------
```

```
//---------------------------START  ATTIC-------------------------------------------------------
-------

public function Attic(RandX, RandZ){

//::::::::::::::::::::::::::::::::::::::::::::::Start
facade::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

        oAttic = GameObject.CreatePrimitive(PrimitiveType.Cube);


        //Positioning the oAttic object

        oAttic.transform.position = Vector3( this.RandX, 13.20, this.RandZ +
2.7851424);

        oAttic.transform.parent = oBuilding2.transform;

        oAttic.name = "Attic";

        //Scaling the oAttic object

        oAttic.transform.localScale.x = 0.30;

        oAttic.transform.localScale.y =  0.15;

        oAttic.transform.localScale.z = 0.20;


        oAttic.renderer.material.color = Color(1, 0.980, 0.804, 0);

        //Attic window
```

```
var wind_Z = this.RandZ + 3.4551424;


oWindow = GameObject.CreatePrimitive(PrimitiveType.Cube);
Destroy(oWindow);
//transparent windows
//oWindow.transform.renderer.material.color = Color.clear;
oWindow.renderer.material.mainTexture = Resources.Load("window_corfu2"); //import texture for attic window


var cloneAtticWind = Instantiate(oWindow, Vector3(this.RandX, 13.20, wind_Z), Quaternion.identity);
cloneAtticWind.transform.parent = oBuilding2.transform; //parenting
cloneAtticWind.name = "attic window";
cloneAtticWind.transform.localScale.x =0.1;
cloneAtticWind.transform.localScale.y = 0.070753;
cloneAtticWind.transform.localScale.z = 0.019;
aCollectionObj_2.Add(cloneAtticWind);


var eaves_Y: float = 12.674193;
    if(oAttic){
        var lft_upper_eaves = Instantiate(oAttic, Vector3(this.RandX + 1.90102, eaves_Y, this.RandZ), Quaternion.identity);
        lft_upper_eaves.transform.parent = oBuilding2.transform; //parenting
        lft_upper_eaves.name = "left upper eaves";
        lft_upper_eaves.transform.localScale.x = 0.332;
        lft_upper_eaves.transform.localScale.y = 0.0125;
        lft_upper_eaves.transform.localScale.z = 1.119;
        aCollectionObj_2.Add(lft_upper_eaves);
```

```
var main_upper_eaves =Instantiate(oAttic, Vector3(this.RandX, eaves_Y - 0.20, this.RandZ), Quaternion.identity);

main_upper_eaves.transform.parent = oBuilding2.transform; //parenting

main_upper_eaves.name = "main upper eaves";

main_upper_eaves.transform.localScale.x = 1.052;

main_upper_eaves.transform.localScale.y = 0.0110;

main_upper_eaves.transform.localScale.z = 1.090;

aCollectionObj_2.Add(main_upper_eaves);


var rgt_upper_eaves = Instantiate(oAttic, Vector3(this.RandX - 1.90102, eaves_Y, this.RandZ), Quaternion.identity);

rgt_upper_eaves.transform.parent = oBuilding2.transform; //parenting

rgt_upper_eaves.name = "right upper eaves";

rgt_upper_eaves.transform.localScale.x = 0.332;

rgt_upper_eaves.transform.localScale.y = 0.0125;

rgt_upper_eaves.transform.localScale.z = 1.119;

aCollectionObj_2.Add(rgt_upper_eaves);


var main_lower_eaves =Instantiate(oAttic, Vector3(this.RandX, eaves_Y - 0.30, this.RandZ), Quaternion.identity);

main_lower_eaves.transform.parent = oBuilding2.transform; //parenting

main_lower_eaves.name = "main lower eaves";

main_lower_eaves.transform.localScale.x = 1.025;

main_lower_eaves.transform.localScale.y = 0.0110;

main_lower_eaves.transform.localScale.z = 1.045;
```

```
                aCollectionObj_2.Add(main_lower_eaves);
            }//end if


        var eavesWind_X = this.RandX + 2.82102;
        var eavesWind_Y = 9.694193;
        for(i=0; i<=1; i++){
            if(oAttic){
                var thrd_flr_eavesWind = Instantiate(oAttic,
Vector3(eavesWind_X, eavesWind_Y, wind_Z + 0.1), Quaternion.identity);
                thrd_flr_eavesWind.transform.parent = oBuilding2.transform;
//parenting
                thrd_flr_eavesWind.name = "3rd floor window eaves " + (++c);
                thrd_flr_eavesWind.transform.localScale.x = 0.15;
                thrd_flr_eavesWind.transform.localScale.y = 0.0125;
                thrd_flr_eavesWind.transform.localScale.z = 0.025;
                aCollectionObj_2.Add(thrd_flr_eavesWind);
                eavesWind_X -= 5.8;
            }//end if
        }//end for
        c = 0;


        eavesWind_X = this.RandX + 2.82102;
        eavesWind_Y = 7.094193;
        for(i=0; i<=4; i++){
            if(oAttic){
                var scnd_flr_eavesWind = Instantiate(oAttic,
Vector3(eavesWind_X, eavesWind_Y, wind_Z + 0.1), Quaternion.identity);
                scnd_flr_eavesWind.transform.parent = oBuilding2.transform;
//parenting
```

```
                scnd_flr_eavesWind.name = "2nd floor window eaves " +
(++c);
                scnd_flr_eavesWind.transform.localScale.x = 0.15;
                scnd_flr_eavesWind.transform.localScale.y = 0.0125;
                scnd_flr_eavesWind.transform.localScale.z = 0.025;
                aCollectionObj_2.Add(scnd_flr_eavesWind);
                eavesWind_X -= 1.45;
            }//end if
        }//end for
        c = 0;

        eavesWind_X = this.RandX + 2.82102;
        eavesWind_Y = 4.694193;
        for(i=0; i<=1; i++){
            if(oAttic){
                var frst_flr_eavesWind = Instantiate(oAttic,
Vector3(eavesWind_X, eavesWind_Y, wind_Z + 0.1), Quaternion.identity);
                frst_flr_eavesWind.transform.parent = oBuilding2.transform;
//parenting
                frst_flr_eavesWind.name = "1st floor window eaves " + (++c);
                frst_flr_eavesWind.transform.localScale.x = 0.15;
                frst_flr_eavesWind.transform.localScale.y = 0.0125;
                frst_flr_eavesWind.transform.localScale.z = 0.025;
                aCollectionObj_2.Add(frst_flr_eavesWind);
                eavesWind_X -= 5.8;
            }//end if
        }//end for
        c = 0;
```

```
        //::::::::::::::::::::::::::::::::::::::::::::End
facade::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

        //::::::::::::::::::::::::::::::::::::::::::::Start                                    Left
Side::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

        var eaves_X = this.RandX + 3.55;

        var n_eaves_Y = 12.174193;

        var eaves_Z = this.RandZ - 1.82102;

        for(i=0; i<=2; i++){

                for(j=0; j<=4; j++){

                if(oAttic){

                        var cloneSideLeftEaves = Instantiate(oAttic, Vector3(eaves_X,
n_eaves_Y, eaves_Z), Quaternion.identity);

                        cloneSideLeftEaves.transform.parent  =  oBuilding2.transform;
//parenting

                        cloneSideLeftEaves.name = "left side eaves " + (++c);

                        cloneSideLeftEaves.transform.localScale.x = 0.15;

                        cloneSideLeftEaves.transform.localScale.y = 0.0125;

                        cloneSideLeftEaves.transform.localScale.z = 0.025;

                        cloneSideLeftEaves.transform.eulerAngles.y  =  90;  //rotates  y
angle 90 degrees

                        aCollectionObj_2.Add(cloneSideLeftEaves);

                        n_eaves_Y-=2.5;

                        if( (i==1) && (j==4) )

                                cloneSideLeftEaves.active = false;

                        }//end if

                }//end for

                n_eaves_Y = 12.174193;

                eaves_Z += 1.45;
```

```
}//end for

c = 0;

//::::::::::::::::::::::::::::::::::::::::End                                    Left
Side::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

//::::::::::::::::::::::::::::::::::::::::Start                                   Right
Side::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

eaves_X = this.RandX - 3.55;

n_eaves_Y = 12.174193;

eaves_Z = this.RandZ - 1.82102;

for(i=0; i<=2; i++){

	for(j=0; j<=4; j++){

		if(oAttic){

		var cloneSideRightEaves = Instantiate(oAttic,
Vector3(eaves_X, n_eaves_Y, eaves_Z), Quaternion.identity);

		cloneSideRightEaves.transform.parent =
oBuilding2.transform; //parenting

		cloneSideRightEaves.name = "right side eaves " + (++c);

		cloneSideRightEaves.transform.localScale.x = 0.15;

		cloneSideRightEaves.transform.localScale.y = 0.0125;

		cloneSideRightEaves.transform.localScale.z = 0.025;

		cloneSideRightEaves.transform.eulerAngles.y = 90; //rotates y
angle 90 degrees

		aCollectionObj_2.Add(cloneSideRightEaves);

		n_eaves_Y-=2.5;

		if( (i==1) && (j==4))

			cloneSideRightEaves.active = false;

		}//end if

	}//end for

	n_eaves_Y = 12.174193;
```

```
			eaves_Z += 1.45;
		}//end for
		c = 0;
		//:::::::::::::::::::::::::::::::::::::::::End Right Side::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
}//end function Attic(RandX, RandZ)
//---------------------------END ATTIC------------------------------------------------------------------

//------------------------------START BALCONY------------------------------------------
public function Balcony(RandX, RandZ){

		oBalconyWind = GameObject.CreatePrimitive(PrimitiveType.Cube);

		//oBalconyWind.transform.renderer.material.color = Color.clear;

		oBalconyWind.renderer.material.mainTexture = Resources.Load("window_corfu1");  //import texture for balcony windows
		//Positioning the oBalconyWind object
		var Balcwind_X = this.RandX + 2.82102;
		var Balcwind_Y = 11.074193;
		var Balcwind_Z = this.RandZ + 3.5051424;
		oBalconyWind.transform.position = Vector3(Balcwind_X - 1.45,Balcwind_Y,Balcwind_Z);
		oBalconyWind.transform.parent = oBuilding2.transform;
		oBalconyWind.name = "Balcony window 0";
		//Scaling the oBalconyWind object
```

```
oBalconyWind.transform.localScale.x = 0.109;

oBalconyWind.transform.localScale.y = 0.176753;

oBalconyWind.transform.localScale.z = 0.019;


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

    if(oBalconyWind){

        var cloneBalconyWind_4th_flr = Instantiate(oBalconyWind,Vector3(Balcwind_X - 2.9, Balcwind_Y, Balcwind_Z), Quaternion.identity);

        cloneBalconyWind_4th_flr.transform.parent = oBuilding2.transform; //parenting

        cloneBalconyWind_4th_flr.name = "4th floor balcony window " + (++c);

        cloneBalconyWind_4th_flr.transform.localScale.x = 0.109;

        cloneBalconyWind_4th_flr.transform.localScale.y = 0.176753;

        cloneBalconyWind_4th_flr.transform.localScale.z = 0.019;

        aCollectionObj_2.Add(cloneBalconyWind_4th_flr);

        Balcwind_X -= 1.45;

    }//end if

}//end for

c = 0;


var nBalcwind_X= this.RandX + 2.82102;

var nBalcwind_Y = 11.074193;

for(i=0; i<=2; i++){

    if(oBalconyWind){

        var cloneBalconyWind_2nd_flr = Instantiate(oBalconyWind,Vector3(nBalcwind_X - 1.45, nBalcwind_Y- 5.0, Balcwind_Z), Quaternion.identity);
```

```
                cloneBalconyWind_2nd_flr.transform.parent = oBuilding2.transform; //parenting

                cloneBalconyWind_2nd_flr.name = "2nd floor balcony window " + (++c);

                cloneBalconyWind_2nd_flr.transform.localScale.x = 0.109;

                cloneBalconyWind_2nd_flr.transform.localScale.y = 0.176753;

                cloneBalconyWind_2nd_flr.transform.localScale.z = 0.019;

                aCollectionObj_2.Add(cloneBalconyWind_4th_flr);

                nBalcwind_X -= 1.45;

            }//end if

        }//end for

        c = 0;


        //::::::::::::::::::Start Balcony::::::::::::::::::::::::::::::::::::::::::::::::

        oAttic = GameObject.CreatePrimitive(PrimitiveType.Cube);

        //solving an issue appearing: destroy visible instances appear as common cubes on the scene

        Destroy (oAttic);


        oAttic.renderer.material.color = Color(1, 0.980, 0.804, 0);


        var balcny_Y: float =  9.904193;

        if(oAttic){

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

                var balcony_flat = Instantiate(oAttic,Vector3(this.RandX, balcny_Y, wind_Z + 1.4 ), Quaternion.identity);

                balcony_flat.transform.parent = oBuilding2.transform; //parenting

                balcony_flat.name = "balcony flat " + (++c);
```

```
                balcony_flat.transform.localScale.x = 0.55;
                balcony_flat.transform.localScale.y = 0.0125;
                balcony_flat.transform.localScale.z = 0.128;
                aCollectionObj_2.Add(balcony_flat);
                balcny_Y -= 5.00;
        }//end for
}//end if
c = 0;
//:::::::::::::::::End Balcony::::::::::::::::::::::::::::::::::::::::::::::::

//::::Start Rails::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
oRails= GameObject.CreatePrimitive(PrimitiveType.Cube);
Destroy (oRails);

oRails.renderer.material.color = Color.white;

var rail_Y:float =  10.604193;
var raildn_Y:float = 10.004193;
wind_Z = this.RandZ + 4.32102;
var nRail_updn_X:float = this.RandX + 1.82102;
var nRail_lfup_lfdn_X:float = this.RandX + 1.83102;
var nRail_rgup_rgdn_X:float = this.RandX + 1.730;

//::::::::Start front Rails::::::::::::::::::::::::::::::::::::::::::::::::::::::

for(i=0; i<=1; i++){
        if(oRails){
```

```
                var rail_up = Instantiate(oRails,Vector3(nRail_updn_X -
1.8195569, rail_Y, wind_Z), Quaternion.identity);
                rail_up.transform.parent = oBuilding2.transform; //parenting
                rail_up.name = "rail up " + (++c);
                rail_up.transform.localScale.x = 0.5539678;
                rail_up.transform.localScale.y = 0.002;
                rail_up.transform.localScale.z = 0.002;
                aCollectionObj_2.Add(rail_up);

                var rail_down = Instantiate(oRails,Vector3(nRail_updn_X -
1.8195569, raildn_Y, wind_Z), Quaternion.identity);
                rail_down.transform.parent = oBuilding2.transform; //parenting
                rail_down.name = "rail down " + (++c);
                rail_down.transform.localScale.x = 0.5539678;
                rail_down.transform.localScale.y = 0.002;
                rail_down.transform.localScale.z = 0.002;
                aCollectionObj_2.Add(rail_down);

                rail_Y -= 5;
                raildn_Y -= 5;
            }//end if
        }//end for
        c = 0;

        nRail_updn_X = this.RandX + 1.82102;
        rail_Y = 10.604193;
        raildn_Y = 10.004193;
```

```
for(i=0; i<=16; i++){
    for(j=0; j<=1; j++){
        if(oRails){
            var rail_lfup = Instantiate(oRails,Vector3(nRail_lfup_lfdn_X, rail_Y - 0.18, wind_Z), Quaternion.identity);
            rail_lfup.transform.parent = oBuilding2.transform; //parenting
            rail_lfup.name = "rail left up " + (++c);
            rail_lfup.transform.localScale.x = 0.030;
            rail_lfup.transform.localScale.y = 0.002;
            rail_lfup.transform.localScale.z = 0.002;
            rail_lfup.transform.eulerAngles.z = 310;
            aCollectionObj_2.Add(rail_lfup);


            var rail_rgup = Instantiate(oRails,Vector3(nRail_rgup_rgdn_X, rail_Y - 0.18, wind_Z), Quaternion.identity);
            rail_rgup.transform.parent = oBuilding2.transform; //parenting
            rail_rgup.name = "rail right up " + (++c);
            rail_rgup.transform.localScale.x = 0.030;
            rail_rgup.transform.localScale.y = 0.002;
            rail_rgup.transform.localScale.z = 0.002;
            rail_rgup.transform.eulerAngles.z = -310;
            aCollectionObj_2.Add(rail_rgup);
```

```
                var rail_lfdn = Instantiate(oRails,Vector3(nRail_lfup_lfdn_X, rail_Y - 0.45, wind_Z), Quaternion.identity);

                rail_lfdn.transform.parent = oBuilding2.transform; //parenting

                rail_lfdn.name = "rail left down " + (++c);

                rail_lfdn.transform.localScale.x = 0.030;

                rail_lfdn.transform.localScale.y = 0.002;

                rail_lfdn.transform.localScale.z = 0.002;

                rail_lfdn.transform.eulerAngles.z = -310;

                aCollectionObj_2.Add(rail_lfdn);



                var rail_rgdn = Instantiate(oRails,Vector3(nRail_rgup_rgdn_X, rail_Y - 0.45, wind_Z), Quaternion.identity);

                rail_rgdn.transform.parent = oBuilding2.transform; //parenting

                rail_rgdn.name = "rail right down " + (++c);

                rail_rgdn.transform.localScale.x = 0.030;

                rail_rgdn.transform.localScale.y = 0.002;

                rail_rgdn.transform.localScale.z = 0.002;

                rail_rgdn.transform.eulerAngles.z = 310;

                aCollectionObj_2.Add(rail_rgdn);



                rail_Y -= 5;

                raildn_Y -= 5;

            }//end if
```

```
            }//end for
            rail_Y =  10.604193;
            raildn_Y = 10.004193;
            nRail_updn_X -= 0.22102;
            nRail_lfup_lfdn_X -= 0.22102;
            nRail_rgup_rgdn_X -= 0.22102;
        }//end for
        c = 0;
        //::::::::End front Rails::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

            //::::::::Start left/right up/down Main Side Rails::::::::::::::::::::::::::::::::::::::::::::::::::::::::
            var rail_Z = this.RandZ + 4.32102;
            nRail_updn_X = this.RandX + 1.92102;
            nRail_lfup_lfdn_X = this.RandX + 1.88102;
            nRail_rgup_rgdn_X = this.RandX + 1.780;

            for(i=0; i<=1; i++){
                for(j=0; j<=1; j++){
                    if(oRails){
                        var rail_up_left = Instantiate(oRails,Vector3(nRail_updn_X , rail_Y, rail_Z -0.4), Quaternion.identity);
                        rail_up_left.transform.parent = oBuilding2.transform; //parenting
                        rail_up_left.name = "rail up left " + (++c);
                        rail_up_left.transform.localScale.x = 0.1189484;
                        rail_up_left.transform.localScale.y = 0.002;
                        rail_up_left.transform.localScale.z = 0.002;
```

```
                                rail_up_left.transform.eulerAngles.y = 90;
                                aCollectionObj_2.Add(rail_up_left);

                                var rail_down_left = Instantiate(oRails,Vector3(nRail_updn_X, raildn_Y, rail_Z -0.4), Quaternion.identity);
                                rail_down_left.transform.parent = oBuilding2.transform; //parenting
                                rail_down_left.name = "rail down left " + (++c);
                                rail_down_left.transform.localScale.x = 0.1189484;
                                rail_down_left.transform.localScale.y = 0.002;
                                rail_down_left.transform.localScale.z = 0.002;
                                rail_down_left.transform.eulerAngles.y = 90;
                                aCollectionObj_2.Add(rail_down_left);

                                rail_Y -= 5;
                                raildn_Y -= 5;
                            }//end if
                        }//end for
                        rail_Y =  10.604193;
                        raildn_Y = 10.004193;
                        nRail_updn_X -= 3.85102;
                    }//end for
                    c = 0;

                    rail_Z = this.RandZ + 4.12102;
                    var rail_Z_rg = this.RandZ + 4.22204;
                    nRail_updn_X = this.RandX + 1.92102;
```

```
nRail_lfup_lfdn_X = this.RandX + 1.92102;
nRail_rgup_rgdn_X = this.RandX + 1.92102;

for(i=0; i<=3; i++){
    for(j=0; j<=1; j++){
        if(oRails){
            var rail_lfup_left = Instantiate(oRails,Vector3(nRail_lfup_lfdn_X, rail_Y - 0.18, rail_Z), Quaternion.identity);
            rail_lfup_left.transform.parent = oBuilding2.transform; //parenting
            rail_lfup_left.name = "rail left up left (left side)" + (++c);
            rail_lfup_left.transform.localScale.x = 0.030;
            rail_lfup_left.transform.localScale.y = 0.002;
            rail_lfup_left.transform.localScale.z = 0.002;
            rail_lfup_left.transform.eulerAngles.z = 310;
            rail_lfup_left.transform.eulerAngles.y = 90;
            aCollectionObj_2.Add(rail_lfup_left);

            var rail_rgup_left = Instantiate(oRails,Vector3(nRail_rgup_rgdn_X, rail_Y - 0.18, rail_Z_rg), Quaternion.identity);
            rail_rgup_left.transform.parent = oBuilding2.transform; //parenting
            rail_rgup_left.name = "rail right up left (left side)" + (++c);
            rail_rgup_left.transform.localScale.x = 0.030;
            rail_rgup_left.transform.localScale.y = 0.002;
```

```
rail_rgup_left.transform.localScale.z = 0.002;
rail_rgup_left.transform.eulerAngles.z = -310;
rail_rgup_left.transform.eulerAngles.y = 90;
aCollectionObj_2.Add(rail_rgup_left);

var rail_lfdn_left = Instantiate(oRails,Vector3(nRail_lfup_lfdn_X, rail_Y - 0.45, rail_Z), Quaternion.identity);
rail_lfdn_left.transform.parent = oBuilding2.transform; //parenting
rail_lfdn_left.name = "rail left down left (left side)" + (++c);
rail_lfdn_left.transform.localScale.x = 0.030;
rail_lfdn_left.transform.localScale.y = 0.002;
rail_lfdn_left.transform.localScale.z = 0.002;
rail_lfdn_left.transform.eulerAngles.z = -310;
rail_lfdn_left.transform.eulerAngles.y = 90;
aCollectionObj_2.Add(rail_lfdn_left);

var rail_rgdn_left = Instantiate(oRails,Vector3(nRail_rgup_rgdn_X, rail_Y - 0.45, rail_Z_rg), Quaternion.identity);
rail_rgdn_left.transform.parent = oBuilding2.transform; //parenting
rail_rgdn_left.name = "rail right down left (left side)" + (++c);
rail_rgdn_left.transform.localScale.x = 0.030;
rail_rgdn_left.transform.localScale.y = 0.002;
```

```
                        rail_rgdn_left.transform.localScale.z = 0.002;
                        rail_rgdn_left.transform.eulerAngles.z = 310;
                        rail_rgdn_left.transform.eulerAngles.y = 90;
                        aCollectionObj_2.Add(rail_rgdn_left);


                        rail_Y -= 5;
                        raildn_Y -= 5;
                    }//end if
                }//end for
                rail_Y =  10.604193;
                raildn_Y = 10.004193;
                rail_Z -= 0.22102;
                rail_Z_rg -= 0.22102;
            }//end for
            c = 0;


            rail_Z = this.RandZ + 4.12102;
            rail_Z_rg = this.RandZ + 4.22204;
            nRail_updn_X = this.RandX - 1.82102;
            nRail_lfup_lfdn_X = this.RandX - 1.92102;
            nRail_rgup_rgdn_X = this.RandX - 1.92102;


            for(i=0; i<=3; i++){
                for(j=0; j<=1; j++){
                    if(oRails){
                        var rail_lfup_right =
Instantiate(oRails,Vector3(nRail_lfup_lfdn_X, rail_Y - 0.18, rail_Z),
Quaternion.identity);
```

```
rail_lfup_right.transform.parent = oBuilding2.transform; //parenting

rail_lfup_right.name = "rail left up left (right side)" + (++c);

rail_lfup_right.transform.localScale.x = 0.030;

rail_lfup_right.transform.localScale.y = 0.002;

rail_lfup_right.transform.localScale.z = 0.002;

rail_lfup_right.transform.eulerAngles.z = 310;

rail_lfup_right.transform.eulerAngles.y = 90;

aCollectionObj_2.Add(rail_lfup_right);



var rail_rgup_right = Instantiate(oRails,Vector3(nRail_rgup_rgdn_X, rail_Y - 0.18, rail_Z_rg), Quaternion.identity);

rail_rgup_right.transform.parent = oBuilding2.transform; //parenting

rail_rgup_right.name = "rail right up left (right side)" + (++c);

rail_rgup_right.transform.localScale.x = 0.030;

rail_rgup_right.transform.localScale.y = 0.002;

rail_rgup_right.transform.localScale.z = 0.002;

rail_rgup_right.transform.eulerAngles.z = -310;

rail_rgup_right.transform.eulerAngles.y = 90;

aCollectionObj_2.Add(rail_rgup_right);



var rail_lfdn_right = Instantiate(oRails,Vector3(nRail_lfup_lfdn_X, rail_Y - 0.45, rail_Z), Quaternion.identity);
```

```
                    rail_lfdn_right.transform.parent = oBuilding2.transform; //parenting

                    rail_lfdn_right.name = "rail left down left (right side)" + (++c);

                    rail_lfdn_right.transform.localScale.x = 0.030;

                    rail_lfdn_right.transform.localScale.y = 0.002;

                    rail_lfdn_right.transform.localScale.z = 0.002;

                    rail_lfdn_right.transform.eulerAngles.z = -310;

                    rail_lfdn_right.transform.eulerAngles.y = 90;

                    aCollectionObj_2.Add(rail_lfdn_right);



                    var rail_rgdn_right = Instantiate(oRails,Vector3(nRail_rgup_rgdn_X, rail_Y - 0.45, rail_Z_rg), Quaternion.identity);

                    rail_rgdn_right.transform.parent = oBuilding2.transform; //parenting

                    rail_rgdn_right.name = "rail right down left (right side)" + (++c);

                    rail_rgdn_right.transform.localScale.x = 0.030;

                    rail_rgdn_right.transform.localScale.y = 0.002;

                    rail_rgdn_right.transform.localScale.z = 0.002;

                    rail_rgdn_right.transform.eulerAngles.z = 310;

                    rail_rgdn_right.transform.eulerAngles.y = 90;

                    aCollectionObj_2.Add(rail_rgdn_right);


                    rail_Y -= 5;

                    raildn_Y -= 5;

                }//end if
```

```
            }//end for
            rail_Y =  10.604193;
            raildn_Y = 10.004193;
            rail_Z -= 0.22102;
            rail_Z_rg -= 0.22102;
        }//end for
        c = 0;

        //::::::::End left/right up/down Main Side Rails::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
    //:::End Rails::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

}//end Balcony(RandX, RandZ)
//----------------------END  BALCONY-----------------------------------------------------------------------------------



//-----------------------START DOORS-------------------------------------------------------
public function Doors(RandX, RandZ){
    oDoor = GameObject.CreatePrimitive(PrimitiveType.Cube);

    //oDoor.transform.renderer.material.color = Color.clear;

    oDoor.renderer.material.mainTexture = Resources.Load("door_corfu2"); //import texture for doors
    //Positioning the oDoor object
    var door_X = this.RandX + 2.82102;
```

```
        var door_Z = this.RandZ + 3.5051424;
        var Balcwind_Z = this.RandZ + 3.4551424;
        oDoor.transform.position = Vector3(door_X, 1.35569, Balcwind_Z);

        oDoor.transform.parent = oBuilding2.transform;
        oDoor.name = "Door 0";
        //Scaling the oDoor object
        oDoor.transform.localScale.x = 0.122;
        oDoor.transform.localScale.y = 0.176753;
        oDoor.transform.localScale.z = 0.019;

        for(i=0; i<=1; i++){
                if(oDoor){
                        var cloneDoor = Instantiate(oDoor, Vector3(door_X - 2.89, 1.35569, door_Z), Quaternion.identity);
                        cloneDoor.transform.parent = oBuilding2.transform; //parenting
                        cloneDoor.name = "door " + (++c);
                        cloneDoor.transform.localScale.x = 0.122;
                        cloneDoor.transform.localScale.y = 0.176753;
                        cloneDoor.transform.localScale.z = 0.019;
                        aCollectionObj_2.Add(cloneDoor);
                        door_X -= 2.89;
                }//end if
        }//end for
        c = 0;
}//end Doors(RandX, RandZ)
```

```
//-----------------------------------END DOORS-----------------------------------------------------
---


//---------------------------------------START  ROOF-------------------------------------------------
-----
//Attic Roof
public function AtticRoof(RandX, RandZ){
        var tmpobj: GameObject = new GameObject();
        var mf: MeshFilter = tmpobj.AddComponent(MeshFilter);

//      tmpobj.AddComponent(MeshFilter);
        tmpobj.AddComponent(MeshRenderer);
        tmpobj.name = "Attic Roof";
        tmpobj.transform.parent = oBuilding2.transform;

        var verts: Vector3[] = new Vector3[6];
        var normals: Vector3[] = new Vector3[6];
        var uv: Vector2[] = new Vector2[6];
        var tri: int[] = new int[18]; //3 vertices * 6 triangles =18


        //vertices positioning
        verts[0] = new Vector3(this.RandX, 14.4, this.RandZ + 3.6); //P1
        verts[1] = new Vector3(this.RandX - 1.2, 14.1, this.RandZ + 3.6); //P2
        verts[2] = new Vector3(this.RandX + 1.2, 14.1, this.RandZ + 3.6); //P3

        verts[3] = new Vector3(this.RandX, 14.4, this.RandZ + 2); //P4
        verts[4] = new Vector3(this.RandX + 1.2, 14.1, this.RandZ + 2); //P5
```

```
verts[5] = new Vector3(this.RandX - 1.2, 14.1, this.RandZ + 2); //P6


uv[0] = new Vector2(0.5, 0);
uv[1] = new Vector2(1, 0);
uv[2] = new Vector2(0, 0);

uv[3] = new Vector2(0.5, 1);
uv[4] = new Vector2(0, 1);
uv[5] = new Vector2(1, 1);


//triangles
tri[0] = 0; //refer to vertices: verts[0]
tri[1] = 1; //refer to vertices: verts[1]
tri[2] = 2; //refer to vertices: verts[2]

tri[3] = 0;
tri[4] = 2;
tri[5] = 4;

tri[6] = 3;
tri[7] = 0;
tri[8] = 4;

tri[9] = 3;
tri[10] = 4;
```

```
        tri[11] = 5;

        tri[12] = 3;
        tri[13] = 5;
        tri[14] = 0;

        tri[15] = 0;
        tri[16] = 5;
        tri[17] = 1;

        var mesh: Mesh = new Mesh();
        mesh.name = "Attic Roof";
        mesh.vertices = verts;
        mesh.triangles = tri;
        mesh.uv = uv;
        mesh.normals = normals;
        mesh.RecalculateNormals();
        mf.mesh = mesh;
        tmpobj.renderer.material.mainTexture =
Resources.Load("tiled_roof_red_grey");
}//end function AtticRoof(RandX, RandZ)

//Main Roof
public function MainRoof(RandX, RandZ){
        var tmpobj: GameObject = new GameObject();
        var mf: MeshFilter = tmpobj.AddComponent(MeshFilter);
```

```
//      tmpobj.AddComponent(MeshFilter);
        tmpobj.AddComponent(MeshRenderer);
        tmpobj.name = "Main Roof";
        tmpobj.transform.parent = oBuilding2.transform;

        var verts: Vector3[] = new Vector3[6];
        var normals: Vector3[] = new Vector3[6];
        var uv: Vector2[] = new Vector2[6];
        var tri: int[] = new int[18]; //3 vertices * 6 triangles =18


        //vertices positioning
        verts[0] = new Vector3(this.RandX - 3.5, 14, this.RandZ + 0.5); //P1
        verts[1] = new Vector3(this.RandX - 3.5, 12.7, this.RandZ - 4); //P2
        verts[2] = new Vector3(this.RandX - 3.5, 12.7, this.RandZ + 4); //P3

        verts[3] = new Vector3(this.RandX + 3.5, 14, this.RandZ + 0.5); //P4
        verts[4] = new Vector3(this.RandX + 3.5, 12.7, this.RandZ + 4); //P5
        verts[5] = new Vector3(this.RandX + 3.5, 12.7, this.RandZ - 4); //P6


        uv[0] = new Vector2(0, 0.5);
        uv[1] = new Vector2(1, 1);
        uv[2] = new Vector2(1, 0);

        uv[3] = new Vector2(1, 0.5);
        uv[4] = new Vector2(0, 0);
```

```
uv[5] = new Vector2(0, 1);



//triangles
tri[0] = 0; //refer to vertices: verts[0]
tri[1] = 1; //refer to vertices: verts[1]
tri[2] = 2; //refer to vertices: verts[2]

tri[3] = 0;
tri[4] = 2;
tri[5] = 4;

tri[6] = 3;
tri[7] = 0;
tri[8] = 4;

tri[9] = 3;
tri[10] = 4;
tri[11] = 5;

tri[12] = 3;
tri[13] = 5;
tri[14] = 0;

tri[15] = 0;
tri[16] = 5;
tri[17] = 1;
```

```
        var mesh: Mesh = new Mesh();
        mesh.name = "Main Roof";
        mesh.vertices = verts;
        mesh.triangles = tri;
        mesh.uv = uv;
        mesh.normals = normals;
        mesh.RecalculateNormals();
        mf.mesh = mesh;
        tmpobj.renderer.material.mainTexture = Resources.Load("tiled_roof_red_grey");
}//end function  MainRoof(RandX, RandZ)
//---------------------------------------END ROOF-----------------------------------------------
```

//House_3.js

```
//create objects
var oBuilding3 : GameObject;
var oBase : GameObject;
var oWindow : GameObject;
var oColumn : GameObject;
var oColumnCube : GameObject;
var oEavesFlr_scnd : GameObject;
var oFrnt_Wall : GameObject;
var tmpobj: GameObject;

//------------------start public variables-------------------------------------------------
public var aCollectionObj_3 = new Array(); // an Array - a collection- for the objects of House_3
```

```
public var i: int = 0;
public var j: int = 0;
public var c: int = 0;

//Random ranges between X, Y coordinates
public var RandX: float;
public var RandZ: float;
//------------------end public variables----------------------------------------------------

//MAIN function
public function mainDO() {

        //RandX = Random.Range(10,247);
        //RandZ = Random.Range(10,247);

        MainBuilding(RandX, RandZ);
        Basement(RandX, RandZ);
        Stairs(RandX, RandZ);
        Attic(RandX, RandZ);
        Windows(RandX, RandZ);
        Columns(RandX, RandZ);
        Eaves(RandX, RandZ);
        Balcony(RandX, RandZ);
        Doors(RandX, RandZ);
        Roof(RandX, RandZ);
        DoArch(RandX, RandZ);
}//end function Start()
```

```
public function Destroy_(){

DestroyImmediate(oBuilding3);

}

public function MainBuilding(RandX, RandZ){
        oBuilding3 = GameObject.CreatePrimitive(PrimitiveType.Cube);
        oBuilding3.transform.position = Vector3(this.RandX, 3, this.RandZ);
        oBuilding3.name = "House_3";
        oBuilding3.renderer.material.color = Color(Random.Range(1,0.9),
Random.Range(0.980,0.880), Random.Range(0.804,0.704), 0);
        oBuilding3.transform.localScale.x = 7;
        oBuilding3.transform.localScale.y = 4.5;
        oBuilding3.transform.localScale.z = 5;
        aCollectionObj_3.Add(oBuilding3);
}//end function MainBuilding(RandX, RandZ)

//--------------------------START BASEMENT-----------------------------------------------------
-------------
public function Basement(RandX, RandZ){
//:::Start
Base::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
        oBase = GameObject.CreatePrimitive(PrimitiveType.Cube);

        oBase.renderer.material.color = Color(1, 0.980, 0.804, 0);
```

```
        oBase.transform.position = Vector3(this.RandX +2.5532604, 0.4, this.RandZ +0.6);

        oBase.transform.parent = oBuilding3.transform; //parenting

        oBase.name = "base left";

        oBase.transform.localScale.x = 0.2686374;

        oBase.transform.localScale.y = 0.15;

        oBase.transform.localScale.z = 1.25;

        aCollectionObj_3.Add(oBase);



        var oBase_right = Instantiate(oBase, Vector3(this.RandX -2.5591145, 0.4, this.RandZ +0.6), Quaternion.identity);

        oBase_right.transform.parent = oBuilding3.transform; //parenting

        oBase_right.name = "base right";

        oBase_right.transform.localScale.x = 0.2686374;

        oBase_right.transform.localScale.y = 0.15;

        oBase_right.transform.localScale.z = 1.25;

        aCollectionObj_3.Add(oBase_right);


        var oBase_middle = Instantiate(oBase, Vector3(this.RandX, 0.4, this.RandZ +0.6), Quaternion.identity);

        oBase_middle.transform.parent = oBuilding3.transform; //parenting

        oBase_middle.name = "base middle";

        oBase_middle.transform.localScale.x = 0.2686374;

        oBase_middle.transform.localScale.y = 0.15;

        oBase_middle.transform.localScale.z = 1.25;

        aCollectionObj_3.Add(oBase_middle);

} //end function Basement(RandX, RandZ)
```

```
//:::End
Base::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::


//:::Start
Stairs::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

public function Stairs(RandX, RandZ){

        var nextPOSX_stairs:float= this.RandX+ 1.26441663;

        var nextPOSY_stairs:float= 0.25;

        var nextPOSZ_stairs:float = this.RandZ +0.6;

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

                for(j=0; j<=1; j++){

                        if(oBase){

                                var oStairs = Instantiate(oBase,
Vector3(nextPOSX_stairs, nextPOSY_stairs, nextPOSZ_stairs),
Quaternion.identity);

                                oStairs.transform.parent = oBuilding3.transform;
//parenting

                                oStairs.transform.localScale.x = 0.10;

                                oStairs.transform.localScale.y = 0.08;

                                oStairs.transform.localScale.z = 1.25;

                                oStairs.name = "stairs " + (++c);

                                aCollectionObj_3.Add(oStairs);

                                nextPOSY_stairs += 0.30;

                                nextPOSZ_stairs = this.RandZ +0.46;

                        }//end if

                }//end for

                nextPOSX_stairs -= 2.5605403;

                nextPOSY_stairs = 0.25;

                nextPOSZ_stairs = this.RandZ +0.6;
```

```
        }//end for

        c = 0;

}//end function Stairs(RandX, RandZ)

//:::End
Stairs:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

//--------------------------END BASEMENT-----------------------------------------------------------
----------



public function Attic(RandX, RandZ){

        oBase = GameObject.CreatePrimitive(PrimitiveType.Cube);

        //solving an issue appearing: destroy visible instances appear as common
cubes on the scene

        Destroy (oBase);


        oBase.renderer.material.color = Color(1, 0.980, 0.804, 0);


        var oAttic = Instantiate(oBase, Vector3(this.RandX, 6.35, this.RandZ + 0.6),
Quaternion.identity);

        oAttic.transform.parent = oBuilding3.transform; //parenting

        oAttic.transform.localScale.x = 0.9954391;

        oAttic.transform.localScale.y = 0.504294;

        oAttic.transform.localScale.z = 1.252738;

        oAttic.name = "attic";

        aCollectionObj_3.Add(oAttic);

}//end function Attic(RandX, RandZ)
```

```
//-------------------------START WINDOWS-----------------------------------------------------------
public function Windows(RandX, RandZ){
	oWindow = GameObject.CreatePrimitive(PrimitiveType.Cube);

	//transparent windows
	//oWindow.transform.renderer.material.color = Color.clear;

	oWindow.renderer.material.mainTexture = Resources.Load("window_corfu2"); //import texture for windows

	var nWind_X:float = this.RandX + 2.82102;
	var nWind_Y:float = 6.6530576;
	var nWind_Z:float = this.RandZ + 3.7562495;

	oWindow.transform.position = Vector3(nWind_X, nWind_Y, nWind_Z);
	oWindow.transform.parent = oBuilding3.transform; //parenting
	oWindow.name = "Window 0";
	oWindow.transform.localScale.x = 0.1;
	oWindow.transform.localScale.y = 0.2402316;
	oWindow.transform.localScale.z = 0.0163088;
	aCollectionObj_3.Add(oWindow);

//Insert the rest 4 windows
	var nextPOSX_wind:float = nWind_X - 1.4;

	for(i=0; i<=3; i++){
		if(oWindow){
```

```
                var cloneWind_attic = Instantiate(oWindow, Vector3(nextPOSX_wind,nWind_Y, nWind_Z), Quaternion.identity);

                cloneWind_attic.transform.parent = oBuilding3.transform; //parenting

                cloneWind_attic.name = "Window attic " + (++c);

                cloneWind_attic.transform.localScale.x =0.1;

                cloneWind_attic.transform.localScale.y = 0.2402316;

                cloneWind_attic.transform.localScale.z = 0.0163088;

                aCollectionObj_3.Add(cloneWind_attic);

                nextPOSX_wind -= 1.4;

            }//end if

        }//end for

        c = 0;


//:::Start Side Windows::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

        var leftSideWind = Instantiate(oWindow, Vector3(this.RandX + 3.5079674, nWind_Y, this.RandZ + 3.1331445), Quaternion.identity); //left window

        leftSideWind.transform.parent = oBuilding3.transform; //parenting

        leftSideWind.name = "left window";

        leftSideWind.transform.localScale.x =0.1;

        leftSideWind.transform.localScale.y = 0.2402316;

        leftSideWind.transform.localScale.z = 0.0163088;

        leftSideWind.transform.eulerAngles.y = 90;

        aCollectionObj_3.Add(leftSideWind);


        var rightSideWind = Instantiate(oWindow, Vector3(this.RandX - 3.5079674, nWind_Y, this.RandZ + 3.1331445), Quaternion.identity); //right window

        rightSideWind.transform.parent = oBuilding3.transform; //parenting
```

```
rightSideWind.name = "right window";
rightSideWind.transform.localScale.x =0.1;
rightSideWind.transform.localScale.y = 0.2402316;
rightSideWind.transform.localScale.z = 0.0163088;
rightSideWind.transform.eulerAngles.y = 90;
aCollectionObj_3.Add(rightSideWind);

var leftSideWind_ScndFlr = Instantiate(oWindow, Vector3(this.RandX + 3.5079674, 4, this.RandZ + 1.8844446249915), Quaternion.identity); //left window
leftSideWind_ScndFlr.transform.parent = oBuilding3.transform; //parenting
leftSideWind_ScndFlr.name = "left window 2nd floor";
leftSideWind_ScndFlr.transform.localScale.x =0.1;
leftSideWind_ScndFlr.transform.localScale.y = 0.2402316;
leftSideWind_ScndFlr.transform.localScale.z = 0.0163088;
leftSideWind_ScndFlr.transform.eulerAngles.y = 90;
aCollectionObj_3.Add(leftSideWind_ScndFlr);

var rightSideWind_ScndFlr = Instantiate(oWindow, Vector3(this.RandX - 3.5079674, 4, this.RandZ + 1.8844446249915), Quaternion.identity); //right window
rightSideWind_ScndFlr.transform.parent = oBuilding3.transform; //parenting
rightSideWind_ScndFlr.name = "right window 2nd floor";
rightSideWind_ScndFlr.transform.localScale.x =0.1;
rightSideWind_ScndFlr.transform.localScale.y = 0.2402316;
rightSideWind_ScndFlr.transform.localScale.z = 0.0163088;
rightSideWind_ScndFlr.transform.eulerAngles.y = 90;
aCollectionObj_3.Add(rightSideWind_ScndFlr);
//:::End Side Windows:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
```

```
//::::Start Second Floor Windows::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

    var Scnd_FlrWind_POS_X:float = nWind_X ;


    for(i=0; i<=1; i++){
        if(oWindow){
            var cloneWind_sndFlr_left = Instantiate(oWindow, Vector3(Scnd_FlrWind_POS_X, 4, nWind_Z - 1.22), Quaternion.identity);
            cloneWind_sndFlr_left.transform.parent = oBuilding3.transform; //parenting
            cloneWind_sndFlr_left.name = "Window second floor left " + (++c);
            cloneWind_sndFlr_left.transform.localScale.x =0.1;
            cloneWind_sndFlr_left.transform.localScale.y = 0.2402316;
            cloneWind_sndFlr_left.transform.localScale.z = 0.0163088;
            aCollectionObj_3.Add(cloneWind_sndFlr_left);
            Scnd_FlrWind_POS_X -= 1.4;
        }//end if
    }//end for
    c = 0;


    Scnd_FlrWind_POS_X= nWind_X ;
    for(i=0; i<=1; i++){
        if(oWindow){
            var cloneWind_sndFlr_right = Instantiate(oWindow, Vector3(Scnd_FlrWind_POS_X - 5.6, 4, nWind_Z - 1.22), Quaternion.identity);
            cloneWind_sndFlr_right.transform.parent = oBuilding3.transform; //parenting
```

```
			cloneWind_sndFlr_right.name = "Window second floor right " + (++c);
			cloneWind_sndFlr_right.transform.localScale.x =0.1;
			cloneWind_sndFlr_right.transform.localScale.y = 0.2402316;
			cloneWind_sndFlr_right.transform.localScale.z = 0.0163088;
			aCollectionObj_3.Add(cloneWind_sndFlr_right);
			Scnd_FlrWind_POS_X += 1.4;
		}//end if
	}//end for
	c = 0;
//::::End Second Floor Windows::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

//::::Start Grnd Floor Windows::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
	Scnd_FlrWind_POS_X= nWind_X ;
	for(i=0; i<=1; i++){
		if(oWindow){
			var cloneWind_grndFlr = Instantiate(oWindow, Vector3(Scnd_FlrWind_POS_X -0.2, 2, nWind_Z - 1.22), Quaternion.identity);
			cloneWind_grndFlr.transform.parent = oBuilding3.transform; //parenting
			cloneWind_grndFlr.name = "Window ground floor " + (++c);
			cloneWind_grndFlr.transform.localScale.x =0.1;
			cloneWind_grndFlr.transform.localScale.y = 0.2402316;
			cloneWind_grndFlr.transform.localScale.z = 0.0163088;
			aCollectionObj_3.Add(cloneWind_grndFlr);
			Scnd_FlrWind_POS_X -= 2.6;
		}//end if
```

```
        }//end for

        c = 0;

//::::End                                    Grnd                                    Floor
Windows::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

}//end function Windows(RandX, RandZ)

//--------------------------END  WINDOWS-----------------------------------------------------
---------



//--------------------------------START  COLUMNS-----------------------------------------------
------------

public function Columns(RandX, RandZ){


        oColumn = GameObject.CreatePrimitive(PrimitiveType.Cube);


        oColumn.renderer.material.color = Color.white;


        oColumn.transform.position = Vector3(this.RandX -3.2, 1.4840475,
this.RandZ + 3.3772203);

        oColumn.transform.parent = oBuilding3.transform; //parenting

        oColumn.name = "Column right 0";

        oColumn.transform.localScale.x = 0.05992945;

        oColumn.transform.localScale.y = 0.2915616;

        oColumn.transform.localScale.z = 0.08529133;

        aCollectionObj_3.Add(oColumn);


        oColumnCube = GameObject.CreatePrimitive(PrimitiveType.Cube);
```

```
oColumnCube.transform.position = Vector3(this.RandX -3.2, 0.8373266, this.RandZ + 3.3772203);

oColumnCube.transform.parent = oColumn.transform; //parenting

oColumnCube.name = "Column base";

oColumnCube.transform.localScale.x = 1.43565;

oColumnCube.transform.localScale.y = 0.1184824;

oColumnCube.transform.localScale.z = 1.61189;

aCollectionObj_3.Add(oColumnCube);



var oColumnCube_Up = Instantiate(oColumnCube, Vector3(this.RandX -3.2, 2.2373266, this.RandZ + 3.3772203), Quaternion.identity);

oColumnCube_Up.transform.parent = oColumn.transform; //parenting

oColumnCube_Up.name = "Column top";

oColumnCube_Up.transform.localScale.x = 1.43565;

oColumnCube_Up.transform.localScale.y = 0.2184824;

oColumnCube_Up.transform.localScale.z = 1.61189;

aCollectionObj_3.Add(oColumnCube_Up);



var nextPOSX_column:float = this.RandX -3.2;

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

    if(oColumn){

        var cloneColumn_right = Instantiate(oColumn, Vector3(nextPOSX_column + 1.3, 1.4840475, this.RandZ + 3.3772203), Quaternion.identity);

        cloneColumn_right.transform.parent = oBuilding3.transform; //parenting

        cloneColumn_right.name = "Column right " + (++c);
```

```
                cloneColumn_right.transform.localScale.x = 0.05992945;
                cloneColumn_right.transform.localScale.y = 0.2915616;
                cloneColumn_right.transform.localScale.z = 0.08529133;
                aCollectionObj_3.Add(cloneColumn_right);
                if(i==1)
                        nextPOSX_column += 1.5;
                else
                        nextPOSX_column += 1.3;
        }//end if
    }//end for
    c = 0;

    nextPOSX_column = this.RandX;
    for(i=0; i<=2; i++){
        if(oColumn){
                var cloneColumn_left = Instantiate(oColumn,
Vector3(nextPOSX_column +0.6, 1.4840475, this.RandZ + 3.3772203),
Quaternion.identity);
                cloneColumn_left.transform.parent = oBuilding3.transform;
//parenting
                cloneColumn_left.name = "Column left " + (++c);
                cloneColumn_left.transform.localScale.x = 0.05992945;
                cloneColumn_left.transform.localScale.y = 0.2915616;
                cloneColumn_left.transform.localScale.z = 0.08529133;
                aCollectionObj_3.Add(cloneColumn_left);
                nextPOSX_column += 1.3;
        }//end if
    }//end for
```

```
c = 0;

//:::Start Second Floor Columns:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

nextPOSX_column = this.RandX -3.381428212338;

for(i=0; i<=2; i++){
    if(oColumn){
        var column_scndFlr_right = Instantiate(oColumn, Vector3(nextPOSX_column, 4.15, this.RandZ + 3.6272203), Quaternion.identity);
        column_scndFlr_right.transform.parent = oBuilding3.transform; //parenting
        column_scndFlr_right.name = "right column second floor " + (++c);
        column_scndFlr_right.transform.localScale.x = 0.02978393;
        column_scndFlr_right.transform.localScale.y = 0.1656943;
        column_scndFlr_right.transform.localScale.z = 0.040946;
        aCollectionObj_3.Add(column_scndFlr_right);
        nextPOSX_column += 1.3;
    }//end if
}//end for
c = 0;

for(i=0; i<=2; i++){
    if(oColumn){
        var column_scndFlr_left = Instantiate(oColumn, Vector3(nextPOSX_column + 0.25, 4.15, this.RandZ + 3.6272203), Quaternion.identity);
        column_scndFlr_left.transform.parent = oBuilding3.transform; //parenting
```

```
                column_scndFlr_left.name = "left column second floor " +
(++c);
                column_scndFlr_left.transform.localScale.x = 0.02978393;
                column_scndFlr_left.transform.localScale.y = 0.1656943;
                column_scndFlr_left.transform.localScale.z = 0.040946;
                aCollectionObj_3.Add(column_scndFlr_left);
                nextPOSX_column += 1.3;
            }//end if
        }//end for
        c = 0;
//:::End Second Floor
Columns::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
}//end function Columns(RandX, RandZ)
//--------------------------------END COLUMNS-------------------------------------------------
--------


//--------------------------------START EAVES-------------------------------------------------
-------
public function Eaves(RandX, RandZ){
        oEavesFlr_scnd = GameObject.CreatePrimitive(PrimitiveType.Cube);
        oEavesFlr_scnd.renderer.material.color = Color.white;


        oEavesFlr_scnd.transform.parent = oBuilding3.transform; //parenting
        oEavesFlr_scnd.name = "eaves 2nd floor";
        oEavesFlr_scnd.transform.position = Vector3(this.RandX, 5.2 , this.RandZ
+ 2.46482);
        oEavesFlr_scnd.transform.localScale.x = 1.0254391;
        oEavesFlr_scnd.transform.localScale.y = 0.025;
        oEavesFlr_scnd.transform.localScale.z = 0.525;
```

```
        aCollectionObj_3.Add(oEavesFlr_scnd);


        var oEavesFlr_frst = Instantiate(oEavesFlr_scnd, Vector3(this.RandX, 3.3 , this.RandZ + 2.46482), Quaternion.identity);

        oEavesFlr_frst.transform.parent = oBuilding3.transform; //parenting

        oEavesFlr_frst.name = "eaves 1st floor";

        oEavesFlr_frst.transform.localScale.x = 1.0254391;

        oEavesFlr_frst.transform.localScale.y = 0.025;

        oEavesFlr_frst.transform.localScale.z = 0.525;

        aCollectionObj_3.Add(oEavesFlr_frst);

}//end function Eaves(RandX, RandZ)

//--------------------------------END EAVES-------------------------------------------------------------

//--------------------------------START BALCONY---------------------------------------------------------

public function Balcony(RandX, RandZ){

        oFrnt_Wall = GameObject.CreatePrimitive(PrimitiveType.Cube);


        oFrnt_Wall.renderer.material.color = Color(1, 0.980, 0.804, 0);


        oFrnt_Wall.transform.parent = oBuilding3.transform; //parenting

        oFrnt_Wall.name = "balcony front wall";

        oFrnt_Wall.transform.position = Vector3(this.RandX, 3.55 , this.RandZ + 3.7112529943107);

        oFrnt_Wall.transform.localScale.x = 1.0254391;

        oFrnt_Wall.transform.localScale.y = 0.08933807;

        oFrnt_Wall.transform.localScale.z = 0.025;

        aCollectionObj_3.Add(oFrnt_Wall);
```

```
	var nextPOS_X_wall:float = this.RandX + 3.5001257969745;
	for(i=0; i<=1; i++){
		if(oFrnt_Wall){
			var oSide_Wall = Instantiate(oFrnt_Wall, Vector3(nextPOS_X_wall, 3.55, this.RandZ + 3.1331445), Quaternion.identity);
			oSide_Wall.transform.parent = oBuilding3.transform; //parenting
			oSide_Wall.name = "balcony side wall " + (++c);
			oSide_Wall.transform.localScale.x = 0.25;
			oSide_Wall.transform.localScale.y = 0.08933807;
			oSide_Wall.transform.localScale.z = 0.025;
			oSide_Wall.transform.eulerAngles.y = 90;
			aCollectionObj_3.Add(oSide_Wall);
			nextPOS_X_wall -= 7.004766394268;
		}//end if
	}//end for
	c = 0;
}//end function Balcony(RandX, RandZ)
//--------------------------------END  BALCONY-----------------------------------------------------------

//--------------------------------START  DOORS-----------------------------------------------------------
public function Doors(RandX, RandZ){
	oWindow = GameObject.CreatePrimitive(PrimitiveType.Cube);
```

```
//solving an issue appearing: destroy visible instances appear as common cubes on the scene

Destroy (oWindow);


//transparent windows

//oWindow.transform.renderer.material.color = Color.clear;

oWindow.renderer.material.mainTexture = Resources.Load("door_corfu2"); //import texture for doors


var nWind_X:float = this.RandX + 2.82102;

var nWind_Z:float = this.RandZ + 3.7562495;


if(oWindow){

        var oDoor_Scnd_Flr = Instantiate(oWindow, Vector3(nWind_X - 2.8, 4.2, nWind_Z - 1.22), Quaternion.identity);

        oDoor_Scnd_Flr.transform.parent = oBuilding3.transform; //parenting

        oDoor_Scnd_Flr.name = "Door second floor";

        oDoor_Scnd_Flr.transform.localScale.x = 0.1;

        oDoor_Scnd_Flr.transform.localScale.y = 0.3734118;

        oDoor_Scnd_Flr.transform.localScale.z = 0.0163088;

        aCollectionObj_3.Add(oDoor_Scnd_Flr);


        var dr_PosX:float = nWind_X - 1.6028176;

        for(i=0; i<=2; i++){

                var oGround_door = Instantiate(oDoor_Scnd_Flr, Vector3(dr_PosX, 1.5511089, nWind_Z - 1.22), Quaternion.identity);

                oGround_door.transform.parent = oBuilding3.transform; //parenting

                oGround_door.name = "ground door " + (++c);
```

```
                oGround_door.transform.localScale.x = 0.1;
                oGround_door.transform.localScale.y = 0.3734118;
                oGround_door.transform.localScale.z = 0.0163088;
                aCollectionObj_3.Add(oGround_door);
                if(i == 1)
                        dr_PosX -= 1.3;
                else
                        dr_PosX -= 2.5;
        }//end for
    }//end if
    c = 0;
}//end function Doors(RandX, RandZ)
//---------------------------------END DOORS-----------------------------------------------------
----

//---------------------------------------START  ROOF-------------------------------------------------
-----
public function Roof(RandX, RandZ){
    var tmpobj: GameObject = new GameObject();
    var mf: MeshFilter = tmpobj.AddComponent(MeshFilter);
    //Renderer renderer = GetComponent<MeshRenderer>().renderer;

    //tmpobj.AddComponent(MeshFilter);
    tmpobj.AddComponent(MeshRenderer);
    tmpobj.name = "Roof";
    tmpobj.transform.parent = oBuilding3.transform;

    var verts: Vector3[] = new Vector3[6];
```

```
var normals: Vector3[] = new Vector3[6];
var uv: Vector2[] = new Vector2[6];
var tri: int[] = new int[18]; //3 vertices * 6 triangles =18


//vertices positioning
verts[0] = new Vector3(this.RandX - 3.5, 8.5, this.RandZ + 0.5); //P1
verts[1] = new Vector3(this.RandX - 3.5, 7.5, this.RandZ - 2.6); //P2
verts[2] = new Vector3(this.RandX - 3.5, 7.5, this.RandZ + 4); //P3

verts[3] = new Vector3(this.RandX + 3.5, 8.5, this.RandZ + 0.5); //P4
verts[4] = new Vector3(this.RandX + 3.5, 7.5, this.RandZ + 4); //P5
verts[5] = new Vector3(this.RandX + 3.5, 7.5, this.RandZ - 2.6); //P6


uv[0] = new Vector2(0,0.5); //refer to vertices: verts[0]
uv[1] = new Vector2(1,1);
uv[2] = new Vector2(1,0);

uv[3] = new Vector2(1,0.5);
uv[4] = new Vector2(0,0);
uv[5] = new Vector2(0,1);


//triangles
tri[0] = 0; //refer to vertices: verts[0]
tri[1] = 1; //refer to vertices: verts[1]
```

```
tri[2] = 2; //refer to vertices: verts[2]

tri[3] = 0;
tri[4] = 2;
tri[5] = 4;

tri[6] = 3;
tri[7] = 0;
tri[8] = 4;

tri[9] = 3;
tri[10] = 4;
tri[11] = 5;

tri[12] = 3;
tri[13] = 5;
tri[14] = 0;

tri[15] = 0;
tri[16] = 5;
tri[17] = 1;

var mesh: Mesh = new Mesh();
mesh.name = "Roof";
mesh.vertices = verts;
mesh.triangles = tri;
mesh.uv = uv;
```

```
        mesh.normals = normals;
        mesh.RecalculateNormals();
        mf.mesh = mesh;
        tmpobj.renderer.material.mainTexture = Resources.Load("tiled_roof_red_grey");
        //Material mat = Resources.Load("tiled_roof_red_grey");
        //renderer.material = mat;
}//end function Roof(RandX, RandZ)
//--------------------------------------END ROOF-----------------------------------------------------

//-------------------------------------START ARCH-----------------------------------------------------
private function Arch(RandX, RandZ, tmpobj) {
        tmpobj = new GameObject();
        var mf: MeshFilter = tmpobj.AddComponent(MeshFilter);

        tmpobj.AddComponent(MeshRenderer);
        tmpobj.name = "Arch";
        tmpobj.renderer.material.color = Color(1, 0.980, 0.804, 0);
        tmpobj.transform.parent = oBuilding3.transform;

        var verts: Vector3[] = new Vector3[14];
        var uv: Vector2[] = new Vector2[14];
        var normals: Vector3[] = new Vector3[14];
        var tri: int[] = new int[72]; //3 vertices * 12 triangles = 36 * 2 (duplicate triangles for the back face) =72
```

```
//vertices positioning
verts[0] = new Vector3(this.RandX - 1.9, 2.5, this.RandZ + 3.777); //P1
verts[1] = new Vector3(this.RandX - 1.9, 2.35, this.RandZ + 3.777); //P2
verts[2] = new Vector3(this.RandX - 1.72, 2.35, this.RandZ + 3.777); //P3
verts[3] = new Vector3(this.RandX - 1.72, 3.3, this.RandZ + 3.777); //P4
verts[4] = new Vector3(this.RandX - 3.2, 3.3, this.RandZ + 3.777); //P5
verts[5] = new Vector3(this.RandX - 3.2, 2.35, this.RandZ + 3.777); //P6
verts[6] = new Vector3(this.RandX - 3.02, 2.35, this.RandZ + 3.777); //P7
verts[7] = new Vector3(this.RandX - 3.02, 2.5, this.RandZ + 3.777); //P8
verts[8] = new Vector3(this.RandX - 2.96, 2.7, this.RandZ + 3.777); //P9
verts[9] = new Vector3(this.RandX - 2.83, 2.9, this.RandZ + 3.777); //P10
verts[10] = new Vector3(this.RandX - 2.62, 3, this.RandZ + 3.777); //P11
verts[11] = new Vector3(this.RandX - 2.3,  3, this.RandZ + 3.777); //P12
verts[12] = new Vector3(this.RandX - 2.09, 2.9, this.RandZ + 3.777); //P13
verts[13] = new Vector3(this.RandX - 1.96, 2.7, this.RandZ + 3.777); //P14

//normals
normals[0] = new Vector3(0, 0, 1);
normals[1] = new Vector3(0, 0, 1);
normals[2] = new Vector3(0, 0, 1);
normals[3] = new Vector3(0, 0, 1);
normals[4] = new Vector3(0, 0, 1);
normals[5] = new Vector3(0, 0, 1);
normals[6] = new Vector3(0, 0, 1);
normals[7] = new Vector3(0, 0, 1);
normals[8] = new Vector3(0, 0, 1);
```

```
normals[9] = new Vector3(0, 0, 1);
normals[10] = new Vector3(0, 0, 1);
normals[11] = new Vector3(0, 0, 1);
normals[12] = new Vector3(0, 0, 1);
normals[13] = new Vector3(0, 0, 1);

//uv mapping (textures)
uv[0] = new Vector2(0, 0.3);
uv[1] = new Vector2(0.2, 0);
uv[2] = new Vector2(0, 0);
uv[3] = new Vector2(0, 1);
uv[4] = new Vector2(1, 1);
uv[5] = new Vector2(1, 0);
uv[6] = new Vector2(0.8, 0);
uv[7] = new Vector2(0.8, 0.3);
uv[8] = new Vector2(0.75, 0.5);
uv[9] = new Vector2(0.65, 0.7);
uv[10] = new Vector2(0.6, 0.9);
uv[11] = new Vector2(0.4, 0.9);
uv[12] = new Vector2(0.45, 0.7);
uv[13] = new Vector2(0.35, 0.5);

//triangles
tri[0] = 0; //refer to vertices: verts[0]
tri[1] = 1; //refer to vertices: verts[1]
tri[2] = 2; //refer to vertices: verts[2]
```

```
tri[3] = 0;
tri[4] = 2;
tri[5] = 3;

tri[6] = 0;
tri[7] = 3;
tri[8] = 13;

tri[9] = 13;
tri[10] = 3;
tri[11] = 12;

tri[12] = 12;
tri[13] = 3;
tri[14] = 11;

tri[15] = 11;
tri[16] = 3;
tri[17] = 4;

tri[18] = 11;
tri[19] = 4;
tri[20] = 10;

tri[21] = 10;
tri[22] = 4;
```

```
tri[23] = 9;

tri[24] = 9;
tri[25] = 4;
tri[26] = 8;

tri[27] = 8;
tri[28] = 4;
tri[29] = 7;

tri[30] = 7;
tri[31] = 4;
tri[32] = 5;

tri[33] = 7;
tri[34] = 5;
tri[35] = 6;

tri[36] = 0;
tri[37] = 2;
tri[38] = 1;

tri[39] = 0;
tri[40] = 3;
tri[41] = 2;

tri[42] = 13;
```

```
tri[43] = 3;
tri[44] = 0;

tri[45] = 12;
tri[46] = 3;
tri[47] = 13;

tri[48] = 11;
tri[49] = 3;
tri[50] = 12;

tri[51] = 10;
tri[52] = 3;
tri[53] = 11;

tri[54] = 10;
tri[55] = 4;
tri[56] = 3;

tri[57] = 9;
tri[58] = 4;
tri[59] = 10;

tri[60] = 8;
tri[61] = 4;
tri[62] = 9;
```

```
        tri[63] = 7;
        tri[64] = 4;
        tri[65] = 8;

        tri[66] = 7;
        tri[67] = 5;
        tri[68] = 4;

        tri[69] = 7;
        tri[70] = 6;
        tri[71] = 5;

        var mesh: Mesh = new Mesh();
        mesh.name = "Arch";
        mesh.vertices = verts;
        mesh.triangles = tri;
        mesh.uv = uv;
        mesh.normals = normals;
        //mesh.RecalculateNormals();
        tmpobj.renderer.material.mainTexture                                    =
Resources.Load("white_brick_wall");
        mf.mesh = mesh;
}//end function Arch(Rand_X, Rand_Z)


private function ArchVertical(RandX, RandZ, tmpobj){
```

```
tmpobj = new GameObject();
var mf: MeshFilter = tmpobj.AddComponent(MeshFilter);

tmpobj.AddComponent(MeshRenderer);
tmpobj.name = "Arch vert";
tmpobj.renderer.material.color = Color(1, 0.980, 0.804, 0);
tmpobj.transform.parent = oBuilding3.transform;

var verts: Vector3[] = new Vector3[14];
var uv: Vector2[] = new Vector2[14];
var normals: Vector3[] = new Vector3[14];
var tri: int[] = new int[72]; //3 vertices * 12 triangles = 36 * 2 (duplicate triangles for the back face) =72

//vertices positioning
verts[0] = new Vector3(this.RandX , 2.5, this.RandZ - 1.9); //P1
verts[1] = new Vector3(this.RandX , 2.35, this.RandZ - 1.9); //P2
verts[2] = new Vector3(this.RandX , 2.35, this.RandZ - 1.72); //P3
verts[3] = new Vector3(this.RandX , 3.3, this.RandZ - 1.72); //P4
verts[4] = new Vector3(this.RandX , 3.3, this.RandZ - 3.2); //P5
verts[5] = new Vector3(this.RandX , 2.35, this.RandZ - 3.2); //P6
verts[6] = new Vector3(this.RandX , 2.35, this.RandZ - 3.02); //P7
verts[7] = new Vector3(this.RandX , 2.5, this.RandZ - 3.02); //P8
verts[8] = new Vector3(this.RandX , 2.7, this.RandZ - 2.96); //P9
verts[9] = new Vector3(this.RandX , 2.9, this.RandZ - 2.83); //P10
verts[10] = new Vector3(this.RandX , 3, this.RandZ - 2.62); //P11
verts[11] = new Vector3(this.RandX ,  3, this.RandZ - 2.3); //P12
```

```
verts[12] = new Vector3(this.RandX , 2.9, this.RandZ - 2.09); //P13
verts[13] = new Vector3(this.RandX , 2.7, this.RandZ - 1.96); //P14

//normals
normals[0] = new Vector3(0, 0, 1);
normals[1] = new Vector3(0, 0, 1);
normals[2] = new Vector3(0, 0, 1);
normals[3] = new Vector3(0, 0, 1);
normals[4] = new Vector3(0, 0, 1);
normals[5] = new Vector3(0, 0, 1);
normals[6] = new Vector3(0, 0, 1);
normals[7] = new Vector3(0, 0, 1);
normals[8] = new Vector3(0, 0, 1);
normals[9] = new Vector3(0, 0, 1);
normals[10] = new Vector3(0, 0, 1);
normals[11] = new Vector3(0, 0, 1);
normals[12] = new Vector3(0, 0, 1);
normals[13] = new Vector3(0, 0, 1);

//uv mapping (textures)
uv[0] = new Vector2(0, 0.3);
uv[1] = new Vector2(0.2, 0);
uv[2] = new Vector2(0, 0);
uv[3] = new Vector2(0, 1);
uv[4] = new Vector2(1, 1);
uv[5] = new Vector2(1, 0);
uv[6] = new Vector2(0.8, 0);
```

```
uv[7] = new Vector2(0.8, 0.3);

uv[8] = new Vector2(0.75, 0.5);

uv[9] = new Vector2(0.65, 0.7);

uv[10] = new Vector2(0.6, 0.9);

uv[11] = new Vector2(0.4, 0.9);

uv[12] = new Vector2(0.45, 0.7);

uv[13] = new Vector2(0.35, 0.5);



//triangles

tri[0] = 0; //refer to vertices: verts[0]

tri[1] = 1; //refer to vertices: verts[1]

tri[2] = 2; //refer to vertices: verts[2]


tri[3] = 0;

tri[4] = 2;

tri[5] = 3;


tri[6] = 0;

tri[7] = 3;

tri[8] = 13;


tri[9] = 13;

tri[10] = 3;

tri[11] = 12;


tri[12] = 12;
```

```
tri[13] = 3;
tri[14] = 11;

tri[15] = 11;
tri[16] = 3;
tri[17] = 4;

tri[18] = 11;
tri[19] = 4;
tri[20] = 10;

tri[21] = 10;
tri[22] = 4;
tri[23] = 9;

tri[24] = 9;
tri[25] = 4;
tri[26] = 8;

tri[27] = 8;
tri[28] = 4;
tri[29] = 7;

tri[30] = 7;
tri[31] = 4;
tri[32] = 5;
```

```
tri[33] = 7;
tri[34] = 5;
tri[35] = 6;

tri[36] = 0;
tri[37] = 2;
tri[38] = 1;

tri[39] = 0;
tri[40] = 3;
tri[41] = 2;

tri[42] = 13;
tri[43] = 3;
tri[44] = 0;

tri[45] = 12;
tri[46] = 3;
tri[47] = 13;

tri[48] = 11;
tri[49] = 3;
tri[50] = 12;

tri[51] = 10;
tri[52] = 3;
tri[53] = 11;
```

```
tri[54] = 10;
tri[55] = 4;
tri[56] = 3;

tri[57] = 9;
tri[58] = 4;
tri[59] = 10;

tri[60] = 8;
tri[61] = 4;
tri[62] = 9;

tri[63] = 7;
tri[64] = 4;
tri[65] = 8;

tri[66] = 7;
tri[67] = 5;
tri[68] = 4;

tri[69] = 7;
tri[70] = 6;
tri[71] = 5;

var mesh: Mesh = new Mesh();
mesh.name = "Arch vert";
```

```
        mesh.vertices = verts;
        mesh.triangles = tri;
        mesh.uv = uv;
        mesh.normals = normals;
        //mesh.RecalculateNormals();
        tmpobj.renderer.material.mainTexture = Resources.Load("white_brick_wall");
        mf.mesh = mesh;

}//end function ArchVertical(Rand_X, Rand_Z, tmpobj)


private function ArchScndFlr(RandX, RandZ, tmpobj) {
        tmpobj = new GameObject();
        var mf: MeshFilter = tmpobj.AddComponent(MeshFilter);

        tmpobj.AddComponent(MeshRenderer);
        tmpobj.name = "Arch second floor";
        tmpobj.renderer.material.color = Color(1, 0.980, 0.804, 0);
        tmpobj.transform.parent = oBuilding3.transform;

        var verts: Vector3[] = new Vector3[14];
        var uv: Vector2[] = new Vector2[14];
        var normals: Vector3[] = new Vector3[14];
        var tri: int[] = new int[72]; //3 vertices * 12 triangles = 36 * 2 (duplicate triangles for the back face) =72

        //vertices positioning
```

```
verts[0] = new Vector3(this.RandX - 1.9, 4.5, this.RandZ); //P1
verts[1] = new Vector3(this.RandX - 1.9, 4.35, this.RandZ); //P2
verts[2] = new Vector3(this.RandX - 1.72, 4.35, this.RandZ ); //P3
verts[3] = new Vector3(this.RandX - 1.72, 5.3, this.RandZ); //P4
verts[4] = new Vector3(this.RandX - 3.2, 5.3, this.RandZ); //P5
verts[5] = new Vector3(this.RandX - 3.2, 4.35, this.RandZ); //P6
verts[6] = new Vector3(this.RandX - 3.02, 4.35, this.RandZ); //P7
verts[7] = new Vector3(this.RandX - 3.02, 4.5, this.RandZ); //P8
verts[8] = new Vector3(this.RandX - 2.96, 4.7, this.RandZ); //P9
verts[9] = new Vector3(this.RandX - 2.83, 4.9, this.RandZ); //P10
verts[10] = new Vector3(this.RandX - 2.62, 5, this.RandZ); //P11
verts[11] = new Vector3(this.RandX - 2.3,  5, this.RandZ); //P12
verts[12] = new Vector3(this.RandX - 2.09, 4.9, this.RandZ); //P13
verts[13] = new Vector3(this.RandX - 1.96, 4.7, this.RandZ); //P14

//normals
normals[0] = new Vector3(0, 0, 1);
normals[1] = new Vector3(0, 0, 1);
normals[2] = new Vector3(0, 0, 1);
normals[3] = new Vector3(0, 0, 1);
normals[4] = new Vector3(0, 0, 1);
normals[5] = new Vector3(0, 0, 1);
normals[6] = new Vector3(0, 0, 1);
normals[7] = new Vector3(0, 0, 1);
normals[8] = new Vector3(0, 0, 1);
normals[9] = new Vector3(0, 0, 1);
normals[10] = new Vector3(0, 0, 1);
```

```
normals[11] = new Vector3(0, 0, 1);

normals[12] = new Vector3(0, 0, 1);

normals[13] = new Vector3(0, 0, 1);


//uv mapping (textures)

uv[0] = new Vector2(0, 0.3);

uv[1] = new Vector2(0.2, 0);

uv[2] = new Vector2(0, 0);

uv[3] = new Vector2(0, 1);

uv[4] = new Vector2(1, 1);

uv[5] = new Vector2(1, 0);

uv[6] = new Vector2(0.8, 0);

uv[7] = new Vector2(0.8, 0.3);

uv[8] = new Vector2(0.75, 0.5);

uv[9] = new Vector2(0.65, 0.7);

uv[10] = new Vector2(0.6, 0.9);

uv[11] = new Vector2(0.4, 0.9);

uv[12] = new Vector2(0.45, 0.7);

uv[13] = new Vector2(0.35, 0.5);



//triangles

tri[0] = 0; //refer to vertices: verts[0]

tri[1] = 1; //refer to vertices: verts[1]

tri[2] = 2; //refer to vertices: verts[2]


tri[3] = 0;
```

```
tri[4] = 2;
tri[5] = 3;

tri[6] = 0;
tri[7] = 3;
tri[8] = 13;

tri[9] = 13;
tri[10] = 3;
tri[11] = 12;

tri[12] = 12;
tri[13] = 3;
tri[14] = 11;

tri[15] = 11;
tri[16] = 3;
tri[17] = 4;

tri[18] = 11;
tri[19] = 4;
tri[20] = 10;

tri[21] = 10;
tri[22] = 4;
tri[23] = 9;
```

```
tri[24] = 9;

tri[25] = 4;

tri[26] = 8;


tri[27] = 8;

tri[28] = 4;

tri[29] = 7;


tri[30] = 7;

tri[31] = 4;

tri[32] = 5;


tri[33] = 7;

tri[34] = 5;

tri[35] = 6;


tri[36] = 0;

tri[37] = 2;

tri[38] = 1;


tri[39] = 0;

tri[40] = 3;

tri[41] = 2;


tri[42] = 13;

tri[43] = 3;

tri[44] = 0;
```

```
tri[45] = 12;
tri[46] = 3;
tri[47] = 13;

tri[48] = 11;
tri[49] = 3;
tri[50] = 12;

tri[51] = 10;
tri[52] = 3;
tri[53] = 11;

tri[54] = 10;
tri[55] = 4;
tri[56] = 3;

tri[57] = 9;
tri[58] = 4;
tri[59] = 10;

tri[60] = 8;
tri[61] = 4;
tri[62] = 9;

tri[63] = 7;
tri[64] = 4;
```

```
        tri[65] = 8;

        tri[66] = 7;
        tri[67] = 5;
        tri[68] = 4;

        tri[69] = 7;
        tri[70] = 6;
        tri[71] = 5;

        var mesh: Mesh = new Mesh();
        mesh.name = "Arch second floor";
        mesh.vertices = verts;
        mesh.triangles = tri;
        mesh.uv = uv;
        mesh.normals = normals;
        //mesh.RecalculateNormals();
        tmpobj.renderer.material.mainTexture                                    =
Resources.Load("white_brick_wall");
        mf.mesh = mesh;
}//end function ArchScndFlr(Rand_X, Rand_Z)



private function ArchScndFlrVertical(RandX, RandZ, tmpobj){

        tmpobj = new GameObject();
```

```
var mf: MeshFilter = tmpobj.AddComponent(MeshFilter);


tmpobj.AddComponent(MeshRenderer);
tmpobj.name = "Arch second floor vert";
tmpobj.renderer.material.color = Color(1, 0.980, 0.804, 0);
tmpobj.transform.parent = oBuilding3.transform;


var verts: Vector3[] = new Vector3[14];
var uv: Vector2[] = new Vector2[14];
var normals: Vector3[] = new Vector3[14];
var tri: int[] = new int[72]; //3 vertices * 12 triangles = 36 * 2 (duplicate triangles for the back face) =72


//vertices positioning
verts[0] = new Vector3(this.RandX , 4.5, this.RandZ - 1.9); //P1
verts[1] = new Vector3(this.RandX , 4.35, this.RandZ - 1.9); //P2
verts[2] = new Vector3(this.RandX , 4.35, this.RandZ - 1.72); //P3
verts[3] = new Vector3(this.RandX , 5.3, this.RandZ - 1.72); //P4
verts[4] = new Vector3(this.RandX , 5.3, this.RandZ - 3.2); //P5
verts[5] = new Vector3(this.RandX , 4.35, this.RandZ - 3.2); //P6
verts[6] = new Vector3(this.RandX , 4.35, this.RandZ - 3.02); //P7
verts[7] = new Vector3(this.RandX , 4.5, this.RandZ - 3.02); //P8
verts[8] = new Vector3(this.RandX , 4.7, this.RandZ - 2.96); //P9
verts[9] = new Vector3(this.RandX , 4.9, this.RandZ - 2.83); //P10
verts[10] = new Vector3(this.RandX , 5, this.RandZ - 2.62); //P11
verts[11] = new Vector3(this.RandX ,  5, this.RandZ - 2.3); //P12
verts[12] = new Vector3(this.RandX , 4.9, this.RandZ - 2.09); //P13
```

```
verts[13] = new Vector3(this.RandX , 4.7, this.RandZ - 1.96); //P14

//normals
normals[0] = new Vector3(0, 0, 1);
normals[1] = new Vector3(0, 0, 1);
normals[2] = new Vector3(0, 0, 1);
normals[3] = new Vector3(0, 0, 1);
normals[4] = new Vector3(0, 0, 1);
normals[5] = new Vector3(0, 0, 1);
normals[6] = new Vector3(0, 0, 1);
normals[7] = new Vector3(0, 0, 1);
normals[8] = new Vector3(0, 0, 1);
normals[9] = new Vector3(0, 0, 1);
normals[10] = new Vector3(0, 0, 1);
normals[11] = new Vector3(0, 0, 1);
normals[12] = new Vector3(0, 0, 1);
normals[13] = new Vector3(0, 0, 1);

//uv mapping (textures)
uv[0] = new Vector2(0, 0.3);
uv[1] = new Vector2(0.2, 0);
uv[2] = new Vector2(0, 0);
uv[3] = new Vector2(0, 1);
uv[4] = new Vector2(1, 1);
uv[5] = new Vector2(1, 0);
uv[6] = new Vector2(0.8, 0);
uv[7] = new Vector2(0.8, 0.3);
```

```
uv[8] = new Vector2(0.75, 0.5);

uv[9] = new Vector2(0.65, 0.7);

uv[10] = new Vector2(0.6, 0.9);

uv[11] = new Vector2(0.4, 0.9);

uv[12] = new Vector2(0.45, 0.7);

uv[13] = new Vector2(0.35, 0.5);



//triangles

tri[0] = 0; //refer to vertices: verts[0]

tri[1] = 1; //refer to vertices: verts[1]

tri[2] = 2; //refer to vertices: verts[2]


tri[3] = 0;

tri[4] = 2;

tri[5] = 3;


tri[6] = 0;

tri[7] = 3;

tri[8] = 13;


tri[9] = 13;

tri[10] = 3;

tri[11] = 12;


tri[12] = 12;

tri[13] = 3;
```

```
tri[14] = 11;


tri[15] = 11;
tri[16] = 3;
tri[17] = 4;


tri[18] = 11;
tri[19] = 4;
tri[20] = 10;


tri[21] = 10;
tri[22] = 4;
tri[23] = 9;


tri[24] = 9;
tri[25] = 4;
tri[26] = 8;


tri[27] = 8;
tri[28] = 4;
tri[29] = 7;


tri[30] = 7;
tri[31] = 4;
tri[32] = 5;


tri[33] = 7;
```

```
tri[34] = 5;
tri[35] = 6;

tri[36] = 0;
tri[37] = 2;
tri[38] = 1;

tri[39] = 0;
tri[40] = 3;
tri[41] = 2;

tri[42] = 13;
tri[43] = 3;
tri[44] = 0;

tri[45] = 12;
tri[46] = 3;
tri[47] = 13;

tri[48] = 11;
tri[49] = 3;
tri[50] = 12;

tri[51] = 10;
tri[52] = 3;
tri[53] = 11;
```

```
tri[54] = 10;
tri[55] = 4;
tri[56] = 3;

tri[57] = 9;
tri[58] = 4;
tri[59] = 10;

tri[60] = 8;
tri[61] = 4;
tri[62] = 9;

tri[63] = 7;
tri[64] = 4;
tri[65] = 8;

tri[66] = 7;
tri[67] = 5;
tri[68] = 4;

tri[69] = 7;
tri[70] = 6;
tri[71] = 5;

var mesh: Mesh = new Mesh();
mesh.name = "Arch second floor vert";
mesh.vertices = verts;
```

```
        mesh.triangles = tri;
        mesh.uv = uv;
        mesh.normals = normals;
        //mesh.RecalculateNormals();
        tmpobj.renderer.material.mainTexture = Resources.Load("white_brick_wall");
        mf.mesh = mesh;
}//end function ArchScndFlrVertical(this.RandX, this.RandZ, tmpobj)



public function DoArch(RandX, RandZ){
        tmpobj = new GameObject();
        Destroy(tmpobj);

//:::: start first floor::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
        var nPOSZ:float = this.RandZ + 0.777;
        this.RandX -= 0.2;
        for(i=0; i<=4; i++){
                if(tmpobj){
                        Arch(this.RandX, nPOSZ, tmpobj);
                        this.RandX += 1.35;
                }//end if
        }//end for

        this.RandX -= 9.98;
        this.RandZ +=5.50;
```

```
	for(i=0; i<=1; i++){
		if(tmpobj){
			ArchVertical(this.RandX, this.RandZ, tmpobj);
			this.RandX += 6.9;
		}//end if
	}//end for
//:::: end first floor::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

//:::: start second floor::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
	this.RandZ -= 1.777;
	this.RandX -= 10.65;
	for(i=0; i<=4; i++){
		if(tmpobj){
			ArchScndFlr(this.RandX, this.RandZ, tmpobj);
			if(i==1)
				this.RandX += 1.45;
			else
				this.RandX += 1.35;
		}//end if
	}//end for

	this.RandX -= 9.98;
	this.RandZ += 1.7;

	for(i=0; i<=1; i++){
		if(tmpobj){
			ArchScndFlrVertical(this.RandX, this.RandZ, tmpobj);
```

```
                this.RandX += 6.8;
            }//end if
        }//end for
//:::: end second floor::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

}//end function DoArch(Rand_X, Rand_Z)
//--------------------------------END ARCH-----------------------------------------------------
```

//House_4

```
//create objects
var oBuilding4 : GameObject;
//var oColumn = GameObject;//.CreatePrimitive(PrimitiveType.Cube);
var oWindow : GameObject;
var oCube : GameObject;
var tmpobj: GameObject;

//-------------------start public variables-------------------------------------------------------
public var aCollectionObj_4 = new Array(); // an Array - a collection- for the objects of House_4

public var i: int = 0;
public var j: int = 0;
public var c: int = 0;

//Random ranges between X, Y coordinates
public var RandX:float;
```

```
public var RandZ:float;
//-------------------end public variables-------------------------------------------------------


//MAIN function
public function mainDO() {


        MainBuilding(RandX,RandZ);
        FirstFloor(RandX,RandZ);
        SecondFloor(RandX, RandZ);
        Eaves(RandX, RandZ);
        Windows(RandX, RandZ);
        Columns(RandX, RandZ);
        Door(RandX, RandZ);
        Roof(RandX, RandZ);
        DoArch(RandX, RandZ);
}//end Start ()


public function Destroy_(){


DestroyImmediate(oBuilding4);


}


//-----------Main Building-ground floor--------------------------
public function MainBuilding(RandX,RandZ){
```

```
        oBuilding4 = GameObject.CreatePrimitive(PrimitiveType.Cube);

        oBuilding4.renderer.material.color = Color(Random.Range(1,0.9), Random.Range(0.980,0.880), Random.Range(0.804,0.704), 0);

        oBuilding4.transform.position = Vector3(this.RandX, 1.722107, this.RandZ);

        oBuilding4.name = "House_4";

        oBuilding4.transform.localScale.x = 7;

        oBuilding4.transform.localScale.y = 3.308559;

        oBuilding4.transform.localScale.z = 3.5;

        aCollectionObj_4.Add(oBuilding4);

}//end function MainBuilding(Rand_X,Rand_Z)



//----------------------------------------START FIRST FLOOR----------------------------------------

public function FirstFloor(RandX,RandZ){

        oCube = GameObject.CreatePrimitive(PrimitiveType.Cube);

        //solving an issue appearing: destroy visible instances appear as common cubes on the scene

        Destroy (oCube);

        oCube.renderer.material.color = Color(1, 0.980, 0.804, 0);

        //First floor

        var oFrst_Flr = Instantiate(oCube, Vector3(this.RandX, 4.527526, this.RandZ + 0.9), Quaternion.identity);

        oFrst_Flr.transform.parent = oBuilding4.transform; //parenting

        oFrst_Flr.name = "First Floor";

        oFrst_Flr.transform.localScale.x = 0.9954391;

        oFrst_Flr.transform.localScale.y = 0.7123187;

        oFrst_Flr.transform.localScale.z = 1.536353;

        aCollectionObj_4.Add(oFrst_Flr);
```

```
}//end function FirstFloor(RandX,RandZ)

//----------------------------------------END FIRST FLOOR----------------------------------------



//----------------------------------------START SECOND FLOOR----------------------------------------

public function SecondFloor(RandX, RandZ){

        oCube = GameObject.CreatePrimitive(PrimitiveType.Cube);

        //solving an issue appearing: destroy visible instances appear as common cubes on the scene

        Destroy (oCube);

        oCube.renderer.material.color = Color(1, 0.980, 0.804, 0);

        //Second Floor

        var oScnd_Floor = Instantiate(oCube, Vector3(this.RandX, 7.2189827, this.RandZ + 0.9), Quaternion.identity);

        oScnd_Floor.transform.parent = oBuilding4.transform; //parenting

        oScnd_Floor.name = "Second Floor";

        oScnd_Floor.transform.localScale.x = 0.9954391;

        oScnd_Floor.transform.localScale.y = 0.9377826;

        oScnd_Floor.transform.localScale.z = 1.536353;

        aCollectionObj_4.Add(oScnd_Floor);

}//end function SecondFloor(RandX,RandZ)

//----------------------------------------END SECOND FLOOR----------------------------------------
```

```
//---------------------------------------START EAVES-----------------------------------------------

public function Eaves(RandX, RandZ){

	oCube = GameObject.CreatePrimitive(PrimitiveType.Cube);

	//solving an issue appearing: destroy visible instances appear as common cubes on the scene

	Destroy (oCube);

	oCube.renderer.material.color = Color.white;

	var nPOSY_eaves:float = 3.322107;

	for(i=0; i<=2; i++){

		if(oCube){

			var oEaves_flr = Instantiate(oCube, Vector3(this.RandX, nPOSY_eaves, this.RandZ + 0.9), Quaternion.identity);

			oEaves_flr.transform.parent = oBuilding4.transform; //parenting

			oEaves_flr.name = "eaves " + (++c);

			oEaves_flr.transform.localScale.x = 1.020326;

			oEaves_flr.transform.localScale.y = 0.06;

			oEaves_flr.transform.localScale.z = 1.593966;

			aCollectionObj_4.Add(oEaves_flr);

			if(i == 1)

				nPOSY_eaves += 3.2030433;

			else

				nPOSY_eaves += 2.368318;

		}//end if

	}//end for

	c = 0;

}//end function Eaves(RandX, RandZ)
```

```
//---------------------------------------END EAVES-----------------------------------------------
-----------




//-----------------------------------START WINDOWS-----------------------------------------------
---------------

public function Windows(RandX, RandZ){

	oWindow = GameObject.CreatePrimitive(PrimitiveType.Cube);

	//solving an issue appearing: destroy visible instances appear as common
cubes on the scene

	Destroy(oWindow);



	//transparent windows

	//oWindow.transform.renderer.material.color = Color.clear;

	oWindow.renderer.material.mainTexture                                    =
Resources.Load("window_corfu1"); //import texture for windows



	var nWind_X:float = this.RandX + 2.82102;

	var nWind_Y:float = 5.0751646;

	var nWind_Z:float = this.RandZ + 3.6062495;

//::::::::::::::::::::::::::::::::::::::::::start
facade::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

//first floor windows

	for(i=0; i<=3; i++){

		if(oWindow){

			var oWindow_frstFlr = Instantiate(oWindow,
Vector3(nWind_X, nWind_Y, nWind_Z), Quaternion.identity);

			oWindow_frstFlr.transform.parent = oBuilding4.transform;
//parenting
```

```
                oWindow_frstFlr.name = "first floor window " + (++c);
                oWindow_frstFlr.transform.localScale.x = 0.1135587;
                oWindow_frstFlr.transform.localScale.y = 0.2774929;
                oWindow_frstFlr.transform.localScale.z = 0.0163088;
                aCollectionObj_4.Add(oWindow_frstFlr);
                nWind_X -= 1.9;
            }//end if
        }//end for
        c = 0;

        nWind_X = this.RandX + 2.82102;

//second floor windows
        for(i=0; i<=3; i++){
            if(oWindow){
                var oWindow_scndFlr = Instantiate(oWindow,
Vector3(nWind_X, nWind_Y + 2.6442788, nWind_Z), Quaternion.identity);
                oWindow_scndFlr.transform.parent = oBuilding4.transform;
//parenting
                oWindow_scndFlr.name = "second floor window " + (++c);
                oWindow_scndFlr.transform.localScale.x = 0.1096849;
                oWindow_scndFlr.transform.localScale.y = 0.4051766;
                oWindow_scndFlr.transform.localScale.z = 0.0163088;
                aCollectionObj_4.Add(oWindow_scndFlr);
                nWind_X -= 1.9;
            }//end if
        }//end for
        c = 0;
```

```
//::::::::::::::::::::::::::::::::::::::::::end
facade::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

//::::::::::::::::::::::::::::::::::::::::::start left side
windows::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

        var nWindPOSX_side:float = this.RandX + 3.5014659;

        var nWindPOSZ_side:float = nWind_Z - 1.3;

//first floor left side windows

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

                if(oWindow){

                        var oWindow_frstFlr_lft = Instantiate(oWindow,
Vector3(nWindPOSX_side, nWind_Y, nWindPOSZ_side), Quaternion.identity);

                        oWindow_frstFlr_lft.transform.parent = oBuilding4.transform;
//parenting

                        oWindow_frstFlr_lft.name = "first floor left side window " +
(++c);

                        oWindow_frstFlr_lft.transform.localScale.x = -0.2409395;

                        oWindow_frstFlr_lft.transform.localScale.y = 0.2774929;

                        oWindow_frstFlr_lft.transform.localScale.z = 0.0163088;

                        oWindow_frstFlr_lft.transform.eulerAngles.y = 90;

                        aCollectionObj_4.Add(oWindow_frstFlr_lft);

                        nWindPOSZ_side -= 3.0;

                }//end if

        }//end for

        c = 0;


        nWindPOSX_side = this.RandX + 3.5014659;

        nWind_Z = this.RandZ + 3.5662495;

        nWindPOSZ_side = nWind_Z - 1.3;
```

```
//second floor left side windows
        for(i=0; i<=1; i++){
                if(oWindow){
                        var oWindow_scndFlr_lft = Instantiate(oWindow, Vector3(nWindPOSX_side, nWind_Y + 2.6442788, nWindPOSZ_side), Quaternion.identity);
                        oWindow_scndFlr_lft.transform.parent = oBuilding4.transform; //parenting
                        oWindow_scndFlr_lft.name = "second floor left side window " + (++c);
                        oWindow_scndFlr_lft.transform.localScale.x = -0.2517607;
                        oWindow_scndFlr_lft.transform.localScale.y = 0.4051766;
                        oWindow_scndFlr_lft.transform.localScale.z = 0.0163088;
                        oWindow_scndFlr_lft.transform.eulerAngles.y = 90;
                        aCollectionObj_4.Add(oWindow_scndFlr_lft);
                        nWindPOSZ_side -= 3.0;
                }//end if
        }//end for
        c = 0;
//::::::::::::::::::::::::::::::::::::::::end left side windows::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::


//::::::::::::::::::::::::::::::::::::::::start right side windows::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
        nWindPOSX_side = this.RandX - 3.5014659;
        nWind_Z = this.RandZ + 3.5662495;
        nWindPOSZ_side = nWind_Z - 1.3;
//first floor right side windows
        for(i=0; i<=1; i++){
```

```
            if(oWindow){

                var oWindow_frstFlr_rght = Instantiate(oWindow, Vector3(nWindPOSX_side, nWind_Y, nWindPOSZ_side), Quaternion.identity);

                oWindow_frstFlr_rght.transform.parent = oBuilding4.transform; //parenting

                oWindow_frstFlr_rght.name = "first floor right side window " + (++c);

                oWindow_frstFlr_rght.transform.localScale.x = -0.2409395;

                oWindow_frstFlr_rght.transform.localScale.y = 0.2774929;

                oWindow_frstFlr_rght.transform.localScale.z = 0.0163088;

                oWindow_frstFlr_rght.transform.eulerAngles.y = 90;

                aCollectionObj_4.Add(oWindow_frstFlr_rght);

                nWindPOSZ_side -= 3.0;

            }//end if

        }//end for

        c = 0;


        nWindPOSX_side = this.RandX - 3.5014659;

        nWind_Z = this.RandZ + 3.5662495;

        nWindPOSZ_side = nWind_Z - 1.3;

//second floor right side windows

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

            if(oWindow){

                var oWindow_scndFlr_rght = Instantiate(oWindow, Vector3(nWindPOSX_side, nWind_Y + 2.6442788, nWindPOSZ_side), Quaternion.identity);

                oWindow_scndFlr_rght.transform.parent = oBuilding4.transform; //parenting

                oWindow_scndFlr_rght.name = "second floor right side window " + (++c);
```

```
            oWindow_scndFlr_rght.transform.localScale.x = -0.2517607;
            oWindow_scndFlr_rght.transform.localScale.y = 0.4051766;
            oWindow_scndFlr_rght.transform.localScale.z = 0.0163088;
            oWindow_scndFlr_rght.transform.eulerAngles.y = 90;
            aCollectionObj_4.Add(oWindow_scndFlr_rght);
            nWindPOSZ_side -= 3.0;
        }//end if
    }//end for
    c = 0;
//::::::::::::::::::::::::::::::::::::::::end right side windows:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

//::::::::::::::::::::::::::::::::::::::::::::::::start ground floor windows::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
nWindPOSX_side = this.RandX + 2.40;
for(i=0; i<=1; i++){
    if(oWindow){
        var oWind_grnd = Instantiate(oWindow, Vector3(nWindPOSX_side, 1.423612, this.RandZ + 1.7776329), Quaternion.identity);
        oWind_grnd.transform.parent = oBuilding4.transform; //parenting
        oWind_grnd.name = "ground floor window " + (++c);
        oWind_grnd.transform.localScale.x = 0.1135587;
        oWind_grnd.transform.localScale.y = 0.4051766;
        oWind_grnd.transform.localScale.z = 0.0163088;
        aCollectionObj_4.Add(oWind_grnd);
        nWindPOSX_side -= 3.10;
    }//end if
}//end for
```

```
c = 0;

//::::::::::::::::::::::::::::::::::::::::::::::::end ground floor windows:::::::::::::::::::::::::::::::::::::::::::::::::::::::::

}//end function Windows(RandX, RandZ)

//--------------------------------------------END WINDOWS----------------------------------------------------



//-----------------------------------------START COLUMNS-----------------------------------------------------

public function Columns(RandX, RandZ){

	oColumn = GameObject.CreatePrimitive(PrimitiveType.Cube);

	oColumn.renderer.material.color = Color.white;


	oColumn.transform.position = Vector3(this.RandX -3.2, 1.302107, this.RandZ + 3.3772203);

	oColumn.transform.parent = oBuilding4.transform; //parenting

	oColumn.name = "Column 0";

	oColumn.transform.localScale.x = 0.07017206;

	oColumn.transform.localScale.y = 0.6454034;

	oColumn.transform.localScale.z = 0.1285665;

	aCollectionObj_4.Add(oColumn);


	oColumnCube = GameObject.CreatePrimitive(PrimitiveType.Cube);


	oColumnCube.transform.position = Vector3(this.RandX -3.2, 0.2373266, this.RandZ + 3.3772203);

	oColumnCube.transform.parent = oColumn.transform; //parenting

	oColumnCube.name = "Column base";
```

```
oColumnCube.transform.localScale.x = 1.43565;
oColumnCube.transform.localScale.y = 0.1184824;
oColumnCube.transform.localScale.z = 1.61189;
aCollectionObj_4.Add(oColumnCube);

var oColumnCube_Up = Instantiate(oColumnCube, Vector3(this.RandX - 3.2, 2.3029889, this.RandZ + 3.3772203), Quaternion.identity);
oColumnCube_Up.transform.parent = oColumn.transform; //parenting
oColumnCube_Up.name = "Column upper part";
oColumnCube_Up.transform.localScale.x = 1.43565;
oColumnCube_Up.transform.localScale.y = 0.1501914;
oColumnCube_Up.transform.localScale.z = 1.61189;
aCollectionObj_4.Add(oColumnCube_Up);

var nextPOSX_column:float = this.RandX - 1.62;
for(i=0; i<=3; i++){
    if(oColumn){
        var cloneColumn_ = Instantiate(oColumn, Vector3(nextPOSX_column, 1.302107, this.RandZ + 3.3772203), Quaternion.identity);
        cloneColumn_.transform.parent = oBuilding4.transform; //parenting
        cloneColumn_.name = "Column " + (++c);
        cloneColumn_.transform.localScale.x = 0.07017206;
        cloneColumn_.transform.localScale.y = 0.6454034;
        cloneColumn_.transform.localScale.z = 0.1285665;
        aCollectionObj_4.Add(cloneColumn_);
        nextPOSX_column += 1.58;
```

```
            }//end if
        }//end for
        c = 0;
}//end function Columns(RandX, RandZ)
//----------------------------------------END COLUMNS-----------------------------------------
------------



//------------------------------START DOOR-------------------------------------------------
--------
public function Door(RandX, RandZ){


        oWindow = GameObject.CreatePrimitive(PrimitiveType.Cube);
        Destroy(oWindow);
        //oWindow.transform.renderer.material.color = Color.clear;
        oWindow.renderer.material.mainTexture = Resources.Load("door_corfu2");
//import texture for door


        var nPosX_door:float = this.RandX + 0.80;
        for(i=0; i<=1; i++){
            if(oWindow){
                var oDoor = Instantiate(oWindow, Vector3(nPosX_door,
1.123612, this.RandZ + 1.7776329), Quaternion.identity);
                oDoor.name = "door " + (++c);
                oDoor.transform.parent = oBuilding4.transform;
                oDoor.transform.localScale.x = 0.1341711;
                oDoor.transform.localScale.y = 0.5980287;
                oDoor.transform.localScale.z = 0.019;
                aCollectionObj_4.Add(oDoor);
```

```
                nPosX_door -= 3.20;
            }//end if
        }//end for
        c = 0;
}//end function Door(RandX, RandZ)
//------------------------------END DOOR-----------------------------------------------------------
----


//-------------------------------------START  ROOF-------------------------------------------------
-----
public function Roof(RandX, RandZ){
        var tmpobj: GameObject = new GameObject();
        var mf: MeshFilter = tmpobj.AddComponent(MeshFilter);

//      tmpobj.AddComponent(MeshFilter);
        tmpobj.AddComponent(MeshRenderer);
        tmpobj.name = "Roof";
        tmpobj.transform.parent = oBuilding4.transform;

        var verts: Vector3[] = new Vector3[6];
        var normals: Vector3[] = new Vector3[6];
        var uv: Vector2[] = new Vector2[6];
        var tri: int[] = new int[18]; //3 vertices * 6 triangles =18


        //vertices positioning
```

```
verts[0] = new Vector3(this.RandX - 3.6, 10, this.RandZ + 1); //P1
verts[1] = new Vector3(this.RandX - 3.6, 9, this.RandZ - 2); //P2
verts[2] = new Vector3(this.RandX - 3.6, 9, this.RandZ + 4); //P3

verts[3] = new Vector3(this.RandX + 3.6, 10, this.RandZ + 1); //P4
verts[4] = new Vector3(this.RandX + 3.6, 9, this.RandZ + 4); //P5
verts[5] = new Vector3(this.RandX + 3.6, 9, this.RandZ - 2); //P6



uv[0] = new Vector2(0, 0.5);
uv[1] = new Vector2(1, 1);
uv[2] = new Vector2(1, 0);

uv[3] = new Vector2(1, 0.5);
uv[4] = new Vector2(0, 0);
uv[5] = new Vector2(0, 1);



//triangles
tri[0] = 0; //refer to vertices: verts[0]
tri[1] = 1; //refer to vertices: verts[1]
tri[2] = 2; //refer to vertices: verts[2]

tri[3] = 0;
tri[4] = 2;
tri[5] = 4;
```

```
        tri[6] = 3;
        tri[7] = 0;
        tri[8] = 4;

        tri[9] = 3;
        tri[10] = 4;
        tri[11] = 5;

        tri[12] = 3;
        tri[13] = 5;
        tri[14] = 0;

        tri[15] = 0;
        tri[16] = 5;
        tri[17] = 1;

        var mesh: Mesh = new Mesh();
        mesh.name = "Roof";
        mesh.vertices = verts;
        mesh.triangles = tri;
        mesh.uv = uv;
        mesh.normals = normals;
        mesh.RecalculateNormals();
        mf.mesh = mesh;
        tmpobj.renderer.material.mainTexture =
Resources.Load("tiled_roof_red_grey");
}//end function Roof(RandX, RandZ)
```

```
//-------------------------------------END ROOF-------------------------------------------------
----



//-----------------------------------START ARCH-----------------------------------------------
-------------

private function Arch(RandX, RandZ, tmpobj) {

        tmpobj = new GameObject();

        var mf: MeshFilter = tmpobj.AddComponent(MeshFilter);


        tmpobj.AddComponent(MeshRenderer);

        tmpobj.name = "Arch";

        tmpobj.renderer.material.color = Color(1, 0.980, 0.804, 0);

        tmpobj.transform.parent = oBuilding4.transform;


        var verts: Vector3[] = new Vector3[14];

        var uv: Vector2[] = new Vector2[14];

        var normals: Vector3[] = new Vector3[14];

        var tri: int[] = new int[72]; //3 vertices * 12 triangles = 36 * 2 (duplicate
triangles for the back face) =72


        //vertices positioning

        verts[0] = new Vector3(this.RandX - 1.9, 2.5, this.RandZ + 3.777); //P1

        verts[1] = new Vector3(this.RandX - 1.9, 2, this.RandZ + 3.777); //P2

        verts[2] = new Vector3(this.RandX - 1.35, 2, this.RandZ + 3.777); //P3

        verts[3] = new Vector3(this.RandX - 1.35, 3.5, this.RandZ + 3.777); //P4

        verts[4] = new Vector3(this.RandX - 3.5, 3.5, this.RandZ + 3.777); //P5

        verts[5] = new Vector3(this.RandX - 3.5, 2, this.RandZ + 3.777); //P6
```

```
verts[6] = new Vector3(this.RandX - 2.95, 2, this.RandZ + 3.777); //P7
verts[7] = new Vector3(this.RandX - 2.95, 2.5, this.RandZ + 3.777); //P8
verts[8] = new Vector3(this.RandX - 2.9, 2.8, this.RandZ + 3.777); //P9
verts[9] = new Vector3(this.RandX - 2.75, 3, this.RandZ + 3.777); //P10
verts[10] = new Vector3(this.RandX - 2.55, 3.1, this.RandZ + 3.777); //P11
verts[11] = new Vector3(this.RandX - 2.3,  3.1, this.RandZ + 3.777); //P12
verts[12] = new Vector3(this.RandX - 2.1, 3, this.RandZ + 3.777); //P13
verts[13] = new Vector3(this.RandX - 1.95, 2.8, this.RandZ + 3.777); //P14

//normals
normals[0] = new Vector3(0, 0, 1);
normals[1] = new Vector3(0, 0, 1);
normals[2] = new Vector3(0, 0, 1);
normals[3] = new Vector3(0, 0, 1);
normals[4] = new Vector3(0, 0, 1);
normals[5] = new Vector3(0, 0, 1);
normals[6] = new Vector3(0, 0, 1);
normals[7] = new Vector3(0, 0, 1);
normals[8] = new Vector3(0, 0, 1);
normals[9] = new Vector3(0, 0, 1);
normals[10] = new Vector3(0, 0, 1);
normals[11] = new Vector3(0, 0, 1);
normals[12] = new Vector3(0, 0, 1);
normals[13] = new Vector3(0, 0, 1);

//uv mapping (textures)
uv[0] = new Vector2(0, 0.3);
```

```
uv[1] = new Vector2(0.2, 0);
uv[2] = new Vector2(0, 0);
uv[3] = new Vector2(0, 1);
uv[4] = new Vector2(1, 1);
uv[5] = new Vector2(1, 0);
uv[6] = new Vector2(0.8, 0);
uv[7] = new Vector2(0.8, 0.3);
uv[8] = new Vector2(0.75, 0.5);
uv[9] = new Vector2(0.65, 0.7);
uv[10] = new Vector2(0.6, 0.9);
uv[11] = new Vector2(0.4, 0.9);
uv[12] = new Vector2(0.45, 0.7);
uv[13] = new Vector2(0.35, 0.5);



//triangles
tri[0] = 0; //refer to vertices: verts[0]
tri[1] = 1; //refer to vertices: verts[1]
tri[2] = 2; //refer to vertices: verts[2]

tri[3] = 0;
tri[4] = 2;
tri[5] = 3;

tri[6] = 0;
tri[7] = 3;
tri[8] = 13;
```

```
tri[9] = 13;
tri[10] = 3;
tri[11] = 12;

tri[12] = 12;
tri[13] = 3;
tri[14] = 11;

tri[15] = 11;
tri[16] = 3;
tri[17] = 4;

tri[18] = 11;
tri[19] = 4;
tri[20] = 10;

tri[21] = 10;
tri[22] = 4;
tri[23] = 9;

tri[24] = 9;
tri[25] = 4;
tri[26] = 8;

tri[27] = 8;
tri[28] = 4;
```

```
tri[29] = 7;


tri[30] = 7;
tri[31] = 4;
tri[32] = 5;


tri[33] = 7;
tri[34] = 5;
tri[35] = 6;


tri[36] = 0;
tri[37] = 2;
tri[38] = 1;


tri[39] = 0;
tri[40] = 3;
tri[41] = 2;


tri[42] = 13;
tri[43] = 3;
tri[44] = 0;


tri[45] = 12;
tri[46] = 3;
tri[47] = 13;


tri[48] = 11;
```

```
tri[49] = 3;
tri[50] = 12;

tri[51] = 10;
tri[52] = 3;
tri[53] = 11;

tri[54] = 10;
tri[55] = 4;
tri[56] = 3;

tri[57] = 9;
tri[58] = 4;
tri[59] = 10;

tri[60] = 8;
tri[61] = 4;
tri[62] = 9;

tri[63] = 7;
tri[64] = 4;
tri[65] = 8;

tri[66] = 7;
tri[67] = 5;
tri[68] = 4;
```

```
        tri[69] = 7;
        tri[70] = 6;
        tri[71] = 5;

        var mesh: Mesh = new Mesh();
        mesh.name = "Arch";
        mesh.vertices = verts;
        mesh.triangles = tri;
        mesh.uv = uv;
        mesh.normals = normals;
        //mesh.RecalculateNormals();
        tmpobj.renderer.material.mainTexture                                    =
Resources.Load("white_brick_wall");
        mf.mesh = mesh;
}//end function Arch(Rand_X, Rand_Z)


private function ArchVertical(RandX, RandZ, tmpobj){

        tmpobj = new GameObject();
        var mf: MeshFilter = tmpobj.AddComponent(MeshFilter);

        tmpobj.AddComponent(MeshRenderer);
        tmpobj.name = "Arch vert";
        tmpobj.renderer.material.color = Color(1, 0.980, 0.804, 0);
        tmpobj.transform.parent = oBuilding4.transform;
```

```
var verts: Vector3[] = new Vector3[14];

var uv: Vector2[] = new Vector2[14];

var normals: Vector3[] = new Vector3[14];

var tri: int[] = new int[72]; //3 vertices * 12 triangles = 36 * 2 (duplicate triangles for the back face) =72


//vertices positioning

verts[0] = new Vector3(this.RandX , 2.5, this.RandZ - 1.9); //P1

verts[1] = new Vector3(this.RandX , 2, this.RandZ - 1.9); //P2

verts[2] = new Vector3(this.RandX , 2, this.RandZ - 1.35); //P3

verts[3] = new Vector3(this.RandX , 3.5, this.RandZ - 1.35); //P4

verts[4] = new Vector3(this.RandX , 3.5, this.RandZ - 3.5); //P5

verts[5] = new Vector3(this.RandX , 2, this.RandZ - 3.5); //P6

verts[6] = new Vector3(this.RandX , 2, this.RandZ - 2.95); //P7

verts[7] = new Vector3(this.RandX , 2.5, this.RandZ - 2.95); //P8

verts[8] = new Vector3(this.RandX , 2.8, this.RandZ - 2.9); //P9

verts[9] = new Vector3(this.RandX , 3, this.RandZ - 2.75); //P10

verts[10] = new Vector3(this.RandX , 3.1, this.RandZ - 2.55); //P11

verts[11] = new Vector3(this.RandX ,  3.1, this.RandZ - 2.3); //P12

verts[12] = new Vector3(this.RandX , 3, this.RandZ - 2.1); //P13

verts[13] = new Vector3(this.RandX , 2.8, this.RandZ - 1.95); //P14


//normals

normals[0] = new Vector3(0, 0, 1);

normals[1] = new Vector3(0, 0, 1);

normals[2] = new Vector3(0, 0, 1);

normals[3] = new Vector3(0, 0, 1);
```

```
normals[4] = new Vector3(0, 0, 1);
normals[5] = new Vector3(0, 0, 1);
normals[6] = new Vector3(0, 0, 1);
normals[7] = new Vector3(0, 0, 1);
normals[8] = new Vector3(0, 0, 1);
normals[9] = new Vector3(0, 0, 1);
normals[10] = new Vector3(0, 0, 1);
normals[11] = new Vector3(0, 0, 1);
normals[12] = new Vector3(0, 0, 1);
normals[13] = new Vector3(0, 0, 1);

//uv mapping (textures)
uv[0] = new Vector2(0, 0.3);
uv[1] = new Vector2(0.2, 0);
uv[2] = new Vector2(0, 0);
uv[3] = new Vector2(0, 1);
uv[4] = new Vector2(1, 1);
uv[5] = new Vector2(1, 0);
uv[6] = new Vector2(0.8, 0);
uv[7] = new Vector2(0.8, 0.3);
uv[8] = new Vector2(0.75, 0.5);
uv[9] = new Vector2(0.65, 0.7);
uv[10] = new Vector2(0.6, 0.9);
uv[11] = new Vector2(0.4, 0.9);
uv[12] = new Vector2(0.45, 0.7);
uv[13] = new Vector2(0.35, 0.5);
```

```
//triangles
tri[0] = 0; //refer to vertices: verts[0]
tri[1] = 1; //refer to vertices: verts[1]
tri[2] = 2; //refer to vertices: verts[2]

tri[3] = 0;
tri[4] = 2;
tri[5] = 3;

tri[6] = 0;
tri[7] = 3;
tri[8] = 13;

tri[9] = 13;
tri[10] = 3;
tri[11] = 12;

tri[12] = 12;
tri[13] = 3;
tri[14] = 11;

tri[15] = 11;
tri[16] = 3;
tri[17] = 4;

tri[18] = 11;
```

```
tri[19] = 4;
tri[20] = 10;

tri[21] = 10;
tri[22] = 4;
tri[23] = 9;

tri[24] = 9;
tri[25] = 4;
tri[26] = 8;

tri[27] = 8;
tri[28] = 4;
tri[29] = 7;

tri[30] = 7;
tri[31] = 4;
tri[32] = 5;

tri[33] = 7;
tri[34] = 5;
tri[35] = 6;

tri[36] = 0;
tri[37] = 2;
tri[38] = 1;
```

```
tri[39] = 0;
tri[40] = 3;
tri[41] = 2;

tri[42] = 13;
tri[43] = 3;
tri[44] = 0;

tri[45] = 12;
tri[46] = 3;
tri[47] = 13;

tri[48] = 11;
tri[49] = 3;
tri[50] = 12;

tri[51] = 10;
tri[52] = 3;
tri[53] = 11;

tri[54] = 10;
tri[55] = 4;
tri[56] = 3;

tri[57] = 9;
tri[58] = 4;
tri[59] = 10;
```

```
        tri[60] = 8;
        tri[61] = 4;
        tri[62] = 9;

        tri[63] = 7;
        tri[64] = 4;
        tri[65] = 8;

        tri[66] = 7;
        tri[67] = 5;
        tri[68] = 4;

        tri[69] = 7;
        tri[70] = 6;
        tri[71] = 5;

        var mesh: Mesh = new Mesh();
        mesh.name = "Arch vert";
        mesh.vertices = verts;
        mesh.triangles = tri;
        mesh.uv = uv;
        mesh.normals = normals;
        //mesh.RecalculateNormals();
        tmpobj.renderer.material.mainTexture                    =
Resources.Load("white_brick_wall");
        mf.mesh = mesh;
```

```
}//end function ArchVertical(Rand_X, Rand_Z, tmpobj)

public function DoArch(RandX, RandZ){
	tmpobj = new GameObject();
	Destroy(tmpobj);

	var nPOSZ:float = this.RandZ + 0.777;

	for(i=0; i<=3; i++){
		if(tmpobj){
			Arch(this.RandX, nPOSZ, tmpobj);
			this.RandX += 1.6;
		}//end if
	}//end for

	this.RandX -= 9.96;
	this.RandZ +=5.15;

	for(i=0; i<=1; i++){
		if(tmpobj){
			ArchVertical(this.RandX, this.RandZ, tmpobj);
			this.RandX += 7.1;
		}//end if
	}//end for
}//end function DoArch(Rand_X, Rand_Z)
//--------------------------------END ARCH-----------------------------------------------------
```

//House_5

```
//create objects
var oBuilding5 : GameObject;
var oWindow : GameObject;
var oCube : GameObject;
var oColumn: GameObject;
var tmpobj: GameObject;
//-----------------------start public variables---------------------------------
public var i: int = 0;
public var j: int = 0;
public var c: int = 0;

//Random ranges between X, Y coordinates
public var Rand_X:float;
public var Rand_Z:float;

public var aCollectionObj_5 = new Array(); // an Array - a collection- for the objects of House_5
//-----------------------end public variables---------------------------------


//MAIN function
public function mainDO() {

    MainBuilding(Rand_X,Rand_Z);
    F_flr(Rand_X,Rand_Z);
```

```
        Eaves(Rand_X,Rand_Z);
        Windows(Rand_X,Rand_Z);
        Columns(Rand_X, Rand_Z);
        Door(Rand_X, Rand_Z);
        Roof(Rand_X, Rand_Z);
        DoArch(Rand_X, Rand_Z);
}//end Start ()


public function Destroy_(){

DestroyImmediate(oBuilding5);

}


//-----------Main Building-ground floor--------------------------
public function MainBuilding(Rand_X,Rand_Z){
        oBuilding5 = GameObject.CreatePrimitive(PrimitiveType.Cube);
        oBuilding5.transform.position                                   =
Vector3(this.Rand_X,1.722107,this.Rand_Z);
        oBuilding5.name = "House_5";
        oBuilding5.renderer.material.color = Color.white;
        oBuilding5.transform.localScale.x = 7;
        oBuilding5.transform.localScale.y = 3.308559;
        oBuilding5.transform.localScale.z = 3.5;
        aCollectionObj_5.Add(oBuilding5);
```

```
}//end function MainBuilding(Rand_X,Rand_Z)



public function F_flr(Rand_X,Rand_Z){

	oCube = GameObject.CreatePrimitive(PrimitiveType.Cube);

	//solving an issue appearing: destroy visible instances appear as common cubes on the scene

	Destroy(oCube);

	oCube.renderer.material.color = Color(Random.Range(0.88627,0.78627), Random.Range(0.43529, 0.33529), Random.Range(0.32156,0.22156), 0);

	//First floor

	var oFrst_Flr_ = Instantiate(oCube, Vector3(this.Rand_X, 4.527526, this.Rand_Z + 0.9), Quaternion.identity);

	oFrst_Flr_.transform.parent = oBuilding5.transform; //parenting

	oFrst_Flr_.name = "First Floor";

	oFrst_Flr_.transform.localScale.x = 0.9954391;

	oFrst_Flr_.transform.localScale.y = 0.7123187;

	oFrst_Flr_.transform.localScale.z = 1.536353;

	aCollectionObj_5.Add(oFrst_Flr_);

}// end function F_flr(Rand_X,Rand_Z)


//----------------------------------------START EAVES----------------------------------------
----------

public function Eaves(Rand_X,Rand_Z){

	oCube = GameObject.CreatePrimitive(PrimitiveType.Cube);

	//solving an issue appearing: destroy visible instances appear as common cubes on the scene

	Destroy(oCube);

	oCube.renderer.material.color = Color.white;
```

```
	var nPOSY_eaves:float = 3.322107;
	for(i=0; i<=1; i++){
		if(oCube){
			var oEaves_flr_ = Instantiate(oCube, Vector3(this.Rand_X, nPOSY_eaves, this.Rand_Z + 0.9), Quaternion.identity);
			oEaves_flr_.transform.parent = oBuilding5.transform; //parenting
			oEaves_flr_.name = "eaves " + (++c);
			oEaves_flr_.transform.localScale.x = 1.020326;
			oEaves_flr_.transform.localScale.y = 0.06;
			oEaves_flr_.transform.localScale.z = 1.593966;
			aCollectionObj_5.Add(oEaves_flr_);
			nPOSY_eaves += 2.368318;
		}//end if
	}//end for
	c = 0;
}//end function Eaves(Rand_X,Rand_Z)
//----------------------------------------END EAVES----------------------------------------------------------


//-----------------------------------START WINDOWS-----------------------------------------------------------


public function Windows(Rand_X,Rand_Z){
	oWindow = GameObject.CreatePrimitive(PrimitiveType.Cube);
	//solving an issue appearing: destroy visible instances appear as common cubes on the scene
	Destroy(oWindow);
```

```
        //transparent windows
        //oWindow.transform.renderer.material.color = Color.clear;
        oWindow.renderer.material.mainTexture = Resources.Load("window_corfu1"); //import texture for windows

        var nWind_X:float = this.Rand_X + 2.82102;
        var nWind_Y:float = 5.0751646;
        var nWind_Z:float = this.Rand_Z + 3.6062495;
//:::::::::::::::::::::::::::::::::::::::::start facade::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
        //first floor windows
        for(i=0; i<=3; i++){
            if(oWindow){
                var oWindow_frstFlr_ = Instantiate(oWindow, Vector3(nWind_X,nWind_Y,nWind_Z), Quaternion.identity);
                oWindow_frstFlr_.transform.parent = oBuilding5.transform; //parenting
                oWindow_frstFlr_.name = "first floor window " + (++c);
                oWindow_frstFlr_.transform.localScale.x = 0.1195358;
                oWindow_frstFlr_.transform.localScale.y = 0.2774929;
                oWindow_frstFlr_.transform.localScale.z = 0.0163088;
                aCollectionObj_5.Add(oWindow_frstFlr_);
                nWind_X -= 1.9;
            }//end if
        }//end for
        c = 0;
//:::::::::::::::::::::::::::::::::::::::::end facade::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
```

```
//::::::::::::::::::::::::::::::::::::::start left side windows::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

    var nWindPOSX_side:float = this.Rand_X + 3.5014659;

    var nWindPOSZ_side:float = nWind_Z - 1.3;

    //first floor left side windows

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

        if(oWindow){

            var oWindow_frstFlr_lft_ = Instantiate(oWindow, Vector3(nWindPOSX_side, nWind_Y, nWindPOSZ_side), Quaternion.identity);

            oWindow_frstFlr_lft_.transform.parent = oBuilding5.transform; //parenting

            oWindow_frstFlr_lft_.name = "first floor left side window " + (++c);

            oWindow_frstFlr_lft_.transform.localScale.x = -0.2409395;

            oWindow_frstFlr_lft_.transform.localScale.y = 0.2774929;

            oWindow_frstFlr_lft_.transform.localScale.z = 0.0163088;

            oWindow_frstFlr_lft_.transform.eulerAngles.y = 90;

            aCollectionObj_5.Add(oWindow_frstFlr_lft_);

            nWindPOSZ_side -= 3.0;

        }//end if

    }//end for

    c = 0;

//::::::::::::::::::::::::::::::::::::::end left side windows::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::


//::::::::::::::::::::::::::::::::::::::start right side windows::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

    nWindPOSX_side = this.Rand_X - 3.5014659;
```

```
nWind_Z = this.Rand_Z + 3.5662495;
nWindPOSZ_side = nWind_Z - 1.3;
//first floor right side windows
for(i=0; i<=1; i++){
    if(oWindow){
        var oWindow_frstFlr_rght_ = Instantiate(oWindow, Vector3(nWindPOSX_side, nWind_Y,nWindPOSZ_side), Quaternion.identity);
        oWindow_frstFlr_rght_.transform.parent = oBuilding5.transform; //parenting
        oWindow_frstFlr_rght_.name = "first floor right side window " + (++c);
        oWindow_frstFlr_rght_.transform.localScale.x = -0.2409395;
        oWindow_frstFlr_rght_.transform.localScale.y = 0.2774929;
        oWindow_frstFlr_rght_.transform.localScale.z = 0.0163088;
        oWindow_frstFlr_rght_.transform.eulerAngles.y = 90;
        aCollectionObj_5.Add(oWindow_frstFlr_rght_);
        nWindPOSZ_side -= 3.0;
    }//end if
}//end for
c = 0;
//:::::::::::::::::::::::::::::::::::::::::end right side windows:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

//:::::::::::::::::::::::::::::::::::::::::::::::::::start ground floor windows::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

nWindPOSX_side = this.Rand_X + 2.40;
for(i=0; i<=1; i++){
    if(oWindow){
```

```
            var oWind_grnd = Instantiate(oWindow, Vector3(nWindPOSX_side, 1.423612, this.Rand_Z + 1.7776329), Quaternion.identity);

            oWind_grnd.transform.parent = oBuilding5.transform; //parenting

            oWind_grnd.name = "ground floor window " + (++c);

            oWind_grnd.transform.localScale.x = 0.1135587;

            oWind_grnd.transform.localScale.y = 0.4051766;

            oWind_grnd.transform.localScale.z = 0.0163088;

            aCollectionObj_5.Add(oWind_grnd);

            nWindPOSX_side -= 3.10;

        }//end if

}//end for

c = 0;

//::::::::::::::::::::::::::::::::::::::::::::::::::end ground floor windows::::::::::::::::::::::::::::::::::::::::::::::::::::::

}//end function Windows(Rand_X,Rand_Z)

//-----------------------------------END  WINDOWS-----------------------------------------------------------------------------



//-------------------------------------START  COLUMNS-------------------------------------------------------------------

public function Columns(Rand_X, Rand_Z){

        oColumn = GameObject.CreatePrimitive(PrimitiveType.Cube);

        oColumn.renderer.material.color = Color.white;


        oColumn.transform.position = Vector3(this.Rand_X -3.2, 1.302107, this.Rand_Z + 3.3772203);

        oColumn.transform.parent = oBuilding5.transform; //parenting

        oColumn.name = "Column 0";
```

```
oColumn.transform.localScale.x = 0.07017206;

oColumn.transform.localScale.y = 0.6454034;

oColumn.transform.localScale.z = 0.1285665;

aCollectionObj_5.Add(oColumn);


oColumnCube = GameObject.CreatePrimitive(PrimitiveType.Cube);


oColumnCube.transform.position = Vector3(this.Rand_X -3.2, 0.2373266, this.Rand_Z + 3.372203);

oColumnCube.transform.parent = oColumn.transform; //parenting

oColumnCube.name = "Column base";

oColumnCube.transform.localScale.x = 1.43565;

oColumnCube.transform.localScale.y = 0.1184824;

oColumnCube.transform.localScale.z = 1.61189;

aCollectionObj_5.Add(oColumnCube);



var oColumnCube_Up = Instantiate(oColumnCube, Vector3(this.Rand_X - 3.2, 2.3029889, this.Rand_Z + 3.3772203), Quaternion.identity);

oColumnCube_Up.transform.parent = oColumn.transform; //parenting

oColumnCube_Up.name = "Column upper part";

oColumnCube_Up.transform.localScale.x = 1.43565;

oColumnCube_Up.transform.localScale.y = 0.1501914;

oColumnCube_Up.transform.localScale.z = 1.61189;

aCollectionObj_5.Add(oColumnCube_Up);


var nextPOSX_column:float = this.Rand_X - 1.62;

for(i=0; i<=3; i++){
```

```
            if(oColumn){
                var cloneColumn_ = Instantiate(oColumn, Vector3(nextPOSX_column, 1.302107, this.Rand_Z + 3.3772203), Quaternion.identity);
                cloneColumn_.transform.parent = oBuilding5.transform; //parenting
                cloneColumn_.name = "Column " + (++c);
                cloneColumn_.transform.localScale.x = 0.07017206;
                cloneColumn_.transform.localScale.y = 0.6454034;
                cloneColumn_.transform.localScale.z = 0.1285665;
                aCollectionObj_5.Add(cloneColumn_);
                nextPOSX_column += 1.58;
            }//end if
        }//end for
        c = 0;
}//end function Columns(RandX, RandZ)
//-----------------------------------END COLUMNS-----------------------------------------------------------



//-------------------------------START  DOOR-----------------------------------------------------------------
public function Door(Rand_X, Rand_Z){

        oWindow = GameObject.CreatePrimitive(PrimitiveType.Cube);
        Destroy(oWindow);
        //oWindow.transform.renderer.material.color = Color.clear;
```

```
        oWindow.renderer.material.mainTexture = Resources.Load("door_corfu2");
//import texture for door


        var nPosX_door:float = this.Rand_X + 0.80;

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

                if(oWindow){

                        var oDoor = Instantiate(oWindow, Vector3(nPosX_door,
1.123612, this.Rand_Z + 1.7776329), Quaternion.identity);

                        oDoor.name = "door " + (++c);

                        oDoor.transform.parent = oBuilding5.transform;

                        oDoor.transform.localScale.x = 0.1341711;

                        oDoor.transform.localScale.y = 0.5980287;

                        oDoor.transform.localScale.z = 0.019;

                        aCollectionObj_5.Add(oDoor);

                        nPosX_door -= 3.20;

                }//end if

        }//end for

        c = 0;

}//end function Door(RandX, RandZ)

//------------------------------END DOOR-----------------------------------------------------------
----




//-------------------------------------START  ROOF-------------------------------------------------
-----

public function Roof(Rand_X, Rand_Z){

        var tmpobj: GameObject = new GameObject();

        var mf: MeshFilter = tmpobj.AddComponent(MeshFilter);
```

```
//      tmpobj.AddComponent(MeshFilter);
        tmpobj.AddComponent(MeshRenderer);
        tmpobj.name = "Roof";
        tmpobj.transform.parent = oBuilding5.transform;

        var verts: Vector3[] = new Vector3[6];
        var normals: Vector3[] = new Vector3[6];
        var uv: Vector2[] = new Vector2[6];
        var tri: int[] = new int[18]; //3 vertices * 6 triangles =18


        //vertices positioning
        verts[0] = new Vector3(this.Rand_X - 3.6, 7, this.Rand_Z + 1); //P1
        verts[1] = new Vector3(this.Rand_X - 3.6, 5.8, this.Rand_Z - 2); //P2
        verts[2] = new Vector3(this.Rand_X - 3.6, 5.8, this.Rand_Z + 4); //P3

        verts[3] = new Vector3(this.Rand_X + 3.6, 7, this.Rand_Z + 1); //P4
        verts[4] = new Vector3(this.Rand_X + 3.6, 5.8, this.Rand_Z + 4); //P5
        verts[5] = new Vector3(this.Rand_X + 3.6, 5.8, this.Rand_Z - 2); //P6


        uv[0] = new Vector2(0, 0.5);
        uv[1] = new Vector2(1, 1);
        uv[2] = new Vector2(1, 0);

        uv[3] = new Vector2(1, 0.5);
```

```
uv[4] = new Vector2(0, 0);
uv[5] = new Vector2(0, 1);



//triangles
tri[0] = 0; //refer to vertices: verts[0]
tri[1] = 1; //refer to vertices: verts[1]
tri[2] = 2; //refer to vertices: verts[2]

tri[3] = 0;
tri[4] = 2;
tri[5] = 4;

tri[6] = 3;
tri[7] = 0;
tri[8] = 4;

tri[9] = 3;
tri[10] = 4;
tri[11] = 5;

tri[12] = 3;
tri[13] = 5;
tri[14] = 0;

tri[15] = 0;
tri[16] = 5;
```

```
        tri[17] = 1;


        var mesh: Mesh = new Mesh();
        mesh.name = "Roof";
        mesh.vertices = verts;
        mesh.triangles = tri;
        mesh.uv = uv;
        mesh.normals = normals;
        mesh.RecalculateNormals();
        mf.mesh = mesh;
        tmpobj.renderer.material.mainTexture =
Resources.Load("tiled_roof_red_grey");
}//end function Roof(RandX, RandZ)
//-------------------------------------END ROOF-------------------------------------------------
----




//-----------------------------------START ARCH-----------------------------------------------
-------------
private function Arch(Rand_X, Rand_Z, tmpobj) {
        tmpobj = new GameObject();
        var mf: MeshFilter = tmpobj.AddComponent(MeshFilter);


        tmpobj.AddComponent(MeshRenderer);
        tmpobj.name = "Arch";
        tmpobj.renderer.material.color = Color(0.827450, 0.827450, 0.827450, 0);
```

```
tmpobj.transform.parent = oBuilding5.transform;


var verts: Vector3[] = new Vector3[14];

var uv: Vector2[] = new Vector2[14];

var normals: Vector3[] = new Vector3[14];

var tri: int[] = new int[72]; //3 vertices * 12 triangles = 36 * 2 (duplicate triangles for the back face) =72


//vertices positioning

verts[0] = new Vector3(this.Rand_X - 1.9, 2.5, this.Rand_Z + 3.777); //P1

verts[1] = new Vector3(this.Rand_X - 1.9, 2, this.Rand_Z + 3.777); //P2

verts[2] = new Vector3(this.Rand_X - 1.35, 2, this.Rand_Z + 3.777); //P3

verts[3] = new Vector3(this.Rand_X - 1.35, 3.5, this.Rand_Z + 3.777); //P4

verts[4] = new Vector3(this.Rand_X - 3.5, 3.5, this.Rand_Z + 3.777); //P5

verts[5] = new Vector3(this.Rand_X - 3.5, 2, this.Rand_Z + 3.777); //P6

verts[6] = new Vector3(this.Rand_X - 2.95, 2, this.Rand_Z + 3.777); //P7

verts[7] = new Vector3(this.Rand_X - 2.95, 2.5, this.Rand_Z + 3.777); //P8

verts[8] = new Vector3(this.Rand_X - 2.9, 2.8, this.Rand_Z + 3.777); //P9

verts[9] = new Vector3(this.Rand_X - 2.75, 3, this.Rand_Z + 3.777); //P10

verts[10] = new Vector3(this.Rand_X - 2.55, 3.1, this.Rand_Z + 3.777); //P11

verts[11] = new Vector3(this.Rand_X - 2.3,  3.1, this.Rand_Z + 3.777); //P12

verts[12] = new Vector3(this.Rand_X - 2.1, 3, this.Rand_Z + 3.777); //P13

verts[13] = new Vector3(this.Rand_X - 1.95, 2.8, this.Rand_Z + 3.777); //P14


//normals

normals[0] = new Vector3(0, 0, 1);
```

```
normals[1] = new Vector3(0, 0, 1);
normals[2] = new Vector3(0, 0, 1);
normals[3] = new Vector3(0, 0, 1);
normals[4] = new Vector3(0, 0, 1);
normals[5] = new Vector3(0, 0, 1);
normals[6] = new Vector3(0, 0, 1);
normals[7] = new Vector3(0, 0, 1);
normals[8] = new Vector3(0, 0, 1);
normals[9] = new Vector3(0, 0, 1);
normals[10] = new Vector3(0, 0, 1);
normals[11] = new Vector3(0, 0, 1);
normals[12] = new Vector3(0, 0, 1);
normals[13] = new Vector3(0, 0, 1);

//uv mapping (textures)
uv[0] = new Vector2(0, 0.3);
uv[1] = new Vector2(0.2, 0);
uv[2] = new Vector2(0, 0);
uv[3] = new Vector2(0, 1);
uv[4] = new Vector2(1, 1);
uv[5] = new Vector2(1, 0);
uv[6] = new Vector2(0.8, 0);
uv[7] = new Vector2(0.8, 0.3);
uv[8] = new Vector2(0.75, 0.5);
uv[9] = new Vector2(0.65, 0.7);
uv[10] = new Vector2(0.6, 0.9);
uv[11] = new Vector2(0.4, 0.9);
```

```
uv[12] = new Vector2(0.45, 0.7);

uv[13] = new Vector2(0.35, 0.5);


//triangles

tri[0] = 0; //refer to vertices: verts[0]

tri[1] = 1; //refer to vertices: verts[1]

tri[2] = 2; //refer to vertices: verts[2]


tri[3] = 0;

tri[4] = 2;

tri[5] = 3;


tri[6] = 0;

tri[7] = 3;

tri[8] = 13;


tri[9] = 13;

tri[10] = 3;

tri[11] = 12;


tri[12] = 12;

tri[13] = 3;

tri[14] = 11;


tri[15] = 11;

tri[16] = 3;

tri[17] = 4;
```

```
tri[18] = 11;
tri[19] = 4;
tri[20] = 10;

tri[21] = 10;
tri[22] = 4;
tri[23] = 9;

tri[24] = 9;
tri[25] = 4;
tri[26] = 8;

tri[27] = 8;
tri[28] = 4;
tri[29] = 7;

tri[30] = 7;
tri[31] = 4;
tri[32] = 5;

tri[33] = 7;
tri[34] = 5;
tri[35] = 6;

tri[36] = 0;
tri[37] = 2;
```

```
tri[38] = 1;


tri[39] = 0;
tri[40] = 3;
tri[41] = 2;


tri[42] = 13;
tri[43] = 3;
tri[44] = 0;


tri[45] = 12;
tri[46] = 3;
tri[47] = 13;


tri[48] = 11;
tri[49] = 3;
tri[50] = 12;


tri[51] = 10;
tri[52] = 3;
tri[53] = 11;


tri[54] = 10;
tri[55] = 4;
tri[56] = 3;


tri[57] = 9;
```

```
tri[58] = 4;
tri[59] = 10;

tri[60] = 8;
tri[61] = 4;
tri[62] = 9;

tri[63] = 7;
tri[64] = 4;
tri[65] = 8;

tri[66] = 7;
tri[67] = 5;
tri[68] = 4;

tri[69] = 7;
tri[70] = 6;
tri[71] = 5;

var mesh: Mesh = new Mesh();
mesh.name = "Arch";
mesh.vertices = verts;
mesh.triangles = tri;
mesh.uv = uv;
mesh.normals = normals;
//mesh.RecalculateNormals();
```

```
        tmpobj.renderer.material.mainTexture = Resources.Load("white_brick_wall");

        mf.mesh = mesh;

}//end function Arch(Rand_X, Rand_Z)



private function ArchVertical(Rand_X, Rand_Z, tmpobj){


        tmpobj = new GameObject();

        var mf: MeshFilter = tmpobj.AddComponent(MeshFilter);


        tmpobj.AddComponent(MeshRenderer);

        tmpobj.name = "Arch vert";

        tmpobj.renderer.material.color = Color(0.827450, 0.827450, 0.827450, 0);

        tmpobj.transform.parent = oBuilding5.transform;


        var verts: Vector3[] = new Vector3[14];

        var uv: Vector2[] = new Vector2[14];

        var normals: Vector3[] = new Vector3[14];

        var tri: int[] = new int[72]; //3 vertices * 12 triangles = 36 * 2 (duplicate triangles for the back face) =72


        //vertices positioning

        verts[0] = new Vector3(this.Rand_X , 2.5, this.Rand_Z - 1.9); //P1

        verts[1] = new Vector3(this.Rand_X , 2, this.Rand_Z - 1.9); //P2

        verts[2] = new Vector3(this.Rand_X , 2, this.Rand_Z - 1.35); //P3

        verts[3] = new Vector3(this.Rand_X , 3.5, this.Rand_Z - 1.35); //P4

        verts[4] = new Vector3(this.Rand_X , 3.5, this.Rand_Z - 3.5); //P5
```

```
verts[5] = new Vector3(this.Rand_X , 2, this.Rand_Z - 3.5); //P6
verts[6] = new Vector3(this.Rand_X , 2, this.Rand_Z - 2.95); //P7
verts[7] = new Vector3(this.Rand_X , 2.5, this.Rand_Z - 2.95); //P8
verts[8] = new Vector3(this.Rand_X , 2.8, this.Rand_Z - 2.9); //P9
verts[9] = new Vector3(this.Rand_X , 3, this.Rand_Z - 2.75); //P10
verts[10] = new Vector3(this.Rand_X , 3.1, this.Rand_Z - 2.55); //P11
verts[11] = new Vector3(this.Rand_X ,  3.1, this.Rand_Z - 2.3); //P12
verts[12] = new Vector3(this.Rand_X , 3, this.Rand_Z - 2.1); //P13
verts[13] = new Vector3(this.Rand_X , 2.8, this.Rand_Z - 1.95); //P14

//normals
normals[0] = new Vector3(0, 0, 1);
normals[1] = new Vector3(0, 0, 1);
normals[2] = new Vector3(0, 0, 1);
normals[3] = new Vector3(0, 0, 1);
normals[4] = new Vector3(0, 0, 1);
normals[5] = new Vector3(0, 0, 1);
normals[6] = new Vector3(0, 0, 1);
normals[7] = new Vector3(0, 0, 1);
normals[8] = new Vector3(0, 0, 1);
normals[9] = new Vector3(0, 0, 1);
normals[10] = new Vector3(0, 0, 1);
normals[11] = new Vector3(0, 0, 1);
normals[12] = new Vector3(0, 0, 1);
normals[13] = new Vector3(0, 0, 1);

//uv mapping (textures)
```

```
uv[0] = new Vector2(0, 0.3);
uv[1] = new Vector2(0.2, 0);
uv[2] = new Vector2(0, 0);
uv[3] = new Vector2(0, 1);
uv[4] = new Vector2(1, 1);
uv[5] = new Vector2(1, 0);
uv[6] = new Vector2(0.8, 0);
uv[7] = new Vector2(0.8, 0.3);
uv[8] = new Vector2(0.75, 0.5);
uv[9] = new Vector2(0.65, 0.7);
uv[10] = new Vector2(0.6, 0.9);
uv[11] = new Vector2(0.4, 0.9);
uv[12] = new Vector2(0.45, 0.7);
uv[13] = new Vector2(0.35, 0.5);


//triangles
tri[0] = 0; //refer to vertices: verts[0]
tri[1] = 1; //refer to vertices: verts[1]
tri[2] = 2; //refer to vertices: verts[2]

tri[3] = 0;
tri[4] = 2;
tri[5] = 3;

tri[6] = 0;
tri[7] = 3;
```

```
tri[8] = 13;

tri[9] = 13;
tri[10] = 3;
tri[11] = 12;

tri[12] = 12;
tri[13] = 3;
tri[14] = 11;

tri[15] = 11;
tri[16] = 3;
tri[17] = 4;

tri[18] = 11;
tri[19] = 4;
tri[20] = 10;

tri[21] = 10;
tri[22] = 4;
tri[23] = 9;

tri[24] = 9;
tri[25] = 4;
tri[26] = 8;

tri[27] = 8;
```

```
tri[28] = 4;
tri[29] = 7;

tri[30] = 7;
tri[31] = 4;
tri[32] = 5;

tri[33] = 7;
tri[34] = 5;
tri[35] = 6;

tri[36] = 0;
tri[37] = 2;
tri[38] = 1;

tri[39] = 0;
tri[40] = 3;
tri[41] = 2;

tri[42] = 13;
tri[43] = 3;
tri[44] = 0;

tri[45] = 12;
tri[46] = 3;
tri[47] = 13;
```

```
tri[48] = 11;
tri[49] = 3;
tri[50] = 12;

tri[51] = 10;
tri[52] = 3;
tri[53] = 11;

tri[54] = 10;
tri[55] = 4;
tri[56] = 3;

tri[57] = 9;
tri[58] = 4;
tri[59] = 10;

tri[60] = 8;
tri[61] = 4;
tri[62] = 9;

tri[63] = 7;
tri[64] = 4;
tri[65] = 8;

tri[66] = 7;
tri[67] = 5;
tri[68] = 4;
```

```
        tri[69] = 7;
        tri[70] = 6;
        tri[71] = 5;

        var mesh: Mesh = new Mesh();
        mesh.name = "Arch vert";
        mesh.vertices = verts;
        mesh.triangles = tri;
        mesh.uv = uv;
        mesh.normals = normals;
        //mesh.RecalculateNormals();
        tmpobj.renderer.material.mainTexture =
Resources.Load("white_brick_wall");
        mf.mesh = mesh;

}//end function ArchVertical(Rand_X, Rand_Z, tmpobj)

public function DoArch(Rand_X, Rand_Z){
        tmpobj = new GameObject();
        Destroy(tmpobj);

        var nPOSZ:float = this.Rand_Z + 0.777;

        for(i=0; i<=3; i++){
                if(tmpobj){
                        Arch(this.Rand_X, nPOSZ, tmpobj);
```

```
                this.Rand_X += 1.6;
            }//end if
        }//end for

        this.Rand_X -= 9.96;
        this.Rand_Z +=5.15;

        for(i=0; i<=1; i++){
            if(tmpobj){
                ArchVertical(this.Rand_X, this.Rand_Z, tmpobj);
                this.Rand_X += 7.1;
            }//end if
        }//end for
}//end function DoArch(Rand_X, Rand_Z)
//---------------------------------END ARCH-----------------------------------------------------------
```

//House_6.js

```
//--Main Building Objects-------------------------------------------------------
var oBuilding6 : GameObject;
var oWindow : GameObject;
var oCube : GameObject;
//-------------------start public variables-------------------------------------------------------
public var aCollectionObj_6 = new Array(); // an Array - a collection- for the objects of House_2
public var i: int = 0;
public var j: int = 0;
```

```
public var c: int = 0;


public var RandX: float;
public var RandZ: float;
//-------------------end public variables-------------------------------------------------------


//MAIN function
function mainDO() {



        MainBuilding(RandX, RandZ);
        Windows(RandX, RandZ);
        Eaves(RandX, RandZ);
        Roof(RandX, RandZ);
}//end function Start()




public function Destroy_(){

DestroyImmediate(oBuilding6);

}



//---Start  MainBuilding-------------------------------------------------------------------------
-----
```

```
public function MainBuilding(RandX, RandZ){
        oBuilding6 = GameObject.CreatePrimitive(PrimitiveType.Cube);
        oBuilding6.transform.position = Vector3(this.RandX, 6.35569, this.RandZ);
        oBuilding6.name = "House_6";
        oBuilding6.renderer.material.color = Color(Random.Range(0.95294, 0.85294), Random.Range(0.894117, 0.794117), Random.Range(0.7647058, 0.6647058), 0); //set color
        oBuilding6.transform.localScale.x = 4.637413;
        oBuilding6.transform.localScale.y = 12.0;
        oBuilding6.transform.localScale.z = 10.23788;
        aCollectionObj_6.Add(oBuilding6);
}//end function MainBuilding(RandX, RandZ)
//---End  MainBuilding----------

//------------Start Windows----------
public function Windows(RandX, RandZ){
        oWindow = GameObject.CreatePrimitive(PrimitiveType.Cube);
        Destroy(oWindow);
        //oWindow.transform.renderer.material.color = Color.clear;
        oWindow.renderer.material.mainTexture = Resources.Load("window_corfu1"); //import texture for windows

        oCube = GameObject.CreatePrimitive(PrimitiveType.Cube);
        Destroy(oCube);

        var wind_X:float = this.RandX + 0.84427;
        var wind_Y:float = 11.474193;
```

```
var wind_Z:float = this.RandZ + 5.07396;
//Start front side::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
//small windows
for(i=0; i<=1; i++){
    for(j=0; j<=1; j++){
        if(oWindow){
            var oWindow_ = Instantiate(oWindow, Vector3(wind_X,wind_Y,wind_Z), Quaternion.identity);
            oWindow_.transform.parent = oBuilding6.transform; //parenting
            oWindow_.name = "window front side " + (++c);
            oWindow_.transform.localScale.x = 0.1562559;
            oWindow_.transform.localScale.y = 0.08680711;
            oWindow_.transform.localScale.z = 0.019;
            aCollectionObj_6.Add(oWindow_);
        }//end if
        wind_Y -= 7.88;
    }//end for
    wind_Y = 11.474193;
    wind_X -= 1.75;
}//end for
c = 0;

//balcony windows
wind_X = this.RandX + 0.84427;
wind_Y = 9.074193;
wind_Z = this.RandZ + 5.07396;
```

```
for(i=0; i<=1; i++){
    for(j=0; j<=1; j++){
        if(oWindow){
            var oBalconyWindow_ = Instantiate(oWindow, Vector3(wind_X,wind_Y,wind_Z), Quaternion.identity);
            oBalconyWindow_.transform.parent = oBuilding6.transform; //parenting
            oBalconyWindow_.name = "balcony window front side " + (++c);
            oBalconyWindow_.transform.localScale.x = 0.1562559;
            oBalconyWindow_.transform.localScale.y = 0.1475996;
            oBalconyWindow_.transform.localScale.z = 0.019;
            aCollectionObj_6.Add(oBalconyWindow_);
        }//end if
        wind_Y -= 2.88;
    }//end for
    wind_Y = 9.074193;
    wind_X -= 1.75;
}//end for
c = 0;

//eaves
wind_X = this.RandX + 0.84427;
wind_Y = 9.974193;
wind_Z = this.RandZ + 5.1207;

for(i=0; i<=1; i++){
```

```
for(j=0; j<=1; j++){
    for(k=0; k<=1; k++){
        if(oCube){
            var oWindowEaves_ = Instantiate(oCube, Vector3(wind_X,wind_Y,wind_Z), Quaternion.identity);
            oWindowEaves_.transform.parent = oBuilding6.transform; //parenting
            oWindowEaves_.name = "window front side eaves" + (++c);
            if( k == 1 ){
                oWindowEaves_.transform.localScale.x = 0.1777411;
                oWindowEaves_.transform.localScale.z = 0.01728884;
            }//end if
            else{
                oWindowEaves_.transform.localScale.x = 0.1562559;
                oWindowEaves_.transform.localScale.z = 0.01329912;
            }//else
            oWindowEaves_.transform.localScale.y = 0.005307435;
            aCollectionObj_6.Add(oWindowEaves_);
        }//end if
        wind_Y += 0.05;
    }//end for
    wind_Y = 9.974193;
    wind_Y -= 5.88;
}//end for
```

```
            wind_Y = 9.974193;
            wind_X -= 1.75;
        }//end for
        c = 0;

        //balcony base
        wind_Y = 8.1;
        wind_Z = this.RandZ + 5.1207;

        for(i=0; i<=1; i++){
            if(oCube){
                var oBalconyBase_ = Instantiate(oCube, Vector3(this.RandX, wind_Y, wind_Z), Quaternion.identity);
                oBalconyBase_.transform.parent = oBuilding6.transform; //parenting
                oBalconyBase_.name = "balcony base " + (++c);
                oBalconyBase_.transform.localScale.x = 0.6634907;
                oBalconyBase_.transform.localScale.y = 0.01063777;
                oBalconyBase_.transform.localScale.z = 0.1267186;
                aCollectionObj_6.Add(oBalconyBase_);
            }//end if
            wind_Y -= 2.89;
        }//end for
        c = 0;

        //balcony rails
        wind_X = this.RandX + 1.5128;
        wind_Y = 8.5;
```

```
wind_Z = this.RandZ + 5.7490;


for(i=0; i<=15; i++){
    for(j=0; j<=1; j++){
        if(oCube){
            var oBalconyRails_ = Instantiate(oCube, Vector3(wind_X, wind_Y, wind_Z), Quaternion.identity);
            oBalconyRails_.transform.parent = oBuilding6.transform; //parenting
            oBalconyRails_.name = "balcony rails " + (++c);
            oBalconyRails_.transform.localScale.x = 0.008769402;
            oBalconyRails_.transform.localScale.y = 0.05327845;
            oBalconyRails_.transform.localScale.z = 0.003533184;
            aCollectionObj_6.Add(oBalconyRails_);
        }//end if
        wind_Y -= 2.89;
    }//end for
    wind_Y = 8.5;
    wind_X -= 0.2;
}//end for
c = 0;


//balcony left rails
wind_X = this.RandX + 1.5128;
wind_Y = 8.5;
wind_Z = this.RandZ + 5.7490;


for(i=0; i<=3; i++){
```

```
            for(j=0; j<=1; j++){
                if(oCube){
                    var oBalconyLftRails_ = Instantiate(oCube, Vector3(wind_X, wind_Y, wind_Z), Quaternion.identity);
                    oBalconyLftRails_.transform.parent = oBuilding6.transform; //parenting
                    oBalconyLftRails_.name = "balcony left rails " + (++c);
                    oBalconyLftRails_.transform.localScale.x = 0.008769402;
                    oBalconyLftRails_.transform.localScale.y = 0.05327845;
                    oBalconyLftRails_.transform.localScale.z = 0.003533184;
                    aCollectionObj_6.Add(oBalconyLftRails_);
                }//end if
                wind_Y -= 2.89;
            }//end for
            wind_Y = 8.5;
            wind_Z -= 0.2;
        }//end for
        c = 0;

        //balcony right rails
        wind_X = this.RandX - 1.4872;
        wind_Y = 8.5;
        wind_Z = this.RandZ + 5.7490;

        for(i=0; i<=3; i++){
            for(j=0; j<=1; j++){
```

```
                    if(oCube){
                        var oBalconyRghtRails_ = Instantiate(oCube, Vector3(wind_X, wind_Y, wind_Z), Quaternion.identity);
                        oBalconyRghtRails_.transform.parent = oBuilding6.transform; //parenting
                        oBalconyRghtRails_.name = "balcony right rails " + (++c);
                        oBalconyRghtRails_.transform.localScale.x = 0.008769402;
                        oBalconyRghtRails_.transform.localScale.y = 0.05327845;
                        oBalconyRghtRails_.transform.localScale.z = 0.003533184;
                        aCollectionObj_6.Add(oBalconyRghtRails_);
                    }//end if
                    wind_Y -= 2.89;
                }//end for
                wind_Y = 8.5;
                wind_Z -= 0.2;
            }//end for
            c = 0;

            //balcony main rails
            wind_X = this.RandX + 0.016416;
            wind_Y = 8.8;
            wind_Z = this.RandZ + 5.7490;

            for(i=0; i<=3; i++){
                if(oWindow){
```

```
                var oMainRails_ = Instantiate(oCube, Vector3(wind_X, wind_Y, wind_Z), Quaternion.identity);
                oMainRails_.transform.parent = oBuilding6.transform; //parenting
                oMainRails_.name = "main rails " + (++c);
                oMainRails_.transform.localScale.x = 0.003585278;
                oMainRails_.transform.localScale.y = 0.6571109;
                oMainRails_.transform.localScale.z = 0.003533184;
                oMainRails_.transform.eulerAngles.z = 90;
                aCollectionObj_6.Add(oMainRails_);
        }//end if
        if( i == 1)
                wind_Y -= 2.29;
        else
                wind_Y -= 0.6;
}//end for
c = 0;

//balcony main side rails
wind_X = this.RandX + 1.5128;
wind_Y = 8.8;
wind_Z = this.RandZ + 5.41732;

for(i=0; i<=1; i++){
        for(j=0; j<=3; j++){
                if(oCube){
                        var oMainLftRails_ = Instantiate(oCube, Vector3(wind_X, wind_Y, wind_Z), Quaternion.identity);
```

```
                        oMainLftRails_.transform.parent = oBuilding6.transform; //parenting
                        oMainLftRails_.name = "main side rails " + (++c);
                        oMainLftRails_.transform.localScale.x = 0.008769402;
                        oMainLftRails_.transform.localScale.y = 0.06881529;
                        oMainLftRails_.transform.localScale.z = 0.003533184;
                        oMainLftRails_.transform.eulerAngles.x = 90;
                        aCollectionObj_6.Add(oMainLftRails_);
                    }//end if
                    if( j == 1)
                        wind_Y -= 2.29;
                    else
                        wind_Y -= 0.6;
                }//end for
                wind_Y = 8.8;
                wind_X -= 3;
            }//end for
            c = 0;

            //door
            wind_X = this.RandX;
            wind_Y = 1.3;
            wind_Z = this.RandZ + 5.07396;

            if(oWindow){
                var oDoor_ = Instantiate(oWindow, Vector3(wind_X, wind_Y, wind_Z), Quaternion.identity);
                oDoor_.transform.parent = oBuilding6.transform; //parenting
```

```
		oDoor_.renderer.material.mainTexture = Resources.Load("door_corfu2"); //import texture for door
		oDoor_.name = "door";
		oDoor_.transform.localScale.x = 0.189462;
		oDoor_.transform.localScale.y = 0.1475996;
		oDoor_.transform.localScale.z = 0.019;
		aCollectionObj_6.Add(oDoor_);
	}//end if
	//End front side::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

	//Start back side::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
	//small windows
	wind_X = this.RandX + 0.84427;
	wind_Y = 11.474193;
	wind_Z = this.RandZ - 5.07396;

	for(i=0; i<=1; i++){
		for(j=0; j<=1; j++){
			if(oWindow){
				var oWindow_bck = Instantiate(oWindow, Vector3(wind_X,wind_Y,wind_Z), Quaternion.identity);
				oWindow_bck.transform.parent = oBuilding6.transform; //parenting
				oWindow_bck.name = "window back side " + (++c);
				oWindow_bck.transform.localScale.x = 0.1562559;
				oWindow_bck.transform.localScale.y = 0.08680711;
				oWindow_bck.transform.localScale.z = 0.019;
```

```
                        aCollectionObj_6.Add(oWindow_bck);
                    }//end if
                    wind_Y -= 7.88;
                }//end for
                wind_Y = 11.474193;
                wind_X -= 1.75;
            }//end for
            c = 0;

            //balcony windows
            wind_X = this.RandX + 0.84427;
            wind_Y = 9.074193;
            wind_Z = this.RandZ - 5.07396;

            for(i=0; i<=1; i++){
                for(j=0; j<=1; j++){
                    if(oWindow){
                        var oBalconyWindow_bck = Instantiate(oWindow, Vector3(wind_X,wind_Y,wind_Z), Quaternion.identity);
                        oBalconyWindow_bck.transform.parent = oBuilding6.transform; //parenting
                        oBalconyWindow_bck.name = "balcony window back side " + (++c);
                        oBalconyWindow_bck.transform.localScale.x = 0.1562559;
                        oBalconyWindow_bck.transform.localScale.y = 0.1475996;
                        oBalconyWindow_bck.transform.localScale.z = 0.019;
                        aCollectionObj_6.Add(oBalconyWindow_bck);
```

```
                }//end if
                wind_Y -= 2.88;
            }//end for
            wind_Y = 9.074193;
            wind_X -= 1.75;
        }//end for
        c = 0;

        //eaves
        wind_X = this.RandX + 0.84427;
        wind_Y = 9.974193;
        wind_Z = this.RandZ - 5.1207;

        for(i=0; i<=1; i++){
            for(j=0; j<=1; j++){
                for(k=0; k<=1; k++){
                    if(oCube){
                        var oWindowEaves_bck = Instantiate(oCube, Vector3(wind_X,wind_Y,wind_Z), Quaternion.identity);
                        oWindowEaves_bck.transform.parent = oBuilding6.transform; //parenting
                        oWindowEaves_bck.name = "window back side eaves" + (++c);
                        if( k == 1 ){

        oWindowEaves_bck.transform.localScale.x = 0.1777411;

        oWindowEaves_bck.transform.localScale.z = 0.01728884;
                        }//end if
```

```
                                    else{

        oWindowEaves_bck.transform.localScale.x = 0.1562559;

        oWindowEaves_bck.transform.localScale.z = 0.01329912;
                                    }//else
                                    oWindowEaves_bck.transform.localScale.y =
0.005307435;
                                    aCollectionObj_6.Add(oWindowEaves_bck);
                                }//end if
                                wind_Y += 0.05;
                            }//end for
                            wind_Y = 9.974193;
                            wind_Y -= 5.88;
                        }//end for
                        wind_Y = 9.974193;
                        wind_X -= 1.75;
        }//end for
        c = 0;


        //balcony base
        wind_Y = 8.1;
        wind_Z = this.RandZ - 5.1207;


        for(i=0; i<=1; i++){
                if(oCube){
                        var oBalconyBase_bck = Instantiate(oCube,
Vector3(this.RandX, wind_Y, wind_Z), Quaternion.identity);
```

```
                    oBalconyBase_bck.transform.parent = oBuilding6.transform;
//parenting
                    oBalconyBase_bck.name = "balcony back side base " + (++c);
                    oBalconyBase_bck.transform.localScale.x = 0.6634907;
                    oBalconyBase_bck.transform.localScale.y = 0.01063777;
                    oBalconyBase_bck.transform.localScale.z = 0.1267186;
                    aCollectionObj_6.Add(oBalconyBase_bck);
            }//end if
            wind_Y -= 2.89;
        }//end for
        c = 0;

        //balcony rails
        wind_X = this.RandX + 1.5128;
        wind_Y = 8.5;
        wind_Z = this.RandZ - 5.7490;

        for(i=0; i<=15; i++){
            for(j=0; j<=1; j++){
                if(oCube){
                    var oBalconyRails_bck = Instantiate(oCube,
Vector3(wind_X, wind_Y, wind_Z), Quaternion.identity);
                    oBalconyRails_bck.transform.parent =
oBuilding6.transform; //parenting
                    oBalconyRails_bck.name = "balcony back side rails " +
(++c);
                    oBalconyRails_bck.transform.localScale.x =
0.008769402;
```

```
                    oBalconyRails_bck.transform.localScale.y = 0.05327845;
                    oBalconyRails_bck.transform.localScale.z = 0.003533184;
                    aCollectionObj_6.Add(oBalconyRails_bck);
                }//end if
                wind_Y -= 2.89;
            }//end for
            wind_Y = 8.5;
            wind_X -= 0.2;
        }//end for
        c = 0;

        //balcony left rails
        wind_X = this.RandX + 1.5128;
        wind_Y = 8.5;
        wind_Z = this.RandZ - 5.7490;

        for(i=0; i<=3; i++){
            for(j=0; j<=1; j++){
                if(oCube){
                    var oBalconyLftRails_bck = Instantiate(oCube, Vector3(wind_X, wind_Y, wind_Z), Quaternion.identity);
                    oBalconyLftRails_bck.transform.parent = oBuilding6.transform; //parenting
                    oBalconyLftRails_bck.name = "balcony back side left rails " + (++c);
                    oBalconyLftRails_bck.transform.localScale.x = 0.008769402;
```

```
                        oBalconyLftRails_bck.transform.localScale.y = 0.05327845;
                        oBalconyLftRails_bck.transform.localScale.z = 0.003533184;
                        aCollectionObj_6.Add(oBalconyLftRails_bck);
                    }//end if
                    wind_Y -= 2.89;
                }//end for
                wind_Y = 8.5;
                wind_Z += 0.2;
            }//end for
            c = 0;

            //balcony right rails
            wind_X = this.RandX - 1.4872;
            wind_Y = 8.5;
            wind_Z = this.RandZ - 5.7490;

            for(i=0; i<=3; i++){
                for(j=0; j<=1; j++){
                    if(oCube){
                        var oBalconyRghtRails_bck = Instantiate(oCube, Vector3(wind_X, wind_Y, wind_Z), Quaternion.identity);
                        oBalconyRghtRails_bck.transform.parent = oBuilding6.transform; //parenting
                        oBalconyRghtRails_bck.name = "balcony back side right rails " + (++c);
                        oBalconyRghtRails_bck.transform.localScale.x = 0.008769402;
```

```
                        oBalconyRghtRails_bck.transform.localScale.y = 0.05327845;
                        oBalconyRghtRails_bck.transform.localScale.z = 0.003533184;
                        aCollectionObj_6.Add(oBalconyRghtRails_bck);
                    }//end if
                    wind_Y -= 2.89;
                }//end for
                wind_Y = 8.5;
                wind_Z += 0.2;
            }//end for
            c = 0;

            //balcony main rails
            wind_X = this.RandX + 0.016416;
            wind_Y = 8.8;
            wind_Z = this.RandZ - 5.7490;

            for(i=0; i<=3; i++){
                if(oCube){
                    var oMainRails_bck = Instantiate(oCube, Vector3(wind_X, wind_Y, wind_Z), Quaternion.identity);
                    oMainRails_bck.transform.parent = oBuilding6.transform; //parenting
                    oMainRails_bck.name = "main back rails " + (++c);
                    oMainRails_bck.transform.localScale.x = 0.003585278;
                    oMainRails_bck.transform.localScale.y = 0.6571109;
                    oMainRails_bck.transform.localScale.z = 0.003533184;
                    oMainRails_bck.transform.eulerAngles.z = 90;
```

```
                aCollectionObj_6.Add(oMainRails_bck);
        }//end if
        if( i == 1)
                wind_Y -= 2.29;
        else
                wind_Y -= 0.6;
}//end for
c = 0;

//balcony main side rails
wind_X = this.RandX + 1.5128;
wind_Y = 8.8;
wind_Z = this.RandZ - 5.41732;

for(i=0; i<=1; i++){
        for(j=0; j<=3; j++){
                if(oCube){
                        var oMainLftRails_bck = Instantiate(oCube, Vector3(wind_X, wind_Y, wind_Z), Quaternion.identity);
                        oMainLftRails_bck.transform.parent = oBuilding6.transform; //parenting
                        oMainLftRails_bck.name = "main back side rails " + (++c);
                        oMainLftRails_bck.transform.localScale.x = 0.008769402;
                        oMainLftRails_bck.transform.localScale.y = 0.06881529;
                        oMainLftRails_bck.transform.localScale.z = 0.003533184;
```

```
                    oMainLftRails_bck.transform.eulerAngles.x = 90;
                    aCollectionObj_6.Add(oMainLftRails_bck);
                }//end if
                if( j == 1)
                    wind_Y -= 2.29;
                else
                    wind_Y -= 0.6;
            }//end for
            wind_Y = 8.8;
            wind_X -= 3;
        }//end for
        c = 0;

        //door
        wind_X = this.RandX;
        wind_Y = 1.3;
        wind_Z = this.RandZ - 5.07396;

        if(oWindow){
            var oDoor_bck = Instantiate(oWindow, Vector3(wind_X, wind_Y, wind_Z), Quaternion.identity);
            oDoor_bck.transform.parent = oBuilding6.transform; //parenting
            oDoor_bck.renderer.material.mainTexture = Resources.Load("door_corfu2"); //import texture for door
            oDoor_bck.name = "back door";
            oDoor_bck.transform.localScale.x = 0.189462;
            oDoor_bck.transform.localScale.y = 0.1475996;
            oDoor_bck.transform.localScale.z = 0.019;
```

```
                aCollectionObj_6.Add(oDoor_bck);
        }//end if
        //End back side:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
}//end function Windows(RandX, RandZ)
//------------End Windows-------------------------------------------------------------------------



//------------------Start Eaves------------------------------------------------------------------
public function Eaves(RandX, RandZ){
        oCube = GameObject.CreatePrimitive(PrimitiveType.Cube);
        Destroy(oCube);

        var eaves_posX: float = this.RandX;
        var eaves_posY: float = 12.374193;
        var eaves_posZ: float = this.RandZ;

        for(i=0; i<=2; i++){
                if(oCube){
                        var oEaves_ = Instantiate(oCube, Vector3(eaves_posX, eaves_posY, eaves_posZ), Quaternion.identity);
                        oEaves_.transform.parent = oBuilding6.transform; //parenting
                        oEaves_.name = "eaves " + (++c);
                        if(i == 1){
                                oEaves_.transform.localScale.x = 1.061839;
                                oEaves_.transform.localScale.z = 1.058063;
                        }//end if
                        else if(i == 2){
```

```
                        oEaves_.transform.localScale.x = 1.089062;
                        oEaves_.transform.localScale.z = 1.07113;
                    }//end else if
                    else{
                        oEaves_.transform.localScale.x = 1.025;
                        oEaves_.transform.localScale.z = 1.045;
                    }//else
                    oEaves_.transform.localScale.y = 0.01419581;
                    aCollectionObj_6.Add(oEaves_);
                }//end if
                eaves_posY += 0.1;
            }//end for
            c = 0;
}//end function Eaves(RandX, RandZ)
//---------------End  Eaves-------------------------------------------------------------------------------
-----



//--------Start Roof-------------------------------------------------------------------------------
public function Roof(RandX, RandZ){
            var tmpobj: GameObject = new GameObject();
            var mf: MeshFilter = tmpobj.AddComponent(MeshFilter);

            tmpobj.AddComponent(MeshRenderer);
            tmpobj.name = "Roof";
            tmpobj.transform.parent = oBuilding6.transform;
```

```
var verts: Vector3[] = new Vector3[6];
var normals: Vector3[] = new Vector3[6];
var uv: Vector2[] = new Vector2[6];
var tri: int[] = new int[18]; //3 vertices * 6 triangles


//vertices positioning
verts[0] = new Vector3(this.RandX - 2.5, 12.7, this.RandZ - 5.5); //P1
verts[1] = new Vector3(this.RandX - 2.5, 12.7, this.RandZ + 5.5); //P2
verts[2] = new Vector3(this.RandX, 13.7, this.RandZ + 4); //P3

verts[3] = new Vector3(this.RandX + 2.5, 12.7, this.RandZ + 5.5); //P4
verts[4] = new Vector3(this.RandX + 2.5, 12.7, this.RandZ - 5.5); //P5
verts[5] = new Vector3(this.RandX, 13.7, this.RandZ - 4); //P6


uv[0] = new Vector2(1, 1);
uv[1] = new Vector2(1, 0);
uv[2] = new Vector2(0.5, 0.2);

uv[3] = new Vector2(0, 0);
uv[4] = new Vector2(0, 1);
uv[5] = new Vector2(0.5, 0.8);


//triangles
```

```
tri[0] = 5; //refer to vertices: verts[0]
tri[1] = 4; //refer to vertices: verts[1]
tri[2] = 0; //refer to vertices: verts[2]

tri[3] = 5;
tri[4] = 0;
tri[5] = 2;

tri[6] = 2;
tri[7] = 0;
tri[8] = 1;

tri[9] = 2;
tri[10] = 1;
tri[11] = 3;

tri[12] = 2;
tri[13] = 3;
tri[14] = 4;

tri[15] = 2;
tri[16] = 4;
tri[17] = 5;

var mesh: Mesh = new Mesh();
mesh.name = "Roof";
mesh.vertices = verts;
```

```
        mesh.triangles = tri;
        mesh.uv = uv;
        mesh.normals = normals;
        mesh.RecalculateNormals();
        mf.mesh = mesh;
        tmpobj.renderer.material.mainTexture = Resources.Load("tiled_roof_red_grey");
}//end function Roof(RandX, RandZ)
//--------------------------------------End Roof-----------------------------------------------------
```

//House_7.js

```
//create objects
var oBuilding7 : GameObject;
var oWindow : GameObject;
var oCube : GameObject;
var tmpobj: GameObject;

//-------------------start public variables-------------------------------------------------------
public var aCollectionObj_7 = new Array(); // an Array - a collection- for the objects of House_7

public var i: int = 0;
public var j: int = 0;
public var c: int = 0;

//Random ranges between X, Y coordinates
public var RandX:float;
public var RandZ:float;
```

```
//-------------------end public variables-----------------------------------------------------


//MAIN function
public function mainDO() {


        MainBuilding(RandX,RandZ);
        FirstFloor(RandX,RandZ);
        SecondFloor(RandX, RandZ);
        ThirdFloor(RandX,RandZ);
        Eaves(RandX, RandZ);
        Windows(RandX, RandZ);
        Columns(RandX, RandZ);
        Door(RandX, RandZ);
        Roof(RandX, RandZ);
        DoArch(RandX, RandZ);
}//end Start ()



public function Destroy_(){

Destroy(oBuilding7);

}
```

```
//-----------Main Building-ground floor--------------------------

public function MainBuilding(RandX,RandZ){

        oBuilding7 = GameObject.CreatePrimitive(PrimitiveType.Cube);

        oBuilding7.renderer.material.color = Color(Random.Range(0.91372,0.81372), Random.Range(0.81960,0.71960), Random.Range(0.76470,0.66470), 0);

        oBuilding7.transform.position = Vector3(this.RandX, 1.722107, this.RandZ);

        oBuilding7.name = "House_7";

        oBuilding7.transform.localScale.x = 7;

        oBuilding7.transform.localScale.y = 3.308559;

        oBuilding7.transform.localScale.z = 3.5;

        aCollectionObj_7.Add(oBuilding7);

}//end function MainBuilding(Rand_X,Rand_Z)



//----------------------------------------START FIRST FLOOR----------------------------------
-------------------------

public function FirstFloor(RandX,RandZ){

        oCube = GameObject.CreatePrimitive(PrimitiveType.Cube);

        //solving an issue appearing: destroy visible instances appear as common cubes on the scene

        Destroy (oCube);

        oCube.renderer.material.color = Color(0.91372, 0.81960, 0.76470, 0);

        //First floor

        var oFrst_Flr = Instantiate(oCube, Vector3(this.RandX, 4.527526, this.RandZ + 0.9), Quaternion.identity);

        oFrst_Flr.transform.parent = oBuilding7.transform; //parenting

        oFrst_Flr.name = "First Floor";
```

```
        oFrst_Flr.transform.localScale.x = 0.9954391;

        oFrst_Flr.transform.localScale.y = 0.7123187;

        oFrst_Flr.transform.localScale.z = 1.536353;

        aCollectionObj_7.Add(oFrst_Flr);

}//end function FirstFloor(RandX,RandZ)

//---------------------------------------END FIRST FLOOR-----------------------------------------
---------------------


//---------------------------------------START SECOND FLOOR---------------------------------------
-----------------------------

public function SecondFloor(RandX, RandZ){

        oCube = GameObject.CreatePrimitive(PrimitiveType.Cube);

        //solving an issue appearing: destroy visible instances appear as common
cubes on the scene

        Destroy (oCube);

        oCube.renderer.material.color = Color(0.91372, 0.81960, 0.76470, 0);

        //Second Floor

        var oScnd_Floor = Instantiate(oCube, Vector3(this.RandX, 7.2189827,
this.RandZ + 0.9), Quaternion.identity);

        oScnd_Floor.transform.parent = oBuilding7.transform; //parenting

        oScnd_Floor.name = "Second Floor";

        oScnd_Floor.transform.localScale.x = 0.9954391;

        oScnd_Floor.transform.localScale.y = 0.9377826;

        oScnd_Floor.transform.localScale.z = 1.536353;

        aCollectionObj_7.Add(oScnd_Floor);

}//end function SecondFloor(RandX,RandZ)

//---------------------------------------END SECOND FLOOR-----------------------------------------
-------------------------
```

```
//-----------------------------------------START THIRD FLOOR-------------------------------------------------------

public function ThirdFloor(RandX,RandZ){

	oCube = GameObject.CreatePrimitive(PrimitiveType.Cube);

	Destroy (oCube);

	//Third floor

	oCube.renderer.material.color = Color(0.91372, 0.81960, 0.76470, 0);

	var oThrd_Flr = Instantiate(oCube, Vector3(this.RandX, 10.027526, this.RandZ + 0.9), Quaternion.identity);

	oThrd_Flr.transform.parent = oBuilding7.transform; //parenting

	oThrd_Flr.name = "Third Floor";

	oThrd_Flr.transform.localScale.x = 0.9954391;

	oThrd_Flr.transform.localScale.y = 0.7123187;

	oThrd_Flr.transform.localScale.z = 1.536353;

	aCollectionObj_7.Add(oThrd_Flr);

}//end function FirstFloor(RandX,RandZ)

//-----------------------------------------END THIRD FLOOR---------------------------------------------------------
```

```
//-----------------------------------------START EAVES-------------------------------------------------------

public function Eaves(RandX, RandZ){

	oCube = GameObject.CreatePrimitive(PrimitiveType.Cube);

	//solving an issue appearing: destroy visible instances appear as common cubes on the scene

	Destroy (oCube);
```

```
		oCube.renderer.material.color = Color.white;
		var nPOSY_eaves:float = 3.322107;
		for(i=0; i<=2; i++){
			if(oCube){
				var oEaves_flr = Instantiate(oCube, Vector3(this.RandX, nPOSY_eaves, this.RandZ + 0.9), Quaternion.identity);
				oEaves_flr.transform.parent = oBuilding7.transform; //parenting
				oEaves_flr.name = "eaves " + (++c);
				oEaves_flr.transform.localScale.x = 1.020326;
				oEaves_flr.transform.localScale.y = 0.06;
				oEaves_flr.transform.localScale.z = 1.593966;
				aCollectionObj_7.Add(oEaves_flr);
				if(i == 1)
					nPOSY_eaves += 3.2030433;
				else
					nPOSY_eaves += 2.368318;
			}//end if
		}//end for
		c = 0;
}//end function Eaves(RandX, RandZ)
//----------------------------------------END EAVES-----------------------------------------------------------


//----------------------------------START WINDOWS-----------------------------------------------------------
public function Windows(RandX, RandZ){
```

```
oWindow = GameObject.CreatePrimitive(PrimitiveType.Cube);

//solving an issue appearing: destroy visible instances appear as common cubes on the scene

Destroy(oWindow);


//transparent windows

//oWindow.transform.renderer.material.color = Color.clear;

oWindow.renderer.material.mainTexture = Resources.Load("window_corfu3"); //import texture for windows


var nWind_X:float = this.RandX + 2.82102;

var nWind_Y:float = 5.0751646;

var nWind_Z:float = this.RandZ + 3.6062495;

//::::::::::::::::::::::::::::::::::::::::start facade::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

//first floor windows

for(i=0; i<=3; i++){

    if(oWindow){

        var oWindow_frstFlr = Instantiate(oWindow, Vector3(nWind_X, nWind_Y, nWind_Z), Quaternion.identity);

        oWindow_frstFlr.transform.parent = oBuilding7.transform; //parenting

        oWindow_frstFlr.name = "first floor window " + (++c);

        oWindow_frstFlr.transform.localScale.x = 0.1135587;

        oWindow_frstFlr.transform.localScale.y = 0.2774929;

        oWindow_frstFlr.transform.localScale.z = 0.0163088;

        aCollectionObj_7.Add(oWindow_frstFlr);

        nWind_X -= 1.9;

    }//end if
```

```
	}//end for
	c = 0;


	nWind_X = this.RandX + 2.82102;
	var wind_posY:float = nWind_Y + 2.6442788;
//second floor windows
	for(i=0; i<=3; i++){
		if(oWindow){
			var oWindow_scndFlr = Instantiate(oWindow, Vector3(nWind_X, wind_posY, nWind_Z), Quaternion.identity);
			oWindow_scndFlr.transform.parent = oBuilding7.transform; //parenting
			oWindow_scndFlr.name = "second floor window " + (++c);
			if(i == 1 || i == 2){
				oWindow_scndFlr.transform.localScale.x = 0.1192862;
				oWindow_scndFlr.transform.localScale.y = 0.6036146;
			}
			else{
				oWindow_scndFlr.transform.localScale.x = 0.1096849;
				oWindow_scndFlr.transform.localScale.y = 0.4051766;
			}//end else
			oWindow_scndFlr.transform.localScale.z = 0.0163088;
			aCollectionObj_7.Add(oWindow_scndFlr);
			nWind_X -= 1.9;
			if( i == 0 || i == 1)
				wind_posY = nWind_Y + 2.3;
			else
				wind_posY = nWind_Y + 2.6442788;
```

```
            }//end if
        }//end for
        c = 0;

        //third floor windows
        nWind_X = this.RandX + 2.82102;
        nWind_Z= this.RandZ + 3.6062495;

        for(i=0; i<=3; i++){
            if(oWindow){
                var oWindow_thrdFlr = Instantiate(oWindow, Vector3(nWind_X, nWind_Y + 5, nWind_Z), Quaternion.identity);
                oWindow_thrdFlr.transform.parent = oBuilding7.transform; //parenting
                oWindow_thrdFlr.name = "third floor window " + (++c);
                oWindow_thrdFlr.transform.localScale.x = 0.1135587;
                oWindow_thrdFlr.transform.localScale.y = 0.2774929;
                oWindow_thrdFlr.transform.localScale.z = 0.0163088;
                aCollectionObj_7.Add(oWindow_thrdFlr);
                nWind_X -= 1.9;
            }//end if
        }//end for
        c = 0;

        //baclony base
        var blcn_X:float = this.RandX;
        var blcn_Y:float = nWind_Y + 1.22;
        var blcn_Z:float = this.RandZ + 3.5662495;
```

```
if(oCube){
        var oBalcony_base = Instantiate(oCube, Vector3(blcn_X, blcn_Y, blcn_Z), Quaternion.identity);
        oBalcony_base.transform.parent = oBuilding7.transform; //parenting
        oBalcony_base.name = "balcony base";
        oBalcony_base.transform.localScale.x = 0.5252227;
        oBalcony_base.transform.localScale.y = 0.05103775;
        oBalcony_base.transform.localScale.z = 0.3886753;
        aCollectionObj_7.Add(oBalcony_base);
}//end if

//balcony front side rails
blcn_X = this.RandX + 1.824236;
blcn_Y = nWind_Y + 1.6;
blcn_Z = this.RandZ + 4.23007;

for(i=0; i<=24; i++){
        if(oCube){
                var oBalconyRails_ = Instantiate(oCube, Vector3(blcn_X, blcn_Y, blcn_Z), Quaternion.identity);
                oBalconyRails_.transform.parent = oBuilding7.transform; //parenting
                oBalconyRails_.name = "balcony front rails " + (++c);
                oBalconyRails_.transform.localScale.x = -0.003082343;
                oBalconyRails_.transform.localScale.y = 0.1696285;
                oBalconyRails_.transform.localScale.z = 0.007928168;
                aCollectionObj_7.Add(oBalconyRails_);
```

```
            }//end if
            blcn_X -= 0.15;
        }//end for
        c = 0;

        //balcony side rails
        blcn_X = this.RandX + 1.824236;
        blcn_Z = this.RandZ + 4.23007;

        for(i=0; i<=4; i++){
            for(j=0; j<=1; j++){
                if(oCube){
                    var oBalconySdRails_ = Instantiate(oCube, Vector3(blcn_X, blcn_Y, blcn_Z), Quaternion.identity);
                    oBalconySdRails_.transform.parent = oBuilding7.transform; //parenting
                    oBalconySdRails_.name = "balcony side rails " + (++c);
                    oBalconySdRails_.transform.localScale.x = -0.003082343;
                    oBalconySdRails_.transform.localScale.y = 0.1696285;
                    oBalconySdRails_.transform.localScale.z = 0.007928168;
                    aCollectionObj_7.Add(oBalconySdRails_);
                }//end if
                blcn_X = this.RandX - 1.775764;
            }//end for
            blcn_X = this.RandX + 1.824236;
            blcn_Z -= 0.15;
        }//end for
```

```
c = 0;


//balcony main front rail
blcn_X = this.RandX + 0.02;
blcn_Y = nWind_Y + 1.88;
blcn_Z = this.RandZ + 4.23007;


for(i=0; i<=1; i++){
    if(oCube){
        var oBalconyMNRails_ = Instantiate(oCube, Vector3(blcn_X, blcn_Y, blcn_Z), Quaternion.identity);
        oBalconyMNRails_.transform.parent = oBuilding7.transform; //parenting
        oBalconyMNRails_.name = "balcony main rails " + (++c);
        oBalconyMNRails_.transform.localScale.x = -0.003082343;
        oBalconyMNRails_.transform.localScale.y = 0.5199003;
        oBalconyMNRails_.transform.localScale.z = 0.007928168;
        oBalconyMNRails_.transform.eulerAngles.z = 90;
        aCollectionObj_7.Add(oBalconyMNRails_);
    }//end if
    blcn_Y -= 0.55;
}//end for
c = 0;


//balcony main side rail
blcn_X = this.RandX + 1.824236;
blcn_Y = nWind_Y + 1.88;
blcn_Z = this.RandZ + 3.923868;
```

```
for(i=0; i<=1; i++){
    for(j=0; j<=1; j++){
        if(oCube){
            var oBalconyMNSdRails_ = Instantiate(oCube, Vector3(blcn_X, blcn_Y, blcn_Z), Quaternion.identity);
            oBalconyMNSdRails_.transform.parent = oBuilding7.transform; //parenting
            oBalconyMNSdRails_.name = "balcony main side rails " + (++c);
            oBalconyMNSdRails_.transform.localScale.x = -0.003082343;
            oBalconyMNSdRails_.transform.localScale.y = 0.1796265;
            oBalconyMNSdRails_.transform.localScale.z = 0.007928168;
            oBalconyMNSdRails_.transform.eulerAngles.x = 90;
            aCollectionObj_7.Add(oBalconyMNSdRails_);
        }//end if
        blcn_Y -= 0.55;
    }//end for
    blcn_Y = nWind_Y + 1.88;
    blcn_X = this.RandX - 1.775764;
}//end for
c = 0;
//::::::::::::::::::::::::::::::::::::::::::end facade::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

//::::::::::::::::::::::::::::::::::::::::::start left side windows::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
```

```
        var nWindPOSX_side:float = this.RandX + 3.5014659;
        var nWindPOSZ_side:float = nWind_Z - 1.3;
        nWind_Y= 5.0751646;
//first floor left side windows
        for(i=0; i<=1; i++){
                if(oWindow){
                        var oWindow_frstFlr_lft = Instantiate(oWindow, Vector3(nWindPOSX_side, nWind_Y, nWindPOSZ_side), Quaternion.identity);
                        oWindow_frstFlr_lft.transform.parent = oBuilding7.transform; //parenting
                        oWindow_frstFlr_lft.name = "first floor left side window " + (++c);
                        oWindow_frstFlr_lft.transform.localScale.x = -0.2409395;
                        oWindow_frstFlr_lft.transform.localScale.y = 0.2774929;
                        oWindow_frstFlr_lft.transform.localScale.z = 0.0163088;
                        oWindow_frstFlr_lft.transform.eulerAngles.y = 90;
                        aCollectionObj_7.Add(oWindow_frstFlr_lft);
                        nWindPOSZ_side -= 3.0;
                }//end if
        }//end for
        c = 0;

        nWindPOSX_side = this.RandX + 3.5014659;
        nWind_Z = this.RandZ + 3.5662495;
        nWindPOSZ_side = nWind_Z - 1.3;
//second floor left side windows
        for(i=0; i<=1; i++){
                if(oWindow){
```

```
                    var oWindow_scndFlr_lft = Instantiate(oWindow, Vector3(nWindPOSX_side, nWind_Y + 2.6442788, nWindPOSZ_side), Quaternion.identity);

                    oWindow_scndFlr_lft.transform.parent = oBuilding7.transform; //parenting

                    oWindow_scndFlr_lft.name = "second floor left side window " + (++c);

                    oWindow_scndFlr_lft.transform.localScale.x = -0.2517607;

                    oWindow_scndFlr_lft.transform.localScale.y = 0.4051766;

                    oWindow_scndFlr_lft.transform.localScale.z = 0.0163088;

                    oWindow_scndFlr_lft.transform.eulerAngles.y = 90;

                    aCollectionObj_7.Add(oWindow_scndFlr_lft);

                    nWindPOSZ_side -= 3.0;

                }//end if

        }//end for

        c = 0;


        nWindPOSX_side = this.RandX + 3.5014659;

        nWindPOSZ_side = this.RandZ + 2.2662495;

//third floor left side windows

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

                if(oWindow){

                    var oWindow_thrdFlr_lft = Instantiate(oWindow, Vector3(nWindPOSX_side, nWind_Y + 5, nWindPOSZ_side), Quaternion.identity);

                    oWindow_thrdFlr_lft.transform.parent = oBuilding7.transform; //parenting

                    oWindow_thrdFlr_lft.name = "third floor left side window " + (++c);

                    oWindow_thrdFlr_lft.transform.localScale.x = -0.2409395;

                    oWindow_thrdFlr_lft.transform.localScale.y = 0.2774929;
```

```
                oWindow_thrdFlr_lft.transform.localScale.z = 0.0163088;
                oWindow_thrdFlr_lft.transform.eulerAngles.y = 90;
                aCollectionObj_7.Add(oWindow_thrdFlr_lft);
                nWindPOSZ_side -= 3.0;
            }//end if
        }//end for
        c = 0;
//::::::::::::::::::::::::::::::::::::::::end left side windows::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::


//::::::::::::::::::::::::::::::::::::::::start right side windows::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
        nWindPOSX_side = this.RandX - 3.5014659;
        nWind_Z = this.RandZ + 3.5662495;
        nWindPOSZ_side = nWind_Z - 1.3;
//first floor right side windows
        for(i=0; i<=1; i++){
            if(oWindow){
                var oWindow_frstFlr_rght = Instantiate(oWindow, Vector3(nWindPOSX_side, nWind_Y, nWindPOSZ_side), Quaternion.identity);
                oWindow_frstFlr_rght.transform.parent = oBuilding7.transform; //parenting
                oWindow_frstFlr_rght.name = "first floor right side window " + (++c);
                oWindow_frstFlr_rght.transform.localScale.x = -0.2409395;
                oWindow_frstFlr_rght.transform.localScale.y = 0.2774929;
                oWindow_frstFlr_rght.transform.localScale.z = 0.0163088;
                oWindow_frstFlr_rght.transform.eulerAngles.y = 90;
                aCollectionObj_7.Add(oWindow_frstFlr_rght);
```

```
                nWindPOSZ_side -= 3.0;
            }//end if
        }//end for
        c = 0;

        nWindPOSX_side = this.RandX - 3.5014659;
        nWind_Z = this.RandZ + 3.5662495;
        nWindPOSZ_side = nWind_Z - 1.3;
//second floor right side windows
        for(i=0; i<=1; i++){
            if(oWindow){
                var oWindow_scndFlr_rght = Instantiate(oWindow, Vector3(nWindPOSX_side, nWind_Y + 2.6442788, nWindPOSZ_side), Quaternion.identity);
                oWindow_scndFlr_rght.transform.parent = oBuilding7.transform; //parenting
                oWindow_scndFlr_rght.name = "second floor right side window " + (++c);
                oWindow_scndFlr_rght.transform.localScale.x = -0.2517607;
                oWindow_scndFlr_rght.transform.localScale.y = 0.4051766;
                oWindow_scndFlr_rght.transform.localScale.z = 0.0163088;
                oWindow_scndFlr_rght.transform.eulerAngles.y = 90;
                aCollectionObj_7.Add(oWindow_scndFlr_rght);
                nWindPOSZ_side -= 3.0;
            }//end if
        }//end for
        c = 0;

        nWindPOSX_side = this.RandX - 3.5014659;
```

```
        nWind_Z = this.RandZ + 3.5662495;
        nWindPOSZ_side = nWind_Z - 1.3;
//third floor right side windows
        for(i=0; i<=1; i++){
                if(oWindow){
                        var oWindow_thrdFlr_rght = Instantiate(oWindow, Vector3(nWindPOSX_side, nWind_Y + 5, nWindPOSZ_side), Quaternion.identity);
                        oWindow_thrdFlr_rght.transform.parent = oBuilding7.transform; //parenting
                        oWindow_thrdFlr_rght.name = "third floor right side window " + (++c);
                        oWindow_thrdFlr_rght.transform.localScale.x = -0.2409395;
                        oWindow_thrdFlr_rght.transform.localScale.y = 0.2774929;
                        oWindow_thrdFlr_rght.transform.localScale.z = 0.0163088;
                        oWindow_thrdFlr_rght.transform.eulerAngles.y = 90;
                        aCollectionObj_7.Add(oWindow_thrdFlr_rght);
                        nWindPOSZ_side -= 3.0;
                }//end if
        }//end for
        c = 0;
//::::::::::::::::::::::::::::::::::::::::::end right side windows::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::


//:::::::::::::::::::::::::::::::::::::::::::::::::start ground floor windows::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
nWindPOSX_side = this.RandX + 2.40;
for(i=0; i<=1; i++){
        if(oWindow){
```

```
            var oWind_grnd = Instantiate(oWindow, Vector3(nWindPOSX_side, 1.423612, this.RandZ + 1.7776329), Quaternion.identity);

            oWind_grnd.transform.parent = oBuilding7.transform; //parenting

            oWind_grnd.name = "ground floor window " + (++c);

            oWind_grnd.transform.localScale.x = 0.1135587;

            oWind_grnd.transform.localScale.y = 0.4051766;

            oWind_grnd.transform.localScale.z = 0.0163088;

            aCollectionObj_7.Add(oWind_grnd);

            nWindPOSX_side -= 3.10;

        }//end if

}//end for

c = 0;

//::::::::::::::::::::::::::::::::::::::::::::::::::end ground floor windows::::::::::::::::::::::::::::::::::::::::::::::::::::::

}//end function Windows(RandX, RandZ)

//-----------------------------------------END  WINDOWS-----------------------------------------------

//-----------------------------------------START  COLUMNS-----------------------------------------------

public function Columns(RandX, RandZ){

        oColumn = GameObject.CreatePrimitive(PrimitiveType.Cube);

        oColumn.renderer.material.color = Color.white;


        oColumn.transform.position = Vector3(this.RandX -3.2, 1.302107, this.RandZ + 3.3772203);

        oColumn.transform.parent = oBuilding7.transform; //parenting

        oColumn.name = "Column 0";
```

```
oColumn.transform.localScale.x = 0.07017206;

oColumn.transform.localScale.y = 0.6454034;

oColumn.transform.localScale.z = 0.1285665;

aCollectionObj_7.Add(oColumn);


oColumnCube = GameObject.CreatePrimitive(PrimitiveType.Cube);


oColumnCube.transform.position = Vector3(this.RandX -3.2, 0.2373266, this.RandZ + 3.3772203);

oColumnCube.transform.parent = oColumn.transform; //parenting

oColumnCube.name = "Column base";

oColumnCube.transform.localScale.x = 1.43565;

oColumnCube.transform.localScale.y = 0.1184824;

oColumnCube.transform.localScale.z = 1.61189;

aCollectionObj_7.Add(oColumnCube);



var oColumnCube_Up = Instantiate(oColumnCube, Vector3(this.RandX - 3.2, 2.3029889, this.RandZ + 3.3772203), Quaternion.identity);

oColumnCube_Up.transform.parent = oColumn.transform; //parenting

oColumnCube_Up.name = "Column upper part";

oColumnCube_Up.transform.localScale.x = 1.43565;

oColumnCube_Up.transform.localScale.y = 0.1501914;

oColumnCube_Up.transform.localScale.z = 1.61189;

aCollectionObj_7.Add(oColumnCube_Up);


var nextPOSX_column:float = this.RandX - 1.62;

for(i=0; i<=3; i++){
```

```
            if(oColumn){
                var cloneColumn_ = Instantiate(oColumn, Vector3(nextPOSX_column, 1.302107, this.RandZ + 3.3772203), Quaternion.identity);
                cloneColumn_.transform.parent = oBuilding7.transform; //parenting
                cloneColumn_.name = "Column " + (++c);
                cloneColumn_.transform.localScale.x = 0.07017206;
                cloneColumn_.transform.localScale.y = 0.6454034;
                cloneColumn_.transform.localScale.z = 0.1285665;
                aCollectionObj_7.Add(cloneColumn_);
                nextPOSX_column += 1.58;
            }//end if
        }//end for
        c = 0;
}//end function Columns(RandX, RandZ)
//-----------------------------------------END COLUMNS-----------------------------------------------


//------------------------------START  DOOR-----------------------------------------------------------
public function Door(RandX, RandZ){

        oWindow = GameObject.CreatePrimitive(PrimitiveType.Cube);
        Destroy(oWindow);
        //oWindow.transform.renderer.material.color = Color.clear;
        oWindow.renderer.material.mainTexture = Resources.Load("door_corfu3"); //import texture for door
```

```
	var nPosX_door:float = this.RandX + 0.80;
	for(i=0; i<=1; i++){
		if(oWindow){
			var oDoor = Instantiate(oWindow, Vector3(nPosX_door, 1.123612, this.RandZ + 1.7776329), Quaternion.identity);
			oDoor.name = "door " + (++c);
			oDoor.transform.parent = oBuilding7.transform;
			oDoor.transform.localScale.x = 0.1341711;
			oDoor.transform.localScale.y = 0.5980287;
			oDoor.transform.localScale.z = 0.019;
			aCollectionObj_7.Add(oDoor);
			nPosX_door -= 3.20;
		}//end if
	}//end for
	c = 0;
}//end function Door(RandX, RandZ)
//------------------------------END DOOR-----------------------------------------------------------
----
//-------------------------------------START  ROOF-------------------------------------------------
-----
public function Roof(RandX, RandZ){
	var tmpobj: GameObject = new GameObject();
	var mf: MeshFilter = tmpobj.AddComponent(MeshFilter);
```

```
//      tmpobj.AddComponent(MeshFilter);
        tmpobj.AddComponent(MeshRenderer);
        tmpobj.name = "Roof";
        tmpobj.transform.parent = oBuilding7.transform;

        var verts: Vector3[] = new Vector3[6];
        var normals: Vector3[] = new Vector3[6];
        var uv: Vector2[] = new Vector2[6];
        var tri: int[] = new int[18]; //3 vertices * 6 triangles =18


        //vertices positioning
        verts[0] = new Vector3(this.RandX - 3.6, 12.2, this.RandZ + 1); //P1
        verts[1] = new Vector3(this.RandX - 3.6, 11.2, this.RandZ - 2); //P2
        verts[2] = new Vector3(this.RandX - 3.6, 11.2, this.RandZ + 4); //P3

        verts[3] = new Vector3(this.RandX + 3.6, 12.2, this.RandZ + 1); //P4
        verts[4] = new Vector3(this.RandX + 3.6, 11.2, this.RandZ + 4); //P5
        verts[5] = new Vector3(this.RandX + 3.6, 11.2, this.RandZ - 2); //P6


        uv[0] = new Vector2(0, 0.5);
        uv[1] = new Vector2(1, 1);
        uv[2] = new Vector2(1, 0);

        uv[3] = new Vector2(1, 0.5);
        uv[4] = new Vector2(0, 0);
```

```
uv[5] = new Vector2(0, 1);



//triangles
tri[0] = 0; //refer to vertices: verts[0]
tri[1] = 1; //refer to vertices: verts[1]
tri[2] = 2; //refer to vertices: verts[2]

tri[3] = 0;
tri[4] = 2;
tri[5] = 4;

tri[6] = 3;
tri[7] = 0;
tri[8] = 4;

tri[9] = 3;
tri[10] = 4;
tri[11] = 5;

tri[12] = 3;
tri[13] = 5;
tri[14] = 0;

tri[15] = 0;
tri[16] = 5;
tri[17] = 1;
```

```
        var mesh: Mesh = new Mesh();
        mesh.name = "Roof";
        mesh.vertices = verts;
        mesh.triangles = tri;
        mesh.uv = uv;
        mesh.normals = normals;
        mesh.RecalculateNormals();
        mf.mesh = mesh;
        tmpobj.renderer.material.mainTexture = Resources.Load("tiled_roof_red_grey");
}//end function Roof(RandX, RandZ)
//--------------------------------------END ROOF-----------------------------------------------------

//-----------------------------------START ARCH-----------------------------------------------------
private function Arch(RandX, RandZ, tmpobj) {
        tmpobj = new GameObject();
        var mf: MeshFilter = tmpobj.AddComponent(MeshFilter);

        tmpobj.AddComponent(MeshRenderer);
        tmpobj.name = "Arch";
        tmpobj.renderer.material.color = Color.white;
        tmpobj.transform.parent = oBuilding7.transform;

        var verts: Vector3[] = new Vector3[14];
```

```
var uv: Vector2[] = new Vector2[14];

var normals: Vector3[] = new Vector3[14];

var tri: int[] = new int[72]; //3 vertices * 12 triangles = 36 * 2 (duplicate triangles for the back face) =72


//vertices positioning

verts[0] = new Vector3(this.RandX - 1.9, 2.5, this.RandZ + 3.777); //P1

verts[1] = new Vector3(this.RandX - 1.9, 2, this.RandZ + 3.777); //P2

verts[2] = new Vector3(this.RandX - 1.35, 2, this.RandZ + 3.777); //P3

verts[3] = new Vector3(this.RandX - 1.35, 3.5, this.RandZ + 3.777); //P4

verts[4] = new Vector3(this.RandX - 3.5, 3.5, this.RandZ + 3.777); //P5

verts[5] = new Vector3(this.RandX - 3.5, 2, this.RandZ + 3.777); //P6

verts[6] = new Vector3(this.RandX - 2.95, 2, this.RandZ + 3.777); //P7

verts[7] = new Vector3(this.RandX - 2.95, 2.5, this.RandZ + 3.777); //P8

verts[8] = new Vector3(this.RandX - 2.9, 2.8, this.RandZ + 3.777); //P9

verts[9] = new Vector3(this.RandX - 2.75, 3, this.RandZ + 3.777); //P10

verts[10] = new Vector3(this.RandX - 2.55, 3.1, this.RandZ + 3.777); //P11

verts[11] = new Vector3(this.RandX - 2.3,  3.1, this.RandZ + 3.777); //P12

verts[12] = new Vector3(this.RandX - 2.1, 3, this.RandZ + 3.777); //P13

verts[13] = new Vector3(this.RandX - 1.95, 2.8, this.RandZ + 3.777); //P14


//normals

normals[0] = new Vector3(0, 0, 1);

normals[1] = new Vector3(0, 0, 1);

normals[2] = new Vector3(0, 0, 1);

normals[3] = new Vector3(0, 0, 1);

normals[4] = new Vector3(0, 0, 1);
```

```
normals[5] = new Vector3(0, 0, 1);
normals[6] = new Vector3(0, 0, 1);
normals[7] = new Vector3(0, 0, 1);
normals[8] = new Vector3(0, 0, 1);
normals[9] = new Vector3(0, 0, 1);
normals[10] = new Vector3(0, 0, 1);
normals[11] = new Vector3(0, 0, 1);
normals[12] = new Vector3(0, 0, 1);
normals[13] = new Vector3(0, 0, 1);

//uv mapping (textures)
uv[0] = new Vector2(0, 0.3);
uv[1] = new Vector2(0.2, 0);
uv[2] = new Vector2(0, 0);
uv[3] = new Vector2(0, 1);
uv[4] = new Vector2(1, 1);
uv[5] = new Vector2(1, 0);
uv[6] = new Vector2(0.8, 0);
uv[7] = new Vector2(0.8, 0.3);
uv[8] = new Vector2(0.75, 0.5);
uv[9] = new Vector2(0.65, 0.7);
uv[10] = new Vector2(0.6, 0.9);
uv[11] = new Vector2(0.4, 0.9);
uv[12] = new Vector2(0.45, 0.7);
uv[13] = new Vector2(0.35, 0.5);
```

```
//triangles
tri[0] = 0; //refer to vertices: verts[0]
tri[1] = 1; //refer to vertices: verts[1]
tri[2] = 2; //refer to vertices: verts[2]

tri[3] = 0;
tri[4] = 2;
tri[5] = 3;

tri[6] = 0;
tri[7] = 3;
tri[8] = 13;

tri[9] = 13;
tri[10] = 3;
tri[11] = 12;

tri[12] = 12;
tri[13] = 3;
tri[14] = 11;

tri[15] = 11;
tri[16] = 3;
tri[17] = 4;

tri[18] = 11;
tri[19] = 4;
```

```
tri[20] = 10;

tri[21] = 10;
tri[22] = 4;
tri[23] = 9;

tri[24] = 9;
tri[25] = 4;
tri[26] = 8;

tri[27] = 8;
tri[28] = 4;
tri[29] = 7;

tri[30] = 7;
tri[31] = 4;
tri[32] = 5;

tri[33] = 7;
tri[34] = 5;
tri[35] = 6;

tri[36] = 0;
tri[37] = 2;
tri[38] = 1;

tri[39] = 0;
```

```
tri[40] = 3;
tri[41] = 2;

tri[42] = 13;
tri[43] = 3;
tri[44] = 0;

tri[45] = 12;
tri[46] = 3;
tri[47] = 13;

tri[48] = 11;
tri[49] = 3;
tri[50] = 12;

tri[51] = 10;
tri[52] = 3;
tri[53] = 11;

tri[54] = 10;
tri[55] = 4;
tri[56] = 3;

tri[57] = 9;
tri[58] = 4;
tri[59] = 10;
```

```
        tri[60] = 8;
        tri[61] = 4;
        tri[62] = 9;

        tri[63] = 7;
        tri[64] = 4;
        tri[65] = 8;

        tri[66] = 7;
        tri[67] = 5;
        tri[68] = 4;

        tri[69] = 7;
        tri[70] = 6;
        tri[71] = 5;

        var mesh: Mesh = new Mesh();
        mesh.name = "Arch";
        mesh.vertices = verts;
        mesh.triangles = tri;
        mesh.uv = uv;
        mesh.normals = normals;
        //mesh.RecalculateNormals();
        tmpobj.renderer.material.mainTexture                    =
Resources.Load("white_brick_wall");
        mf.mesh = mesh;
}//end function Arch(Rand_X, Rand_Z)
```

```
private function ArchVertical(RandX, RandZ, tmpobj){

        tmpobj = new GameObject();
        var mf: MeshFilter = tmpobj.AddComponent(MeshFilter);

        tmpobj.AddComponent(MeshRenderer);
        tmpobj.name = "Arch vert";
        tmpobj.renderer.material.color = Color.white;
        tmpobj.transform.parent = oBuilding7.transform;

        var verts: Vector3[] = new Vector3[14];
        var uv: Vector2[] = new Vector2[14];
        var normals: Vector3[] = new Vector3[14];
        var tri: int[] = new int[72]; //3 vertices * 12 triangles = 36 * 2 (duplicate triangles for the back face) =72

        //vertices positioning
        verts[0] = new Vector3(this.RandX , 2.5, this.RandZ - 1.9); //P1
        verts[1] = new Vector3(this.RandX , 2, this.RandZ - 1.9); //P2
        verts[2] = new Vector3(this.RandX , 2, this.RandZ - 1.35); //P3
        verts[3] = new Vector3(this.RandX , 3.5, this.RandZ - 1.35); //P4
        verts[4] = new Vector3(this.RandX , 3.5, this.RandZ - 3.5); //P5
        verts[5] = new Vector3(this.RandX , 2, this.RandZ - 3.5); //P6
        verts[6] = new Vector3(this.RandX , 2, this.RandZ - 2.95); //P7
        verts[7] = new Vector3(this.RandX , 2.5, this.RandZ - 2.95); //P8
```

```
verts[8] = new Vector3(this.RandX , 2.8, this.RandZ - 2.9); //P9
verts[9] = new Vector3(this.RandX , 3, this.RandZ - 2.75); //P10
verts[10] = new Vector3(this.RandX , 3.1, this.RandZ - 2.55); //P11
verts[11] = new Vector3(this.RandX ,  3.1, this.RandZ - 2.3); //P12
verts[12] = new Vector3(this.RandX , 3, this.RandZ - 2.1); //P13
verts[13] = new Vector3(this.RandX , 2.8, this.RandZ - 1.95); //P14

//normals
normals[0] = new Vector3(0, 0, 1);
normals[1] = new Vector3(0, 0, 1);
normals[2] = new Vector3(0, 0, 1);
normals[3] = new Vector3(0, 0, 1);
normals[4] = new Vector3(0, 0, 1);
normals[5] = new Vector3(0, 0, 1);
normals[6] = new Vector3(0, 0, 1);
normals[7] = new Vector3(0, 0, 1);
normals[8] = new Vector3(0, 0, 1);
normals[9] = new Vector3(0, 0, 1);
normals[10] = new Vector3(0, 0, 1);
normals[11] = new Vector3(0, 0, 1);
normals[12] = new Vector3(0, 0, 1);
normals[13] = new Vector3(0, 0, 1);

//uv mapping (textures)
uv[0] = new Vector2(0, 0.3);
uv[1] = new Vector2(0.2, 0);
uv[2] = new Vector2(0, 0);
```

```
uv[3] = new Vector2(0, 1);
uv[4] = new Vector2(1, 1);
uv[5] = new Vector2(1, 0);
uv[6] = new Vector2(0.8, 0);
uv[7] = new Vector2(0.8, 0.3);
uv[8] = new Vector2(0.75, 0.5);
uv[9] = new Vector2(0.65, 0.7);
uv[10] = new Vector2(0.6, 0.9);
uv[11] = new Vector2(0.4, 0.9);
uv[12] = new Vector2(0.45, 0.7);
uv[13] = new Vector2(0.35, 0.5);

//triangles
tri[0] = 0; //refer to vertices: verts[0]
tri[1] = 1; //refer to vertices: verts[1]
tri[2] = 2; //refer to vertices: verts[2]

tri[3] = 0;
tri[4] = 2;
tri[5] = 3;

tri[6] = 0;
tri[7] = 3;
tri[8] = 13;

tri[9] = 13;
tri[10] = 3;
```

```
tri[11] = 12;


tri[12] = 12;
tri[13] = 3;
tri[14] = 11;


tri[15] = 11;
tri[16] = 3;
tri[17] = 4;


tri[18] = 11;
tri[19] = 4;
tri[20] = 10;


tri[21] = 10;
tri[22] = 4;
tri[23] = 9;


tri[24] = 9;
tri[25] = 4;
tri[26] = 8;


tri[27] = 8;
tri[28] = 4;
tri[29] = 7;


tri[30] = 7;
```

```
tri[31] = 4;
tri[32] = 5;

tri[33] = 7;
tri[34] = 5;
tri[35] = 6;

tri[36] = 0;
tri[37] = 2;
tri[38] = 1;

tri[39] = 0;
tri[40] = 3;
tri[41] = 2;

tri[42] = 13;
tri[43] = 3;
tri[44] = 0;

tri[45] = 12;
tri[46] = 3;
tri[47] = 13;

tri[48] = 11;
tri[49] = 3;
tri[50] = 12;
```

```
tri[51] = 10;
tri[52] = 3;
tri[53] = 11;

tri[54] = 10;
tri[55] = 4;
tri[56] = 3;

tri[57] = 9;
tri[58] = 4;
tri[59] = 10;

tri[60] = 8;
tri[61] = 4;
tri[62] = 9;

tri[63] = 7;
tri[64] = 4;
tri[65] = 8;

tri[66] = 7;
tri[67] = 5;
tri[68] = 4;

tri[69] = 7;
tri[70] = 6;
tri[71] = 5;
```

```
        var mesh: Mesh = new Mesh();
        mesh.name = "Arch vert";
        mesh.vertices = verts;
        mesh.triangles = tri;
        mesh.uv = uv;
        mesh.normals = normals;
        //mesh.RecalculateNormals();
        tmpobj.renderer.material.mainTexture =
Resources.Load("white_brick_wall");
        mf.mesh = mesh;

}//end function ArchVertical(Rand_X, Rand_Z, tmpobj)

public function DoArch(RandX, RandZ){
        tmpobj = new GameObject();
        Destroy(tmpobj);

        var nPOSZ:float = this.RandZ + 0.777;

        for(i=0; i<=3; i++){
                if(tmpobj){
                        Arch(this.RandX, nPOSZ, tmpobj);
                        this.RandX += 1.6;
                }//end if
        }//end for
```

```
        this.RandX -= 9.96;
        this.RandZ +=5.15;

        for(i=0; i<=1; i++){
                if(tmpobj){
                        ArchVertical(this.RandX, this.RandZ, tmpobj);
                        this.RandX += 7.1;
                }//end if
        }//end for
}//end function DoArch(Rand_X, Rand_Z)
//---------------------------------END ARCH-----------------------------------------------------------
```

//House_8.js

```
//create objects
var oBuilding8 : GameObject;
//var oColumn = GameObject;//.CreatePrimitive(PrimitiveType.Cube);
var oWindow : GameObject;
var oCube : GameObject;
var tmpobj: GameObject;

//-------------------start public variables-------------------------------------------------------
public var aCollectionObj_8 = new Array(); // an Array - a collection- for the objects of House_4

public var i: int = 0;
public var j: int = 0;
public var c: int = 0;
```

```
//Random ranges between X, Y coordinates
public var RandX:float;
public var RandZ:float;
//-------------------end public variables-------------------------------------------------------


//MAIN function
public function mainDO() {


        MainBuilding(RandX,RandZ);
        FirstFloor(RandX,RandZ);
        SecondFloor(RandX, RandZ);
        ThirdFloor(RandX,RandZ);
        Eaves(RandX, RandZ);
        Windows(RandX, RandZ);
        Columns(RandX, RandZ);
        Door(RandX, RandZ);
        Roof(RandX, RandZ);
        DoArch(RandX, RandZ);
}//end Start ()


public function Destroy_(){

DestroyImmediate(oBuilding8);
```

```
}

//----------Main Building-ground floor-------------------------

public function MainBuilding(RandX,RandZ){

        oBuilding8 = GameObject.CreatePrimitive(PrimitiveType.Cube);

        oBuilding8.renderer.material.color = Color(Random.Range(1,0.9), Random.Range(0.980,0.880), Random.Range(0.804,0.704), 0);

        oBuilding8.transform.position = Vector3(this.RandX, 1.722107, this.RandZ);

        oBuilding8.name = "House_8";

        oBuilding8.transform.localScale.x = 7;

        oBuilding8.transform.localScale.y = 3.308559;

        oBuilding8.transform.localScale.z = 3.5;

        aCollectionObj_8.Add(oBuilding8);

}//end function MainBuilding(Rand_X,Rand_Z)



//---------------------------------------START FIRST FLOOR-----------------------------------------

public function FirstFloor(RandX,RandZ){

        oCube = GameObject.CreatePrimitive(PrimitiveType.Cube);

        Destroy (oCube);

        oCube.renderer.material.color = Color(1, 0.980, 0.804, 0);

        //First floor

        var oFrst_Flr = Instantiate(oCube, Vector3(this.RandX, 4.527526, this.RandZ + 0.9), Quaternion.identity);

        oFrst_Flr.transform.parent = oBuilding8.transform; //parenting

        oFrst_Flr.name = "First Floor";
```

```
        oFrst_Flr.transform.localScale.x = 0.9954391;
        oFrst_Flr.transform.localScale.y = 0.7123187;
        oFrst_Flr.transform.localScale.z = 1.536353;
        aCollectionObj_8.Add(oFrst_Flr);
}//end function FirstFloor(RandX,RandZ)
//----------------------------------------END FIRST FLOOR----------------------------------------


//----------------------------------------START SECOND FLOOR----------------------------------------
public function SecondFloor(RandX, RandZ){
        oCube = GameObject.CreatePrimitive(PrimitiveType.Cube);
        Destroy (oCube);
        oCube.renderer.material.color = Color(1, 0.980, 0.804, 0);
        //Second Floor
        var oScnd_Floor = Instantiate(oCube, Vector3(this.RandX, 7.2189827, this.RandZ + 0.9), Quaternion.identity);
        oScnd_Floor.transform.parent = oBuilding8.transform; //parenting
        oScnd_Floor.name = "Second Floor";
        oScnd_Floor.transform.localScale.x = 0.9954391;
        oScnd_Floor.transform.localScale.y = 0.9377826;
        oScnd_Floor.transform.localScale.z = 1.536353;
        aCollectionObj_8.Add(oScnd_Floor);
}//end function SecondFloor(RandX,RandZ)
//----------------------------------------END SECOND FLOOR----------------------------------------
```

```
//----------------------------------------START THIRD FLOOR---------------------------------
-------------------------

public function ThirdFloor(RandX,RandZ){

	oCube = GameObject.CreatePrimitive(PrimitiveType.Cube);

	Destroy (oCube);

	//Third floor

	oCube.renderer.material.color = Color(1, 0.980, 0.804, 0);

	var oThrd_Flr = Instantiate(oCube, Vector3(this.RandX, 9.927526,
this.RandZ + 0.9), Quaternion.identity);

	oThrd_Flr.transform.parent = oBuilding8.transform; //parenting

	oThrd_Flr.name = "Third Floor";

	oThrd_Flr.transform.localScale.x = 0.9954391;

	oThrd_Flr.transform.localScale.y = 0.7123187;

	oThrd_Flr.transform.localScale.z = 1.536353;

	aCollectionObj_8.Add(oThrd_Flr);

}//end function FirstFloor(RandX,RandZ)

//----------------------------------------END THIRD FLOOR-----------------------------------
----------------------


//----------------------------------------START EAVES---------------------------------------
---------------

public function Eaves(RandX, RandZ){

	oCube = GameObject.CreatePrimitive(PrimitiveType.Cube);

	Destroy (oCube);

	oCube.renderer.material.color = Color.white;

	var nPOSY_eaves:float = 11.027526;

	for(i=0; i<=2; i++){
```

```
            if(oCube){
                var oEaves_flr = Instantiate(oCube, Vector3(this.RandX, nPOSY_eaves, this.RandZ + 0.9), Quaternion.identity);
                oEaves_flr.transform.parent = oBuilding8.transform; //parenting
                oEaves_flr.name = "eaves " + (++c);
                oEaves_flr.transform.localScale.x = 1.020326;
                oEaves_flr.transform.localScale.z = 1.593966;
                aCollectionObj_8.Add(oEaves_flr);
                if(i == 1){
                    oEaves_flr.transform.localScale.x = 1.045835;
                    oEaves_flr.transform.localScale.z = 1.633815;
                }//end if
                else if(i == 2){
                    oEaves_flr.transform.localScale.x = 1.057959;
                    oEaves_flr.transform.localScale.z = 1.65374;
                }//end else if
                oEaves_flr.transform.localScale.y = 0.02458715;
                aCollectionObj_8.Add(oEaves_flr);
            }//end if
            nPOSY_eaves += 0.07;
        }//end for
        c = 0;

        var eaves_X:float = this.RandX + 2.82102;
        var eaves_Y:float = 8.4;
        var eaves_Z:float = this.RandZ + 3.5662495;
        //window eaves facade
```

```
for(i=0; i<=3; i++){
    for(j=0; j<=1; j++){
        for(k=0; k<=1; k++){
            if(oCube){
                var oWindowEaves_ = Instantiate(oCube, Vector3(eaves_X, eaves_Y, eaves_Z), Quaternion.identity);
                oWindowEaves_.transform.parent = oBuilding8.transform; //parenting
                oWindowEaves_.name = "window front side eaves" + (++c);
                if( k == 1 ){
                    oWindowEaves_.transform.localScale.x = 0.1356827;
                    oWindowEaves_.transform.localScale.z = 0.05289786;
                }//end if
                else{
                    oWindowEaves_.transform.localScale.x = 0.1096849;
                    oWindowEaves_.transform.localScale.z = 0.03796064;
                }//else
                oWindowEaves_.transform.localScale.y = 0.01727708;
                aCollectionObj_8.Add(oWindowEaves_);
            }//end if
            eaves_Y += 0.05;
        }//end for
        eaves_Y = 8.4;
        eaves_Y -= 3;
```

```
            }//end for
            eaves_Y = 8.4;
            eaves_X -= 1.9;
        }//end for
        c = 0;

        //window eaves left side
        eaves_X = this.RandX + 3.4414659;
        eaves_Y = 8.4;
        for(i=0; i<=1; i++){
            for(j=0; j<=1; j++){
                for(k=0; k<=1; k++){
                    if(oCube){
                        var oWindowEaves_lft = Instantiate(oCube, Vector3(eaves_X, eaves_Y, eaves_Z - 1.3), Quaternion.identity);
                        oWindowEaves_lft.transform.parent = oBuilding8.transform; //parenting
                        oWindowEaves_lft.name = "left side window eaves" + (++c);
                        if( k == 1 ){
                            oWindowEaves_lft.transform.localScale.x = 0.305287;
                            oWindowEaves_lft.transform.localScale.z = 0.05289786;
                        }//end if
                        else{
                            oWindowEaves_lft.transform.localScale.x = 0.2618665;
                            oWindowEaves_lft.transform.localScale.z = 0.03796064;
```

```
                                }//else
                                oWindowEaves_lft.transform.localScale.y = 0.01727708;
                                oWindowEaves_lft.transform.eulerAngles.y = 90;
                                aCollectionObj_8.Add(oWindowEaves_lft);
                            }//end if
                            eaves_Y += 0.05;
                        }//end for
                        eaves_Y = 8.4;
                        eaves_Y -= 3;
                    }//end for
                    eaves_Y = 8.4;
                    eaves_Z -= 3.0;
                }//end for
                c = 0;

                //window eaves right side
                eaves_X = this.RandX - 3.4414659;
                eaves_Y = 8.4;
                eaves_Z = this.RandZ + 3.5662495;
                for(i=0; i<=1; i++){
                    for(j=0; j<=1; j++){
                        for(k=0; k<=1; k++){
                            if(oCube){
                                var oWindowEaves_rght = Instantiate(oCube, Vector3(eaves_X, eaves_Y, eaves_Z - 1.3), Quaternion.identity);
                                oWindowEaves_rght.transform.parent = oBuilding8.transform; //parenting
```

```
                                    oWindowEaves_rght.name = "right side window
eaves" + (++c);
                                    if( k == 1 ){

        oWindowEaves_rght.transform.localScale.x = 0.305287;

        oWindowEaves_rght.transform.localScale.z = 0.05289786;
                                    }//end if
                                    else{

        oWindowEaves_rght.transform.localScale.x = 0.2618665;

        oWindowEaves_rght.transform.localScale.z = 0.03796064;
                                    }//else
                                    oWindowEaves_rght.transform.localScale.y    =
0.01727708;
                                    oWindowEaves_rght.transform.eulerAngles.y   =
90;
                                    aCollectionObj_8.Add(oWindowEaves_rght);
                                }//end if
                                eaves_Y += 0.05;
                            }//end for
                            eaves_Y = 8.4;
                            eaves_Y -= 3;
                        }//end for
                        eaves_Y = 8.4;
                        eaves_Z -= 3.0;
        }//end for
        c = 0;
}//end function Eaves(RandX, RandZ)
```

```
//---------------------------------------END EAVES-----------------------------------------------
-----------




//-----------------------------------START WINDOWS-----------------------------------------------
---------------

public function Windows(RandX, RandZ){

	oWindow = GameObject.CreatePrimitive(PrimitiveType.Cube);

	//solving an issue appearing: destroy visible instances appear as common
cubes on the scene

	Destroy(oWindow);



	//transparent windows

	//oWindow.transform.renderer.material.color = Color.clear;

	oWindow.renderer.material.mainTexture                                    =
Resources.Load("window_corfu2"); //import texture for windows



	var nWind_X:float = this.RandX + 2.82102;

	var nWind_Y:float = 5.0751646;

	var nWind_Z:float = this.RandZ + 3.6062495;

//::::::::::::::::::::::::::::::::::::::::::start
facade::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

//first floor windows

	for(i=0; i<=3; i++){

		if(oWindow){

			var oWindow_frstFlr = Instantiate(oWindow,
Vector3(nWind_X, nWind_Y - 0.5, nWind_Z), Quaternion.identity);

			oWindow_frstFlr.transform.parent = oBuilding8.transform;
//parenting
```

```
                oWindow_frstFlr.name = "first floor window " + (++c);
                oWindow_frstFlr.transform.localScale.x = 0.1135587;
                oWindow_frstFlr.transform.localScale.y = 0.513514;
                oWindow_frstFlr.transform.localScale.z = 0.0163088;
                aCollectionObj_8.Add(oWindow_frstFlr);
                nWind_X -= 1.9;
            }//end if
        }//end for
        c = 0;

        nWind_X = this.RandX + 2.82102;

//second floor windows
        for(i=0; i<=3; i++){
            if(oWindow){
                var oWindow_scndFlr = Instantiate(oWindow,
Vector3(nWind_X, nWind_Y + 2.6442788, nWind_Z), Quaternion.identity);
                oWindow_scndFlr.transform.parent = oBuilding8.transform;
//parenting
                oWindow_scndFlr.name = "second floor window " + (++c);
                oWindow_scndFlr.transform.localScale.x = 0.1096849;
                oWindow_scndFlr.transform.localScale.y = 0.4051766;
                oWindow_scndFlr.transform.localScale.z = 0.0163088;
                aCollectionObj_8.Add(oWindow_scndFlr);
                nWind_X -= 1.9;
            }//end if
        }//end for
        c = 0;
```

```
	//third floor windows
	nWind_X = this.RandX + 2.82102;
	nWind_Z= this.RandZ + 3.6062495;

	for(i=0; i<=3; i++){
		if(oWindow){
			var oWindow_thrdFlr = Instantiate(oWindow, Vector3(nWind_X, nWind_Y + 5, nWind_Z), Quaternion.identity);
			oWindow_thrdFlr.transform.parent = oBuilding8.transform; //parenting
			oWindow_thrdFlr.name = "third floor window " + (++c);
			oWindow_thrdFlr.transform.localScale.x = 0.1135587;
			oWindow_thrdFlr.transform.localScale.y = 0.2774929;
			oWindow_thrdFlr.transform.localScale.z = 0.0163088;
			aCollectionObj_8.Add(oWindow_thrdFlr);
			nWind_X -= 1.9;
		}//end if
	}//end for
	c = 0;
//::::::::::::::::::::::::::::::::::::::::end facade:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

//::::::::::::::::::::::::::::::::::::::::start left side windows:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
	var nWindPOSX_side:float = this.RandX + 3.5014659;
	var nWindPOSZ_side:float = nWind_Z - 1.3;
//first floor left side windows
	for(i=0; i<=1; i++){
```

```
            if(oWindow){
                var oWindow_frstFlr_lft = Instantiate(oWindow, Vector3(nWindPOSX_side, nWind_Y - 0.5, nWindPOSZ_side), Quaternion.identity);
                oWindow_frstFlr_lft.transform.parent = oBuilding8.transform; //parenting
                oWindow_frstFlr_lft.name = "first floor left side window " + (++c);
                oWindow_frstFlr_lft.transform.localScale.x = -0.2409395;
                oWindow_frstFlr_lft.transform.localScale.y = 0.513514;
                oWindow_frstFlr_lft.transform.localScale.z = 0.0163088;
                oWindow_frstFlr_lft.transform.eulerAngles.y = 90;
                aCollectionObj_8.Add(oWindow_frstFlr_lft);
                nWindPOSZ_side -= 3.0;
            }//end if
        }//end for
        c = 0;

        nWindPOSX_side = this.RandX + 3.5014659;
        nWind_Z = this.RandZ + 3.5662495;
        nWindPOSZ_side = nWind_Z - 1.3;
//second floor left side windows
        for(i=0; i<=1; i++){
            if(oWindow){
                var oWindow_scndFlr_lft = Instantiate(oWindow, Vector3(nWindPOSX_side, nWind_Y + 2.6442788, nWindPOSZ_side), Quaternion.identity);
                oWindow_scndFlr_lft.transform.parent = oBuilding8.transform; //parenting
```

```
                oWindow_scndFlr_lft.name = "second floor left side window " + (++c);
                oWindow_scndFlr_lft.transform.localScale.x = -0.2517607;
                oWindow_scndFlr_lft.transform.localScale.y = 0.4051766;
                oWindow_scndFlr_lft.transform.localScale.z = 0.0163088;
                oWindow_scndFlr_lft.transform.eulerAngles.y = 90;
                aCollectionObj_8.Add(oWindow_scndFlr_lft);
                nWindPOSZ_side -= 3.0;
            }//end if
        }//end for
        c = 0;

        nWindPOSX_side = this.RandX + 3.5014659;
        nWindPOSZ_side = this.RandZ + 2.2662495;
//third floor left side windows
        for(i=0; i<=1; i++){
            if(oWindow){
                var oWindow_thrdFlr_lft = Instantiate(oWindow, Vector3(nWindPOSX_side, nWind_Y + 5, nWindPOSZ_side), Quaternion.identity);
                oWindow_thrdFlr_lft.transform.parent = oBuilding8.transform; //parenting
                oWindow_thrdFlr_lft.name = "third floor left side window " + (++c);
                oWindow_thrdFlr_lft.transform.localScale.x = -0.2409395;
                oWindow_thrdFlr_lft.transform.localScale.y = 0.2774929;
                oWindow_thrdFlr_lft.transform.localScale.z = 0.0163088;
                oWindow_thrdFlr_lft.transform.eulerAngles.y = 90;
                aCollectionObj_8.Add(oWindow_thrdFlr_lft);
                nWindPOSZ_side -= 3.0;
```

```
        }//end if
    }//end for
    c = 0;
//:::::::::::::::::::::::::::::::::::::::end left side windows:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

//:::::::::::::::::::::::::::::::::::::::start right side windows:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
    nWindPOSX_side = this.RandX - 3.5014659;
    nWind_Z = this.RandZ + 3.5662495;
    nWindPOSZ_side = nWind_Z - 1.3;
//first floor right side windows
    for(i=0; i<=1; i++){
        if(oWindow){
            var oWindow_frstFlr_rght = Instantiate(oWindow, Vector3(nWindPOSX_side, nWind_Y - 0.5, nWindPOSZ_side), Quaternion.identity);
            oWindow_frstFlr_rght.transform.parent = oBuilding8.transform; //parenting
            oWindow_frstFlr_rght.name = "first floor right side window " + (++c);
            oWindow_frstFlr_rght.transform.localScale.x = -0.2409395;
            oWindow_frstFlr_rght.transform.localScale.y = 0.513514;
            oWindow_frstFlr_rght.transform.localScale.z = 0.0163088;
            oWindow_frstFlr_rght.transform.eulerAngles.y = 90;
            aCollectionObj_8.Add(oWindow_frstFlr_rght);
            nWindPOSZ_side -= 3.0;
        }//end if
    }//end for
```

```
        c = 0;


        nWindPOSX_side = this.RandX - 3.5014659;
        nWind_Z = this.RandZ + 3.5662495;
        nWindPOSZ_side = nWind_Z - 1.3;
//second floor right side windows
        for(i=0; i<=1; i++){
                if(oWindow){
                        var oWindow_scndFlr_rght = Instantiate(oWindow, Vector3(nWindPOSX_side, nWind_Y + 2.6442788, nWindPOSZ_side), Quaternion.identity);
                        oWindow_scndFlr_rght.transform.parent = oBuilding8.transform; //parenting
                        oWindow_scndFlr_rght.name = "second floor right side window " + (++c);
                        oWindow_scndFlr_rght.transform.localScale.x = -0.2517607;
                        oWindow_scndFlr_rght.transform.localScale.y = 0.4051766;
                        oWindow_scndFlr_rght.transform.localScale.z = 0.0163088;
                        oWindow_scndFlr_rght.transform.eulerAngles.y = 90;
                        aCollectionObj_8.Add(oWindow_scndFlr_rght);
                        nWindPOSZ_side -= 3.0;
                }//end if
        }//end for
        c = 0;


        nWindPOSX_side = this.RandX - 3.5014659;
        nWind_Z = this.RandZ + 3.5662495;
        nWindPOSZ_side = nWind_Z - 1.3;
//third floor right side windows
```

```
        for(i=0; i<=1; i++){

                if(oWindow){

                        var oWindow_thrdFlr_rght = Instantiate(oWindow, Vector3(nWindPOSX_side, nWind_Y + 5, nWindPOSZ_side), Quaternion.identity);

                        oWindow_thrdFlr_rght.transform.parent = oBuilding8.transform; //parenting

                        oWindow_thrdFlr_rght.name = "third floor right side window " + (++c);

                        oWindow_thrdFlr_rght.transform.localScale.x = -0.2409395;

                        oWindow_thrdFlr_rght.transform.localScale.y = 0.2774929;

                        oWindow_thrdFlr_rght.transform.localScale.z = 0.0163088;

                        oWindow_thrdFlr_rght.transform.eulerAngles.y = 90;

                        aCollectionObj_8.Add(oWindow_thrdFlr_rght);

                        nWindPOSZ_side -= 3.0;

                }//end if

        }//end for

        c = 0;

//::::::::::::::::::::::::::::::::::::::::::end right side windows::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::


//:::::::::::::::::::::::::::::::::::::::::::::::::start ground floor windows::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

nWindPOSX_side = this.RandX + 2.40;

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

        if(oWindow){

                var oWind_grnd = Instantiate(oWindow, Vector3(nWindPOSX_side, 1.423612, this.RandZ + 1.7776329), Quaternion.identity);

                oWind_grnd.transform.parent = oBuilding8.transform; //parenting

                oWind_grnd.name = "ground floor window " + (++c);
```

```
            oWind_grnd.transform.localScale.x = 0.1135587;
            oWind_grnd.transform.localScale.y = 0.4051766;
            oWind_grnd.transform.localScale.z = 0.0163088;
            aCollectionObj_8.Add(oWind_grnd);
            nWindPOSX_side -= 3.10;
        }//end if
}//end for
c = 0;
//::::::::::::::::::::::::::::::::::::::::::::::::::::::::::end ground floor windows::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
}//end function Windows(RandX, RandZ)
//-----------------------------------------END  WINDOWS-----------------------------------------------------


//-----------------------------------------START  COLUMNS-----------------------------------------------------
public function Columns(RandX, RandZ){
        oColumn = GameObject.CreatePrimitive(PrimitiveType.Cube);
        oColumn.renderer.material.color = Color.white;

        oColumn.transform.position = Vector3(this.RandX -3.2, 1.302107, this.RandZ + 3.3772203);
        oColumn.transform.parent = oBuilding8.transform; //parenting
        oColumn.name = "Column 0";
        oColumn.transform.localScale.x = 0.07017206;
        oColumn.transform.localScale.y = 0.6454034;
        oColumn.transform.localScale.z = 0.1285665;
        aCollectionObj_8.Add(oColumn);
```

```
oColumnCube = GameObject.CreatePrimitive(PrimitiveType.Cube);

oColumnCube.transform.position = Vector3(this.RandX -3.2, 0.2373266, this.RandZ + 3.3772203);
oColumnCube.transform.parent = oColumn.transform; //parenting
oColumnCube.name = "Column base";
oColumnCube.transform.localScale.x = 1.43565;
oColumnCube.transform.localScale.y = 0.1184824;
oColumnCube.transform.localScale.z = 1.61189;
aCollectionObj_8.Add(oColumnCube);

var oColumnCube_Up = Instantiate(oColumnCube, Vector3(this.RandX - 3.2, 2.3029889, this.RandZ + 3.3772203), Quaternion.identity);
oColumnCube_Up.transform.parent = oColumn.transform; //parenting
oColumnCube_Up.name = "Column upper part";
oColumnCube_Up.transform.localScale.x = 1.43565;
oColumnCube_Up.transform.localScale.y = 0.1501914;
oColumnCube_Up.transform.localScale.z = 1.61189;
aCollectionObj_8.Add(oColumnCube_Up);

var nextPOSX_column:float = this.RandX - 1.62;
for(i=0; i<=3; i++){
    if(oColumn){
        var cloneColumn_ = Instantiate(oColumn, Vector3(nextPOSX_column, 1.302107, this.RandZ + 3.3772203), Quaternion.identity);
```

```
                cloneColumn_.transform.parent = oBuilding8.transform; //parenting
                cloneColumn_.name = "Column " + (++c);
                cloneColumn_.transform.localScale.x = 0.07017206;
                cloneColumn_.transform.localScale.y = 0.6454034;
                cloneColumn_.transform.localScale.z = 0.1285665;
                aCollectionObj_8.Add(cloneColumn_);
                nextPOSX_column += 1.58;
            }//end if
        }//end for
        c = 0;
}//end function Columns(RandX, RandZ)
//----------------------------------------END COLUMNS----------------------------------------


//------------------------------START  DOOR-------------------------------------------------
public function Door(RandX, RandZ){

        oWindow = GameObject.CreatePrimitive(PrimitiveType.Cube);
        Destroy(oWindow);
        //oWindow.transform.renderer.material.color = Color.clear;
        oWindow.renderer.material.mainTexture = Resources.Load("door_corfu2"); //import texture for door

        var nPosX_door:float = this.RandX + 0.80;
        for(i=0; i<=1; i++){
            if(oWindow){
```

```
                    var oDoor = Instantiate(oWindow, Vector3(nPosX_door, 1.123612, this.RandZ + 1.7776329), Quaternion.identity);
                    oDoor.name = "door " + (++c);
                    oDoor.transform.parent = oBuilding8.transform;
                    oDoor.transform.localScale.x = 0.1341711;
                    oDoor.transform.localScale.y = 0.5980287;
                    oDoor.transform.localScale.z = 0.019;
                    aCollectionObj_8.Add(oDoor);
                    nPosX_door -= 3.20;
                }//end if
        }//end for
        c = 0;
}//end function Door(RandX, RandZ)
//------------------------------END DOOR-------------------------------------------------------------

//-------------------------------------START  ROOF-----------------------------------------------------
public function Roof(RandX, RandZ){
        var tmpobj: GameObject = new GameObject();
        var mf: MeshFilter = tmpobj.AddComponent(MeshFilter);

//      tmpobj.AddComponent(MeshFilter);
        tmpobj.AddComponent(MeshRenderer);
        tmpobj.name = "Roof";
        tmpobj.transform.parent = oBuilding8.transform;
```

```
var verts: Vector3[] = new Vector3[6];
var normals: Vector3[] = new Vector3[6];
var uv: Vector2[] = new Vector2[6];
var tri: int[] = new int[18]; //3 vertices * 6 triangles =18


//vertices positioning
verts[0] = new Vector3(this.RandX - 3.6, 12.2, this.RandZ + 1); //P1
verts[1] = new Vector3(this.RandX - 3.6, 11.2, this.RandZ - 2); //P2
verts[2] = new Vector3(this.RandX - 3.6, 11.2, this.RandZ + 4); //P3

verts[3] = new Vector3(this.RandX + 3.6, 12.2, this.RandZ + 1); //P4
verts[4] = new Vector3(this.RandX + 3.6, 11.2, this.RandZ + 4); //P5
verts[5] = new Vector3(this.RandX + 3.6, 11.2, this.RandZ - 2); //P6


uv[0] = new Vector2(0, 0.5);
uv[1] = new Vector2(1, 1);
uv[2] = new Vector2(1, 0);

uv[3] = new Vector2(1, 0.5);
uv[4] = new Vector2(0, 0);
uv[5] = new Vector2(0, 1);


//triangles
```

```
tri[0] = 0; //refer to vertices: verts[0]
tri[1] = 1; //refer to vertices: verts[1]
tri[2] = 2; //refer to vertices: verts[2]

tri[3] = 0;
tri[4] = 2;
tri[5] = 4;

tri[6] = 3;
tri[7] = 0;
tri[8] = 4;

tri[9] = 3;
tri[10] = 4;
tri[11] = 5;

tri[12] = 3;
tri[13] = 5;
tri[14] = 0;

tri[15] = 0;
tri[16] = 5;
tri[17] = 1;

var mesh: Mesh = new Mesh();
mesh.name = "Roof";
mesh.vertices = verts;
```

```
        mesh.triangles = tri;
        mesh.uv = uv;
        mesh.normals = normals;
        mesh.RecalculateNormals();
        mf.mesh = mesh;
        tmpobj.renderer.material.mainTexture = Resources.Load("tiled_roof_red_grey");
}//end function Roof(RandX, RandZ)
//--------------------------------------END ROOF-----------------------------------------------------

//-----------------------------------START ARCH-----------------------------------------------------
private function Arch(RandX, RandZ, tmpobj) {
        tmpobj = new GameObject();
        var mf: MeshFilter = tmpobj.AddComponent(MeshFilter);

        tmpobj.AddComponent(MeshRenderer);
        tmpobj.name = "Arch";
        tmpobj.renderer.material.color = Color(1, 0.980, 0.804, 0);
        tmpobj.transform.parent = oBuilding8.transform;

        var verts: Vector3[] = new Vector3[14];
        var uv: Vector2[] = new Vector2[14];
        var normals: Vector3[] = new Vector3[14];
        var tri: int[] = new int[72]; //3 vertices * 12 triangles = 36 * 2 (duplicate triangles for the back face) =72
```

```
//vertices positioning
verts[0] = new Vector3(this.RandX - 1.9, 2.5, this.RandZ + 3.777); //P1
verts[1] = new Vector3(this.RandX - 1.9, 2, this.RandZ + 3.777); //P2
verts[2] = new Vector3(this.RandX - 1.35, 2, this.RandZ + 3.777); //P3
verts[3] = new Vector3(this.RandX - 1.35, 3.5, this.RandZ + 3.777); //P4
verts[4] = new Vector3(this.RandX - 3.5, 3.5, this.RandZ + 3.777); //P5
verts[5] = new Vector3(this.RandX - 3.5, 2, this.RandZ + 3.777); //P6
verts[6] = new Vector3(this.RandX - 2.95, 2, this.RandZ + 3.777); //P7
verts[7] = new Vector3(this.RandX - 2.95, 2.5, this.RandZ + 3.777); //P8
verts[8] = new Vector3(this.RandX - 2.9, 2.8, this.RandZ + 3.777); //P9
verts[9] = new Vector3(this.RandX - 2.75, 3, this.RandZ + 3.777); //P10
verts[10] = new Vector3(this.RandX - 2.55, 3.1, this.RandZ + 3.777); //P11
verts[11] = new Vector3(this.RandX - 2.3,  3.1, this.RandZ + 3.777); //P12
verts[12] = new Vector3(this.RandX - 2.1, 3, this.RandZ + 3.777); //P13
verts[13] = new Vector3(this.RandX - 1.95, 2.8, this.RandZ + 3.777); //P14

//normals
normals[0] = new Vector3(0, 0, 1);
normals[1] = new Vector3(0, 0, 1);
normals[2] = new Vector3(0, 0, 1);
normals[3] = new Vector3(0, 0, 1);
normals[4] = new Vector3(0, 0, 1);
normals[5] = new Vector3(0, 0, 1);
normals[6] = new Vector3(0, 0, 1);
normals[7] = new Vector3(0, 0, 1);
normals[8] = new Vector3(0, 0, 1);
normals[9] = new Vector3(0, 0, 1);
```

```
normals[10] = new Vector3(0, 0, 1);

normals[11] = new Vector3(0, 0, 1);

normals[12] = new Vector3(0, 0, 1);

normals[13] = new Vector3(0, 0, 1);


//uv mapping (textures)

uv[0] = new Vector2(0, 0.3);

uv[1] = new Vector2(0.2, 0);

uv[2] = new Vector2(0, 0);

uv[3] = new Vector2(0, 1);

uv[4] = new Vector2(1, 1);

uv[5] = new Vector2(1, 0);

uv[6] = new Vector2(0.8, 0);

uv[7] = new Vector2(0.8, 0.3);

uv[8] = new Vector2(0.75, 0.5);

uv[9] = new Vector2(0.65, 0.7);

uv[10] = new Vector2(0.6, 0.9);

uv[11] = new Vector2(0.4, 0.9);

uv[12] = new Vector2(0.45, 0.7);

uv[13] = new Vector2(0.35, 0.5);



//triangles

tri[0] = 0; //refer to vertices: verts[0]

tri[1] = 1; //refer to vertices: verts[1]

tri[2] = 2; //refer to vertices: verts[2]
```

```
tri[3] = 0;

tri[4] = 2;

tri[5] = 3;


tri[6] = 0;

tri[7] = 3;

tri[8] = 13;


tri[9] = 13;

tri[10] = 3;

tri[11] = 12;


tri[12] = 12;

tri[13] = 3;

tri[14] = 11;


tri[15] = 11;

tri[16] = 3;

tri[17] = 4;


tri[18] = 11;

tri[19] = 4;

tri[20] = 10;


tri[21] = 10;

tri[22] = 4;

tri[23] = 9;
```

```
tri[24] = 9;
tri[25] = 4;
tri[26] = 8;

tri[27] = 8;
tri[28] = 4;
tri[29] = 7;

tri[30] = 7;
tri[31] = 4;
tri[32] = 5;

tri[33] = 7;
tri[34] = 5;
tri[35] = 6;

tri[36] = 0;
tri[37] = 2;
tri[38] = 1;

tri[39] = 0;
tri[40] = 3;
tri[41] = 2;

tri[42] = 13;
tri[43] = 3;
```

```
tri[44] = 0;

tri[45] = 12;
tri[46] = 3;
tri[47] = 13;

tri[48] = 11;
tri[49] = 3;
tri[50] = 12;

tri[51] = 10;
tri[52] = 3;
tri[53] = 11;

tri[54] = 10;
tri[55] = 4;
tri[56] = 3;

tri[57] = 9;
tri[58] = 4;
tri[59] = 10;

tri[60] = 8;
tri[61] = 4;
tri[62] = 9;

tri[63] = 7;
```

```
        tri[64] = 4;
        tri[65] = 8;

        tri[66] = 7;
        tri[67] = 5;
        tri[68] = 4;

        tri[69] = 7;
        tri[70] = 6;
        tri[71] = 5;

        var mesh: Mesh = new Mesh();
        mesh.name = "Arch";
        mesh.vertices = verts;
        mesh.triangles = tri;
        mesh.uv = uv;
        mesh.normals = normals;
        //mesh.RecalculateNormals();
        tmpobj.renderer.material.mainTexture                                    =
Resources.Load("white_brick_wall");
        mf.mesh = mesh;
}//end function Arch(Rand_X, Rand_Z)


private function ArchVertical(RandX, RandZ, tmpobj){

        tmpobj = new GameObject();
```

```
var mf: MeshFilter = tmpobj.AddComponent(MeshFilter);


tmpobj.AddComponent(MeshRenderer);
tmpobj.name = "Arch vert";
tmpobj.renderer.material.color = Color(1, 0.980, 0.804, 0);
tmpobj.transform.parent = oBuilding8.transform;


var verts: Vector3[] = new Vector3[14];
var uv: Vector2[] = new Vector2[14];
var normals: Vector3[] = new Vector3[14];
var tri: int[] = new int[72]; //3 vertices * 12 triangles = 36 * 2 (duplicate triangles for the back face) =72


//vertices positioning
verts[0] = new Vector3(this.RandX , 2.5, this.RandZ - 1.9); //P1
verts[1] = new Vector3(this.RandX , 2, this.RandZ - 1.9); //P2
verts[2] = new Vector3(this.RandX , 2, this.RandZ - 1.35); //P3
verts[3] = new Vector3(this.RandX , 3.5, this.RandZ - 1.35); //P4
verts[4] = new Vector3(this.RandX , 3.5, this.RandZ - 3.5); //P5
verts[5] = new Vector3(this.RandX , 2, this.RandZ - 3.5); //P6
verts[6] = new Vector3(this.RandX , 2, this.RandZ - 2.95); //P7
verts[7] = new Vector3(this.RandX , 2.5, this.RandZ - 2.95); //P8
verts[8] = new Vector3(this.RandX , 2.8, this.RandZ - 2.9); //P9
verts[9] = new Vector3(this.RandX , 3, this.RandZ - 2.75); //P10
verts[10] = new Vector3(this.RandX , 3.1, this.RandZ - 2.55); //P11
verts[11] = new Vector3(this.RandX ,  3.1, this.RandZ - 2.3); //P12
verts[12] = new Vector3(this.RandX , 3, this.RandZ - 2.1); //P13
```

```
verts[13] = new Vector3(this.RandX , 2.8, this.RandZ - 1.95); //P14

//normals
normals[0] = new Vector3(0, 0, 1);
normals[1] = new Vector3(0, 0, 1);
normals[2] = new Vector3(0, 0, 1);
normals[3] = new Vector3(0, 0, 1);
normals[4] = new Vector3(0, 0, 1);
normals[5] = new Vector3(0, 0, 1);
normals[6] = new Vector3(0, 0, 1);
normals[7] = new Vector3(0, 0, 1);
normals[8] = new Vector3(0, 0, 1);
normals[9] = new Vector3(0, 0, 1);
normals[10] = new Vector3(0, 0, 1);
normals[11] = new Vector3(0, 0, 1);
normals[12] = new Vector3(0, 0, 1);
normals[13] = new Vector3(0, 0, 1);

//uv mapping (textures)
uv[0] = new Vector2(0, 0.3);
uv[1] = new Vector2(0.2, 0);
uv[2] = new Vector2(0, 0);
uv[3] = new Vector2(0, 1);
uv[4] = new Vector2(1, 1);
uv[5] = new Vector2(1, 0);
uv[6] = new Vector2(0.8, 0);
uv[7] = new Vector2(0.8, 0.3);
```

```
uv[8] = new Vector2(0.75, 0.5);

uv[9] = new Vector2(0.65, 0.7);

uv[10] = new Vector2(0.6, 0.9);

uv[11] = new Vector2(0.4, 0.9);

uv[12] = new Vector2(0.45, 0.7);

uv[13] = new Vector2(0.35, 0.5);



//triangles

tri[0] = 0; //refer to vertices: verts[0]

tri[1] = 1; //refer to vertices: verts[1]

tri[2] = 2; //refer to vertices: verts[2]


tri[3] = 0;

tri[4] = 2;

tri[5] = 3;


tri[6] = 0;

tri[7] = 3;

tri[8] = 13;


tri[9] = 13;

tri[10] = 3;

tri[11] = 12;


tri[12] = 12;

tri[13] = 3;
```

```
tri[14] = 11;

tri[15] = 11;
tri[16] = 3;
tri[17] = 4;

tri[18] = 11;
tri[19] = 4;
tri[20] = 10;

tri[21] = 10;
tri[22] = 4;
tri[23] = 9;

tri[24] = 9;
tri[25] = 4;
tri[26] = 8;

tri[27] = 8;
tri[28] = 4;
tri[29] = 7;

tri[30] = 7;
tri[31] = 4;
tri[32] = 5;

tri[33] = 7;
```

```
tri[34] = 5;
tri[35] = 6;

tri[36] = 0;
tri[37] = 2;
tri[38] = 1;

tri[39] = 0;
tri[40] = 3;
tri[41] = 2;

tri[42] = 13;
tri[43] = 3;
tri[44] = 0;

tri[45] = 12;
tri[46] = 3;
tri[47] = 13;

tri[48] = 11;
tri[49] = 3;
tri[50] = 12;

tri[51] = 10;
tri[52] = 3;
tri[53] = 11;
```

```
tri[54] = 10;
tri[55] = 4;
tri[56] = 3;

tri[57] = 9;
tri[58] = 4;
tri[59] = 10;

tri[60] = 8;
tri[61] = 4;
tri[62] = 9;

tri[63] = 7;
tri[64] = 4;
tri[65] = 8;

tri[66] = 7;
tri[67] = 5;
tri[68] = 4;

tri[69] = 7;
tri[70] = 6;
tri[71] = 5;

var mesh: Mesh = new Mesh();
mesh.name = "Arch vert";
mesh.vertices = verts;
```

```
        mesh.triangles = tri;
        mesh.uv = uv;
        mesh.normals = normals;
        //mesh.RecalculateNormals();
        tmpobj.renderer.material.mainTexture = Resources.Load("white_brick_wall");
        mf.mesh = mesh;

}//end function ArchVertical(Rand_X, Rand_Z, tmpobj)

public function DoArch(RandX, RandZ){
        tmpobj = new GameObject();
        Destroy(tmpobj);

        var nPOSZ:float = this.RandZ + 0.777;

        for(i=0; i<=3; i++){
                if(tmpobj){
                        Arch(this.RandX, nPOSZ, tmpobj);
                        this.RandX += 1.6;
                }//end if
        }//end for

        this.RandX -= 9.96;
        this.RandZ +=5.15;

        for(i=0; i<=1; i++){
```

```
            if(tmpobj){
                ArchVertical(this.RandX, this.RandZ, tmpobj);
                this.RandX += 7.1;
            }//end if
        }//end for
}//end function DoArch(Rand_X, Rand_Z)
//--------------------------------END ARCH-----------------------------------------------------------
```

//Pl_lvl_1a.js, Pl_lvl_1b.js, Pl_lvl_1c.js, Pl_lvl_1d.js

```
public var h1: House_1;
public var h3: House_3;
public var h4: House_4;

public var pln: GameObject;
pln = GameObject.Find("Plane_lvl_1a");
var rand: int;

public function Plane_lvl_1a() {

    rand = Random.Range(0,3); //random choice between integer numbers 0-2, which means h1 or h3 or h4

    if( rand == 0 ){
        h3 = gameObject.AddComponent("House_3");
        h3.RandX = pln.transform.position.x;
        h3.RandZ = pln.transform.position.z;
        h3.transform.rotation.y = pln.transform.rotation.y;
```

```
            h3.mainDO();
        }
        else if( rand == 1 ){
            h4 = gameObject.AddComponent("House_4");
            h4.RandX = pln.transform.position.x;
            h4.RandZ = pln.transform.position.z;
            h4.transform.rotation.y = pln.transform.rotation.y;
            h4.mainDO();
        }
        else if( rand == 2 ){
            h1 = gameObject.AddComponent("House_1");
            h1.RandX = pln.transform.position.x;
            h1.RandZ = pln.transform.position.z;
            h1.transform.rotation.y = pln.transform.rotation.y;
            h1.mainDO();
        }
}

public function Demolish(){ //demolish the house
        if(h3)
            h3.Destroy_();
        if(h4)
            h4.Destroy_();
        if(h1)
            h1.Destroy_();
}
```

```
//Pl_lvl_2a.js, Pl_lvl_2b.js, Pl_lvl_2c.js, Pl_lvl_2d.js, Pl_lvl_2e.js
public var h2: House_2;
public var h7: House_7;
public var h8: House_8;

public var pln: GameObject;
pln = GameObject.Find("Plane_lvl_2a");
var rand: int;
public function Plane_lvl_2a() {

    rand = Random.Range(0,3); //random choice between integer numbers 0-3, which means h2 or h6 or h7 or h8

    if( rand == 0 ){
        h2 = gameObject.AddComponent("House_2");
        h2.RandX = pln.transform.position.x;
        h2.RandZ = pln.transform.position.z;
        h2.mainDO();
    }
    else if( rand == 1 ){
        h7 = gameObject.AddComponent("House_7");
        h7.RandX = pln.transform.position.x;
        h7.RandZ = pln.transform.position.z;
        h7.mainDO();
    }
    else if( rand == 2 ){
        h8 = gameObject.AddComponent("House_8");
```

```
                h8.RandX = pln.transform.position.x;
                h8.RandZ = pln.transform.position.z;
                h8.mainDO();
        }
}

public function Demolish(){
        if(h2)
                h2.Destroy_();
        if(h7)
                h7.Destroy_();
        if(h8)
                h8.Destroy_();
}
```

//Pl_lvl_3a.js, Pl_lvl_3b.js, Pl_lvl_3c.js, Pl_lvl_3d.js, Pl_lvl_3e.js, Pl_lvl_3f.js, Pl_lvl_3g.js, Pl_lvl_3h.js

```
public var h5: House_5;
public var h7: House_7;
public var h8: House_8;

public var pln: GameObject;
pln = GameObject.Find("Plane_lvl_3a");
var rand: int;

public function Plane_lvl_3a() {
```

```
        rand = Random.Range(0,3);

        if( rand == 0 ){
                h5 = gameObject.AddComponent("House_5");
                h5.Rand_X = pln.transform.position.x;
                h5.Rand_Z = pln.transform.position.z;
                h5.mainDO();
        }
        else if( rand == 1 ){
                h7 = gameObject.AddComponent("House_7");
                h7.RandX = pln.transform.position.x;
                h7.RandZ = pln.transform.position.z;
                h7.mainDO();
        }
        else if( rand == 2 ){
                h8 = gameObject.AddComponent("House_8");
                h8.RandX = pln.transform.position.x;
                h8.RandZ = pln.transform.position.z;
                h8.mainDO();
        }
}

public function Demolish(){
        if(h5)
                h5.Destroy_();
        if(h7)
                h7.Destroy_();
```

```
	if(h8)
		h8.Destroy_();
}
```

//Pl_lvl_4.js

```
public var ch: Church;
public var pln: GameObject;
pln = GameObject.Find("Plane_lvl_4");

public function Plane_lvl_4() {

	ch = gameObject.AddComponent("Church");
	ch.RandX = pln.transform.position.x;
	ch.RandZ = pln.transform.position.z;
	ch.mainDO();
}

public function Demolish(){
	ch.Destroy_();
}
```

//Pl_lvl_5a.js, Pl_lvl_5b.js, Pl_lvl_5c.js, Pl_lvl_5d.js, Pl_lvl_5e.js, Pl_lvl_5f.js, Pl_lvl_5g.js, Pl_lvl_5h.js

```
public var h6: House_6;

public var pln: GameObject;
pln = GameObject.Find("Plane_lvl_5a");
```

```
public function Plane_lvl_5a() {

        h6 = gameObject.AddComponent("House_6");
        h6.RandX = pln.transform.position.x;
        h6.RandZ = pln.transform.position.z;
        h6.mainDO();
}

public function Demolish(){
        h6.Destroy_();
}
```

ΟΔΗΓΟΣ ΧΡΗΣΗΣ ΛΟΓΙΣΜΙΚΟΥ

Όπου εφαρμόζεται.