TECHNICAL UNIVERSITY OF CRETE

TUCapp

A 3D mobile application for social networking of an academic community

Lyngeridis Christos

A thesis in Electronic and Computer Engineering

10/11/2015

ΕΥΧΑΡΙΣΤΙΕΣ

Δε θα τα είχα καταφέρει χωρίς

τη δασκάλα μου,
τη μαμά μου, το μπαμπά μου,
τον αδερφούλη μου, το καλύτερο μου φίλο
και όλους τους άλλους

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΥΧΑΡΙΣΤΙΕΣ
ΠΕΡΙΕΧΟΜΕΝΑ
ΛΙΣΤΑ ΕΙΚΟΝΩΝ

ΚΕΦΑΛΑΙΟ 1
ΕΙΣΑΓΩΓΗ

1.1 Εισαγωγή
1.2 Περιγραφή Εφαρμογής
1.3 Στόχοι

ΚΕΦΑΛΑΙΟ 2
ΕΠΙΣΚΟΠΗΣΗ ΣΧΕΤΙΚΗΣ ΕΡΕΥΝΑΣ

2.1 Όροι και Έννοιες
2.2 Εργαλεία Ανάπτυξης
2.2.1 Unity3D
2.2.2 Drupal
2.2.3 SketchUp

ΚΕΦΑΛΑΙΟ 3
ΑΡΧΙΚΟΣ ΣΧΕΔΙΑΣΜΟΣ & ΓΡΑΦΙΚΗ ΔΙΕΠΑΦΗ ΧΡΗΣΤΗ

3.1 Ρύθμιση Unity Terrain
3.1.1 Δημιουργία Terrain
3.1.2 Ρύθμιση κάμερας
3.2 Sketchup map screenshot και απλά 3D μοντέλα
3.2.1 Δημιουργία SketchUp Google maps Screenshot
3.2.2 Δημιουργία αρχικών κτιρίων
3.2.3 Ενσωμάτωση στην εφαρμογή
3.3 Μενού γραφικής διεπαφής χρήστη
3.3.1 Δυνατότητες
3.3.2 Σχεδιασμός μενού

ΚΕΦΑΛΑΙΟ 4
ΔΗΜΙΟΥΡΓΊΑ ΤΡΙΣΔΙΑΣΤΑΤΟΥ ΠΕΡΙΒΑΛΛΟΝΤΟΣ

4.1 Τρισδιάστατη μοντελοποίηση
4.1.1 Δημιουργία πολύπλοκων μοντέλων κτιρίων με τη χρήση SketchUp
4.1.2 Χρήση τρισδιάστατου μοντέλου
4.1.3 Γλόμποι
4.1.4 Terrain
4.2 Κάμερα και φωτισμός
4.2.1 Κάμερα
4.2.2 Φωτισμός

ΚΕΦΑΛΑΙΟ 5
ΔΗΜΙΟΥΡΓΊΑ ΠΛΑΤΦΟΡΜΑΣ ΔΙΑΧΕΙΡΙΣΗΣ & ΣΥΝΔΕΣΗ ΜΕ ΕΦΑΡΜΟΓΗ

5.1 Προγραμματισμός με Drupal
5.1.1 Ρύθμιση Drupal backend
5.1.2 Βάσεις Δεδομένων drupal
5.1.3 Δημιουργία προγράμματος μαθημάτων
5.2 Σύνδεση εφαρμογής με βάση δεδομένων
5.2.1 Drupal – Δημιουργία TucApp API module
5.2.2 Unity3d- Ανάκτηση και αποκωδικοποίηση πληροφοριών βάσης δεδομένων

ΚΕΦΑΛΑΙΟ 6
ΥΛΟΠΟΙΗΣΗ ΧΕΙΡΙΣΜΟΥ & ΓΡΑΦΙΚΗΣ ΔΙΕΠΑΦΗΣ

6.1 Εντοπισμός κτιρίου με χρήση αφής
6.2 Προγραμματισμός γραφικής διεπαφής
6.2.1 Λογική GUI
6.2.2 Υλοποίηση μενού
6.3 Εμφάνιση θέσης στο χώρο με χρήση και μετατροπή συντεταγμένων GPS
6.4 Αξιολόγηση

ΚΕΦΑΛΑΙΟ 7
ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΜΕΛΛΟΝΤΙΚΕΣ ΕΠΕΚΤΑΣΕΙΣ

7.1 Συμπεράσματα
7.2 Μελλοντικές Επεκτάσεις

ΒΙΒΛΙΟΓΡΑΦΙΑ

ΛΙΣΤΑ ΕΙΚΟΝΩΝ

Εικόνα 1.1: Κόστος σήμανσης & Κατάσταση σήμανσης πολυτεχνείου κρήτης
Εικόνα 2.1: Περιβάλλον Unity3D
Εικόνα 2.2: Περιβάλλον Drupal
Εικόνα 2.3: Περιβάλλον Sketchup
Εικόνα 3.1: Terrain inspector
Εικόνα 3.2: Terrain resolution
Εικόνα 3.3: Perspective & orthographic κάμερα
Εικόνα 3.4: Κάμερα inspector
Εικόνα 3.5: Λήψη screenshot
Εικόνα 3.6: Προβολή screenshot
Εικόνα 3.7: Συναρμολόγηση screenshot
Εικόνα 3.8: Δημιουργία επιφάνειας από περίγραμμα κτιρίου
Εικόνα 3.9: SketchUp – Δημιουργία απλού κτιρίου
Εικόνα 3.10: Μακέτα σκηνής εφαρμογής
Εικόνα 3.11: Μενού αρχικής σελίδας
Εικόνα 3.12: Μενού αιθουσών
Εικόνα 3.13: Μενού μαθημάτων & μενού καθηγητών
Εικόνα 3.14: Μενού αναζήτησης
Εικόνα 3.15: Μενού ρυθμίσεων
Εικόνα 3.16: Μενού σύνδεσης χωρίς στοιχεία
Εικόνα 3.17: Μενού σύνδεσης με στοιχεία
Εικόνα 3.18: Μενού εγγραφής
Εικόνα 3.19: Μενού αποσύνδεσης
Εικόνα 3.20: Μενού πληροφοριών
Εικόνα 3.21: Μενού βοήθειας
Εικόνα 3.22: Μενού προσώπου
Εικόνα 3.23: Μενού μαθήματος
Εικόνα 3.24: Μενού αίθουσας

Εικόνα 3.25: Μενού δημοσίευσης μηνύματος
Εικόνα 3.26: Μενού σχολιασμού μηνύματος
Εικόνα 3.27: Μενού προειδοποίησης & σφάλματος
Εικόνα 3.28: Επιθυμητό αποτέλεσμα
Εικόνα 4.1: Σύγκριση τρισδιάστατων μοντέλων με φωτογραφίες
Εικόνα 4.2: Προσθήκη ορόφων στο μοντέλο
Εικόνα 4.3: Προσθήκη εσοχών στο μοντέλο
Εικόνα 4.4: Προσθήκη υλικών στο μοντέλο
Εικόνα 4.5: Επεξεργασία υλικού
Εικόνα 4.6: Αντιστοίχιση υλικών
Εικόνα 4.7: Τελικό μοντέλο κτιρίου
Εικόνα 4.8: Επίδραση επεξεργασίας material
Εικόνα 4.9: Επιλογή prefab
Εικόνα 4.10: Επεξεργασία prefab
Εικόνα 4.11: Mesh filter
Εικόνα 4.12: Mesh renderer
Εικόνα 4.13: Εισαγωγή prefab κτιρίου στη σκηνή
Εικόνα 4.14: Ανάθεση mesh collider
Εικόνα 4.15: Mesh collider & mesh filter
Εικόνα 4.16: Δημιουργία γηπέδων
Εικόνα 4.17: Ορισμός Tag & Layer αντικειμένου
Εικόνα 4.18: Prefab γλόμπου
Εικόνα 4.19: Αντίγραφο γλόμπου
Εικόνα 4.20: Υλικό γλόμπου
Εικόνα 4.21: GameObject γλόμπου
Εικόνα 4.22: Διάγραμμα ροής συναρτήσεων
Εικόνα 4.23: FloatIdea
Εικόνα 4.24: Rigidbody component
Εικόνα 4.25: Βούρτσα δέντρων
Εικόνα 4.26: Τοποθέτηση δέντρων
Εικόνα 4.27: Τοποθέτηση πινακίδων

Εικόνα 4.28: Προσθήκη βλάστησης
Εικόνα 4.29: Χρωματισμός terrain
Εικόνα 4.30: Σύγκριση εικόνας χάρτη με χρωματισμένο terrain
Εικόνα 4.31: Επιλογές RuntimePlatform
Εικόνα 4.32: Επιλογές TouchPhase
Εικόνα 4.33: Raycasting
Εικόνα 4.34: Δημιουργία Layer
Εικόνα 4.35: Έλεγχος κατάστασης κάμερας
Εικόνα 4.36: Υπολογισμός απόστασης δυο σημείων
Εικόνα 4.37: Συνάρτηση grabCamera
Εικόνα 4.38: Σύρσιμο κάμερας
Εικόνα 4.39: Συνάρτηση PointToGlide
Εικόνα 4.40: Προσθήκης δύναμης ώθησης σε κάμερα
Εικόνα 4.41: Μετακίνηση κάμερας μέσω του μετασχηματισμού
Εικόνα 4.42: Φως σημείου
Εικόνα 4.43: Σποτ
Εικόνα 4.44: Φως κατεύθυνσης
Εικόνα 4.45: Φως περιοχής
Εικόνα 4.46: Προσθήκη φωτισμού στη σκηνή
Εικόνα 4.47: Φωτισμός κτιρίου
Εικόνα 4.48: Ρύθμιση φωτισμού κτιρίου
Εικόνα 4.49: Συνάρτηση SetLight
Εικόνα 4.50: Παράδειγμα φωτισμού κτιρίου
Εικόνα 4.51: Mesh renderers αντικειμένων
Εικόνα 4.52: Ρυθμίσεις ποιότητας γραφικών
Εικόνα 5.1: Content Types
Εικόνα 5.2: Δημιουργία Content Types
Εικόνα 5.3: Λίστα nodes
Εικόνα 5.4: Δημιουργία Taxonomy
Εικόνα 5.5: Συσχετίσεις parent-child Taxonomy
Εικόνα 5.6: Καταχώρηση κτιρίου

Εικόνα 5.7: Πίνακας node
Εικόνα 5.8: Πίνακες custom πεδίων node
Εικόνα 5.9: Δημιουργία προγράμματος μαθημάτων
Εικόνα 5.10: Μορφή JSON
Εικόνα 5.11: Επιλογή συνάρτησης ανάκτησης
Εικόνα 5.12: Συναρτήσεις ανάκτησης πληροφοριών
Εικόνα 5.13: Συνάρτηση αποθήκευσης πληροφοριών
Εικόνα 5.14: Συνάρτηση κωδικοποίησης πληροφορίας
Εικόνα 5.15: Συνάρτηση GetWithOneArg
Εικόνα 5.16: Πίνακας απεικόνισης Dictionary
Εικόνα 5.17: Κλήση συνάρτησης WWWForm
Εικόνα 5.18: Parser αποκωδικοποίησης πληροφορίας
Εικόνα 5.19: Λειτουργία συστήματος λήψης πληροφοριών
Εικόνα 6.1: Εντοπισμός κτιρίου με χρήση αφής
Εικόνα 6.2: GuiObject components
Εικόνα 6.3: Συνάρτηση changeGuiWindow
Εικόνα 6.4: Μηχανισμός ενεργοποίησης μενού
Εικόνα 6.5: Συνάρτηση convertToArrLst
Εικόνα 6.6: Υλοποίηση μενού κτιρίου
Εικόνα 6.7: Υλοποίηση κουμπιού On/Off
Εικόνα 6.8: Αλλαγή χρώματος γλόμπου
Εικόνα 6.9: Ενέργεια συστήματος για άνοιγμα URI σε windows και android
Εικόνα 6.10: Έλεγχοι με κανονικές εκφράσεις
Εικόνα 6.11: Γεωειδές συστήματος WGS84
Εικόνα 6.12: Πλέγμα συστήματος UTM
Εικόνα 6.13: Προβολή Mercator
Εικόνα 6.14: Υπολογισμός συντεταγμένων στο terrain
Εικόνα 6.15: Σημεία αναφοράς για υπολογισμό συντεταγμένων
Εικόνα 6.16: Χρωματισμός κουμπιού GPS
Εικόνα 7.1: tucapp.gr – ιστοσελίδα φιλοξενίας εφαρμογής
Εικόνα 7.2: Σχολιασμός στα social media

Εικόνα 7.3: Γράφημα επισκεψιμότητας
Εικόνα 7.4: Στατιστικά λειτουργικού συστήματος χρηστών
Εικόνα 7.5: Μοντέλα φορητών συσκευών χρηστών
Εικόνα 7.6: Χαρακτηριστικά οθονών χρηστών

ΕΙΣΑΓΩΓΗ

ΚΕΦΑΛΑΙΟ 1

1.1 Εισαγωγή

Η ραγδαία ανάπτυξη της τεχνολογίας έχει καταστήσει σαφές ότι θα γίνει αναπόσπαστο κομμάτι της καθημερινότητας μας, καθώς όχι μόνο μπορεί να μας προσφέρει τρόπους διασκέδασης και ενημέρωσης αλλά πλέον η χρήση νέων τεχνολογιών και υπηρεσιών γίνεται διαθέσιμη σε όλο και ευρύτερο φάσμα ηλικίας λόγω της απλότητας που προσφέρουν αλλά και της ευκολίας διάθεσης τους. Σήμερα η πρόσβαση στο διαδίκτυο θεωρείται βασική ανάγκη του σύγχρονου ανθρώπου, είναι δυνατή ακόμα και σε περιοχές απομακρυσμένες από μεγάλα αστικά κέντρα και μπορεί να επιτευχθεί μέσω προσωπικού υπολογιστή, συσκευών κινητής τηλεφωνίας, ταμπλέτες ακόμα και τηλεοράσεις. Η διάθεση κινητών συσκευών με εύχρηστο περιβάλλον χρήστη είναι η κύρια αιτία που στις μέρες μας παρατηρούμε μικρά παιδιά να μπορούν να χειριστούν φορητές συσκευές αλλά και άτομα μεγαλύτερης ηλικίας, τα οποία δυσκολευόντουσαν στη χρήση προσωπικού υπολογιστή, να έχουν τη δυνατότητα να περιηγηθούν στο διαδίκτυο ακόμα και με ελάχιστες γνώσεις πάνω σε νέες τεχνολογίες.

Αυτή η έκρηξη τεχνολογικής εξέλιξης έχει ως αποτέλεσμα τη συνεχή κυκλοφορία νέων φορητών συσκευών, με μεγαλύτερη υπολογιστική δύναμη και καθιστά δυνατή τη διάθεση απαιτητικών προγραμμάτων σε ακόμα μεγαλύτερο κοινό. Τρισδιάστατα γραφικά μπορούν πολύ εύκολα να αναπαρασταθούν σε φορητές συσκευές χειρός και να δημιουργήσουν ένα εύχρηστο και όμορφο περιβάλλον για τον χρήστη ο οποίος περνάει όλο και περισσότερο χρόνο της ζωής του online.

1.2 Περιγραφή Εφαρμογής

Η έκταση του πολυτεχνείου Κρήτης, η αρχιτεκτονική διαρρύθμιση αλλά και η έλλειψη οργάνωσης πληροφοριών για άτομα και υπηρεσίες γέννησαν την ιδέα του TUCapp. Ενός social network το οποίο εκτός από την επικοινωνία των μελών του ιδρύματος θα προσφέρει ένα διαδραστικό τρισδιάστατο χάρτη στον οποίο οι φοιτητές αλλά και επισκέπτες του ιδρύματος θα μπορούν να περιηγηθούν, να βρουν πληροφορίες για τα κτίρια του πολυτεχνείου, τις υπηρεσίες αλλά και τους υπαλλήλους. Οι χρήστες θα μπορούν να βρουν πληροφορίες για το που βρίσκονται αίθουσες, γραφεία και τι μαθήματα γίνονται σε κάθε αίθουσα ανα πάσα στιγμή και οι υπάλληλοι θα μπορούν εύκολα να επεξεργαστούν πληροφορίες που τους αφορούν με τη χρήση του περιβάλλοντος διαχείρισης του Drupal. Ο χρήστης θα μπορεί να περιηγηθεί εναέρια, πάνω από μια τρισδιάστατη αναπαράσταση του πολυτεχνείου, αλλά και να κάνει χρήση των μενού για να βρει τις πληροφορίες που χρειάζεται.

1.3 Στόχοι

Κύριος στόχος αυτής τις διπλωματικής εργασίας είναι η δημιουργία του πυρήνα μιας εφαρμογής η οποία θα είναι διαθέσιμη αρχικά για συσκευές με λειτουργικό android αλλά θα μπορεί εύκολα να εξαχθεί και σε άλλου είδους συσκευές ή web applications και θα έχει όσο δυνατόν μεγαλύτερη διάρκεια ζωής με όσο το δυνατόν λιγότερες εργατοώρες συντήρησης.

Σε αυτή τη διπλωματική εργασία στοχεύουμε στην υλοποίηση των εξής δυνατοτήτων:

- Υποστήριξη χρηστών με ξεχωριστά δικαιώματα. Εγγραφή και σύνδεση μέσω της εφαρμογής
- Δημιουργία online πλατφόρμας εύκολης επεξεργασίας πληροφοριών και αποθήκευση σε online βάση δεδομένων
- Δημιουργία προγράμματος μαθημάτων και αποθήκευση σε μορφή JSON
- Εμφάνιση νέων πληροφοριών σε πραγματικό χρόνο στον χρήστη
- Δημιουργία τρισδιάστατου χάρτη
- Εντοπισμός θέσης στο χάρτη με τη χρήση GPS
- Δυνατότητα δημοσίευσης μηνυμάτων στη τρέχουσα θέση, σχολιασμός και ψήφιση δημοσιεύσεων άλλων χρηστών
- Σχεδιασμός που επιτρέπει την εύκολη προσθήκη τρισδιάστατων μοντέλων στην εφαρμογή και αντιστοίχιση τους με την online βάση δεδομένων χωρίς την ανάγκη προγραμματισμού
- Λογική μενού η οποία επιτρέπει την εύκολη προσθήκη νέων μενού

Απώτερος σκοπός είναι η δημιουργία μιας εφαρμογής η οποία εκτός από το να συγκεντρώνει πληροφορίες για όλα τα μαθήματα, αίθουσες και υπαλλήλους του πολυτεχνείου, θα εξυπηρετεί και σκοπούς ηλεκτρονικής σήμανσης του πολυτεχνείου Κρήτης. Έτσι όχι μόνο το πολυτεχνείο θα μπορέσει στο μέλλον να εξοικονομήσει ποσά της τάξης των 68.000 € όπου κόστισε η κατασκευή σήμανσης στο πολυτεχνείο, αλλά θα αποφευχθούν και φαινόμενα φυσικών φθορών και βανδαλισμού πινακίδων σήμανσης, (Εικόνα 1.1) αφού η σήμανση πλέον μεταφέρεται σε ηλεκτρονική μορφή στη συσκευή του κάθε χρήστη.

ΥΠΟΛΟΓΙΣΜΟΣ ΚΟΣΤΟΥΣ ΣΗΜΑΝΣΗΣ ΠΟΛΥΤΕΧΝΕΙΟΥ ΚΡΗΤΗΣ

Κωδικοί Κατασκευής	Σύνολα Α	Σύνολα Β	Σύνολα Γ	Σύνολα Δ	Σύνολα Ε	Σύνολα Κ	Σύνολα Μ	Σύνολα Εξωτερικών	Σύνολα Πάρκου	Γενικά Σύνολα	Τιμή μονάδας	Κόστος ανά Κωδικό
C01	0	10	11	75	111	86	150	0	0	**443**	€ 35.00	**€ 15,505.00**
C02.1	0	1	0	2	2	7	5	0	0	**17**	€ 60.00	**€ 1,020.00**
C02.2	0	3	1	4	5	3	3	0	0	**19**	€ 70.00	**€ 1,330.00**
C02.3	0	1	0	1	0	2	11	0	0	**15**	€ 80.00	**€ 1,200.00**
C02.4	0	0	0	0	0	4	0	0	0	**4**	€ 100.00	**€ 400.00**
C02.5	0	0	0	0	0	0	0	0	0	**0**	€ 120.00	**€ -**
C02.6	0	0	0	0	0	0	1	0	0	**1**	€ 130.00	**€ 130.00**
C03	0	0	0	1	2	2	1	0	0	**6**	€ 450.00	**€ 2,700.00**
C04	0	2	0	0	1	1	2	0	0	**6**	€ 100.00	**€ 600.00**
C05	0	0	0	0	0	0	0	9	0	**9**	€ 700.00	**€ 6,300.00**
C06	0	0	0	0	0	0	0	8	0	**8**	€ 700.00	**€ 5,600.00**
C07	0	0	0	0	0	0	0	25	0	**25**	€ 400.00	**€ 10,000.00**
C08.1	0	0	0	0	0	0	0	0	0	**0**	€ 500.00	**€ -**
C08.2	0	0	0	0	0	0	0	1	0	**1**	€ 650.00	**€ 650.00**
C08.3	0	0	0	0	0	0	0	2	0	**2**	€ 600.00	**€ 1,200.00**
C09	0	0	0	0	0	0	0	3	0	**3**	€ 1,200.00	**€ 3,600.00**
C10.1	0	23	2	39	60	41	94	0	0	**259**	€ 30.00	**€ 7,770.00**
C10.2	0	5	1	17	18	18	41	0	0	**100**	€ 15.00	**€ 1,500.00**
C11	0	0	0	0	0	0	0	13	0	**13**	€ 70.00	**€ 910.00**
C12	0	0	0	0	0	0	0	1	0	**1**	€ 4,500.00	**€ 4,500.00**
C13.1	1	0	0	0	0	0	0	0	1	**1**	€ 700.00	**€ 700.00**
C13.2	1	0	0	0	0	0	0	0	1	**1**	€ 700.00	**€ 700.00**
C14	5	0	0	0	0	0	0	0	5	**5**	€ 250.00	**€ 1,250.00**
Σύνολα	7	45	15	139	199	164	308	62	7	**939**		**€ 67,565.00**

Εκτυπώσεις	Πλάτος (m)	Ύψος (m)	Τετρ. Μέτρα	Τεμάχια	Έκταση	Τιμή/Τετρ. Μέτρο	Κόστος ανά Κωδικό
C03	0.5	1	0.5	6	3	€ 50.00	€ 150.00
C05	1	0.8	0.8	9	7.2	€ 50.00	€ 360.00
C09	0.5	0.9	0.45	3	1.35	€ 50.00	€ 67.50
C08.2	1.9	1.3	2.47	1	2.47	€ 50.00	€ 123.50
Σύνολο							**€ 701.00**
Γενικό Σύνολο							**€ 68,266.00**

Εικόνα 1.1: Κόστος σήμανσης & Κατάσταση σήμανσης πολυτεχνείου Κρήτης

ΕΠΙΣΚΟΠΗΣΗ ΣΧΕΤΙΚΗΣ ΕΡΕΥΝΑΣ

ΚΕΦΑΛΑΙΟ 2

2.1 Όροι και Έννοιες

Πριν ξεκινήσουμε το σχεδιασμό της εφαρμογής είναι σημαντικό να κατανοήσουμε κάποιες βασικές έννοιες και δυνατότητες που μας παρέχονται από τις πλατφόρμες τις οποίες θα χρησιμοποιήσουμε.

Vertex ονομάζεται ένα σημείο στο χώρο. Στο τρισδιάστατο χώρο μπορεί να αναπαρασταθεί με τρεις συντεταγμένες (x,y,z). Vertices ονομάζεται το σύνολο των vertex. Τρία vertices σχηματίζουν ένα τρίγωνο. Το τρίγωνο είναι το πιο απλό πολύγωνο το οποίο μπορεί να αναπαρασταθεί. Όλα τα τρισδιάστατα μοντέλα αποτελούνται από ένα συνδυασμό τριγώνων τα οποία σχηματίζουν τα πολύγωνα του μοντέλου. Από όσο περισσότερα πολύγωνα αποτελείται ένα μοντέλο, τόσο πιο βαρύ γίνεται για τη κάρτα γραφικών

Το πλέγμα πολυγώνων το οποίο αναπαριστά το σχήμα ενός τρισδιάστατου μοντέλου ονομάζεται mesh. Μπορούμε να έχουμε διάφορους τύπους meshes. Ο mesh collider για παράδειγμα είναι ένα πλέγμα το οποίο χρησιμεύει στη πλατφόρμα του Unity ώστε να εντοπίζει συγκρούσεις. Ο Mesh Collider εφαρμόζεται σε ένα αντικείμενο στη σκηνή μας ώστε να εντοπίζει συγκρούσεις με άλλα αντικείμενα. Όπως το απλό Mesh, είναι ένα πλέγμα πολυγώνων. Όσο περισσότερα πολύγωνα το αποτελούν τόσο πιο πολύπλοκες συγκρούσεις πρέπει να εντοπίζονται και επιβαρύνεται το πρόγραμμα. Αφού ο Mesh Collider δεν είναι ορατός στο χρήστη και χρησιμοποιείται μόνο για τον εντοπισμό συγκρούσεων μπορούμε κάποιες φορές να χρησιμοποιούμε απλούς colliders στα μοντέλα μας, όπως για παράδειγμα κύβους, όταν η μεγάλη ακρίβεια είναι κάτι το οποίο δε χρειάζεται. Αντίθετα ένας mesh renderer χρησιμεύει στην προβολή ενός τρισδιάστατου μοντέλου.

Μια απλή εικόνα bitmap η οποία μπορεί να εφαρμοστεί σε ένα mesh και να τυλίξει ένα τρισδιάστατο μοντέλο ώστε να του δώσει χρώμα ονομάζεται texture.

Το material καθορίζει την εμφάνιση μιας επιφάνειας, και περιέχει πληροφορίες για τα textures που χρησιμοποιούνται, πληροφορίες για τις διαστάσεις τους, τα χρώματα και άλλα. Οι διαθέσιμες πληροφορίες του εξαρτώνται από τον shader τον οποίο χρησιμοποιεί το material. Ένα material κάποιες φορές μπορεί να δώσει ακόμα και τη ψευδαίσθηση ανάγλυφου ή εκπομπής φωτός σε αντίθεση με τη χρήση απλών textures, λόγω του ότι χρησιμοποιεί shaders.

Οι shaders είναι μικρά script τα οποία περιέχουν μαθηματικούς υπολογισμούς και αλγόριθμους για τον υπολογισμό του χρώματος κάθε εμφανιζόμενου pixel, ανάλογα το φωτισμό και το material. Οι shaders ίσως είναι το πιο σημαντικό κομμάτι της αναπαράστασης ενός τρισδιάστατου μοντέλου, καθώς μπορεί να αλλάξει τελείως την εμφάνιση των γραφικών.

Τα scripts είναι το βασικό συστατικό κάθε προγράμματος. Είναι κομμάτια κώδικα τα οποία εκτός από το χειρισμό ενός προγράμματος μπορούν να καθορίσουν μέχρι και την εμφάνιση γραφικών, τεχνητή νοημοσύνη αντιπάλων σε ένα παιχνίδι και γενικά τη λειτουργικότητα ενός προγράμματος. Στα πλαίσια αυτής της διπλωματικής θα προγραμματίσουμε κυρίως με C# scripts με τη βοήθεια του .NET Framework της microsoft, αλλά θα χρησιμοποιήσουμε και javascripts για το προγραμματισμό του website της εφαρμογής

Τα Assets είναι τα μοντέλα, τα textures, οι ήχοι, τα scripts και οποιοδήποτε αρχείο χρησιμοποιούμε στο unity μας. Τα assets μας σχηματίζουν τα GameObjects είναι τα αντικείμενα στο Unity τα οποία μπορούν να αντιπροσωπεύουν χαρακτήρες, μοντέλα και σκηνικά. Μόνα τους δε προσφέρουν κάτι αλλά χρησιμεύουν σαν δοχεία για Components, τα οποία υλοποιούν τη λειτουργικότητα τους. Τα Components, υλοποιούν τη λειτουργικότητα ενός GameObject. Σε ένα GameObject μπορούμε να προσθέσουμε σαν components, scripts, meshes, φωτισμό και γενικά οποιοδήποτε asset επιθυμούμε. Οποιοδήποτε asset θέλουμε να χρησιμοποιήσουμε πρέπει να υπάρχει σαν component σε ένα GameObject στη σκηνή μας, για παράδειγμα αν θέλουμε να τρέξουμε ένα script αυτό θα πρέπει να γίνει καλώντας το GameObject στο οποίο έχουμε προσθέσει σαν component το συγκεκριμένο script.

Rigidbodies ονομάζονται τα components τα οποία επιτρέπουν σε ένα GameObject να δέχεται δυνάμεις φυσικής. Αν ένα GameObject έχρι Rigidbody τότε θα επηρεάζεται από τη βαρύτητα ή τις τριβές που προκαλεί η μηχανή φυσικής του Unity. Αν έχει και Mesh Collider τότε θα αλληλεπιδρά με άλλα αντικείμενα που έχουν Mesh Collider και το ένα θα εκτοπίζει το άλλο

Αν ένα GameObject ή Asset εμφανίζεται πολλές φορές στη σκηνή μας και θέλουμε εύκολα να επεξεργαστούμε κάθε αντίγραφο του τότε μπορούμε να δημιουργήσουμε ένα prefab για αυτό. Τα prefabs μπορούν να είναι από GameObjects με components μέχρι shaders και materials. Αν για παράδειγμα ένας άσπρος τοίχος για τον οποίο χρησιμοποιούμε ένα prefab material, εμφανίζεται πολλές φορές στη σκηνή μας και εμείς αλλάξουμε το χρώμα του material από άσπρο σε μαύρο, τότε η αλλαγή θα εφαρμοστεί αυτόματα σε όλα τα αντίγραφα του prefab στη σκηνή μας, χωρίς να χρειάζεται να αλλάζουμε το κάθε GameObject ξεχωριστά.

Οι κάμερες στο Unity όπως και στο πραγματικό κόσμο χρησιμεύουν στο να παρουσιάζουν μια σκηνή. Στο Unity έχουμε τουλάχιστον μια κάμερα, αλλά μπορούμε να χρησιμοποιήσουμε και περισσότερες.

Τα φώτα είναι σημαντικά σε μια σκηνή. Ενώ τα meshes και τα textures καθορίζουν το σχήμα και την εμφάνιση μιας σκηνής, τα φώτα καθορίζουν το χρώμα και τη διάθεση ενός περιβάλλοντος. GUI ονομάζεται η γραφική διεπαφή χρήστη. Αποτελείται κυρίως από δισδιάστατα γραφικά και κείμενα στη μορφή μενού, αλλά μπορεί να είναι και τρισδιάστατα αντικείμενα στο χώρο με τα οποία ο χρήστης αλληλεπιδρά. Για να αλληλεπιδράσει ο χρήστης με ένα αντικείμενο στη σκηνή κάνουμε χρήση των δυνατοτήτων Raycasting του Unity. Ένα Raycast χρησιμοποιεί Rays ώστε να εντοπίσει αντικείμενα. Ray ονομάζεται μια ακτίνα στο χώρο. Η ακτίνα μπορεί να έχει συγκεκριμένο μήκος αν είναι μεταξύ δυο σημείων στο χώρο ή να έχει άπειρο μήκος αν πηγάζει από ένα σημείο και εκτείνεται προς μια συγκεκριμένη κατεύθυνση.

Ένα Raycast στέλνει μια ακτίνα προς μια συγκεκριμένη κατεύθυνση και μπορεί να επιστρέψει αποτελέσματα ανάλογα με το τι "χτύπησε" χρησιμοποιώντας τη RaycastHit. Μπορούμε να κάνουμε Raycast είτε από ένα σημείο (για παράδειγμα τη κάμερα) προς συγκεκριμένη κατεύθυνση αλλά ακόμα και προς όλες τις κατευθύνσεις που πηγάζουν από ένα σημείο.

Εκτελώντας ένα Raycast μπορούμε να αποθηκεύσουμε τα αποτελέσματα του σε μια δομή RaycastHit η οποία περιέχει πληροφορίες σχετικά με την επιφάνεια στην οποία προσέκρουσε μια ακτίνα. Οι πληροφορίες μπορεί να είναι από τις συντεταγμένες και το χρώμα μιας επιφάνειας, μέχρι και το όνομα του GameObject που τη περιέχει σαν component._To Event System του Unity χρησιμοποιείται για τον εντοπισμό γεγονότων που βασίζονται σε ενέργειες του χρήστη όπως άγγιγμα της οθόνης ή χρήση των μενού, αλλά και χρήση Raycasting.

Η κλάση Player Prefs αποθηκεύει και επεξεργάζεται πληροφορίες μεταξύ συνεδριών. Είναι χρήσιμη αν θέλουμε το πρόγραμμα να "θυμάται" όταν τερματίζει, συγκεκριμένες πληροφορίες για την επόμενη συνεδρία.

Το GPS είναι ένα σύστημα πλοήγησης που παρέχει πληροφορίες τοποθεσίας και ώρας, οπουδήποτε πάνω ή κοντά στη γη όπου υπάρχει ορατότητα σε τουλάχιστον τέσσερις δορυφόρους GPS. Οι Ηνωμένες Πολιτείες το δημιούργησαν, το συντηρούν και το κάνουν διαθέσιμο σε οποιονδήποτε με δέκτη GPS.

Το WGS είναι ένα πρότυπο για χρήση στη χαρτογραφία, γεωδαισία, και πλοήγηση με τη χρήση GPS. Το WGS 84 είναι η τελευταία έκδοση του και η γεωδαιτική στάθμη αναφοράς επιφάνειας είναι ένα πεπλατυσμένο σφαιροειδές.Η αρχή συντεταγμένων WGS 84 βρίσκεται στο κέντρο της γης και το σφάλμα εκτιμάται ότι είναι κάτω από 2 εκ. Η αναπαράσταση των συντεταγμένων μπορεί να γίνει με μοίρες ή δεκαδική μορφή.

Το σύστημα UTM είναι ένας τρόπος αναπαράστασης σε καρτεσιανές συντεταγμένες ο οποίος χωρίζει το χάρτη σε πεδία χρησιμοποιώντας ένα πλέγμα τετραγώνων. Η αρίθμηση του πλέγματος γίνεται με αριθμούς για το γεωγραφικό μήκος και γράμματα για το γεωγραφικό πλάτος. Για την αναπαράσταση ενός σημείου στο χάρτη χρησιμοποιούμε το κωδικό του πεδίου στο πλέγμα στο οποίο βρισκόμαστε, για παράδειγμα "34S", το easting, το οποίο είναι η συντεταγμένη X από την αρχή των αξόνων και το northing το οποίο είναι η συντεταγμένη Y.

2.2 Εργαλεία Ανάπτυξης

Οι πλατφόρμες ανάπτυξης που χρησιμοποιήθηκαν είναι τρεις. Για τη κατασκευή της ιστοσελίδας του TUCapp χρησιμοποιήθηκε η πλατφόρμα ανοικτού κώδικα Drupal. Η κατασκευή των τρισδιάστατων μοντέλων έγινε με το SketchUp, ένα πρόγραμμα τρισδιάστατης μοντελοποίησης από τη Google το οποίο χρησιμοποιείται κυρίως για αρχιτεκτονικά σχέδια, ενώ για τη δημιουργία της εφαρμογής για android έγινε χρήση της μηχανής γραφικών Unity3D, η οποία παρέχει δυνατότητες κατασκευής και επεξεργασίας γραφικών αλλά και προγραμματισμού scripts. Θα καλύψουμε αναλυτικά τις ξεχωριστές δυνατότητες κάθε εργαλείου στα επόμενα κεφάλαια, αλλά σε αυτό το σημείο θα ήταν καλό να δούμε επιγραμματικά κάποιες τις οποίες προσφέρουν, ώστε να γίνει κατανοητό το γιατί επιλέξαμε τα συγκεκριμένα εργαλεία για την υλοποίηση αυτής της διπλωματικής εργασίας.

2.2.1 Unity3D

Η μηχανή γραφικών Unity3D δίνει στο χρήστη τη δυνατότητα να δημιουργήσει τρισδιάστατους κόσμους εύκολα και γρήγορα. Η ευκολία χρήσης δίνει τη δυνατότητα να ρυθμιστούν τα γραφικά μιας εφαρμογής, η φυσική και άλλες παράμετροι ακόμη και χωρίς τη χρήση κώδικα ώστε να μπορεί να εστιάσει στο προγραμματισμό της λειτουργικότητας ή των ξεχωριστών δυνατοτήτων της εφαρμογής, χωρίς όμως να καθιστά δύσκολη την επεξεργασία ή δημιουργία γραφικών και φυσικής για παράδειγμα, με τη χρήση προγραμματισμού. Η εύχρηστη διεπαφή (Εικόνα 2.1) και οι βιβλιοθήκες που παρέχει η Unity3D επιτρέπουν τη δημιουργία πολύπλοκων cross platform εφαρμογών, εφαρμογών δηλαδή οι οποίες μπορούν να τρέξουν σε διάφορα λειτουργικά συστήματα, όπως για παράδειγμα συστήματα με λειτουργικό android, ios ή windows, χωρίς να χρειάζεται να γίνουν αλλαγές στο κώδικα της εφαρμογής. Η εργασία με Unity3D μπορεί να μειώσει δραστικά το χρόνο εργασιών και συντήρησης που χρειάζεται μια εφαρμογή ώστε να λειτουργεί σε διάφορα λειτουργικά συστήματα και μας βοηθά να δημιουργήσουμε cross platform εφαρμογές.

Η γραφική διεπαφή του Unity μπορεί να παραμετροποιηθεί πλήρως . Μπορούμε να προσθέσουμε ή να αφαιρέσουμε καρτέλες και να αλλάξουμε τη θέση τους. Οι κύριες καρτέλες του Unity είναι οι Inspector, Scene, Game, Hierarchy και Project.

Στη καρτέλα Project βλέπουμε όλα τα αρχεία του project μας, είτε είναι αρχεία script ή πολύπλοκα τρισδιάστατα μοντέλα εμφανίζονται εκεί. Στη καρτέλα scene μπορούμε να επεξεργαστούμε τη σκηνή μας. Η σκηνή είναι ο κόσμος της εφαρμογής μας. Εκεί τοποθετούμε όλα τα αντικείμενα τα οποία θέλουμε να χρησιμοποιήσουμε. Όλα τα αντικείμενα που βρίσκονται στη σκηνή μας εμφανίζονται στη καρτέλα Hierarchy με το όνομα τους. Για να τοποθετήσουμε ένα αρχείο στη σκηνή αρκεί να το σύρουμε με το ποντίκι από τη καρτέλα project σε μια από τις καρτέλες scene ή hierarchy.

Στη καρτέλα hierarchy μπορούμε να ορίσουμε σχέσεις parent-child μεταξύ αντικειμένων της σκηνής. Αν κάνουμε κάτι τέτοιο οι συντεταγμένες του child αντικειμένου στο χώρο θα εμφανίζονται σε σχέση με αυτές του parent και όχι με βάση την αρχή των αξόνων στον κόσμο μας. Αν μετακινηθεί ή αλλάξει το μέγεθος του parent ολα τα αντικείμενα που έχουν οριστεί ως child ακολουθούν το μετασχηματισμό.

Εικόνα 2.1: Περιβάλλον Unity3D

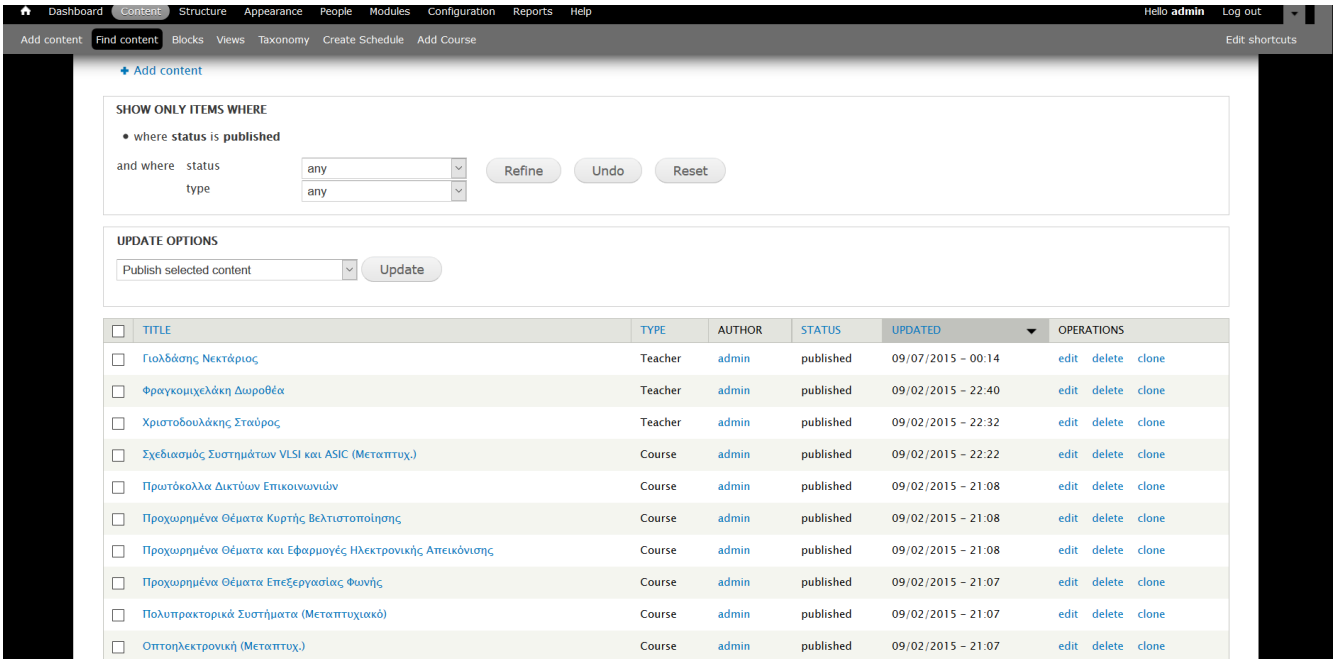
Για να επεξεργαστούμε ένα αντικείμενο μπορούμε να επεξεργαστούμε το μετασχηματισμό του από τη σκηνή, για παράδειγμα να το μετακινήσουμε ή να το μεγαλώσουμε με τα βελάκια που φαίνονται στο πάνω αριστερό μέρος της οθόνης. Ή μπορούμε αν θέλουμε να επεξεργαστούμε με ακρίβεια το μετασχηματισμό του να το επιλέξουμε και να αλλάξουμε τις ρυθμίσεις του από τη καρτέλα inspector. Σε αυτή τη καρτέλα, όταν επιλέγουμε ένα αντικείμενο, μπορούμε να επεξεργαστούμε όλες τις τιμές του. Από τον inspector μπορούμε να προσθέσουμε και components σε ένα αντικείμενο μας.

Στη καρτέλα Game εμφανίζεται ότι είναι ορατό στο χρήστη. Μπορούμε με τη χρήση του κουμπιού play να ξεκινήσουμε μια προσομοίωση της εφαρμογής και να τη χειριστούμε από τη καρτέλα play.

Το Unity είναι μια ισχυρή μηχανή γραφικών, που μας δίνει τη δυνατότητα προγραμματισμού σε τρεις διαφορετικές γλώσσες scripting, και με τη βοήθεια των κλάσεων της μπορούμε να χρησιμοποιήσουμε τη φυσική του unity σε μια σκηνή, να προσθέσουμε κίνηση σε γραφικά και τρισδιάστατα μοντέλα και να δημιουργήσουμε παράθυρα γραφικής διεπαφής.

2.2.2 Drupal

Το Drupal είναι μια πλατφόρμα διαχείρισης περιεχομένου, ανοικτού κώδικα η οποία χρησιμοποιείται από εκατομμύρια ιστοσελίδες και εφαρμογές. Επιτρέπει την προσθήκη και επεξεργασία περιεχομένου σε μια ιστοσελίδα αλλά δίνει και την επιλογή προσθήκης επιπλέον δυνατοτήτων με τη χρήση modules τα οποία μπορούν είτε να βρεθούν online ή να τα γράψουμε μόνοι μας και να τα προσθέσουμε στη σελίδα μας. Τα παραπάνω αλλά και η δυνατότητα υποστήριξης χρηστών με διαφορετικά δικαιώματα, είναι ο λόγος που έγινε επιλογή του Drupal για τη κατασκευή της σελίδας της εφαρμογής αλλά και της πλατφόρμας διαχείρισης. Καταφέραμε έτσι να επικεντρωθούμε στην υλοποίηση πιο σύνθετης λειτουργικότητας και όχι τόσο στην υλοποίηση υποστήριξης χρηστών και γραφικής διεπαφής για τη καταχώρηση περιεχομένου (Εικόνα 2.2). Μειονέκτημα της χρήσης Drupal είναι το ότι δε μπορούμε να επιλέξουμε τη δομή της βάσης δεδομένων, καθώς δημιουργείται αυτόματα από το Drupal και έχει συγκεκριμένη δομή. Αυτό έχει ως αποτέλεσμα τη χρήση πιο πολύπλοκων ερωτημάτων βάσης δεδομένων, αλλά αυτό είναι κάτι το οποίο μπορούμε να παραβλέψουμε μπροστά στην ευκολία χρήσης της πλατφόρμας Drupal.



Εικόνα 2.2: Περιβάλλον Drupal

Το Drupal προσφέρει έτοιμες πλατφόρμες διαχείρισης περιεχομένου τις οποίες μπορεί να χρησιμοποιήσει οποιοσδήποτε κατέχει βασικές γνώσεις χρήσης υπολογιστή. Κάτι το οποίο θέλουμε να πετύχουμε, ώστε να δημιουργήσουμε μια εύχρηστη εφαρμογή την οποία θα μπορεί να διαχειριστεί οποιοδήποτε μέλος της πολυτεχνικής κοινότητας.

2.2.3 SketchUp

Για τη κατασκευή των τρισδιάστατων μοντέλων χρησιμοποιήθηκε το SketchUp. Το SketchUp είναι ένα πρόγραμμα τρισδιάστατης μοντελοποίησης από τη Google το οποίο χρησιμοποιείται κυρίως από αρχιτέκτονες και προσφέρει ένα εύχρηστο περιβάλλον με δυνατότητες τρισδιάστατης μοντελοποίησης και εφαρμογή materials στο τρισδιάστατο μοντέλο (Εικόνα 2.3). Το μειονέκτημα της μοντελοποίησης με SketchUp είναι ότι δε προσφέρει δυνατότητες κίνησης στα τρισδιάστατα μοντέλα μας καθώς προορίζεται για τη κατασκευή κτιρίων. Λόγω όμως του ότι στα πλαίσια αυτής της διπλωματικής δε χρειαζόμαστε κινούμενα αλλά στατικά μοντέλα αλλά και ότι τα περισσότερα από αυτά είναι κτίρια το SketchUp είναι η ιδανική επιλογή καθώς όχι μόνο οι δυνατότητες που προσφέρει βοηθούν στην ευκολότερη μοντελοποίηση κτιρίων, αλλά και μπορεί να γίνει εξαγωγή των μοντέλων με μορφή αρχείου που μπορεί να χρησιμοποιηθεί από το Unity3D.

Εικόνα 2.3: Περιβάλλον Sketchup

ΑΡΧΙΚΟΣ ΣΧΕΔΙΑΣΜΟΣ & ΓΡΑΦΙΚΗ ΔΙΕΠΑΦΗ ΧΡΗΣΤΗ

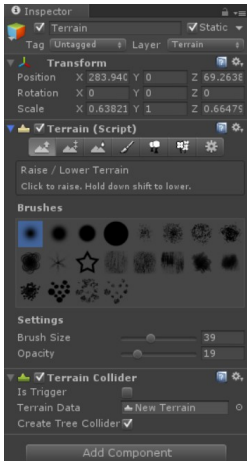
ΚΕΦΑΛΑΙΟ 3

3.4 Ρύθμιση Unity Terrain

Το πρώτο βήμα είναι η δημιουργία του terrain. Το terrain είναι η επίπεδη επιφάνεια πάνω στην οποία θα τοποθετηθούν κτίρια, δέντρα και θα σχεδιαστεί ο χάρτης του πολυτεχνείου. Έπειτα θα πρέπει να ρυθμίσουμε και τη κάμερα μας, η οποία θα εμφανίζει το περιεχόμενο της εφαρμογής στο χρήστη.

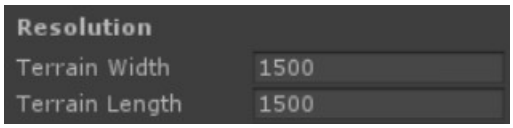
3.4.1 Δημιουργία Terrain

Η εύκολη δημιουργία terrain είναι μια από τις αμέτρητες δυνατότητες του Unity3D οι οποίες διευκολύνουν τον προγραμματιστή. Η δημιουργία του terrain όπως και κάθε άλλου GameObject μπορεί να γίνει από το αντίστοιχο μενού του Unity3D. Μόλις επιλέξουμε τη δημιουργία terrain μια επίπεδη επιφάνεια θα εμφανιστεί στη σκηνή μας, της οποίας τις ρυθμίσεις μπορούμε να αλλάξουμε επιλέγοντας το GameObject από το hierarchy. Μόλις το κάνουμε αυτό θα δούμε στον inspector τις ξεχωριστές ρυθμίσεις που μπορούν να γίνουν. Το terrain μας δίνει τη δυνατότητα αυξομείωσης του υψομέτρου, χρωματισμού του εδάφους, μαζικής προσθήκης δέντρων και βλάστησης εκτός των επιπλέον δυνατοτήτων ρύθμισης και προσθήκης components που προσφέρουν GameObjects (Εικόνα 3.1).



Εικόνα 3.1: Terrain inspector

Αρχικά η επίπεδη λευκή επιφάνεια μας αρκεί ώστε να σχηματιστεί ένα πρωτότυπο. Ρυθμίζουμε μόνο την ανάλυση του terrain σε 1500x1500 world units (Εικόνα 3.2). Είναι μια καλή ανάλυση για τον αρχικό σχεδιασμό της εφαρμογής η οποία προσφέρει αρκετό "χώρο" για τη προσθήκη των τρισδιάστατων μοντέλων αλλά και τη χρήση GPS. Όσο μεγαλύτερη η επιφάνεια του terrain, τόσο μεγαλύτερη και η ακρίβεια που θα έχουμε.



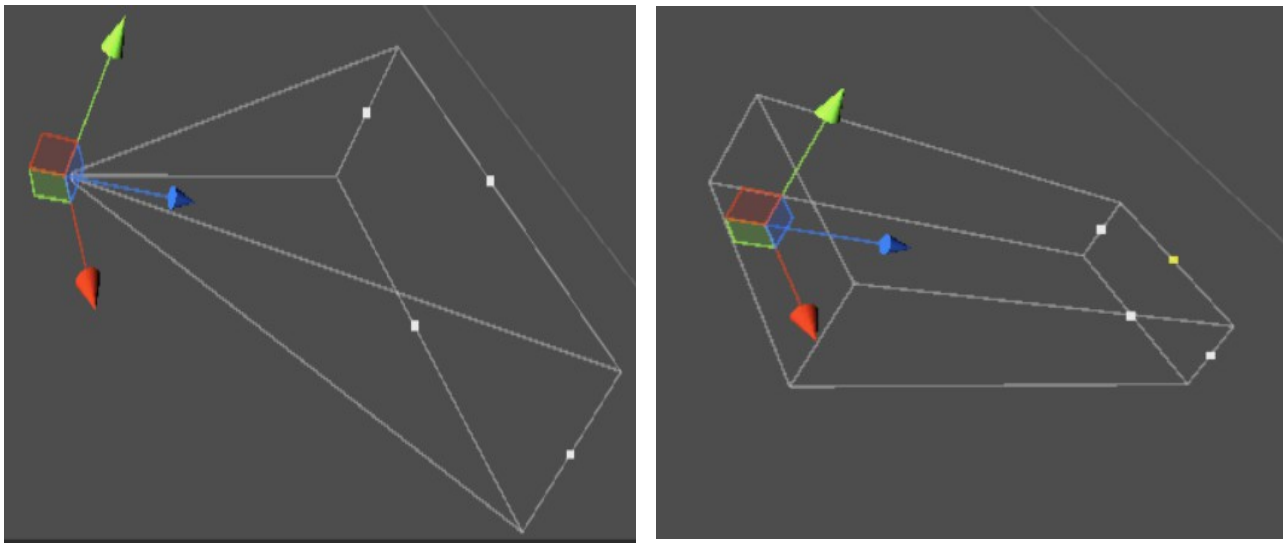
Εικόνα 3.2: Terrain resolution

3.4.2 Ρύθμιση κάμερας

Τώρα αρκεί να ρυθμίσουμε τη κάμερα. Η κάμερα είναι το μέσο το οποίο προβάλει στο χρήστη την εικόνα που βλέπει και κατά τη δημιουργία του project υπάρχει πάντα μια κάμερα στη σκηνή. Πρέπει να υπάρχει πάντα τουλάχιστον μια κάμερα στη σκηνή μας αλλά μπορούν να χρησιμοποιηθούν και περισσότερες. Κάτι τέτοιο θα ήταν χρήσιμο σε ένα παιχνίδι αυτοκινήτων όπου η μια κάμερα για παράδειγμα θα προβάλει το δρόμο μπροστά από το αυτοκίνητο, ενώ μια δεύτερη θα μπορούσε να χρησιμοποιηθεί ώστε να προβάλει το δρόμο πίσω από το αυτοκίνητο, πάνω στο καθρέπτη, ώστε να δημιουργήσει τη ψευδαίσθηση του καθρεφτίσματος. Στη περίπτωση μας, μια κάμερα αρκεί, καθώς το μόνο που θέλουμε να προβάλλεται είναι μια πανοραμική λήψη του terrain μας.

Αρχικά θέλουμε να ρυθμίσουμε τη κάμερα ώστε να έχει perspective προβολή. Αυτή η ρύθμιση δίνει βάθος στην εικόνα. Όπως βλέπουμε στην εικόνα, σε αυτή τη περίπτωση το οπτικό πεδίο της κάμερας είναι σαν ένα πρίσμα, και ότι περιβάλλεται από αυτό το πρίσμα, προβάλλεται στην οθόνη του χρήστη, δίνοντας τη ψευδαίσθηση βάθους. Αυτή είναι η κατάλληλη προβολή για τρισδιάστατες εφαρμογές.

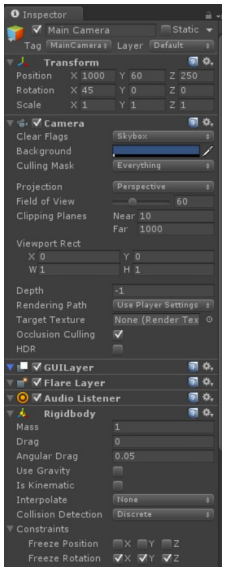
Αντίθετα η επιλογή orthographic προβολής, μετατρέπει το οπτικό πεδίο της κάμερας σε ένα ορθογώνιο παραλληλεπίπεδο, με αποτέλεσμα να μη διακρίνεται βάθος στην εικόνα και συνίσταται για δισδιάστατες εφαρμογές (Εικόνα 3.3).


Εικόνα 3.3: Perspective & orthographic κάμερα

Το επόμενο βήμα είναι να ορίσουμε το ύψος, τη περιστροφή αλλά και το πεδίο οράσεως της κάμερας.

Αυτό γίνεται εύκολα από τη σκηνή, μετακινώντας τη κάμερα με τα αντίστοιχα βελάκια στη θέση που θέλουμε και καθορίζοντας το εύρος του πεδίου οράσεως της κάμερας χρησιμοποιώντας τα άσπρα τετράγωνα που φαίνονται στις παραπάνω εικόνες.

Κατά τη διάρκεια ρύθμισης της κάμερας βλέπουμε στη καρτέλα Game, μια προ-επισκόπηση του τι θα είναι ορατό στο χρήστη. Έτσι μπορούμε να δούμε πως επηρεάζει τη προβολή η κάθε αλλαγή χωρίς να χρειαστεί να τρέξουμε την εφαρμογή.



Εικόνα 3.4: Κάμερα inspector

Όταν είμαστε ικανοποιημένοι με τη προβολή μπορούμε από τη καρτέλα Inspector να δούμε τη τρέχουσα θέση της κάμερας, τη περιστροφή της αλλά και να κάνουμε επιπλέον αλλαγές, όπως για παράδειγμα το τι θα προβάλει η κάμερα, μέσω της επιλογής Culling Mask. Στη περίπτωση μας θέλουμε να προβάλλει τα πάντα, οπότε αφήνουμε το προεπιλεγμένο Everything (Εικόνα 3.4).

Μόλις τελειώσουμε με τις ρυθμίσεις, προσθέτουμε σαν component στη κάμερα μας ένα Rigidbody. Αυτό όχι μόνο θα μας επιτρέψει μετέπειτα να ασκήσουμε στη κάμερα δυνάμεις φυσικής για να τη μετακινήσουμε, αλλά μας δίνει και τη δυνατότητα να "κλειδώσουμε" τη περιστροφή ή τη θέση της κάμερας. Στη περίπτωση μας, κλειδώνουμε το X,Y,Z της περιστροφής καθώς δε θέλουμε η κάμερα να περιστρέφεται αλλά μόνο να μετακινείται.

Μόλις είμαστε ικανοποιημένοι με το αποτέλεσμα που βλέπουμε, η κάμερα μας είναι έτοιμη. Τώρα αρκεί να προβάλουμε κάτι περισσότερο από μια επίπεδη επιφάνεια ώστε η σκηνή μας να πάρει μορφή.

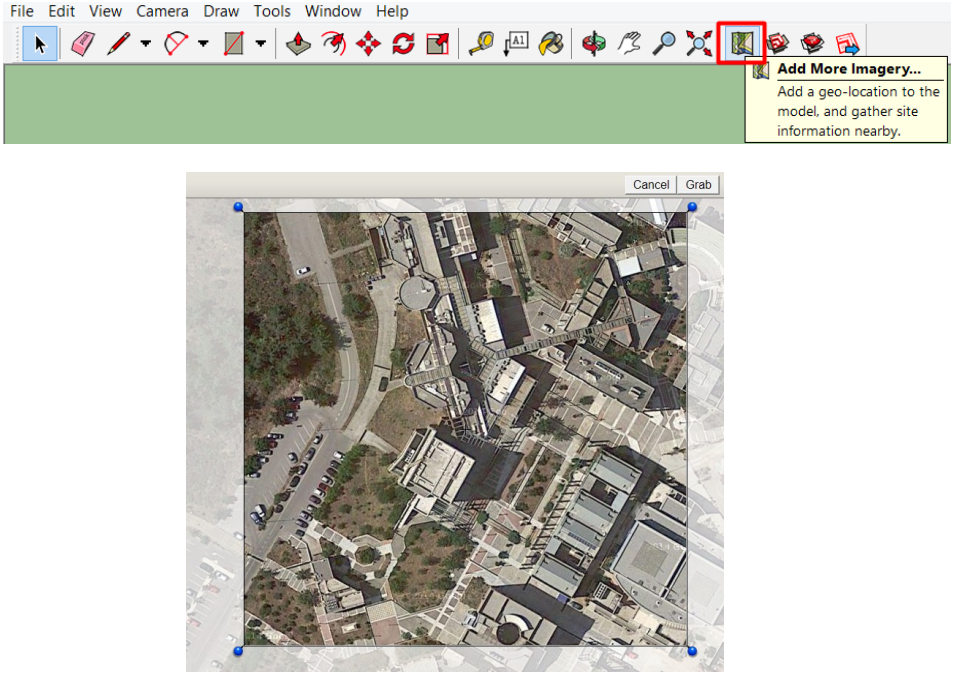
3.5 Sketchup map screenshot και απλά 3D μοντέλα

Θέλουμε να δημιουργήσουμε το χάρτη και τα κτίρια. Για τον αρχικό σχεδιασμό του χάρτη, αρκεί ένα απλό texture με τη φωτογραφία του πολυτεχνείου από δορυφόρο - μπορούμε να βρούμε τέτοια φωτογραφία από τους δωρεάν χάρτες στο διαδίκτυο - και για τα κτίρια μπορούμε να χρησιμοποιήσουμε μοντέλα χωρίς λεπτομέρειες και χρώμα.

3.5.1 Δημιουργία SketchUp Google maps Screenshot

Δυστυχώς αν προσπαθήσουμε να πάρουμε ένα απλό screenshot ενώ χρησιμοποιούμε τους χάρτες στον υπολογιστή μας, η ανάλυση της εικόνας δε μας ικανοποιεί. Καθώς αν εφαρμόσουμε το texture πάνω στο λευκό terrain, παραμορφώνεται και δε διακρίνονται καθαρά οι δρόμοι και τα κτίρια της φωτογραφίας.

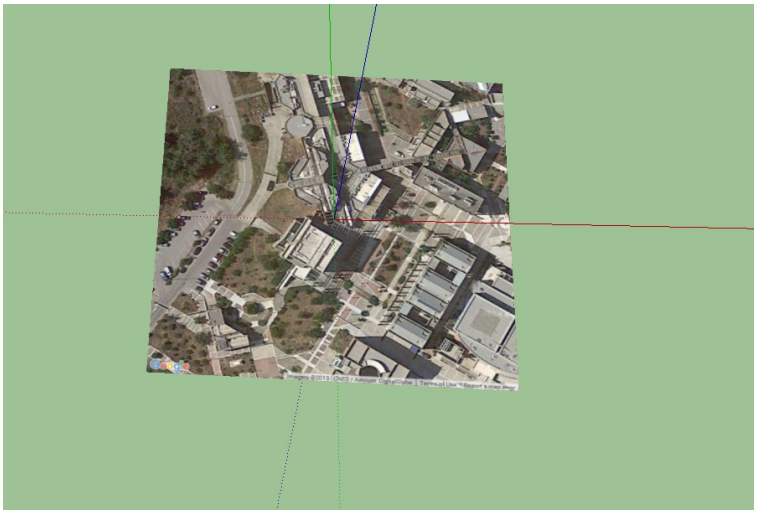
Μπορούμε όμως να χρησιμοποιήσουμε τη δυνατότητα λήψης screenshot από τους Google Maps την οποία προσφέρει το SketchUp.



Εικόνα 3.5: Λήψη screenshot

Πατώντας στο κουμπί για να ανοίξουν οι χάρτες, μας δίνεται η δυνατότητα περιήγησης στους χάρτες Google και επιλογής ενός τμήματος του χάρτη ώστε να το εισάγουμε σαν texture στο project μας στο SketchUp. Το πλεονέκτημα είναι ότι δε χρειάζεται να επιλέξουμε κατευθείαν ολόκληρο το κομμάτι του χάρτη που θέλουμε να χρησιμοποιήσουμε, μπορούμε αντίθετα να μεγεθύνουμε το χάρτη και να επιλέξουμε μικρότερο τμήμα του χάρτη, έχοντας έτσι μεγαλύτερη ανάλυση (Εικόνα 3.5).

Μόλις το κάνουμε αυτό το κομμάτι που επιλέξαμε θα εισαχθεί αυτόματα στο project μας σαν εικόνα (Εικόνα 3.6).



Εικόνα 3.6: Προβολή screenshot

Τώρα αρκεί να ακολουθήσουμε την ίδια διαδικασία και το SketchUp θα τοποθετήσει κάθε screenshot στη κατάλληλη θέση (Εικόνα 3.7). Δημιουργώντας έτσι ένα texture υψηλής ανάλυσης το οποίο θα μπορέσουμε να εξάγουμε σαν αρχείο fbx και να χρησιμοποιήσουμε στο Unity3D για το σχεδιασμό του χάρτη.

Εικόνα 3.7: Συναρμολόγηση screenshot

3.5.2 Δημιουργία αρχικών κτιρίων

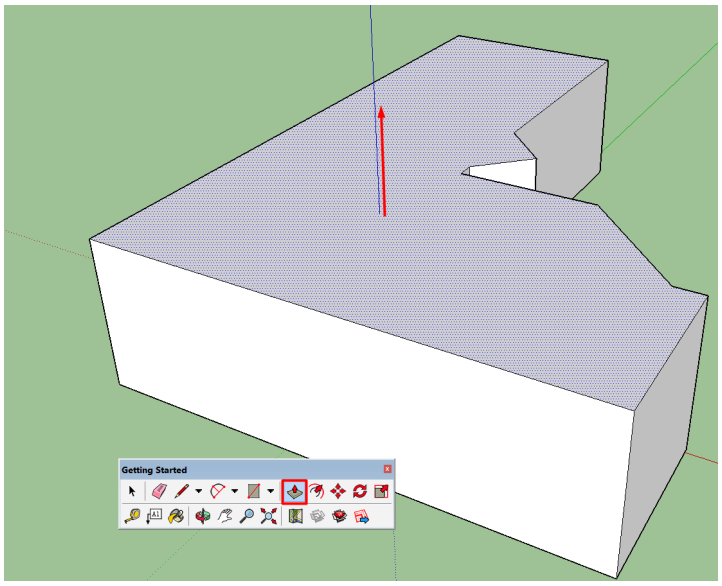
Το επόμενο βήμα είναι ο σχεδιασμός των αρχικών μοντέλων των κτιρίων. Αρχικά μας αρκούν απλά μοντέλα, χωρίς ιδιαίτερες λεπτομέρειες και χρωματισμό. Ο σχεδιασμός ενός απλού, "τετραγωνισμένου" μοντέλου είναι σχετικά απλός με τη χρήση SketchUp χάρη στη δυνατότητα Push/Pull επιφανειών που προσφέρει.

Αρκεί να σχεδιάσουμε μια μόνο επιφάνεια ώστε να δημιουργήσουμε από αυτή ένα απλό τρισδιάστατο μοντέλο. Για να σχεδιάσουμε αυτή την επιφάνεια αρκεί να χρησιμοποιήσουμε ένα από τα περιγράμματα κτιρίων που υπάρχουν σε ένα από τα αρχεία adobe illustrator, της μελέτης σήμανσης την οποία μας παρείχε η τεχνική υπηρεσία του πολυτεχνείου[17].

Για τα νέα κτίρια τα οποία δε περιλαμβάνονται στη μελέτη σήμανσης θα χρειαστεί να σχεδιάσουμε μόνοι μας το περίγραμμα χρησιμοποιώντας τη λειτουργία του μολυβιού του SketchUp.

Εικόνα 3.8: Δημιουργία επιφάνειας από περίγραμμα κτιρίου

Μόλις σχεδιάσουμε το περίγραμμα του κτιρίου στο SketchUp δημιουργείται αυτόματα η πρώτη μας επιφάνεια (Εικόνα 3.8). Χρησιμοποιώντας αυτή την επιφάνεια και το εργαλείο ανόρθωσης επιπέδων του SketchUp μπορούμε να δώσουμε όγκο στο σχήμα μας και να δημιουργήσουμε το πρώτο μας τρισδιάστατο μοντέλο (Εικόνα 3.9). Προσέχουμε η αρχή των αξόνων να βρίσκεται στο κέντρο του σχήματος μας, καθώς αυτό το σημείο θα είναι οι συντεταγμένες του μοντέλου μας στο χώρο, όταν αυτό ενσωματωθεί στην εφαρμογή μας και θέλουμε οι συντεταγμένες του αντικειμένου να “δείχνουν” το κέντρο του.



Εικόνα 3.9: Δημιουργία απλού κτιρίου

Μόλις το μοντέλο μας είναι έτοιμο, το μόνο που χρειάζεται να κάνουμε είναι να το εξάγουμε σε αρχείο με επέκταση .fbx το οποίο είναι μια από τις πολλές μορφές αρχείων τρισδιάστατων μοντέλων τα οποία υποστηρίζει το Unity.

3.5.3 Ενσωμάτωση στην εφαρμογή

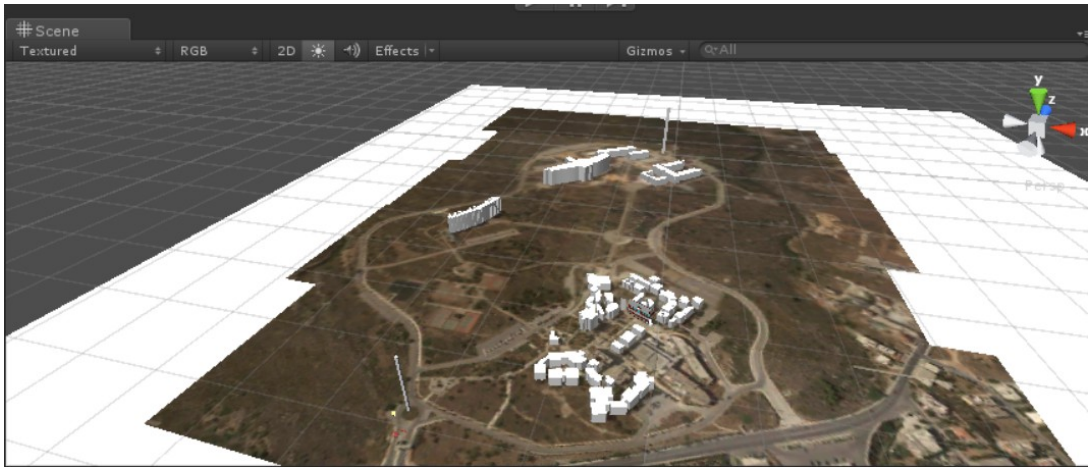
Τώρα που έχουμε μια εικόνα για να χρησιμοποιήσουμε σαν οδηγό ώστε να σχεδιάσουμε το terrain και τρισδιάστατα μοντέλα για τα κτίρια μας, μπορούμε να δημιουργήσουμε την αρχική μακέτα της εφαρμογής στο Unity.

Για να χρησιμοποιήσουμε την εικόνα του χάρτη και τα μοντέλα των κτιρίων στο Unity project μας, το μόνο που χρειάζεται να κάνουμε είναι να αποθηκεύσουμε αυτά τα αρχεία στο φάκελο Assets ο οποίος βρίσκεται στο project workspace του Unity.

Όλα τα αρχεία που βρίσκονται σε αυτό το φάκελο εμφανίζονται στη καρτέλα Project της διεπαφής του Unity. Αυτή είναι μια αρκετά χρήσιμη δυνατότητα καθώς έτσι όχι μόνο μπορούμε να προσθέσουμε ένα αντικείμενο κατευθείαν στη σκηνή μας απλώς σέρνοντας το και τοποθετώντας το εκεί που θέλουμε να βρίσκεται, αλλά μπορούμε ακόμα και να επεξεργαστούμε ένα από αυτά τα αρχεία, όπως για παράδειγμα το αρχείο ενός τρισδιάστατου μοντέλου, εφαρμόζοντας οποιαδήποτε αλλαγή κατευθείαν σε κάθε αντίγραφο του συγκεκριμένου αρχείου το οποίο βρίσκεται στη σκηνή μας.

Αρκεί λοιπόν να σύρουμε το αρχείο fbx του χάρτη στη σκηνή μας και θα δημιουργηθεί αυτόματα ένα αντίγραφο του στη σκηνή μας. Όπως με κάθε αντικείμενο, μπορούμε να το επιλέξουμε είτε από τη σκηνή, είτε κάνοντας κλικ στο όνομα του στη καρτέλα Hierarchy του Unity και να επεξεργαστούμε τις ιδιότητες του. Μετασχηματίζουμε λοιπόν το αντικείμενο επιλέγοντας το και χρησιμοποιώντας τη καρτέλα Inspector, όπως κάναμε και για τη ρύθμιση της κάμερας, ώστε η εικόνα του χάρτη να καλύπτει το terrain μας και έχουμε το χάρτη του πολυτεχνείου πάνω στον οποίο μπορούμε να εργαστούμε ώστε να χτίσουμε τη τρισδιάστατη σκηνή μας.

Επόμενος στόχος μας είναι να προσθέσουμε στη σκηνή τα μοντέλα των κτιρίων. Εργαζόμαστε με τον ίδιο τρόπο ώστε να τα προσθέσουμε στη σκηνή, να μετακινήσουμε το μοντέλο κάθε κτιρίου στη θέση του με βάση το χάρτη που τώρα έχουμε, να το περιστρέψουμε και να του αλλάξουμε το μέγεθος.



Εικόνα 3.10: Μακέτα σκηνής εφαρμογής

Παρατηρούμε ότι το Unity μας δίνει τη δυνατότητα εύκολου μετασχηματισμού τρισδιάστατων μοντέλων στη σκηνή, κάνοντας έτσι το σχεδιασμό τους αρκετά πιο εύκολο, καθώς δε χρειάζεται να μας απασχολούν οι διαστάσεις του μοντέλου από τη στιγμή που εύκολα μπορούμε να τις αυξομειώσουμε μέσα από το Unity ώστε να συμβαδίζουν με τις απαιτήσεις του εκάστοτε project, αλλά μας ενδιαφέρουν μόνο οι αναλογίες όπως θα δούμε παρακάτω, στο σχεδιασμό λεπτομερειών των τρισδιάστατων μοντέλων.

Η αρχική μακέτα μας είναι έτοιμη και έχουμε μια εικόνα του πως θα είναι ο χάρτης της εφαρμογής και μέρος της γραφικής διεπαφής χρήστη, καθώς ο χρήστης θα μπορεί να αλληλεπιδράσει με το χάρτη και να περιηγηθεί σε αυτόν, αλλά και επιλέγοντας τρισδιάστατα μοντέλα για να πάρει πληροφορίες για αυτό το οποίο αντιπροσωπεύουν (Εικόνα 3.10).

3.6 Μενού γραφικής διεπαφής χρήστη

Τώρα αρκεί να σχεδιάσουμε και τα μενού της γραφικής διεπαφής, ώστε να έχουμε μια ολοκληρωμένη εικόνα της γραφικής διεπαφής. Τα μενού ο χρήστης θα μπορεί να τα ανοίξει είτε επιλέγοντας αντικείμενα στη σκηνή ή επιλέγοντας κουμπιά που θα βρει στην αρχική σελίδα της εφαρμογής, όπως το κουμπί αναζήτησης. Όλα τα κουμπιά στα μενού θα μπορούν να επιλεχθούν και αν δεν ανοίγουν άλλα μενού με πληροφορίες θα εκτελούν ενέργειες, όπως για παράδειγμα κλήση τηλεφώνου και αποστολή email.

3.6.1 Δυνατότητες

Πρώτο μας μέλημα είναι ο καθορισμός των διάφορων μενού που θα χρειαστούμε. Θέλουμε ο χρήστης εκτός από τη δυνατότητα άμεσης πρόσβασης στις πληροφορίες ενός κτιρίου επιλέγοντας το, να έχει γρήγορη πρόσβαση στις λίστες όλων των μαθημάτων, αιθουσών και προσωπικού. Θέλουμε ακόμα να μπορεί να αναζητήσει εύκολα οποιαδήποτε πληροφορία, να χρησιμοποιήσει το GPS αλλά και να εμφανίσει/εξαφανίσει τις δημοσιεύσεις άλλων χρηστών στο χάρτη καθώς και να έχει τη δυνατότητα γρήγορης δημοσίευσης του δικού του κειμένου στη θέση την οποία βρίσκεται μέσα στο πολυτεχνείο. Επίσης θα πρέπει να έχει εύκολη πρόσβαση στο μενού ρυθμίσεων.

Για αυτό το λόγο θα χρειαστεί να δημιουργήσουμε τα κατάλληλα κουμπιά για να ανοίγουν τα αντίστοιχα μενού, και τα οποία μαζί με το τρισδιάστατο χάρτη του πολυτεχνείου θα αποτελούν το κύριο μενού της εφαρμογής μας.

Τα μενού που θα χρειαστούμε είναι τα εξής:

- Μενού κτιρίου

  Εμφανίζει τις αίθουσες ενός κτιρίου και τα μαθήματα τα οποία γίνονται σε κάθε αίθουσα, αν υπάρχει τέτοια πληροφορία διαθέσιμη.

- Μενού λίστας όλων των μαθημάτων

  Εμφανίζει όλα τα μαθήματα και την αίθουσα στην οποία παραδίδονται αυτή τη στιγμή, αν υπάρχει τέτοια πληροφορία διαθέσιμη.

- Μενού μαθήματος

  Εμφανίζει πληροφορίες για το επιλεγμένο μάθημα.

- Μενού λίστας όλων των αιθουσών

  Εμφανίζει όλες τις αίθουσες και τα μαθήματα τα οποία παραδίδονται αυτή τη στιγμή, αν υπάρχει τέτοια πληροφορία διαθέσιμη.

- Μενού αίθουσας

  Εμφανίζει πληροφορίες για την επιλεγμένη αίθουσα.

- Μενού λίστας του προσωπικού του πολυτεχνείου

  Εμφανίζει όλο το προσωπικό του πολυτεχνείου.

- Μενού προσώπου

  Εμφανίζει πληροφορίες για τον υπάλληλο του πολυτεχνείου ο οποίος επιλέχθηκε.

- Μενού αποτελεσμάτων αναζήτησης

  Εμφανίζει μια λίστα με τα αποτελέσματα της αναζήτησης.

- Μενού δημιουργίας δημοσίευσης

  Ο χρήστης θα μπορεί να γράψει και να δημοσιεύσει ένα κείμενο το οποίο θα είναι ορατό στους υπόλοιπους χρήστες.

- Μενού για τη προβολή δημοσίευσης

  Εμφανίζει τη δημοσίευση ενός άλλου χρήστη, τα σχόλια τρίτων, αλλά και θα επιτρέπει την απάντηση ή τη ψήφιση της δημοσίευσης.

- Μενού ρυθμίσεων

  Προσφέρει στο χρήστη τη δυνατότητα εγγραφής/σύνδεσης/αποσύνδεσης στο λογαριασμό του, αλλά και τη προβολή του μενού βοήθειας ή πληροφοριών σχετικά με την εφαρμογή.

- Μενού δημιουργίας λογαριασμού

  Δίνει τη δυνατότητα δημιουργίας λογαριασμού.

- Μενού σύνδεσης

  Επιτρέπει στο χρήστη να συνδεθεί στο λογαριασμό του.

- Μενού αποσύνδεσης

  Αποσύνδεση του λογαριασμού.

- Μενού βοήθειας

  Επεξήγηση των δυνατοτήτων της εφαρμογής.

- Μενού πληροφοριών εφαρμογής

  Προσφέρει πληροφορίες σχετικά με την εφαρμογή, και τις εκδόσεις της.

- Μενού εξόδου

  Επιβεβαιώνει την έξοδο από την εφαρμογή

- Μενού προειδοποίησης

  Δίνει πληροφορίες σχετικά με το αποτέλεσμα μιας ενέργειας.

- Μενού σφάλματος

  Εμφανίζει πληροφορίες σχετικά με κάποιο σφάλμα που μπορεί να προκύψει από μια ενέργεια.

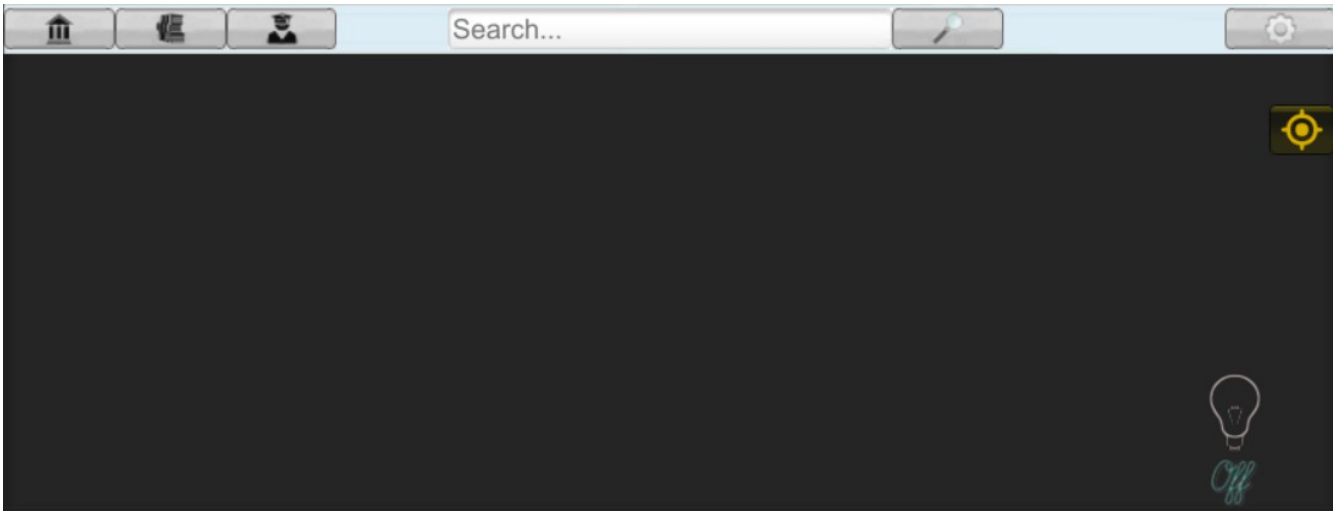
3.6.2 Σχεδιασμός μενού

Το επόμενο βήμα, είναι ο σχεδιασμός των μενού. Για τη δημιουργία εικόνων φόντου των μενού, των κουμπιών και άλλων στοιχείων του GUI έγινε χρήση του προγράμματος GIMP. Οι εικόνες που θα δημιουργήσουμε με αυτό το πρόγραμμά όπως και κάθε αρχείο εικόνας, μπορούν να χρησιμοποιηθούν από το Unity ώστε να δώσουν χρώμα στα μενού μας.

Πιο πολύπλοκες εικόνες, όπως οι εικόνες για τα κουμπιά εμφάνισης των λιστών μαθημάτων, αιθουσών και προσώπων, αλλά και των κουμπιών δημιουργίας δημοσίευσης και εμφάνισης δημοσιεύσεων όπως και των κουμπιών προβολής της ιστοσελίδας ή βίντεο ενός μαθήματος έγινε χρήση δωρεάν εικόνων που υπάρχουν στο διαδίκτυο. Έχοντας λοιπόν τις απαραίτητες εικόνες για το σχεδιασμό των μενού μας μπορούμε να τα σχεδιάσουμε.

Όπως προαναφέρθηκε το κύριο μενού της εφαρμογής μας θα αποτελείται από το τρισδιάστατο χάρτη με τα κτίρια, στον οποίο θα μπορεί ο χρήστης να περιηγηθεί αλλά και από μια συλλογή από κουμπιά τα οποία θα μας παρέχουν γρήγορη πρόσβαση στις βασικές δυνατότητες της εφαρμογής.

Για αυτό το λόγο είναι απαραίτητη μια μπάρα μενού η οποία θα είναι συνεχώς ορατή και θα βρίσκεται στο πάνω μέρος της οθόνης αλλά και κουμπιών τα οποία θα βρίσκονται σε σταθερό σημείο στην οθόνη του χρήστη (Εικόνα 3.11).



Εικόνα 3.11: Μενού αρχικής σελίδας

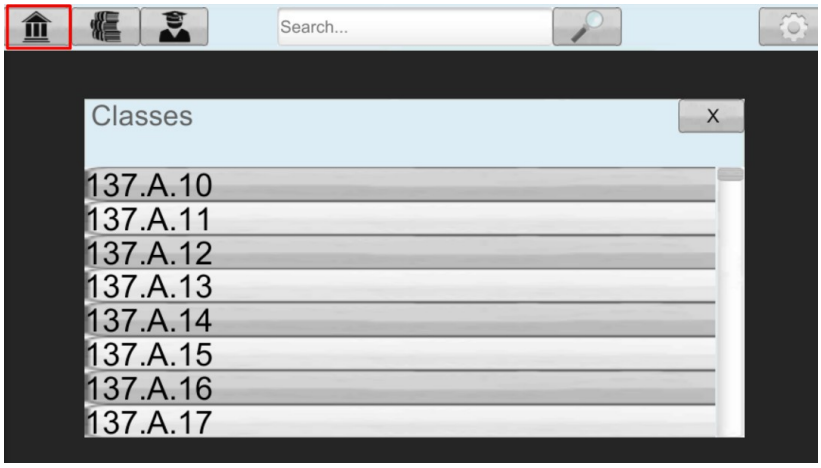
Στη παραπάνω εικόνα βλέπουμε τη πάνω μπάρα μενού, η οποία αποτελείται από τα κουμπιά για τη προβολή λίστας αιθουσών, μαθημάτων και προσωπικού, ένα πεδίο κειμένου και το κουμπί αναζήτησης και ένα κουμπί για το άνοιγμα του μενού ρυθμίσεων.

Κάτω από το κουμπί ρυθμίσεων, υπάρχει το κουμπί για τη λειτουργία του GPS. Στην εικόνα μας φαίνεται κίτρινο αλλά το χρώμα μπορεί να αλλάζει ανάλογα με τη κατάσταση λειτουργίας του GPS.

Στο κάτω δεξί μέρος της οθόνης βλέπουμε το κουμπί δημοσίευσης το οποίο έχει σαν εικονίδιο μια λάμπα η οποία επίσης μπορεί να αλλάζει χρώμα ανάλογα με τη δυνατότητα δημοσίευσης του χρήστη, και ακριβώς από κάτω παρατηρούμε το κουμπί On/Off το οποίο ενεργοποιεί και απενεργοποιεί τη προβολή δημοσιεύσεων άλλων χρηστών.

Τα παραπάνω σε συνδυασμό με το χάρτη και τα κτίρια αποτελούν το κύριο μενού της εφαρμογής μας και είναι πάντα ορατά. Η μόνη περίπτωση όπως θα δούμε αργότερα όπου τα κουμπιά τα οποία αφορούν τις λειτουργίες δημοσίευσης και προβολής δημοσιεύσεων και GPS δεν είναι ορατά, είναι όταν έχει ανοίξει κάποιο παράθυρο μενού.

Επιλέγοντας το κουμπί με το εικονίδιο του κτιρίου[3] που βρίσκεται στη πάνω αριστερά γωνία θα ανοίγει το μενού με τη λίστα αιθουσών (Εικόνα 3.12). Αν σύμφωνα με το πρόγραμμα γίνονται μαθήματα αυτή τη στιγμή στο πολυτεχνείο, δίπλα από το όνομα κάθε αίθουσας θα εμφανίζεται και το όνομα του μαθήματος το οποίο γίνεται στην αντίστοιχη αίθουσα.

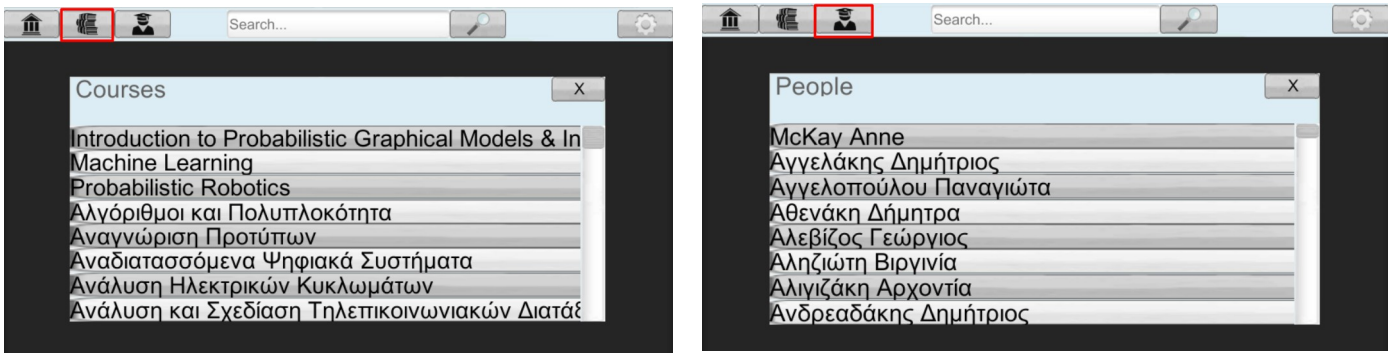


Εικόνα 3.12: Μενού αιθουσών

Επιλέγοντας μια αίθουσα ή ένα μάθημα θα ανοίγει το κατάλληλο μενού.

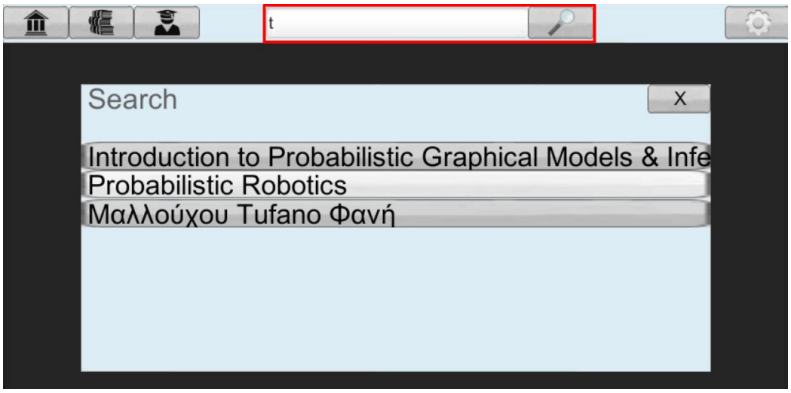
Ίδια εμφάνιση με το μενού λίστας αιθουσών θα έχει και το μενού κτιρίου, με τη μόνη διαφορά ότι θα εμφανίζονται μόνο οι αίθουσες που βρίσκονται στο επιλεγμένο κτίριο.

Με τον ίδιο τρόπο σχεδιάζουμε και το μενού λίστας μαθημάτων, το οποίο προβάλλει όλα τα μαθήματα και δίπλα στο κάθε μάθημα την αίθουσα στην οποία γίνεται αυτή τη στιγμή το μάθημα. Στο μενού λίστας καθηγητών η μόνη ορατή πληροφορία είναι τα ονόματα των καθηγητών (Εικόνα 3.13).



Εικόνα 3.13: Μενού μαθημάτων & μενού καθηγητών

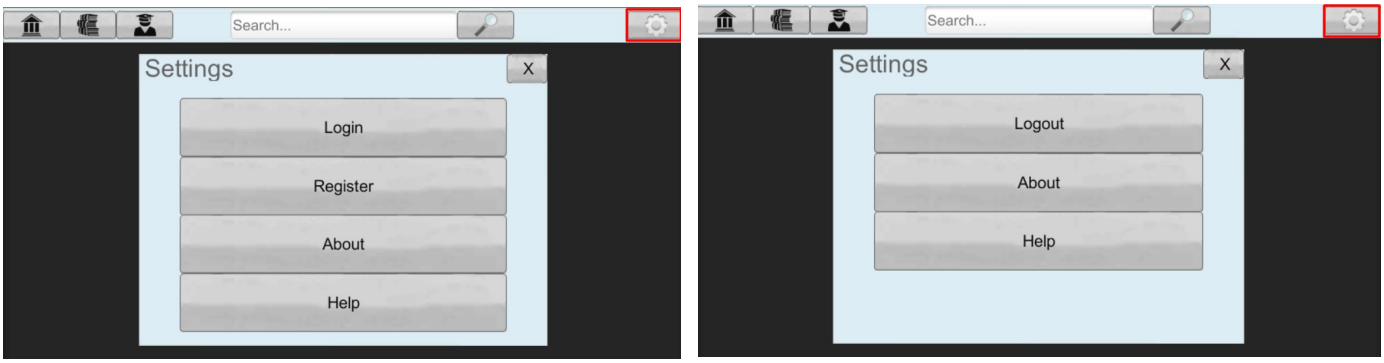
Όλες οι πληροφορίες σε αυτά τα μενού λιστών θα είναι κουμπιά τα οποία θα μπορούμε να επιλέξουμε ώστε να ανοίξει το μενού με τις αναλυτικές πληροφορίες για την επιλογή μας.



Εικόνα 3.14: Μενού αναζήτησης

Το ίδιο ισχύει και για το μενού αποτελεσμάτων αναζήτησης. Με τη βοήθεια της οποίας ο χρήστης θα μπορεί να αναζητήσει μαθήματα, αίθουσες ή πρόσωπα (Εικόνα 3.14).

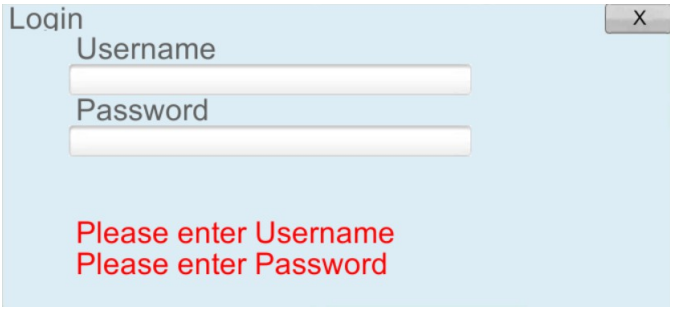
Το μενού των ρυθμίσεων θα διαφέρει, ανάλογα με το αν ο χρήστης είναι συνδεδεμένος στο λογαριασμό του ή όχι.



Εικόνα 3.15: Μενού ρυθμίσεων

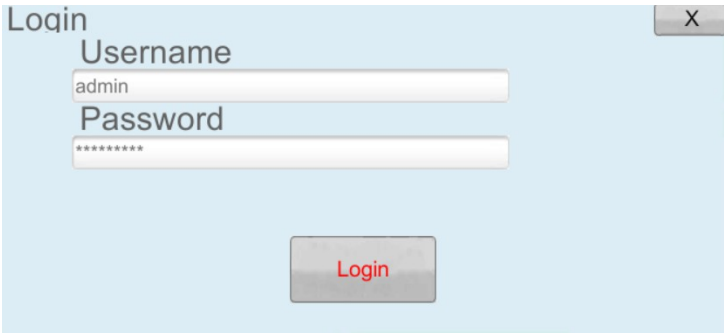
Από το μενού των ρυθμίσεων έχουμε πρόσβαση στα μενού τα οποία αφορούν τη διαχείριση του λογαριασμού του χρήστη. Ανάλογα με το αν ο χρήστης έχει συνδεθεί ή όχι στο λογαριασμό του θα εμφανίζονται τα κουμπιά σύνδεσης, εγγραφής ή αποσύνδεσης (Εικόνα 3.15).

Το μενού σύνδεσης, το οποίο παραπέμπει το χρήστη στην εισαγωγή ονόματος χρήστη και κωδικού. Οι έλεγχοι για το αν ο χρήστης έχει εισάγει τα στοιχεία του θα γίνονται επιτόπου και θα εμφανίζονται τα κατάλληλα μηνύματα με κόκκινα γράμματα αν δεν έχει εισάγει κάποιο απαραίτητο στοιχείο για τη σύνδεση του (Εικόνα 3.16).



Εικόνα 3.16: Μενού σύνδεσης χωρίς στοιχεία

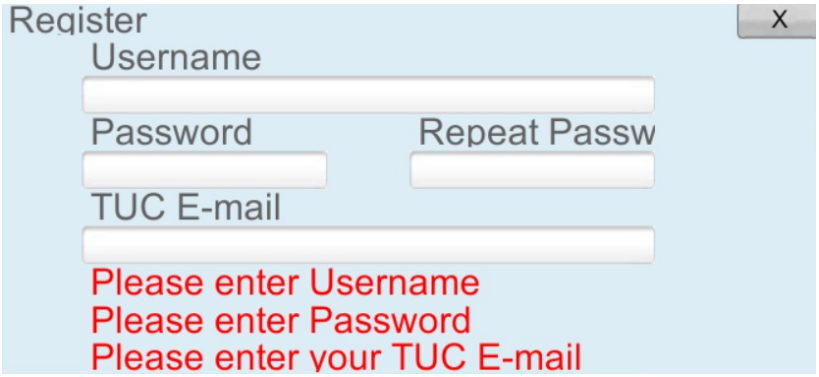
Όταν ο χρήστης συμπληρώσει τα στοιχεία του τα μηνύματα οδηγιών που εμφανίζονται με κόκκινα γράμματα θα εξαφανίζονται και τη θέση τους θα παίρνει ένα κουμπί το οποίο θα πραγματοποιεί τη σύνδεση (Εικόνα 3.17).



Εικόνα 3.17: Μενού σύνδεσης με στοιχεία

Με τον ίδιο τρόπο θα λειτουργεί και το μενού εγγραφής, στο οποίο ο χρήστης θα μπορεί να συμπληρώσει τα απαιτούμενα στοιχεία για τη δημιουργία νέου λογαριασμού (Εικόνα 3.18).

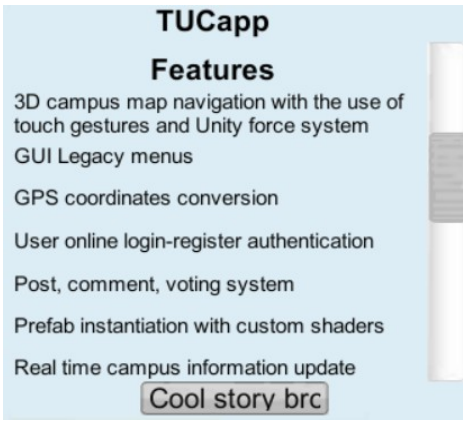
Στο μενού θα εμφανίζονται αντίστοιχα μηνύματα οδηγιών, όπως για παράδειγμα ότι "οι κωδικοί που έχει εισάγει ο χρήστης δε ταιριάζουν". Μόλις η φόρμα είναι συμπληρωμένη σωστά, τα μηνύματα οδηγιών θα αντικαθίστανται με ένα κουμπί για τη πραγματοποίηση της εγγραφής.



Εικόνα 3.18: Μενού εγγραφής

Το μενού αποσύνδεσης όπως και το μενού εξόδου από την εφαρμογή, είναι ένα απλό μενού επιβεβαίωσης της ενέργειας, με δυο κουμπιά (Εικόνα 3.19).

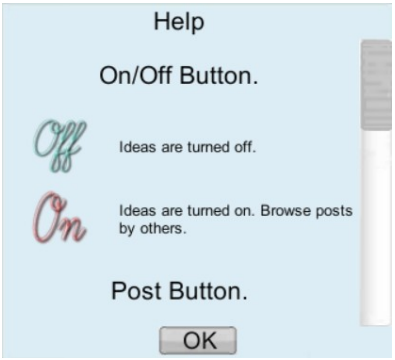
Εικόνα 3.19: Μενού αποσύνδεσης



Εικόνα 3.20: Μενού πληροφοριών

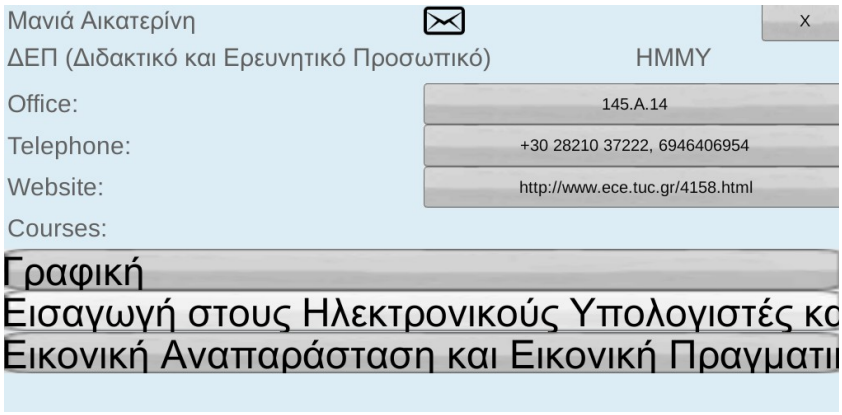
Από το μενού των ρυθμίσεων ο χρήστης έχει πρόσβαση και στο μενού πληροφοριών της εφαρμογής, στο οποίο θα μπορεί να βρει πληροφορίες σχετικά με τις δυνατότητες της εφαρμογής, τις τεχνολογίες που χρησιμοποιήθηκαν και τις αλλαγές που ίσως γίνουν σε μελλοντικές εκδόσεις (Εικόνα 3.20).

Ακόμα, στις ρυθμίσεις θα υπάρχει και το κατάλληλο κουμπί ώστε να μπορεί να ανοίξει το μενού βοήθειας της εφαρμογής το οποίο θα δίνει πληροφορίες για τις δυνατότητες της εφαρμογής και θα επεξηγεί τη λειτουργία της διεπαφής (Εικόνα 3.21).



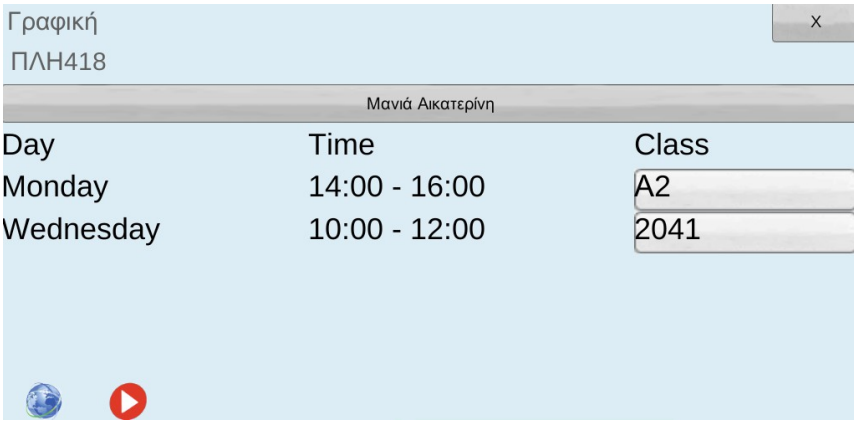
Εικόνα 3.21: Μενού βοήθειας

Επιλέγοντας έναν υπάλληλο του πολυτεχνείου, θα ανοίγει ένα μενού το οποίο θα δίνει τις διαθέσιμες πληροφορίες στο χρήστη. Κάνοντας κλικ στο κουμπί με το όνομα της αίθουσας ή ενός μαθήματος, θα ανοίγει το αντίστοιχο μενού. Επιλέγοντας το κουμπί του τηλεφώνου, της ιστοσελίδας ή το εικονίδιο με το φάκελο, ο χρήστης θα μπορεί να πραγματοποιήσει απευθείας τηλεφωνική κλήση, να επισκεφθεί το website, ή να στείλει email στο πρόσωπο της επιλογής του (Εικόνα 3.22).



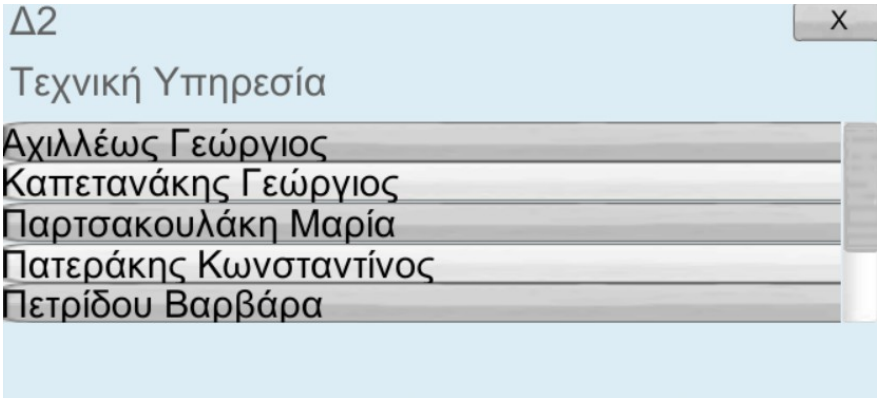
Εικόνα 3.22: Μενού προσώπου

Επιλέγοντας ένα μάθημα, θα ανοίγει το μενού του μαθήματος προσφέροντας στο χρήστη πληροφορίες όπως το ποιος ή ποιοι καθηγητές κάνουν το μάθημα αλλά και το πρόγραμμα του μαθήματος, με τη μορφή κουμπιών τα οποία θα ανοίγουν τα αντίστοιχα μενού. Στο κάτω μέρος του παραθύρου θα υπάρχουν κουμπιά με εξωτερικούς συνδέσμους προς την ιστοσελίδα του μαθήματος και βίντεο διαλέξεων (Εικόνα 3.23).



Εικόνα 3.23: Μενού μαθήματος

Επιλέγοντας μια αίθουσα θα εμφανίζεται το όνομα του υπαλλήλου του οποίου γραφείο είναι αυτή η αίθουσα. Αν μια αίθουσα είναι γραφείο περισσότερων από ένα ατόμων, τότε θα εμφανίζονται όλα τα ονόματα με αλφαβητική σειρά (Εικόνα 3.24).



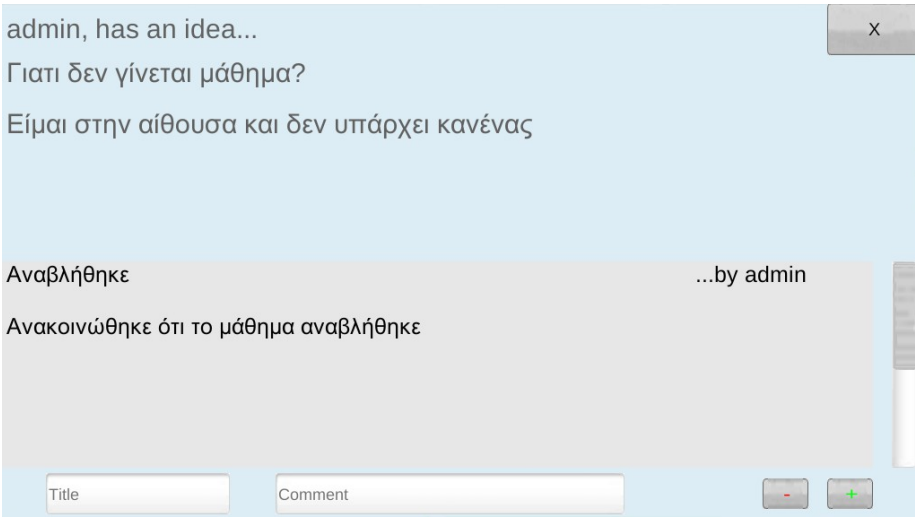
Εικόνα 3.24: Μενού αίθουσας

Θα μας χρειαστούν ακόμα δυο μενού για τη δυνατότητα δημοσίευσης των χρηστών.

Το ένα μενού αφορά τη δημοσίευση ενός μηνύματος από τον συνδεδεμένο χρήστη. Ο χρήστης θα έχει τη δυνατότητα επιλέγοντας το εικονίδιο με τη λάμπα που θα βρίσκεται στο κάτω δεξί μέρος της οθόνης να δημοσιεύσει ένα μήνυμα. Θα ανοίγει το παράθυρο δημοσίευσης όπου ο χρήστης θα μπορεί να συμπληρώσει ένα τίτλο και το μήνυμα του και να το δημοσιεύσει στη τρέχουσα θέση του στο πολυτεχνείο (Εικόνα 3.25).

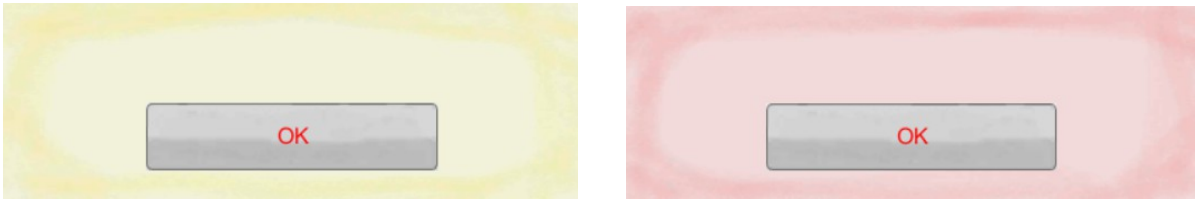
Εικόνα 3.25: Μενού δημοσίευσης μηνύματος

Για να εμφανίσει όλες τις ενεργές δημοσιεύσεις, ο χρήστης θα πρέπει να επιλέξει το κουμπί ON/OFF το οποίο βρίσκεται κάτω από το εικονίδιο δημοσίευσης (εικονίδιο λάμπας). Μόλις ο χρήστης επιλέξει το κουμπί ON/OFF θα εμφανιστούν όλες οι ενεργές δημοσιεύσεις στο χάρτη με τη μορφή τρισδιάστατων μοντέλων, των οποίων την υλοποίηση θα περιγράψουμε στο επόμενο κεφάλαιο. Επιλέγοντας ένα τρισδιάστατο μοντέλο, θα ανοίγει το αντίστοιχο μενού για την επιλεγμένη δημοσίευση όπου ο χρήστης θα μπορεί να διαβάσει την αρχική δημοσίευση, τις απαντήσεις άλλων χρηστών, αλλά και να απαντήσει ή να ψηφίσει με + ή – τη συγκεκριμένη δημοσίευση (Εικόνα 3.26).



Εικόνα 3.26: Μενού σχολιασμού μηνύματος

Τέλος σχεδιάζουμε δυο απλά μενού αποτελούμενα από μια λεζάντα κειμένου και ένα κουμπί επιβεβαίωσης. Αυτά τα μενού θα μας χρησιμεύσουν στο να εμφανίζουμε ειδοποιήσεις ή τυχόν σφάλματα στο χρήστη. Το μενού με το κόκκινο φόντο είναι το μενού σφάλματος, ενώ το μενού με το κίτρινο φόντο είναι το μενού προειδοποίησης (Εικόνα 3.27).



Εικόνα 3.27: Μενού προειδοποίησης & σφάλματος

Στις εικόνες μας η λεζάντα κειμένου είναι κενή και είναι ορατό μόνο το κουμπί OK. Στην εφαρμογή μας στο πάνω μισό παράθυρο των μενού θα εμφανίζεται το κατάλληλο κείμενο που θέλουμε να εμφανίσουμε στο χρήστη.

Το τελικό μας αποτέλεσμα θα είναι μια εναέρια περιήγηση πάνω από ένα τρισδιάστατο χάρτη του πολυτεχνείου, όπου ο χρήστης θα μπορεί να επιλέξει τα κτίρια για να αντλήσει περισσότερες πληροφορίες, να περιηγηθεί στα μενού αλλά και με τη χρήση του GPS να μπορεί να δημοσιεύει μηνύματα στη τρέχουσα θέση του. Για να εμφανιστούν δημοσιεύσεις άλλων χρηστών θα υπάρχει η δυνατότητα με τη χρήση του κουμπιού On/Off εμφάνισης των δημοσιεύσεων με τη μορφή γλόμπων. Ανάλογα τη βαθμολογία που έχει μια δημοσίευση από τους χρήστες της εφαρμογής θα αλλάζει το μέγεθος και το χρώμα του γλόμπου (Εικόνα 3.28).

Εικόνα 3.28: Επιθυμητό αποτέλεσμα

ΔΗΜΙΟΥΡΓΊΑ ΤΡΙΣΔΙΑΣΤΑΤΟΥ ΠΕΡΙΒΑΛΛΟΝΤΟΣ

ΚΕΦΑΛΑΙΟ 4

4.1 Τρισδιάστατη μοντελοποίηση

Σε αυτό το κεφάλαιο θα δούμε πως μπορούμε να σχεδιάσουμε και να χρησιμοποιήσουμε στο Unity πολύπλοκα τρισδιάστατα μοντέλα. Τις δυνατότητες που μας προσφέρει το terrain του Unity, και θα κάνουμε χρήση του φωτισμού ώστε να δημιουργήσουμε ένα όμορφο οπτικά αποτέλεσμα, με σκιές και αντανακλάσεις φωτός.

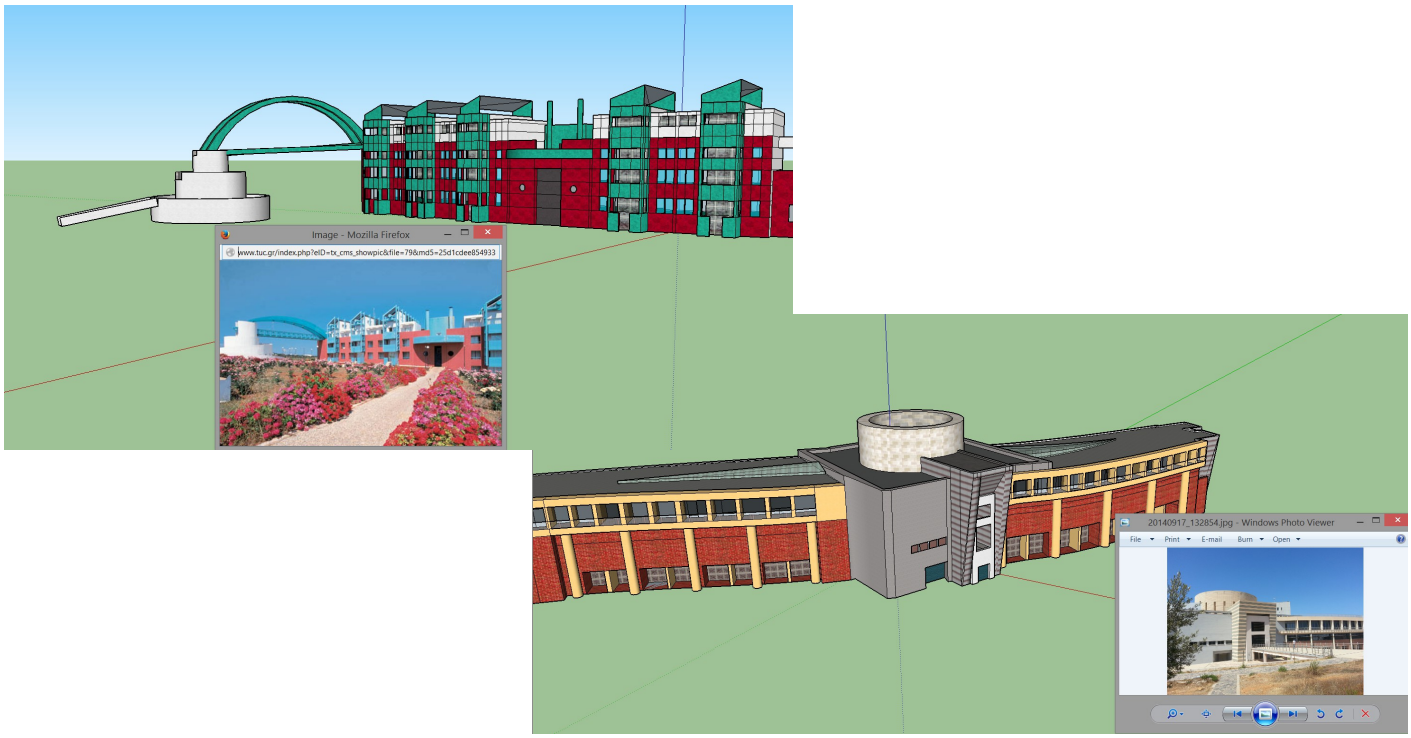
4.1.1 Δημιουργία πολύπλοκων μοντέλων κτιρίων με τη χρήση SketchUp

Είδαμε πως μπορούμε να σχεδιάσουμε απλά τρισδιάστατα μοντέλα με τη χρήση SketchUp. Φυσικά αν και έχουμε μια εικόνα του χάρτη και των κτιρίων του πολυτεχνείου, θέλουμε τα κτίρια να είναι όσο το δυνατόν πιο ρεαλιστικά και να αντιπροσωπεύουν τα κτίρια του πολυτεχνείου. Αυτό σημαίνει ότι θα πρέπει όχι μόνο να σχεδιάσουμε τις λεπτομέρειες των κτιρίων, ώστε να μην είναι απλά κουτιά, αλλά και να προσθέσουμε σε αυτά χρώμα και υφή.

Όπως προαναφέρθηκε, λόγω της δυνατότητας παραμόρφωσης ενός τρισδιάστατου μοντέλου μέσω της διεπαφής του Unity, ώστε να πάρει τη μορφή και το μέγεθος που χρειάζεται, δε μας απασχολούν οι διαστάσεις του μοντέλου που θα σχεδιάσουμε στο SketchUp αλλά οι αναλογίες.

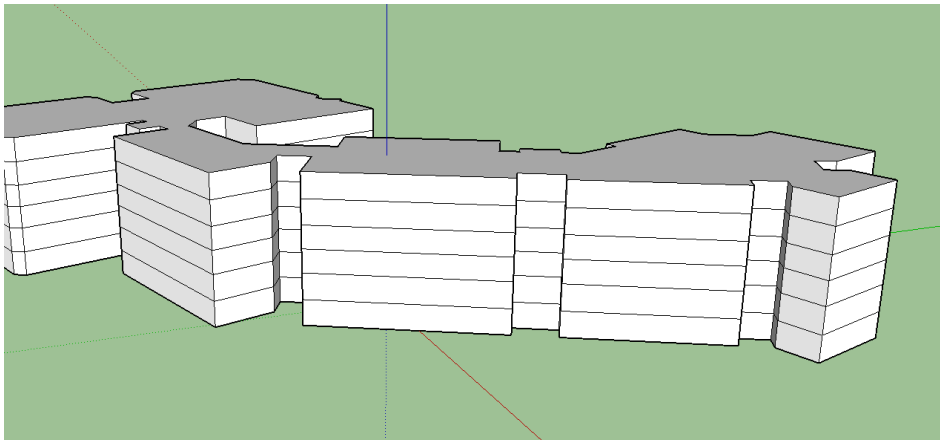
Αυτό σημαίνει ότι αν θέλουμε ένα μοντέλο στη σκηνή μας να έχει για παράδειγμα μέγιστο μήκος 10 μονάδες, πλάτος 20 και ύψος 30 αρκεί να σχεδιάσουμε στο SketchUp ένα τρισδιάστατο μοντέλο με μήκος 1 μονάδα, πλάτος 2 και ύψος 3, έτσι ώστε αν εφαρμόσουμε μετασχηματισμό που πολλαπλασιάζει επί 10 κάθε διάσταση του μοντέλου να μην έχουμε παραμορφώσεις, όπως για παράδειγμα μακρόστενα παράθυρα ή φαρδιές πόρτες. Αν και ακόμα και σε αυτή τη περίπτωση μπορούμε ακόμα και να μετασχηματίσουμε μόνο μια διάσταση μέσω της διεπαφής του Unity ώστε να μειώσουμε πιθανά φαινόμενα παραμορφώσεων.

Έχοντας ήδη τις σωστές αναλογίες μήκους και πλάτους, από τα περιγράμματα κτιρίων που πήραμε από τη μελέτη σήμανσης το μόνο που μας απασχολεί είναι το ύψος του κτιρίου. Ο σχεδιασμός όλων των κτιρίων έγινε με τη χρήση φωτογραφιών (Εικόνα 4.1).



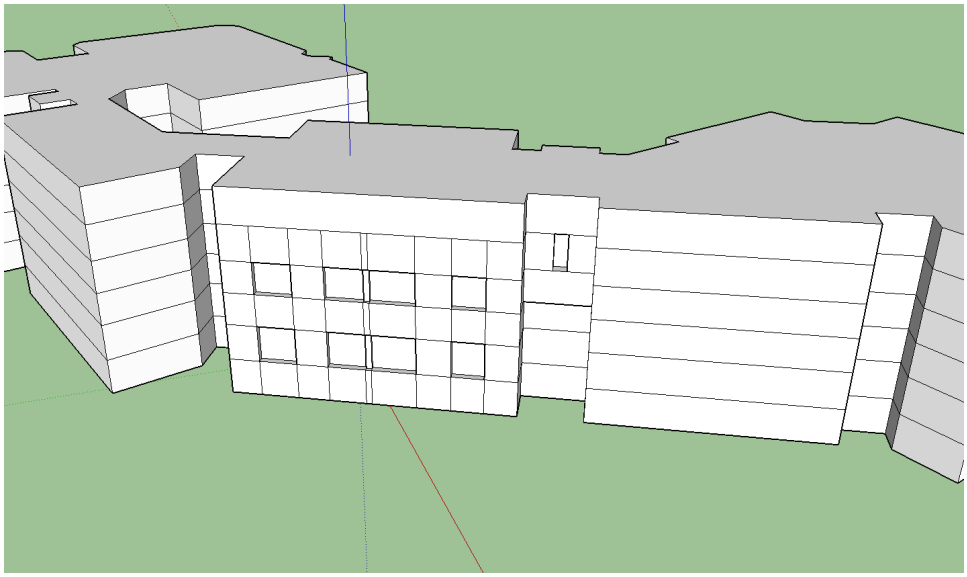
Εικόνα 4.1: Σύγκριση τρισδιάστατων μοντέλων με φωτογραφίες

Για να σχεδιάσουμε ένα κτίριο το χωρίζουμε σε επίπεδα (Εικόνα 4.2). Αυτό θα μας διευκολύνει ώστε να "σκαλίσουμε" τα παράθυρα και τις λεπτομέρειες αφού έτσι θα μπορούμε εύκολα τους ορόφους του κτιρίου. Για κάθε όροφο θα χρειαστούμε πάνω από ένα επίπεδα στο μοντέλο μας και αυτό γιατί έτσι, χρησιμοποιώντας τις οριζόντιες γραμμές που σχηματίζονται, θα μπορέσουμε να σχεδιάσουμε τα παράθυρα, τις πόρτες και τις υπόλοιπες εξωτερικές λεπτομέρειες κάθε κτιρίου. Το ύψος κάθε επιπέδου δε μας απασχολεί, προσέχουμε όμως το συνολικό ύψος του κτιρίου να μην είναι οπτικά δυσανάλογο με τις άλλες δυο διαστάσεις του και όλα τα επίπεδα να έχουν το ίδιο ύψος.



Εικόνα 4.2: Προσθήκη ορόφων στο μοντέλο

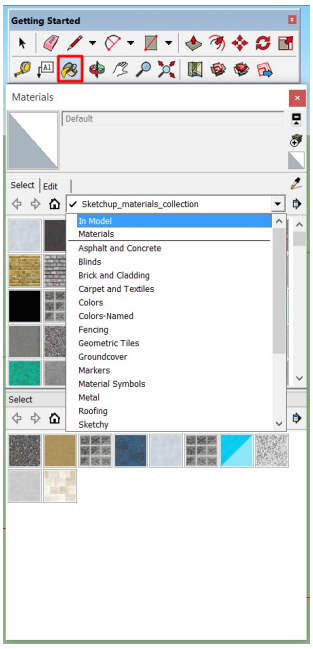
Τώρα που έχουμε χωρίσει το μοντέλο μας σε επίπεδα μπορούμε να σχεδιάσουμε τις λεπτομέρειες. Για να σχεδιάσουμε ένα παράθυρο φέρνουμε τις καθέτους ώστε να σχηματιστεί το παράθυρο και με το εργαλείο Push/Pull του δίνουμε βάθος (Εικόνα 4.3). Με τη χρήση των εργαλείων του SketchUp μπορούμε να σχεδιάσουμε κάθε λεπτομέρεια ενός κτιρίου.



Εικόνα 4.3: Προσθήκη εσοχών στο μοντέλο

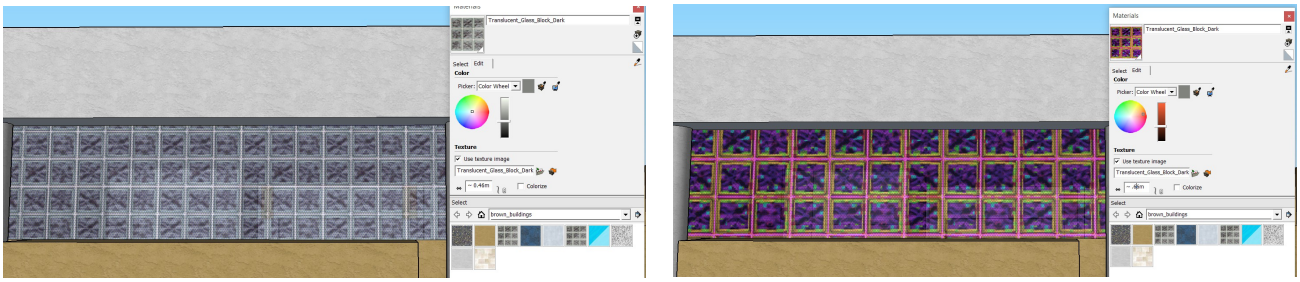
Ο λόγος που το κάνουμε αυτό είναι ώστε να σχηματίσουμε ανάγλυφες επιφάνειες στο μοντέλο μας οι οποίες θα δίνουν μια αίσθηση ρεαλισμού, βάθους και να δημιουργούν σκιές. Θα μπορούσαμε να χρησιμοποιήσουμε μόνο εικόνες για τα παράθυρα στα κτίρια μας χωρίς να τους δώσουμε βάθος, και με τη χρήση των κατάλληλων shaders να δώσουμε μια ψευδαίσθηση βάθους, όπως θα δούμε παρακάτω, αλλά εκτός του ότι έτσι δε θα δημιουργούνται σκιές από τις εσοχές του τρισδιάστατου μοντέλου, αν η κάμερα βλέπει υπό συγκεκριμένη γωνία το μοντέλο η ψευδαίσθηση ανάγλυφης επιφάνειας μπορεί να χαθεί. Μόλις το ανάγλυφο μοντέλο μας είναι έτοιμο, το μόνο που μένει να κάνουμε είναι να δώσουμε χρώμα στο κτίριο. Και για να γίνει αυτό θα χρησιμοποιήσουμε materials.

Το SketchUp προσφέρει μεγάλη ποικιλία από έτοιμα materials τα οποία μπορούμε να χρησιμοποιήσουμε στα μοντέλα μας. Μπορούμε να βρούμε materials για μεταλλικές επιφάνειες τα οποία όταν εισαχθούν στο Unity μέσω του τρισδιάστατου μοντέλου μας θα έχουν γυαλάδα, ή materials για τα τζάμια, τα οποία θα έχουν διαφάνεια. Για να θέσουμε ένα material σε μια επιφάνεια του μοντέλου που σχεδιάσαμε χρησιμοποιούμε τη δυνατότητα “Paint Bucket” του SketchUp με την οποία μπορούμε να χρωματίσουμε μια επιφάνεια με το material της επιλογής μας (Εικόνα 4.4).



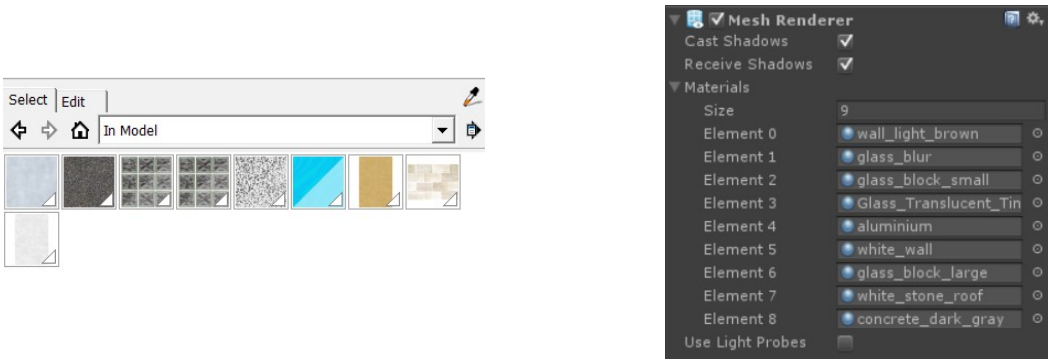
Εικόνα 4.4: Προσθήκη υλικών στο μοντέλο

Φυσικά μπορούμε να επεξεργαστούμε ένα material στο SketchUp ώστε να του αλλάξουμε το χρώμα, τις διαστάσεις ή ακόμα και να δημιουργήσουμε ένα καινούργιο από το texture της επιλογής μας (Εικόνα 4.5).



Εικόνα 4.5: Επεξεργασία υλικού

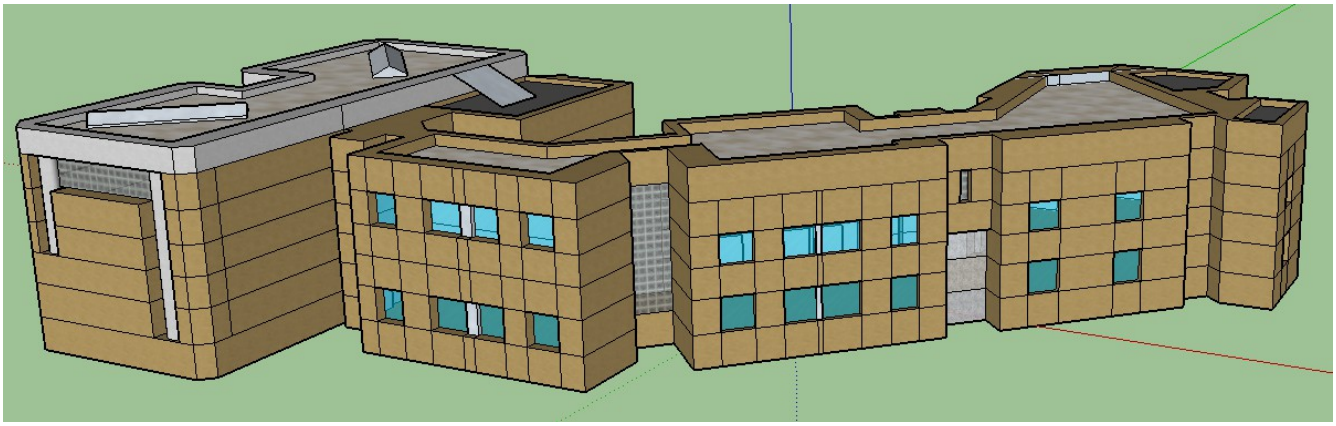
Μπορούμε να αλλάξουμε όλα τα materials ενός μοντέλου και μέσα από το Unity, το οποίο επίσης προσφέρει τη δυνατότητα δημιουργίας και επεξεργασίας materials. Πρέπει όμως να ορίσουμε materials για το μοντέλο μας κατά τη δημιουργία του με το SketchUp ακόμα και αν μετά θέλουμε να τα αλλάξουμε μέσω του Unity και αυτό ώστε το μοντέλο μας να χωριστεί σε ομάδες επιφανειών οι οποίες χρησιμοποιούν το ίδιο material. Έτσι αν αλλάξουμε το material των παραθύρων μέσω Unity θα αλλάξει αυτόματα σε όλες τις επιφάνειες οι οποίες χρησιμοποιούν το ίδιο material.



Εικόνα 4.6: Αντιστοίχιση υλικών

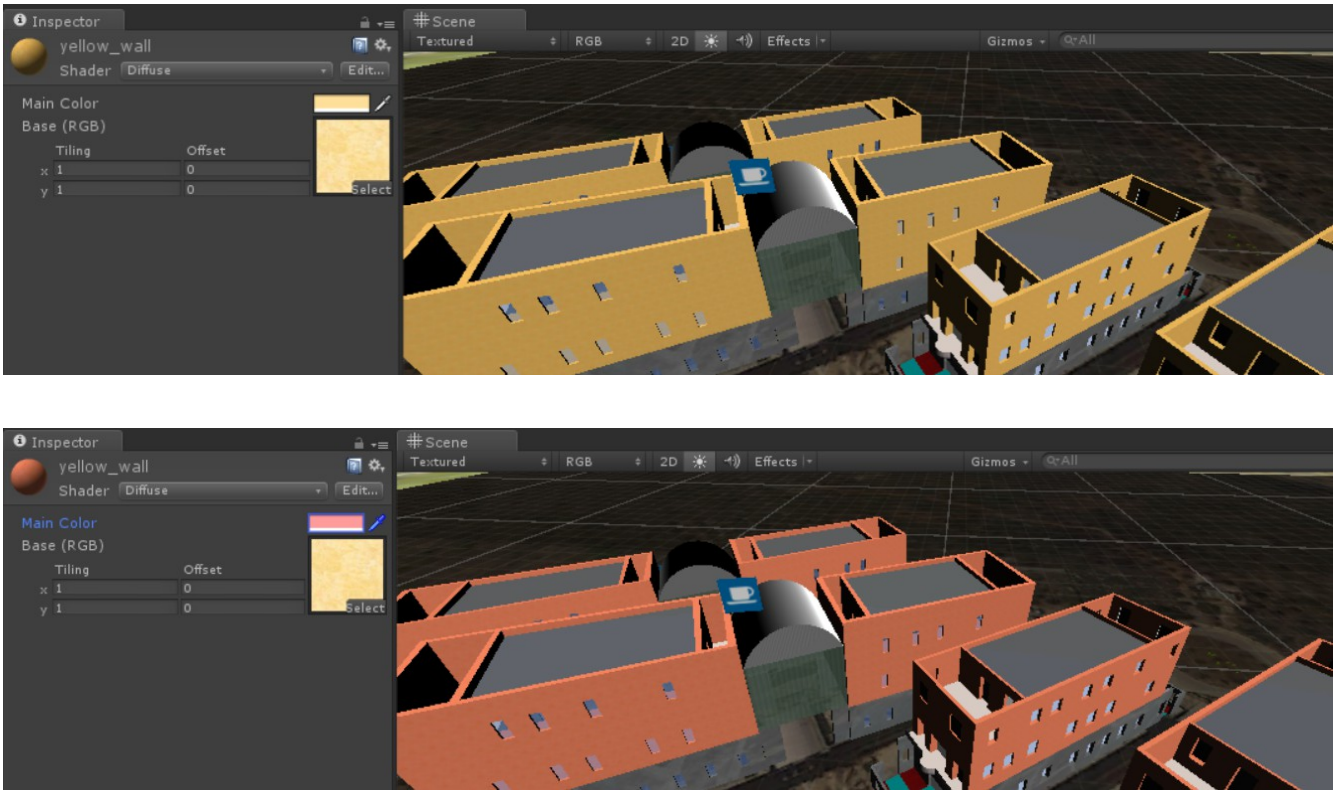
Στη πρώτη εικόνα βλέπουμε τα materials που χρησιμοποιούνται στο μοντέλο μας στο SketchUp, ενώ στη δεύτερη βλέπουμε τα ίδια materials να αντιστοιχίζονται στις ομάδες επιφανειών του μοντέλου μας στο Unity (Εικόνα 4.6). Αν δεν είχαμε ορίσει materials στο μοντέλο μας τότε αντί για 9 material elements θα είχαμε μόνο ένα και δε θα μπορούσαμε για παράδειγμα να ορίσουμε διαφορετικά materials για παράθυρα και τοίχους.

Όταν το τρισδιάστατο μοντέλο μας είναι έτοιμο (Εικόνα 4.7), και το έχουμε εμπλουτίσει με materials τότε αρκεί να το εξάγουμε σε αρχείο με κατάληξη fbx το οποίο μπορούμε να εισάγουμε στο Unity.



Εικόνα 4.7: Τελικό μοντέλο κτιρίου

Δε ξεχνάμε να αποθηκεύσουμε τα materials που χρησιμοποιήσαμε, μέσω SketchUp, ώστε να τα ξανά-χρησιμοποιήσουμε σε άλλα μοντέλα που ίσως έχουν ίδιο χρώμα τοίχων ή παραθύρων. Κάνοντας το αυτό και χρησιμοποιώντας τα ίδια materials και για άλλα μοντέλα, θα καταλήξουμε με διάφορα αρχεία fbx τα οποία θα εισάγουμε στο Unity και θα μοιράζονται κάποια κοινά materials. Αυτό μας διευκολύνει όχι μόνο στο ότι μπορούμε να επεξεργαστούμε το κοινό material και να αλλάξει το χρώμα στους τοίχους για παράδειγμα όλων των κτιρίων που το χρησιμοποιούν (Εικόνα 4.8) αλλά με αυτό το τρόπο η εφαρμογή μας χρησιμοποιεί λιγότερα assets καθώς δεν έχουμε πολλά αντίγραφα όμοιων materials.



Εικόνα 4.8: Επίδραση επεξεργασίας material

Το SketchUp είναι πρόγραμμα σχεδιασμού τρισδιάστατων μοντέλων για αρχιτέκτονες. Αυτό σημαίνει ότι ενδείκνυται για το σχεδιασμό μοντέλων κτιρίων ή κατασκευών με τα εργαλεία σχεδιασμού που παρέχει, αλλά δυσκολεύει το σχεδιασμό πιο πολύπλοκων μοντέλων, όπως για παράδειγμα το μοντέλο ενός ανθρώπου.

Το Unity μας δίνει τη δυνατότητα να εισάγουμε πολύπλοκα μοντέλα από άλλα προγράμματα τα οποία θα υποστηρίζουν και λειτουργίες κίνησης, όπως για παράδειγμα βάδισμα, αλλά μπορούμε να εισάγουμε και μοντέλα με σκελετό. Δυστυχώς το SketchUp δεν υποστηρίζει όλες αυτές τις δυνατότητες, αφού είναι περιττές στον αρχιτεκτονικό σχεδιασμό τρισδιάστατων μοντέλων.

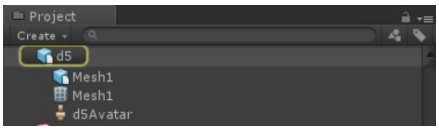
Τέτοιες δυνατότητες όμως δεν ήταν απαραίτητες στα πλαίσια αυτής της διπλωματικής, οπότε το SketchUp είναι το ιδανικό πρόγραμμα για το σχεδιασμό των μοντέλων μας, χάρη στην απλότητα που προσφέρει και τον εύκολο χειρισμό του.

4.1.2 Χρήση τρισδιάστατου μοντέλου

Έχοντας σχεδιάσει όλα τα τρισδιάστατα μοντέλα των κτιρίων και αφού τα εξάγουμε σε τύπο αρχείου fbx το μόνο που χρειάζεται να κάνουμε ώστε να εμφανιστούν στη σκηνή μας είναι να τα αντιγράψουμε στο φάκελο Assets του project μας στο Unity. Αφού υπάρχουν ήδη τα αρχικά μοντέλα των κτιρίων σε αυτό το φάκελο, αρκεί να δώσουμε το ίδιο όνομα στα αντίστοιχα αρχεία πολύπλοκων μοντέλων που μόλις σχεδιάσαμε, και να αντικαταστήσουμε τα παλιά μας μοντέλα. Μόλις το κάνουμε αυτό όχι μόνο θα αντικαταστήσουμε τα αρχεία μας στο φάκελο Assets με τα καινούργια, αλλά στη θέση των κουτιών που είχαμε στη σκηνή θα εμφανιστούν τα νέα μοντέλα.

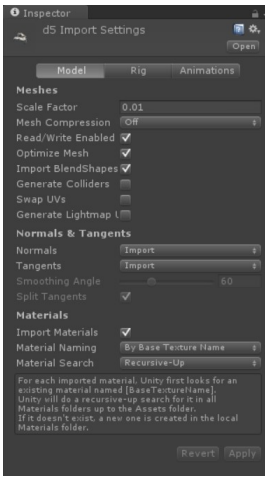
Οι μετασχηματισμοί που είχαμε εφαρμόσει στα παλιά μας μοντέλα στη σκηνή θα παραμείνουν στα νέα, άρα αν δεν έχουμε αλλάξει τις διαστάσεις του μοντέλου κατά το σχεδιασμό στο SketchUp, αλλά τις έχουμε κρατήσει ίδιες, τότε στη σκηνή μας, θα εμφανίζεται ήδη στο σωστό μέγεθος.Αν αυτό δεν ισχύει τότε μετασχηματίζουμε το μοντέλο εκ νέου στη σκηνή.

Κάνοντας κλικ στο μοντέλο μας, στη καρτέλα Project του Unity, μπορούμε να δούμε και να αλλάξουμε τις ρυθμίσεις του μοντέλου που έχουμε εισάγει (Εικόνα 4.9).



Εικόνα 4.9: Επιλογή prefab

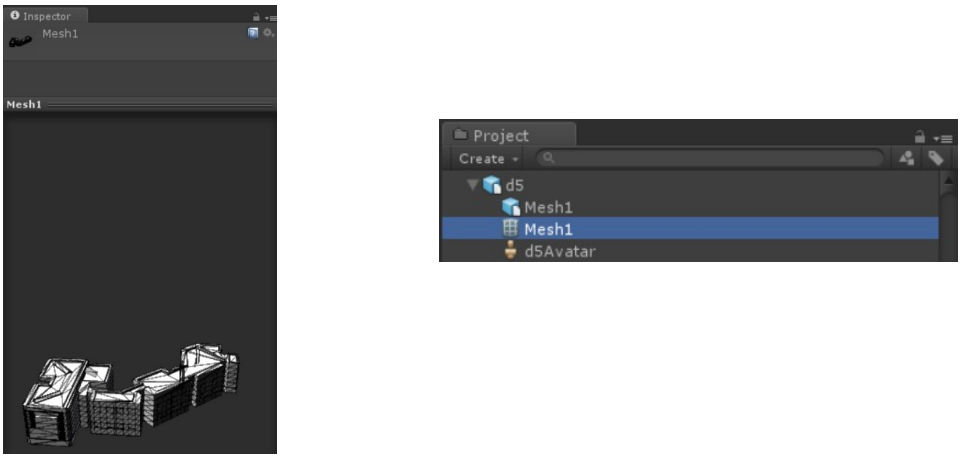
Για παράδειγμα μπορούμε να επιλέξουμε αν θα χρησιμοποιηθούν τα materials που εισάγαμε αλλά και που να ψάξει το unity για να τα βρει. Πολύ χρήσιμο αν θέλουμε να χρησιμοποιήσουμε δικά μας materials ή να χρησιμοποιήσουμε κοινά materials. Μπορούμε επίσης να επιλέξουμε αν οι normals και tangents θα εισαχθούν από το μοντέλο ή αν θα υπολογιστούν από τη μηχανή γραφικών του Unity ώστε να επιλέξουμε ακόμα και να απαλύνουμε τις γωνίες ώστε να βελτιώσουμε την απεικόνιση του μοντέλου (Εικόνα 4.10).



Εικόνα 4.10: Επεξεργασία prefab

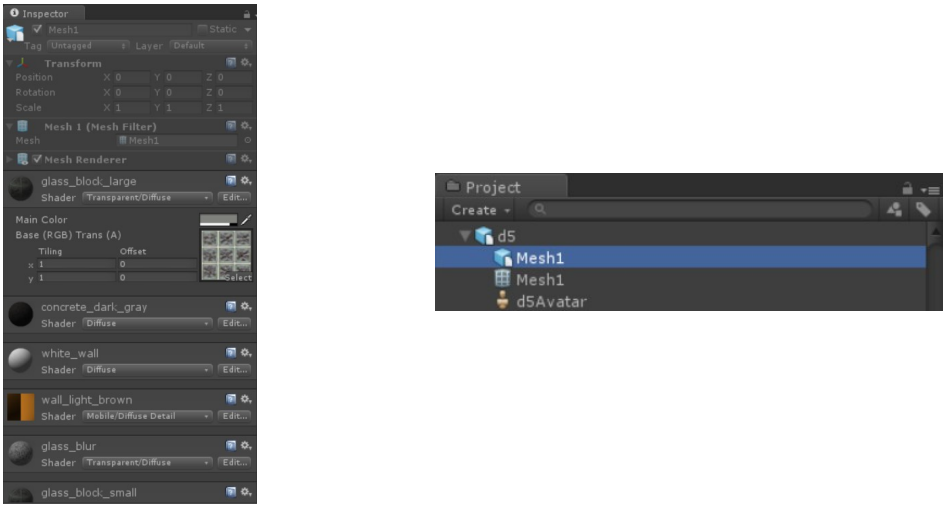
Για κάθε αντικείμενο τρισδιάστατου μοντέλου που εισάγουμε στο Unity, στη καρτέλα Project παρατηρούμε ότι εισάγονται και κάποια child objects, τα οποία εμφανίζονται κάνοντας κλικ στο βελάκι αριστερά από το αντικείμενο του μοντέλου μας. Δυο από αυτά ονομάζονται Mesh1 και το τρίτο, το οποίο δε θα μας απασχολήσει σε αυτή τη διπλωματική φέρει το όνομα του αρχείου του τρισδιάστατου μοντέλου μας ακολουθούμενο από τη λέξη Avatar.

Το δεύτερο child object με τίτλο Mesh1 και εικονίδιο πλέγματος, περιέχει πληροφορίες σχετικά με το πλέγμα πολυγώνων που σχηματίζουν το μοντέλο μας. Αυτό ονομάζεται Mesh Filter (Εικόνα 4.11).



Εικόνα 4.11: Mesh filter

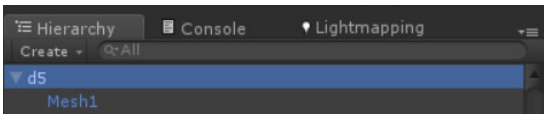
Το πρώτο child object με το ίδιο όνομα περιέχει πληροφορίες σχετικά με το πως θα εμφανίζεται το μοντέλο μας στη σκηνή. Αυτό ονομάζεται Mesh Renderer (Εικόνα 4.12). Ο Mesh Renderer παίρνει τη γεωμετρία του Mesh Filter, και με βάση τις πληροφορίες για το μετασχηματισμό του μοντέλου, τα materials και τους shaders που θα χρησιμοποιηθούν καθορίζει την αναπαράσταση του μοντέλου.



Εικόνα 4.12: Mesh renderer

Εδώ μπορούμε να αλλάξουμε την εμφάνιση του τρισδιάστατου μοντέλου, είτε αλλάζοντας τον shader ενός material, το texture, το χρώμα του ή όποια άλλη πληροφορία είναι διαθέσιμη.

Έχοντας δημιουργήσει ένα αντίγραφο του prefab μοντέλου μας στη σκηνή, μπορούμε να δούμε το όνομα του αντιγράφου στη καρτέλα Hierarchy και κάνοντας κλικ στο βελάκι αριστερά από το όνομα του μοντέλου, θα εμφανιστούν τα child objects που τα οποία έχουν εισαχθεί μαζί με το μοντέλο μας. Στη περίπτωση μας το μόνο child object που εισάγεται είναι ο mesh renderer του μοντέλου (Εικόνα 4.13).

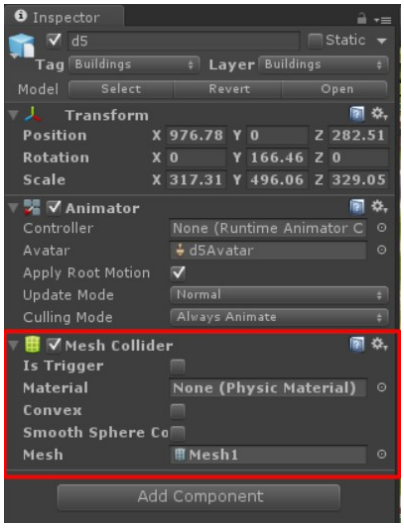


Εικόνα 4.13: Εισαγωγή prefab κτιρίου στη σκηνή

Το μπλε χρώμα ονόματος σημαίνει ότι το μοντέλο στη σκηνή είναι συνδεδεμένο με το prefab και όποια αλλαγή γίνει στο prefab, για παράδειγμα αν αλλάξουμε ένα από τα materials του mesh renderer, θα εφαρμοστεί αυτόματα στο μοντέλο στη σκηνή. Μπορούμε ανα πάσα στιγμή να σπάσουμε αυτή τη σύνδεση μεταξύ prefab και scene object ώστε οι αλλαγές σε ένα prefab να μην επηρεάζουν τα αντίγραφα στη σκηνή. Αν έχει σπάσει αυτή η σύνδεση τότε το όνομα του αντικειμένου της σκηνής θα εμφανίζεται με άσπρα γράμματα.

Έχοντας πλέον το μοντέλο στη σκηνή μας, το μόνο που μένει είναι να του δώσουμε υπόσταση. Δεν αρκεί μόνο να είναι ορατό αλλά πρέπει και να μπορεί να αλληλεπιδράσει με άλλα τρισδιάστατα μοντέλα, αλλά και να μπορούμε να το επιλέξουμε. Για αυτή τη δουλειά προσθέτουμε το component Mesh Collider (Εικόνα 4.14).

Σαν Mesh Collider μπορούμε είτε να χρησιμοποιήσουμε το ίδιο το Mesh Filter του μοντέλου μας ή να δημιουργήσουμε ένα πιο απλό με τη βοήθεια του Unity, για παράδειγμα ένα κύβο.



Εικόνα 4.14: Ανάθεση mesh collider

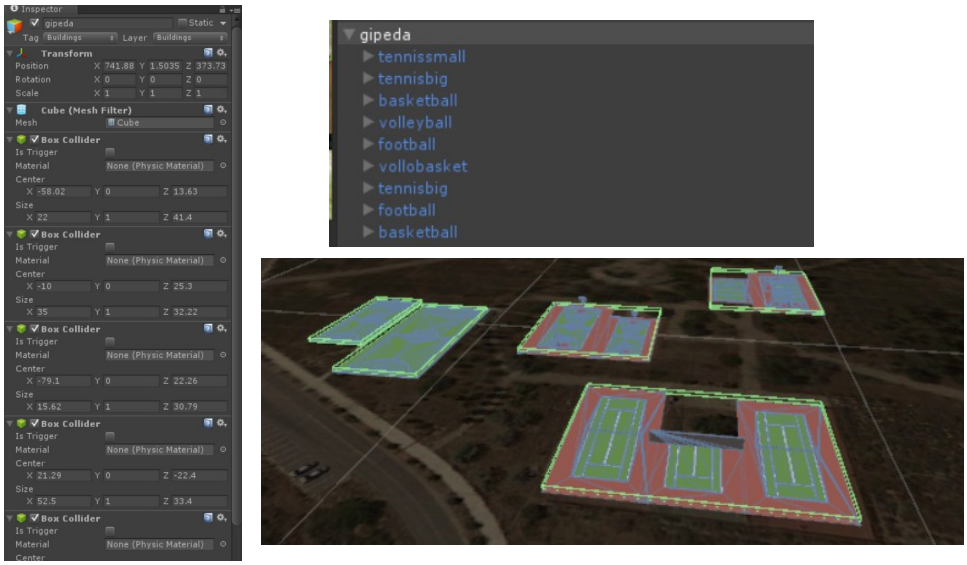
Όσο πιο πολύπλοκος είναι ένας Mesh Collider τόσο περισσότερο επιβαρύνει τη λειτουργία της εφαρμογής μας καθώς οι έλεγχοι οι οποίοι πρέπει να γίνονται ανά πάσα στιγμή από τη μηχανή του Unity όσον αφορά το αν υπάρχει κάποια αλληλεπίδραση με ένα αντικείμενο είναι περισσότεροι από ότι θα ήταν αν ο Mesh Collider ήταν ένας κύβος ή ακόμα καλύτερα μια επίπεδη επιφάνεια.

Το μειονέκτημα της χρήσης απλών Mesh Colliders είναι η ακρίβεια, λόγω του ότι ο Collider δε μπορεί να εφαρμόσει ακριβώς στο μοντέλο που απεικονίζεται.

Λόγω του ότι πέραν από την αλληλεπίδραση με το χρήστη τα κτίρια θα αλληλεπιδρούν σπάνια με άλλα αντικείμενα χρησιμοποιήσαμε σαν Colliders τα Mesh Filters των μοντέλων. Αν επιλέξουμε ένα αντικείμενο στη σκηνή το οποίο έχει Collider component τότε αυτό θα εμφανιστεί με πράσινο χρώμα στη σκηνή μας. Αν δεν έχουμε ορίσει Collider τότε θα δούμε μόνο το Mesh Filter του να διαγράφεται με μπλε χρώμα (Εικόνα 4.15).

Εικόνα 4.15: Mesh collider & mesh filter

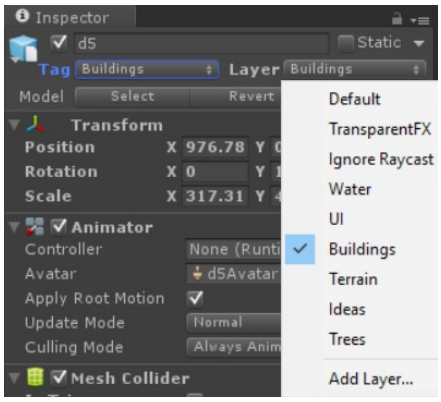
Εξαίρεση αποτελούν τα γήπεδα στα οποία δημιουργήσαμε στη σκηνή ένα GameObject στο οποίο ορίσαμε σαν child objects τα μοντέλα των γηπέδων (Εικόνα 4.16).



Εικόνα 4.16: Δημιουργία γηπέδων

Έπειτα στο parent object δημιουργήσαμε Box Colliders τους οποίους μετασχηματίσαμε ώστε να εφαρμόσουν όσο το δυνατόν καλύτερα στα τρισδιάστατα μοντέλα μας.

Έχοντας πλέον ρυθμίσει όλα μας τα αντικείμενα κτιρίων στη σκηνή το μόνο που μένει είναι να ορίσουμε Tag και Label για τα αντικείμενα κτιρίων. Μπορούμε εύκολα να χρησιμοποιήσουμε κάποια από τις προεπιλογές ή να δημιουργήσουμε τις δικές μας επιλογές επιλέγοντας ένα αντικείμενο στη σκηνή και ανοίγοντας τα αντίστοιχα αναδυόμενα μενού. Για τα κτίρια δημιουργήσαμε νέα Tag και Label με όνομα Buildings (Εικόνα 4.17). Αυτό επιτρέπει την ομαδοποίηση αντικειμένων στη σκηνή ώστε να μπορούμε γρηγορότερα να ανατρέξουμε σε αυτά ή να εκτελέσουμε διαφορετικές ενέργειες ανάλογα με την ομάδα στην οποία ανήκει ένα αντικείμενο.

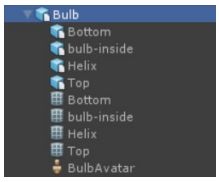


Εικόνα 4.17: Ορισμός Tag & Layer αντικειμένου

4.1.3 Γλόμποι

Για την εμφάνιση των δημοσιεύσεων των χρηστών στο χάρτη, θα χρησιμοποιήσουμε το τρισδιάστατο μοντέλο ενός λαμπτήρα. Ο χρήστης θα μπορεί να ενεργοποιήσει τη προβολή δημοσιεύσεων ώστε να εμφανίσει τις δημοσιεύσεις όλων των χρηστών με τη μορφή λαμπτήρων οι οποίοι θα ίπτανται πάνω από το έδαφος στο σημείο που οι χρήστες έχουν κάνει μια δημοσίευση.

Σε αντίθεση με τα κτίρια τα οποία είναι μοναδικά, το μοντέλο του λαμπτήρα είναι αρκετά κοινό, οπότε μπορούμε να χρησιμοποιήσουμε κάποιο έτοιμο δωρεάν μοντέλο που θα βρούμε στο internet[1], και να το επεξεργαστούμε ώστε να ικανοποιεί τις απαιτήσεις μας, αντί να δημιουργήσουμε το δικό μας μοντέλο από την αρχή.

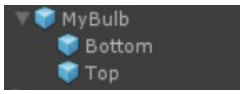


Εικόνα 4.18: Prefab γλόμπου

Το μοντέλο το οποίο θα χρησιμοποιήσουμε είναι αρκετά σύνθετο. Αποτελείται από τέσσερις Mesh Renderers, δυο εκ των οποίων δε μας χρειάζονται γιατί δε θα είναι ορατοί (Εικόνα 4.18).

Αυτοί είναι ο bulb-inside και Helix, οι οποίοι αποτελούν τον εσωτερικό μηχανισμό του γλόμπου. Ο γλόμπος στη σκηνή μας δε θα είναι διαφανής, αλλά υπάρχει και το ενδεχόμενο από τη στιγμή που οι γλόμποι αντιπροσωπεύουν δημοσιεύσεις χρηστών, να υπάρχουν ταυτόχρονα στη σκηνή μας δεκάδες αν όχι εκατοντάδες αντίγραφα. Το να απλουστεύσουμε το μοντέλο, μειώνοντας τον αριθμό πολυγώνων που το απαρτίζουν, αφαιρώντας κομμάτια τα οποία δε μας χρειάζονται, μπορεί να βελτιώσει σημαντικά την απόδοση. Λόγω του ότι δε μπορούμε να επεξεργαστούμε το prefab μοντέλο θα πρέπει να βρούμε ένα τρόπο να δημιουργήσουμε ένα νέο prefab όπως ακριβώς το χρειαζόμαστε.

Το Unity μας προσφέρει έναν εύκολο τρόπο να το κάνουμε αυτό. Αρκεί να δημιουργήσουμε ένα αντίγραφο του prefab το οποίο έχουμε στη σκηνή μας. Μόλις το κάνουμε αυτό μπορούμε να σπάσουμε τη σύνδεση μεταξύ prefab και του GameObject το οποίο δημιουργήσαμε στη σκηνή. Αυτό το κάνουμε επιλέγοντας το αντικείμενο στη σκηνή, και χρησιμοποιώντας την επιλογή break prefab insance που θα βρούμε στο μενού GameObject.Τώρα μπορούμε να επεξεργαστούμε το μοντέλο μας στη σκηνή (Εικόνα 4.19).

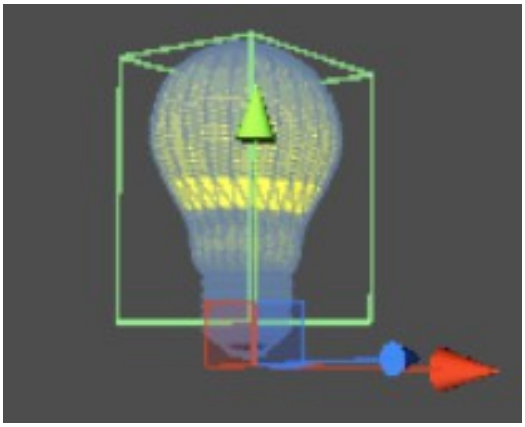


Εικόνα 4.19: Αντίγραφο γλόμπου

Το πρώτο μας βήμα είναι να διαγράψουμε τους Mesh Renderers τους οποίους δε χρειαζόμαστε. Έπειτα επεξεργαζόμαστε τα materials των Renderers τους οποίους κρατήσαμε ώστε να έχουμε ένα οπτικό αποτέλεσμα που μας ικανοποιεί. Εκτός από τα χρώματα μπορούμε να αλλάξουμε και τους shaders, και χρησιμοποιώντας ένα self illumin shader να δώσουμε έτσι στο πάνω μέρος του γλόμπου τη δυνατότητα να εκπέμπει φως (Εικόνα 4.20).

Εικόνα 4.20: Υλικό γλόμπου

Θα χρειαστεί να προσθέσουμε στο γλόμπο ένα Collider component. Λόγω του ότι μπορεί ο αριθμός των αντιγράφων του μοντέλου στη σκηνή να είναι μεγάλος επιλέγουμε τη χρήση ενός Box Collider (Εικόνα 4.21). Αν χρησιμοποιούσαμε σαν collider το Mesh Filter του γλόμπου τότε με μερικές δεκάδες γλόμπους οι οποίοι έρχονται σε επαφή θα παρατηρούσαμε καθυστερήσεις στους υπολογισμούς.



Εικόνα 4.21: GameObject γλόμπου

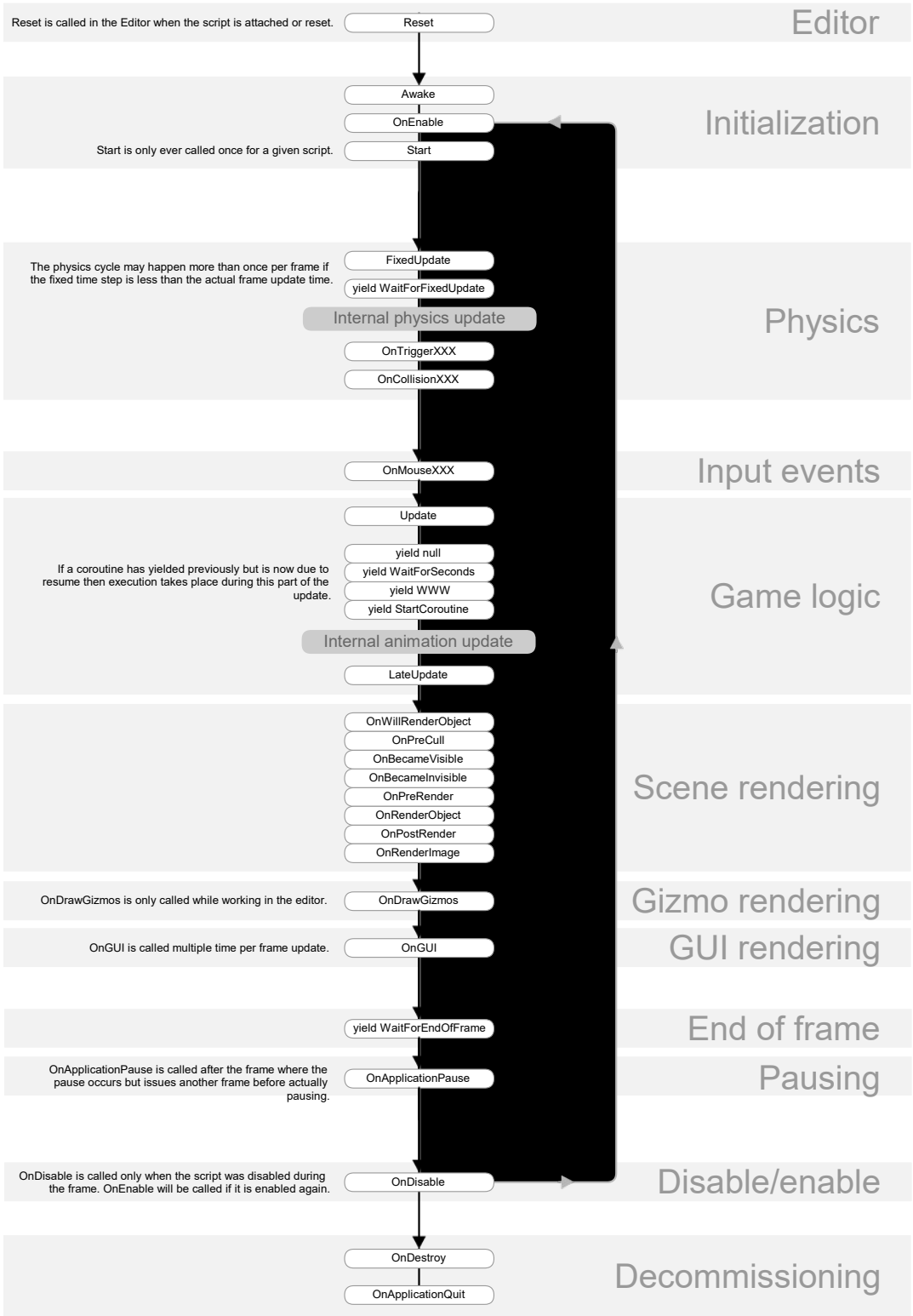
Αφού δημιουργήσουμε και ένα Layer με τίτλο Ideas το οποίο θέτουμε σαν layer του γλόμπου μπορούμε να τον αποθηκεύσουμε σαν prefab. Τέλος, στο μετασχηματισμό θέτουμε Y=30 και αυτό θα είναι το ύψος πάνω στο οποίο θα αιωρείται ο γλόμπος.

Για να το κάνουμε αυτό αρκεί να τον σύρουμε από τη σκηνή, στη καρτέλα Project. Το νέο prefab θα δημιουργηθεί σύμφωνα με τις προτιμήσεις μας και τώρα μπορούμε να το χρησιμοποιήσουμε.

Για να κάνουμε ένα γλόμπο να αιωρείται πάνω από το έδαφος θα πρέπει να χρησιμοποιήσουμε τη δυνατότητα scripting του Unity. Για να δημιουργήσουμε το πρώτο μας script χρησιμοποιούμε το μενού Create στη καρτέλα Project. Το μενού αυτό μας προσφέρει πολλές επιλογές δημιουργίας GameObjects, μια εκ των οποίων είναι το C# Script το οποίο χρειαζόμαστε.

Δίνουμε στο script όνομα FloatIdea και το ανοίγουμε με τον editor του Unity, παρατηρούμε ότι υπάρχουν δυο κενές συναρτήσεις, οι Start και Update (Εικόνα 4.23).

Η συνάρτηση Start τρέχει μια φορά τη πρώτη φορά που θα ενεργοποιηθεί το Script ενώ η Update τρέχει μια φορά για κάθε frame. Στο Unity υπάρχει μια πληθώρα συναρτήσεων οι οποίες εκτελούνται με προκαθορισμένη σειρά κατά την λειτουργία ενός script (Εικόνα 4.22).



Εικόνα 4.22: Διάγραμμα ροής συναρτήσεων[10]

Δηλώνουμε τρεις public μεταβλητές. Οι δυο θα αποθηκεύουν το X και το Z του γλόμπου και η τρίτη ένα RigidBody. Το Unity μας δίνει τη δυνατότητα να δηλώσουμε μεταβλητές οι οποίες μπορούν να αποθηκεύουν GameObjects, vectors, χρώματα και γενικά οποιοδήποτε component ή τιμή ενός στοιχείου του είναι προσβάσιμο και μπορεί να αλλάξει μέσω script.

```
1 using UnityEngine;
2 using System.Collections;
3
4 public class FloatIdea : MonoBehaviour {
5
6
7     public Rigidbody rb;
8     public float x;
9     public float z;
10
11    void Start() {
12        rb = GetComponent<Rigidbody>();
13        x = this.transform.position.x;
14        z = this.transform.position.z;
15    }
16
17
18
19    void FixedUpdate() {
20            // add force towards a Vector3
21            Vector3 direction = new Vector3 ( x, 30, z) - this.transform.position;
22            rb.AddForceAtPosition(direction.normalized, transform.position);
23    }
24
25    |
26 }
```

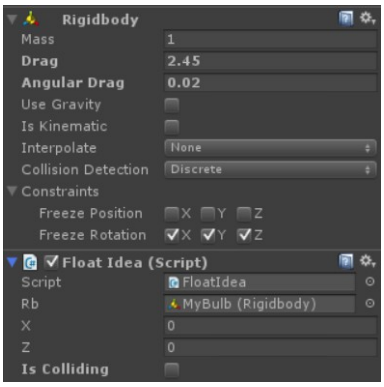
Εικόνα 4.23: FloatIdea

Όταν μια μεταβλητή δηλώνεται ως public γίνεται ορατή στον inspector από όπου μπορούμε να αλλάξουμε ή να δούμε τη τιμή της.

Μπορούμε να αρχικοποιήσουμε τις μεταβλητές X και Y χρησιμοποιώντας τις αντίστοιχες μεταβλητές του μετασχηματισμού του αντικειμένου στο οποίο θα προσθέσουμε το script σαν component και τη μεταβλητή RigidBody χρησιμοποιώντας ένα component RigidBody το οποίο επίσης θα προσθέσουμε στο μοντέλο μας.

Χρησιμοποιούμε τη συνάρτηση FixedUpdate ούτως ώστε σε περιπτώσεις όπου το frame rate είναι μικρό η κίνηση του γλόμπου να είναι ομαλή, κάτι το οποίο δε θα συνέβαινε με τη συνάρτηση Update, και με τη χρήση της ασκούμε μια δύναμη στο αντικείμενο μας η οποία το ωθεί προς το σημείο αιώρησης. Αυτό το πετυχαίνουμε υπολογίζοντας τη φορά της δύναμης και χρησιμοποιώντας τη για να ασκήσουμε δύναμη σε ένα σημείο του RigidBody με την εντολή AddForceAtPosition.

Το Script μας είναι έτοιμο και μπορούμε να το προσθέσουμε σαν component στο prefab του γλόμπου το οποίο δημιουργήσαμε νωρίτερα. Εκτός από το script πρέπει να προσθέσουμε και ένα RigidBody component καθώς είναι απαραίτητο ώστε να ασκήσουμε δυνάμεις φυσικής.



Εικόνα 4.24: Rigidbody component

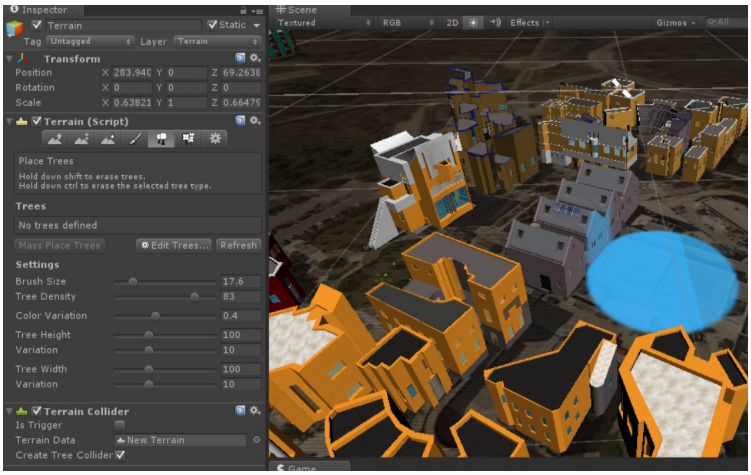
Παρατηρούμε ότι το RigidBody μας προσφέρει τη δυνατότητα να παγώσουμε τη περιστροφή ή τη μετακίνηση σε κάθε άξονα. Επιλέγουμε να παγώσουμε τη περιστροφή του αντικειμένου σε όλους τους άξονες, αφού θέλουμε να αιωρείται χωρίς να περιστρέφεται αλλά δε παγώνουμε τη θέση του αφού θέλουμε να μπορεί να μετακινηθεί αν συγκρουστεί με άλλο αντικείμενο (Εικόνα 4.24).

Ο γλόμπος μας είναι έτοιμος και αν τοποθετήσουμε ένα γλόμπο στη σκηνή θα τον δούμε να αιωρείται κατευθυνόμενος σε υψόμετρο Y=30. Πειραματιζόμαστε με τις επιλογές αντίστασης που προσφέρει το RigidBody ώστε να πετύχουμε τη κίνηση και τη ταχύτητα που θέλουμε, μπορούμε έτσι να ελαττώσουμε ή να αυξήσουμε τη ταχύτητα με την οποία κινείται ένας γλόμπος αν για παράδειγμα έρθει σε σύγκρουση με κάποιο άλλο αντικείμενο.

4.1.4 Terrain

Επόμενο βήμα θα είναι να σχεδιάσουμε τις λεπτομέρειες του Terrain. Φυσικά και με τη φωτογραφία δορυφόρου που έχουμε από τους χάρτες, ο χρήστης μπορεί να κατατοπιστεί πλήρως, αλλά θέλουμε να σχεδιάσουμε ένα πιο λεπτομερή, μοναδικό χάρτη. Για αυτό το λόγο θα χρειαστεί να προσθέσουμε δέντρα, φυτά, πινακίδες αλλά και να χρωματίσουμε το terrain με textures.

Στο Unity μπορούμε μέσω της επιλογής Place Trees που θα βρούμε στη καρτέλα inspector του terrain να επιλέξουμε ένα μοντέλο δέντρου και να τοποθετήσουμε αντίγραφα του χρησιμοποιώντας το εργαλείο βούρτσας (Εικόνα 4.25). Με αυτό το εργαλείο μπορούμε να “βάψουμε” το terrain με δέντρα επιλέγοντας από τη συχνότητα με την οποία θα εμφανίζονται τα δέντρα μέχρι και αν το φύλλωμα τους θα έχει αποχρώσεις. Μπορούμε επίσης να δημιουργήσουμε τα δικά μας μοντέλα δέντρων μέσω Unity όπου μπορούμε να σχεδιάσουμε το κορμό του δέντρου και να προσθέσουμε φύλλωμα μέσω της δυνατότητας Create tree στο μενού GameObject και να χρησιμοποιήσουμε αυτό το μοντέλο για να γεμίσουμε το terrain μας.

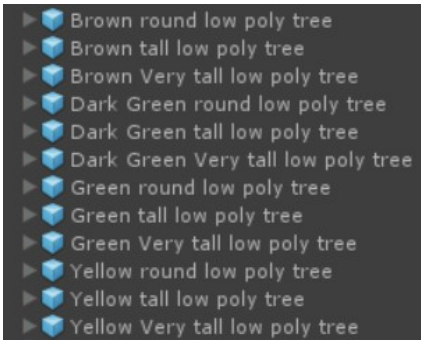


Εικόνα 4.25: Βούρτσα δέντρων

Εμείς για τα δέντρα χρησιμοποιήσαμε ένα δωρεάν τρισδιάστατο μοντέλο που βρήκαμε στο internet[2] και αφού το επεξεργαστήκαμε προσθέσαμε χειροκίνητα τα δέντρα στο terrain ώστε να μη τοποθετηθούν τυχαία αλλά να έχουν τη διάταξη που εμείς θέλουμε, όπως για παράδειγμα ο ελαιώνας.

Για να δημιουργήσουμε τα prefabs των μοντέλων των δέντρων εργαστήκαμε με τον ίδιο τρόπο όπως με τους λαμπτήρες. Αφού δημιουργήσαμε ένα αντίγραφο του μοντέλου που βρήκαμε στο internet στη σκηνή μας το επεξεργαστήκαμε και δημιουργήσαμε prefab από το μοντέλο της σκηνής.

Για να έχουμε ποικιλία και το αποτέλεσμα να είναι καλύτερο οπτικά δημιουργήσαμε 9 prefabs σε τρία διαφορετικά ύψη και χρώματα, καφέ, πράσινο και σκούρο πράσινο σε κοντό, ψηλό και πολύ ψηλό (Εικόνα 4.26). Τώρα που τα prefabs είναι έτοιμα μπορούμε να τα τοποθετήσουμε στη σκηνή.



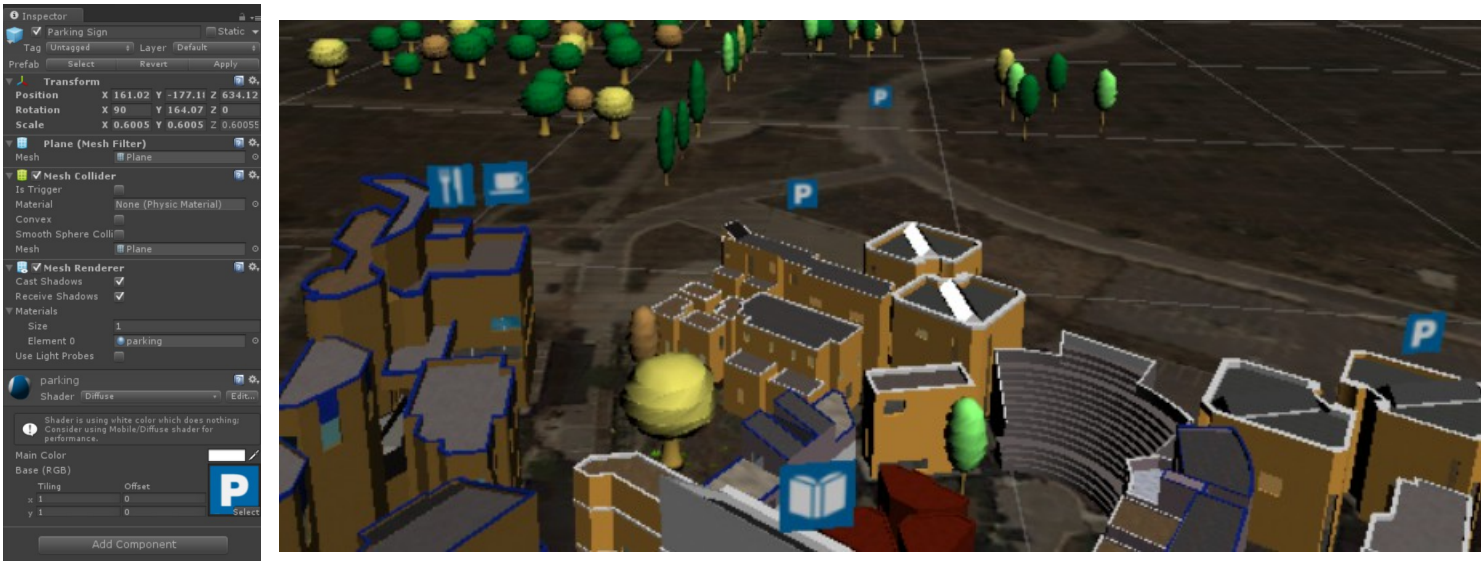

Εικόνα 4.26: Τοποθέτηση δέντρων

Για να προσθέσουμε σήμανση στο χάρτη, όπως για παράδειγμα πινακίδες για να υποδεικνύουν το πάρκινγκ θα δημιουργήσουμε ένα απλό τρισδιάστατο μοντέλο με τη βοήθεια του Unity.

Πρώτα πρέπει να φτιάξουμε τα materials που θα χρησιμοποιηθούν στις πινακίδες. Στη καρτέλα project επιλέγουμε Create material και στο νέο material που θα δημιουργηθεί, επιλέγουμε το texture που θέλουμε να χρησιμοποιείται, όπως για παράδειγμα το σήμα του πάρκινγκ. Επιλέγουμε τον shader για το material και είναι έτοιμο προς χρήση (Εικόνα 4.27).

Από το μενού GameObject δημιουργούμε μια επίπεδη επιφάνεια. Στο component Mesh Renderer επιλέγουμε το material που μόλις δημιουργήσαμε και μετασχηματίζουμε την επιφάνεια ώστε να αιωρείται και να είναι στραμμένη προς τη κάμερα.

Δημιουργούμε ένα prefab ώστε να μη χρειάζεται να επαναλαμβάνουμε τη διαδικασία για κάθε πινακίδα και μπορούμε να τις τοποθετήσουμε στο χάρτη.



Εικόνα 4.27: Τοποθέτηση πινακίδων

Επόμενο βήμα είναι η προσθήκη βλάστησης. Όπως και με τα δέντρα έτσι και με τη βλάστηση το Unity μας προσφέρει εργαλεία με τα οποία μπορούμε να σχεδιάσουμε εύκολα τις λεπτομέρειες του terrain μας. Για αυτή τη δουλειά θα χρησιμοποιήσουμε το εργαλείο paint details του terrain (Εικόνα 4.28).

Αυτό το εργαλείο μας δίνει τη δυνατότητα να χρησιμοποιήσουμε textures ή ακόμα και τρισδιάστατα μοντέλα ώστε να σχεδιάσουμε λεπτομέρειες σε ένα terrain. Για τη βλάστηση θα χρησιμοποιήσουμε τα δυο textures γρασιδιού που περιέχει το Unity.

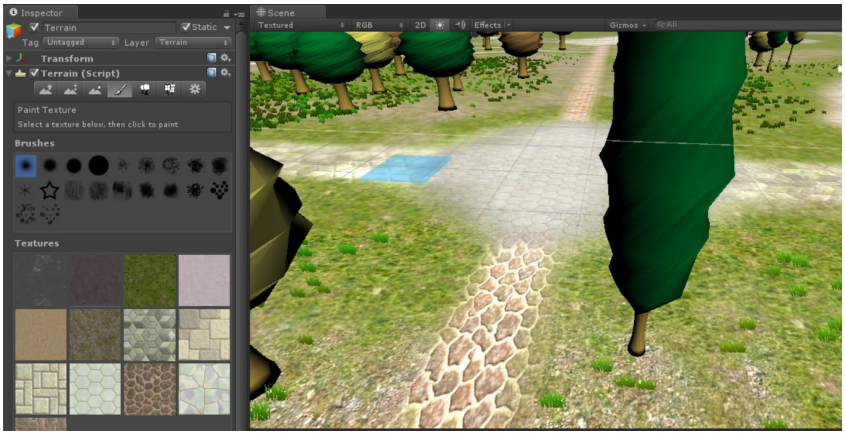
Επιλέγουμε το texture και ρυθμίζοντας το μέγεθος της βούρτσας και τη συχνότητα με την οποία θα εμφανίζεται το γρασίδι προσθέτουμε λεπτομέρειες στο έδαφος.

Εικόνα 4.28: Προσθήκη βλάστησης

Το τελευταίο πράγμα που μένει να κάνουμε όσον αφορά το terrain είναι ο χρωματισμός του, ώστε να σχεδιάσουμε δρόμους και χωράφια και να μη χρειάζεται να χρησιμοποιούμε το texture που πήραμε από τους χάρτες.

Στο εργαλείο paint texture που θα βρούμε στον inspector του terrain, εισάγουμε τα textures τα οποία θα χρησιμοποιηθούν. Χρειαζόμαστε texture γρασιδιού, χώματος, άσφαλτου και γενικά textures τα οποία απεικονίζουν τις επιφάνειες εδάφους του πολυτεχνείου.

Μόλις εισάγουμε όλα τα textures και επιλέγοντας το κατάλληλο μέγεθος βούρτσας ξεκινάμε να βάφουμε με κάθε texture πάνω από τη φωτογραφία των χαρτών (Εικόνα 4.29). Για παράδειγμα βάφουμε όλους τους δρόμους με το texture ασφάλτου ώστε να ξέρουμε που βρίσκονται οι δρόμοι όταν αφαιρέσουμε την εικόνα των χαρτών.



Εικόνα 4.29: Χρωματισμός terrain

Μόλις τελειώσουμε με το χρωματισμό των βασικών λεπτομερειών του χάρτη και απενεργοποιήσουμε το texture με το screenshot που χρησιμοποιούσαμε μέχρι σήμερα (Εικόνα 4.30), μπορούμε πειράζοντας τις ρυθμίσεις της βούρτσας να σχεδιάσουμε τις λεπτομέρειες του χρωματισμού του εδάφους, προκαλώντας για παράδειγμα ομαλές εναλλαγές μεταξύ ασφάλτου και γρασιδιού.

Εικόνα 4.30: Σύγκριση εικόνας χάρτη με χρωματισμένο terrain

4.2 Κάμερα και φωτισμός

Το μόνο που μένει να κάνουμε ώστε να ολοκληρώσουμε το σχεδιασμό του τρισδιάστατου περιβάλλοντος είναι να ρυθμίσουμε περαιτέρω τη κάμερα, δίνοντας στο χρήστη τη δυνατότητα περιήγησης στους χώρους του πολυτεχνείου και να φτιάξουμε το φωτισμό της σκηνής ώστε όχι μόνο να δημιουργούνται σκιές και να έχουμε καλύτερο φωτισμό με εναλλαγές αλλά και να φωτίζονται συγκεκριμένα κτίρια όταν τα επιλέγουμε.

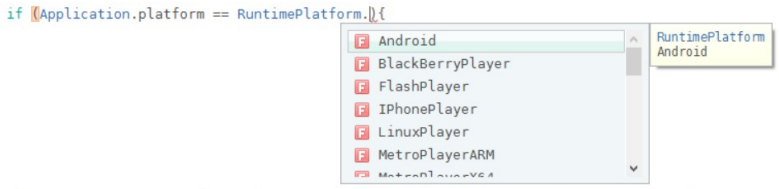
4.2.1 Κάμερα

Στον αρχικό σχεδιασμό ορίσαμε το ύψος της κάμερας από το έδαφος Y=60. Σε αυτό το υψόμετρο θα κινείται η κάμερα και αυτό μπορεί να γίνει με δυο τρόπους. Μπορούμε είτε να ασκήσουμε δυνάμεις φυσικής στη κάμερα ώστε να κινηθεί ή να τη μετακινήσουμε αλλάζοντας τη θέση μέσω του μετασχηματισμού της. Θα χρησιμοποιήσουμε και τους δυο τρόπους και για να το κάνουμε αυτό θα πρέπει να προσθέσουμε ένα component RigidBody στη κάμερα και να κλειδώσουμε τη περιστροφή της, αφού δε θα χρειαστούμε αυτή τη δυνατότητα, όπως κάναμε και με τους γλόμπους.

Για να μετακινήσουμε τη κάμερα θα κάνουμε χρήση της οθόνης αφής του adnroid κινητού μας. Για αυτό το λόγο δημιουργούμε ένα script component στη κάμερα με όνομα detectTouch.cs στο οποίο θα γίνεται η ανίχνευση συμβάντων αφής και θα λειτουργεί σαν controller ώστε να εκτελεί τις απαραίτητες ενέργειες ανάλογα με τη κατάσταση στην οποία βρισκόμαστε.

Αυτό το script είναι ο πυρήνας του χειρισμού της εφαρμογής μας.

Στη συνάρτηση FixedUpdate, γίνεται έλεγχος σχετικά με το αν η συσκευή μας έχει λειτουργικό android. Αν ναι τότε θα γίνονται οι έλεγχοι που αφορούν το χειρισμό για συσκευές android, εναλλακτικά μπορούμε να ενεργοποιήσουμε χειρισμό με βελάκια του πληκτρολογίου αν η συσκευή μας είναι υπολογιστής ή laptop, ή χειρισμό για συσκευή iOS. Όλα αυτά μπορούν να υλοποιηθούν με if-else εντολές και δείχνουν τη δύναμη και την ευελιξία της μηχανής του Unity, αφού μπορούμε σε ένα script να έχουμε το χειρισμό για κάθε διαφορετική συσκευή.



Εικόνα 4.31: Επιλογές RuntimePlatform

Μπορούμε εύκολα να δημιουργήσουμε μια cross platform εφαρμογή γράφοντας τα script μας με τέτοιο τρόπο ώστε να μη χρειάζονται αλλαγές και διαφορετικές εκδόσεις της εφαρμογής για κάθε συσκευή αλλά να ενεργοποιούνται αυτόματα διαφορετικά scripts ανάλογα με τη συσκευή στην οποία τρέχει η εφαρμογή μας και ανάλογα τις απαιτήσεις και δυνατότητες αυτής της συσκευής. Όλα αυτά επιτυγχάνονται χάρη στο πακέτο εντολών που μας προσφέρει το Unity οι οποίες στη πλειοψηφία τους λειτουργούν χωρίς προβλήματα σε κάθε συσκευή για την οποία μπορούμε να εξάγουμε την εφαρμογή μας.

Για παράδειγμα η εντολή Application.platform == RuntimePlatform.Android επιστρέφει true αν η συσκευή μας έχει λειτουργικό android και αντίστοιχοι έλεγχοι μπορούν να γίνουν για κάθε είδους συσκευή ή σύστημα το οποίο μπορεί να εξαχθεί ένα project του Unity (Εικόνα 4.31).

Στη αυτή τη διπλωματική θα ασχοληθούμε μόνο με android συσκευές. Οπότε αν η παραπάνω συνθήκη είναι αληθής κάνουμε έλεγχο της κατάστασης στην οποία βρίσκεται η κάμερα. Η κάμερα μας μπορεί

να βρίσκεται σε τρεις καταστάσεις. Μπορεί να είναι είτε ακίνητη, είτε να την έχει πιάσει ο χρήστης ή να κινείται και αυτό το αναπαριστούμε με τη χρήση της μεταβλητής Status που μπορεί να πάρει τιμές idle, grab και moving αντίστοιχα. Η μεταβλητή Status αρχικοποιείται ως idle αφού η κάμερα είναι ακίνητη στην αρχή.

Κάνοντας έλεγχο της Status και αλλάζοντας την όταν περνάμε σε μια από τις άλλες καταστάσεις μπορούμε να υλοποιήσουμε το χειρισμό της κάμερας. Μπορούμε για κάθε κατάσταση του Status να ελέγξουμε αν ο χρήστης ξεκίνησε να αγγίζει την οθόνη, αν το άγγιγμα του παραμένει σταθερό ή αν μετακινείται με τη χρήση του TouchPhase enumeration που διαθέτει το Unity και της κλάσης Input (Εικόνα 4.32).

Εικόνα 4.32: Επιλογές TouchPhase

Όταν λοιπόν η κάμερα είναι ακίνητη και ο χρήστης αγγίξει την οθόνη θέλουμε να κρατήσουμε το σημείο το οποίο άγγιξε ο χρήστης στο χάρτη. Αυτό το σημείο θα είναι το πρώτο μας σημείο στο χάρτη.

Αν ο χρήστης μετακινήσει το δάκτυλο του θέλουμε η κάμερα να μετακινηθεί προς το δεύτερο σημείο το οποίο άγγιξε.

Μόλις σχεδιάσουμε τη λογική του χειρισμού και την υλοποιήσουμε με το συνδυασμό της μεταβλητής Status, τις εντολές ελέγχου touch screen που μας δίνει το Unity και τη χρήση μετρητών και άλλων βοηθητικών μεταβλητών το μόνο που μένει είναι να εντοπίσουμε τα σημεία στο τρισδιάστατο χώρο τα οποία άγγιξε ο χρήστης. Τη λύση δίνει για άλλη μια φορά το Unity με τις δομές Ray και RaycastHit.

```
Ray ray = camera.ScreenPointToRay (new Vector3 (Input.GetTouch (0).position.x, Input.GetTouch (0).position.y));
```

Δηλώνουμε ένα Ray το οποίο δημιουργεί μια ακτίνα η οποία ξεκινάει από τη κοντινή επιφάνεια του πρίσματος της κάμερας και περνάει από ένα σημείο (x,y) της οθόνης. Στη περίπτωση μας αυτό το σημείο το παίρνουμε από το σημείο που γίνεται αφή.

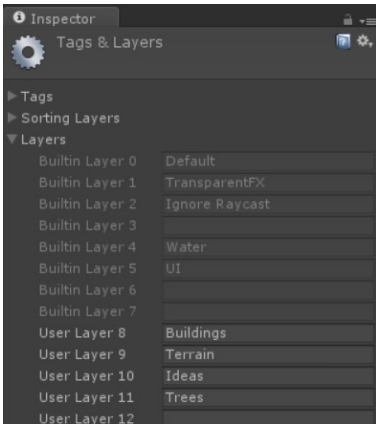
```
RaycastHit hit;
```

Δηλώνουμε ένα RaycastHit το οποίο θα κρατά πληροφορίες για το αντικείμενο που πετυχαίνει η ακτίνα μας.

```
if (Physics.Raycast (ray, out hit, Mathf.Infinity, 1 << layersToHit)) {
        nameOfHit = hit.transform.name;
        pointOfHit = hit.point;
}
```

Εικόνα 4.33: Raycasting

Τώρα μπορούμε να εκτελέσουμε ένα Raycast με τη βοήθεια της κλάσης Physics του Unity και να πάρουμε ένα αποτέλεσμα για το τι πέτυχε η ακτίνα μας (Εικόνα 4.33). Λόγω του ότι το Raycast γίνεται με τη χρήση της κλάσης Physics τα αντικείμενα που θέλουμε να εντοπίσουμε θα πρέπει να έχουν σαν component ένα collider ώστε να μπορεί να τα εντοπίσει το Ray. Η μεταβλητή layersToHit είναι ακέραιος αριθμός ο οποίος αντιπροσωπεύει τον αριθμό του layer στο οποίο ανήκουν τα αντικείμενα που θέλουμε να πετύχουμε (Εικόνα 4.34) και με bit shifting δημιουργείται ένα bitmask το οποίο χρησιμοποιεί η εντολή Raycast.



Εικόνα 4.34: Δημιουργία Layer

Τώρα που είδαμε πως μπορούμε να χρησιμοποιήσουμε Rays ώστε να εντοπίσουμε αντικείμενα στη σκηνή με τη χρήση αφής, μπορούμε να υλοποιήσουμε το χειρισμό της κάμερας.

Η πρώτη από τις κινήσεις που θέλουμε να πραγματοποιεί η κάμερα είναι να μετακινείται όταν ο χρήστης την έχει "πιάσει", όταν δηλαδή αγγίζει παρατεταμένα την οθόνη του και μετακινεί το δάκτυλο του χωρίς να το σηκώνει.

Στον controller του script DetectTouch που περιγράψαμε νωρίτερα θα χρειαστεί να κρατάμε 2 σημεία στο terrain για να υλοποιήσουμε αυτή τη δυνατότητα. Το πρώτο σημείο θα το κρατάμε με τη χρήση ενός Raycast το οποίο θα χτυπάει μόνο το terrain με τη χρήση του σωστού layer. Το δεύτερο σημείο θα το παίρνουμε πάλι με τον ίδιο τρόπο και θα είναι το τελευταίο σημείο αφής όταν ο controller είναι σε κατάσταση "grab".

Προσθέτουμε ένα ακόμα script component στη κάμερα μας με όνομα moveCamera.cs στο οποίο θα συμπεριλάβουμε τις συναρτήσεις οι οποίες θα πραγματοποιούν μετακίνηση της κάμερας.

Στη FixedUpdate αυτού του script σχεδιάζουμε ένα controller ο οποίος θα ελέγχει σε τι κατάσταση βρίσκεται η κάμερα. Οι καταστάσεις αυτές διαφέρουν από εκείνες του DetectTouch controller καθώς δεν αφορούν μόνο το είδος της κίνησης που κάνει η κάμερα. Μια από αυτές τις καταστάσεις υποδηλώνει για παράδειγμα ότι η κάμερα βρίσκεται εκτός ορίων του χάρτη ώστε να επιστρέψει πάλι στο κέντρο. Αν λοιπόν στο DetectTouch βρισκόμαστε στη κατάσταση "grab" καλούμε δυο συναρτήσεις από το moveCamera. Τη CalculateDistance και grabCamera (Εικόνα 4.35).

```
}else if (State == "grab"){
    if(Input.GetTouch(0).phase == TouchPhase.Stationary || (Input.GetTouch(0).phase == TouchPhase.Moved && Input.GetTouch(0).deltaPosition.magnitude < 10)){
        secondTerrainPoint = Camera.main.GetComponent<castRay>().Cast(9,State).point; //cast on terrain and return point of hit
        Camera.main.GetComponent<moveCamera>().CalculateDistance(firstTerrainPoint,secondTerrainPoint);
        Camera.main.GetComponent<moveCamera>().grabCamera();
    }
```

Εικόνα 4.35: Έλεγχος κατάστασης κάμερας

Η CalculateDistance υπολογίζει την απόσταση μεταξύ των δυο σημείων που έχουμε κρατήσει και την αποθηκεύει σε δυο μεταβλητές τις οποίες μπορούμε να χρησιμοποιήσουμε από οποιοδήποτε άλλη συνάρτηση του script (Εικόνα 4.36).

```
public float DistanceX;
public float DistanceZ;

public void CalculateDistance (Vector3 Point1, Vector3 Point2){ //calculates terrain distance between 2 RayHits
    DistanceX = Point1.x - Point2.x;
    DistanceZ = Point1.z - Point2.z;

}
```

Εικόνα 4.36: Υπολογισμός απόστασης δυο σημείων

Η grabCamera αλλάζει τις βοηθητικές μεταβλητές που χρησιμοποιούμε για τον controller της κίνησης της κάμερας ώστε στη FixedUpdate να τρέξει κατάλληλος κώδικας για τη κίνηση της κάμερας (Εικόνα 4.37).

```
public bool isGliding = false;
public bool isGrabbed = false;
public bool isOffLimits = false;
public bool isMovingToPosition = false;

public void grabCamera(){
    isGliding = false;
    isMovingToPosition = false;
    isGrabbed = true;
    glideSpeed = 0;
    fracJourney = 1;

}
```

Εικόνα 4.37: Συνάρτηση grabCamera

Στη FixedUpdate μηδενίζουμε την οποιαδήποτε ταχύτητα μπορεί να έχει η κάμερα μας από άλλες κινήσεις που πραγματοποιεί και προσθέτοντας τη διαφορά απόστασης στα σημεία του terrain τα οποία αγγίξαμε στο μετασχηματισμό της κάμερας τη μετακινούμε το νέο σημείο (Εικόνα 4.38).

```
...}else if(isGrabbed == true){
        Camera.main.rigidbody.velocity = new Vector3(0, 0, 0);
        transform.position = new Vector3(transform.position.x+DistanceX, 60, transform.position.z+DistanceZ);
        isGrabbed = false;
    }else if(isOffLimits == true){ ...
```

Εικόνα 4.38: Σύρσιμο κάμερας

Στη περίπτωση που ο χρήστης κάνει μια γρήγορη κίνηση στην οθόνη με το δάκτυλο του θέλουμε η κάμερα να ξεκινήσει να κινείται με ταχύτητα η οποία θα ελαττώνεται προς την αντίθετη φορά της κίνησης του δακτύλου του χρήστη.

Αυτή τη κίνηση μπορούμε να τη πετύχουμε ασκώντας μια δύναμη στη κάμερα. Όπως και με τη προηγούμενη περίπτωση συρσίματος της κάμερας, έτσι και εδώ καλούμε τη συνάρτηση CalculateDistance για να υπολογίσουμε την απόσταση μεταξύ δυο σημείων που άγγιξε ο χρήστης.

```
public Vector3 glidePoint;
Vector3 newCameraPlace;
Vector3 oldCameraPlace;

public void PointToGlide(){
    newCameraPlace = new Vector3 (transform.position.x+DistanceX, 60, transform.position.z+DistanceZ);
    oldCameraPlace = new Vector3 (transform.position.x, 60, transform.position.z);

    glidePoint = newCameraPlace - oldCameraPlace; // move from oldcamera to newcamera

    isGliding = true;
    glideSpeed = glideSpeed + 20;


}
```

Εικόνα 4.39: Συνάρτηση PointToGlide

Σε αυτή τη περίπτωση καλούμε τη συνάρτηση PointToGlide στην οποία υπολογίζουμε το σημείο στο οποίο θέλουμε να μετακινηθεί η κάμερα και αρχικοποιούμε τη ταχύτητα με την οποία θα κινείται (Εικόνα 4.39). Μπορούμε έτσι χρησιμοποιώντας την AddForceToPosition να ασκήσουμε μια δύναμη στη κάμερα μας η οποία θα την ωθεί προς το σημείο που υπολογίσαμε (Εικόνα 4.40). Σε κάθε frame μειώνουμε τη ταχύτητα με την οποία θα κινείται η κάμερα ούτως ώστε η κίνηση να είναι επιβραδυνόμενη.

```
if (isGliding == true) {
    Camera.main.rigidbody.velocity = new Vector3(0, 0, 0);
    Camera.main.rigidbody.AddForceAtPosition(glidePoint*glideSpeed, transform.position, ForceMode.Acceleration);
```

Εικόνα 4.40: Προσθήκης δύναμης ώθησης σε κάμερα

Για να μετακινήσουμε τη κάμερα σε συγκεκριμένο σημείο στο χάρτη, αλλά η κίνηση να είναι ομαλή, σε αντίθεση με τη κίνηση που γίνεται με τη χρήση της συνάρτησης grabCamera στην οποία η κάμερα μεταφέρεται ακαριαία στο σημείο το οποίο θέτουμε, θα χρησιμοποιήσουμε τη συνάρτηση Lerp του Unity (Εικόνα 4.41).

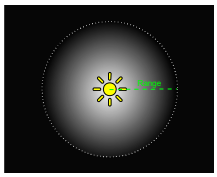
```
}else if(isMovingToPosition == true){
    Camera.main.rigidbody.velocity = new Vector3(0, 0, 0);
    transform.position = Vector3.Lerp(oldCameraPlace, newCameraPlace, fracJourney);
}
```

Εικόνα 4.41: Μετακίνηση κάμερας μέσω του μετασχηματισμού

Αυτή η κίνηση θα μας φανεί χρήσιμη αν για παράδειγμα θέλουμε να μετακινήσουμε τη κάμερα σε ένα κτίριο με βάση τις συντεταγμένες του στο terrain ή στη θέση του χρήστη στο terrain.Η χρήση της συνάρτησης είναι απλή και χρησιμεύει στο να γίνεται ομαλή μετακίνηση της κάμερας μεταξύ δυο σημείων. Η μεταβλητή fracjourney υποδηλώνει το ποσοστό της απόστασης μεταξύ των δυο σημείων στο οποίο βρίσκεται η μετάβαση, με 0 να δηλώνει την αρχή του ταξιδιού και το 1 το τέλος. Αλλάζουμε αυτή τη μεταβλητή σε κάθε frame προσθέτοντας το χρόνο εκτέλεσης του επί τη ταχύτητα μετάβασης και μπορούμε να έχουμε μια ομαλή μετάβαση.

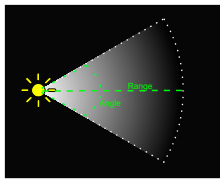
4.2.2 Φωτισμός

Ο φωτισμός παίζει σημαντικό ρόλο στη προβολή της σκηνής μας. Η ένταση του φωτισμού αλλά και η γωνία επηρεάζουν σημαντικά το πως φαίνονται τα αντικείμενα μας. Στο Unity μπορούμε να έχουμε τέσσερα είδη φωτισμού και να χρησιμοποιήσουμε όσες πηγές φωτισμού επιθυμούμε σε μια σκηνή. Ο συνδυασμός διαφορετικών πηγών φωτισμού συνεισφέρει στο να έχουμε ένα όμορφο αποτέλεσμα.



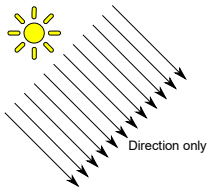
Ένα φως σημείου εκπέμπει ομοιόμορφα το φως από ένα συγκεκριμένο σημείο στο χώρο προς όλες τις κατευθύνσεις και για συγκεκριμένη ακτίνα (Εικόνα 4.42).

Εικόνα 4.42: Φως σημείου



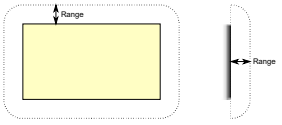
Ένα σποτ εκπέμπει πάλι από ένα συγκεκριμένο σημείο και με συγκεκριμένη ακτίνα αλλά το φως του περιορίζεται από μια γωνία (Εικόνα 4.43).

Εικόνα 4.43: Σποτ



Ένα φως κατεύθυνσης δε χρειάζεται να τοποθετηθεί σε συγκεκριμένο σημείο. Τοποθετείται οπουδήποτε και όλα τα αντικείμενα της σκηνής φωτίζονται υπό μια συγκεκριμένη γωνία (Εικόνα 4.44).

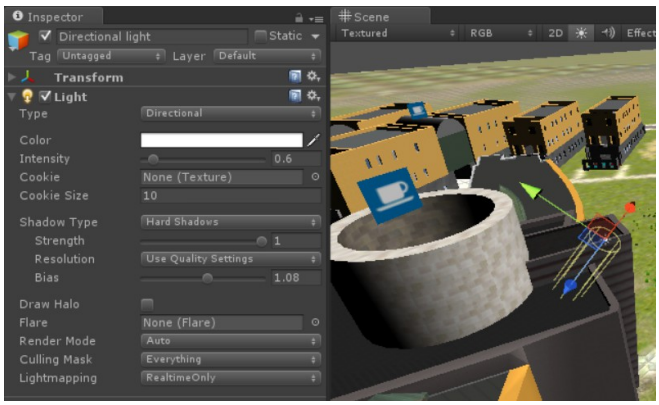
Εικόνα 4.44: Φως κατεύθυνσης



Ένα φως περιοχής ορίζεται από ένα ορθογώνιο στο χώρο. Εκπέμπεται φως προς όλες τις κατευθύνσεις αλλά μόνο από τη μια πλευρά του ορθογωνίου (Εικόνα 4.45).

Εικόνα 4.45: Φως περιοχής

Για το φωτισμό της σκηνής χρησιμοποιούμε ένα φως κατεύθυνσης (Εικόνα 4.46). Ρυθμίζουμε τη γωνία με τέτοιο τρόπο ώστε να φωτίζεται η πλευρά των κτιρίων η οποία φαίνεται από τη κάμερα. Αυτό το φως μιμείται το φως του ήλιου, φωτίζει τα πάντα στη σκηνή υπό συγκεκριμένη γωνία.

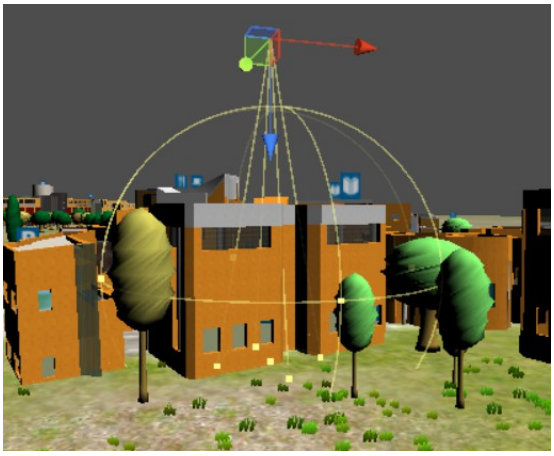


Εικόνα 4.46: Προσθήκη φωτισμού στη σκηνή

Αν θέλουμε να φωτίζονται ελαφρώς και οι άλλες πλευρές των κτιρίων μπορούμε να προσθέσουμε περισσότερα φώτα με διαφορετική γωνία, εμάς όμως μας ικανοποιεί το αποτέλεσμα με το ένα φως, οπότε και για λόγους απόδοσης δε θα φορτώσουμε τη σκηνή με περισσότερα φώτα.

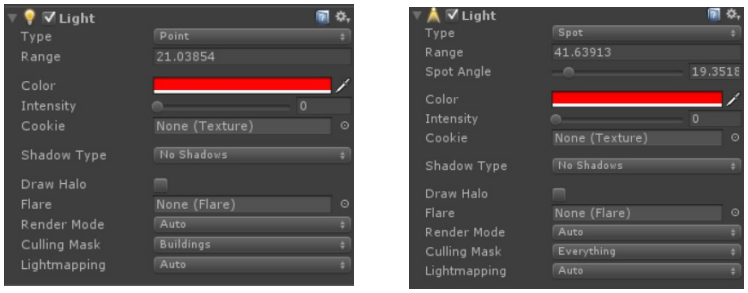
Για κάθε φως που τοποθετούμε στη σκηνή μπορούμε να επιλέξουμε το αν θα δημιουργεί σκιές αλλά και το είδος των σκιών. Εμείς θέλουμε αυτό το φως να δημιουργεί τις σκιές άρα θα επιλέξουμε hard shadows. Αν θέλαμε να προσθέσουμε και άλλα φώτα για να φωτίζονται τα κτίρια και από άλλες γωνίες αλλά δε θέλαμε να δημιουργούνται πολλαπλές σκιές για κάθε μοντέλο, θα έπρεπε να απενεργοποιήσουμε τη δημιουργία σκιών από τα άλλα φώτα.

Όταν επιλέγουμε ένα κτίριο θέλουμε αυτό το κτίριο να φωτίζεται ώστε να ξεχωρίζει από τα υπόλοιπα. Για να πετύχουμε αυτό το εφέ θα χρησιμοποιήσουμε δυο φώτα. Ένα σποτ και ένα φως σημείου τα οποία θα έχουν κόκκινο χρώμα και δε θα δημιουργούν σκιές. Το φως σημείου θέλουμε να εμφανίζεται στο εσωτερικό ενός κτιρίου ενώ το σποτ από πάνω του και να φέγγει την οροφή του (Εικόνα 4.47).



Εικόνα 4.47: Φωτισμός κτιρίου

Θέλουμε τα δυο φώτα να είναι σβηστά και να φωτίζουν ένα κτίριο μόνο όταν το επιλέγουμε και για λίγα δευτερόλεπτα. Για αυτό και τα δυο φώτα ορίζουμε το Intensity = 0 και προσθέτουμε ένα script component το οποίο θα ρυθμίζει τη θέση και την ένταση τους.

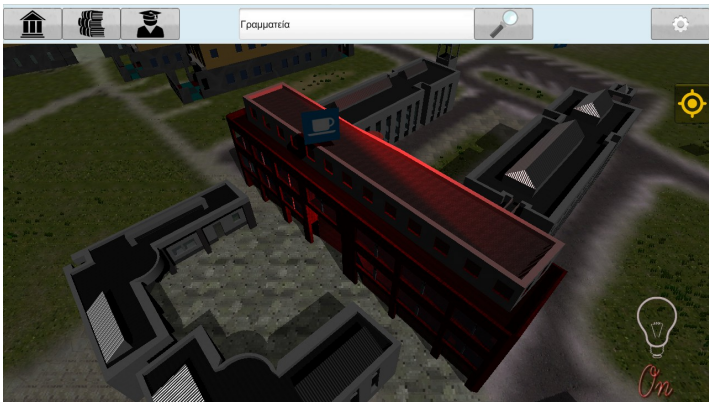


Εικόνα 4.48: Ρύθμιση φωτισμού κτιρίου

Όταν επιλέγουμε ένα κτίριο θα δίνουμε τις συντεταγμένες X,Y του κτιρίου στη συνάρτηση SetLight του κάθε component και το φως θα ανάβει και θα μετακινείται σε αυτές τις συντεταγμένες οπού και θα παραμένει ενώ μειώνεται σταδιακά η ένταση του (Εικόνα 4.49).

```
void Update () {
    if (MyLight.intensity > 0f){
        MyLight.intensity = MyLight.intensity - 0.01f;
    }
}

public void SetLight(float myX, float myZ){
    MyLight.intensity = 8f;
    MyLight.transform.position = new Vector3 (myX, MyLight.transform.position.y, myZ);

}
```

Εικόνα 4.49: Συνάρτηση SetLight

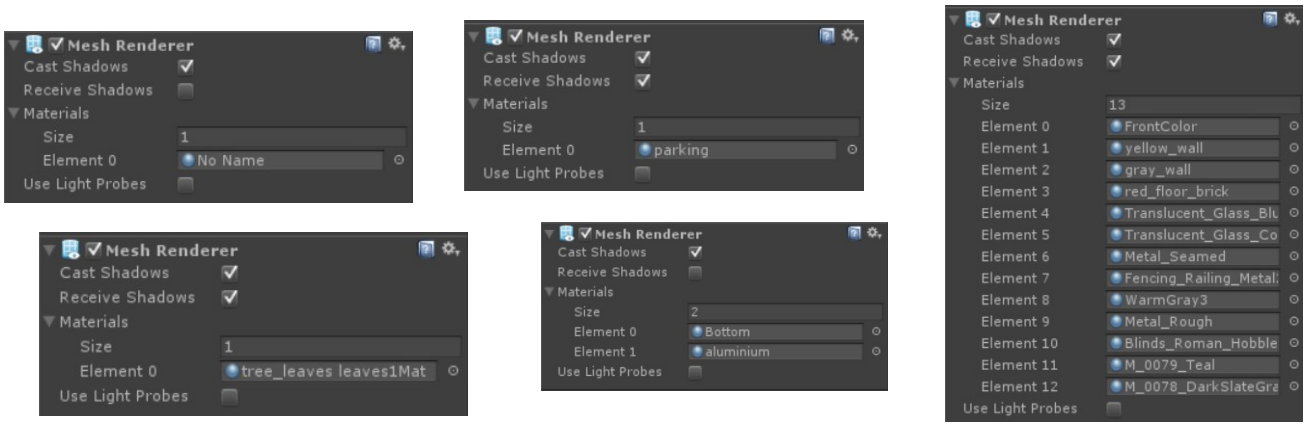


Εικόνα 4.50: Παράδειγμα φωτισμού κτιρίου

Έχοντας ρυθμίσει το φωτισμό της σκηνής μας και το ποια φώτα θα δημιουργούν σκιές το μόνο που μένει να κάνουμε για να δημιουργούνται σκιές στη σκηνή μας είναι να ορίσουμε ποια αντικείμενα θα έχουν σκιά.

Σκιά μπορεί να δημιουργήσει οποιοδήποτε αντικείμενο στη σκηνή μας έχει σαν component έναν mesh renderer. Σε κάθε αντικείμενο το οποίο έχει mesh renderer μπορούμε να επιλέξουμε αν θα δημιουργεί ή θα λαμβάνει σκιές. Το μόνο που χρειάζεται να κάνουμε είναι να επιλέξουμε τον mesh renderer και επιλέγοντας τα κατάλληλα κουτάκια να αλλάξουμε τις ρυθμίσεις του (Εικόνα 4.51).

Αν ένα αντικείμενο αποτελείται από περισσότερους εκ του ενός mesh renderers πρέπει να ρυθμίσουμε τον κάθε ένα ξεχωριστά. Για παράδειγμα οι γλόμποι μας αποτελούνται από δυο mesh renderers, ένα για το πάνω μέρος του και ένα για το στρίφωμα. Επιλέξαμε και οι δυο να δημιουργούν σκιές αλλά να μη δέχονται σκιές. Και αυτό γιατί αφού έχουμε ρυθμίσει τους γλόμπους να φέγγουν με τη χρήση του self illumin shader δε θα φαίνεται ωραίο το να εμφανίζονται σκιές από άλλα αντικείμενα πάνω στους γλόμπους μας οι οποίοι υποτίθεται εκπέμπουν φως.

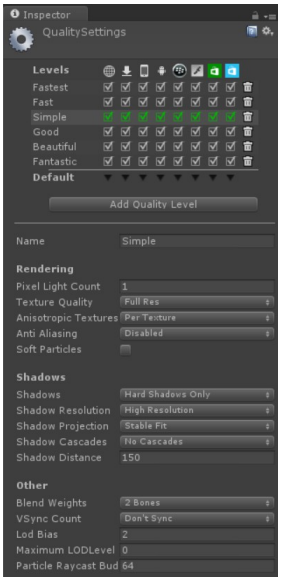


Εικόνα 4.51: Mesh renderers αντικειμένων

Ρυθμίζουμε τις σκιές για κάθε mesh renderer κάθε αντικειμένου στη σκηνή μας, για κάθε κτίριο πινακίδα και δέντρο ώστε να έχουμε το αποτέλεσμα που θέλουμε. Αλλάζοντας τη ρύθμιση απευθείας στα prefabs μας, εφαρμόζεται αυτόματα σε όλα τα συνδεδεμένα αντίγραφα που έχουμε στη σκηνή, χωρίς να χρειάζεται έτσι να αλλάζουμε για παράδειγμα τη ρύθμιση σε κάθε δέντρο ξεχωριστά.

Για να επιλέξουμε αν το Terrain μας θα δημιουργεί σκιές το επιλέγουμε και βρίσκουμε την αντίστοιχη δυνατότητα στις ρυθμίσεις του Terrain στη καρτέλα του inspector.

Τέλος λόγω του ότι η κάμερα μπορεί να βρίσκεται αρκετά μακριά από το σημείο στο οποίο εμφανίζονται οι σκιές πρέπει να αλλάξουμε το Draw Distance των σκιών, το από πόσο μακριά δηλαδή θα είναι ορατές (Εικόνα 4.52). Για να το κάνουμε αυτό ανοίγουμε τις ρυθμίσεις ποιότητας όπου μπορούμε να κάνουμε αρκετές ρυθμίσεις για τις σκιές μας, όπως για παράδειγμα τη ποιότητα ανάλυσης των σκιών ή το τύπο των σκιών που θέλουμε να δημιουργούνται.



Εικόνα 4.52: Ρυθμίσεις ποιότητας γραφικών

ΔΗΜΙΟΥΡΓΊΑ ΠΛΑΤΦΟΡΜΑΣ ΔΙΑΧΕΙΡΙΣΗΣ & ΣΥΝΔΕΣΗ ΜΕ ΕΦΑΡΜΟΓΗ

ΚΕΦΑΛΑΙΟ 5

5.1 Προγραμματισμός με Drupal

Έχοντας σχεδιάσει το περιβάλλον της εφαρμογής μας είμαστε έτοιμοι να συνδέσουμε την εφαρμογή μας με μια online βάση δεδομένων. Είναι απαραίτητο να σχεδιάσουμε τη βάση δεδομένων σε αυτό το σημείο ώστε να μπορέσουμε να συνεχίσουμε το προγραμματισμό της εφαρμογής και την υλοποίηση δυνατοτήτων οι οποίες για να λειτουργήσουν χρειάζονται δεδομένα τα οποία θα αποθηκεύονται σε αυτή τη βάση.

Για τη δημιουργία αυτών των βάσεων θα χρησιμοποιήσουμε τη πλατφόρμα του Drupal. Το Drupal δεν είναι πρόγραμμα σχεδιασμού βάσεων δεδομένων αλλά ιστοσελίδας. Εμείς εκτός από τις βάσεις δεδομένων θέλουμε να υλοποιήσουμε και μια πλατφόρμα διαχείρισης όπου θα μπορούν χρήστες με τα απαραίτητα δικαιώματα να προσθέσουν πληροφορίες όπως μαθήματα και υπαλλήλους ή να επεξεργαστούν τις ήδη υπάρχουσες πληροφορίες εύκολα και χωρίς γνώσης βάσεων δεδομένων.

Θα σχεδιάσουμε λοιπόν αυτή τη πλατφόρμα διαχείρισης και θα χρησιμοποιήσουμε τις βάσεις που θα δημιουργηθούν αυτόματα από το Drupal ώστε να ανακτήσουμε τις απαραίτητες πληροφορίες της εφαρμογής.

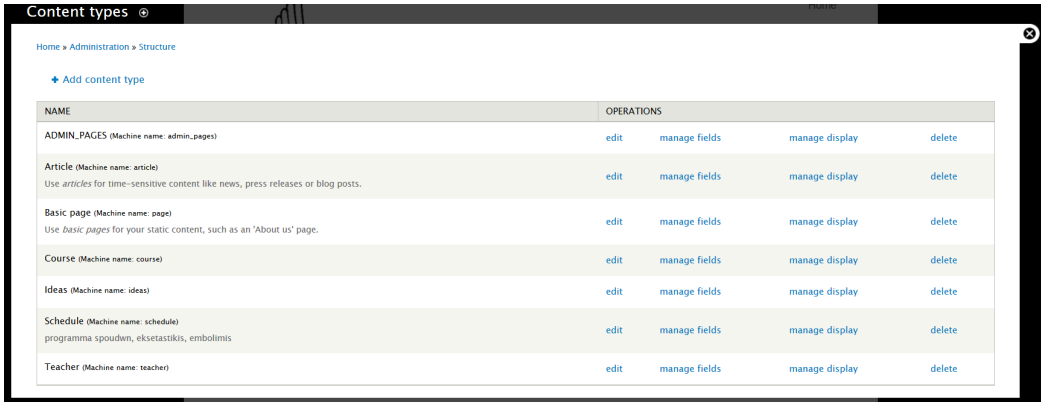
5.1.1 Ρύθμιση Drupal backend

Πρώτο μας βήμα είναι να σχεδιάσουμε το backend του website μας όπου θα μπορούν να συνδεθούν χρήστες της εφαρμογής και ανάλογα τα δικαιώματα τους να επεξεργαστούν είτε πληροφορίες που αφορούν τους ίδιους, όπως για τα μαθήματα τους, το email και το τηλέφωνο τους, ή να επεξεργαστούν και να προσθέσουν νέες πληροφορίες αν έχουν για παράδειγμα δικαιώματα content manager. Τα δικαιώματα στους χρήστες τα καθορίζει ο admin του website, ο οποίος μπορεί με τη χρήστη του backend να προσθέσει ή να αφαιρέσει δικαιώματα από έναν εγγεγραμμένο χρήστη.

Για τη ρύθμιση του backend θα χρησιμοποιήσουμε τις δυνατότητες που μας προσφέρει το Drupal αλλά και modules τα οποία μπορούμε να βρούμε online και μας δίνουν τη δυνατότητα να προσθέσουμε νέες λειτουργίες εύκολα και γρήγορα όπως για παράδειγμα τη σύνδεση του website με τις υπηρεσίες google analytics ή τις δυνατότητες δικαιωμάτων χρηστών που προαναφέραμε.

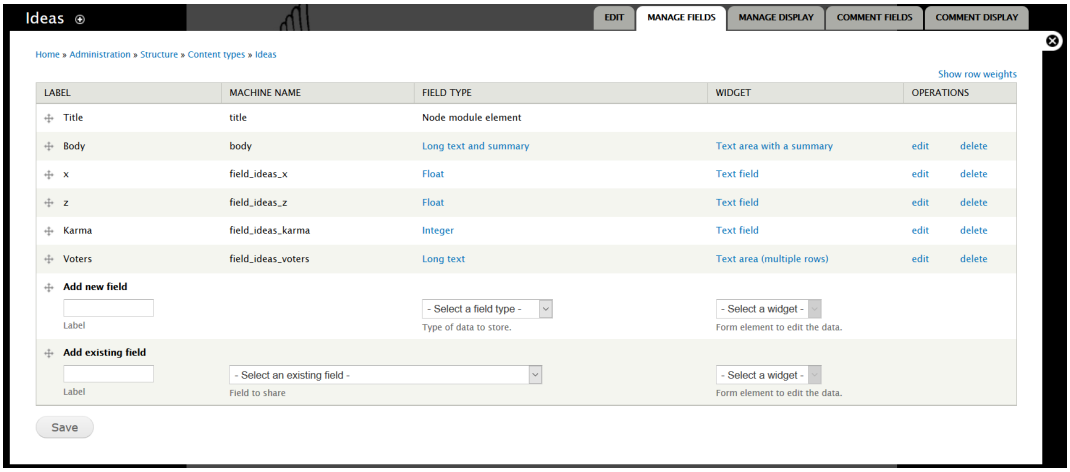
Για να εγκαταστήσουμε ένα module το μόνο που χρειάζεται να κάνουμε είναι να αντιγράψουμε τα αρχεία του στο φάκελο modules στο directory του Drupal. Μόλις το κάνουμε αυτό θα δούμε θα εμφανιστεί στη λίστα των διαθέσιμων modules στην καρτέλα που θα βρούμε κάνοντας κλικ στο αντίστοιχο κουμπί στη πάνω μπάρα. Εκεί μπορούμε να ενεργοποιήσουμε ή να απενεργοποιήσουμε modules και να επεξεργαστούμε τις παραμέτρους τους. Φυσικά μπορούμε να γράψουμε και τα δικά μας modules όπως και έγινε για να πετύχουμε τη διασύνδεση της εφαρμογής με τη βάση δεδομένων.

Τα content types αντιπροσωπεύουν σύνολα δεδομένων που έχουμε. Το Drupal μας προσφέρει κάποια έτοιμα content types αλλά μπορούμε φυσικά να προσθέσουμε και τα δικά μας (Εικόνα 5.1). Για την εφαρμογή μας θα δημιουργήσουμε τέσσερα νέα content types. Θα τα ονομάσουμε Course, Ideas, Schedule και Teacher. Μια καταχώρηση ενός content type, όπως για παράδειγμα ένα καταχωρημένο μάθημα για το type Course, καλείται node.



Εικόνα 5.1: Content Types

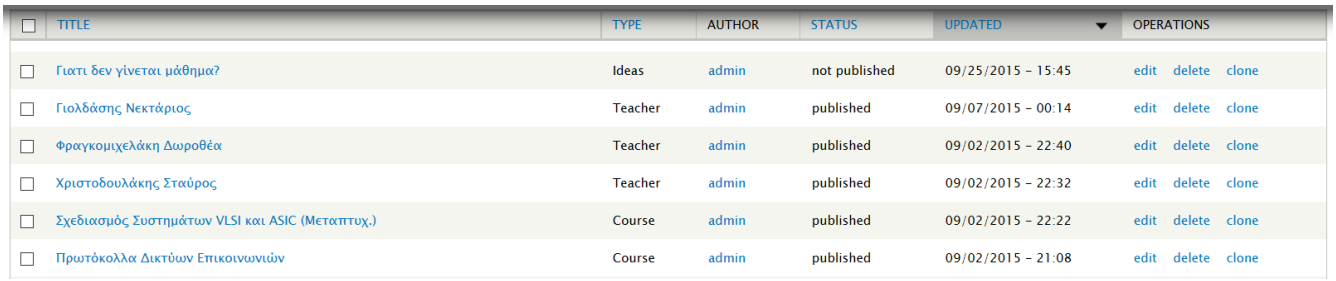
Όταν δημιουργούμε ένα νέο content type, επιλέγουμε τα πεδία που θέλουμε να έχει και το τύπο του κάθε πεδίου (Εικόνα 5.2). Αν θα είναι δηλαδή μια μεταβλητή κειμένου ή ακεραίου. Αν θέλουμε να συσχετίσουμε δύο nodes διαφορετικών content types μεταξύ τους τότε μπορούμε να δηλώσουμε ένα πεδίο σαν node reference.



Εικόνα 5.2: Δημιουργία Content Types

Μόλις αποθηκεύσουμε το νέο content type, μπορούμε πλέον να προσθέσουμε πληροφορίες. Χρησιμοποιώντας την επιλογή “Add content” του drupal μπορούμε με τη βοήθεια του γραφικού περιβάλλοντος να συμπληρώσουμε τα πεδία και να αποθηκεύσουμε το νέο content (Εικόνα 5.3).

Το content θα αποθηκευτεί και θα είναι διαθέσιμο στο μενού contents του drupal, όπου μπορούμε να το επεξεργαστούμε, να αντιγράψουμε ή να διαγράψουμε, πάλι με τη βοήθεια του γραφικού περιβάλλοντος.

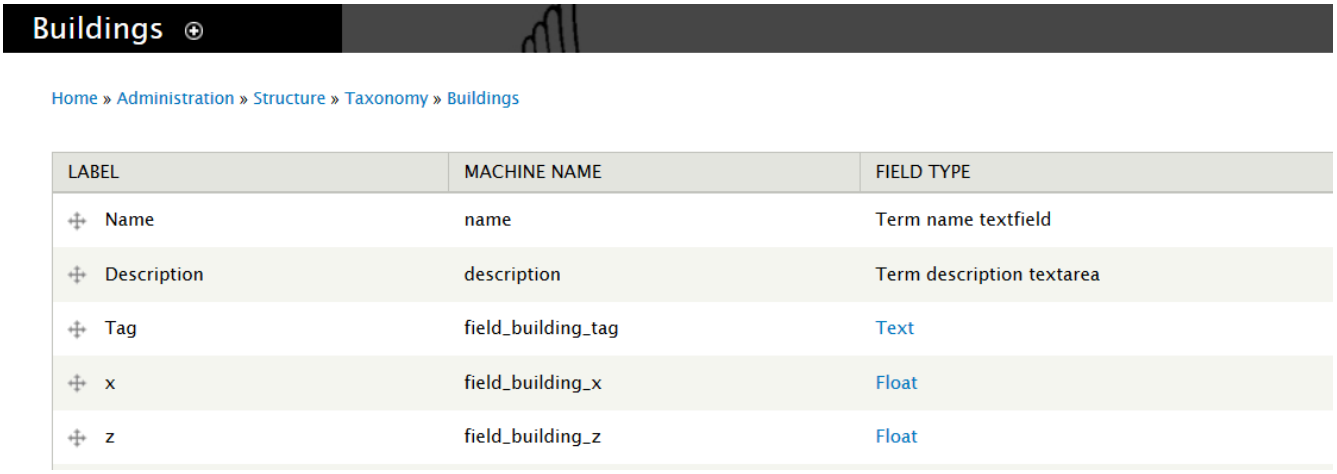


TITLE	TYPE	AUTHOR	STATUS	UPDATED	OPERATIONS
Γιατι δεν γίνεται μάθημα?	Ideas	admin	not published	09/25/2015 - 15:45	edit delete clone
Γιολδάσης Νεκτάριος	Teacher	admin	published	09/07/2015 - 00:14	edit delete clone
Φραγκομιχελάκη Δωροθέα	Teacher	admin	published	09/02/2015 - 22:40	edit delete clone
Χριστοδουλάκης Σταύρος	Teacher	admin	published	09/02/2015 - 22:32	edit delete clone
Σχεδιασμός Συστημάτων VLSI και ASIC (Μεταπτυχ.)	Course	admin	published	09/02/2015 - 22:22	edit delete clone
Πρωτόκολλα Δικτύων Επικοινωνιών	Course	admin	published	09/02/2015 - 21:08	edit delete clone

Εικόνα 5.3: Λίστα nodes

Κατά τη δημιουργία νέου περιεχομένου οι πληροφορίες αποθηκεύονται στην online βάση του drupal, άρα βλέπουμε ότι είναι δυνατόν κάποιος χρήστης χωρίς γνώσεις προγραμματισμού να επεξεργαστεί και να προσθέσει εύκολα πληροφορίες στη βάση μας χάρη στο εύχρηστο περιβάλλον του drupal, αν φυσικά του χορηγήσουμε τα απαραίτητα δικαιώματα για να το κάνει αυτό.

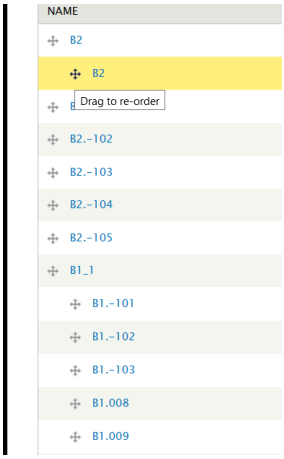
Το Taxonomy είναι μια δυνατότητα που μας παρέχει το drupal για κατηγοριοποίηση πληροφοριών. Θα δημιουργήσουμε τέσσερα νέα Taxonomy. Buildings, Departments, Schedule Type και Tags.



Εικόνα 5.4: Δημιουργία Taxonomy

Μπορούμε πάλι να δημιουργήσουμε τις δικές μας κατηγορίες επιλέγοντας για κάθε κατηγορία τα πεδία τα οποία θέλουμε και το τύπο τους (Εικόνα 5.4) αλλά μας προσφέρει και μια επιπλέον δυνατότητα η οποία δεν είναι διαθέσιμη για τα content types.

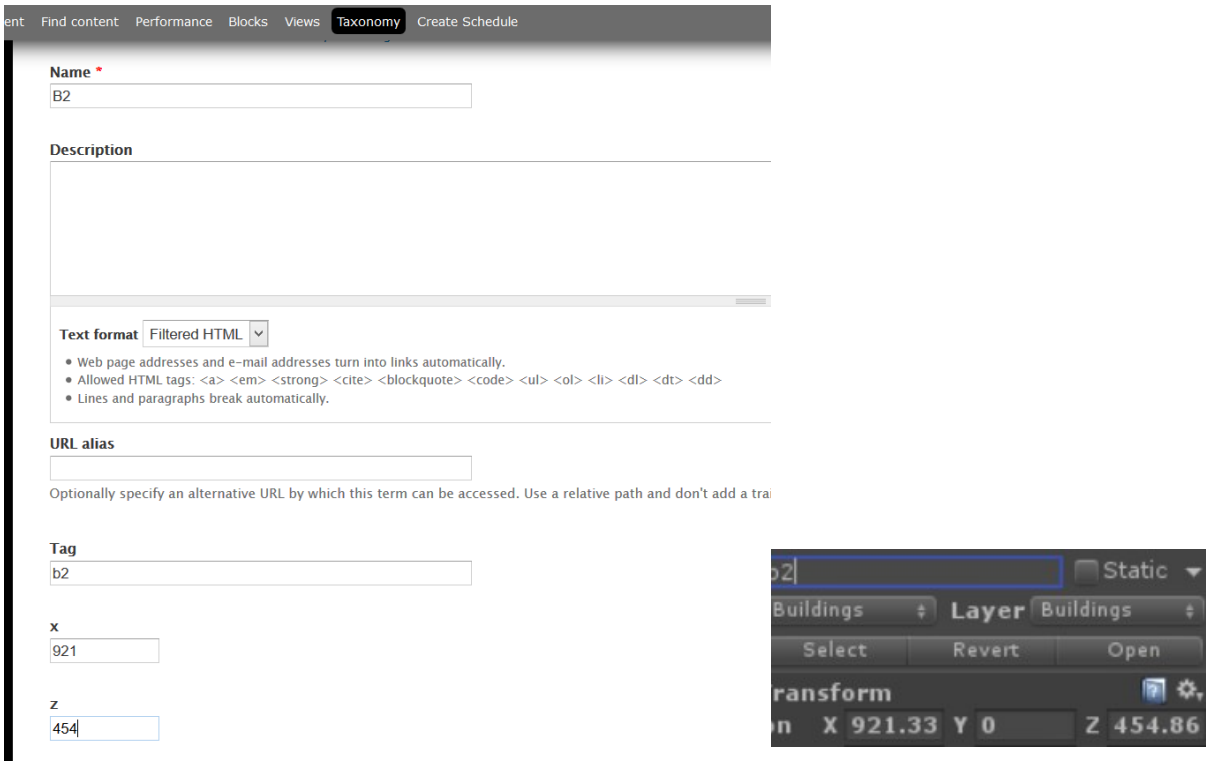
Μπορούμε να ορίσουμε συσχέτιση parent-child μεταξύ δυο στοιχείων ενός taxonomy, μια δυνατότητα που θα μας φανεί ιδιαίτερα χρήσιμη για τη κατηγοριοποίηση κτιρίων και αιθουσών. Για να οριστεί αυτή η συσχέτιση το μόνο που χρειάζεται να κάνουμε είναι να μετακινήσουμε το στοιχείο που θέλουμε να ορίσουμε ως child κάτω και δεξιά από ένα άλλο με το ποντίκι μας (Εικόνα 5.5).



Εικόνα 5.5: Συσχετίσεις parent-child Taxonomy

Ένα Taxonomy μπορεί να χρησιμοποιηθεί σαν μεταβλητή για τα πεδία των Content Types μας. Οπότε μπορούμε έτσι να αντιστοιχίσουμε καθηγητές ή άλλο προσωπικό του πολυτεχνείου με αίθουσες.

Στο Taxonomy με τίτλο Buildings προσθέτουμε ένα πεδίο Tag το οποίο θα χρειαστούμε ώστε να γίνεται η αντιστοίχιση ενός τρισδιάστατου μοντέλου στη σκηνή της εφαρμογής μας με τις πληροφορίες του κτιρίου στη βάση δεδομένων και η τιμή του πεδίου θα είναι το όνομα του μοντέλου στη σκηνή. Ακόμα προσθέτουμε δυο πεδία στα οποία θα αποθηκεύουμε το x και το z του μετασχηματισμού του μοντέλου στη σκηνή, ώστε να έχουμε στη βάση μας τις συντεταγμένες του κτιρίου στη σκηνή (Εικόνα 5.6). Τις συντεταγμένες x και z καλό είναι να τις ορίσουμε αφότου είμαστε σίγουροι ότι δε θα χρειαστεί να μετακινήσουμε το μοντέλο στη σκηνή, αλλιώς δε θα πρέπει να ξεχάσουμε να αλλάξουμε και τη πληροφορία στη βάση μας. Λόγω των τριών αυτών πεδίων τα οποία για να συμπληρωθούν απαιτούν γνώσεις σχετικά με τη θέση και το όνομα των μοντέλων στη σκηνή, η προσθήκη ενός νέου κτιρίου είναι η μόνη δυνατότητα για την οποία ο χρήστης χρειάζεται πρόσβαση στην εφαρμογή και θέλει τη βοήθεια του προγραμματιστή. Αυτό είναι κάτι που δε μπορεί να αποφευχθεί, αλλά η προσθήκη νέων κτιρίων είναι κάτι το οποίο δεν αναμένεται να γίνεται συχνά αφού αυτό συνεπάγεται το κτίσιμο νέων εγκαταστάσεων στο πολυτεχνείο. Η προθήκη κάθε άλλης πληροφορίας, ακόμα και αιθουσών ενός κτιρίου δεν απαιτεί τη παρέμβαση του προγραμματιστή.

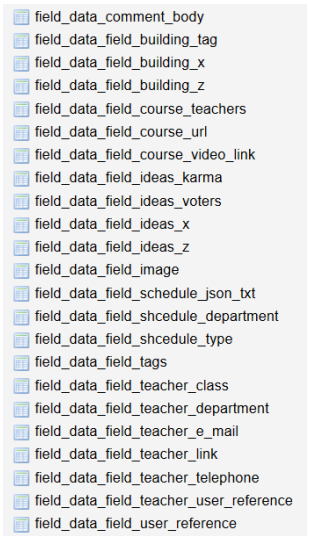


Εικόνα 5.6: Καταχώρηση κτιρίου

5.1.2 Βάσεις Δεδομένων drupal

Είδαμε πως μπορούμε να σχεδιάσουμε το backend ώστε να ικανοποιεί τις απαιτήσεις της εφαρμογής και να μπορούν χρήστες χωρίς γνώσεις προγραμματισμού να προσθέσουν ή να επεξεργαστούν εύκολα το περιεχόμενο. Κάθε node ή στοιχείο taxonomy που προσθέτουμε καταχωρείται αυτόματα στη βάση δεδομένων μας. Για να επεξεργαστούμε ή να δούμε τις βάσεις θα χρησιμοποιήσουμε το εργαλείο phpmyadmin.

Η δομή των βάσεων δεν είναι αυτή που θα περιμέναμε. Δε δημιουργούνται δηλαδή ξεχωριστοί πίνακες για κάθε content type όπως ίσως κάναμε αν σχεδιάζαμε τη βάση χειροκίνητα, αλλά όλα τα nodes των content types αποθηκεύονται στον ίδιο πίνακα με όνομα node (Εικόνα 5.7). Οι διαφορές που μπορεί να υπάρχουν μεταξύ των content types όσον αφορά τα πεδία τους, έχουν ως αποτέλεσμα στο πίνακα node να αποθηκεύονται μόνο κάποιες βασικές πληροφορίες όπως το node id, το content type στο οποίο ανήκει το node, ο τίτλος του ο οποίος μπορεί να είναι είτε ο τίτλος μιας δημοσίευσης χρήστη είτε το όνομα ενός υπαλλήλου ή μαθήματος και κάποιες άλλες πληροφορίες.



nid	vid	type	language	title	uid	status	created	changed	comment
1	1	ideas	und	Γιατι δεν γίνεται μάθημα?	1	1	1437500604	1437593071	2
2	2	ideas	und	Sup dawg	2	1	1426787601	1426787601	2
3	3	teacher	und	Αγγελάκης Δημήτριος	1	1	1427141573	1441119037	1
4	4	teacher	und	Μανιά Αικατερίνη	1	1	1427142634	1441136341	1
5	5	teacher	und	Γιολδάσης Νεκτάριος	1	1	1427142773	1441124099	1
6	6	course	und	Γυμναστική	1	1	1427142815	1428162985	1

Εικόνα 5.7: Πίνακας node

Για κάθε πεδίο ενός content type που έχουμε προσθέσει εμείς, δημιουργείται ένας ξεχωριστός πίνακας ο οποίος σε μια στήλη entity id μας δίνει το node id με το οποίο γίνεται η αντιστοίχιση της πληροφορίας που αφορά ο πίνακας (Εικόνα 5.8).

bundle	deleted	entity_id	revision_id	language	delta	field_course_teachers_nid
course	0	6	6	und	0	5
course	0	7	7	und	0	3
course	0	7	7	und	1	4
course	0	424	424	und	0	99
course	0	425	425	und	0	126
course	0	426	426	und	0	141
course	0	427	427	und	0	252
course	0	428	428	und	0	414
course	0	429	429	und	0	414

entity_type	bundle	deleted	entity_id	revision_id	language	delta	field_course_url_value
node	course	0	7	7	und	0	http://courses.ced.tuc.gr/
node	course	0	445	445	und	0	http://courses.ced.tuc.gr/
node	course	0	455	455	und	0	http://courses.ced.tuc.gr/
node	course	0	522	522	und	0	http://courses.ced.tuc.gr/

Εικόνα 5.8: Πίνακες custom πεδίων node

Η πληροφορία μπορεί να είναι ακόμα και ένα node id, αν για παράδειγμα το πεδίο το οποίο αφορά ο πίνακας είναι τύπου node reference, όπως στην αντιστοίχιση προσώπων με αίθουσες.

Βλέπουμε λοιπόν ότι οι βάσεις δεδομένων που δημιουργεί αυτόματα το drupal έχουν αρκετά πολύπλοκη δομή αλλά αυτό είναι κάτι το οποίο μπορούμε να παραβλέψουμε, αφού η χρήση του drupal κάνει την υλοποίηση δυνατοτήτων όπως είναι η εγγραφή και σύνδεση χρηστών με διαφορετικά δικαιώματα πρόσβασης, η δημοσίευση και επεξεργασία πληροφοριών και γενικά το στήσιμο μιας πλατφόρμας διαχείρισης πολύ πιο απλή.

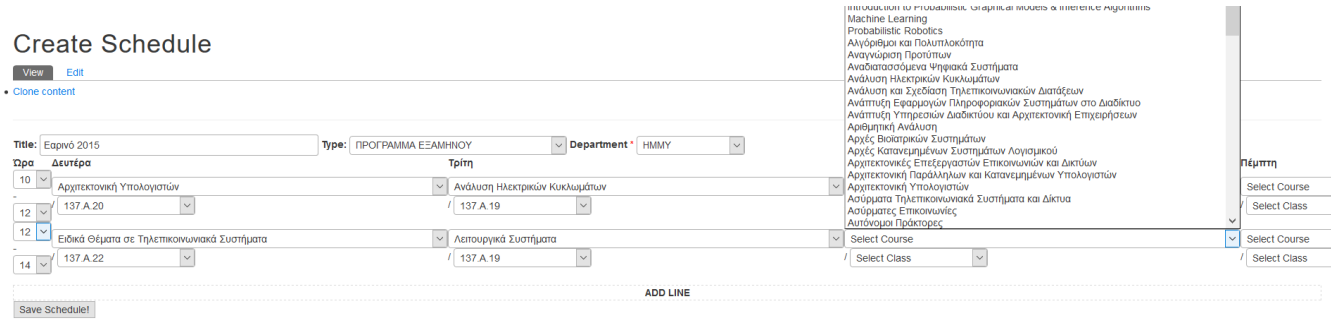
5.1.3 Δημιουργία προγράμματος μαθημάτων

Στο content type Schedule έχουμε δηλώσει ένα πεδίο με όνομα json txt και τύπο κειμένου. Σε αυτό το πεδίο θα αποθηκεύουμε σε μορφή JSON το ωρολόγιο πρόγραμμα των μαθημάτων της κάθε σχολής. Έτσι το πρόγραμμα κάθε σχολής θα αποθηκεύεται σε ένα node από όπου θα μπορούμε να ανακτήσουμε αυτό αυτό το κείμενο, να το αποκωδικοποιήσουμε και να το χρησιμοποιήσουμε στην εφαρμογή μας.

Για τη δημιουργία της πλατφόρμας δημιουργίας προγράμματος μαθημάτων θα δημιουργήσουμε δυο αρχεία.

Στο αρχείο page--node--24.tpl.php δημιουργούμε το θέμα το οποίο θέλουμε να έχει η σελίδα της δημιουργίας προγράμματος. Λόγω του ότι κατά τη δημιουργία ενός node για το content type Schedule θέλουμε να γίνονται κάποιοι έλεγχοι, να έχουμε κουμπιά τα οποία θα προσθέτουν επιπλέον πεδία αλλά και να αποθηκεύουμε τις πληροφορίες από τα πεδία αυτά σε μορφή JSON θα πρέπει να δημιουργήσουμε τη δικιά μας φόρμα, σε αντίθεση με τα απλά content types στα οποία χρησιμοποιούμε την φόρμα που μας παρέχει η επιλογή add content στη πλατφόρμα του drupal.

Στη φόρμα αυτή θα προσθέτουμε γραμμές στις οποίες θα επιλέγουμε διάρκεια, μαθήματα και αίθουσες για κάθε ημέρα της εβδομάδας από αναδυόμενα μενού. Με τα αντίστοιχα κουμπιά θα μπορούμε να προσθέσουμε επιπλέον γραμμές ή να αποθηκεύσουμε το πρόγραμμα σε ένα νέο node (Εικόνα 5.9).



Εικόνα 5.9: Δημιουργία προγράμματος μαθημάτων

Για τη λειτουργικότητα των αναδυόμενων μενού αλλά και των κουμπιών δημιουργήσαμε και χρησιμοποιήσαμε ένα αρχείο javascript με όνομα scheduler.js

Σε αυτό το αρχείο εκτός από τους ελέγχους για τα αναδυόμενα μενού, όπως για παράδειγμα αν μια αίθουσα χρησιμοποιείται ήδη το επιλεγμένο δίωρο ώστε να μη μπορεί ο χρήστης να την επιλέξει, γίνεται και η εξαγωγή του προγράμματος σε μορφή JSON (Εικόνα 5.10).

```
schedule: {
    "Monday":[
        {"from_time":"","to_time":"","nid":"", "courseName":"", "tid":"", "className":""},
    ],
    "Tuesday":[
        {"from_time":"","to_time":"","nid":"", "courseName":"", "tid":"", "className":""},
    ],
    "Wednesday":[
        {"from_time":"","to_time":"","nid":"", "courseName":"", "tid":"", "className":""},
    ],
    "Thursday":[
        {"from_time":"","to_time":"","nid":"", "courseName":"", "tid":"", "className":""},
    ],
    "Friday":[
        {"from_time":"","to_time":"","nid":"", "courseName":"", "tid":"", "className":""},
    ],
    "Saturday":[
        {"from_time":"","to_time":"","nid":"", "courseName":"", "tid":"", "className":""},
    ],
    "Sunday":[
        {"from_time":"","to_time":"","nid":"", "courseName":"", "tid":"", "className":""},
    ],
},
```

Εικόνα 5.10: Μορφή JSON

5.2 Σύνδεση εφαρμογής με βάση δεδομένων

Έχοντας ετοιμάσει τη πλατφόρμα διαχείρισης του Drupal ώστε να μπορούν οι χρήστες να προσθέτουν και να βλέπουν πληροφορίες, αλλά και γνωρίζοντας πλέον το τρόπο με τον οποίο αποθηκεύονται οι πληροφορίες στη βάση δεδομένων, θα προχωρήσουμε στη σύνδεση της βάσης δεδομένων με την εφαρμογή.

5.2.1 Drupal – Δημιουργία TucApp API module

Το Unity μπορεί να πάρει δεδομένα από μια ιστοσελίδα σε μορφή string. Για να επιστρέψουμε λοιπόν πληροφορίες από τη βάση δεδομένων στην εφαρμογή μας θα πρέπει να τα κωδικοποιούμε σε αυτή τη μορφή ώστε να μπορεί να τα χρησιμοποιήσει το Unity. Αν για παράδειγμα θέλουμε να επιστρέψουμε στο Unity όλες τις αίθουσες με τα id τους θα πρέπει να τις κωδικοποιήσουμε σε ένα string με τέτοιο τρόπο ώστε το Unity να μπορεί να πάρει τις πληροφορίες.

Για αυτό το σκοπό θα δημιουργήσουμε το tucapp.module και το ονομάζουμε TucApp API.

Η δημιουργία του δικού μας module για τη διασύνδεση του website με την εφαρμογή μας επιτρέπει όχι μόνο να ενεργοποιούμε και να απενεργοποιούμε το module από τη πλατφόρμα διαχείρισης αλλά και να μεταφέρουμε το module σε άλλες ιστοσελίδες που βασίζονται στο drupal.

Θα ζητάμε δεδομένα από την ιστοσελίδα μας με δύο τρόπους. Είτε θα ζητάμε να μας επιστραφεί μια προκαθορισμένη λίστα δεδομένων, όπως για παράδειγμα όλα τα μαθήματα, ή θα ζητάμε πληροφορίες για κάτι συγκεκριμένο, όπως για ένα μάθημα.

Για τη πρώτη περίπτωση θα επισκεπτόμαστε ένα url της παρακάτω μορφής ώστε να πάρουμε τις πληροφορίες:

http://www.tucapp.gr/tucappapi/courses

Όπου θα αλλάζουμε τη λέξη courses με το είδος των πληροφοριών που θέλουμε να αντλήσουμε. Για παράδειγμα αντί για courses θα μπορούσαμε να χρησιμοποιήσουμε classes ή teachers. Για τη δεύτερη περίπτωση θα χρησιμοποιούμε ένα url της παρακάτω μορφής:

http://www.tucapp.gr/tucappapi/course/?a[0]=508

Όπου εκτός από τη λέξη κλειδί για το είδος της πληροφορίας, χρησιμοποιούμε και μια παράμετρο. Στο παραπάνω παράδειγμα, το 508 είναι το id ενός μαθήματος στη βάση δεδομένων. Θα μπορούσαμε αν θέλαμε να ψάξουμε με το όνομα του μαθήματος και να χρησιμοποιήσουμε αυτό σαν παράμετρο στο url.

Υλοποιώντας το module σε php με τις κατάλληλες συνθήκες (Εικόνα 5.11) μπορούμε λοιπόν να σχεδιάσουμε συναρτήσεις, οι οποίες ανάλογα το url που επισκέπτεται η εφαρμογή, θα βρίσκουν τις ζητούμενες πληροφορίες, θα τις κωδικοποιούν σε ένα string και θα το εμφανίζουν στο url ώστε να μπορεί να το χρησιμοποιήσει η εφαρμογή.

```
if(arg(0)=="tucappapi"){
    switch (arg(1)) {
        case "login":
            return2unity(getUser($arg[0]));
            break;
        case "register":
            return2unity(regUser($arg[0]));
            break;
        case "classes":
            return2unity(getClasses());
            break;
        case "getClass":
            return2unity(getClass($arg[0]));
            break;
        case "courses":
            return2unity(getCourses());
            break;
        case "course":
            return2unity(getCourse($arg[0]));
            break;
        case "teachers":
            return2unity(getTeachers());
            break;
        case "teacher":
            return2unity(getTeacher($arg[0]));
            break;
        case "ideas":
```

Εικόνα 5.11: Επιλογή συνάρτησης ανάκτησης

Η δομή των συναρτήσεων που ανακτούν τις πληροφορίες είναι σχετικά απλή. Εκτελούν ένα ερώτημα στη βάση δεδομένων, κάνουν push τα αποτελέσματα σε ένα array και το επιστρέφουν.

Σε αυτό το σημείο παρατηρούμε το μειονέκτημα των βάσεων δεδομένων που δημιουργεί το drupal. Μπορεί ένα ερώτημα βάσης να είναι απλό, όπως στη περίπτωση της λίστας μαθημάτων που χρειαζόμαστε πληροφορίες οι οποίες βρίσκονται όλες στον πίνακα node της βάσης, αλλά αν θέλουμε να επιστραφούν πληροφορίες από περισσότερους πίνακες θα χρειαστεί να χρησιμοποιήσουμε joins στα ερωτήματα μας, όπως στη περίπτωση ενός μαθήματος (Εικόνα 5.12). Αυτό συμβαίνει γιατί όπως είδαμε, για κάθε custom πεδίο που προσθέσαμε στα content types και taxonomy, δημιουργήθηκε ένας ξεχωριστός πίνακας. Χρησιμοποιούμε left outer join γιατί μπορεί κάποιο από τα πεδία να είναι κενό, αλλά παρόλα αυτά θέλουμε να επιστραφεί ο κενός χαρακτήρας ώστε να γνωρίζει η εφαρμογή μας ότι δεν υπάρχει πληροφορία για το συγκεκριμένο πεδίο.

```php
function getCourse($nid){
    $nid = $nid[0];
    $query="SELECT
            n.nid AS ID,
            n.title AS Course,
            db.body_value AS Description,
            ct.field_course_teachers_nid AS TeacherID,
            curl.field_course_url_value AS Website,
            cvl.field_course_video_link_value AS Videos,
            nd.title AS Teacher,
            nd.status AS teacherStat
            FROM `node` n
            LEFT OUTER JOIN field_revision_field_course_teachers ct
            ON n.nid = ct.entity_id
            LEFT OUTER JOIN field_data_body db
            ON n.nid = db.entity_id
            LEFT OUTER JOIN node nd
            ON nd.type = 'teacher' AND nd.nid = ct.field_course_teachers_nid
            LEFT OUTER JOIN field_data_field_course_url curl
            ON n.nid = curl.entity_id
            LEFT OUTER JOIN field_data_field_course_video_link  cvl
            ON n.nid = cvl.entity_id
            WHERE n.nid=".$nid." and n.type='course'
        ";

    $result = db_query($query);
    $course = array();
    foreach($result as $res){
        array_push($course,$res);
    }

    return $course;
}
```

```php
function getCourses(){

    $query = "
        SELECT node.nid AS nid, node.title AS Course
        FROM
        node node
        WHERE (( (node.status = '1') AND (node.type IN  ('course')) ))
        ORDER BY node.title
    ";
    $result = db_query($query);
    $courses = array();
    foreach($result as $course){
        array_push($courses,$course);
    }
    return $courses;

}
```

Εικόνα 5.12: Συναρτήσεις ανάκτησης πληροφοριών

Για περιπτώσεις όπου θέλουμε ο χρήστης της εφαρμογής να προσθέσει κάτι στη βάση δεδομένων γράφουμε συναρτήσεις όπου παίρνοντας τις παραμέτρους του url τις οποίες στέλνει ο χρήστης, δημιουργούμε νέα nodes και αποθηκεύουμε τις πληροφορίες των παραμέτρων στη βάση δεδομένων (Εικόνα 5.13). Έτσι ο χρήστης μπορεί να δημοσιεύει ιδέες με τη μορφή γλόμπων, να ψηφίζει άλλες δημοσιεύσεις ή να σχολιάζει και οι πληροφορίες να είναι άμεσα διαθέσιμες στους υπόλοιπους χρήστες.

```php
function putIdea($args){
    $uid = $args[0];
    $x = $args[1];
    $z = $args[2];
    $title = $args[3];
    $desc = $args[4];

    $node = new stdClass();
    $node->title = $title;
    $node->type = "ideas";
    node_object_prepare($node); // Sets some defaults. Invokes hook_prepare() and hook_node_prepare().
    $node->language = LANGUAGE_NONE; // Or e.g. 'en' if locale is enabled
    $node->uid = $uid;
    $node->status = 1; //(1 or 0): published or not
    $node->promote = 0; //(1 or 0): promoted to front page
    $node->comment = 2; // 0 = comments disabled, 1 = read only, 2 = read/write
    $node->created = !empty($node->date) ? strtotime($node->date) : REQUEST_TIME;

    $node->body[$node->language][0]['value'] = $desc;
    $node->body[$node->language][0]['summary'] = text_summary($desc);
    $node->body[$node->language][0]['format'] = 'filtered_html';

    $node->field_ideas_x[$node->language][0]['value'] = $x;
    $node->field_ideas_z[$node->language][0]['value'] = $z;


    $node = node_submit($node); // Prepare node for saving
    node_save($node);
}
```

Εικόνα 5.13: Συνάρτηση αποθήκευσης πληροφοριών

Όπως προαναφέρθηκε, το Unity μπορεί να πάρει πληροφορίες από μια ιστοσελίδα μόνο σε μορφή κειμένου. Πρέπει να μορφοποιήσουμε το array το οποίο δημιουργούν οι συναρτήσεις ανάκτησης πληροφοριών και να τις προβάλουμε με τη μορφή ενός string στην ιστοσελίδα. Για αυτό το λόγο στέλνουμε το array κάθε συνάρτησης ανάκτησης στη συνάρτηση retyrn2unity όπου χρησιμοποιούμε το σύμβολο $ για να ξεχωρίσουμε το όνομα από τη τιμή ενός πεδίου του αποτελέσματος, το σύμβολο | για να ξεχωρίσουμε τα πεδία ενός αποτελέσματος και το σύμβολο % για να ξεχωρίσουμε τα αποτελέσματα ενός querry στη περίπτωση που έχουν επιστρέψει πάνω από ένα (Εικόνα 5.14). Το string που δημιουργείται εκτυπώνεται στην ιστοσελίδα μας.

```
function return2unity($object){
    //die(var_dump($object));
    $daString = "";
    foreach($object as $obj){
        foreach($obj as $key=>$value){
            $daString .=$key."$".$value."|";
        }
        $daString = substr($daString, 0, -1);
        $daString .="%";
    }
    $daString = substr($daString, 0, -1);
    print $daString;
    die();
    //return $daString;
}
```

Εικόνα 5.14: Συνάρτηση κωδικοποίησης πληροφορίας

Η χρήση αυτών των τριών απλών χαρακτήρων μπορεί να δημιουργήσει προβλήματα κατά την αποκωδικοποίηση, αν κάποιος από αυτούς τους χαρακτήρες χρησιμοποιηθεί στη τιμή ενός πεδίου. Σε αυτή τη περίπτωση θα μπορούσαμε να χρησιμοποιήσουμε ακολουθίες χαρακτήρων για τη κωδικοποίηση, αλλά αυτό είναι κάτι το οποίο δε θα μας απασχολήσει στα πλαίσια αυτής της διπλωματικής.

Παρακάτω βλέπουμε παράδειγμα της κωδικοποίησης ενός αποτελέσματος όπως εμφανίζεται στο url http://www.tucapp.gr/tucappapi/course/?a[0]=508

ID$508|Course$Introduction to Probabilistic Graphical Models & Inference Algorithms | Description$ΤΗΛ000|TeacherID$252|Website$|Videos$|Teacher$Μπλέτσας Άγγελος|teacherStat$1

Παρατηρούμε ότι τα πεδία Website και Videos δεν έχουν τιμή. Αν στο querry ανάκτησης δεν είχαμε χρησιμοποιήσει left outer join δε θα παίρναμε αυτό το αποτέλεσμα.

Στη περίπτωση που επιστρέφουν πάνω από ένα αποτελέσματα για ένα querry εμφανίζονται όπως στο παράδειγμα του url http://www.tucapp.gr/tucappapi/getClass/?a[0]=72

tid$72|name$Δ2|description$Τεχνική Υπηρεσία|teacher_id$78|teacher_name$Αχιλλέως Γεώργιος | x$970|z$305|teacherStat$1%tid$72|name$Δ2|description$Τεχνική Υπηρεσία|teacher_id$153| teacher_name$Καπετανάκης Γεώργιος |x$970|z$305|teacherStat$1%tid$72|name$Δ2| description$Τεχνική Υπηρεσία|teacher_id$306|teacher_name$Παρτσακουλάκη Μαρία |x$970|z$305| teacherStat$1%tid$72|name$Δ2|description$Τεχνική Υπηρεσία|teacher_id$312| teacher_name$Πατεράκης Κωνσταντίνος |x$970|z$305|teacherStat$1%tid$72|name$Δ2| description$Τεχνική Υπηρεσία|teacher_id$321|teacher_name$Πετρίδου Βαρβάρα|x$970|z$305| teacherStat$1%tid$72|name$Δ2|description$Τεχνική Υπηρεσία|teacher_id$372| teacher_name$Τζανάκης Γεώργιος |x$970|z$305|teacherStat$1%tid$72|name$Δ2|description$Τεχνική Υπηρεσία|teacher_id$374|teacher_name$Τζούγκαρης Γεώργιος |x$970|z$305|teacherStat$1%tid$72| name$Δ2|description$Τεχνική Υπηρεσία|teacher_id$385|teacher_name$Τσιναράκης Θεόδωρος|x$970| z$305|teacherStat$1

Παρατηρούμε ότι κάποιες πληροφορίες που είναι κοινές για κάποιο αποτέλεσμα επαναλαμβάνονται, αυτό δε μας απασχολεί όμως καθώς η ομοιομορφία στα πεδία των αποτελεσμάτων θα μας διευκολύνει κατά την αποκωδικοποίηση.

5.2.2 Unity3d- Ανάκτηση και αποκωδικοποίηση πληροφοριών βάσης δεδομένων

Έχοντας υλοποιήσει την ανάκτηση και εμφάνιση πληροφοριών στην ιστοσελίδα μας, το επόμενο βήμα είναι η διασύνδεση με την εφαρμογή, ώστε να εμφανίζονται οι πληροφορίες στο χρήστη. Θέλουμε να καλούμε δυο ειδών url, απλά και με παραμέτρους. Αυτό μπορούμε να το πετύχουμε με τη χρήση των κλάσεων WWW και WWWForm που μας παρέχει το Unity για απλή σύνδεση με ιστοσελίδες. Δημιουργούμε το αρχείο Connector.cs το οποίο θα περιέχει δυο συναρτήσεις, μια για κάθε περίπτωση. Η πρώτη συνάρτηση αφορά τη κλήση απλού url με τη χρήστη της βοηθητικής κλάσης WWW και είναι τύπου IEnumerator. (Εικόνα 5.15) Δηλώνοντας τη συνάρτηση σαν IEnumerator μπορούμε να την εκτελέσουμε σαν Coroutine. Θα μπορεί η συνάρτηση να αναστείλει την εκτέλεση της και δε θα χρειάζεται να ολοκληρωθεί κατά τη διάρκεια ενός frame. Αυτή η δυνατότητα είναι απαραίτητη, καθώς μπορεί ο χρόνος σύνδεσης με την ιστοσελίδα να διαφέρει, ανάλογα το δίκτυο του χρήστη.

```
public Dictionary<string, List<string>> information;

public IEnumerator GetWithOneArg (string function, System.Action<Dictionary<string, List<string>>> result) {
    url = "http://www.tucapp.gr/tucappapi/" + function;
    WWW www = new WWW (url);
    yield return www;
    urlText = www.text;

    yield return StartCoroutine(Camera.main.GetComponent<Parser>().parseString(urlText, value => information = value));

    yield return null;
    result (information);
}
```

Εικόνα 5.15: Συνάρτηση GetWithOneArg

Σαν ορίσματα η συνάρτηση παίρνει ένα string το οποίο αντιπροσωπεύει το αναγνωριστικό της κάθε συνάρτησης του module που φτιάξαμε, για παράδειγμα courses για το url http://www.tucapp.gr/tucappapi/courses , και επιστρέφει μια μεταβλητή τύπου Dictionary<string, List<string>>

Το dictionary είναι μια δυνατότητα που μας παρέχει το .NET Framework και αντιπροσωπεύει μια συλλογή από κλειδιά και τιμές. Τα κλειδιά του είναι τύπου string ενώ οι τιμές τύπου μονοδιάστατης λίστας όπου τα στοιχεία της είναι strings.

Αν θέλαμε να αναπαραστήσουμε σχηματικά ένα dictionary θα μπορούσαμε να το φανταστούμε το πίνακα της εικόνας 5.16. Σε αυτό το παράδειγμα έχουμε τρία κλειδιά και το κάθε κλειδί έχει τέσσερις τιμές. Αυτό δηλαδή αντιπροσωπεύει τη περίπτωση όπου ένα querry έχει επιστρέψει τέσσερα αποτελέσματα και με entry.Value[i] μπορούμε να προσπελάσουμε κάθε τιμή.

String Keys	List<string> Values
Key1	entry.Value[0] Value1 entry.Value[1] Value2 entry.Value[2] Value3 entry.Value[3] Value4
Key2	entry.Value[0] Value5 entry.Value[1] Value6 entry.Value[2] Value7 entry.Value[3] Value8
Key3	entry.Value[0] Value9 entry.Value[1] Value10 entry.Value[2] Value11 entry.Value[3] Value12

Εικόνα 5.16: Πίνακας απεικόνισης Dictionary

Για τη κλήση url με παραμέτρους, θα χρησιμοποιήσουμε τη κλάση WWWForm. Η κλήση της WWWForm γίνεται μέσω της συνάρτησης PostForm (Εικόνα 5.17). Η διαφορά των δυο συναρτήσεων είναι ότι η PostForm δέχεται σαν όρισμα και ένα string[]. Με αυτό το string[] θα δίνουμε τις παραμέτρους στο website. Τις παραμέτρους αυτές τις προσθέτουμε σε μια WWWForm.

```
public Dictionary<string, List<string>> FormReply;

public IEnumerator PostForm (string indicator, string[] data, System.Action<Dictionary<string, List<string>>> result) {
    WWWForm form = new WWWForm();

    int numberOfData = data.Length;
    for (int i = 0; i < numberOfData ; i++) {
        form.AddField ("a[]" , data[i]);
    }

    WWW webRequest = new WWW("http://www.tucapp.gr/tucappapi/" + indicator+"/", form);
    yield return webRequest;

    yield return StartCoroutine(Camera.main.GetComponent<Parser>().parseString(webRequest.text, value => FormReply = value));

    result (FormReply);

}
```

Εικόνα 5.17: Συνάρτηση PostForm

Αυτή τη φόρμα θα χρησιμοποιήσουμε ώστε να σχηματίσουμε το σύνδεσμο που θα μας επιστρέψει το string το οποίο θα αποκωδικοποιήσουμε για να χρησιμοποιήσουμε στην εφαρμογή. Και αυτή η συνάρτηση χρειάζεται να αποκωδικοποιεί τις πληροφορίες σε μορφή dictionary ώστε να μπορέσουμε να τις χρησιμοποιήσουμε.

Η μετατροπή του string που πήραμε από το url, σε Dictionary, θα γίνει με την εκτέλεση της Coroutine του Parser, parsestring (Εικόνα 5.18), όπου δίνουμε σαν όρισμα το string και η συνάρτηση που εκτελείται δημιουργεί και επιστρέφει με Lambda Expressions το Dictionary. Τα Lambda Expressions είναι μια ακόμα δυνατότητα που προσφέρει το .NET Framework.

Έπειτα η Coroutine ανάκτησης πληροφοριών επιστρέφει πάλι με Lambda Expressions το Dictionary στο script το οποίο τη κάλεσε ώστε να χρησιμοποιηθούν οι πληροφορίες.

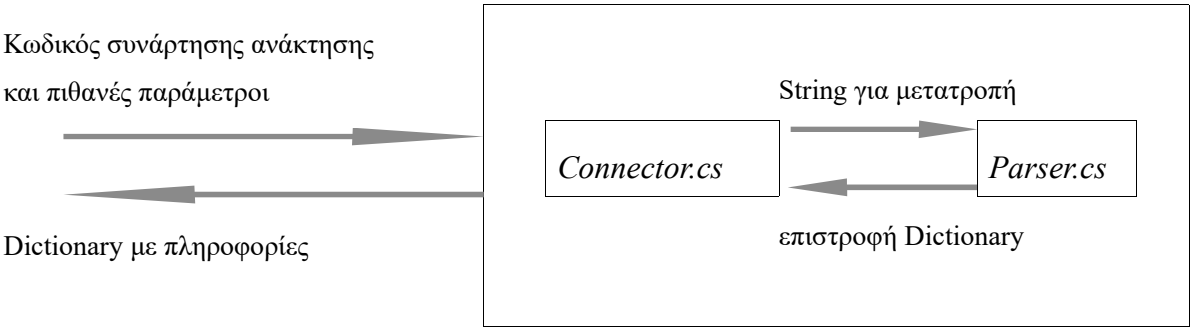
Ο parser λειτουργεί σπάζοντας το string στους χαρακτήρες διαχωρισμού, προσθέτοντας τα αποτελέσματα του διαχωρισμού σε ArrayLists και μετρώντας τα στοιχεία των ArrayList που δημιουργούνται, ώστε με τους κατάλληλους υπολογισμούς να γνωρίζουμε πόσα κλειδιά και πόσες τιμές για κάθε κλειδί υπάρχουν. Τέλος με τη χρήση του βοηθητικού ArrayList, finalArrayList, ελέγχουμε ένα ένα τα ζευγάρια κλειδιών-τιμών και δημιουργούμε το Dictionary το οποίο επιστρέφουμε στη συνάρτηση ανάκτησης πληροφοριών.

```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;

public class Parser : MonoBehaviour {

    public static int numberOfelements;
    public static int numberOflists;
    ArrayList myAL = new ArrayList(); //we use arraylist cause we dont know the exact number of elements
    ArrayList myAL2 = new ArrayList();
    public static ArrayList finalArrayList = new ArrayList();
    public static Dictionary<string, List<string>> guiInformation;

    public IEnumerator parseString (string aString, System.Action<Dictionary<string, List<string>>> result) {
        myAL.Clear ();
        myAL.AddRange (aString.Split ('%'));
        numberOfelements = myAL.Count;
        myAL2.Clear ();

        for (int i = 0; i < myAL.Count; i++) {
            aString = myAL[i].ToString();
            myAL2.AddRange (aString.Split ('|'));
        }
        numberOflists = (myAL2.Count / numberOfelements);

        Dictionary<string,List<string>> guiInformation = new Dictionary<string,List<string>>();

        for (int i = 0; i < myAL2.Count; i++) {
            aString = myAL2[i].ToString();
            finalArrayList.AddRange (aString.Split ('$'));
            if(!guiInformation.ContainsKey(finalArrayList [0].ToString())) {
                guiInformation.Add(finalArrayList [0].ToString(), new List<string>());
                guiInformation[finalArrayList [0].ToString()].Add(finalArrayList [1].ToString());
            }else{
                guiInformation[finalArrayList [0].ToString()].Add(finalArrayList [1].ToString());
            }
            finalArrayList.Clear();
        }

        yield return null;
        result (guiInformation);
    }
}
```

Εικόνα 5.18: Parser αποκωδικοποίησης πληροφορίας

Έτσι τα scripts Connector.cs και Parser.cs σχηματίζουν ένα μηχανισμό ανάκτησης πληροφοριών από το server (Εικόνα 5.19). Έχουμε λοιπόν τη δυνατότητα να καλέσουμε μια από τις συναρτήσεις του Connector από οποιοδήποτε σημείο του κώδικα μας δίνοντας σαν ορίσματα το αναγνωριστικό μιας συνάρτησης του module που φτιάξαμε και τις παραμέτρους αν αυτές χρειάζονται, και να πάρουμε ένα Dictionary προκαθορισμένης μορφής.

Με αυτό το τρόπο η προσθήκη νέων συναρτήσεων ανάκτησης πληροφοριών από το server είναι πλέον εύκολη, καθώς το μόνο που χρειάζεται είναι να προσθέσουμε μια συνάρτηση στο php module και γνωρίζοντας μόνο το αναγνωριστικό της και το είδος παραμέτρων που χρειάζεται, όπως για παράδειγμα id κτιρίου, να πάρουμε τις πληροφορίες που χρειαζόμαστε σε εύχρηστη μορφή.



Εικόνα 5.19: Λειτουργία συστήματος λήψης πληροφοριών

ΥΛΟΠΟΙΗΣΗ ΧΕΙΡΙΣΜΟΥ & ΓΡΑΦΙΚΗΣ ΔΙΕΠΑΦΗΣ

ΚΕΦΑΛΑΙΟ 6

6.1 Εντοπισμός κτιρίου με χρήση αφής

Είδαμε πως μπορούμε να χρησιμοποιήσουμε Raycasting για να εντοπίσουμε αντικείμενα στη σκηνή μας. Χρησιμοποιώντας λοιπόν ένα Raycast και στοχεύοντας για αντικείμενα που έχουν label Buildings μπορούμε να βρούμε το όνομα του αντικειμένου ενός κτιρίου το οποίο επέλεξε ο χρήστης, και να το χρησιμοποιήσουμε σαν παράμετρο για να αντλήσουμε πληροφορίες από το server για αυτό το κτίριο με το τρόπο που περιγράψαμε στην ενότητα 5.2.2

Το Raycast αυτό θα εκτελείται κάθε φορά που ο χρήστης αγγίζει την οθόνη. Στο detectTouch.cs το οποίο περιγράψαμε στην ενότητα 4.2.1 κάθε φορά που γίνεται ένα νέο άγγιγμα στην οθόνη θα εκπέμπονται δυο Rays. Το ένα θα στοχεύει μόνο στο έδαφος ώστε να έχουμε τις πληροφορίες για το σημείο εδάφους στο οποίο άγγιξε ο χρήστης για τη περίπτωση που θέλει να μετακινήσει τη κάμερα και το άλλο θα στοχεύει όλα τα αντικείμενα ανεξαρτήτου label. Έτσι αν ο χρήστης δε θέλει να μετακινήσει τη κάμερα και αγγίξει στιγμιαία την οθόνη σε ένα κτίριο ή ένα γλόμπο, το ray θα εντοπίσει το πρώτο αντικείμενο που θα συναντήσει ώστε να ανοίξουμε το αντίστοιχο μενού.

Στη περίπτωση λοιπόν που το Status είναι Idle και σταματήσει το άγγιγμα, σημαίνει ότι ο χρήστης δε θέλει να μετακινήσει τη κάμερα και απλώς άγγιξε ένα αντικείμενο, οπότε ανάλογα με το tag του αντικειμένου που πέτυχε το Ray ανοίγουμε το αντίστοιχο μενού (Εικόνα 6.1).

```
if (State == "idle"){
    if (Input.GetTouch (0).phase == TouchPhase.Began){
        touching = true;
        touchposition = Input.GetTouch(0).position;
        firstTerrainPoint = rayComp.Cast(9,State).point; //cast on terrain and return point of hit
        tapHit = rayComp.CastALL(); //cast on all objects
        hitName = tapHit.transform.name;
        hitTag = tapHit.transform.tag;
    }else if (touchCounter >= 0.1 && (Input.GetTouch(0).phase == TouchPhase.Stationary || (Input.GetTouch(0).phase == TouchPhase.Moved && Input.GetTouch(0).deltaPosition.magnitude < 10) ) ){
        secondTerrainPoint = Camera.main.GetComponent<castRay>().Cast(9,State).point; //cast on terrain and return point of hit
        State = "grab";
    }else if (Input.GetTouch(0).phase == TouchPhase.Moved && Input.GetTouch(0).deltaPosition.magnitude >= 10){
        secondTerrainPoint = Camera.main.GetComponent<castRay>().Cast(9,State).point; //cast on terrain and return point of hit
        State = "moving";
    }else if (Input.GetTouch(0).phase == TouchPhase.Ended && (!topBar.Contains(touchposition)) && (!onoff.Contains(touchposition)) && (!postidea.Contains(touchposition)) && (!gpsbutton.Contains(touchposition))){
        touching = false;
        if (hitName !="" ){
            if (hitTag == "UserIdeas"){
                GuiObj.GetComponent<theGui>().changeGuiWindow();
                GuiObj.GetComponent<theGui>().ideaMenu(hitName); //this holds the name of the idea
            }else if (hitTag == "Buildings"){
                GuiObj.GetComponent<theGui>().changeGuiWindow();
                GuiObj.GetComponent<theGui>().buildingMenu(hitName); //this holds the name of the building
            }
        }

    }else if (Input.GetTouch(0).phase == TouchPhase.Ended){
        touching = false;
    }
}
```

Εικόνα 6.1: Εντοπισμός κτιρίου με χρήση αφής

Επίσης χρησιμοποιώντας τη μεταβλητή touchposition, όταν ξεκινάει ένα άγγιγμα κρατάμε τις συντεταγμένες x, y σε pixels στην οθόνη, όπου έγινε το άγγιγμα. Αυτές οι συντεταγμένες μας βοηθούν ώστε να ελέγξουμε αν το άγγιγμα έγινε σε πάνω σε κάποιο στοιχείο του GUI του Unity, το οποίο η ακτίνα Ray αγνοεί. Έτσι αν ο χρήστης αγγίξει για παράδειγμα ένα κουμπί στη πάνω μπάρα του μενού, ακόμα και αν το Ray πετύχει ένα κτίριο το οποίο βρίσκεται πίσω από το κουμπί, μπορούμε

γνωρίζοντας το σημείο αγγίγματος στην οθόνη να αγνοήσουμε τα αποτελέσματα του Ray ώστε να ανοίξει το μενού του κουμπιού. Αυτό γίνεται με τη χρήση της δομής Rect που μας παρέχει το Unity με τις οποίες δηλώνουμε δισδιάστατα τετράγωνα στην οθόνη.

```
Rect topBar = new Rect(0, 9*Screen.height/10, Screen.width, Screen.height/10);
Rect onoff = new Rect(35*Screen.width/40, 0, 4*Screen.width/40, 4*Screen.height/40);
Rect postidea = new Rect(35*Screen.width/40, 4*Screen.height/40, 4*Screen.width/40, 7*Screen.height/40);
Rect gpsbutton = new Rect(19*Screen.width/20, 7*Screen.height/10, Screen.width/20, Screen.height/10);
```

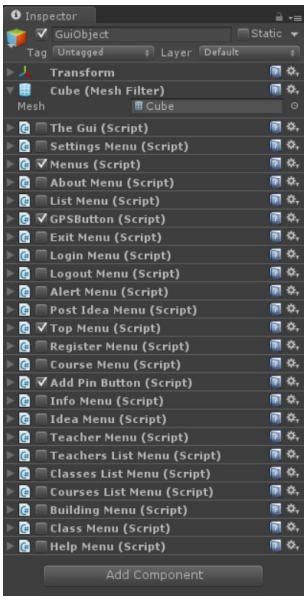
Αξίζει εδώ να σημειώσουμε ότι οι συντεταγμένες ενός Rect διαφέρουν ανάλογα με το αν αντιπροσωπεύει ένα στοιχείο του GUI ή όχι. Στη περίπτωση μας το σημείο (0,0) βρίσκεται στο κάτω αριστερό μέρος της οθόνης, ενώ αν δηλώνουμε ένα Rect για να χρησιμοποιηθεί για το σχεδιασμό στοιχείων του GUI τότε το (0,0) βρίσκεται στο πάνω αριστερό μέρος της οθόνης. Αυτή είναι μια ιδιοτροπία του Unity που μπορεί να μας δυσκολέψει, ιδιαίτερα αν δηλώνουμε τις συντεταγμένες ως συνάρτηση των διαστάσεων της οθόνης ώστε να μπορούν να λειτουργήσουν σωστά ανεξάρτητα από το μέγεθος της.

6.2 Προγραμματισμός γραφικής διεπαφής

Έχοντας πλέον τη δυνατότητα εντοπισμού αντικειμένων στη σκηνή, αλλά και της λήψης πληροφοριών από το server μπορούμε να δημιουργήσουμε τα μενού της εφαρμογής μας. Το Unity μας προσφέρει μεγάλη ποικιλία κλάσεων οι οποίες βοηθούν στη δημιουργία πολύπλοκων λειτουργικών μενού.

6.2.1 Λογική GUI

Για την υλοποίηση των script τα οποία θα εμφανίζουν τα μενού θα χρειαστεί να δημιουργήσουμε ένα gameobject στη σκηνή μας το οποίο θα έχει σαν components όλα τα scripts τα οποία υλοποιούν τα μενού της γραφικής διεπαφής μας (Εικόνα 6.2). Κάθε script το οποίο είναι ενεργό στη σκηνή μας μπορεί να προβάλει γραφικά στοιχεία GUI, αρκεί να χρησιμοποιήσουμε τη συνάρτηση OnGUI() του Unity. Μέσα στην OnGUI() ενός script μπορούμε να χρησιμοποιήσουμε ξεχωριστές κλάσεις του Unity, όπως η GUI, GUILayout, GUIContent και άλλες, οι οποίες χρησιμεύουν για τη προβολή στοιχείων γραφικής διεπαφής όπως κουμπιά και λεζάντες. Εμείς όμως για κάθε παράθυρο μενού θα χρησιμοποιούμε ένα ξεχωριστό script ώστε να μπορούμε να ελέγχουμε ευκολότερα ποιο μενού θα ενεργοποιείται κάθε φορά.



Εικόνα 6.2: GuiObject components

Εκτός από τα script για κάθε μενού, θα προσθέσουμε σαν component το TheGUI.cs το οποίο θα υλοποιεί τη λογική των μενού μας. Αρχικά μπορούμε να θεωρήσουμε ότι αν το script είναι ενεργοποιημένο, τότε έχουμε ανοικτό κάποιο παράθυρο GUI και έτσι προσθέτοντας έναν έλεγχο στο DetectTouch.cs να απενεργοποιούμε τον χειρισμό της κάμερας ή τη χρήση Raycasting για τον εντοπισμό αντικειμένων στη σκηνή όταν έχουμε ένα ανοικτό παράθυρο.

Το TheGUI.cs θα είναι απενεργοποιημένο από προεπιλογή και η αλλαγή της κατάστασης του θα γίνεται με τη χρήση συναρτήσεων όπως η changeGuiWindow() στην οποία απενεργοποιούμε όλα τα script components του GuiObject εκτός από τη πάνω μπάρα και το TheGUI (Εικόνα 6.3). Με παρόμοιο τρόπο υλοποιούμε συναρτήσεις που θα καλούνται για παράδειγμα όταν κλείνουμε ένα παράθυρο της γραφικής διεπαφής ή όταν θέλουμε να απενεργοποιήσουμε και να ενεργοποιήσουμε μαζικά κάποια script components.

```
// This is called when a button is pressed in a window before the action take place. Tap on objs also counts.
public void  changeGuiWindow(){
    MonoBehaviour[] scripts = GuiObj.GetComponents<MonoBehaviour>();
    foreach(MonoBehaviour script in scripts)
    {
        script.enabled = false;
    }
    GuiObj.GetComponent<theGui>().enabled = true;
    GuiObj.GetComponent<TopMenu>().enabled = true;
}
```

Εικόνα 6.3: Συνάρτηση changeGuiWindow

Ακόμα, προσθέτουμε συναρτήσεις τις οποίες θα καλούμε ώστε να αλλάξουμε τη τιμή σε μια μεταβλητή η οποία υποδηλώνει το μενού το οποίο θέλουμε να ανοίξει το TheGUI αλλά και να για να μεταφέρουμε στο script του μενού που θα ανοίξει, μια τιμή όπως το όνομα του κτιρίου που επιλέξαμε ή το id ενός καθηγητή. Η τιμή αυτή αποθηκεύεται σε μια static μεταβλητή στην οποία έχει πρόσβαση οποιοδήποτε script.

```
public void  courseMenu(string id){
    idToOpen = id;
    activeMenu = "courseMenu";
}
```

Τέλος στην Update() του TheGUI υλοποιούμε ένα βρόγχο switch-case όπου ανάλογα με τη τιμή που έχει η μεταβλητή activeMenu ενεργοποιείται το αντίστοιχο script για να ανοίξει το μενού (Εικόνα 6.4).

```
void Update () {

    switch (activeMenu){
    case "confirmExitMenu":
        GuiObj.GetComponent<ExitMenu>().enabled = true; //Enable the ExitMenu script
        break;
    case "buildingMenu":
        GuiObj.GetComponent<BuildingMenu>().enabled = true;
        break;
    case "classMenu":
        GuiObj.GetComponent<ClassMenu>().enabled = true;
        break;
    case "classes":
        GuiObj.GetComponent<ClassesListMenu>().enabled = true;
        break;
    case "courses":
        GuiObj.GetComponent<CoursesListMenu>().enabled = true;
        break;
    case "teachers":
        GuiObj.GetComponent<TeachersListMenu>().enabled = true;
        break;
    case "searchme":
```

Εικόνα 6.4: Μηχανισμός ενεργοποίησης μενού

Καταφέραμε να δημιουργήσουμε ένα σύστημα για την εύκολη διαχείριση των μενού μας. Τώρα για να ανοίξουμε ένα μενού αρκεί να καλέσουμε την changeGuiWindow() για να ενεργοποιηθεί το TheGUi.cs και έπειτα να καλέσουμε τη συνάρτηση για την ενεργοποίηση ενός παραθύρου. Για παράδειγμα τη buildingMenu(hitName) στη περίπτωση αγγίγματος ενός κτιρίου.

6.2.2 Σχεδιασμός GUI

Μόλις ενεργοποιηθεί ένα script μενού, χρησιμοποιούμε τη συνάρτηση OnEnable() για να αρχικοποιήσουμε τις μεταβλητές που θα μας χρειαστούν για την εμφάνιση του μενού και ξεκινάμε τη Coroutine fetchGuiData() η οποία ξεκινάει μια σειρά από άλλες Coroutines σκοπός των οποίων είναι η ανάκτηση πληροφοριών από το server με τη χρήση του Connector, η άντληση πληροφοριών από το πρόγραμμα μαθημάτων με το ScheduleCashe αν το μενού που μόλις άνοιξε τις χρειάζεται, και η μετατροπή των παραπάνω Dictionaries σε ArrayLists με τη κλήση της συνάρτησης convertToArrLst() (Εικόνα 6.5).

```
public IEnumerator convertToArrLst(Dictionary<string, List<string>> dictionaryForConv, Dictionary<string, List<string>> schedForConv){

    foreach (KeyValuePair<string, List<string>> entry in dictionaryForConv) {
        elementsNo=0;
        for (int i = 0; i < entry.Value.Count; i++) {
            elementsNo = elementsNo + 1;
        }
    }

    listsNo = dictionaryForConv.Count;
    informationToUse = new string[listsNo, elementsNo];
    listOfKeys = new string[listsNo];
    labelscounter = 0;

    foreach (KeyValuePair<string, List<string>> entry in dictionaryForConv) {
        listOfKeys[labelscounter] = entry.Key.ToString ();
        for (int i = 0; i < entry.Value.Count; i++) {
            informationToUse[labelscounter,i] = entry.Value [i];
        }
        labelscounter = labelscounter + 1;
    }

    foreach (KeyValuePair<string, List<string>> entry in schedForConv) {
        schedNo=entry.Value.Count;
    }

    schedInfo = new string[4, schedNo];

    foreach (KeyValuePair<string, List<string>> entry in schedForConv){ //create the schedule lists
        for (int i = 0; i < entry.Value.Count; i++) {
            if ( entry.Key.ToString() == "courseid"){
                schedInfo[0,i] = entry.Value[i];
            }else if ( entry.Key.ToString() == "coursename"){
                schedInfo[1,i] = entry.Value[i];
            }else if ( entry.Key.ToString() == "classid"){
                schedInfo[2,i] = entry.Value[i];
            }else if ( entry.Key.ToString() == "classname"){
                schedInfo[3,i] = entry.Value[i];
            }
            Debug.Log(entry.Value[i]);
        }
    }
    arrReady = true;
    yield return null;
}
```

Εικόνα 6.5: Συνάρτηση convertToArrLst

Η δήλωση της fetchGuiData() σαν Ienumerator είναι απαραίτητη για να μπορέσουμε μέσα σε αυτή να ξεκινήσουμε τις υπόλοιπες Coroutines χρησιμοποιώντας yield return ώστε να μην εκτελούνται παράλληλα και έχουμε απώλειες δεδομένων. Απαραίτητη είναι και η μετατροπή των Dictionaries σε ArrayList για να μπορούν να χρησιμοποιηθούν οι πληροφορίες στην OnGUI(), καθώς δε μπορούν μέσα σε αυτή να χρησιμοποιηθούν πληροφορίες από Dictionaries τα οποία δεν έχουν γίνει serialized. Θα μπορούσαμε να μη χρησιμοποιήσουμε Dictionaries αλλά ArrayLists από την αρχή, αλλά θέλουμε να κρατάμε τις πληροφορίες σε Dictionaries καθώς η προσπέλαση και επεξεργασία τους είναι πιο εύκολη και αυτή η μορφή μας βοηθά σε άλλες πράξεις όπως για παράδειγμα στη δημιουργία Dictionaries με μόνο τις απαραίτητες πληροφορίες για τη τρέχουσα ημέρα και ώρα, από το Dictionary που έχουμε το πρόγραμμα εξαμήνου.

Μόλις τελειώσει η εκτέλεση της convertToArrLst() και έχουμε τις πληροφορίες σε μορφή ArrayList είμαστε έτοιμοι να τις εμφανίσουμε στο μενού μας. Αυτό το πετυχαίνουμε με τη χρήση μιας boolean μεταβλητής η οποία δηλώνει αν έχει τελειώσει η μετατροπή. Με έναν έλεγχο της μεταβλητής μπορούμε να εμφανίζουμε ένα μήνυμα αναμονής στο χρήστη για όσο διαρκεί η σύνδεση με το server και η μετατροπή των δεδομένων και όταν αλλάξει η τιμή της να εμφανίσουμε τις πληροφορίες στο μενού. Αυτός ο έλεγχος γίνεται στη συνάρτηση OnGUI() του script.

Η δημιουργία ενός μενού είναι σχετικά απλή με τη χρήση των κλάσεων του Unity (Εικόνα 6.6). Μπορούμε να δημιουργήσουμε τετράγωνα, κουμπιά, λεζάντες και μπάρες κύλισης μενού δηλώνοντας απλώς τις διαστάσεις τους και το Unity θα τα εμφανίσει στη θέση που δηλώσαμε χρησιμοποιώντας τα default textures του μενού. Η ανίχνευση της αφής ενός κουμπιού GUI γίνεται αυτόματα από το Unity αν ο χρήστης ακουμπήσει σε εκείνο σημείο στην οθόνη και σε αντίθεση με τα κτίρια και τους γλόμπους όπου έπρεπε να χρησιμοποιήσουμε Raycasting για να λειτουργήσει ένα κουμπί αρκεί να το δημιουργήσουμε σε ένα βρόγχο if μέσα στον οποίο θα εκτελούμε τις ενέργειες του κουμπιού, όπως για παράδειγμα το άνοιγμα ενός νέου μενού.

Όλες τις διαστάσεις των στοιχείων των μενού τις δηλώνουμε σε αναλογία με τις διαστάσεις της οθόνης της συσκευής στην οποία τρέχει η εφαρμογή. Αποφεύγουμε να δηλώνουμε τις διαστάσεις με ακριβή αριθμό pixels γιατί δε θα εμφανίζονται σωστά σε μικρές και μεγάλες οθόνες.

Για να σχεδιάσουμε κυλιόμενα μενού τα οποία δεν έχουν συγκεκριμένο αριθμό στοιχείων, όπως για παράδειγμα το μενού των κτιρίων, όπου ο αριθμός των στοιχείων που θέλουμε να εμφανίζουμε διαφέρει από κτίριο σε κτίριο, κάνουμε χρήση της κλάσης GUILayout. Η κλάση αυτή μας δίνει τη δυνατότητα δημιουργίας οριζόντιων και κάθετων τμημάτων στα οποία μπορούμε να στοιχίσουμε κουμπιά, λεζάντες και άλλα. Η στοίχιση γίνεται αυτόματα και ομοιόμορφα ανάλογα το μέγεθος του μεγαλύτερου στοιχείου, άρα δε χρειάζεται να μας απασχολεί το μέγεθος τους. Επειδή όμως θέλουμε να γνωρίζουμε το συνολικό ύψος των κουμπιών που θα εμφανίσουμε στο κυλιόμενο μενού, ώστε να ρυθμίσουμε τη μπάρα κύλισης του Scroolview, θα δώσουμε συγκεκριμένο ύψος στα κουμπιά μας για να μπορούμε να το υπολογίσουμε.

```
void OnGUI () {
    GUI.skin = WhiteSkin;
    GUI.BeginGroup (listMenu);
    GUI.Box (new Rect (0, 0, listMenu.width, listMenu.height), GUIContent.none);

    if (GUI.Button (new Rect (9*listMenu.width/10, 0, listMenu.width/10, listMenu.height/10), "X")){ //make the X button
        this.CloseMenu();
    }

    if (arrReady == true) {

        scrollViewVector = GUI.BeginScrollView (new Rect (0, 2*listMenu.height/10, listMenu.width, 8*listMenu.height/10), scrollViewVector, new Rect (0, 0, 19*listMenu.width/20, elementsNo*listMenu.height/10));
        // Put something inside the ScrollView
        GUILayout.BeginArea (new Rect (0, 0, listMenu.width, elementsNo *listMenu.height/10));
        GUILayout.BeginHorizontal();
        GUILayout.BeginVertical();

            for (int j = 0; j < elementsNo; j++) { //number of elements of the column with the most elements
                if ( j%2 != 0){
                    CurrentButtonStyle.normal.background = WhiteButton;
                }else{
                    CurrentButtonStyle.normal.background = GrayButton;
                }

                if (GUILayout.Button (informationToUse[1,j], CurrentButtonStyle, GUILayout.Height(listMenu.height/10), GUILayout.ExpandHeight(false))){
                    GuiObj.GetComponent<theGui>().changeGuiWindow();
                    GuiObj.GetComponent<theGui>().classMenu(informationToUse[0,j]);
                }
            }

        GUILayout.EndVertical();
        GUILayout.BeginVertical();

            for (int j = 0; j < elementsNo; j++) { //number of elements of the column with the most elements
                if ( j%2 != 0){
                    CurrentButtonStyle.normal.background = WhiteButton;
                }else{
                    CurrentButtonStyle.normal.background = GrayButton;
                }

                for (int i = 0; i < schedNo; i++) {
                    if ( informationToUse[0,j] == schedInfo[2,i] ){
                        if (GUILayout.Button (schedInfo[1,i], CurrentButtonStyle, GUILayout.Height(listMenu.height/10), GUILayout.ExpandHeight(false))){
                            GuiObj.GetComponent<theGui>().changeGuiWindow();
                            GuiObj.GetComponent<theGui>().courseMenu(schedInfo[0,i]);
                        }
                    }
                }
            }
        GUILayout.EndVertical();
        GUILayout.EndHorizontal();
        GUILayout.EndArea ();
        // End the ScrollView
        GUI.EndScrollView();

    }else{
        GUILayout.BeginArea (new Rect (0, listMenu.height/10, listMenu.width, listMenu.height/10));
        GUILayout.Label ("Loading Data");
        GUILayout.EndArea ();
    }
    GUI.EndGroup ();
}
```

Εικόνα 6.6: Υλοποίηση μενού κτιρίου

Για την αποκωδικοποίηση του αρχείου JSON του προγράμματος δημιουργήσαμε το ScheduleCashe.cs το οποίο είναι ενεργοποιημένο από προεπιλογή και δημιουργεί ένα Dictionary για το πρόγραμμα κατά την εκκίνηση της εφαρμογής. Η αποκωδικοποίηση γίνεται με τη βοήθεια της κλάσης JSONObject[11] η οποία παρέχει χρήσιμα εργαλεία για δημιουργία και αποκωδικοποίηση, είναι γραμμένη σε C# script το οποίο μπορούμε να χρησιμοποιήσουμε στο Unity και μπορούμε να τη βρούμε δωρεάν στο internet.

Έχοντας δημιουργήσει GUI skins για τα διάφορα μενού μας, τα οποία καθορίζουν την εμφάνιση των στοιχείων, όπως για παράδειγμα το χρώμα ενός κουμπιού ή την εικόνα φόντου μιας λεζάντας, μπορούμε να εφαρμόσουμε αυτά τα skins στο μενού μας απλώς δηλώνοντας το σαν μεταβλητή και χρησιμοποιώντας το στο μενού μας. Αν θέλουμε να χρησιμοποιήσουμε μια άλλη εικόνα για ένα κουμπί μπορούμε να το κάνουμε με τη χρήση κώδικα. Το Unity μας προσφέρει δυνατότητες ρύθμισης των στοιχείων του GUI είτε μαζικά είτε ξεχωριστά κάθε στοιχείο.

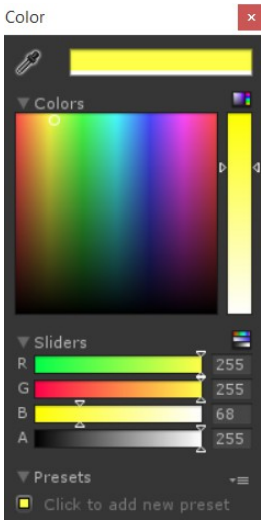
Εκτός από την εμφάνιση πληροφοριών και το άνοιγμα παραθύρων μπορούμε να χρησιμοποιήσουμε κουμπιά για να εκτελέσουμε οποιαδήποτε ενέργεια. Ένα καλό παράδειγμα τέτοιας περίπτωσης είναι το κουμπί On/Off (Εικόνα 6.7).

```
if (GUI.Button (OnOffBtn, myBool ? OnbtnTexture:OffbtnTexture, IdeaButtonStyle)) {
    myBool = !myBool;

    if (myBool == true) {
        StartCoroutine(this.loadAndPlaceIdeas());
        GUI.color = Color.yellow;
    }else {
        StartCoroutine(this.removeIdeas());
        GUI.color = Color.gray;
    }
}
```

Εικόνα 6.7: Υλοποίηση κουμπιού On/Off

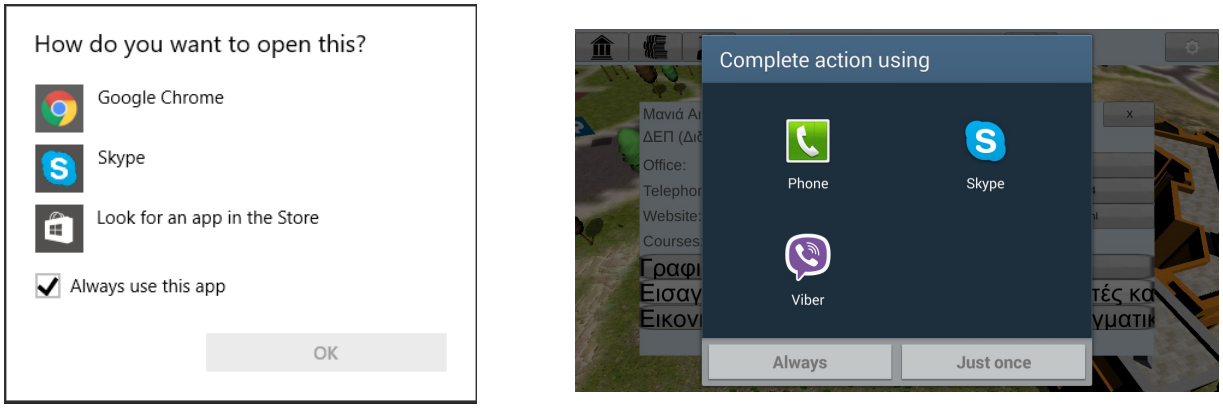
Επιλέγοντας αυτό το κουμπί ο χρήστης μπορεί να εμφανίζει και να εξαφανίσει τους γλόμπους στο χάρτη. Αυτή η δυνατότητα προστέθηκε ώστε ο χρήστης να μπορεί να απενεργοποιεί τους γλόμπους και να έχει καλύτερη ορατότητα στο χάρτη. Όταν ο χρήστης επιλέγει το κουμπί ώστε να εμφανιστούν οι ιδέες ξεκινάει μια Coroutine η οποία δημιουργεί αντίγραφα του γλόμπου μας και αλλάζει το χρώμα τους ανάλογα τη βαθμολογία ψήφων (Εικόνα 6.8). Για να υλοποιήσουμε αυτή τη δυνατότητα δημιουργούμε για κάθε αντίγραφο ένα νέο material από αυτό που ήδη έχει ο γλόμπος μας και με τις κατάλληλες πράξεις στις τιμές R,G,B,A του χρώματος του material αλλάζουμε το χρώμα συναρτήσει της βαθμολογίας. Μπορούμε να δούμε τις ακολουθίες χρωμάτων από τη παλέτα χρωμάτων αν επιλέξουμε το χρώμα του material στον inspector, για να μας βοηθήσει στις πράξεις. Βλέπουμε για παράδειγμα ότι αν μειωθεί το G το material κοκκινίζει.

```
public IEnumerator placeIdeas(Dictionary<string, List<string>> Dict){

    Renderer rend = idea.transform.Find("Top").renderer;
    NumberOfTotalIdeas = Dict.ElementAtOrDefault(0).Value.Count;
    float colorR;
    float colorG;
    float colorB;
    float colorA;

    for (int i = 0; i < NumberOfTotalIdeas; i++) { // run through every idea and fetch the needed values to place it
        tempIdeaClone = (Transform) Instantiate(idea, new Vector3(float.Parse(Dict.ElementAtOrDefault(1).Value[i]), 25, float.Parse(Dict.ElementAtOrDefault(2).Value[i])), Quaternion.identity);
        tempIdeaClone.name = Dict.ElementAtOrDefault(0).Value[i];
        tempIdeaClone.tag = "UserIdeas";
        tempIdeaClone.gameObject.layer = 10;
        float ideaKarma = float.Parse(Dict.ElementAtOrDefault(3).Value[i]) * 0.01f;
        Renderer tempRend = tempIdeaClone.transform.Find("Top").renderer;
        tempRend.material = new Material(shader); // create a temp material for each bulb

        if ( ideaKarma >= 0.0f && ideaKarma <= 1.0f){
            colorR = 1.0f;
            colorG = 1.0f - ideaKarma;
            colorB = 0.0f;
            colorA = 1.0f;
        }else if ( ideaKarma < 0.0f && ideaKarma >= -1.0f){
            colorR = 1.0f;
            colorG = 1.0f;
            colorB = 0.0f - ideaKarma;
            colorA = 1.0f;
        }else if ( ideaKarma < -1.0f ){
            colorR = -1.0f/ideaKarma;
            colorG = -1.0f/ideaKarma;
            colorB = -1.0f/ideaKarma;
            colorA = 1.0f;
        }else{
            colorR = 1.0f;
            colorG = 0.0f;
            colorB = 0.0f;
            colorA = 1.0f;
        }
        tempRend.material.color = new Color (colorR, colorG, colorB, colorA);
        tempIdeaClone.transform.localScale += new Vector3 (16 +ideaKarma*10, 16 +ideaKarma*10, 16 +ideaKarma*10);
    }
    yield return null;
}
```



Εικόνα 6.8: Αλλαγή χρώματος γλόμπου

Με το Unity μπορούμε να εκτελέσουμε ενέργειες, όπως η αποστολή email, η κλήση ενός τηλεφώνου ή το άνοιγμα ενός εξωτερικού συνδέσμου με τη χρήση Uniform Resource Identifier (URI)[13] και τη βοήθεια της κλάσης Application.OpenURL() η οποία ψάχνει αυτόματα για διαθέσιμες εφαρμογές στη συσκευή για να εκτελέσει τις ενέργειες και προσφέρει στο χρήστη τη δυνατότητα επιλογής. Έτσι μπορούμε γράφοντας Application.OpenURL("tel://" + informationToUse); να ορίσουμε ένα κουμπί το οποίο πραγματοποιεί τηλεφωνική κλήση και ανάλογα τη συσκευή την οποία χρησιμοποιούμε να εκτελεστεί η ενέργεια με βάση τις δυνατότητες της (Εικόνα 6.9).



Εικόνα 6.9: Ενέργεια συστήματος για άνοιγμα URI σε windows και android

Για να κάνουμε ελέγχους στο όνομα χρήστη και το email για μη επιτρεπτούς χαρακτήρες ή μη αποδεκτή μορφή χρησιμοποιούμε τη κλάση Regex του .Net framework η οποία μας παρέχει χρήσιμα εργαλεία για την υλοποίηση ελέγχων κανονικών εκφράσεων.

Μπορούμε για παράδειγμα δηλώνοντας μια κανονική έκφραση γράφοντας:

```
string validCharsRegex = @"^[A-Z0-9_-]+$";
```

να κάνουμε ελέγχους στην OnGUI() με τη βοήθεια των μεθόδων της κλάσης Regex και να εμφανίζουμε στο χρήστη τα κατάλληλα μηνύματα ή να ενεργοποιήσουμε τα αντίστοιχα κουμπιά (Εικόνα 6.10).

```
bool validUsername = Regex.IsMatch(username, validCharsRegex, RegexOptions.IgnoreCase);
int usernameLength = Regex.Match(username, validCharsRegex, RegexOptions.IgnoreCase).Length;

bool validPassword = Regex.IsMatch(password, validCharsRegex, RegexOptions.IgnoreCase);
int passwordLength = Regex.Match(password, validCharsRegex, RegexOptions.IgnoreCase).Length;

if (validUsername == false){
    GuiObj.GetComponent<theGui>().infoMenu("Username must consist of \n chars A-Z and 0-9");
}else if (usernameLength < minLength){
    GuiObj.GetComponent<theGui>().infoMenu("Username must be \n at least " +minLength+ " chars");
}else if (validPassword == false){
    GuiObj.GetComponent<theGui>().infoMenu("Password must consist of \n chars A-Z and 0-9]");
}else if (passwordLength < minLength){
    GuiObj.GetComponent<theGui>().infoMenu("Password must be \n at least " +minLength+ " chars");
}
```

Εικόνα 6.10: Έλεγχοι με κανονικές εκφράσεις

Βλέπουμε ότι το Unity σε συνδυασμό με τις δυνατότητες που προσφέρει το .Net framework μας διευκολύνουν ακόμα και στο σχεδιασμό του γραφικού περιβάλλοντος, με κλάσεις οι οποίες υλοποιούν πολύπλοκα μενού αλλά και ελέγχους σε κάθε frame χωρίς να επιβαρύνουν το σύστημα κατά την εκτέλεση της εφαρμογής.

6.3 Εμφάνιση θέσης στο χώρο με χρήση και μετατροπή συντεταγμένων GPS

Μπορεί να μη το αντιλαμβανόμαστε στη καθημερινότητα μας αλλά υπάρχουν πολλοί τρόποι ώστε να εντοπίσουμε ένα σημείο σε έναν πλανήτη και πολλά διαφορετικά συστήματα συντεταγμένων. Στη καθημερινότητα μας στους χάρτες που βλέπουμε στο διαδίκτυο ή στις συσκευές μας συναντάμε συντεταγμένες σε μορφή WGS84[14] οι οποίες αναπαριστούν συντεταγμένες πάνω σε ένα γεωειδές που βασίζεται στο σχήμα της γης και είναι σχεδόν όμοιο με το ελλειψοειδές GRS 80 (Εικόνα 6.11).

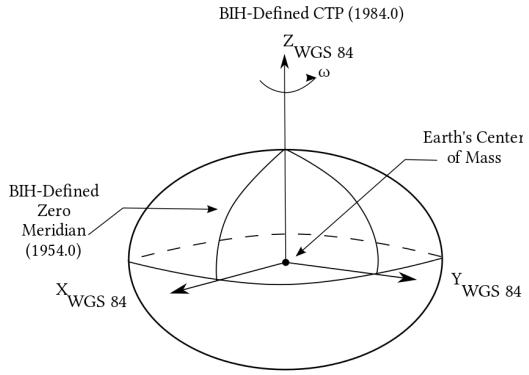


Figure 1.1 WGS 84 Reference Frame

Εικόνα 6.11: Γεωειδές συστήματος WGS84[14]

Για την αναπαράσταση των συντεταγμένων χρησιμοποιούμε δυο σημεία, το γεωγραφικό πλάτος και γεωγραφικό μήκος.

Το γεωγραφικό πλάτος μπορεί να αναπαρασταθεί με τις εξής μορφές:

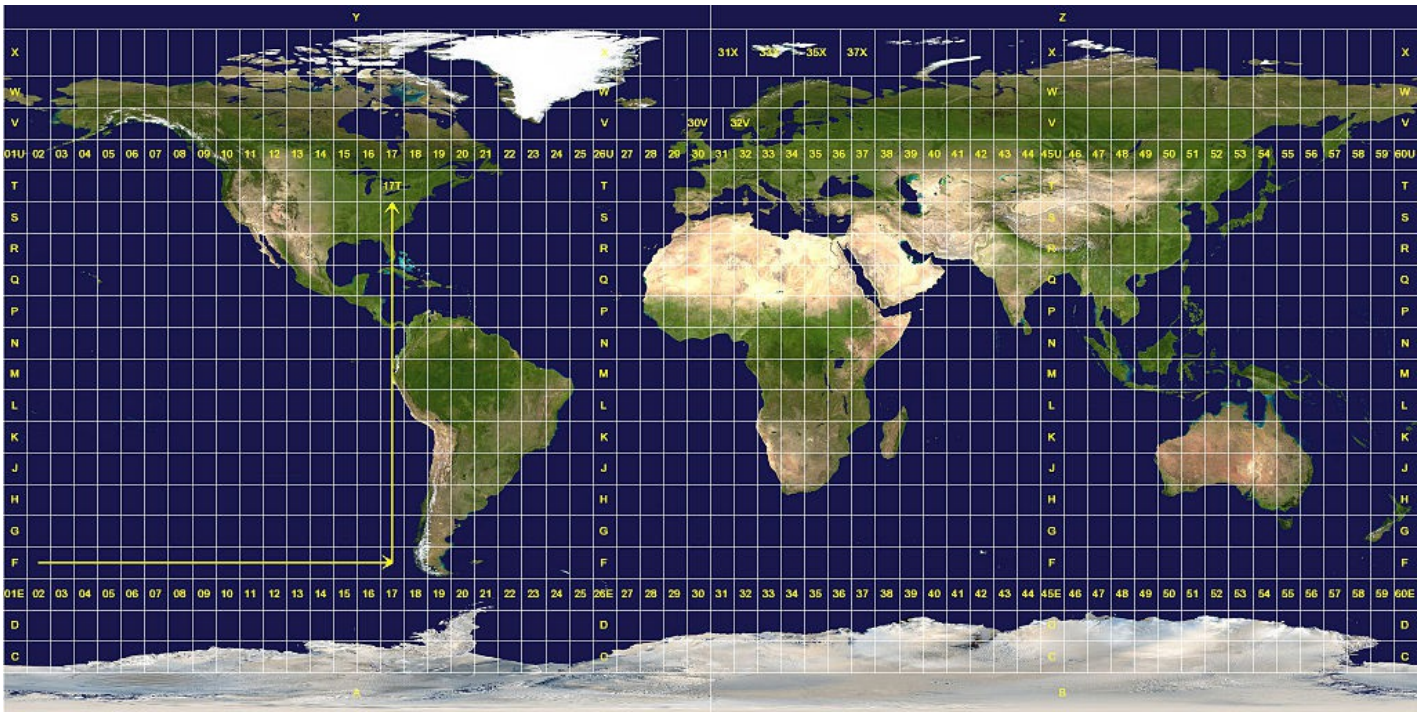
N43°38'19.39"
43°38'19.39"N
43 38 19.39
43.63871944444445

Αντίστοιχα μπορούμε να αναπαραστήσουμε το γεωγραφικό μήκος ως εξής:

W116°14'28.86"
116°14'28.86"W
-116 14 28.86
-116.2413513485235

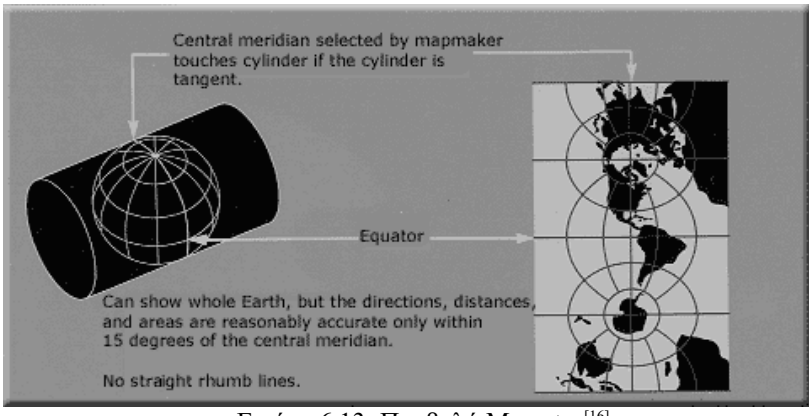
Αν εκφραστούν σε δεκαδική μορφή, βόρεια γεωγραφικά πλάτη και ανατολικά γεωγραφικά μήκη είναι θετικός αριθμός, νότια γεωγραφικά πλάτη και δυτικά γεωγραφικά μήκη είναι αρνητικά.

Για να αναπαραστήσουμε τις συντεταγμένες σε ένα καρτεσιανό σύστημα συντεταγμένων, όπως είναι οι συντεταγμένες στο terrain μας, θα χρησιμοποιήσουμε το σύστημα συντεταγμένων UTM.[15] (Εικόνα 6.12) Το σύστημα UTM χωρίζει τη γη σε 60 ζώνες, κάθε μια εκ των οποίων έχει 6° γεωγραφικού μήκους σε πάχος. Η ζώνη 1 καλύπτει από το γεωγραφικό πλάτος 180° μέχρι το 174° W και η αρίθμηση αυξάνεται προς την ανατολή μέχρι το 60, το οποίο καλύπτει το γεωγραφικό μήκος από 174° μέχρι 180° Ε.



Εικόνα 6.12: Πλέγμα συστήματος UTM[15]

Κάθε μια από τις 60 ζώνες χρησιμοποιεί εγκάρσια προβολή Mercator η οποία μπορεί να απεικονίσει μια περιοχή εκτενούς γεωγραφικού πλάτους με χαμηλή παραμόρφωση (Εικόνα 6.13). Το σύστημα εγκάρσιας προβολής Mercator είναι μια παραλλαγή της κλασικής προβολής Mercator.[16] Η εγκάρσια προβολή χρησιμοποιείται σε εθνικούς και παγκόσμιους χάρτες παγκοσμίων και σε συνδυασμό με ένα σύστημα συντεταγμένων δίνει μεγάλη ακρίβεια.



Εικόνα 6.13: Προβολή Mercator[16]

Χρησιμοποιώντας στενές ζώνες των 6° γεωγραφικού μήκους (μέχρι 800 χλμ) σε πάχος, και μειώνοντας το συντελεστή μεγέθους του μεσημβρινού σε 0,9996 (μία μείωση του 1:2500), η παραμόρφωση είναι 1 προς 1.000 μέσα σε κάθε ζώνη. Το γεωγραφικό μήκος επίσης χωρίζεται σε ζώνες, αλλά με διαφορετικό τρόπο. Οι ζώνες σημειώνονται με γράμματα.

Ένα σημείο στη γη ορίζεται από έναν αριθμό ζώνης UTM και ένα ζευγάρι τετμημένης (easting) και τεταγμένης (northing) για αυτό το σημείο. Η αρχή των αξόνων μιας UTM ζώνης είναι η τομή του ισημερινού και του πρώτου μεσημβρινού της ζώνης, αλλά για να αποφύγουμε αρνητικούς αριθμούς ο κεντρικός μεσημβρινός ορίζεται στα 500.000 μέτρα Ανατολικά. Οι τετμημένες UTM μπορεί να απέχουν από 167.000 μέτρα κοντά στους πόλους μέχρι 833.000 μέτρα στον ισημερινό. Τα σημεία στο Βόρειο ημισφαίριο υπολογίζονται ξεκινώντας από το μηδέν στον ισημερινό, η μέγιστη τιμή τεταγμένης είναι περίπου 9.300.000 μέτρα σε πλάτος 84 μοίρες Βόρεια. Στο Νότιο ημισφαίριο η τεταγμένη φτάνει τα 1.100.000 μέτρα στις 80 μοίρες Νότια. Η τεταγμένη στον ισημερινό ορίζεται στα 10.000.000 μέτρα ώστε κανένα σημείο να μην έχει αρνητικές τιμές.

Έτσι για αναπαραστήσουμε ένα σημείο σε ένα χάρτη UTM χρειαζόμαστε εκτός από το easting και το northing, τον αριθμό και το γράμμα της ζώνης και αυτό γιατί για ένα ζευγάρι easting και northing μπορούμε να έχουμε πάνω από ένα σημεία στο χάρτη. Η αναπαράσταση ενός σημείου είναι της μορφής:

35S 34195,6 35737,1

Όσο περισσότερα δεκαδικά έχει το easting και το northing τόσο μεγαλύτερη η ακρίβεια.

Για να μετατρέψουμε συντεταγμένες από το σύστημα WGS 84 σε UTM χρησιμοποιούμε μια απλούστερη έκδοση των τύπων για πλάτυνση πλανητών με εγκάρσια προβολή Mercator η οποία προήλθε από τον Johann Heinrich Louis Krüger το 1912. Έχουν ακρίβεια χιλιοστού για αποστάσεις μέχρι 3.000 χλμ από το κεντρικό μεσημβρινό.

Το σύστημα WGS 84 περιγράφει τη γη σαν ένα πεπλατυσμένο σφαιροειδές με ακτίνα ισημερινού α=6378,137 χλμ και πλάτυνση f =0,0033528106643315515. Εξ' ορισμού ισχύει ότι στο Βόρειο ημισφαίριο No=0 χλμ και στο Νότιο No=10000 χλμ. Επίσης ko=0,9996 και Eo=500 χλμ

Για να υπολογίσουμε τις συντεταγμένες easting και northing (E,N) από το γεωγραφικό πλάτος και μήκος (φ,λ) χρησιμοποιούμε λοιπόν τις παρακάτω φόρμουλες:[15]

$$n = \frac{f}{2-f}, \quad A = \frac{a}{1+n}\left(1 + \frac{n^2}{4} + \frac{n^4}{64} + \cdots\right),$$

$$\alpha_1 = \frac{1}{2}n - \frac{2}{3}n^2 + \frac{5}{16}n^3, \;\; \alpha_2 = \frac{13}{48}n^2 - \frac{3}{5}n^3, \;\; \alpha_3 = \frac{61}{240}n^3,$$

$$\beta_1 = \frac{1}{2}n - \frac{2}{3}n^2 + \frac{37}{96}n^3, \;\; \beta_2 = \frac{1}{48}n^2 + \frac{1}{15}n^3, \;\; \beta_3 = \frac{17}{480}n^3,$$

$$\delta_1 = 2n - \frac{2}{3}n^2 - 2n^3, \;\; \delta_2 = \frac{7}{3}n^2 - \frac{8}{5}n^3, \;\; \delta_3 = \frac{56}{15}n^3.$$

$$t = \sinh\left(\tanh^{-1}\sin\varphi - \frac{2\sqrt{n}}{1+n}\tanh^{-1}\left(\frac{2\sqrt{n}}{1+n}\sin\varphi\right)\right),$$

$$\xi' = \tan^{-1}\left(\frac{t}{\cos(\lambda-\lambda_0)}\right), \;\; \eta' = \tanh^{-1}\left(\frac{\sin(\lambda-\lambda_0)}{\sqrt{1+t^2}}\right),$$

$$\sigma = 1 + \sum_{j=1}^{3} 2j\alpha_j \cos(2j\xi')\cosh(2j\eta'), \;\; \tau = \sum_{j=1}^{3} 2j\alpha_j \sin(2j\xi')\sinh(2j\eta').$$

$$k = \frac{k_0 A}{a}\sqrt{\left\{1 + \left(\frac{1-n}{1+n}\tan\varphi\right)^2\right\}\frac{\sigma^2+\tau^2}{t^2+\cos^2(\lambda-\lambda_0)}},$$

$$\gamma = \tan^{-1}\left(\frac{\tau\sqrt{1+t^2} + \sigma t\tan(\lambda-\lambda_0)}{\sigma\sqrt{1+t^2} - \tau t\tan(\lambda-\lambda_0)}\right).$$

$$E = E_0 + k_0 A\left(\eta' + \sum_{j=1}^{3}\alpha_j \cos(2j\xi')\sinh(2j\eta')\right),$$

$$N = N_0 + k_0 A\left(\xi' + \sum_{j=1}^{3}\alpha_j \sin(2j\xi')\cosh(2j\eta')\right),$$

Στο Unity μπορούμε να πάρουμε συντεταγμένες αν η συσκευή έχει δυνατότητες GPS με τη χρήση της κλάσης LocationService η οποία μπορεί να ξεκινήσει ή να σταματήσει τη λειτουργία του GPS και να επιστρέψει τις τελευταίες γνωστές πληροφορίες τοποθεσίας, μπορούμε να προσπελάσουμε τις πληροφορίες τοποθεσίας με τη βοήθεια της δομής LocationInfo η οποία περιέχει πληροφορίες γεωγραφικού πλάτους και μήκους, ύψους αλλά ακόμα και την ακρίβεια της μέτρησης και να δούμε σε τι κατάσταση βρίσκεται η αναζήτηση συντεταγμένων με το enumeration LocationServiceStatus. Το γεωγραφικό πλάτος και μήκος που παίρνουμε βασίζεται στο σύστημα συντεταγμένων WGS 84 και έχουν δεκαδική μορφή. Ιδανική για να τη χρησιμοποιήσουμε στις παραπάνω φόρμουλες και να μετατρέψουμε τις συντεταγμένες σε μορφή καρτεσιανών συντεταγμένων UTM, τις οποίες θα μπορέσουμε να αναπαραστήσουμε στο terrain μας.

Για τη μετατροπή των συντεταγμένων θα δημιουργήσουμε το GPSConnect.cs το οποίο θα περιέχει τη συνάρτηση IEnumerator tryGPS() η οποία θα παίρνει τις συντεταγμένες από το GPS, θα υπολογίζει το easting και το northing θα το αντιστοιχίζει με συντεταγμένες (x,y) στο terrain μας και θα μετακινεί το αντικείμενο που δείχνει τη θέση του χρήστη σε εκείνο το σημείο (Εικόνα 6.14). Η υλοποίηση του GPSConnect.cs βασίστηκε στο μετατροπέα συντεταγμένων Jscoord[8] όπου με τις κατάλληλες αλλαγές και προσθήκες ικανοποιεί τις απαιτήσεις της εφαρμογής μας. Έτσι αν θέλουμε να κάνουμε έλεγχο της θέσης του χρήστη, όπως για παράδειγμα στη περίπτωση δημοσίευσης ενός γλόμπου, ελέγχουμε τη θέση του αντικειμένου στο terrain χωρίς να χρειάζεται να στέλνουμε τις μεταβλητές (x,y) σε συναρτήσεις άλλων script.

```
            - (3 * eSquared / 8
            + 3 * eSquared * eSquared / 32
            + 45 * eSquared * eSquared * eSquared / 1024)
            * Math.Sin(2 * latitudeRad)
            + (15 * eSquared * eSquared / 256
            + 45 * eSquared * eSquared * eSquared / 1024)
            * Math.Sin(4 * latitudeRad)
            - (35 * eSquared * eSquared * eSquared / 3072)
            * Math.Sin(6 * latitudeRad));

UTMEasting =
    (UTM_F0
     * n
     * (A
       + (1 - t + c) * Math.Pow(A, 3.0) / 6
       + (5 - 18 * t + t * t + 72 * c - 58 * ePrimeSquared)
       * Math.Pow(A, 5.0)
       / 120)
     + 500000.0);

UTMNorthing =
    (UTM_F0
     * (M
       + n
       * Math.Tan(latitudeRad)
       * (A * A / 2
       + (5 - t + (9 * c) + (4 * c * c)) * Math.Pow(A, 4.0) / 24
       + (61 - (58 * t) + (t * t) + (600 * c) - (330 * ePrimeSquared))
       * Math.Pow(A, 6.0)
       / 720)));

Input.location.Stop();

myX = (float)UTMEasting-233345; // convert and substract
myZ = (float)UTMNorthing-3935095;

GameObject.Find("Position").GetComponent<DropPosition>().ChangePostition(myX,myZ);
```

Εικόνα 6.14: Υπολογισμός συντεταγμένων στο terrain

Αν και κανονικά για να προσδιορίσουμε τις συντεταγμένες στο χάρτη θα χρειαζόμασταν και το κωδικό της ζώνης, στη περίπτωση μας δε θα μας χρειαστεί, καθώς δε θέλουμε να υπολογίσουμε τη θέση στο χάρτη UTM αλλά να αναπαραστήσουμε τις συντεταγμένες μας στο καρτεσιανό επίπεδο.

Κατά τη μεταφορά του αντικειμένου προσδιορισμού της θέσης του χρήστη μπορούμε να μεταφέρουμε και τη κάμερα στη συγκεκριμένη θέση χρησιμοποιώντας τις μεταβλητές (x,y) εκτελώντας κάποια από τις συναρτήσεις της κίνησης κάμερας που περιγράψαμε.

Για να αντιστοιχίσουμε τις συντεταγμένες που πήραμε σε μορφή UTM με συντεταγμένες στο χάρτη θα χρειαστεί να πάρουμε δυο σταθερά σημεία. Τα σημεία αυτά θα είναι οι δυο πλατείες του πολυτεχνείου όπου δημιουργούμε δυο κύβους (Εικόνα 6.15). Το x και το z κάθε κύβου θα πρέπει να εκφραστεί σε σχέση με το easting και northing εκείνου του σημείου.

Πριν προχωρήσουμε θα πρέπει να ορίσουμε όλα τα αντικείμενα της σκηνής μας σαν child objects του terrain. Αυτό θα έχει ως αποτέλεσμα, αν μετακινήσουμε ή παραμορφώσουμε το terrain, όλα τα αντικείμενα που έχουμε ορίσει ως childs να ακολουθήσουν τον ίδιο μετασχηματισμό. Έτσι το στήσιμο της σκηνής μας και ο τρισδιάστατος χάρτης δε θα χαλάσει.

Εικόνα 6.15: Σημεία αναφοράς για υπολογισμό συντεταγμένων

Υπολογίζουμε τις συντεταγμένες Easting1 και Northing1 για τη πρώτη πλατεία του πολυτεχνείου και παρατηρούμε ότι οι συντεταγμένες X1 και Z1 του αντίστοιχου κύβου ισούνται με

X1 = Easting1 - 233345 και

Z1 = Northing1 – 3935095

Έπειτα υπολογίζουμε τις συντεταγμένες Easting2 και Northing2 για τη δεύτερη πλατεία του πολυτεχνείου, Θα πρέπει ο δεύτερος κύβος να βρίσκεται στο σημείο

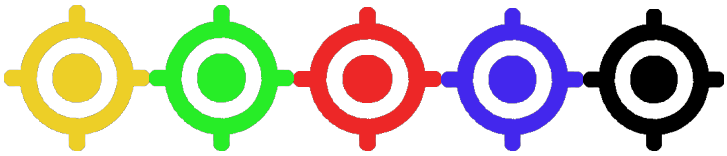
X2 = Easting2 - 233345 και

Z2 = Northing2 – 3935095

Κρατώντας το πρώτο κύβο σε σταθερό σημείο παραμορφώνουμε το terrain μας ώστε η δεύτερη πλατεία και ο δεύτερος κύβος να βρεθούν στο σημείο (x,z) που πρέπει.

Όταν και οι δυο κύβοι είναι στο σωστό σημείο, οποιοδήποτε άλλο ζευγάρι συντεταγμένων πάρουμε σε UTM θα αντικατοπτρίζει με την αφαίρεση των παραπάνω σταθερών το αντίστοιχο σημείο στο δικό μας τρισδιάστατο χάρτη.

Το χρώμα του κουμπιού του GPS της εφαρμογής αλλάζει χρώμα ανάλογα με τη κατάσταση στην οποία βρίσκεται το GPS, (Εικόνα 6.16) αν δηλαδή προσπαθεί να γίνει σύνδεση με το δορυφόρο ή αν αυτή απέτυχε με βάση πληροφορίες που παίρνουμε από τη LocationServiceStatus, αλλά και ανάλογα με το αν ο χρήστης βρίσκεται εντός του χώρου του πολυτεχνείου ή όχι.



Εικόνα 6.16: Χρωματισμός κουμπιού GPS

6.4 Αξιολόγηση

Για να δούμε αν ο απλός χρήστης μπορεί να χρησιμοποιήσει την εφαρμογή μας αλλά και αν η εφαρμογή εξυπηρετεί τις ανάγκες του, καθ' όλη τη διαδικασία υλοποίησης της εφαρμογής έπρεπε να δεχόμαστε συνεχή κριτική από χρήστες της εφαρμογής. Για αυτό το λόγο ένας κλειστός κύκλος από φίλους και συγγενείς μας συμβούλευε σχετικά με αλλαγές ή διορθώσεις που πρέπει να γίνουν.

Η πλειοψηφία των χρηστών που περιηγήθηκαν στα μενού της εφαρμογής προσπάθησαν να κυλήσουν τη μπάρα των μενού με τη χρήση swipes μέσα στο μενού. Η πληθώρα των εφαρμογών για συσκευές με οθόνες αφής υποστηρίζουν χειρονομίες για τη περιήγηση στα μενού αλλά αυτή η λειτουργικότητα αν και φιλική προς τον χρήστη δεν είναι ο κανόνας. Το Unity μας παρέχει λειτουργικότητα των μενού αλλά αυτή δεν υποστηρίζει swipes και χειρονομίες για τη περιήγηση στα μενού. Η περιήγηση στα μενού γίνεται με απλά touch. Ο χρήστης μπορεί να αγγίξει ένα κουμπί για να το ενεργοποιήσει ή να αγγίξει και να σύρει μια μπάρα κύλισης για να περιηγηθεί στο μενού. Αν θέλαμε να υποστηρίζονται χειρονομίες για τη περιήγηση στα μενού τότε θα έπρεπε να υλοποιήσουμε το χειρισμό με τον ίδιο τρόπο που υλοποιήσαμε το χειρισμό της κάμερας. Εμείς στοχεύουμε σε μια εφαρμογή η οποία θα μπορεί να λειτουργήσει σε κάθε συσκευή με όσο το δυνατόν λιγότερε αλλαγές. Για αυτό το λόγο κρατήσαμε το προεπιλεγμένο χειρισμό των μενού με απλά αγγίγματα. Έτσι αν εξάγουμε την εφαρμογή για συσκευές που δεν υποστηρίζουν touch δε χρειάζεται να υλοποιήσουμε διαφορετικό χειρισμό, το Unity φροντίζει ώστε να εντοπίζει ένα άγγιγμα ή κλικ ποντικιού στα στοιχεία του GUI και υλοποιεί αυτόματα την ενέργεια.

Χρήστες οι οποίοι δοκίμασαν τη λειτουργία της εφαρμογής σε αρχικά στάδια επίσης θεώρησαν απαραίτητο στοιχείο την ανάδειξη ενός κτιρίου το οποίο έχει επιλεχθεί. Έτσι εκτός από τη μετακίνηση της κάμερας στη θέση ενός κτιρίου η οποία ήταν από τις βασικές δυνατότητες της εφαρμογής όταν αυτό επιλέγεται, υλοποιήθηκε και ο φωτισμός του, ώστε να μπορεί ο χρήστης να διακρίνει το κτίριο που έχει επιλέξει. Μια ακόμα δυνατότητα η οποία υλοποιήθηκε μετά από προτροπές των χρηστών είναι το μενού βοήθειας. Στο μενού βοήθειας επεξηγούμε το χρωματικό κώδικα του κουμπιού του GPS ώστε να γνωρίζει ο χρήστης τι σημαίνει το κάθε χρώμα, όπως και το χρωματικό κώδικα του κουμπιού δημοσίευσης μηνύματος με τη μορφή γλόμπου αλλά και τη λειτουργία του κουμπιού On/Off. Η πλειοψηφία των χρηστών χωρίς αυτό το μενού βοηθείας φαίνεται πως δυσκολεύτηκε να αντιληφθεί ότι το κουμπί On/Off και το κουμπί δημοσίευσης γλόμπου είναι δυο ξεχωριστά κουμπιά με ξεχωριστές δυνατότητες.

Η απόφαση μας να υλοποιήσουμε τη λειτουργικότητα του GPS με βάση ένα κουμπί το οποίο θα επιλέγει ο χρήστης ώστε να ξεκινήσει η λειτουργία του φαίνεται πως δε δυσκόλεψε στη χρήση αυτής της δυνατότητας. Αντιθέτως, με αυτό το τρόπο ο χρήστης μπορεί να επιλέξει μόνος του αν θα θέσει σε λειτουργία το GPS και σε αντίθεση με άλλες εφαρμογές στις οποίες το GPS λειτουργεί συνέχεια και παίρνει αυτόματα στοιχεία της θέσης των χρηστών έχουμε σημαντική εξοικονόμηση μπαταρίας.

Τέλος απαραίτητη κρίθηκε και η χρήση οδηγού με εικόνες στη σελίδα www.tucapp.gr από όπου οι χρήστες μπορούν να κατεβάσουν την εφαρμογή. Με τη χρήση εικόνων για τη περιγραφή της λειτουργικότητας και των δυνατοτήτων της εφαρμογής οι χρήστες μπορούν να γνωρίζουν το τρόπο λειτουργίας της εφαρμογής αλλά και τις δυνατότητες τις οποίες προσφέρει πριν την εγκαταστήσουν στη συσκευή τους. Έτσι μπορούν να περιηγηθούν στα μενού και να χρησιμοποιήσουν τις δυνατότητες της ακόμα και αν δεν έχουν διαβάσει τις πληροφορίες στο μενού βοηθείας της εφαρμογής.

Η χρήση συνδέσμου dropbox για τη λήψη της εφαρμογής από την ιστοσελίδα δε φαίνεται να δυσκόλεψε πολλούς χρήστες. Παρόλα αυτά ένας μικρός αριθμός χρηστών βρήκε τη διαδικασία λήψης της εφαρμογής μέσω dropbox πολύπλοκη σε σχέση με τη λήψη ενός αρχείου το οποίο φιλοξενείται στο server μας, δυστυχώς όμως η χρήση του συνδέσμου dropbox είναι απαραίτητη ώστε να μειώσουμε το φόρτο στο server μας.

Τέλος όσον αφορά τη πλατφόρμα διαχείρισης του drupal, αν και χρήστες οι οποίοι δεν είχαν μεγάλη εμπειρία σε περιβάλλοντα διαχείρισης ιστοσελίδων αρχικά δυσκολεύτηκαν, μετά από επεξήγηση των δυνατοτήτων της, μπορούσαν εύκολα να προσθέτουν και να επεξεργάζονται περιεχόμενο. Βλέπουμε έτσι ότι η επιλογή του Drupal για την υλοποίηση της πλατφόρμας διαχείρισης ήταν σωστή αφού ακόμα και άπειροι χρήστες μπορούν εύκολα και γρήγορα να κατανοήσουν το τρόπο λειτουργίας της ώστε να διαχειρίζονται την εφαρμογή ακόμα και χωρίς γνώσεις προγραμματισμού.

ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΜΕΛΛΟΝΤΙΚΕΣ ΕΠΕΚΤΑΣΕΙΣ

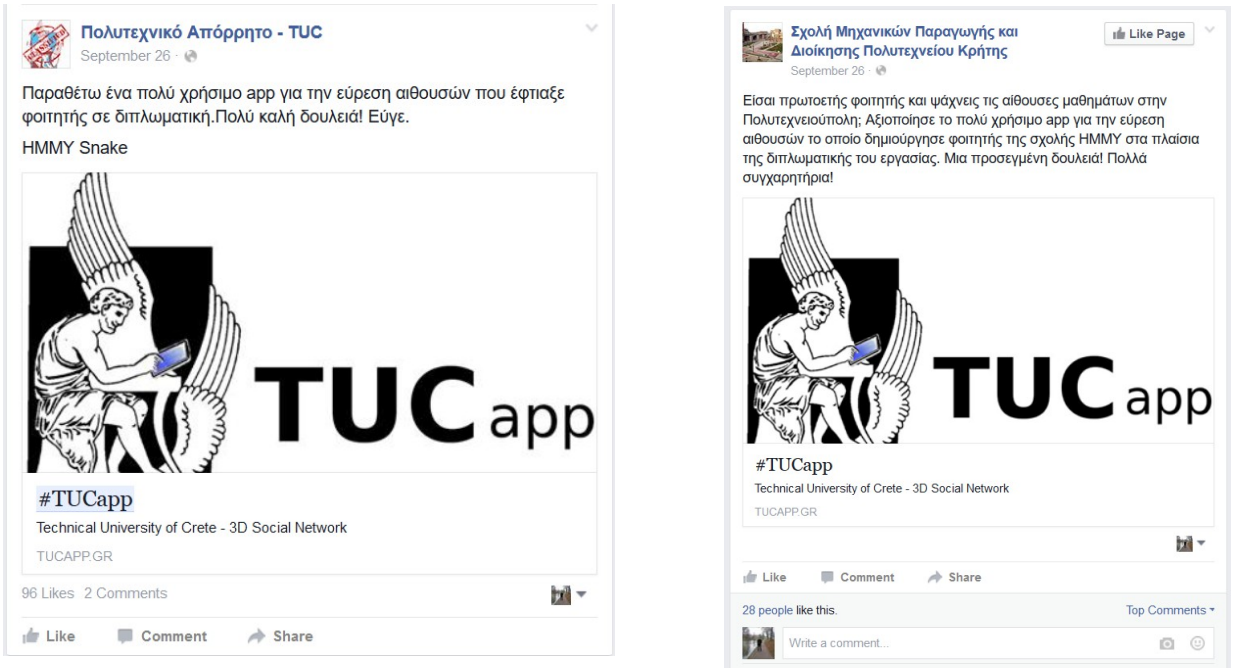
ΚΕΦΑΛΑΙΟ 7

7.1 Συμπεράσματα

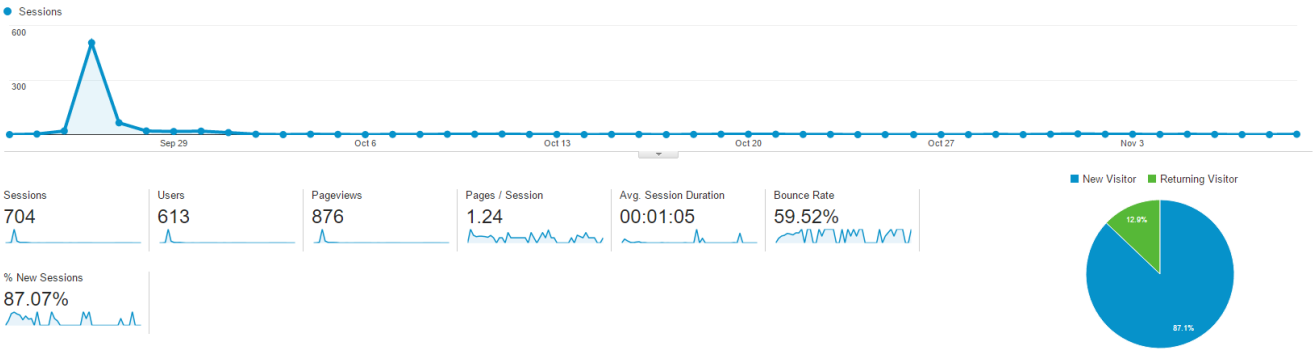
Αφού εξάγουμε την εφαρμογή μας για πλατφόρμα android είμαστε έτοιμοι να τη κάνουμε διαθέσιμη στους χρήστες. Για αυτό το σκοπό δημιουργήθηκε η σελίδα http://tucapp.gr/ (Εικόνα 7.1) την οποία οι χρήστες μπορούν να επισκεφτούν για να κατεβάσουν την εφαρμογή στο κινητό τους.

Εικόνα 7.1: tucapp.gr – ιστοσελίδα φιλοξενίας εφαρμογής

Η δημοσίευση του συνδέσμου έγινε στο φόρουμ του πολυτεχνείου όπου το site έγινε γνωστό στη κοινότητα του πολυτεχνείου. Μετά από δυο αναδημοσιεύσεις του συνδέσμου από δυο σελίδες στο facebook (Εικόνα 7.2) οι οποίες απαρτίζονται κυρίως από μέλη του πολυτεχνείου, παρατηρήθηκε μεγάλη επισκεψιμότητα στη σελίδα μας (Εικόνα 7.3) η οποία σε συνδυασμό με το module google analytics που έχουμε προσθέσει στο site μας μας βοηθά να βγάλουμε κάποια στατιστικά συμπεράσματα.



Εικόνα 7.2: Σχολιασμός στα social media



Εικόνα 7.3: Γράφημα επισκεψιμότητας

Βλέπουμε ότι την ημέρα που έγινε η αναδημοσίευση του συνδέσμου στις δυο αυτές σελίδες η επισκεψιμότητα ανέβηκε θεαματικά. Αρκετοί χρήστες δήλωσαν ότι τους αρέσει η εφαρμογή είτε με σχόλιο είτε επιλέγοντας μου αρέσει στο facebook και η μεγάλη διαφορά ανταπόκρισης στις δημοσιεύσεις για το TUCapp σε σχέση με άλλες δημοσιεύσεις στις σελίδες αυτές υποδεικνύει ότι τα μέλη του πολυτεχνείου θεωρούν πως μια εφαρμογή ενός τρισδιάστατου χάρτη του πολυτεχνείου, όπου οι φοιτητές θα μπορούν να βρουν πληροφορίες και να αλληλεπιδράσουν μεταξύ τους θα μπορούσε να φανεί χρήσιμη.

Αλλά η ανταπόκριση των χρηστών δεν αρκεί ώστε να εξάγουμε συμπεράσματα σχετικά με τους επισκέπτες της σελίδας μας. Από τα γραφήματα που παίρνουμε από τα google μπορούμε να δούμε τι λειτουργικό είχαν οι συσκευές των χρηστών που επισκέφτηκαν το site της εφαρμογής μας (Εικόνα 7.4). Να σημειώσουμε ότι στις δημοσιεύσεις του συνδέσμου, δεν αναφέρεται ότι η εφαρμογή είναι διαθέσιμη μόνο για android συσκευές άρα οι χρήστες δε μπορούσαν να γνωρίζουν αν η εφαρμογή είναι διαθέσιμη για τη συσκευή τους.

	Operating System	Acquisition			Behavior		
		Sessions ↓	% New Sessions	New Users	Bounce Rate	Pages / Session	Avg. Session Duration
		704 % of Total: 100.00% (704)	87.07% Avg for View: 87.07% (0.00%)	613 % of Total: 100.00% (613)	59.52% Avg for View: 59.52% (0.00%)	1.24 Avg for View: 1.24 (0.00%)	00:01:05 Avg for View: 00:01:05 (0.00%)
☐	1. Windows	410 (58.24%)	84.39%	346 (56.44%)	71.22%	1.23	00:01:10
☐	2. Android	203 (28.84%)	91.13%	185 (30.18%)	27.09%	1.33	00:01:10
☐	3. iOS	52 (7.39%)	92.31%	48 (7.83%)	84.62%	1.13	00:00:32
☐	4. Linux	21 (2.98%)	95.24%	20 (3.26%)	66.67%	1.05	00:00:50
☐	5. Macintosh	15 (2.13%)	86.67%	13 (2.12%)	80.00%	1.20	00:00:02
☐	6. Windows Phone	3 (0.43%)	33.33%	1 (0.16%)	66.67%	1.00	00:00:21

Εικόνα 7.4: Στατιστικά λειτουργικού συστήματος χρηστών

Παρατηρούμε ότι η μεγαλύτερη πλειοψηφία των επισκεπτών χρησιμοποιεί υπολογιστή με windows. Η δεύτερη καταχώρηση είναι οι συσκευές android και τρίτες συσκευές iOS. Βλέπουμε ότι οι χρήστες με android συσκευές είναι πολύ περισσότεροι από χρήστες με κινητά τηλέφωνα άλλων λειτουργικών. Η επιλογή μας λοιπόν να προγραμματίσουμε μια android εφαρμογή ήταν σωστή καθώς έτσι εξυπηρετούμε μεγαλύτερο αριθμό χρηστών, αλλά θα μπορούσαμε να έχουμε υλοποιήσει και την εφαρμογή για υπολογιστές windows αφού ο αριθμός των επισκεπτών από υπολογιστή είναι πάνω από το μισό του συνολικού αριθμού επισκεπτών.

Χάρη στις δυνατότητες του Unity μπορούμε εύκολα να εξάγουμε την εφαρμογή μας και για άλλες πλατφόρμες, αρκεί να υλοποιήσουμε το κατάλληλο χειρισμό.

Όσον αφορά τις συσκευές κινητής τηλεφωνίας παρατηρούμε ότι αν και οι χρήστες android είναι περισσότεροι στη κορυφή της λίστας μοντέλων κινητών είναι το Apple iphone (Εικόνα 7.5). Αυτό συμβαίνει διότι η ποικιλία συσκευών android είναι κατά πολύ μεγαλύτερη από αυτή των συσκευών με iOS. Αυτό μας υπενθυμίζει ότι όταν προγραμματίζουμε σε android πρέπει να λαμβάνουμε υπ' όψιν ότι η εφαρμογή μας θα πρέπει να είναι συμβατή με όλες τις συσκευές android. Πρέπει λοιπόν πάντα να φτιάχνουμε τα μενού μας με τέτοιο τρόπο ώστε να εμφανίζονται σωστά σε κάθε συσκευή ανεξαρτήτου μεγέθους οθόνης.

Mobile Device Info	Sessions	% New Sess	New Users	Bounce Rate	Pages / Ses	Avg. Sessior	Goal Convers	Goal Comple	Goal Value
Apple iPhone	48	91.67%	44	83.33%	1.15	34.75	0.00%	0	0.00
(not set)	14	100.00%	14	28.57%	1.50	108.64	0.00%	0	0.00
Samsung GT-I950	11	100.00%	11	18.18%	1.36	65.18	0.00%	0	0.00
Sony E2303 Xperi	9	66.67%	6	22.22%	1.22	17.44	0.00%	0	0.00
Samsung GT-I930	8	100.00%	8	12.50%	1.38	38.75	0.00%	0	0.00
Samsung SM-G8(	8	62.50%	5	37.50%	1.38	62.88	0.00%	0	0.00
LG D802 G2	7	85.71%	6	57.14%	1.00	20.00	0.00%	0	0.00
Samsung GT-I919	7	85.71%	6	42.86%	1.29	29.43	0.00%	0	0.00
LG D855 G3	6	100.00%	6	33.33%	1.17	34.50	0.00%	0	0.00
Samsung SM-G9(	5	100.00%	5	0.00%	2.00	132.00	0.00%	0	0.00
Apple iPad	4	100.00%	4	100.00%	1.00	0.00	0.00%	0	0.00
Google Nexus 5	4	100.00%	4	0.00%	1.00	10.25	0.00%	0	0.00
Huawei ALE-L21	4	100.00%	4	0.00%	1.50	57.25	0.00%	0	0.00
Huawei H30-T00 H	4	100.00%	4	0.00%	1.00	14.50	0.00%	0	0.00
LG D620 G2 Mini	4	75.00%	3	50.00%	1.75	243.00	0.00%	0	0.00
Motorola XT1032 M	4	100.00%	4	25.00%	2.00	123.25	0.00%	0	0.00
Samsung GT-I826	4	100.00%	4	50.00%	1.00	21.50	0.00%	0	0.00
Samsung GT-I906	4	100.00%	4	0.00%	1.00	76.25	0.00%	0	0.00
Samsung GT-I930	4	50.00%	2	50.00%	1.25	72.75	0.00%	0	0.00
Samsung SM-N75	4	50.00%	2	50.00%	1.00	10.25	0.00%	0	0.00
Huawei T1 MediaP	3	33.33%	1	33.33%	1.67	524.00	0.00%	0	0.00
Lenovo A5000	3	100.00%	3	0.00%	2.67	415.33	0.00%	0	0.00
LG D802T G2	3	100.00%	3	0.00%	1.67	90.67	0.00%	0	0.00
Microsoft Lumia 5	3	33.33%	1	66.67%	1.00	21.00	0.00%	0	0.00
Samsung SM-A3(	3	66.67%	2	33.33%	1.00	57.67	0.00%	0	0.00
KingZone N3 Plus	2	100.00%	2	50.00%	1.00	20.00	0.00%	0	0.00
LG D722 G3 Mini	2	100.00%	2	50.00%	1.00	14.00	0.00%	0	0.00
Mozilla Firefox for	2	100.00%	2	50.00%	1.00	3.00	0.00%	0	0.00
OnePlus A0001 O	2	100.00%	2	50.00%	1.00	12.00	0.00%	0	0.00
Samsung GT-I919	2	50.00%	1	50.00%	1.00	44.00	0.00%	0	0.00
Samsung GT-I950	2	100.00%	2	0.00%	1.00	8.00	0.00%	0	0.00
Samsung I9506 G	2	100.00%	2	0.00%	1.50	46.50	0.00%	0	0.00
Samsung SM-G9(	2	50.00%	1	50.00%	1.00	19.50	0.00%	0	0.00
Samsung SM-G9:	2	100.00%	2	0.00%	1.00	9.00	0.00%	0	0.00
Samsung SM-N9(	2	100.00%	2	0.00%	1.50	26.00	0.00%	0	0.00
Samsung SM-N91	2	100.00%	2	50.00%	1.50	24.50	0.00%	0	0.00
Samsung SM-T31	2	100.00%	2	50.00%	1.00	4.50	0.00%	0	0.00
Sony C1905 Xperi	2	100.00%	2	0.00%	1.50	336.50	0.00%	0	0.00
Sony D2303 Xperi	2	100.00%	2	50.00%	1.00	12.50	0.00%	0	0.00
Alcatel 6043D On	1	100.00%	1	0.00%	3.00	64.00	0.00%	0	0.00
Alcatel ONE TOU(	1	100.00%	1	100.00%	1.00	0.00	0.00%	0	0.00
Alcatel OT-4033D	1	100.00%	1	0.00%	1.00	165.00	0.00%	0	0.00
Cubot X9	1	100.00%	1	0.00%	2.00	69.00	0.00%	0	0.00
Gigabyte GSmart	1	100.00%	1	0.00%	1.00	33.00	0.00%	0	0.00
HTC Desire Desire	1	100.00%	1	0.00%	1.00	67.00	0.00%	0	0.00
HTC M8 One M8	1	100.00%	1	0.00%	2.00	45.00	0.00%	0	0.00
Huawei Y550-L01	1	100.00%	1	100.00%	1.00	0.00	0.00%	0	0.00
Lenovo A10-70F L	1	100.00%	1	0.00%	1.00	15.00	0.00%	0	0.00
Lenovo A328	1	100.00%	1	100.00%	1.00	0.00	0.00%	0	0.00
Lenovo A536	1	100.00%	1	0.00%	1.00	13.00	0.00%	0	0.00
Lenovo A6000	1	100.00%	1	100.00%	1.00	0.00	0.00%	0	0.00
Lenovo S650	1	100.00%	1	0.00%	1.00	19.00	0.00%	0	0.00
Lenovo S960 Vibe	1	100.00%	1	100.00%	1.00	0.00	0.00%	0	0.00
Lenovo X2-CU Vib	1	100.00%	1	0.00%	2.00	104.00	0.00%	0	0.00
LG D280	1	100.00%	1	0.00%	1.00	19.00	0.00%	0	0.00
LG D320 Optimus	1	100.00%	1	100.00%	1.00	0.00	0.00%	0	0.00
LG D390n	1	100.00%	1	100.00%	1.00	0.00	0.00%	0	0.00
LG D605 Optimus	1	100.00%	1	100.00%	1.00	0.00	0.00%	0	0.00
LG P710 Optimus	1	100.00%	1	0.00%	1.00	24.00	0.00%	0	0.00
LG V500 G Pad 8	1	100.00%	1	0.00%	1.00	28.00	0.00%	0	0.00
Meizu MX4 Pro	1	100.00%	1	0.00%	3.00	65.00	0.00%	0	0.00
Motorola XT1068 M	1	100.00%	1	0.00%	1.00	19.00	0.00%	0	0.00
Samsung GT-I819	1	100.00%	1	0.00%	1.00	21.00	0.00%	0	0.00
Samsung GT-I820	1	100.00%	1	0.00%	1.00	78.00	0.00%	0	0.00
Samsung GT-I820	1	100.00%	1	0.00%	1.00	27.00	0.00%	0	0.00
Samsung GT-I826	1	100.00%	1	0.00%	1.00	69.00	0.00%	0	0.00
Samsung GT-I906	1	100.00%	1	0.00%	1.00	27.00	0.00%	0	0.00
Samsung GT-I910	1	100.00%	1	100.00%	1.00	0.00	0.00%	0	0.00
Samsung GT-I951	1	100.00%	1	0.00%	1.00	4.00	0.00%	0	0.00
Samsung GT-N71	1	100.00%	1	0.00%	1.00	20.00	0.00%	0	0.00
Samsung GT-N71	1	100.00%	1	0.00%	1.00	13.00	0.00%	0	0.00
Samsung GT-S75	1	100.00%	1	0.00%	1.00	13.00	0.00%	0	0.00
Samsung SM-G11	1	100.00%	1	100.00%	1.00	0.00	0.00%	0	0.00
Samsung SM-G3	1	100.00%	1	0.00%	1.00	51.00	0.00%	0	0.00
Samsung SM-G3(	1	100.00%	1	0.00%	2.00	56.00	0.00%	0	0.00
Samsung SM-G3	1	100.00%	1	0.00%	1.00	30.00	0.00%	0	0.00
Samsung SM-G8	1	100.00%	1	0.00%	2.00	893.00	0.00%	0	0.00
Samsung SM-G8	1	100.00%	1	0.00%	1.00	28.00	0.00%	0	0.00
Sony C5303 Xperi	1	100.00%	1	0.00%	1.00	18.00	0.00%	0	0.00
Sony C6802 Xperi	1	100.00%	1	0.00%	2.00	52.00	0.00%	0	0.00
Sony C6903 Xperi	1	100.00%	1	0.00%	1.00	11.00	0.00%	0	0.00
Sony D2302 Xperi	1	100.00%	1	100.00%	1.00	0.00	0.00%	0	0.00
Sony D2403 Xperi	1	100.00%	1	0.00%	1.00	20.00	0.00%	0	0.00
Sony D5306 Xperi	1	100.00%	1	100.00%	1.00	0.00	0.00%	0	0.00
Sony D5503 Xperi	1	100.00%	1	0.00%	5.00	651.00	0.00%	0	0.00
Sony D6603 Xperi	1	100.00%	1	100.00%	1.00	0.00	0.00%	0	0.00
SonyEricsson C6(	1	100.00%	1	0.00%	2.00	58.00	0.00%	0	0.00
Vodafone 895n Sr	1	100.00%	1	0.00%	2.00	0.00	0.00%	0	0.00
Vodafone 990N Sr	1	100.00%	1	0.00%	1.00	32.00	0.00%	0	0.00
Xiaomi MI 4LTE	1	100.00%	1	100.00%	1.00	0.00	0.00%	0	0.00
ZTE V788D Kis Pl	1	100.00%	1	0.00%	1.00	17.00	0.00%	0	0.00
	258	90.70%	234	39.15%	1.28	62.13	0.00%	0	0.00

Εικόνα 7.5: Μοντέλα φορητών συσκευών χρηστών

	Screen Resolution	Acquisition			Behavior			Conversions		
		Sessions ↓	% New Sessions	New Users	Bounce Rate	Pages / Session	Avg. Session Duration	Goal Conversion Rate	Goal Completions	Goal Value
		258 % of Total: 36.65% (704)	90.70% Avg for View: 87.07% (4.16%)	234 % of Total: 38.17% (613)	39.15% Avg for View: 59.52% (-34.23%)	1.28 Avg for View: 1.24 (3.10%)	00:01:02 Avg for View: 00:01:05 (-4.26%)	0.00% Avg for View: 0.00% (0.00%)	0 % of Total: 0.00% (0)	$0.00 % of Total: 0.00% ($0.00)
1.	360x640	129 (50.00%)	89.15%	115 (49.15%)	22.48%	1.40	00:01:15	0.00%	0 (0.00%)	$0.00 (0.00%)
2.	320x568	26 (10.08%)	96.15%	25 (10.68%)	84.62%	1.15	00:00:04	0.00%	0 (0.00%)	$0.00 (0.00%)
3.	480x800	15 (5.81%)	100.00%	15 (6.41%)	20.00%	1.00	00:00:48	0.00%	0 (0.00%)	$0.00 (0.00%)
4.	320x480	11 (4.26%)	100.00%	11 (4.70%)	72.73%	1.27	00:02:21	0.00%	0 (0.00%)	$0.00 (0.00%)
5.	720x1280	11 (4.26%)	81.82%	9 (3.85%)	27.27%	1.09	00:00:36	0.00%	0 (0.00%)	$0.00 (0.00%)
6.	360x592	10 (3.88%)	100.00%	10 (4.27%)	50.00%	1.20	00:00:30	0.00%	0 (0.00%)	$0.00 (0.00%)
7.	320x534	7 (2.71%)	100.00%	7 (2.99%)	14.29%	1.14	00:00:36	0.00%	0 (0.00%)	$0.00 (0.00%)
8.	375x667	7 (2.71%)	100.00%	7 (2.99%)	85.71%	1.00	00:00:02	0.00%	0 (0.00%)	$0.00 (0.00%)
9.	320x570	6 (2.33%)	100.00%	6 (2.56%)	50.00%	1.17	00:01:54	0.00%	0 (0.00%)	$0.00 (0.00%)
10.	768x1024	5 (1.94%)	100.00%	5 (2.14%)	80.00%	1.00	00:00:02	0.00%	0 (0.00%)	$0.00 (0.00%)

Show rows: 10 Go to: 1 1 - 10 of 31

Εικόνα 7.6: Χαρακτηριστικά οθονών χρηστών

Η ανάγκη υλοποίησης της εφαρμογής μας με τέτοιο τρόπο ώστε τα μενού να προβάλλονται σωστά ανεξαρτήτου μεγέθους οθόνης φαίνεται καθαρά στην αντίστοιχη καρτέλα των analytics, όπου βλέπουμε πως οι συσκευές που επισκέφτηκαν τη σελίδα μας είχαν συνολικά 31 διαφορετικές αναλύσεις οθόνης (Εικόνα 7.6).

Αντιλαμβανόμαστε λοιπόν ότι η υλοποίηση με Unity διευκολύνει σημαντικά την υλοποίηση μιας τέτοιας εφαρμογής. Μπορούμε με τις εξειδικευμένες κλάσεις του να προγραμματίσουμε εφαρμογές που όχι μόνο καλύπτουν τις ανάγκες των χρηστών υποστηρίζοντας συσκευές με διαφορετικό λειτουργικό, αφού μπορούμε να εξάγουμε την εφαρμογή μας για κάθε λειτουργικό συσκευής η οποία επισκέφτηκε την ιστοσελίδα, αλλά μας δίνει και τη δυνατότητα προγραμματισμού μενού των οποίων η εμφάνιση θα παραμένει σταθερή σε κάθε οθόνη.

Ο προγραμματισμός σε C# σε συνδυασμό με τις δυνατότητες του .Net Framework το οποίο υποστηρίζει, αλλά και η αντικειμενοστραφής λογική την οποία υιοθετεί το Unity, μας βοηθά να δημιουργήσουμε αρχεία τα οποία θα είναι επαναχρησιμοποιήσιμα. Κάθε prefab του project μας μπορεί να χρησιμοποιηθεί σε άλλα project χωρίς να χρειαστεί περαιτέρω ρυθμίσεις.

Αυτή η αντικειμενοστραφής λογική μας βοηθά να εργαστούμε παράλληλα με μια πλατφόρμα Drupal, ώστε να δημιουργήσουμε εφαρμογές στις οποίες η προσθήκη νέων δυνατοτήτων θα απαιτούν όσο το δυνατόν λιγότερες εργατοώρες.

Πλέον έχουμε δημιουργήσει ένα σύστημα στο οποίο μπορούμε εύκολα να προσθέσουμε νέα μενού, απλώς γράφοντας το αντίστοιχο script για το μενού στο Unity και προσθέτοντας τη συνάρτηση ανάκτησης πληροφοριών στο module του Drupal. Ακόμα και η προσθήκη νέων κτιρίων είναι απλή υπόθεση, αφού αρκεί να προσθέσουμε μια νέα καταχώρηση στο Taxonomy του Drupal για να γίνει η αντιστοίχιση.

Με τη χρήση της πλατφόρμας διαχείρισης του Drupal μπορεί οποιοσδήποτε, ακόμα και χωρίς γνώσεις προγραμματισμού, να προσθέσει και να επεξεργαστεί περιεχόμενο της βάσης δεδομένων το οποίο εμφανίζεται σε πραγματικό χρόνο στους χρήστες της εφαρμογής. Αρκετά σημαντικό επίτευγμα, αφού έτσι η διατήρηση του περιεχόμενου της εφαρμογής ενημερωμένου, μπορεί να γίνει από ένα μόνον άτομο, χωρίς τη παρέμβαση του προγραμματιστή. Έτσι εκτός από την ενημέρωση των πληροφοριών της εφαρμογής δημιουργείται και συνεχώς εμπλουτίζεται μια βάση δεδομένων η οποία περιέχει συγκεντρωμένες όλες τις πληροφορίες σχετικά με τις εγκαταστάσεις, τους υπαλλήλους και τις

υπηρεσίες του πολυτεχνείου ώστε να μπορεί ο χρήστης της εφαρμογής να τις βρει όλες σε ένα μέρος.

Η δυνατότητα ψηφοφορίας δημοσιεύσεων των χρηστών με ανάδειξη εκείνων οι οποίες έχουν υψηλή βαθμολογία και απόκρυψη αυτών που έχουν καταψηφιστεί, αλλά και η δυνατότητα παραχώρησης δικαιωμάτων διαχείρισης του δικού τους περιεχομένου σε υπαλλήλους του πολυτεχνείου επιτυγχάνει τη μείωση του φόρτου εργασίας για τον διαχειριστή περιεχομένου εξασφαλίζοντας μεγαλύτερη διάρκεια ζωής για την εφαρμογή αφού ο χρόνος που του ζητείται να αφιερώνει μειώνεται για κάθε χρήστη.

Δοκιμάζοντας τη λειτουργία της εφαρμογής στο Samsung Galaxy S III, κινητό αρκετά παλιό κατά την εκπόνηση αυτής της διπλωματικής, και βλέποντας τη να λειτουργεί χωρίς προβλήματα ακόμα και με δεκάδες τρισδιάστατα μοντέλα να πραγματοποιούν κινήσεις και κρούσεις, καταλαβαίνουμε ότι η 3D απεικόνιση είναι το μέλλον της ψηφιακής εικόνας.

7.2 Μελλοντικές επεκτάσεις

Έχοντας δημιουργήσει ένα σύστημα το οποίο μπορεί να δεχτεί προσθήκες με μεγάλη ευκολία, είτε αυτές είναι προσθήκες δεδομένων ή ακόμα και νέων δυνατοτήτων, ο μόνος περιορισμός μας για μελλοντικές επεκτάσεις είναι μόνο η φαντασία μας.

Πέραν του ότι το σύστημα μας μπορεί να παραμετροποιηθεί ώστε να καλύπτει τις ανάγκες οποιουδήποτε εκπαιδευτικού ιδρύματος, η προσθήκη νέων δυνατοτήτων είναι στην ουσία προσθήκη ενός επιπλέον μενού.

Αν θέλαμε για παράδειγμα να δώσουμε τη δυνατότητα στους φοιτητές να δημιουργούν ομάδες μελέτης, θα μπορούσαμε να το κάνουμε απλώς προσθέτοντας τις κατάλληλες συναρτήσεις στο module του drupal και υλοποιώντας τη γραφική διεπαφή σε ένα C# script στο Unity.Οι χρήστες θα μπορούσαν να έχουν τη δυνατότητα ψήφισης και σχολιασμού ενός μαθήματος αρκεί να το παρακολούθησαν. Έλεγχος ο οποίος μπορεί να γίνει με τη χρήση GPS.

Σε επόμενες εκδόσεις της εφαρμογής θα μπορούσε ο χρήστης να λαμβάνει πληροφορίες για τα δρομολόγια λεωφορείων ή να δοθεί η δυνατότητα στις γραμματείες των σχολών να στέλνουν ειδοποιήσεις push notifications στους χρήστες. Ομοίως θα μπορούσαμε να υλοποιήσουμε αυτόματες ειδοποιήσεις του συστήματος, οι οποίες θα αποστέλλονται στο χρήστη αν για παράδειγμα έχει καιρό να παρευρεθεί σε μάθημα ώστε να τον αφυπνίσει.

Φυσικά εκτός από το πρόγραμμα του εξαμήνου σε μελλοντικές εκδόσεις, είναι απαραίτητο να προστεθεί η δυνατότητα αποθήκευσης προγράμματος εξεταστικής.

Χάρη στην ευκολία προγραμματισμού με Unity και Drupal, η προσθήκη δυνατότητας κράτησης αίθουσας ή γηπέδου στις αθλητικές εγκαταστάσεις του πολυτεχνείου μέσω της εφαρμογής ακόμα και για επισκέπτες, είναι πολύ πιο εύκολη από ότι φαντάζει αφού μπορούμε απλώς να προσθέσουμε τα κατάλληλα πεδία στο content type των κτιρίων για να αποθηκεύουν τις κρατήσεις και να δημιουργήσουμε το μενού που θα καλεί τη συνάρτηση τροποποίησης του πεδίου.

Από απλά μενού μέχρι δυνατότητες περιήγησης στους δρόμους του πολυτεχνείου, ή ακόμα και μέσα στα κτίρια, το Unity μας δίνει τα εργαλεία να τα υλοποιήσουμε.

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. http://tf3dm.com/3d-model/light-bulb-29897.html
2. http://tf3dm.com/3d-model/low-poly-tree-24775.html
3. http://duckfiles.com/education-icons-set-free-psd-and-png/
4. http://stackoverflow.com
5. https://en.wikipedia.org
6. https://msdn.microsoft.com/
7. http://www.unity3d.com
8. http://www.jstott.me.uk/jscoord/
9. http://www.earthpoint.us/Convert.aspx
10. http://docs.unity3d.com/Manual/ExecutionOrder.html
11. http://birdseyestock.deviantart.com/
12. https://github.com/mtschoen/JSONObject
13. https://en.wikipedia.org/wiki/Uniform_Resource_Identifier
14. https://en.wikipedia.org/wiki/World_Geodetic_System
15. https://en.wikipedia.org/wiki/Universal_Transverse_Mercator_coordinate_system
16. https://en.wikipedia.org/wiki/Transverse_Mercator_projection
17. Νίκος Μ. Αφεντάκης : Μελέτη Σήμανσης Για τις Εγκαταστάσεις της Πολυτεχνειούπολης στο Ακρωτήρι Χανίων Κρήτης, 2004