ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ «ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ και ΥΠΟΛΟΓΙΣΤΩΝ»

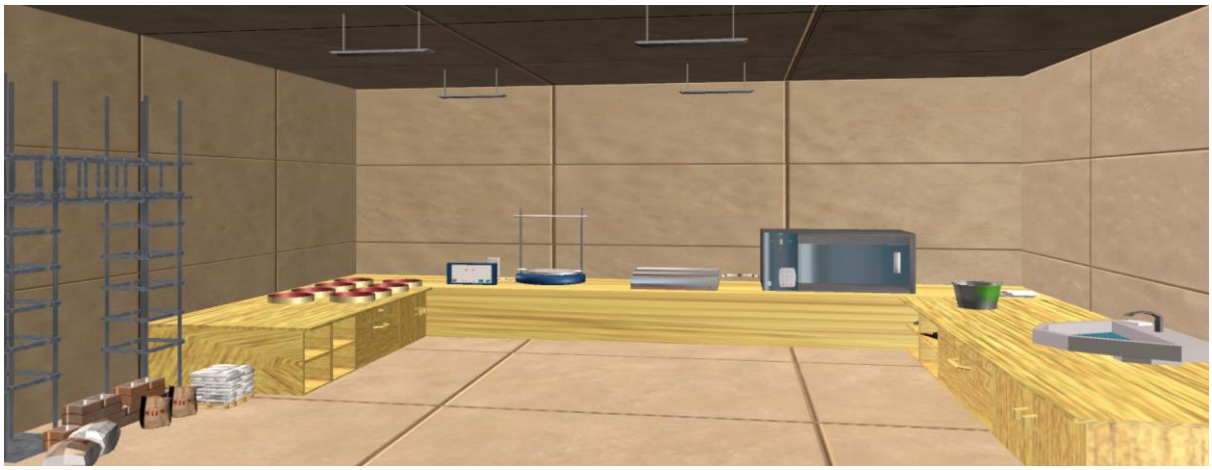
ΜΕΤΑΠΤΥΧΙΑΚΟ ΠΡΟΓΡΑΜΜΑ «ΕΠΙΣΤΗΜΗ και ΤΕΧΝΟΛΟΓΙΑ της ΠΛΗΡΟΦΟΡΙΚΗΣ και των ΥΠΟΛΟΓΙΣΤΩΝ»

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Τα 3D Games στην εκπαίδευση και δημιουργία εκπαιδευτικού παιχνιδιού»

Ουρανία Γολεμάτη

ΕΠΙΒΛΕΠΟΥΣΑ ΚΑΘΗΓΗΤΡΙΑ: Δρ. Σγουροπούλου Κλειώ



Αθήνα, Μάιος 2020

Μέλη Επιτροπής

KLEIO SGOUROPOULOU
Digitally signed by KLEIO SGOUROPOULOU
Date: 2021.06.08 19:08:27 +03'00'

Επιβλέπουσα Καθηγήτρια

Σγουροπούλου Κλειώ

Καθηγήτρια

Ioannis Voyiatzis
Digitally signed by Ioannis Voyiatzis
Date: 2021.06.08 19:38:45 +03'00'

Συν-Επιβλέπων Καθηγητής

Βογιατζής Ιωάννης

Καθηγητής

Athanasios Voulodimos
Digitally signed by Athanasios Voulodimos
Date: 2021.06.08 19:37:07 +03'00'

Συν-Επιβλέπων Καθηγητής

Βουλόδημος Αθανάσιος

Επίκουρος Καθηγητής

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΜΕΤΑΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

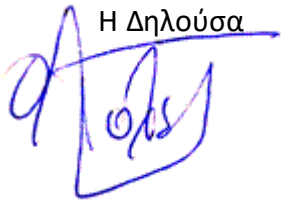
Ο/η κάτωθι υπογεγραμμένος/η Γολεμάτη Ουρανία του Χρήστου, με αριθμό μητρώου 18032 φοιτητής/τρια του Προγράμματος Μεταπτυχιακών Σπουδών Επιστήμη και Τεχνολογία της Πληροφορικής και των Υπολογιστών του Τμήματος Μηχανικών Πληροφορικής και Υπολογιστών της Σχολής Μηχανικών του Πανεπιστημίου Δυτικής Αττικής, δηλώνω ότι:

«Είμαι συγγραφέας αυτής της μεταπτυχιακής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.

Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου».

Επιθυμώ την απαγόρευση πρόσβασης στο πλήρες κείμενο της εργασίας μου μέχρι ..... και έπειτα από αίτηση μου στη Βιβλιοθήκη και έγκριση του επιβλέποντα καθηγητή.

Η Δηλούσα



Αφιερώνω την επιτυχή ολοκλήρωση της διπλωματικής,

σε όλους όσους δεν είναι πλέον δίπλα μου…

Περίληψη: Η ραγδαία ανάπτυξη της τεχνολογίας και ειδικότερα του τομέα της πληροφορικής, μας έχει δώσει πλέον την δυνατότητα να μπορούμε να προσομοιώνουμε το χειρισμό αντικειμένων και να δημιουργούμε εικονικές πραγματικότητες με τη χρήση υπολογιστικών συστημάτων. Μπορούμε πλέον να προσομοιώσουμε εργαστηριακά πειράματα τα οποία ειδικά στην εκπαίδευση, μπορούν να είναι τόσο εκπαιδευτικά χρήσιμα, όσο και η δια ζώσης παρακολούθηση ενός εργαστηρίου. Αυτές οι προσομοιώσεις γίνονται πολύ επιτυχημένες, όταν πραγματοποιούνται σε ένα εύχρηστο και ευχάριστο περιβάλλον το οποίο είναι ένα 3D παιχνίδι.

Μπορούμε πλέον να δημιουργήσουμε 3D παιχνίδια για εργαστηριακά πειράματα, τα οποία ειδικά στην εκπαίδευση, μπορούν να είναι τόσο εκπαιδευτικά χρήσιμα, όσο και η δια ζώσης παρακολούθηση ενός εργαστηρίου. Ερευνώντας στο διαδίκτυο μπορούμε να εντοπίσουμε πολλά ξένα Πανεπιστήμια τα οποία ήδη εφαρμόζουν τα virtual labs ή 3D games σε πολλά επιστημονικά πεδία. Ειδικότερα στις ιστοσελίδες: http://vlab.co.in και http://virtual-labs.ac.in, βλέπουμε ότι συνεργάζονται 12 ξένα Πανεπιστήμια για τα εικονικά εργαστήρια σε 10 διαφορετικά γνωστικά πεδία, όπως και στη σελίδα https://www.labster.com/ με 72 προσομοιώσεις. Αλλά και στην Ελλάδα οι προσομοιώσεις κερδίζουν συνέχεια έδαφος.

Ο στόχος του 3D παιχνιδιού του εργαστηριακού πειράματος «Προσδιορισμός κοκκομετρικής σύνθεσης αδρανούς υλικού», είναι να αναπτυχθεί μια εφαρμογή η οποία θα χρησιμοποιηθεί από σπουδαστές του γνωστικού αντικειμένου του Πολιτικού Μηχανικού. Είναι γνωστό ότι πέρα από το θεωρητικό μέρος ενός μαθήματος, σημαντικό ρόλο παίζει και η εμπειρία που αποκτάνε οι σπουδαστές μέσα από τη διαδικασία των εργαστηρίων. Εδώ έρχεται να συμπληρώσει αυτή τη γνώση, η τρέχουσα εφαρμογή. Εφόσον οι σπουδαστές διδάσκονται το θεωρητικό υπόβαθρο ενός μαθήματος, και ταυτόχρονα βιώνουν στο εργαστήριο τις διάφορες πειραματικές τεχνικές, τους δίνεται η δυνατότητα μέσω των προσομοιώσεων, να μπορούν να επαναλάβουν κάποια διαδικασία και να εξασκηθούν στις γνώσεις που έχουν αποκτήσει.

Για τη παρούσα πτυχιακή εργασία, χρησιμοποιήθηκε η πλατφόρμα υλοποίησης παιχνιδιών της Unity. Από τις γλώσσες προγραμματισμού που μπορούσαν να χρησιμοποιηθούν (C#, JavaScript και Boo) επιλέχθηκε η C#.

Το 3D παιχνίδι, παρ΄ ότι έχει την πειραματική διάταξη ήδη σχεδιασμένη και αμετάβλητη, δηλαδή δεν μπορεί ο χρήστης να παραμετροποιήσει τα δεδομένα, θεωρείται ιδιαίτερα σημαντικό ως μαθησιακή δραστηριότητα της γνώσης που αποκτάει ο σπουδαστής. Στο άμεσο μέλλον, η προσομοίωση εργαστηριακού πειράματος μέσω παιχνιδιών προβλέπεται να γίνει το βέλτιστο «εργαλείο» εκμάθησης και υποστήριξης της γνώσης, σε πολλά επιστημονικά πεδία.

Λέξεις-κλειδιά: 3D παιχνίδια, Προσομοίωση Εργαστηριακού Πειράματος, Πολιτικός Μηχανικός, Unity, C#.

Περιεχόμενο: Κείμενο, πίνακες, εικόνες.

Περιεχόμενα

1. ΕΙΣΑΓΩΓΗ

1.1 Εκπαιδευτικά Λογισμικά

Ο χαρακτηρισμός ενός λογισμικού ως εκπαιδευτικού λαμβάνει υπόψη του τόσο την παιδαγωγική όσο και την τεχνολογική διάσταση. Το εκπαιδευτικό λογισμικό θεωρείται ότι εμπεριέχει διδακτικούς στόχους, ολοκληρωμένα σενάρια, αλληγορίες με παιδαγωγική σημασία και κυρίως επιφέρει συγκεκριμένα διδακτικά και μαθησιακά αποτελέσματα. Το εκπαιδευτικό λογισμικό από τεχνική άποψη εξετάζεται ως προς την ποιότητα του περιβάλλοντος διεπαφής, την εργονομία, το είδος της αλληλεπίδρασης που επιτρέπει με τον χρήστη, τα χρησιμοποιούμενα μέσα (εικόνα, ήχος κ.λ.π.) και την αισθητική του. Συνήθως ως εκπαιδευτικό λογισμικό θεωρούνται και τα πακέτα εφαρμογών επιμορφωτικού, εγκυκλοπαιδικού και ψυχαγωγικού τύπου.

1.2 Κατηγορίες εκπαιδευτικών λογισμικών

1. Κατηγοριοποίηση με βάση τις υποκείμενες θεωρίες μάθησης και τις συνεπαγόμενες διδακτικές πρακτικές:

- Λογισμικά καθοδηγούμενης διδασκαλίας, tutorials πρακτικής και εκγύμνασης. Στηρίζονται κυρίως σε συμπεριφορικές και γνωστικές θεωρίες μάθησης.
- Λογισμικά καθοδηγούμενης ανακάλυψης και διερεύνησης. Στηρίζονται κυρίες σε γνωστικές και επικοδομιστικές θεωρίες μάθησης.

- Λογισμικά έκφρασης, επικοινωνίας, συνεργασίας, δημιουργίας. Στηρίζονται σε επικοδομικστικές και κοινωνικοπολιτισμικές θεωρείες μάθησης.

2. Κατηγοριοποίηση με βάση τις τεχνολογίες ανάπτυξης και τα παιδαγωγικά ρεύματα:

- Λογισμικά στα οποία το σύστημα λειτουργεί ως «δάσκαλος»
- Λογισμικά στα οποία το σύστημα λειτουργεί ως «μαθητής»
- Λογισμικά στα οποία το σύστημα λειτουργεί ως «συνεργάτης» του μαθητή ή ως εργαλείο μάθησης

3. Κατηγοριοποίηση με βάση τη χρήση του λογισμικού στη μαθησιακή διαδικασία:

- Λογισμικό εξάσκησης – εφαρμογών (drill & practice). Αποβλέπει στην αναδραστική απάντηση του μαθητή, χρησιμοποιείται για εξάσκηση στη διδαχθείσα ύλη και δίνει στον εκπαιδευτικό τη δυνατότητα να το χρησιμοποιήσει και για την αξιολόγηση των μαθητών.
- Λογισμικό παρουσίασης διαλέξεων (tutorial). Παρέχει τη δυνατότητα παρουσίασης κυρίως στοιχείων θεωρίας με χρήση πολυμέσων και γίνεται το μάθημα ελκυστικότερο.
- Λογισμικό προσομοίωσης (simulation). Είναι ένα σύνολο λειτουργιών / κανόνων για τη λειτουργία ενός συστήματος, την εξήγηση ενός φαινομένου ή την επίλυση ενός προβλήματος. Επιτρέπει την εικονική αναπαράσταση και μοντελοποίηση ενός φαινομένου ή ενός πραγματικού συστήματος υπό συνθήκες που προσεγγίζουν τις πραγματικές. Επιτρέπει στον χρήστη να μεταβάλει τις συνθήκες στο εικονικό πείραμα, βοηθώντας έτσι την κατανόησή του. Ενσωματώνει συνήθως ένα μαθηματικό μοντέλο, γ΄ αυτό και αναφέρεται συχνά ως μοντέλο προσομοίωσης.

- Λογισμικό επίλυσης προβλημάτων (problem solving) και αυτοξιολόγησης. Ο μαθητής μπορεί να αναπτύξει διάφορες στρατηγικές επίλυσης, να αυτοαξιολογείται και αναδεικνύονται έτσι τυχόν παρανοήσεις ή αδυναμίες του.
- Διδακτικά παιχνίδια (games). Είναι ένα δελεαστικό περιβάλλον και κίνητρο για την επίτευξη ορισμένων διδακτικών και μαθησιακών στόχων.

4. Κατηγοριοποίηση με βάση το βαθμό αλληλεπίδρασης του λογισμικού:

- Ανοιχτού περιβάλλοντος. Το περιβάλλον αποτελείται από ένα σύνολο πρωταρχικών αντικειμένων που επιδρούν σ΄ αυτό και ένα σύνολο κανόνων που διέπουν την εν λόγω επίδραση.
- Κλειστού περιβάλλοντος. Υποστηρίζει το παραδοσιακό μοντέλο διδασκαλίας.

5. Κατηγοριοποίηση με βάση την παιδαγωγική προσέγγιση

- Διερευνητικού χαρακτήρα. Εισάγουν και υποστηρίζουν νέους τρόπους διδασκαλίας βοηθώντας τους μαθητές να αποκτήσουν νέες δεξιότητες, να οξύνουν την αντίληψή τους και να αποκτούν κριτική σκέψη.
- Πρακτικής και εκγύμνασης.

1.1.1 Παραδείγματα εκπαιδευτικών λογισμικών

Πολλά εκπαιδευτικά λογισμικά βρίσκονται στην ιστοσελίδα του Υπουργείου Παιδείας, στην οποία περιέχονται όλων των κατηγοριών εφαρμογές που βοηθούν την εκπαιδευτική διαδικασία από το νηπιαγωγείο μέχρι το πανεπιστήμιο:

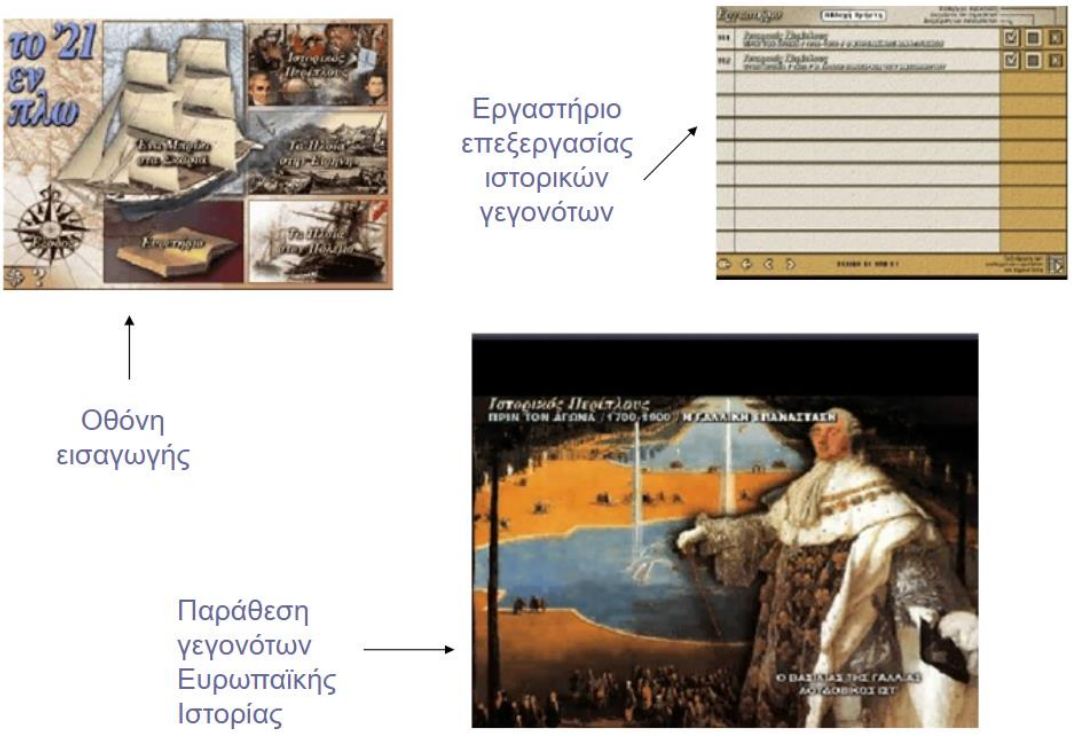
- http://edu-gate.minedu.gov.gr/index.php?option=com_k2&view=itemlist&task=category&id=99:thetikes-epistimes

- http://edu-gate.minedu.gov.gr/index.php?option=com_k2&view=itemlist&task=category&id=81:elevthero-logismiko-logismiko-anoixtoy-kodika

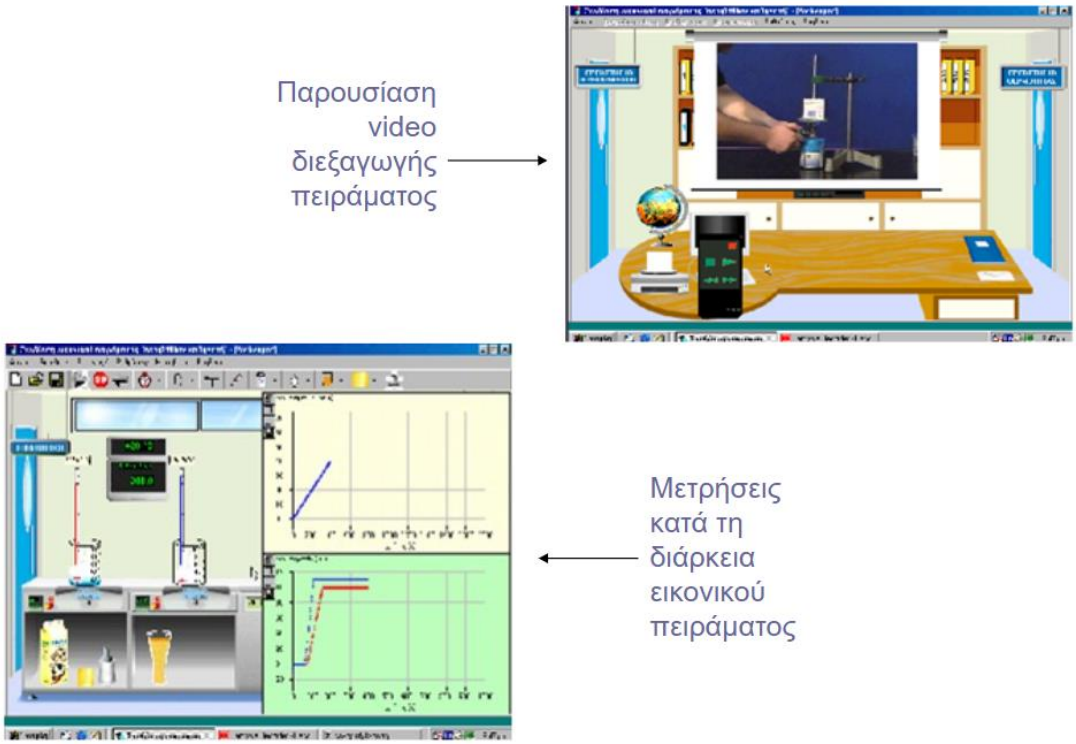
Ενδεικτικά αναφέρονται:

ΗΡΟΔΟΤΟΣ: Εφαρμογή για το μάθημα των Αρχαίων Ελληνικών, της Ιστορίας της Αρχαίας Ελλάδας και των Νέων Ελληνικών

ΓΑΙΑ: Λογισμικό για τη διδασκαλία Γεωγραφίας, Φυσικής και Μαθηματικών στο Γυμνάσιο.



21 ΕΝ ΠΛΩ: Παρουσίαση με χρονολογική σειρά των ιστορικών γεγονότων του 1821 μέσω ιστορικών γεωγραφικών χαρτών.



Σ.Ε.Π. (Σύνθετο Εργαστηριακό Περιβάλλον): Ανάπτυξη εφαρμογών πειραματικής διδασκαλάις.

ΞΕΝΙΟΣ: Υπολογιστικό και δικτυακό περιβάλλον για τη διδασκαλία ξένων γλωσσών.

Τα 3D παιχνίδια και οι προσομοιώσεις πειραμάτων, ειδικότερα στην εκπαιδευτική διαδικασία κερδίζουν καθημερινά έδαφος. Χρησιμοποιούνται πλέον, όχι μόνο στα σχολεία, αλλά και από τα Πανεπιστήμια σε δύσκολα γνωστικά αντικείμενα, όπως τη Βιολογία, την Ιατρική και τους Πολιτικούς Μηχανικούς.

Χαρακτηριστικό παράδειγμα εφαρμογής εικονικού πειράματος είναι το Onlabs του Ελληνικού Ανοιχτού Πανεπιστημίου, το οποίο προσομοιώνει το εργαστήριο βιολογίας και χρησιμοποιείται για την εκπαίδευση των φοιτητών βιολογίας στην κατάλληλη χρήση του εργαστηριακού εξοπλισμού και τη διεξαγωγή πειραμάτων. Το συγκεκριμένο έργο αποτέλεσε έμπνευση για την παρούσα πτυχιακή εργασία. Αναλυτικές πληροφορίες μπορεί να βρει ο αναγνώστης στην ιστοσελίδα https://sites.google.com/site/onlabseap/.

1.3 Ωφέλειες προσομοίωσης – 3D παιχνιδιού

Το εικονικό περιβάλλον «κεντρίζει» το ενδιαφέρον μαθητών/σπουδαστών με αποτέλεσμα την καλύτερη κατανόηση του μαθήματος. Με τις προσομοιώσεις τα μαθήματα γίνονται πιο οικεία λόγω του φιλικού περιβάλλοντος προς τον χρήστη, και παρακινούν τους μαθητές/σπουδαστές για περαιτέρω έρευνα και μελέτη. Δημιουργούνται επίσης και οικονομικά οφέλη, διότι αρκετά εργαστηριακά πειράματα απαιτούν ακριβά υλικά και εξοπλισμό, τα οποία με τη πάροδο του χρόνου όταν καταστρέφονται δεν μπορούν εύκολα να αντικατασταθούν ή έχουν υψηλό κόστος.

2. ΕΡΓΑΛΕΙΑ

2.1 Επιλογή εργαλείου, Γιατί Unity

Για τη παρούσα πτυχιακή εργασία στην οποία υλοποιείται 3D παιχνίδι εργαστηριακού πειράματος στο πεδίο του Πολιτικού Μηχανικού, χρησιμοποιήθηκε η πλατφόρμα υλοποίησης παιχνιδιών της Unity[1]. Γιατί όμως Unity; Η έρευνα για την αναζήτηση της κατάλληλης πλατφόρμας επικεντρώθηκε στα εξής σημεία:

να είναι δωρεάν,

να είναι εύκολο στη χρήση,

το εκτελέσιμο αρχείο να τρέχει σε όσο τον δυνατόν περισσότερα λογισμικά υπολογιστών,

να τρέχει σε λογισμικά κινητών τηλεφώνων

Συγκεντρώθηκαν όλες οι προδιαγραφές επιλογής κατάλληλης πλατφόρμας, και συγκρίνοντας μεταξύ των έξι (6) καλύτερων 3D game engines για αρχάριους (βλ. Πίνακα 1), τελικά επιλέχθηκε η Unity.

[1] Ο αναγνώστης μπορεί να βρει όλες τις σχετικές πληροφορίες στην ιστοσελίδα: https://unity3d.com

Πίνακας 1: Στοιχεία για Πλατφόρμες Παιχνιδιών:

Πλατφόρμες	Τιμή	Desktop targets	Mobile targets	Dev platforms	Console targets	Languages	TV targets
Unity	Free or 30$/month	Windows, OSX, Linux	Windows, iOS, Android, BlackBerry10, Tizen	Windows, OSX,	Xob360, XboxOne, Wii U, Playstation3,4,Vita, Nintendo Switch	C#, UnityScript, Boo	Android, Apple, SamsungSmart
Unreal Engine	-	Windows, Mac OSX, Linux, SteamOs, HTML5	iOS, Android	Windows, MacOSX, Linux	-	C++, Blueprints	
Wave	-	Windows, OSX, Linux	iOS, Android	Windows, OSX, Linux	-	C++, VB, F#	
Xenko	-	-	iOS, Android, UWP	-	-	-	
Atomic Game Engine	Free	Windows, Linux	-	-	-	-	
GameGuru	$14.99	Windows	None	Windows	none	-	

2.2 Δυνατότητες της Unity

Τα τελευταία χρόνια που έχουν εισέλθει στη ζωή μας οι τρισδιάστατες εικόνες, τα παιχνίδια λόγου χάρη, με 3D περιβάλλον και με τα αντίστοιχα 3D αντικείμενα που αλληλοεπιδρούν με αυτό, είναι σαφώς πιο ελκυστικά για τον καταναλωτή. Η εταιρεία Unity Technologies όταν δημιούργησε την πλατφόρμα Unity, εξασφάλισε στους προγραμματιστές διαφορετικούς τρόπους προσέγγισης ελέγχου των αντικειμένων, έχοντας δημιουργήσει την δυνατότητα παραμετροποίησης των αντικειμένων, την ευελιξία στη συγγραφή κώδικα και γενικά έδωσε ελευθερία κινήσεων στον προγραμματιστή. Έτσι, ανάλογα με τις δεξιότητες που κατέχει ο καθένας, δημιουργεί διαφόρων επιπέδων δυσκολίας παιχνίδια. Μας δίνει την επιπλέον δυνατότητα να χρησιμοποιήσουμε άλλα προγράμματα δημιουργίας 3D αντικειμένων όπως π.χ Blender[2], SketchUp[3] και να τα εισάγουμε προς παραμετροποίηση.

2.3 Γενική περιγραφή της πλατφόρμας Unity

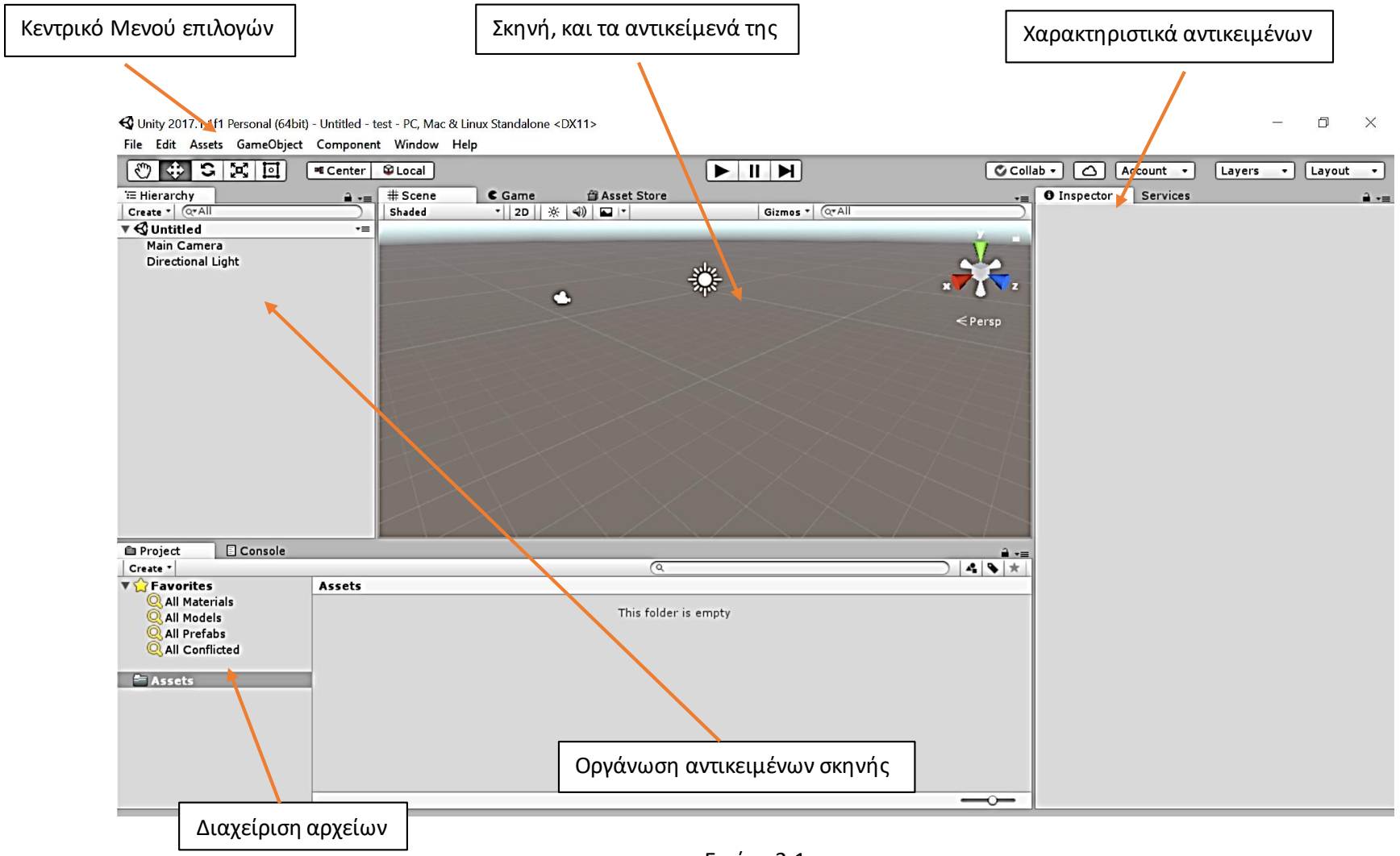
Ξεκινώντας τη μοντελοποίηση, θα πρέπει να ορίσουμε τα αντικείμενά μας σε μια σκηνή. Τα αντικείμενα που συνθέτουν μια σκηνή, για να μην είναι στατικά θα πρέπει να ενσωματώσει ο χρήστης ιδιότητες ώστε να διαμορφωθούν οι συμπεριφορές τους, να αποκρίνονται σε διάφορες ενέργειες και να αλληλοεπιδρούν με άλλα αντικείμενα και το περιβάλλον τους. Οι ιδιότητες αυτές, μπορεί να είναι οι μορφοποιημένες ήδη από την πλατφόρμα ή μπορούν να εμπλουτιστούν και να παραμετροποιηθούν μέσω κώδικα από τον χρήστη. Οι γλώσσες προγραμματισμού που μπορούν να χρησιμοποιηθούν είναι C#, JavaScript και Boo.

2.3.1 Περιβάλλον Unity

Το πρωτεύον περιβάλλον της εφαρμογής φαίνεται στην Εικόνα2.1 :

[2] Σχετικές πληροφορίες στην ιστοσελίδα: https://www.blender.org/

[3] Σχετικές πληροφορίες στην ιστοσελίδα: https://www.sketchup.com/


Κεντρικό Μενού επιλογών
Σκηνή, και τα αντικείμενά της
Χαρακτηριστικά αντικειμένων
File Edit Assets GameObject Component Window Help
Center Local
Collab Account Layers Layout
Hierarchy
Create
Untitled
Main Camera
Directional Light
Scene Game Asset Store
Shaded 2D Gizmos
Persp
Inspector Services
Project Console
Create
Favorites
All Materials
All Models
All Prefabs
All Conflicted
Assets
Assets
This folder is empty
Οργάνωση αντικειμένων σκηνής
Διαχείριση αρχείων

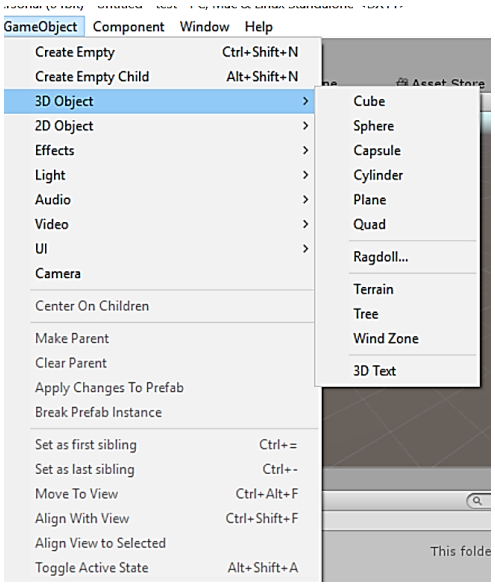
Εικόνα 2.1

2.3.2 Σκηνές Δράσης

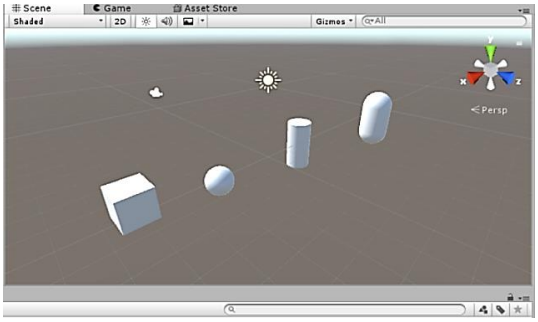
Η σκηνή, απαρτίζεται από τα αντικείμενα που έχουν εισαχθεί. Η κάθε σκηνή μπορεί να έχει αντικείμενα με διαφορετική ιεραρχική δομή, αλλά και μεταξύ τους οι σκηνές να υπόκεινται σε ιεραρχία. Με τον τρόπο αυτό, δημιουργούνται τα επίπεδα του παιχνιδιού και υπάρχει αλληλουχία στις αλληλεπιδράσεις.

2.3.3 Αντικείμενα

Ως αντικείμενα ορίζονται τα GameObject. Η εισαγωγή τους γίνεται από το κεντρικό μενού (Εικόνα 2.2 και Εικόνα 2.3), μπορούν να οργανωθούν ιεραρχικά ανάλογα με τις ανάγκες, να έχουν ιδιότητες και χαρακτηριστικά, ώστε να μετατραπούν από στατικά στοιχεία σε διαδραστικά, που αλληλοεπιδρούν με άλλα αντικείμενα και το περιβάλλον.



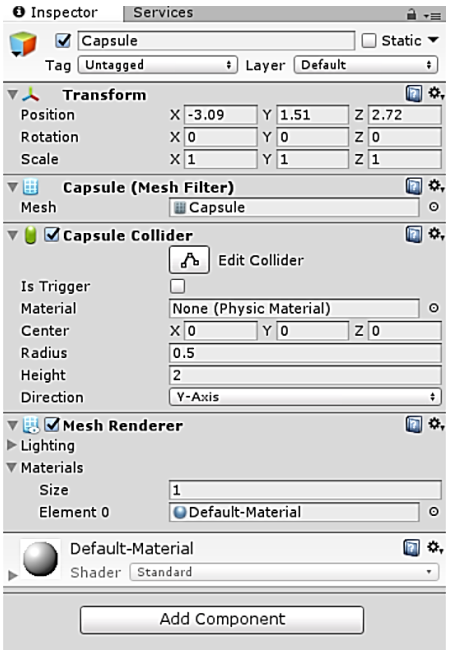
Εικόνα 2.2



Εικόνα 2.3

2.3.4 Ιδιότητες αντικειμένων

Το κάθε αντικείμενο πριν τη χρήση ιδιοτήτων και χαρακτηριστικών, είναι μια γραφική παράσταση ορισμένο στους άξονες x,y,z και βρίσκεται σε ορισμένο σημείο. Ενσωματώνοντας ιδιότητες στο αντικείμενο διαμορφώνουμε διαφορετικές συμπεριφορές. Με την εισαγωγή των Components (Εικόνα 2.4) δίνουμε την δυνατότητα στα αντικείμενα να έχουν συμπεριφορές και χαρακτηριστικά για τη προσομοίωση που επιθυμούμε. Η σύνδεση των διαφορετικών χαρακτηριστικών που μπορεί να έχουν τα αντικείμενα επιτυγχάνεται με τη χρήση έτοιμων από το πρόγραμμα μεθόδων, και μπορούν να επεκταθούν με παραμετροποίηση μέσω κώδικα.



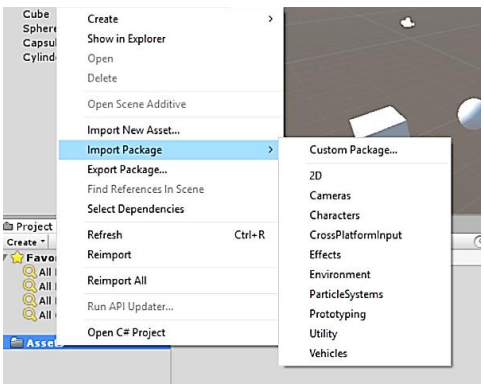
Εικόνα 2.4

2.3.5 Κλάση Monobehaviour

Η κλάση αυτή που είναι ενσωματωμένη στη Unity, μας δίνει την δυνατότητα για τον προγραμματισμό και την παραμετροποίηση των αντικειμένων ώστε να μετατραπούν σε διαδραστικά στοιχεία.

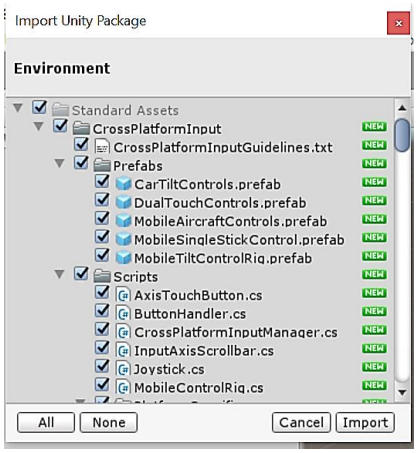
2.3.6 Assets

Η πλατφόρμα Unity δίνει την δυνατότητα να διαμορφώσει ο εκάστοτε προγραμματιστής τις υφές και την εμφάνιση που επιθυμεί να έχουν τα αντικείμενα. Τα έτοιμα πακέτα που υπάρχουν, μπορούν εύκολα να εισαχθούν στην εφαρμογή (π.χ. το πακέτο Environment Εικόνα 2.5 και Εικόνα 2.6), όπως επίσης και να εισαχθούν πακέτα, από το Asset Store[4] που διατίθενται δωρεάν ή με πληρωμή από την εταιρία. Το πιο εκπληκτικό είναι, ότι ο εκάστοτε προγραμματιστής, μπορεί να δημιουργήσει τις δικές του υφές στην εμφάνιση των αντικειμένων. Λόγου χάρη, μπορεί να δώσει την υφή «τοίχου» σε κάποιο material το οποίο θα εφαρμοστεί στο αντικείμενο που θέλει. (Εικόνα 2.7 και Εικόνα 2.8).

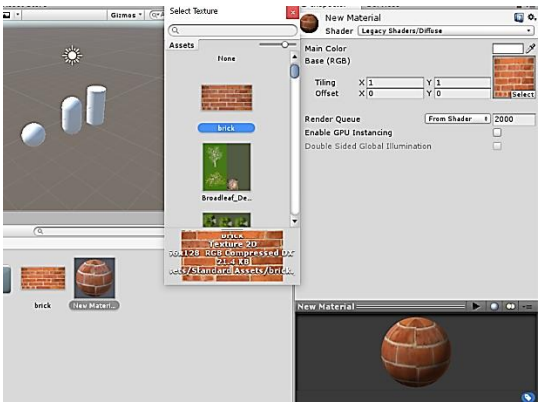


Εικόνα 2.5

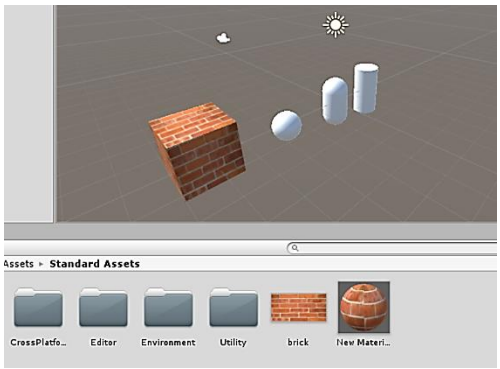
[4] Σχετικές πληροφορίες στην ιστοσελίδα: https://www.assetstore.unity3d.com/en/



Εικόνα 2.6



Εικόνα 2.7



Εικόνα 2.8

3. ΘΕΩΡΙΑ και ΠΡΟΔΙΑΓΡΑΦΕΣ του ΠΕΙΡΑΜΑΤΟΣ

Το αντικείμενο της εκπαίδευσης του Πολιτικού Μηχανικού είναι η παροχή γνώσεων στους σπουδαστές σε θέματα που αφορούν το σχεδιασμό, την κατασκευή και τη διοίκηση των έργων Πολιτικού Μηχανικού. Σκοπός είναι να προετοιμάσει μηχανικούς που θα ασχοληθούν με έργα Πολιτικού Μηχανικού όπως: δομικά έργα, έργα οδοποιίας, συστήματα κυκλοφορίας & μεταφορών, γέφυρες, σήραγγες, εγκαταστάσεις καθαρισμού λυμάτων, υδρεύσεις, αρδεύσεις, στραγγίσεις, αποχετεύσεις, λιμενικά έργα, αντιπλημμυρικά έργα, κλπ. Το πρόγραμμα σπουδών του Τμήματος Πολιτικών Μηχανικών του πρώην ΤΕΙ Αθήνας, παρέχει 27 εργαστήρια στο σύνολο των 40 μαθημάτων που διδάσκονται οι σπουδαστές. Κατά τη διάρκεια των εργαστηριακών μαθημάτων, πραγματοποιούνται ποικίλα πειράματα, ώστε να εμπλουτίσουν και να συμπληρώσουν οι σπουδαστές τις θεωρητικές τους γνώσεις.

Έχοντας στη διάθεσή μας όλη αυτή τη γκάμα επιλογών για τα προσφερόμενα εργαστήρια, ξεκίνησε η έρευνα στην εύρεση του κατάλληλου πειράματος, ώστε να υλοποιηθεί η προσομοίωσή του. Η έρευνα κατέληξε σε μια τελική λίστα εννιά (9) υποψηφίων πειραμάτων. Για να καταλήξουμε στο τελικό, ερευνήθηκαν οι δυνατότητες της Unity ως προς τη παραμετροποίηση των αντικειμένων που απαιτεί το εκάστοτε πείραμα. Πειράματα που περιείχαν ως βασικό στοιχείο το νερό, απορρίφθηκαν διότι δεν υπήρχε ή ήταν πολύ δύσκολο σε επίπεδο πτυχιακής, η παραμετροποίησή του. Επίσης απορρίφθηκαν για τους ίδιους λόγους, πειράματα τα οποία θα έπρεπε να δημιουργηθούν πολλά και δύσκολα animation ή πειράματα στα οποία θα έπρεπε ο χρήστης κατά την υλοποίησή τους, να εισάγει διάφορα δεδομένα.

3.1 Αναλυτικό θεωρητικό υπόβαθρο πειράματος

Η απλούστερη και συνηθέστερη μέθοδος διαχωρισμού δειγμάτων κατά μέγεθος είναι η εκτέλεση κοκκομετρικής ανάλυσης με τη χρήση κοσκίνων δοκιμών. Μία πλήρης σειρά πρότυπων κοσκίνων διατάσσεται κατά μέγεθος ανοιγμάτων σε στήλη με το μικρότερο άνοιγμα στη βάση και το μεγαλύτερο στην κορυφή. Η ανάλυση διεξάγεται με τοποθέτηση του δείγματος στο κόσκινο της κορυφής και οριζόντια ανακίνηση της στήλης μηχανικά για ορισμένο χρόνο. Το συγκρατούμενο σε κάθε κόσκινο κοκκώδες υλικό απομακρύνεται και συγκρατούμενες μάζες σε κάθε κόσκινο μετατρέπονται σε κλάσματα μάζας ή ποσοστά μάζας επί τοις εκατό επί του ολικού δείγματος. Το κοκκώδες υλικό που διέρχεται λεπτότερου κόσκινου συλλέγεται σε δίσκο στη βάση της στήλης.

Τα αποτελέσματα μίας κοκκομετρικής ανάλυσης διατάσσονται σε πίνακες ώστε να δείχνουν το κλάσμα της συγκρατούμενης μάζας σε κάθε κόσκινο συναρτήσει του μεγέθους της οπής του κοσκίνου. Εφόσον οι συγκρατούμενοι κόκκοι του υλικού οποιουδήποτε κοσκίνου διέρχονται δια του αμέσως υπεράνω αυτού κοσκίνου, χρειάζονται δύο αριθμοί για να καθορισθεί η τάξη μεγέθους της συγκρατούμενης μάζας σε αυτό. Ένας για το κόσκινο μέσα από το οποίο πέρασε κλάσμα της μάζας και ένας για το κόσκινο που συγκράτησε κλάσμα της μάζας. Έτσι έκφραση 14/20 σημαίνει «διαμέσου ανοίγματος 14 και επί ανοίγματος 20». Η ανάλυση που διατάσσεται κατά τον τρόπο αυτό σε πίνακα καλείται διαφορική ανάλυση.

Ο δεύτερος και πιο συνηθισμένος τύπος κοκκομετρικής ανάλυσης είναι η αθροιστική ανάλυση. Η αθροιστική ανάλυση παράγεται από τη διαφορική ανάλυση με πρόσθεση των επιμέρους διαφορικών συγκρατούμενων μαζών, αρχής γενομένης από τη συγκρατούμενη μάζα στο κόσκινο με το μεγαλύτερο άνοιγμα. Τα ποσοστό αυτά παρουσιάζονται σε ημιλογαριθμικό διάγραμμα, στο οποίο ο οριζόντιος σε λογαριθμική κλίματα άξονας, αντιστοιχεί στο άνοιγμα της οπής, ενώ ο κατακόρυφος στο συνολικά συγκρατούμενο ή / και στο συμπληρωματικό του συνολικά διερχόμενο.

Για κοκκομετρική ανάλυση διατίθενται 3 σειρές από κόσκινα:

- Σειρά ελληνικών κοσκίνων. Τα ελληνικά κόσκινα είναι κυκλικής οπής με διαμέτρους 1,3,5,7,10,15,30,50,70 mm και ένα κόσκινο με τετραγωνική οπή και άνοιγμα 0,2 mm. Τα ελληνικά κόσκινα συμβολίζονται με το σύμβολο Φ που γράφεται πριν τον αριθμό του κοσκίνου.
- Η σειρά των γερμανικών κοσκίνων ή γερμανικά κόσκινα όπως αυτά περιγράφονται στα DIN 4187 και 4188. Φέρουν οπές τετραγωνικής μορφής και τα πιο χρησιμοποιούμενα είναι τα εξής: - πλέγματος: 0.25 , 0.50 , 1 , 2 mm - ελάσματος: 4 , 8 , 16 , 31.5 , 63 mm.
- Η σειρά των αμερικανικών κοσκίνων ή αμερικάνικα κόσκινα, όπως αυτά περιγράφονται στο πρότυπο ASTM E11. Τα κόσκινα αυτά φέρουν οπές τετραγωνικής μορφής από πλέγμα και συμβολίζονται είτε με τον αριθμό των οπών που φέρουν ανά γραμμική ίντσα για τα πιο λεπτά (μέχρι το No 4), είτε με βάση το άνοιγμα της οπής σε ίντσες για τα μεγαλυτέρου ανοίγματος. Έτσι το κόσκινο 3/4 in ή 3/4" δηλώνει αμερικάνικο κόσκινο οπής 3/4 της ίντσας (19 mm περίπου). Αντίστοιχα η ονομασία No 12 δηλώνει αμερικάνικο κόσκινο που φέρει 12 οπές ανά γραμμική ίντσα. (1 in=2,54 cm).

Οι ελληνικοί κανονισμοί δίδουν σε πίνακες τα συνιστώμενα κατώτερα και ανώτερα όρια του ποσοστού των αδρανών που διέρχονται από κάθε κόσκινο, καθώς και τα διαγράμματα κοκκομετρικής σύνθεσης.

3.2 Σκοπός του πειράματος

Σκοπός του πειράματος είναι η εύρεση της κοκκομετρικής καμπύλης ενός δείγματος αμμοχάλικου, δηλαδή, τη διαβάθμιση της κοκκομετρικής αναλογίας των διαφορετικών κλασμάτων του αδρανούς υλικού. Τα σκεύη και τα υλικά που χρησιμοποιούνται είναι:

20 κιλά (kg) αμμοχάλικο

Σειρά Γερμανικά Κόσκινα

Σείστρο

Ηλεκτρονική Ζυγαριά Ακριβείας

Ηλεκτρικός Φούρνος 105°C
(Κλίβανος Ξήρανσης)

3.3 Μεθοδολογία πειράματος

α. Παίρνουμε με τετραμερισμό αντιπροσωπευτικό δείγμα αμμοχάλικου, περίπου (3 κιλά), [κατά περίπτωση, ανάλογα με την κοκκομετρία του].

β. Ζυγίζουμε την ποσότητα που πήραμε από το αντιπροσωπευτικό δείγμα.

γ. Ξηραίνουμε την παραπάνω ποσότητα στον φούρνο σε θερμοκρασία 105°C – 110°C , μέχρι σταθερού βάρους. [για τουλάχιστον 24 ώρες]

δ. Τοποθετούμε τα κόσκινα στο σείστρο με την σειρά, ανάλογα με το άνοιγμα της οπής του κόσκινου (το μικρότερο κάτω). Η σειρά από κάτω προς τα πάνω είναι: - 0,25 – 0,50 – 1,00 – 2,00 – 4,00 – 8,00 – 16,00 – 31,50 (Γερμανική Σειρά)

ε. Ρίχνουμε το εξεταζόμενο δείγμα αδρανούς στο κορυφαίο κόσκινο και το ασφαλίζουμε με το καπάκι του σείστρου. Ενεργοποιούμε την συσκευή για να κοσκινίσει το υλικό για 3 – 5 λεπτά

στ. Ζυγίζουμε το υλικό που συγκρατήθηκε σε κάθε κόσκινο και συμπληρώνουμε σταδιακά την στήλη (2).

ζ. Συμπληρώνουμε την στήλη (3) του πίνακα με το συνολικό συγκρατούμενο βάρος του υλικού σε κάθε κόσκινο. Δηλαδή προσθέτουμε το βάρος που συγκρατήθηκε σε κάθε κόσκινο με το βάρος όλων των πάνω από αυτό ποσοτήτων που συγκρατήθηκαν.

η. Προσθέτουμε και το βάρος που συγκρατήθηκε στον πυθμένα.

θ. Αφού συμπληρώσουμε και τις δύο στήλες (2) και (3) του πίνακα, ελέγχουμε αν έχει προκύψει απώλεια κατά την διαδικασία της δοκιμής. Η επιτρεπόμενη απώλεια του εξεταζόμενου υλικού είναι ελάχιστη του (1% - 2%).

ι. Συμπληρώνουμε την στήλη (4). Δηλαδή το συγκρατούμενο ποσοστό κάθε κοσκίνου.

ια. Στην στήλη (5), στρογγυλοποίηση των αποτελεσμάτων της στήλης (4)

ιβ. Συμπληρώνουμε την στήλη (6) για το Διερχόμενο Βάρος (gr) για κάθε κόσκινο Την στήλη (7), συμπληρώνουμε τα διερχόμενα ποσοστά, αφαιρώντας από το 100 τα αποτελέσματα της στήλης (5). Δηλαδή: για το κόσκινο Φ8, έχουμε (100 – 61 = 39%).

3.4 Αποτελέσματα πειράματος

Για δοκιμή με 3Kg αμμοχάλικο ενδεικτικά μπορούμε να έχουμε τα κάτωθι αποτελέσματα:

ΠΙΝΑΚΑΣ ΕΓΓΡΑΦΗΣ ΑΠΟΤΕΛΕΣΜΑΤΩΝ ΚΟΚΚΟΜΕΤΡΙΚΗΣ ΑΝΑΛΥΣΗΣ

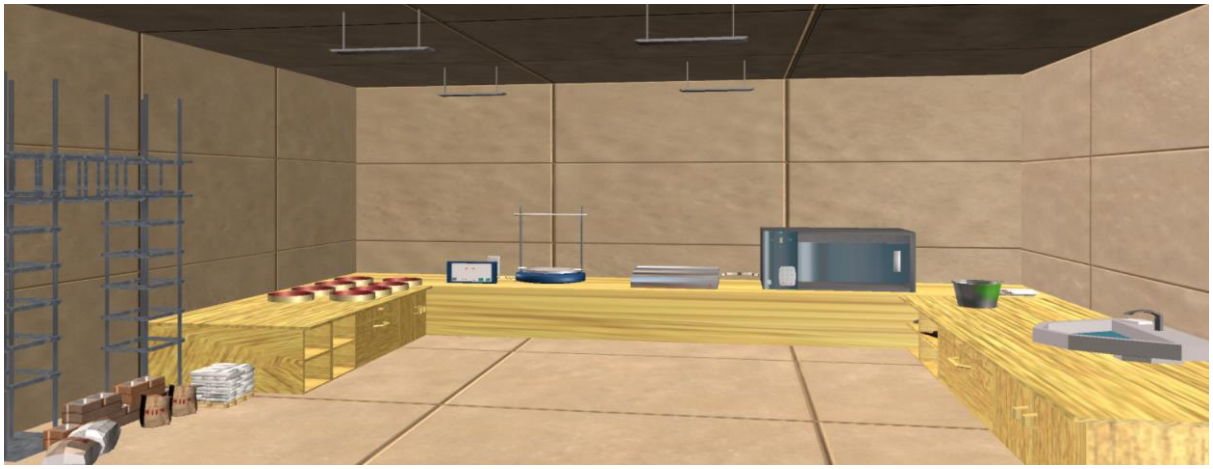
1	2	3	4	5	6	7
Κόσκινα (mm)	**Συγκρατούμενο Βάρος (gr)**	**Συνολικό Συγκρατούμενο (gr)**	**Ποσοστό Συγκρατούμενο (%)**	**Συγκρατούμενο (%) Στρογγυλοποίηση**	**Διερχόμενο Βάρος (gr)**	**Ποσοστό Διερχόμενο (%)**
31,5	0	0	0,00	0	2975,4	100
16	599	599	20,13	20	2376,4	80
8	1214,3	1813,3	60,94	61	1162,1	39
4	307	2120,3	71,26	71	855,1	29
2	275	2395,3	80,50	81	580,1	20
1	191,3	2586,6	86,93	87	388,8	13
0,5	148,5	2735,1	91,92	92	240,3	8
0,25	95,2	2830,3	95,12	95	145,1	5
Πυθμένα	145,1	2975,4	100,00	100	0	0
Σύνολο	2975,4					

4. Το 3D παιχνίδι του ΠΕΙΡΑΜΑΤΟΣ στη UNITY

4.1 Δημιουργία σκηνής και των αντικειμένων της (Εικόνα 4.1).

Θεωρήθηκε σκόπιμο το γραφικό περιβάλλον να προσεγγίζει σε αρκετό βαθμό το πραγματικό εργαστήριο στο οποίο οι σπουδαστές υλοποιούν τα πειράματά τους, και τα αντικείμενα να βρίσκονται στην ίδια περίπου χωρική διάταξη. Η σκηνή απαρτίζεται από τους τοίχους, το δάπεδο, τους εργαστηριακούς πάγκους, και όλα τα αντικείμενα που είναι απαραίτητα για το εργαστηριακό πείραμα. Για τη δημιουργία τους, χρησιμοποιήθηκαν μερικά από τα έτοιμα αντικείμενα που υπάρχουν στην εφαρμογή της Unity (Cube, Sphere, Capsule, Cylinder, Plane, Quad), διατάχθηκαν αρμονικά με την απαραίτητη ιεραρχία και ορίστηκαν στις κατάλληλες διαστάσεις. Τα κόσκινα (sieve) εισήχθησαν από την εφαρμογή SketchUp[5], η οποία παρέχει στους χρήστες της έτοιμα, δωρεάν 3D αντικείμενα[6]. Αυτά τα αντικείμενα μπορούν να γίνουν εισαγωγή και να επεξεργαστούν, είτε στην ίδια την εφαρμογή είτε στη Unity. Υπάρχουν τρείς εκδοχές. Η pro, free και for Schools, επομένως ο ενδιαφερόμενος επιλέγει σύμφωνα με του όρους της κάθε έκδοσης και σύμφωνα με τους λόγους για τους οποίους επιθυμεί να χρησιμοποιήσει την εφαρμογή.

[5] Σχετικές πληροφορίες στην ιστοσελίδα: https://www.sketchup.com/
[6] Σχετικές πληροφορίες στην ιστοσελίδα: https://3dwarehouse.sketchup.com/?hl=en



(Εικόνα 4.1)

Τα αντικείμενα δίσκος (tray), σχάρα (rack) εισήχθησαν[7] ως έτοιμα 3D αντικείμενα. Η ιστοσελίδα αυτή, δεν απαιτεί εγγραφή, διαθέτει πληθώρα από 3D αντικείμενα, τα οποία πολλά από αυτά είναι δωρεάν. Εναπόκειται λοιπόν στην επιλογή του πελάτη αν θα χρησιμοποιήσει τα δωρεάν ή τα επί πληρωμή αντικείμενα.

Και στις δύο περιπτώσεις, το μόνο που χρειάστηκε ήταν να κατέβει το αντίστοιχο αρχείο επιλογής του 3D αντικειμένου, να αποθηκευτεί, και έπειτα να εισαχθεί μέσα από τη Unity.

4.1.1 Μετατροπή αντικειμένων σε διαδραστικά στοιχεία

Πέραν των στατικών αντικειμένων τα οποία δεν έχουν ιδιότητες π.χ τοίχος, πάγκος κ.τ.λ., τα υπόλοιπα αντικείμενα που δημιουργήθηκαν, μετατράπηκαν σε διαδραστικά στοιχεία μέσω κώδικα. Η μετατροπή ενός στατικού αντικειμένου σε διαδραστικό, σημαίνει τη μετατροπή του σε ένα «ζωντανό» και ρεαλιστικό αντικείμενο. Πόσο όμως ρεαλιστικό και «ζωντανό» μπορεί να μετατραπεί ένα αντικείμενο; Την απόφαση αυτή που θα πάρει ο προγραμματιστής, θα διαμορφώσει και τον ρόλο του αντικειμένου στην αλληλεπίδρασή του είτε με άλλο αντικείμενο είτε με τον χρήστη. Ας δώσουμε ένα απλό παράδειγμα. Επιθυμούμε ο χρήστης να πατήσει έναν διακόπτη φωτός. Ο προγραμματιστής με κατάλληλο κώδικα, μπορεί να δώσει

[7] Σχετικές πληροφορίες στην ιστοσελίδα: https://www.turbosquid.com/

την ρεαλιστική απεικόνιση αλλαγής στη θέση του διακόπτη για το on/off. Ανάλογα με το αντικείμενο που έχουμε, και ποια θα είναι η χρησιμότητά του, μπορούμε να προβάλουμε μέσω κώδικα όσες πραγματικές λεπτομέρειες επιθυμούμε. Όσο παραπάνω χαρακτηριστικά ρεαλισμού αποδίδουμε σε κάποιο αντικείμενο, τόσο καλύτερη θα είναι και η προσομοίωσή του, σε σχέση με την πραγματικότητα.

4.2 Οι δράσεις μέσω της Κλάσης Monobehaviour

Για τη μετατροπή των αντικειμένων σε διαδραστικά στοιχεία χρησιμοποιείται η κλάση Monobehaviour. Προσφέρει ένα σύνολο συναρτήσεων και ρουτινών ικανών να καταστήσουν την ανάπτυξη προγραμμάτων ευκολότερη. Επειδή υπάρχει η δυνατότητα επέκτασής τους με κατάλληλο κώδικα, μας δίδεται και η δύναμη να επιτύχουμε τις συμπεριφορές που επιθυμούμε για τα αντικείμενα, με όσες ρεαλιστικές λεπτομέρειες επιθυμούμε. Μέρος του κώδικα ο οποίος καθορίζει κοινές δράσεις αντικειμένων μας, αναλύεται στην επόμενη παράγραφο.

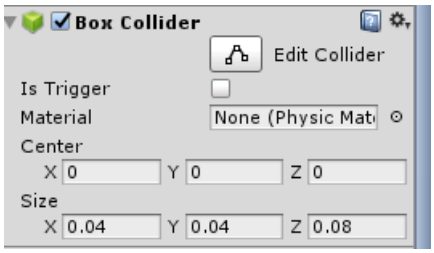
4.2.1 Γενικές αλληλοεπιδράσεις χρήστη και αντικειμένων

Για λόγους φιλικότητας ως προς τον χρήστη, δημιουργήθηκε μέσω κώδικα η δυνατότητα όταν κάποιο αντικείμενο που αλληλοεπιδρά με τον χρήστη να αλλάζει χρώμα, ο δείκτης του ποντικιού να αλλάζει σχήμα ανάλογα με την δράση που έχει, το αντικείμενο να μπορεί περιστρέφεται ή να μετακινείται. Η κλάση (script) (κώδικας 1) περιέχει συναρτήσεις οι οποίες αφορούν αυτού του είδους τις γενικές ιδιότητες, και έχει ενσωματωθεί ως component σε όλα τα διαδραστικά αντικείμενα. Οι κυριότερες συναρτήσεις που χρησιμοποιήθηκαν είναι:

Αναλυτικότερα χρησιμοποιήθηκαν οι κάτωθι συναρτήσεις που διαθέτει η Unity:

- Συνάρτηση `void OnMouseEnter()`: Ενεργοποιείται όταν ο δείκτης του ποντικιού εισέρχεται στην περιοχή του συγκεκριμένου αντικειμένου. Για να υφίσταται αυτή η ενεργοποίηση είναι απαραίτητο στο διαδραστικό αντικείμενο να υπάρχει ένα Box

Collider (Εικόνα 4.2) με την βοήθεια του οποίου γίνεται η ανίχνευση του δείκτη του ποντικιού. Γενικά τα **Collider**, είναι αόρατα από τον χρήστη πλέγματα, τα οποία μπορούν να εφαρμοστούν σε όλα τα αντικείμενα, ώστε να ξεκινήσει η διάδραση μεταξύ αντικειμένων. Η Unity διαθέτει Collider σε σχήμα κύκλου, κύβου, κάψουλας κ.τ.λ. Ένα είδος είναι και το Box Collider. Επειδή το Collider είναι απαραίτητο για όλα τα διαδραστικά αντικείμενα, δεν θα αναφερθούμε ξανά για την ύπαρξή του σε κάποιο αντικείμενο.



(Εικόνα 4.2)

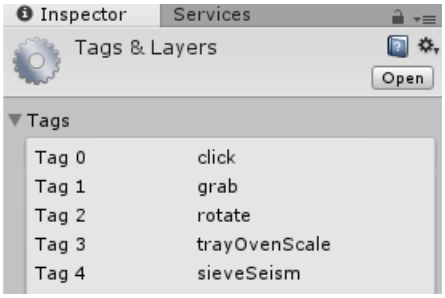
- Συνάρτηση `void OnMouseExit()`: Ενεργοποιείται όταν ο δείκτης του ποντικιού εξέρχεται από την περιοχή του αντικειμένου.

- Συνάρτηση `void OnMouseUpAsButton()`: Ενεργοποιείται όταν γίνει απλό αριστερό κλικ πάνω στο αντικείμενο.

- Συνάρτηση `void OnMouseDown()`: Ενεργοποιείται όταν είναι πατημένο το αριστερό κουμπί του ποντικιού.

- Συνάρτηση `void OnMouseUp()`:Ενεργοποιείται όταν απελευθερωθεί το αριστερό κουμπί του ποντικιού.

Στο κυρίως σώμα των συναρτήσεων δόθηκαν οι κατάλληλες εντολές για την πραγματοποίηση των αντίστοιχων αλλαγών. Επειδή τα διαδραστικά αυτά αντικείμενα μπορούν να συμπεριφέρονται διαφορετικά, η Unity μας δίνει την δυνατότητα να τους δώσουμε «ταμπέλες» (tags) (Εικόνα 4.3) ώστε να ξεχωρίζουν.



(Εικόνα 4.3)

Όπως βλέπουμε στη παραπάνω εικόνα, δημιουργήθηκαν 5 tags. Σε όλα δίνεται η δυνατότητα αλλαγής εμφάνισης του ποντικιού και αλλαγή του χρώματος στο αντικείμενο. Ειδικότερα:

Εάν έχουμε αντικείμενο τύπου button, έχει δηλωθεί το tag «**ckick**» με αλλαγή του κέρσορα όπως η διπλανή εικόνα. Ειδικότερα σε αυτή τη περίπτωση, μέσα στο σώμα της `OnMouseUpAsButton()` αναγράφεται η εντολή `gameObject.SendMessage("press");` η οποία με τη σειρά της καλεί την αντίστοιχη συνάρτηση `void press()` και εκτελεί ότι έχει στο δικό της σώμα. Αυτή όμως βρίσκεται στο script που αφορά το εκάστοτε αντικείμενο που μας ενδιαφέρει, και όχι στο γενικό script `mouseEvents`. Επίσης όταν ο χρήστης εισέλθει στο Collider του αντικειμένου, με τη βοήθεια της συνάρτησης `OnMouseEnter()` αλλάζει το χρώμα του, και στην έξοδο του χρήστη από το Collider του αντικειμένου, με τη συνάρτηση `OnMouseExit()` επανέρχεται το αρχικό του χρώμα.

Στη περίπτωση την οποία επιθυμούμε ο χρήστης να σηκώσει κάποιο αντικείμενο ώστε να το μεταφέρει σε άλλο σημείο της σκηνής, έχουμε ορίσει το tag «**grab**». Και σε αυτή τη περίπτωση τροποποιείται η εμφάνιση ποντικιού και το χρώμα του αντικειμένου.

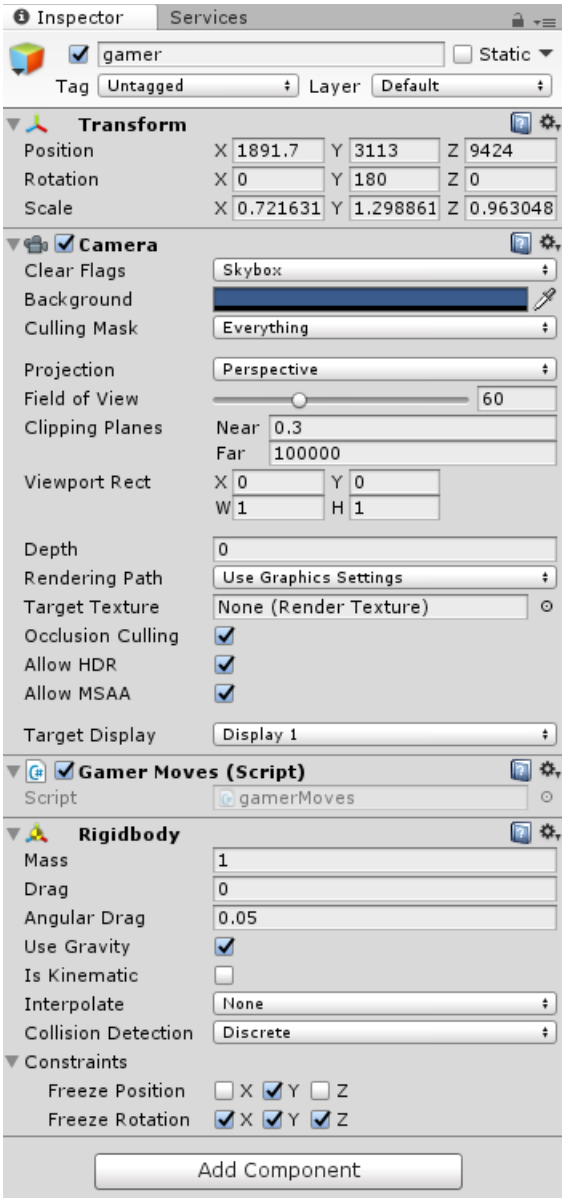
Όταν θέλουμε για κάποιο αντικείμενο να έχει την ιδιότητα της περιστροφής από τον χρήστη, δηλαδή είναι κάποιος κοχλίας, ορίζουμε το tag «**rotate**» με τις ίδιες δυνατές αλλαγές.

Τέλος, τα tags «**trayOvenScale**» και «**sieveSeism**» έχουν οριστεί για αντικείμενα, που ο χρήστης θα πρέπει να μεταφέρει και να τοποθετήσει πάνω σε άλλο αντικείμενο, και επομένως θα υπάρξει συνδυαστική χρήση των δύο αυτών αντικειμένων. Έχουν τη δυνατότητα αλλαγής χρώματος και αλλαγής εμφάνισης ποντικιού, αλλά λόγω της συνδυαστικής τους χρήσης, θα αναφερθούμε αναλυτικότερα σε αυτά στη συνέχεια.

4.2.2 Διαχείριση κίνησης και οπτικής γωνίας

Όπως έχουμε ήδη πει, η Unity είναι μια εφαρμογή από την οποία μπορούμε να δημιουργήσουμε παιχνίδια. Στα παιχνίδια ο παίχτης «βρίσκεται μέσα» στις σκηνές και μπορεί να κινείται προς όλες τις κατευθύνσεις. Η λογική αυτή εφαρμόστηκε και στη προσομοίωση του εργαστηριακού πειράματος. Πώς όμως ο χρήστης βρέθηκε «μέσα» στη σκηνή που είναι το εργαστήριο και πώς κινείται σε αυτή;

Εισήχθη ένα «άδειο» αντικείμενο, ονομάστηκε gamer, τοποθετήθηκε στη σκηνή και ενσωματώθηκαν σε αυτό components (Εικόνα 4.4). Τα components αυτά περιέχουν τις ιδιότητες που έχει ο χρήστης.



(Εικόνα 4.4)

- Το component Rigibody, αφορά στο νόμο της βαρύτητας της φύσης. Εξασφαλίζει, τη μη «πτώση» του gamer στον άξονα Y (με αποτέλεσμα να πέσει και να φύγει από τη σκηνή) και την πρόσκρουσή του με άλλα αντικείμενα. Έχοντας όλα τα αντικείμενα όπως έχουμε πει Collider, εξασφαλίζουμε ότι το αντικείμενο gamer, κινούμενο στον χώρο, δεν θα συμπέσει στη θέση άλλου αντικειμένου αλλά θα υπάρξει σύγκρουση και θα σταματήσει.

- Το component camera, είναι αυτό που μας δίνει την εικόνα του χώρου. Με χρήση κώδικα εμπλουτίστηκε με την ιδιότητας της κίνησης. Με αυτό τον τρόπο δημιουργείται η οπτική «απάτη» και η ψευδαίσθηση, ότι ο εκπαιδευόμενος βρίσκεται στον εργαστηριακό χώρο, μπορεί να κινηθεί ελεύθερα και να έχει 360 μοίρες οπτική επαφή γύρω του, όπως άλλωστε θα γινόταν και στην πραγματικότητα. Ο σχετικός κώδικας είναι στην κλάση `gamerMoves` (κώδικας 2). Ο χρήστης πατώντας ορισμένα πλήκτρα, να μπορεί να κινείται ως εξής:

Πλήκτρο	**Κίνηση**
A	δεξιά περιστροφή 360 μοίρες
D	αριστερή περιστροφή 360 μοίρες
W	προς τα εμπρός
S	προς τα πίσω
↑	προς τα επάνω (ψηλά)
↓	προς τα κάτω (χαμηλά)

Χρησιμοποιήθηκε η συνάρτηση `Input.GetKey()` η οποία παίρνει ως παράμετρο το πλήκτρο που έχει πατηθεί. Οι συναρτήσεις transform.Translate() και transform.Rotate() με παραμέτρους τις συντεταγμένες x, y, z μας βοηθάνε στον μετασχηματισμό των συντεταγμένων στις οποίες βρίσκεται το αντικείμενο. Στο παραπάνω παράδειγμα, αν πατηθεί το πάνω βέλος η μετακίνηση του αντικειμένου γίνεται προς τα ψηλά και ισούται με μεταβολή του καρτεσιανού επιπέδου y με ταχύτητα 150 ενώ αντίθετα αν πατηθεί το κάτω βέλος η μετακίνηση γίνεται προς τα χαμηλά αλλά με αρνητικό πρόσημο ταχύτητας. Το αρνητικό πρόσημο σηματοδοτεί την αντίθετη κίνηση.

4.2.3 Γενικά Χαρακτηριστικά Αντικειμένων

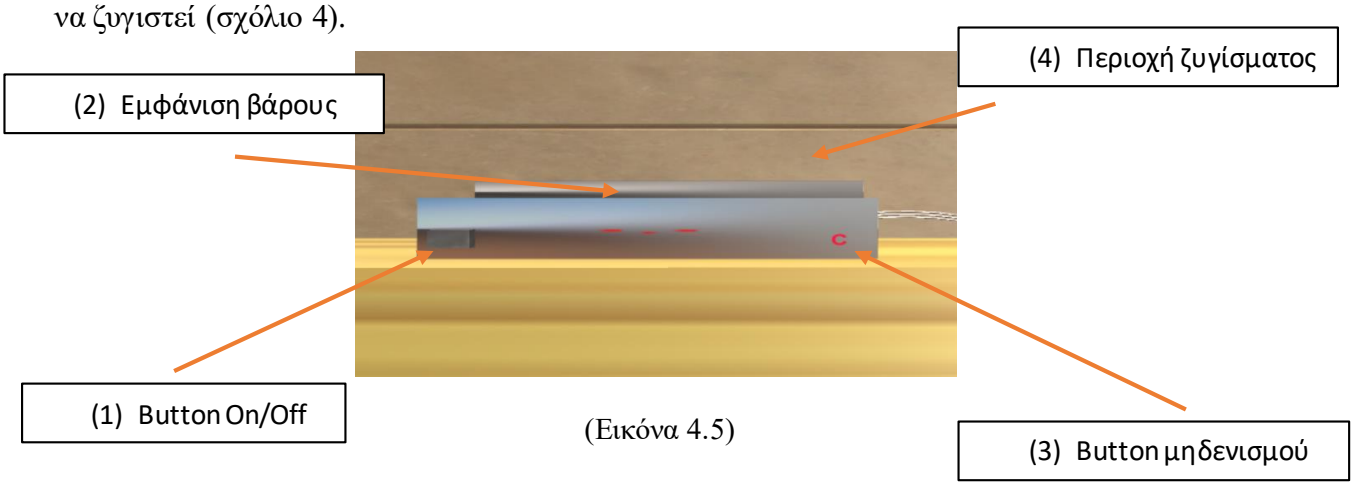
Όπως είναι γνωστό, μέσα στα πλαίσια της εκπαιδευτικής διαδικασίας, και ειδικά σε εργαστηριακά πειράματα, θα υπάρξουν λάθη από τους φοιτητές - εκπαιδευόμενους. Μέσα όμως από αυτά τα λάθη, θα μπορέσουν να κατανοήσουν καλύτερα την όλη διαδικασία, και θα είναι ικανοί να θυμηθούν και να αποφύγουν στο μέλλον τις λάθος ενέργειες.

Μετατρέποντας λοιπόν ορισμένα αντικείμενα σε διαδραστικά στοιχεία, θα πρέπει να αποφασίσουμε αν στη γενική υλοποίηση της προσομοίωσης, θα έχουμε ή όχι περιορισμούς στη δράση του κάθε αντικειμένου. Δηλαδή, αν θα είναι επιτρεπτό, ο χρήστης να μπορεί να αλληλοεπιδρά σε όλα τα αντικείμενα και με ποιόν τρόπο, και πώς θα αλληλοεπιδρούν μεταξύ τους τα αντικείμενα.

Λαμβάνοντας υπόψη την εκπαιδευτική ωφέλεια από τα λάθη, το γεγονός ότι το πείραμα προσομοιώνει ένα πραγματικό πείραμα, πάρθηκαν αποφάσεις σχετικά με την ευελιξία κινήσεων που θα έχει ο χρήστης, καθώς και τις επιτρεπτές ή όχι δράσεις μεταξύ των αντικειμένων. Αναλύοντας στην συνέχεια τον κώδικα για τις δράσεις του κάθε αντικειμένου, θα αναλυθούν και οι λόγοι περιορισμού ή όχι των δράσεων για το κάθε αντικείμενο.

4.2.4 Ζυγαριά (Εικόνα 4.5 με σχόλια)

Συμπεριφορές του αντικειμένου είναι το άνοιγμα και κλείσιμο της ζυγαριάς από τον χρήστη (σχόλιο 1), η εμφάνιση του βάρους (σχόλιο 2), η δυνατότητα μηδενισμού της ζυγαριάς (σχόλιο 3), και το σημαντικότερο, η «υποδοχή» ενός άλλου δεύτερου αντικειμένου το οποίο θα πρέπει να ζυγιστεί (σχόλιο 4).



(Εικόνα 4.5)

Όπως και στην πραγματικότητα, έτσι και εδώ στην προσσομοίωση, μπορεί ο χρήστης να τοποθετήσει πάνω στη ζυγαριά οποιοδήποτε διαδραστικό αντικειμένο επιθυμεί. Στη περίπτωση αυτή, δεν θα πραγματοποιηθεί καμία δράση από τη ζυγαριά, και ειδικότερα από τη περιοχή ζυγίσματος της ζυγαριάς (σχόλιο 4 εικόνας 4.5). Η περιοχή του ζυγίσματος θα ενεργοποιηθεί για το αντικείμενο το οποίο έχει tag «**trayOvenScale**», διότι με αυτή την ετικέτα έχουν οριστεί αντικείμενα με συνδυαστική χρήση τη ζυγαριά.

Μια άλλη δράση, αφορά το κουμπί ανοίγματος/κλεισίματος της ζυγαριάς (σχόλιο 1 εικόνας 4.5). Δεν κρίθηκε απαραίτητο να προηγείται το άνοιγμα της ζυγαριάς από τη τοποθέτηση του δίσκου, αλλά δίνεται η ευελιξία στον χρήστη, σε περίπτωση που ο δίσκος ακουμπήσει τη περιοχή ζυγίσματος (σχόλιο 4 εικόνας 4.5) και είναι κλειστή η ζυγαριά, να εμφανιστεί σχετικό μήνυμα υπενθύμισης για το άνοιγμά της.

Εφόσον πραγματοποιηθούν αυτές οι δύο δράσεις, ανεξαρτήτου σειράς, μόνο τότε ενεργοποιείται η δράση εμφάνισης του βάρους του δείγματος (σχόλιο 2 εικόνας 4.5).

Τέλος, η δράση μηδενισμού της ζυγαριάς (σχόλιο 3 εικόνας 4.5) ενεργοποιείται εφόσον έχει εμφανιστεί το βάρος. Αυτό, διότι σε καμία άλλη περίπτωση δεν έχει πραγματικό νόημα ο μηδενισμός.

Ο (κώδικας 3) ενσωματώνει αυτές τις προγραμματιστικές αποφάσεις, σχετικά με τις προαναφερθείσες δράσεις.

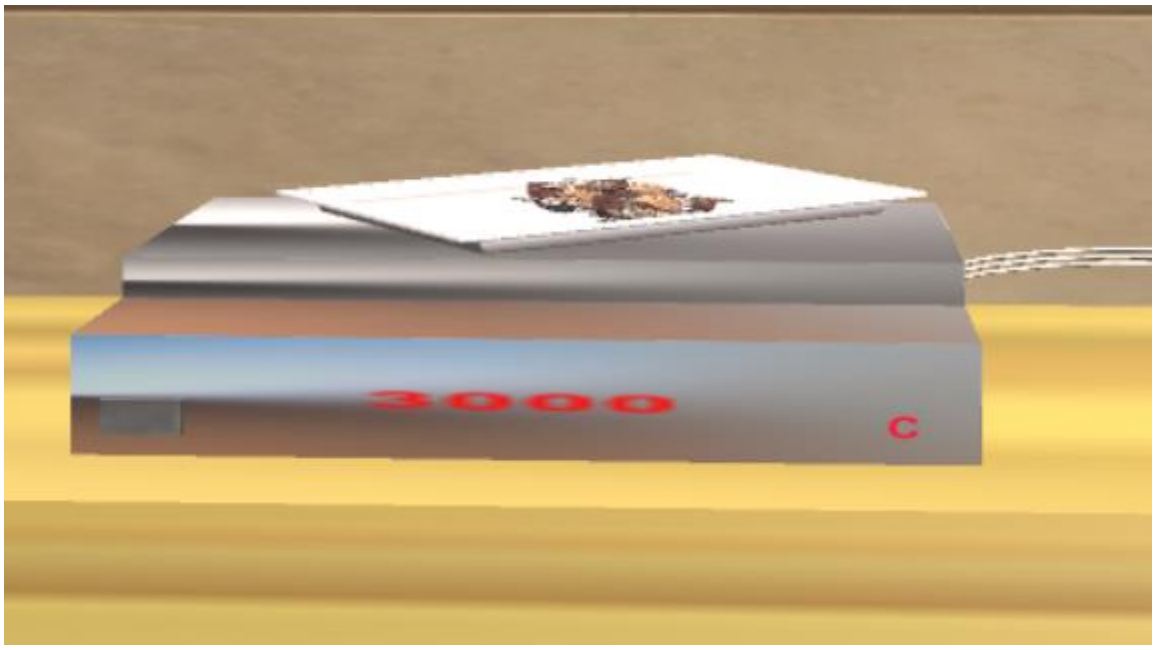
Αναλύοντας τον κώδικα, βλέπουμε ότι:

- συντάχθηκε η συνάρτηση `void updateStateScale(bool state)` η οποία θα πάρει ως εισερχόμενο όρισμα true/false από το script του κουμπιού ανοίγματος, ανάλογα αν η ζυγαριά είναι ανοιχτή ή κλειστή. Εφόσον είναι κλειστή, θα εμφανιστεί και το αντίστοιχο ενημερωτικό μήνυμα προς τον χρήστη.

- η συνάρτηση `void OnCollisionEnter(Collision col)`, ανιχνεύει τη λεγόμενη σύγκρουση (**collision**) όταν ένα δεύτερο αντικείμενο (προς ζύγισμα) εισέρχεται στο collider του πρώτου (ζυγαριά), θα ελέγξει το tag του αντικειμένου, και αναλόγως να πραγματοποιηθεί κάποια δράση ή όχι.

- συντάχθηκε η συνάρτηση `public void setStateTray()` η οποία εφόσον το δείγμα ζυγιστεί, θέτει true τη τρέχουσα μεταβλητή ώστε να ενημερωθεί και το script της σχάρας (rack) του φούρνου. Το σημείο αυτό είναι λίγο κρίσιμο. Όπως θα δούμε και θα αιτιολογήσουμε αργότερα αναλύοντας το αντικείμενο «φούρνος», αφήνουμε τον χρήστη να βάλει τον δίσκο στον φούρνο αν δεν το έχει ζυγίσει πρώτα, αλλά δεν τον αφήνουμε να ξεκινήσει το ψήσιμο.

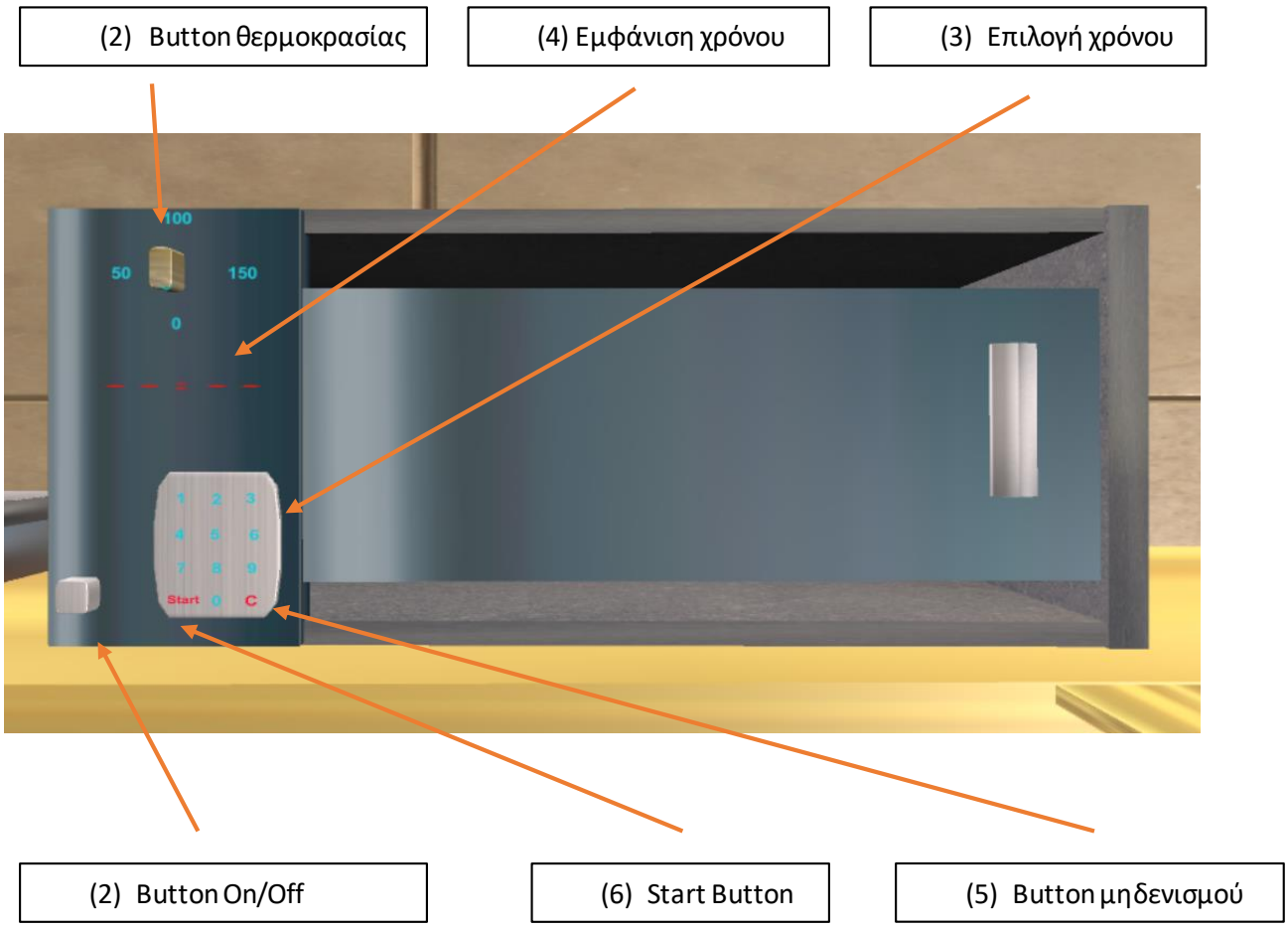
Εφόσον τοποθετηθεί στη περιοχή ζυγίσματος ο δίσκος, τότε ενεργοποιείται και η δράση εμφάνισης του βάρους. Το τελικό αποτέλεσμα, φαίνεται στη παρακάτω εικόνα:



(Εικόνα 4.6) – σχόλιο- η ζυγαριά δείχνει γραμμάρια

4.2.5 Φούρνος (Εικόνα 4.7 με σχόλια)

Συμπεριφορές του αντικειμένου είναι το άνοιγμα και κλείσιμο της φούρνου από τον χρήστη (σχόλιο 1), η ρύθμιση της θερμοκρασίας (σχόλιο 2), η ρύθμιση της ώρας ξήρανσης (σχόλιο 3), η εμφάνιση της ώρας (σχόλιο 4), η δυνατότητα μηδενισμού της ώρας (σχόλιο 5), και η έναρξη της διαδικασίας ξήρανσης (σχόλιο 6).



(Εικόνα 4.7)

Από τον κώδικα βλέπουμε ότι ο παραπάνω έλεγχος γίνεται με τη βοήθεια της γνωστής μας πλέον, συνάρτησης `void OnCollisionEnter(Collision col)`, η οποία ανιχνεύει την

επαφή της σχάρας με τα αντικείμενα κόσκινα, που έχουν tag «**sieveSeism**». Τότε ενεργοποιείται και το σχετικό μήνυμα προς τον χρήστη για τη μη καταλληλόλητα του αντικειμένου να μπει στο φούρνο, με τη βοήθεια της συνάρτησης `void OnGUI().`

Οι δράσεις του ορισμού θερμοκρασίας και διάρκειας ψησίματος (σχόλια 2 και 3 εικόνας 4.7), δεν ενεργοποιούνται εφόσον δεν έχει προηγηθεί η δράση ανοίγματος του φούρνου. Αυτό πραγματοποιείται από το κουμπί ανοίγματος/κλεισίματος του φούρνου (σχόλιο 1 εικόνας 4.7) (κώδικας 5).

Εφόσον έχει ανοιχτεί ο φούρνος, ο χρήστης θα πρέπει να ορίσει τη σωστή θερμοκρασία (σχόλιο 2 εικόνας 4.7) και τη σωστή διάρκεια ξήρανσης (σχόλιο 3 και 4 εικόνας 4.7). Θεωρήθηκε σκόπιμο για τη μεν θερμοκρασία να βγαίνει ενημερωτικό μήνυμα όταν το κουμπί φτάσει τη μέγιστη σωστή θερμοκρασία (κώδικας 6), και για δε τη χρονική διάρκεια να εμφανίζεται μήνυμα για την επιλογή λάθος χρόνου, χωρίς όμως να αναφέρεται η σωστή διάρκεια που απαιτεί το εργαστηριακό πείραμα (κώδικας 7), και στη συνέχεια να μηδενίζεται η οθόνης εμφάνισης της χρονικής διάρκειας.

Επόμενη δράση, είναι η έναρξη της ξήρανσης (κώδικας 8). Προυποθέσεις είναι να έχει οριστεί η θερμοκρασία, η διάρκεια ψησίματος, και να έχει ζυγιστεί το αρχικό δείγμα. Εδώ πρέπει να σημειωθεί, ότι ο χρήστης είναι ελεύθερος να ορίσει μόνος του τη σειρά κινήσεων που αφορούν την τοποθέτηση του αντικειμένου στον φούρνο, τον ορισμό της θερμοκρασίας και τον καθορισμό της διάρκειας ψησίματος.

Αναλύοντας τον παραπάνω κώδικα, βλέπουμε ότι οι συναρτήσεις:

- `void updateStateOven (bool state)`
- `void updateTemperatureOven (bool stateOfTemperature)`
- `void updateTimeOven (bool stateOfTime)` και
- `void updateStateTrayForOven (bool state)` παίρνουν ως εισερχόμενα ορίσματα true/false από τα αντίστοιχα scripts του κουμπιού ανοίγματος, της θερμοκρασίας, της χρονικής διάρκειας και της ζυγαριάς, και μόνο αν όλα είναι true

μπορεί ο χρήστης πατώντας το κουμπί start (σχόλιο 6 εικόνας 4.7) να ξεκινήσει η διαδικασία ξήρανσης.

- η συνάρτηση `void OnGUI()`, ενεργοποιεί τα μηνύματα προς τον χρήστη. Εμφανίζεται ενημερωτικό μήνυμα εάν δεν είναι ανοιχτός ο φούρνος, εάν δεν έχει οριστεί θερμοκρασία, εάν δεν έχει οριστεί η διάρκεια ξήρανσης, και μήνυμα αντίστροφης μέτρησης για την ολοκλήρωση της διαδικασίας ξήρανσης.

Σε αυτό το σημείο θα πρέπει να αναφερθεί, ότι το εργαστηριακό πείραμα επιβάλλει χρονική διάρκεια ξήρανσης 24ωρών. Για ευνόητους λόγους, δεν διαρκεί η ξήρανση στη προσομοίωση το ίδιο, αλλά έχει δημιουργηθεί επιτάχυνση στην αντίστροφη μέτρηση.

Τελευταία δράση, είναι η δυνατότητα μηδενισμού της χρονικής διάρκειας (σχόλιο 5 εικόνας 4.7). Ο χρήστης έχει τη δυνατότητα να μηδενίσει τον χρόνο, οποιαδήποτε στιγμή αντιληφθεί ότι δεν είναι σωστός ο χρόνος που έχει ορίσει. Μια επιπλέον παρατήρηση. Από τον προηγούμενο κώδικα (κώδικα 8) βλέπουμε ότι όταν τελειώσει η διαδικασία ξήρανσης του δείγματος, γίνεται αυτόματα επαναφορά του κουμπιού θερμοκρασίας στην αρχική του θέση και μηδενισμός της χρονικής διάρκειας, μέσω των συναρτήσεων:

- `void pressResetClock()`
- `void pressResetTemperature()`

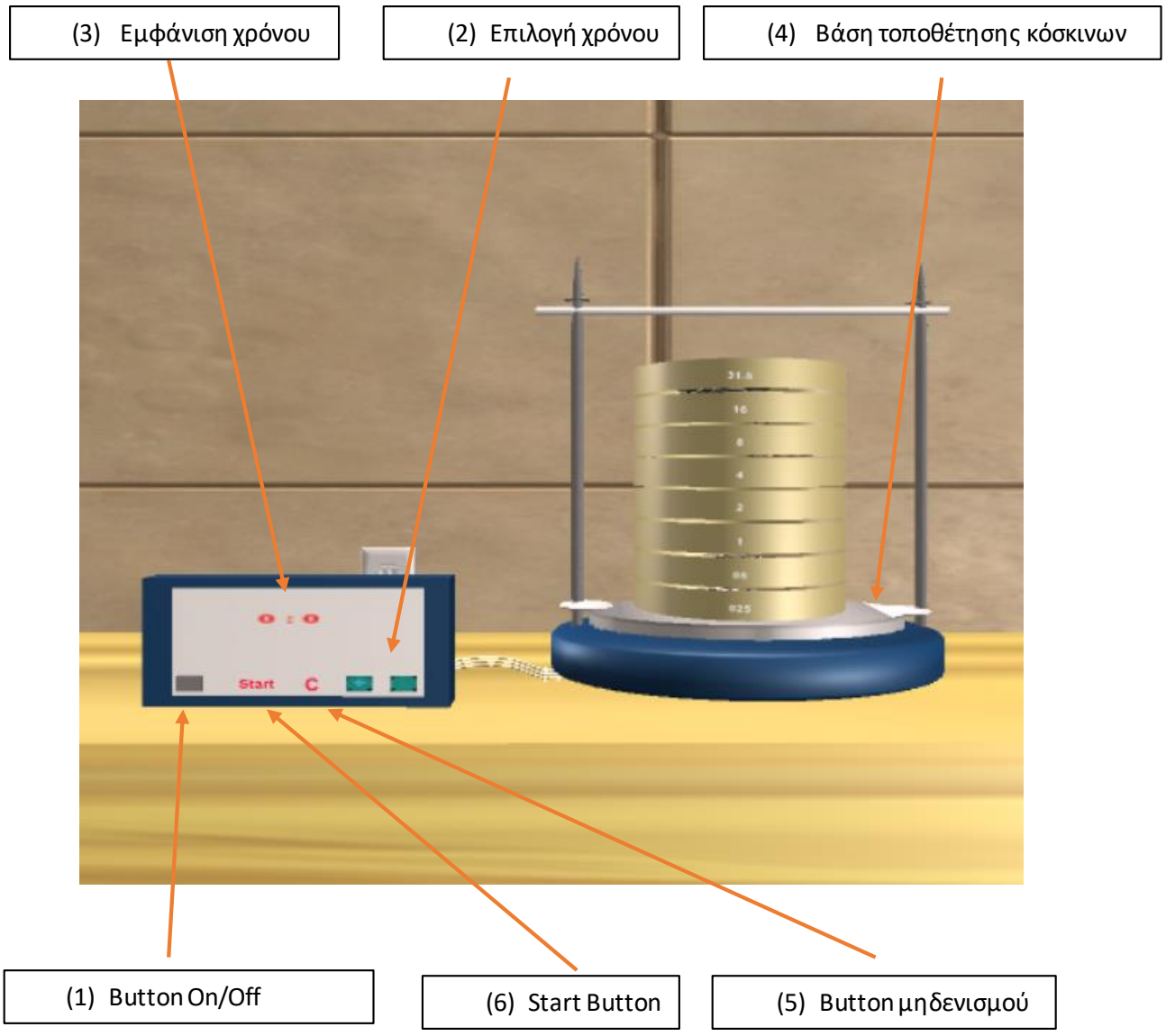
Οι συναρτήσεις αυτές εκτελούνται σε αντίστοιχα scripts για το κουμπί θερμοκρασίας (κωδικας 9) και το κουμπί επιλογής χρόνου (κωδικας 10).

Εφόσον τοποθετηθεί στη σχάρα το ζυγισμένο δείγμα, ορισθούν σωστά θερμοκρασία και διάρκεια ξήρανσης, μπορεί να ενεργοποιηθεί από τον χρήστη η δράση έναρξης του ψησίματος. Το τελικό αποτέλεσμα, φαίνεται στη παρακάτω εικόνα:

(Εικόνα 4.8)

4.2.6 Σείστρο (Εικόνα 4.9 με σχόλια)

Συμπεριφορές του αντικειμένου είναι το άνοιγμα και κλείσιμο του σείστρου από τον χρήστη (σχόλιο 1), η ρύθμιση της ώρας κοσκινίσματος (σχόλιο 2), η εμφάνιση της ώρας (σχόλιο 3), η βάση τοποθέτησης των κόσκινων (σχόλιο 4), η δυνατότητα μηδενισμού της ώρας (σχόλιο 5), και η έναρξη της διαδικασίας κοσκινίσματος (σχόλιο 6).



(Εικόνα 4.9)

Η ανάλυση των δράσεων τούτου του αντικειμένου, θα ξεκινήσει από τις προϋποθέσεις που θα πρέπει να έχει η δράση της έναρξης του κοσκινίσματος (σχόλιο 6 εικόνας 4.9). Πλήν της προυπόθεσης της σωστής χρονικής διάρκειας, και του ζυγισμένου και ξηραμένου δείγματος, υπάρχει η απαίτηση της σωστής σειράς τοποθέτησης των κόσκινων στη βάση (σχόλιο 4 εικόνας 4.9 και κώδικας 11), ανάλογα με το νούμερο που έχουν.

Επίσης θα πρέπει, πρώτα όλα τα κόσκινα να τοποθετηθούν στη βάση, και έπειτα να τοποθετηθεί το δείγμα στο πρώτο από πάνω κόσκινο. Εάν δεν συντασσόντουσαν στον κώδικα οι παραπάνω έλεγχοι, το πείραμα δεν θα κατέληγε σε σωστές μετρήσεις, με αποτέλεσμα να βγει λάθος τελικό συμπέρασμα.

Τέλος, παρουσιάστηκε δυσκολία στο κεντράρισμα που θα πρέπει να έχουν τα κόσκινα όταν τα τοποθετεί ο χρήστης στη βάση, καθώς και στην τοποθέτησή τους σε ομοιόμορφη στοίβα ώστε να μην γέρνουν. Θα δούμε πώς αντιμετωπίστηκε το πρόβλημα αυτό, όταν αναλύσουμε τον παρακάτω κώδικα.

Ξεκινώντας λοιπόν από τη δράση της βάσης (σχόλιο 4 εικόνας 4.9) του σείστρου έχουμε τις συναρτήσεις:

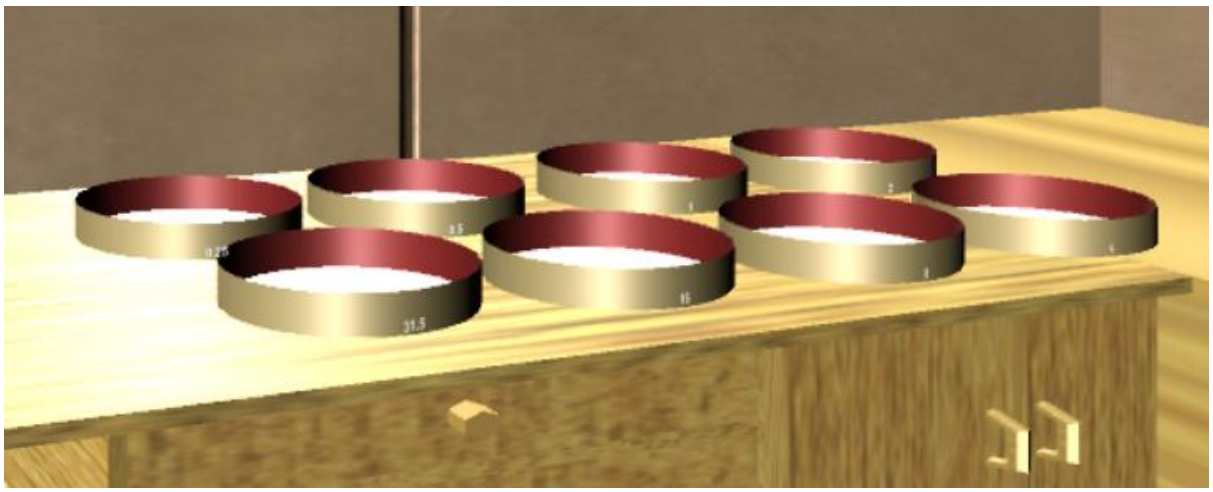
- `void OnCollisionEnter(Collision col)` η οποία μας διασφαλίζει ότι στη βάση θα τοποθετηθούν αντικείμενα με tag «**sieveSeism**». Εάν είναι το σωστό κόσκινο, τότε καλείται η συνάρτηση:

- `void putFisrtSieveOnSiesm()` η οποία με τη σειρά της θα εμφανίσει στη βάση του σείστρου ένα «φάντασμα» κόσκινο και θα εξαφανίσει το επιλεγμένο από τον χρήστη, ώστε να εμφανίζεται κεντραρισμένο στη βάση.

- η συνάρτηση `void OnGUI()` θα ενημερώσει τον χρήση ότι το πρώτο κόσκινο που τοποθέτησε δεν είναι το σωστό, χωρίς όμως να αναφέρεται το σωστό νούμερο κόσκινου το οποίο θα πρέπει να επιλέξει, ούτε τη σωστή σειρά τοποθέτησής τους.

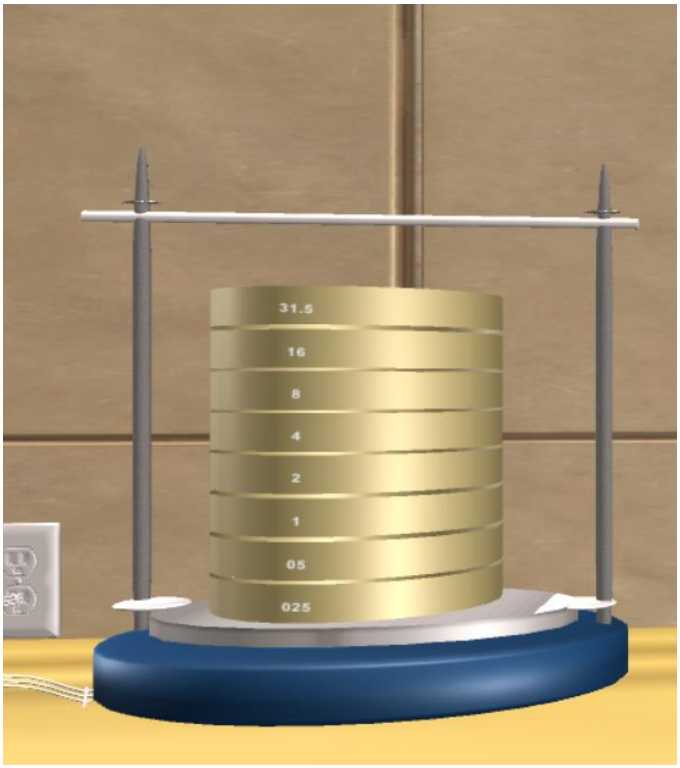
Στο σημείο αυτό, πρέπει να κάνουνε μια μικρή παρένθεση ώστε να γίνει κατανοητό πώς δημιουργείται αυτή η στοίβα από τα κόσκινα στη βάση. Κάθε κόσκινο που επιλέγει ο χρήστης

από τον εργαστηριακό πάγκο (εικόνα 4.10) , ελέγχεται εάν έχει tag «**sieveSeism**», και έπειτα εάν το όνομα του επιλεγέντος κόσκινου είναι το σωστό (κώδικας 12). Εφόσον είναι, τότε το κόσκινο «κάθεται» κεντραρισμένο στη βάση. Εκεί τη στιγμή που ακουμπάει το αντικείμενο κόσκινο το αντικείμενο βάση, τότε εξαφανίζεται το κόσκινο που ο χρήστης κρατούσε, και εμφανίζεται στη βάση ένα άλλο κόσκινο, ένα ghost κόσκινο. Η επόμενη κίνηση του χρήστη, είναι να πάρει το επόμενο σωστό κόσκινο και να το τοποθετήσει πάνω στο κόσκινο που βρίσκεται ήδη στη βάση. Την ώρα που το δεύτερο αυτό κόσκινο ακουμπάει το ghost κόσκινο, τότε πάλι εξαφανίζεται αυτό που ο χρήστης κρατούσε, και εμφανίζεται ένα δεύτερο ghost κόσκινο, ομοιόμορφα τοποθετημένο πάνω στο πρώτο ghost.



(Εικόνα 4.10)

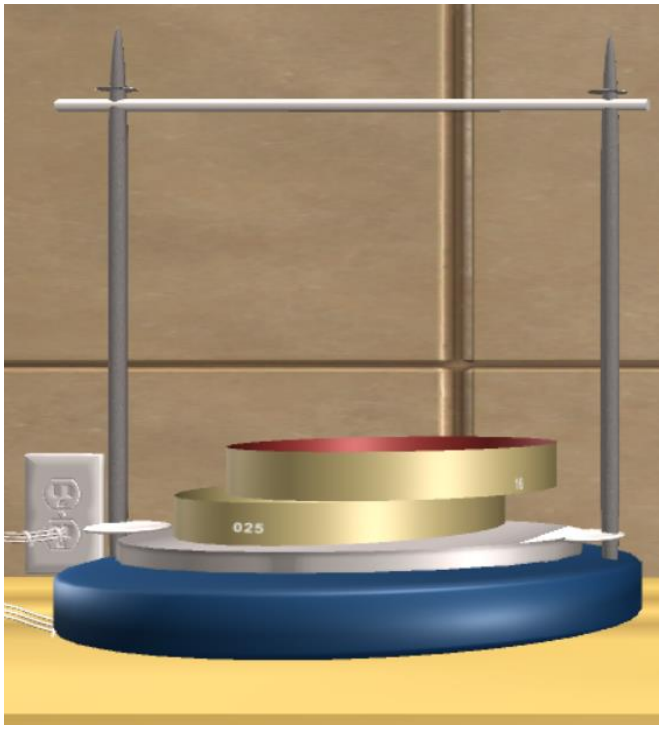
Δημιουργώντας με αυτό τον τρόπο ένα ντόμινο συναρτήσεων μεταξύ των κόσκινων, και πραγματοποιώντας ένα «τρικ» «εμφάνισης/εξαφάνισης» των κόσκινων, διασφαλίστηκε ότι τοποθετούνται από τον χρήστη τα κόσκινα με τη σωστή σειρά και ότι εμφανισιακά είναι μια κεντραρισμένη στοίβα από κόσκινα πάνω στη βάση του σείστρου (εικόνα 4.11).



(Εικόνα 4.11)

Επιπλέον, με την τοποθέτηση του τελευταίου κόσκινου ενημερώνεται ο χρήστης για τον τρόπο με τον οποίο θα ρίξει το δείγμα στο κόσκινο (κώδικας 13). Ο τρόπος αυτός, θα αναλυθεί παρακάτω, όταν θα μιλήσουμε για το αντικείμενο «δίσκος» και θα αναλύσουμε σημεία του κώδικα για τα αντικείμενα «κόσκινα».

Στη περίπτωση που δεν τοποθετήσει ο χρήστης κάποιο κόσκινο στη σωστή του σειρά, δεν θα εμφανιστεί το «φάντασμα» κόσκινο, δεν θα είναι τοποθετημένο στη στοίβα ομοιόμορφα (Εικόνα 4.12), με αποτέλεσμα να μην συμβεί κανένα από τα προαναφερθέντα και ο χρήστης θα πρέπει να επαναλάβει τη διαδικασία τοποθέτησης των κόσκινων με τη σωστή όμως σειρά, ώστε να μπορέσει να συνεχίσει το πείραμα.



(Εικόνα 4.12)

Τέλος, είναι χρήσιμο να αναφερθεί ότι μπορεί ο χρήστης πρώτα να τοποθετήσει τα κόσκινα, να ορίσει τη διάρκεια κοσκινίσματος και έπειτα να ζυγίσει και να ξηράνει το δείγμα. Για αυτή τη περίπτωση έχει γίνει η πρόβλεψη, να ενημερώνεται ο χρήστης με το ίδιο μήνυμα για τον τρόπο που θα ρίξει το δείγμα στο κόσκινο μετά την ολοκλήρωση της ξήρανσης.

Για τη δράση του ορισμού χρόνου κοσκινίσματος (σχόλιο 2 εικόνας 4.9) πατώντας ο χρήστης τα κουμπιά αύξησης (κώδικας 14) ή μείωσης (κώδικας 15), συντάχθηκαν οι κώδικες:

Στα δύο αυτά scripts, παρατηρεί κάποιος ότι μερικές συναρτήσεις είναι απενεργοποιημένες. Η αρχική σχεδιαστική ιδέα, ήταν τα κουμπιά της προσθήκης ή αφαίρεσης να αφορούσαν τα λεπτά και τα δευτερόλεπτα. Επειδή όμως το εργαστηριακό πείραμα δεν χρήζει της ανάγκης των δευτερολέπτων, αλλά μόνο τον σωστό ορισμό του χρόνου κοσκινίσματος σε ελάχιστα και μέγιστα λεπτά, τελικά ο σχεδιασμός έγινε μόνο για τη προσθαφαίρεση των λεπτών. Συνοπτικά έχουμε τις συναρτήσεις:

- `void press()` για το script της προσθήκης λεπτών (κώδικας 13), η οποία όταν πατήσει ο χρήστης το πλήκτρο «συν» θα ενεργοποιηθεί η συνάρτηση:

- `void addToMinutes(int minute)` ώστε να προστεθεί ένα (1) λεπτό **και** η συνάρτηση:

- `void sendMinutesToSubstract(int minute)` η οποία θα στείλει στο script αφαίρεσης (κώδικας 14) μέσω της συνάρτησης:

- `void receiveMinute(int sendMin)` ώστε να αφαιρεθεί ένα (1) λεπτό.

- `void press()` για το script της αφαίρεσης λεπτών (κώδικας 14), η οποία όταν πατήσει ο χρήστης το πλήκτρο «πλην» θα ενεργοποιηθεί η συνάρτηση:

- `void substractToMinutes(int minute)` ώστε να αφαιρεθεί ένα (1) λεπτό **και** η συνάρτηση:

- `void sendMinutesToPlus(int minute)` η οποία θα στείλει στο script προσθήκης (κώδικας 13) μέσω της συνάρτησης:

- `void receiveMinute(int sendMin)` ώστε να προστεθεί ένα (1) λεπτό.

Για τη δράση της έναρξης κοσκινίσματος, συντάχθηκε ο (κώδικας 16) ο οποίος συγκεντρώνει όλους τους απαιτούμενους ελέγχους για τη σωστή διαδικασία της τρέχουσας φάσης του πειράματος.

Αναλύοντας τον κώδικα, έχουμε τις συναρτήσεις:

- `void showTime(int Intime)` η οποία έχει ως εισερχόμενο όρισμα από την οθόνη εμφάνισης του χρόνου (σχόλιο 3 εικόνας 4.9) τα λεπτά που έχουν ορισθεί από τον χρήστη, και πραγματοποιεί τον έλεγχο, εάν αυτά τα λεπτά είναι ανάμεσα στα 3 και στα 5, όπως θα πρέπει.

- `void updateRocksBeingWeighted(bool weighted)` με εισερχόμενο όρισμα true/false από τη ζυγαριά ότι το δείγμα έχει ζυγιστεί.

- `void updateRocksBeingCooked(bool cooked)` με εισερχόμενο όρισμα true/false από τον φούρνο ότι το δείγμα έχει ξηραθεί.

- `void updateRocksOnSieve31(bool stateRocks)` με εισερχόμενο όρισμα true/false από το τελευταίο από πάνω κόσκινο, ότι έχει τοποθετηθεί σε αυτό το δείγμα (κώδικας 16).

- `void startShaking()`, η οποία εφόσον όλες οι παραπάνω συναρτήσεις είναι true τότε θα ξεκινήσει η διαδικασία του κοσκινίσματος.

- `void showAllRocks()`, μας εμφανίζει μετά το κοσκίνισμα τα συγκρατούμενα δείγματα που έχουν τα κόσκινα ώστε να συνεχιστεί το εργαστηριακό πείραμα.

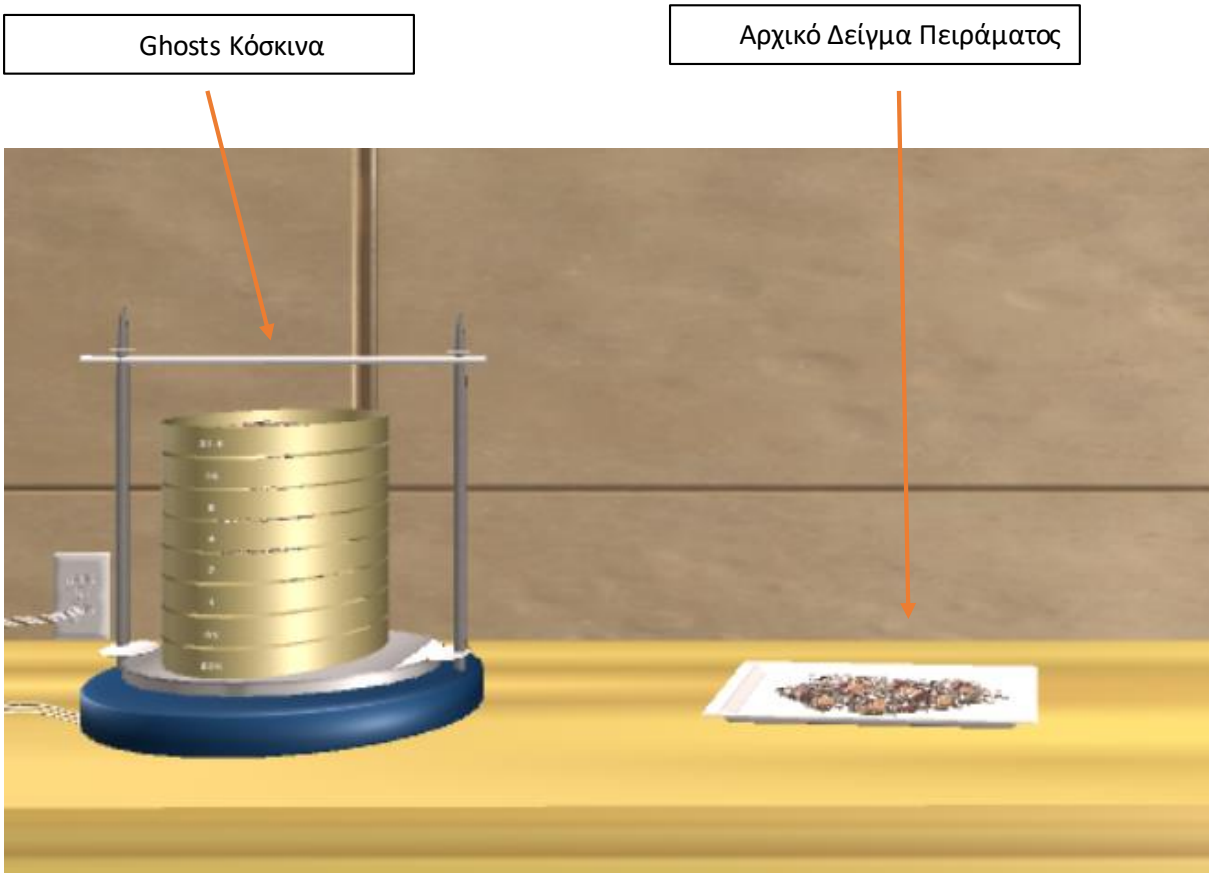
- τέλος, η `συνάρτηση void OnGUI()`, ενεργοποιεί τα μηνύματα προς τον χρήστη. Εμφανίζεται ενημερωτικό μήνυμα για το άνοιγμα του σείστρου εάν είναι κλειστό, τη λάθος επιλογή χρόνου κοσκινίσματος (χωρίς να δίδονται περισσότερες οδηγίες), τη μη τοποθέτηση του δείγματος, και την αντίστροφη μέτρηση της διαδικασίας του κοσκινίσματος.

Εφόσον όλη η διαδικασία πραγματοποιηθεί σωστά, θα έχουμε το τελικό αποτέλεσμα της παρακάτω εικόνας:

(Εικόνα 4.13)

4.2.7 Δίσκος/Κόσκινα (Εικόνα 4.14)

Το αντικείμενο «δίσκος» αποθηκεύει όλες τις προϋποθέσεις του εργαστηριακού πειράματος (δείγμα ζυγισμένο, ψημένο, τοποθετημένο δείγμα προς κοσκίνισμα), και τα κόσκινα δρουν εξαρτώμενα από τη σωστή μετέπειτα δράση του δίσκου μετά το πέρας και του κοσκινίσματος. Λόγω αυτής της άμεσης σύνδεσης, θα αναλύσουμε τον κώδικα και για το αντικείμενο «δίσκος» (κώδικας 17) και για τα αντικείμενα «κόσκινα» (κώδικας 18).



(Εικόνα 4.14)

Έχουμε ήδη αναφερθεί, ότι για να ξεκινήσει η διαδικασία του κοσκινίσματος θα πρέπει εκτός από δείγμα ζυγισμένο, ψημένο, και θα πρέπει να τοποθετηθεί αυτό, στο σωστό κόσκινο. Εδώ θα σταθούμε στο σημείο του κώδικα στον οποίο γίνεται το rotate του δίσκου, ώστε να πραγματοποιηθεί η παραπάνω κίνηση.

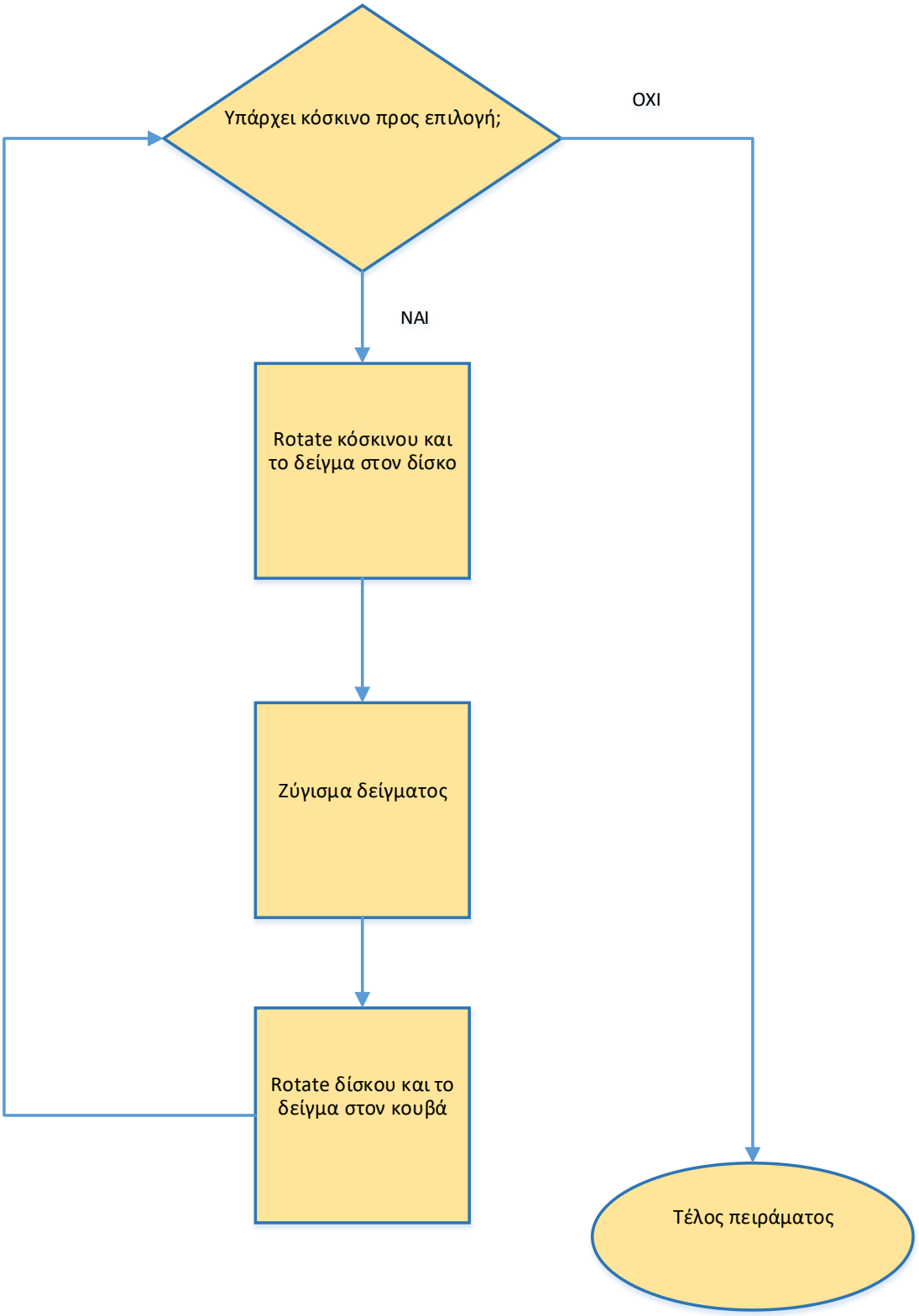
Ο τρόπος που θα γίνει το rotate, εμφανίζεται σε μήνυμα προς το χρήστη αμέσως μετά το ψήσιμο του δείγματος, και χρησιμοποιούνται οι κάτωθι συναρτήσεις:

- `public IEnumerator rotate()`και

- `StartCoroutine(rotate())`

Η `StartCoroutine(rotate())` βρίσκεται μέσα στη συνάρτηση `void Update()` και ενεργοποιείται από τις ορισθείσες προϋποθέσεις των `if`. Τότε, γίνεται κλήση της `IEnumerator rotate()` η οποία στο σώμα της περιέχει τις εντολές του rotate του δίσκου και την επαναφορά του στην αρχική οριζόντια θέση.

Μετά το κοσκίνισμα, και σύμφωνα με το εργαστηριακό πείραμα, θα πρέπει ο χρήστης να πάρει το κάθε κόσκινο, να του κάνει rotate ώστε να πέσει το αντίστοιχο συγκρατούμενο δείγμα στον δίσκο, και να το ζυγίσει. Στη συνέχεια ρίχνει πάλι με rotate του δίσκου το ζυγισμένο δείγμα στον κουβά, και παίρνει το επόμενο κόσκινο. Με αυτή την επαναληπτική διαδικασία, όταν θα ζυγιστούν όλα τα συγκρατούμενα δείγματα, και θα βρίσκονται στον κουβά, ο χρήστης δεν θα έχει πλέον άλλες κινήσεις, και το εργαστηριακό πείραμα θα τελειώσει. Η διαδικασία αυτή με χρήση διαγράμματος ροής προγράμματος είναι:


Υπάρχει κόσκινο προς επιλογή;
ΟΧΙ
ΝΑΙ
Rotate κόσκινου και το δείγμα στον δίσκο
Ζύγισμα δείγματος
Rotate δίσκου και το δείγμα στον κουβά
Τέλος πειράματος

Για την επαναληπτική αυτή διαδικασία, χρησιμοποιούνται οι συναρτήσεις:

- `takeSieve16.GetComponent<ghosts>().chooseNumberSieve(sieve16)` η οποία παίρνει ως όρισμα το κόσκινο το οποίο θα πρέπει να πάρει ο χρήστης μετά το κοσκίνισμα. Βρίσκεται στο script `startSeism`, (κώδικας 16 ο οποίος έχει ήδη αναλυθεί) και είναι η αρχική εντολή για την έναρξη των παραπάνω επαναληπτικών κινήσεων. (Να σημειωθεί εδώ, ότι δεν παίρνει ο χρήστης το πρώτο από πάνω κόσκινο της στοίβας, διότι δεν έχει συγκρατούμενο δείγμα, επομένως δεν χρειάζεται να γίνει καμία κίνηση)

- εφόσον πάρει το κόσκινο με νούμερο 16, για το rotate του κόσκινου χρησιμοποιούνται οι ίδιες προαναφερθείσες εντολές. Μόνο που αυτή τη φορά μέσα στο σώμα της `public IEnumerator rotate()` (κώδικας 18) έχουμε σύγκριση του κόσκινου που είχε η προηγούμενη συνάρτηση `chooseNumberSieve(sieve16)` με κάποιο από τα κόσκινα, και αντίστοιχα ενεργοποιεί την εντολή:

- `detachFromParent16()` κ.τ.λ. (κώδικας 16) ώστε την ώρα του rotate του κόσκινου, να πέσει το αντίστοιχο δείγμα του στον δίσκο.

- μέσα στο σώμα της προηγούμενης συνάρτησης `detachFromParent16()`, υπάρχει η `weight16.updateRealWeight(rocksWeight16)` η οποία θα εμφανίσει το αντίστοιχο βάρος του δείγματος όταν ο δίσκος τοποθετηθεί στη ζυγαριά.

- όταν ζυγιστεί το δείγμα, εκτελείται η συνάρτηση: `trayForBucket.GetComponent<tray>().chooseSieveForBucket(sieve16)` η οποία βρίσκεται στο script `plateScale`, και έχει ως σκοπό να δώσει την εντολή στον δίσκο να κάνει rotate ώστε να πέσει το ζυγισμένο πλέον δείγμα στον κουβά.

- η συνάρτηση `public void chooseSieveForBucket(GameObject _current)` (κώδικας 17) θα συγκρίνει το κόσκινο από το οποίο έχει προέλθει το

δείγμα που έχει ο δίσκος, με κάποια από τα κόσκινα, και με την εντολή:

- `public IEnumerator rotateForBucket()` θα γίνει το rotate του δίσκου, ώστε με την εντολή:

- `public void detach16ForBucket()` κ.τ.λ. να πέσει το δείγμα στον κουβά, η οποία μόλις εκτελεστεί καλεί την:

- `nextSieve8.GetComponent<ghosts>().chooseNumberSieve(sieve8)` η οποία θα πάρει ως νέο όρισμα πλέον το επόμενο κόσκινο.

Με αυτόν τον τρόπο, συνεχίζεται η διαδικασία για όλα τα κόσκινα, έως ότου ζυγιστούν όλα τα συγκρατούμενα δείγματα, ριχθούν στον κουβά και ο χρήστης δεν έχει άλλες κινήσεις.

Κάπου εδώ, τελειώνει και το εργαστηριακό πείραμα «Προσδιορισμός Κοκκομετρικής Σύνθεσης Αδρανούς Υλικού» στο πεδίο του Πολιτικού Μηχανικού.

4.3 User Interface

Για να μπορέσει ο χρήστης να ξεκινήσει την προσομοίωση, δημιουργήθηκε μια επιπλέον σκηνή η οποία περιέχει τον τίτλο του εργαστηριακού πειράματος και το κουμπί Έναρξης του παιχνιδιού. (Εικόνα 4.15)

(Εικόνα 4.15)

Εισήχθη το λεγόμενο panel μέσα στο οποίο μπορούμε να εισαγάγουμε οποιαδήποτε άλλα button μας χρειάζονται. Στο background του panel εισήχθη φωτογραφία με θέμα του πεδίο του Πολιτικού Μηχανικού. Όταν ο χρήστης πατήσει το κουμπί έναρξης τότε μεταφέρεται στη σκηνή του εργαστηρίου και μπορεί να ξεκινήσει τη προσομοίωση του πειράματος.

Εφόσον ο χρήστης τελειώσει το πείραμα και δεν έχει πλέον άλλες κινήσεις εμφανίζεται η σκηνή τερματισμού (Εικόνα 4.16), στην οποία μπορεί εάν επιθυμεί, να ξεκινήσει το πείραμα από την αρχή.

(Εικόνα 4.16)

5. ΔΙΑΧΕΙΡΙΣΗ ΕΡΓΟΥ

Για την παρακολούθηση της διαδικασίας σχεδιασμού και υλοποίησης του 3D παιχνιδιού, χρησιμοποιήθηκαν ορισμένες διαδικασίες διαχείρισης έργου, οι οποίες όμως προσαρμόστηκαν στις ανάγκες της παρούσης διπλωματικής εργασίας.

Για να αξιολογηθεί η χρησιμότητα αυτής της εφαρμογής ως βοηθητικό εργαλείο εκμάθησης στα εργαστηριακά και θεωρητικά μαθήματα που διδάσκονται οι φοιτητές, πραγματοποιήθηκαν σύντομες συνεντεύξεις με φοιτητές. Οι ερωτήσεις της συνέντευξης ήταν:

- Πόσα εργαστήρια ολοκληρώνετε επιτυχώς κάθε ακαδημαϊκό έτος;
- Πόσο εποικοδομητική είναι η παρακολούθηση του εργαστηρίου, με βάση το πλήθος των σπουδαστών, τον εξοπλισμό και τον χρόνο διδασκαλίας;
- Θα επιθυμούσατε να έχετε την δυνατότητα με προσομοίωση, να ξανακάνετε κάποιο εργαστηριακό πείραμα από τον υπολογιστή σας;
- Πόσο πιστεύετε ότι αυτό θα σας βοηθούσε στην καλύτερη κατανόηση του μαθήματος;
- Σε ποια άλλα μαθήματα θα επιθυμούσατε να υπάρχουν προσομοιώσεις πειραμάτων;

<u>Το στατιστικό αποτέλεσμα των συνεντεύξεων:</u>

- Μέσος Όρος επιτυχούς ολοκλήρωσης εργαστηρίων στο έτος: 4
- Εποικοδομητική παρακολούθηση εργαστηριακού μαθήματος: 50%
- Επιθυμία δυνατότητας προσομοίωσης εργαστηριακού πειράματος: 80%
- Βελτίωση στη κατανόηση του μαθήματος: 70%
- Επιθυμητά μαθήματα για δημιουργία εργαστηριακών προσομοιώσεων: σχεδόν όλα.

5.1 Προσδιορισμός Απαιτήσεων και Υπολογισμός Χρόνου Υλοποίησης

Κατεγράφησαν όλες οι προδιαγραφές του εργαστηριακού πειράματος, οι οποίες είναι απαραίτητες για την ορθή υλοποίησή του και τη σωστή εξαγωγή δεδομένων και εκτιμήθηκε η χρονική διάρκεια που απαιτείται για την υλοποίηση του έργου.

Πίνακας 1 Εκτιμώμενη Απαιτούμενη Χρονική Προσπάθεια:

A/A	Απαίτηση / Προδιαγραφές	Ανθρωποημέρες
R1	➢ Εύρεση βιβλιογραφίας για εκπαιδευτικές μεθόδους ➢ Συλλογή πληροφοριών σχετικά με η Unity	**30**
R2	➢ Σχεδιασμός Αντικειμένων στη Unity α) περιβάλλον εργαστηρίου β) αντικείμενο «ζυγαριά» γ) αντικείμενο «φούρνος» δ) αντικείμενο «σείστρο» ε) αντικείμενα «κόσκινα» στ) αντικείμενο «δίσκος»	**70**
R3	➢ Υλοποίηση πειράματος στη Unity α) ζύγισμα δείγματος β) ψήσιμο δείγματος β1) χρονική διάρκεια ψησίματος β2) θερμοκρασία ψησίματος γ) κοσκίνισμα δείγματος γ1) χρονική διάρκεια κοσκινίσματος γ2) ορθή τοποθέτηση κόσκινων στο σείστρο γ3) τοποθέτηση δείγματος στο σωστό κόσκινο	**70**

	δ) πλήθος κόσκινων ε) ζύγισμα συγκρατούμενων δειγμάτων για κάθε κόσκινο	
R4	➢ Έλεγχος / Διόρθωση υλοποιημένου πειράματος ➢ Συγγραφή / Διόρθωση τελικού κειμένου	**30**
	ΣΥΝΟΛΟ	**200**

Τέλος, θα πρέπει να γίνει η παρατήρηση, ότι ορισμένες απαιτήσεις μπορούν να γίνουν ταυτόχρονα με κάποιες άλλες. Παρατίθεται σε επόμενο κεφάλαιο το αναλυτικό χρονοδιάγραμμα του έργου, αφού πρώτα έχουμε χωρίσουμε το όλο έργο σε κάποιες φάσεις. Εκεί θα αποφασίσουμε ποιες απαιτήσεις μπορούν να υλοποιηθούν ταυτόχρονα και σε αυτό το σημείο θα φανεί και η μείωση της χρονικής διάρκειας υλοποίησης του έργου.

5.1.1 Ανάλυση Προδιαγραφών

Στο σημείο αυτό, θα αναλυθούν οι προδιαγραφές της απαίτησης R3 οι οποίες έχουν το σχεδιαστικό και προγραμματιστικό ενδιαφέρον. Αναλυτικότερα:

α. ζύγισμα δείγματος: Σύμφωνα με τη θεωρία του εργαστηριακού πειράματος, το αρχικό δείγμα θα πρέπει πρώτα απ΄ όλα να ζυγιστεί.

β. ψήσιμο δείγματος: Εφόσον έχει ζυγιστεί, τότε και μόνο τότε θα πρέπει να τοποθετηθεί για ψήσιμο.

β1. χρονική διάρκεια ψησίματος: Απαιτείται 24ώρες ψησίματος.
β2. θερμοκρασία ψησίματος: Απαιτείται θερμοκρασία έως 105 – 110^{c}.

γ. κοσκίνισμα δείγματος: Μόνο όταν ολοκληρωθεί το ζύγισμα και το ψήσιμο του αρχικού δείγματος, μπορούμε να το κοσκινίσουμε.

γ1. χρονική διάρκεια κοσκινίσματος: Απαιτείται 3 έως 5 λεπτά κοσκινίσματος

γ2. ορθή τοποθέτηση κόσκινων στο σείστρο: Τα κόσκινα θα πρέπει να τοποθετηθούν στο

σείστρο με συγκριμένη σειρά. Στη βάση του σείστρου θα μπει πρώτα το μικρότερο σε

νούμερο κόσκινο, και με αύξουσα σειρά έως το τελευταίο με το μεγαλύτερο σε νούμερο.

γ3. τοποθέτηση δείγματος στο σωστό κόσκινο: Το δείγμα θα πρέπει να τοποθετηθεί σε

συγκεκριμένο κόσκινο, και ειδικότερα σε αυτό που έχει το μεγαλύτερο νούμερο και θα

βρίσκεται στην κορυφή της στοίβας.

δ. πλήθος κόσκινων: Απαιτείται η χρήση οκτώ (8) γερμανικών κόσκινων.

ε. ζύγισμα συγκρατούμενων δειγμάτων για κάθε κόσκινο: Μετά το πέρας του κοσκινίσματος, για κάθε κόσκινο που έχει συγκρατούμενο δείγμα χώματος θα πρέπει αυτό να ζυγιστεί.

5.1.2 Προγραμματιστικές Αποφάσεις Προδιαγραφών

Για όλες τις παραπάνω προδιαγραφές ο προγραμματιστής καλείται να πάρει κάποιες αποφάσεις σχετικά με την ευελιξία κινήσεων που θα έχει ή όχι ο χρήστης της προσομοίωσης του πειράματος. Αναλυτικά οι αποφάσεις που πάρθηκαν είναι:

- Απαίτηση R2:

  α. περιβάλλον εργαστηρίου: Θεωρήθηκε σκόπιμο για την πιο εύκολη προσαρμογή του χρήστη στο εικονικό περιβάλλον του εργαστηρίου, να είναι πολύ κοντά εμφανισιακά στο πραγματικό εργαστήριο. Όπως επίσης και τα αντικείμενα που υπάρχουν στη σκηνή, να είναι τοποθετημένα στην περίπου ίδια χωρική διάταξη.

  β. αντικείμενο «ζυγαριά»: Εφόσον τοποθετηθεί ο δίσκος που περιέχει το αρχικό δείγμα πάνω στη ζυγαριά, η οθόνη της θα δείξει το καθαρό βάρος. Σε περίπτωση τοποθέτησης άλλου αντικειμένου, η οθόνη δεν δείχνει τίποτα.

  γ. αντικείμενο «φούρνος»: Εάν τοποθετηθεί μέσα στον φούρνο κάποιο από τα κόσκινα, ενημερώνεται ο χρήστης με μήνυμα ότι δεν επιτρέπεται αυτή η κίνηση. Αυτό διότι, αν τα κόσκινα ψηθούν θα καταστραφούν.

  δ. αντικείμενο «σείστρο»: Στη περίπτωση που τοποθετηθεί κάποιο άλλο αντικείμενο πλην των κόσκινων, το σείστρο δεν θα ξεκινήσει τη διαδικασία κοσκινίσματος.

  ε. αντικείμενα «κόσκινα»: -

  στ. αντικείμενο «δίσκος»: -

- Απαίτηση R3: Η απαίτηση αυτή είχε και το μεγαλύτερο προγραμματιστικό ενδιαφέρον ως προς την απόφαση ελευθερίας κινήσεων του χρήστη. Καλείται ο προγραμματιστής να αποφασίσει σε ποια σημεία θα μπορεί ο χρήστης να κινηθεί ή όχι χωρίς περιορισμούς, σε ποια σημεία είναι άκρως απαραίτητο η ενημέρωσή του για

κάποια λάθος κίνηση η οποία θα έχει επίπτωση στη μη σωστή υλοποίηση του πειράματος. Αναλυτικά πάρθηκαν οι κάτωθι αποφάσεις:

α. ζύγισμα δείγματος: Ενημερώνεται ο χρήστης στη περίπτωση που τοποθετήσει τον δίσκο χωρίς να έχει ανοίξει τη ζυγαριά ότι θα πρέπει να πατήσει το αντίστοιχο button ανοίγματός της.

β. ψήσιμο δείγματος: Διακρίνουμε τα κάτωθι:

- εάν τοποθετηθεί το δείγμα ζυγισμένο χωρίς να έχει ορισθεί θερμοκρασία ψησίματος,
- εάν τοποθετηθεί το δείγμα ζυγισμένο χωρίς να έχει ορισθεί χρονική διάρκεια ψησίματος,
- εάν έχουν ορισθεί χρονική διάρκεια και θερμοκρασία αλλά δεν έχει τοποθετηθεί ο δίσκος με το δείγμα,
- εάν ορισθεί χρονική διάρκεια, θερμοκρασία, έχει τοποθετηθεί ο δίσκος με το δείγμα αλλά δεν έχει ανοιχτεί ο φούρνος, τότε ο χρήστης σε όλες αυτές τις υποπεριπτώσεις ενημερώνεται με σχετικό μήνυμα.

και:

- εάν ορισθεί λάθος χρονική διάρκεια,
- εάν ορισθεί λάθος θερμοκρασία,
- εάν έχουν ορισθεί χρονική διάρκεια και θερμοκρασία αλλά τοποθετηθεί το δείγμα χωρίς να είναι ζυγισμένο, ενημερώνεται ο χρήστης για το λάθος, χωρίς όμως να δίδονται περισσότερες πληροφορίες, διότι θα πρέπει να γνωρίζει τη σωστή διαδικασία υλοποίησης του πειράματος.

γ. κοσκίνισμα δείγματος: Έχουμε τη περίπτωση:

- εάν δεν ορισθεί η κατάλληλη διάρκεια κοσκινίσματος, ενημερώνεται ο χρήστης, αλλά χωρίς περισσότερες πληροφορίες.

δ. πλήθος κόσκινων: -

ε. ορθή τοποθέτηση κόσκινων στο σείστρο: Στη βάση του κόσκινου εάν δεν τοποθετηθεί το σωστό κόσκινο δεν εμφανίζεται το ghost κόσκινο και ενημερώνεται ο χρήστης για τη λάθος κίνηση, χωρίς όμως να αναγράφεται ποιο είναι το σωστό κόσκινο. Έπειτα για κάθε κόσκινο που τοποθετείται πάνω στη στοίβα από τα κόσκινα, εάν δεν είναι το σωστό, δεν εμφανίζεται το αντίστοιχό του ghost κόσκινο, ενημερώνεται ο χρήστης με μήνυμα αλλά χωρίς περαιτέρω λεπτομέρειες. Θεωρείται ότι ο χρήστης γνωρίζει τη σωστή σειρά τοποθέτησης των κόσκινων.

στ. τοποθέτηση δείγματος στο σωστό κόσκινο: Το δείγμα δεν μπορεί να τοποθετεί (να γίνει rotate του δίσκου) εάν δεν είναι ψημένο και ζυγισμένο. Δεν ενημερώνεται ο χρήστης με κάποιο μήνυμα, διότι κρίθηκε ότι είναι απαραίτητη αυτή η θεωρητική γνώση από τον χρήστη.

ζ. ζύγισμα συγκρατούμενων δειγμάτων για κάθε κόσκινο: Μετά το πέρας κοσκινίσματος, ξεκινώντας από την κορυφή στης στοίβας και προς τα κάτω, θα μπορεί ο χρήστης να παίρνει το κόσκινο ώστε να κάνει rotate και να πέφτει το αντίστοιχο συγκρατούμενο δείγμα του στον δίσκο. Μετά το ζύγισμα του πρώτου συγκρατούμενου δείγματος, ενεργοποιείται η δυνατότητα rotate του επόμενου κόσκινου. Ούτε σε αυτή τη περίπτωση ενημερώνεται ο χρήστης από κάποιο μήνυμα.

Συμπερασματικά: Για λάθος κινήσεις του χρήστη οι οποίες αφορούν τη θεωρητική γνώση στην οποία βασίζεται το εργαστηριακό πείραμα, ναι μεν ενημερώνεται ο χρήστης, αλλά δεν δίδονται περαιτέρω λεπτομέρειες ή οδηγίες. Επίσης δεν εμφανίζονται μηνύματα για τα βήματα

της όλης διαδικασίας, επομένως δεν είναι μια προσομοίωση καθοδηγούμενη. Τα μηνύματα που λαμβάνει ο χρήστης, δεν αφορούν το θεωρητικό υπόβαθρο του πειράματος, αλλά είναι μηνύματα γενικής φύσεως ή υπενθύμισης.

5.2 Χρονοδιάγραμμα

Το έργο της πτυχιακής χωρίστηκε σε 4 φάσεις. Σε αυτό το σημείο αποφασίστηκε ποιες απαιτήσεις και φάσεις του έργου μπορούν να έχουν ταυτόχρονη υλοποίηση. Ο κάτωθι πίνακας περιέχει τις επιμέρους εργασίες των τεσσάρων φάσεων, που απαιτήθηκαν για την εκπόνηση της διπλωματικής εργασίας και με βάση αυτόν τον πίνακα δημιουργήθηκε και το αντίστοιχο διάγραμμα Gantt.

Πίνακας 2: Πίνακας Εργασιών

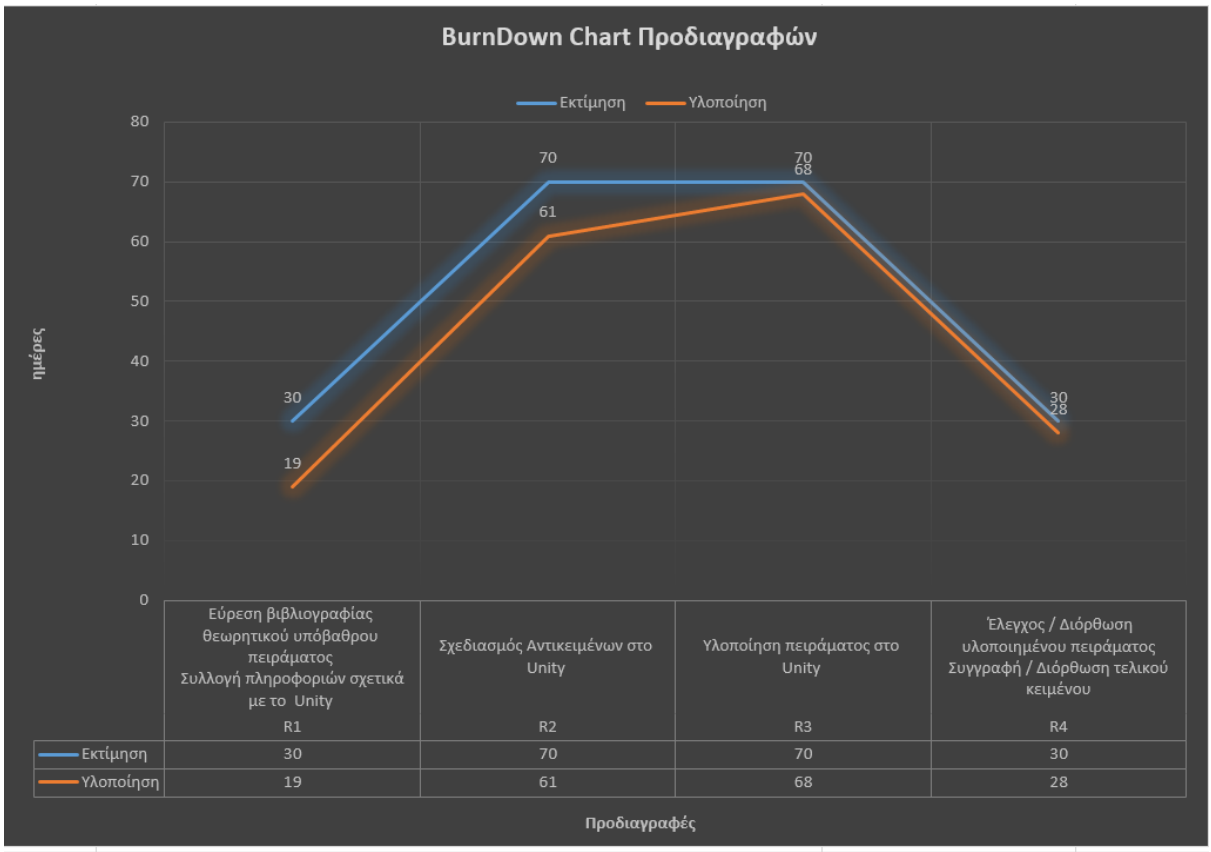
		SPRINT							
		Φάση 1η		**Φάση 2η**		**Φάση 3η**		**Φάση 4η**	
		Μήνες							
Απαίτηση	**Εργασία**	Οκτώβριος	Νοέμβριος	Δεκέμβριος	Ιανουάριος	Φεβρουάριος	Μάρτιος	Απρίλιος	Μάιος
R1	Εύρεση βιβλιογραφίας για εκπαιδευτικές μεθόδους	√							
	Συλλογή πληροφοριών σχετικά με τη Unity	√	√	√	√	√	√	√	√
R2	Σχεδιασμός Αντικειμένων στη Unity		√	√	√				
R3	Υλοποίηση πειράματος στη Unity				√	√	√		
R4	Έλεγχος /Διόρθωση υλοποιημένου πειράματος						√	√	√
	Συγγραφή/Διόρθωση τελικού κειμένου							√	√

Σύμφωνα με τον παραπάνω πίνακα εργασιών, τροποποιήθηκε και ο αρχικός πίνακας υπολογισμού της χρονικής προσπάθειας υλοποίησης του έργου. Θα δούμε ότι η τελική χρονική διάρκεια μειώθηκε αρκετά της προβλεπόμενης, διότι υπήρξαν απαιτήσεις οι οποίες μπορούσαν να πραγματοποιηθούν ταυτόχρονα, επειδή η μία δεν είχε προαπαιτούμενη κάποια άλλη πλην του τελικού ελέγχου και διόρθωσης του πειράματος που απαιτεί την ολοκλήρωση των R2, R3. Είναι εφικτή η υλοποίηση μέρους του πειράματος με τα ήδη σχεδιασμένα αντικείμενα, πχ μπορεί να ψηθεί το αρχικό δείγμα ώστε να ελέγξει ο προγραμματιστής κάποιες διεργασίες του αντικειμένου «φούρνος» χωρίς τις απαραίτητες προϋποθέσεις ή μέρος αυτών, και αυτές να τις εισαγάγει μετέπειτα.

Πίνακας 3 Τελικός Χρόνος Υλοποίησης:

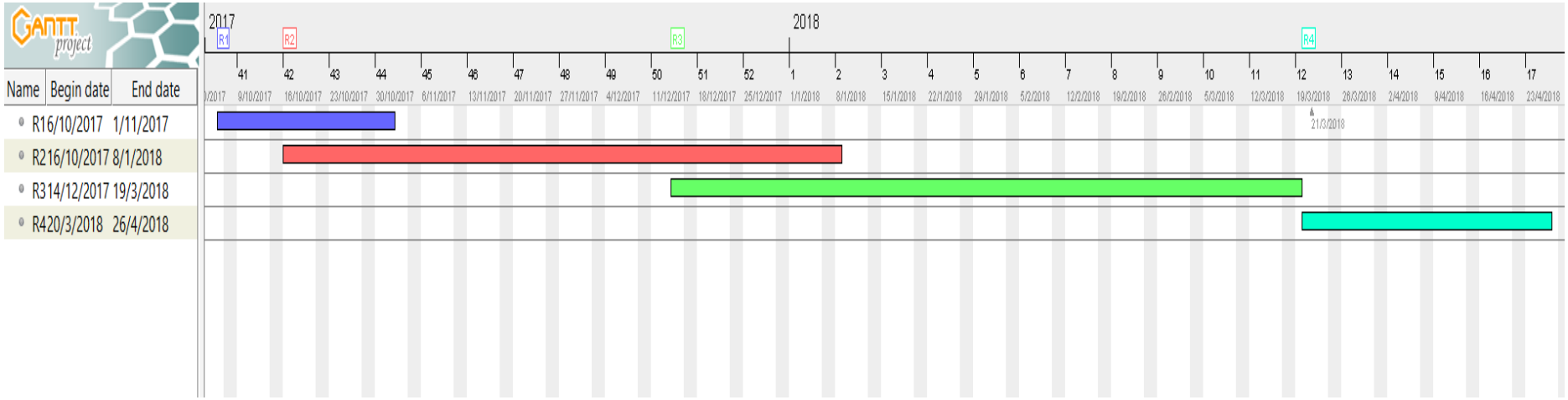
A/A	Απαίτηση / Προδιαγραφές	Ανθρωποημέρες
R1	➢ Εύρεση βιβλιογραφίας για εκπαιδευτικές μεθόδους ➢ Συλλογή πληροφοριών σχετικά με τη Unity	**19**
R2	➢ Σχεδιασμός Αντικειμένων στη Unity α) περιβάλλον εργαστηρίου β) αντικείμενο «ζυγαριά» γ) αντικείμενο «φούρνος» δ) αντικείμενο «σείστρο» ε) αντικείμενα «κόσκινα» στ) αντικείμενο «δίσκος»	**61**
R3	➢ Υλοποίηση πειράματος στη Unity α) ζύγισμα δείγματος β) ψήσιμο δείγματος β1) χρονική διάρκεια ψησίματος β2) θερμοκρασία ψησίματος γ) κοσκίνισμα δείγματος γ1) χρονική διάρκεια κοσκινίσματος γ2) ορθή τοποθέτηση κόσκινων στο σείστρο γ3) τοποθέτηση δείγματος στο σωστό κόσκινο δ) πλήθος κόσκινων ε) ζύγισμα συγκρατούμενων δειγμάτων για κάθε κόσκινο	**68**
R4	➢ Έλεγχος / Διόρθωση υλοποιημένου πειράματος ➢ Συγγραφή / Διόρθωση τελικού κειμένου	**28**
	ΣΥΝΟΛΟ:	**176**

Η σύγκριση μεταξύ του εκτιμώμενου χρόνου υλοποίησης του έργου και του τελικού χρόνου φαίνεται στο διάγραμμα κατανάλωσης προσπάθειας:



	Εύρεση βιβλιογραφίας θεωρητικού υπόβαθρου πειράματος Συλλογή πληροφοριών σχετικά με το Unity	Σχεδιασμός Αντικειμένων στο Unity	Υλοποίηση πειράματος στο Unity	Έλεγχος / Διόρθωση υλοποιημένου πειράματος Συγγραφή / Διόρθωση τελικού κειμένου
	R1	R2	R3	R4
Εκτίμηση	30	70	70	30
Υλοποίηση	19	61	68	28

Διάγραμμα Gantt:



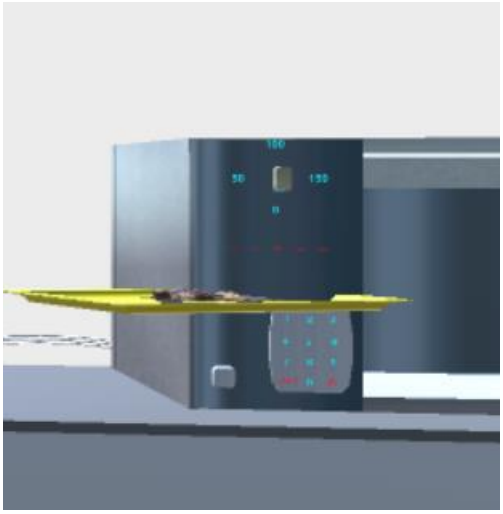
6. ΛΑΘΗ/ΣΥΜΠΕΡΑΣΜΑΤΑ

Στη παρούσα εργασία έγινε μια προσπάθεια να προσομοιωθεί σε ένα 3D παιχνίδι ένα εργαστηριακό πείραμα στο πεδίο του Πολιτικού Μηχανικού. Οι προδιαγραφές για τη σωστή του υλοποίηση ήταν αρκετές και μερικές από αυτές αρκετά δύσκολες. Οι προγραμματιστικές αποφάσεις που πάρθηκαν είχαν εκπαιδευτικό γνώμονα και λογική. Σκοπός του παιχνιδιού είναι η εξοικείωση του χρήστη τόσο με τον εργαστηριακό χώρο, αλλά περισσότερο με τη γνώση που αποκτά ο χρήστης, παίζοντας ένα παιχνίδι. Μη έχοντας οδηγίες για τα βήματα που έχει το πείραμα και μη έχοντας ολική ελευθερία κινήσεων, δίνεται η ευκαιρία στον χρήστη, εφόσον δεν γνωρίζει την όλη διαδικασία του πειράματος, πρώτα να τη μάθει διαβάζοντας τη σχετική θεωρία, και μετά να προσπαθήσει να εφαρμόσει αυτή τη γνώση.

6.1 Λάθη στο 3D παιχνίδι

Είναι απαραίτητο να γίνει αναφορά στα προγραμματιστικά λάθη ή ελλείψεις της παρούσας εργασίας. Καταβλήθηκε προσπάθεια να γίνει όσο το δυνατόν καλύτερη υλοποίηση των απαιτούμενων προδιαγραφών. Αυτή η προσπάθεια ήταν επιτυχής, αλλά δεν ήταν τόσο εφικτή, κυρίως λόγω χρόνου, η διόρθωση ορισμένων σημείων της προσομοίωσης. Τα σημεία αυτά παρουσιάζονται επιγραμματικά και είναι:

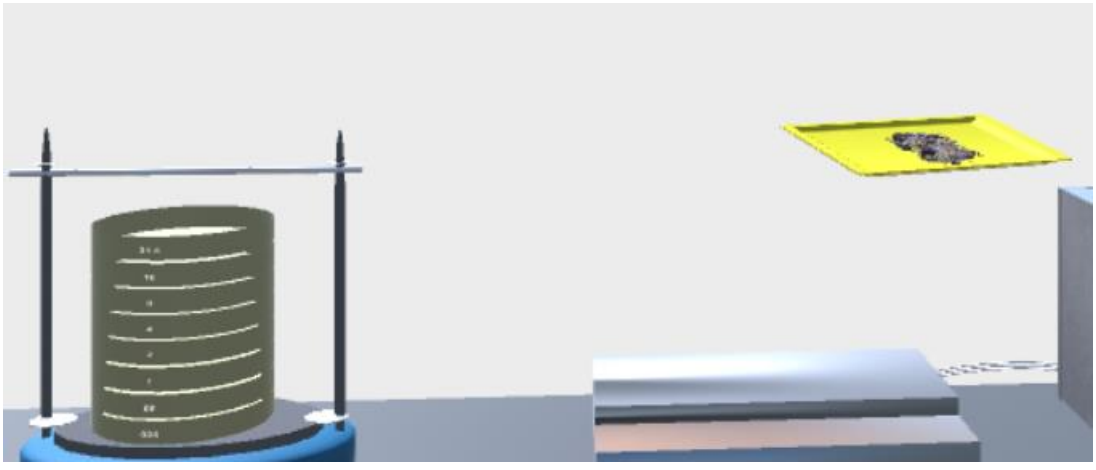
- Δεν αντιμετωπίστηκε το πρόβλημα της μη σύγκρουσης μεταξύ αντικειμένων. Όταν κάποιο αντικείμενο πλησιάζει κάποιο άλλο, θα πρέπει να υπάρξει η λεγόμενη σύγκρουση, ώστε να μην διαπερνάει το ένα το άλλο. (Εικόνα 6.1)



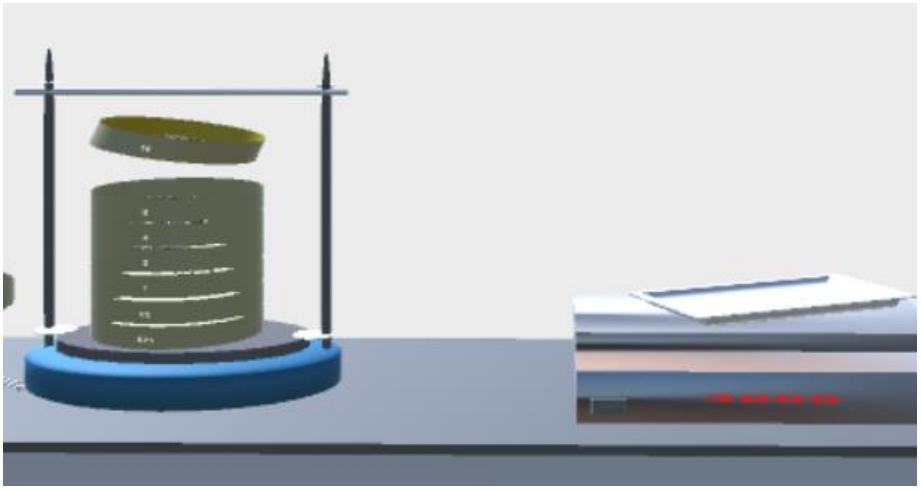
(Εικόνα 6.1)

- Δεν ήταν εφικτή η σωστή επαναφορά (αρχική θέση) του button του φούρνου για τη θερμοκρασία.

- Δεν λύθηκε ότι το rotate του δίσκου ή το rotate των κόσκινων θα πρέπει να γίνεται όταν αυτά τα αντικείμενα τα κρατά ο χρήστης, ακριβώς από πάνω από άλλο. Ο δίσκος θα πρέπει να κάνει rotate μόνο όταν βρίσκεται πάνω από το σωστό κόσκινο, και τα κόσκινα θα πρέπει να κάνουν rotate μόνο όταν βρίσκονται πάνω από τον δίσκο. (Εικόνα 6.2, Εικόνα 6.3)



(Εικόνα 6.2)



(Εικόνα 6.3)

6.2 Μελλοντικές Επεκτάσεις 3D παιχνιδιού

Είναι ιδιαίτερα χρήσιμο, οποιαδήποτε εφαρμογή να είναι ευέλικτη ως προς μελλοντικές διορθώσεις και επεκτάσεις. Θα πρέπει να είναι εφικτό, σε μια εφαρμογή όπως αυτή της προσομοίωσης, να μπορεί να εφαρμοστεί και κάποιο άλλο πείραμα. Στην προσομοίωση που παρουσιάστηκε, οι διορθώσεις / επεκτάσεις που κατά προσωπική άποψη μπορούν να υλοποιηθούν είναι:

- Τα κουμπιά προσθαφαίρεσης του σείστρου, να αφορούν λεπτά και δευτερόλεπτα. Αυτό έχει ήδη δημιουργηθεί (βλ. ανάλυση για σείστρο 4.2), αλλά απενεργοποιήθηκαν για τα δευτερόλεπτα λόγω μη άμεσης χρησιμότητας για το τρέχον πείραμα.

- Εμφάνιση μηνυμάτων σχετικά με όλες τις απαιτούμενες κινήσεις για την σωστή πραγματοποίηση του εργαστηριακού πειράματος και όχι απλό ενημερωτικό μήνυμα (Εικόνα 6.4). Δηλαδή η προσομοίωση να καθηγηθεί απόλυτα τον χρήστη.

(Εικόνα 6.4) αν πατήσει ο χρήστης λάθος διάρκεια ψησίματος, η ζυγαριά μηδενίζει και εμφανίζεται ενημερωτικό μήνυμα

- Ολική αφαίρεση μηνυμάτων, είτε ενημερωτικών είτε για τις σωστές κινήσεις που απαιτούνται, ώστε ο χρήστης να μπαίνει σε συνεχή διαδικασίας σκέψης και αναζήτησης των σωστών κινήσεων.

- Δημιουργία πολλαπλών αρχικών δειγμάτων και συγκρατούμενων δειγμάτων, ώστε να μπορεί ο χρήστης να συγκρίνει και να βγάζει αντίστοιχα αποτελέσματα και συμπεράσματα.

- Στον ίδιο περιβάλλον χώρο, μπορούμε να σχεδιάσουμε και άλλα εργαστηριακά όργανα, ώστε να δημιουργήσουμε είτε σύνθετα πειράματα που το ένα έχει συνέχεια από το άλλο, είτε να είναι ανεξάρτητα μεταξύ τους.

6.3 Συμπεράσματα

Το παρόν 3D παιχνίδι, δημιουργήθηκε ώστε να υποβοηθήσει τους φοιτητές του Τμήματος Πολιτικών Μηχανικών να κατανοήσουν τη θεωρητική διαδικασία της κοκκομετρικής σύνθεσης αδρανούς υλικού.

Ο φοιτητής μπορεί να κινηθεί ελεύθερα στον εικονικό χώρο του εργαστηρίου και να πειραματιστεί με τα διαδραστικά αντικείμενά του.

Γνωρίζοντας τη θεωρία του συγκεκριμένου πειράματος, μπορεί να την εφαρμόσει στη πράξη και να βγάλει τα διάφορα επιστημονικά συμπεράσματα.

Κατά τη διάρκεια του παιχνιδιού, ο χρήστης λαμβάνει μηνύματα λάθους, στις βασικές μόνο κινήσεις που θα πρέπει να γίνουν. Επειδή δεν δίδονται αναλυτικές οδηγίες, ο φοιτητής μπαίνει σε περαιτέρω διαδικασία σκέψης και αναζήτησης των λεπτομερειών του πειράματος πχ. κάποιας θερμοκρασίας.

Μπορούμε να έχουμε δύο επιπλέον προσεγγίσεις για ίδιο πείραμα στο μέλλον. Την εμφάνιση μηνυμάτων στον χρήστη για κάθε ένα βήμα και για κάθε λεπτομέρεια ή το ακριβώς αντίθετο, την πλήρη εξάλειψη μηνυμάτων.

Αυτές είναι δύο «ακραίες» εκπαιδευτικές προσεγγίσεις μάθησης μέσω του 3D παιχνιδιού. Στη μία περίπτωση καθοδηγείται πλήρως ο φοιτητής, σαν να βρισκόταν δια ζώσης στο εργαστήριο με τον καθηγητή του, και στην άλλη περίπτωση, εξασκείται στην όλη διαδικασία χωρίς κάποια βοήθεια.

Στη παρούσα εργασία, χρησιμοποιήθηκε η μέση οδός, δηλαδή, μία στοιχειώδη καθοδήγηση - υπενθύμιση του χρήστη. Όλες αυτές οι επιλογές στο 3D παιχνίδι εξαρτώνται από τον καθηγητή, και κατά πόσο επιθυμεί το εκπαιδευτικό λογισμικό να συμμετέχει ως καθοδηγητής στο μάθημα ή να επικουρεί και συνεργάζεται στη διαδικασία της μάθησης με τον χρήστη.

7. ΑΝΑΦΟΡΕΣ/ΒΙΒΛΙΟΓΡΑΦΙΑ

1. Forum Unity: https://forum.unity.com/

2. Ν. Παπαχαρίσης & Ν. Μάνου-Ανδρεάδη & Ι. Γραμματικόπουλος (2010), «Γεωτεχνική Μηχανική», Κυριακίδη ΑΦΟΙ

3. Οδηγός Unity: https://answers.unity.com/index.html

4. Θ.Κ. Τσιάτσος (2015), «Εκπαιδευτικά Περιβάλλοντα Διαδικτύου», Κάλλιπος – Ελληνικά Ακαδημαϊκά Ηλεκτρονικά Συγγράμματα και Βοηθήματα.

5. Σ. Κωστόπουλος (2006), «Πειραματική Γεωτεχνική Μηχανική», ΙΩΝ

6. YouTube, βιντεομαθήματα

7. ΦΕΚ 955/31-12-1986 «Προδιαγραφές Εργαστηριακών Δοκιμών Εδαφομηχανικής»

8. ΠΑΡΑΡΤΗΜΑ (κώδικας C#)

Παρατίθεται μέρος του κώδικα, που αφορά τα κυριότερα σημεία αλληλεπίδρασης των αντικειμένων μεταξύ τους ή με τον χρήστη.

```
public class mouseEvents : MonoBehaviour
{

    void Start()
    {
        defaultC = gameObject.GetComponent<Renderer>().material.color;
        yellow = new Color(1.0F, 0.92F, 0.016F, 1.0F);

        clickCurson = Resources.Load("finger") as Texture2D;    //για click το
ποντίκι γίνεται finger
        grabCursor = Resources.Load("grab") as Texture2D;       //για drag το
ποντίκι γίνεται grab
        rotateCursor = Resources.Load("hand") as Texture2D;     //για rotate το
ποντίκι γίνεται hand
        cursor = CursorMode.Auto;
        mouseClick = new Vector2(9, 2);
        mouseGrab = new Vector2(7, 2);
        mouseRotate = new Vector2(11, 3);
        mouseCombine = new Vector2(8, 2);

    }
    void changeColor(Color _color)  // μέθοδος για αλλαλή χρώματος αντικειμένου
    {
        GetComponent<Renderer>().material.color = _color;
    }
    void restoreColor(Color _color) // μέθοδος για επαναφορά αρχικού χρώματος
αντικειμένου
    {
        GetComponent<Renderer>().material.color = _color;
    }
    public void OnMouseEnter()
    {
        changeColor(yellow);    //αλλαλή χρώματος αντικειμένου όταν το ποντίκι
εισέρχεται στο collider
        if (gameObject.tag == "click")              // για αλλαγή εμφάνισης
ποντικιού
        {
            Cursor.SetCursor(clickCurson, mouseClick, cursor);
        }
        else if (gameObject.tag == "grab")
        {
```

```
            Cursor.SetCursor(grabCursor, mouseGrab, cursor);
        }
        else if (gameObject.tag == "rotate")
        {
            Cursor.SetCursor(rotateCursor, mouseRotate, cursor);
        }
        else if( gameObject.tag == "trayOvenScale")
        {
            Cursor.SetCursor(grabCursor, mouseGrab, cursor);
        }
    }
    public void OnMouseExit()
    {
        restoreColor(defaultC); //επαναφορά αρχικού χρώματος αντικειμένου όταν το
ποντίκι φεύγει από το collider του αντικειμένου
        if (gameObject.tag == "click" || gameObject.tag == "grab" || gameObject.tag
== "rotate" || gameObject.tag =="trayOvenScale") //επαναφορά default σχήματος
        {
            Cursor.SetCursor(null, Vector2.zero, cursor); //επαναφορά της εμφάνισης
του ποντικιού
        }
    }
    public void OnMouseUpAsButton()
    {
        if (gameObject.tag == "click" )
        {
            gameObject.SendMessage("press");
        }

    }
}
```

(κώδικας 1)

```
public class gamerMoves : MonoBehaviour
{
     void Update ()
    {
        if (Input.GetKey(KeyCode.W))      //μπροστά
        {
            transform.Translate(0f, 0f, 150f);
        }
        else if (Input.GetKey(KeyCode.S))   //πίσω
        {
            transform.Translate(0f, 0f, -150f);
        }
        if (Input.GetKey(KeyCode.UpArrow))        //ψηλά
        {
            transform.Translate(0f, 150f, 0f);
        }
        else if (Input.GetKey(KeyCode.DownArrow))   //χαμηλά
        {
            transform.Translate(0f, -150f, 0f);
        }
        if (Input.GetKey(KeyCode.A))    //δεξιά
        {
            transform.Rotate(0f, 0.5f, 0f);
        }
```

```
        else if (Input.GetKey(KeyCode.D))    //αριστερά
        {
            transform.Rotate(0f, -0.5f, 0f);
        }
    }}
```

(κώδικας 2)

```
void Start ()
    {
       //η τοπική μεταβλητή για κλήση του component του αντίστοιχου object
        onOffPlate = onOffButton.GetComponent<onOffScale>();
        screenPlate = screen.GetComponent<screenScale>();
      //  τρέχουσα μεταβλητή = gameObject....<script τρέχουσας μεταβλητής>
        stateTrayForRack = rack.GetComponent<rack>();
    }
    public void updateStateScale(bool state)
    {
        buttonOnOffScale = state;
    }
    public void updateRealWeight(float _realWeight)
    {
        realWeight = _realWeight;
    }
    public void setStateTray(bool _state)
    {
        stateOfWeight = _state;
        stateTrayForRack.updateStateTrayForOven(stateOfWeight); //κλήση της
συνάρτησης updateStateTrayForOven της κλάσης rack
    }
    void OnCollisionEnter(Collision col)
    {
        collidedWith = col.gameObject;
        if(collidedWith.tag == "trayOvenScale")
        {
            if (buttonOnOffScale == ON)
            {
                screenPlate.setWeight(realWeight);  //κλήση συνάρτησης setWeight της
κλάσης screenScale με όρισμα το καθαρό βάρος για εμφάνιση στην οθόνη
                setStateTray(true); // τίθεται true η μεταβλητή ότι το δείγμα
ζυγίστηκε
            }
            else if (buttonOnOffScale == OFF)
            {
                showText1 = true;
            }
        }
    }

    void OnGUI()
    {
        if (showText1 == true)
        {
            GUI.Box(new Rect(0, 50, Screen.width, Screen.height - 50), text1);
            if (Input.GetKeyDown(KeyCode.Escape))
            {
```

```
                showText1 = false;
            }
        }
    }
```

(κώδικας 3)

```
public class rack : MonoBehaviour
{
    void OnCollisionEnter(Collision col)
    {
        collidedWith = col.gameObject;
             if (collidedWith.tag == "sieveSeism")
             {
                 showText1 = true;
             }
    }
    void OnGUI()
    {
        if (showText1 == true)
        {
            GUI.Box(new Rect(0, 50, Screen.width, Screen.height - 50), text1);
            if (Input.GetKeyDown(KeyCode.Escape))
            {
                showText1 = false;
            }
        }
    }
}
```

(κώδικας 4)

```
   public void setState(bool _state)
    {
        state = _state;
        updateStateOven.updateStateOven(state); // κλήση της συνάρτησης
updateStateOven της κλάσης digitalClock
        updateOven.updateOven(state);
        updateOvenForStart.updateStateOven(state);
    }
    // Use this for initialization
    void Start()
    {
        updateStateOven = digitalClock.GetComponent<digitalClock>();
        updateOven = temperatureButton.GetComponent<setTemperature>();
        updateOvenForStart = startButtonOven.GetComponent<startOven>();
    }
```

(κώδικας 5)

```
void OnMouseDrag()
```

```
    {
        if (buttonOnOffOven == ON )
        {
            rotation += Input.GetAxis("Mouse X") * rotateSpeed * Time.deltaTime;
            transform.Rotate(0, 0, rotation);
            rotatedPosition = new Vector3(0, 0, rotation);
            if (rotation >= 7.5)
            {
                showText1 = true;
                temperatureDone(true);
            }
        }}
```

(κώδικας 6)

```
  public void showDigit(int _digit)
    {
        if (buttonOnOffOven == ON )
        {
            if (nextDigitIndex < 4)
            {
                timeDone(false);
                digitCharacter[nextDigitIndex] = _digit.ToString()[0];
                GetComponent<TextMesh>().text = "";
                GetComponent<TextMesh>().text = string.Concat(digitCharacter[0] + "
" + digitCharacter[1] + " : " + digitCharacter[2] + " " + digitCharacter[3]);
                nextDigitIndex++;
            }
            if ( nextDigitIndex == 4)
            {
                if( (digitCharacter[0] == '2') &&  (digitCharacter[1] == '4') &&
(digitCharacter[2] == '0') &&  (digitCharacter[3] == '0') )
                {
                     timeDone(true);
                }
                else
                {
                    showText1 = true;
                    resetNumbers();
                }
            }
          }
 }
```

(κώδικας 7)

```
public class startOven : MonoBehaviour
{
    // Use this for initialization
    void Start()
    {
        cookedForTray = tray.GetComponent<tray>();
        cookedForSeism = buttonStartSeism.GetComponent<startSeism>();
        cookedForGhostSieve31 = ghostSieve31.GetComponent<sieve31>();
```

```
        forRocks = sieve31.GetComponent<sieve31>();
    }
    public void updateStateOven(bool state)     //συναρτήσεις για ανοιγμένο φούρνο,
ορισμένη θερμοκρασία, ορισμένο χρόνο ψησίματος και ζυγισμένο δείγμα
    {
        buttonOnOffOven = state;
    }

    public void updateTemperatureOven(bool stateOfTemperature)
    {
        temperatureIsSet = stateOfTemperature;
    }
    public void updateTimeOven(bool stateOfTime)
    {
        timeIsSet = stateOfTime;
    }
    public void updateStateTrayForOven(bool state)
    {
        rocksBeingWeighted = state;

    }
    public void startBaking()
    {
        if(otherCollider.tag == "trayOvenScale")
        {

            if (buttonOnOffOven == ON && temperatureIsSet == ON && timeIsSet == ON
&& rocksBeingWeighted == ON)
            {
                showText1 = true;       //έναρξη ψησίματος
            }
            if (buttonOnOffOven == OFF)
            {
                showText2 = true;       //μήνυμα για άνοιγμα φούρνου
            }

            if (buttonOnOffOven == ON && temperatureIsSet == OFF)
            {
                showText3 = true;       //μήνυμα για ορισμό θερμοκρασίας
            }
            if (buttonOnOffOven == ON && timeIsSet == OFF)
            {
                showText4 = true;       //μήνυμα για ορισμό χρόνου ψησίματος
            }
           }
      }

    public void rocksCooked(bool cooked)
    {
        rocksBeingCooked = cooked;
        cookedForTray.updateRocksBeingCooked(rocksBeingCooked); // κλήση της
συνάρτησης updateRocksBeingCooked της κλάσης tray
        cookedForSeism.updateRocksBeingCooked(rocksBeingCooked);
        cookedForGhostSieve31.updateRocksBeingCooked(rocksBeingCooked);
        forRocks.GetComponent<sieve31>().rocksPutOnSieve();
    }
    void OnGUI()
```

```
    {
        if (showText1 == true)
        {

           myTimer -= Time.deltaTime * 5000;  // * 5000 για επιτάχυνση του χρόνου
           int seconds = (int)(myTimer % 60);
           int minutes = (int)(myTimer / 60) % 60;
           int hours = (int)(myTimer / 3600) % 24;
            string timerString = string.Format("{0:0} : {1:00} : {2:00}", hours,
minutes, seconds);
            text1 = ("Χρόνος ψησίματος: ") + timerString ;
            if (myTimer < endTime)
            {
                showText1 = false;
                rocksCooked(true);
                pressResetClock();
                pressResetTemperature();
              }
            GUI.Box(new Rect(0, 50, Screen.width, Screen.height - 50), text1);

            if (Input.GetKeyDown(KeyCode.Escape))
            {
                showText1 = false;
            }
        }

        if (showText2 == true)
        {
            GUI.Box(new Rect(0, 50, Screen.width, Screen.height - 50), text2);
            if (Input.GetKeyDown(KeyCode.Escape))
            {
                showText2 = false;
            }
        }
        if (showText3 == true)
        {
            GUI.Box(new Rect(0, 50, Screen.width, Screen.height - 50), text3);
            if (Input.GetKeyDown(KeyCode.Escape))
            {
                showText3 = false;
            }
        }
        if (showText4 == true)
        {
            GUI.Box(new Rect(0, 50, Screen.width, Screen.height - 50), text4);
            if (Input.GetKeyDown(KeyCode.Escape))
            {
                showText4 = false;
            }
        }
    }

     void press()
     {
        startBaking();
      }
```

```
    public void pressResetClock()
    {
        digitalClock.GetComponent<digitalClock>().resetNumbers();       //reset η
ώρα
    }

    public void pressResetTemperature()      //ΔΕΝ ΔΟΥΛΕΥΕΙ ΝΑ GYRIZEI STHN ARXIKH
THESI
    {
        temperatureButton.GetComponent<setTemperature>().resetTemperature();
//reset στη θερμοκρασία
    }

    // Update is called once per frame
    void Update()
    {

    }
}
```

(κώδικας 8)

```
public void resetTemperature()      //ΔΕΝ ΔΟΥΛΕΥΕΙ
    {
        restoredPosition = new Vector3(0, 0, 0);
    }
```

(κωδικας 9)

```
public void resetNumbers()
    {
        nextDigitIndex = 0;
        digitCharacter[0] = '-';
        digitCharacter[1] = '-';
        digitCharacter[2] = '-';
        digitCharacter[3] = '-';
        GetComponent<TextMesh>().text = string.Concat(digitCharacter[0] + " " +
digitCharacter[1] + " : " + digitCharacter[2] + " " + digitCharacter[3]);
    }
```

(κωδικας 10)

```
public class baseSeism : MonoBehaviour
{
    // Use this for initialization
    void Start()
    {
        ghostSieveAfterSeism31.SetActive(false);   //να μην εμφανίζονται τα κόσκινα
        ghostSieveAfterSeism16.SetActive(false);
        ghostSieveAfterSeism8.SetActive(false);
```

```
        ghostSieveAfterSeism4.SetActive(false);
        ghostSieveAfterSeism2.SetActive(false);
        ghostSieveAfterSeism1.SetActive(false);
        ghostSieveAfterSeism05.SetActive(false);
        ghostSieveAfterSeism025.SetActive(false);
        ghostRocks025.SetActive(false);       //να μην εμφανίζεται το αμμοχάλικο στα
κόσκινα
        ghostRocks05.SetActive(false);
        ghostRocks1.SetActive(false);
        ghostRocks2.SetActive(false);
        ghostRocks4.SetActive(false);
        ghostRocks8.SetActive(false);
        ghostRocks16.SetActive(false);
        ghostRocks31.SetActive(false);

     }
    void OnCollisionEnter(Collision col)
    {
        collidedWith = col.gameObject;
        if (collidedWith.tag == "sieveSeism")
        {

            print(col.collider.name);

            if ( col.collider.name == "sieve05"  || col.collider.name == "sieve1"
||                 col.collider.name == "sieve2" || col.collider.name == "sieve4"  ||
                col.collider.name == "sieve8" || col.collider.name == "sieve16"  ||
                col.collider.name == "sieve31" )
            {
                showText1 = true;
                print(col.collider.name );
            }
            else
            {
                putFisrtSieveOnSiesm();
            }
        }
    }
    void putFisrtSieveOnSiesm()
    {
       if (collidedWith.name == "sieve025" )
        {
            sieve025.SetActive(false);
            ghostSieveAfterSeism025.SetActive(true);
        }
     }
    void OnGUI()
    {
        if (showText1 == true)
        {
            GUI.Box(new Rect(0, 50, Screen.width, Screen.height - 50), text1);
            if (Input.GetKeyDown(KeyCode.Escape))
            {
                showText1 = false;
            }
        }
    }
```

```
}
```

(κώδικας 11)

```
public class sieve025 : MonoBehaviour

void OnCollisionEnter(Collision col)
    {
        collidedWith = col.gameObject;
        if (collidedWith.tag == "sieveSeism")
        {
            if( col.collider.name == "sieve05")
            {
                putSecondSieveOnSiesm();
            }
        }
     }
   void putSecondSieveOnSiesm()
    {

        if (collidedWith.name == "sieve05" )
        {
            sieve05.SetActive(false);
            ghostSieveAfterSeism05.SetActive(true);
         }
    }
```

(κωδικας 12)

```
public class sieve31 : MonoBehaviour
{
    public void updateRocksBeingCooked(bool cooked)
    {
        rocksCooked = cooked;
    }
    public void updateRocksBeingWeighted(bool weighted)
    {
        rocksWeighted = weighted;
    }
    public void rocksPutOnSieve()
     {
       if (rocksWeighted == true && rocksCooked == true )
        {
            showText1 = true;
            trayHasGamer.GetComponent<tray>().gamerHasTray(true);
        }
    }
    void OnGUI()
    {
        if (showText1 == true)
        {
            GUI.Box(new Rect(0, 50, Screen.width, Screen.height - 50), text1);
```

```
            if (Input.GetKeyDown(KeyCode.Escape))
            {
                showText1 = false;            }
        }
    }
}
```

(κώδικας 13)

```
{    public void updateStatePlusSeism(bool state)
    {
        buttonOnOffSeism = state;
    }
  /*  public void receiveSecond(int sendSec)
    {
        receiveSec = sendSec;
        setSeconds(receiveSec);
    }*/
    public void receiveMinute(int sendMin)
    {
        receiveMin = sendMin;
        setMinutes(receiveMin);
    }
    public void setMinutes(int _minute) //συνάρτηση για εμφάνιση στην οθόνη των
λεπτών
    {
        minute = _minute;
      //  screenSeismMinutes.GetComponent<screenSeismMinutes>().showMinute(minute);
OXI HTAN ENERGOPOIHMENO
    }
   /*  public void setSeconds(int _second)
    {
        second = _second;
        //
screenSeismSeconds.GetComponent<screenSeismSeconds>().showSecond(second);     OXI
HTAN ENERGOPOIHMENO
    }*/
    public void addToMinutes(int minute)    //προσθήκη λεπτών
    {
       minute++;
       setMinutes(minute);
    }
    /* public void addToSeconds(int _second)   //προσθήκη δευτερολέπτων
    {
        second++;
        //  setSeconds(second);         OXI HTAN ENERGOPOIHMENO
        screenSeismSeconds.GetComponent<screenSeismSeconds>().showSecond(second);
    }*/
    void press()
    {
        if (buttonOnOffSeism == ON)
        {
           addToMinutes(minute);
           sendMinutesToSubstract(minute);
           // if (minute == maxMinutes)
           // {
```

```
            //    sendMinutesToSubstract(minute);
            //   print("OXI PERISSOTERA APO 5");
            // addToMinutes(minute);
            // setSeconds(0);
            /* if ( minute !=0)     OXI HTAN ENERGOPOIHMENO
             {                 minutesAreSet = true;
             }*/
            //  showMinite.GetComponent<TextMesh>().text = string.Concat(minute);
OXI HTAN ENERGOPOIHMENO
            /* if (minute < maxMinutes)     OXI HTAN ENERGOPOIHMENO
             {
               //  setSeconds(0);
                 addToSeconds(second);
                 //    showSecond.GetComponent<TextMesh>().text =
string.Concat(second);
             }*/
            // }
        }
       }
    /* public void sendSecondsToSubstract(int second)      //συνάρτησης για μείωση
όταν πατηθεί το -
      {                     sendSec = second;
          sendSecond.receiveSecond(sendSec);   //κλήση συνάρτης receiveSecond του
script minusKey για εισαγωγή επιλεγμένων seconds προς μείωση
      }*/
      public void sendMinutesToSubstract(int minute)
      {
          sendMin = minute;
          sendMinute.receiveMinute(sendMin);
      }
}
```

(κώδικας 14)

```
public class minusKey : MonoBehaviour
{
     public void updateStateMinusSeism(bool state)
      {
          buttonOnOffSeism = state;
      }
     /* public void receiveSecond(int sendSec)
      {
          receiveSec = sendSec;
          setSeconds(receiveSec);
      }*/
      public void receiveMinute(int sendMin)
      {
          receiveMin = sendMin;
          setMinutes(receiveMin);
      }
      public void setMinutes(int _minute) //συνάρτηση για εμφάνιση στην οθόνη
      {
          minute = _minute;
          screenSeismMinutes.GetComponent<screenSeismMinutes>().showMinute(minute);
```

```
        }
      /* public void setSeconds(int _second)
       {
           second = _second;
           screenSeismSeconds.GetComponent<screenSeismSeconds>().showSecond(second);
       }*/
       public void substractToMinutes(int minute)    //αφαίρεση λεπτών
       {
         minute--;
         setMinutes(minute);
        }
     /*  public void substractToSeconds(int _second)   //αφαίρεση δευτερολέπτων
       {
           if (_second > minSeconds)
           {
               second--;
               setSeconds(second);
           }
           if( _second == 1)
           {
               setSeconds(59);
               second--;
           }
       }*/
       void press( )
       {
           if (buttonOnOffSeism == ON)
           {
             substractToMinutes(minute);
             sendMinutesToPlus(minute);
           }
        }     /*  public void sendSecondsToPlus(int second)   //συνάρτηση για αύξηση
όταν πατηθεί το +
        {
            sendSec = second;
            sendSecond.receiveSecond(sendSec);   //κλήση συνάρτης receiveSecond του
script plusKey για εισαγωγή επιλεγμένων seconds προς αύξηση
        }*/
        public void sendMinutesToPlus(int minute)
        {
            sendMin = minute;
            sendMinute.receiveMinute(sendMin);
        }
 }
```

(κώδικας 15)

```
public class startSeism : MonoBehaviour
{
    public void updateStateSeism(bool state)     //συναρτήσεις για ανοιγμένο σείστρο,
ορισμένο χρόνο, τοποθετημένα τα κόσκινα ζυγισμένο δείγμα, ψημένο δείγμα και
τοποθετημένο δείγμα
    {
        buttonOnOffSeism = state;
    }
    public void showTime(int Intime)
```

```
    {
        timeIsSet = Intime;
        if ((timeIsSet >= 3) && (timeIsSet <= 5))
        {
            rightMinutes = true;
        }
        else
        {
            rightMinutes = false;
        }
    }
    public void updateRocksBeingCooked(bool cooked)
    {
        rocksCooked = cooked;
    }
    public void updateRocksBeingWeighted(bool weighted)
    {
        rocksWeighed = weighted;
    }
    public void updateRocksOnSieve31(bool stateRocks)
    {
        rocksOnSieve31 = stateRocks;
     }
    // Use this for initialization
    void Start()
    {
        ghostRocks025.SetActive(false);      //να μην εμφανίζεται το αμμοχάλικο στα
κόσκινα
        ghostRocks05.SetActive(false);
        ghostRocks1.SetActive(false);
        ghostRocks2.SetActive(false);
        ghostRocks4.SetActive(false);
        ghostRocks8.SetActive(false);
        ghostRocks16.SetActive(false);
        ghostRocks31.SetActive(false);
    }
    public void startShaking()
    {
        if (buttonOnOffSeism == ON && rightMinutes == true && rocksOnSieve31 == true)
        {

            showText1 = true;        //μήνυμα για αρχή ψησίματος
        }
        if (buttonOnOffSeism == ON && rightMinutes == false && rocksOnSieve31 == true)
        {
            showText2 = true;        //μήνυμα για σωστό χρόνο κοσκινίσματος
        }
        if (buttonOnOffSeism == ON && rightMinutes == true && rocksOnSieve31 == false)
        {
            showText3 = true;        // μήνυμα για τοποθέτηση του δείγματος
        }
        if (buttonOnOffSeism == OFF)
        {
            showText4 = true;        //μήνυμα για άνοιγμα σείστρου
        }
    }
    void OnGUI()
```

```
    {
        if (showText1 == true)
        {
            if (timeIsSet == 3)
            {
                myTimerMinimum -= Time.deltaTime * 20;// * 20 για επιτάχυνση του
χρόνου
                int seconds = (int)(myTimerMinimum % 60);
                int minutes = (int)(myTimerMinimum / 60) % 60;
                int hours = 0;
                string timerString = string.Format("{0:0} : {1:00} : {2:00}", hours,
minutes, seconds);
                text1 = ("Χρόνος ψησίματος: ") + timerString;
            }
            showAllRocks();

            if (timeIsSet == 4)
            {
                myTimerMedium -= Time.deltaTime * 20;
                int seconds = (int)(myTimerMedium % 60);
                int minutes = (int)(myTimerMedium / 60) % 60;
                int hours = 0;
                string timerString = string.Format("{0:0} : {1:00} : {2:00}", hours,
minutes, seconds);
                text1 = ("Χρόνος ψησίματος: ") + timerString;
            }
            showAllRocks();
            if (timeIsSet == 5)
            {
                myTimerMaximun -= Time.deltaTime * 20;
                int seconds = (int)(myTimerMaximun % 60);
                int minutes = (int)(myTimerMaximun / 60) % 60;
                int hours = 0;
                string timerString = string.Format("{0:0} : {1:00} : {2:00}", hours,
minutes, seconds);
                text1 = ("Χρόνος ψησίματος: ") + timerString;
            }
            showAllRocks();
            if (myTimerMinimum < endTimeMini)
            {
                showText1 = false;
                pressForEndClock();
            }
            if (myTimerMedium < endTimeMedi)
            {
                showText1 = false;
                pressForEndClock();
            }
            if (myTimerMaximun < endTimeMaxi)
            {
                showText1 = false;
                pressForEndClock();
            }
            GUI.Box(new Rect(0, 50, Screen.width, Screen.height - 50), text1);
            if (Input.GetKeyDown(KeyCode.Escape))
            {
                showText1 = false;
```

```
            }
        }
        if (showText2 == true)
        {
            GUI.Box(new Rect(0, 50, Screen.width, Screen.height - 50), text2);
            if (Input.GetKeyDown(KeyCode.Escape))
            {
                showText2 = false;
            }
        }
        if (showText3 == true)
        {
            GUI.Box(new Rect(0, 50, Screen.width, Screen.height - 50), text3);
            if (Input.GetKeyDown(KeyCode.Escape))
            {
                showText3 = false;
            }
        }
        if (showText4 == true)
        {
            GUI.Box(new Rect(0, 50, Screen.width, Screen.height - 50), text4);
            if (Input.GetKeyDown(KeyCode.Escape))
            {
                showText4 = false;
            }
        }

    }
    void press()
    {
        startShaking();
    }
    public void pressForEndClock()
    {
        screenSeismMinutes.GetComponent<screenSeismMinutes>().resetClock();
        //  screenSeismSeconds.GetComponent<screenSeismSeconds>().resetClock();
    }
    public void showAllRocks()
    {
        ghostRocks025.SetActive(true);          //να  εμφανίζεται το αμμοχάλικο στα
κόσκινα
        ghostRocks05.SetActive(true);
        ghostRocks1.SetActive(true);
        ghostRocks2.SetActive(true);
        ghostRocks4.SetActive(true);
        ghostRocks8.SetActive(true);
        ghostRocks16.SetActive(true);
        ghostRocks31.SetActive(false);
       chooseSieve(sieve31);
    }
}
```

(κώδικας 16)

```
public class tray : MonoBehaviour
{
```

```
    void Start()
    {
        trayWeight = 50;
        grossWeight = 3050;
        setRealWeight();
        ghostRocks025after.SetActive(false);
        ghostRocks05after.SetActive(false);
        ghostRocks1after.SetActive(false);
        ghostRocks2after.SetActive(false);
        ghostRocks4after.SetActive(false);
        ghostRocks8after.SetActive(false);
        ghostRocks16after.SetActive(false);
        forTrayWeight = plate.GetComponent<plateScale>();
        forTrayCook = startButton.GetComponent<startOven>();
        forRocksGhostSieve31 = seismButton.GetComponent<startSeism>();
      }
    public void putRocksOnSieve(bool rocksOn)
    {
        rocksOnSieve31 = rocksOn;
        forRocksGhostSieve31.updateRocksOnSieve31(rocksOnSieve31);

    }
    public void updateRocksBeingCooked(bool cooked)
    {
        rocksCooked = cooked;
    }
    public void updateRocksBeingWeighted(bool weighted)
    {
        rocksWeighted = weighted;
    }
    public void updateNewRocksBeingWeighted( bool weighted)
    {
        newRocksAreWeighted = weighted;
    }
    public void setRealWeight()     //υπολογισμός καθαρού βάρους ωστε να εμφανιστεί
στην οθόνη
    {
        rocksWeight = grossWeight - trayWeight;
        forTrayWeight.updateRealWeight(rocksWeight);          //κλήση     συνάρτησης
upDateRealWeight της κλάσης plateScale με όρισμα το καθαρό βάρος
    }
    public void resetWeight( )  //μηδενισμός βάρους
    {
        forTrayResetWeight.updateRealWeight(0f);
    }
    public void gamerHasTray(bool gamersHands)
    {
       gamerTray = gamersHands;
    }
    public void gamerHasTrayForBucket(bool gamersHandsBucket)
    {
        gamerBucket = gamersHandsBucket;

    }
    public void chooseSieveForBucket(GameObject _current)
    {
        currentSieve = _current;
```

```
        if (_current == sieve16)
        {
            gamerHasTrayForBucket(true);
        }
       else if (_current == sieve8)
        {
          gamerHasTrayForBucket(true);
        }
        else if (_current == sieve4)
        {
           gamerHasTrayForBucket(true);
        }
        else if (_current == sieve2)
        {
           gamerHasTrayForBucket(true);
        }
        else if (_current == sieve1)
        {
           gamerHasTrayForBucket(true);
        }
        else if (_current == sieve05)
        {
           gamerHasTrayForBucket(true);
        }
        else if (_current == sieve025)
        {
           gamerHasTrayForBucket(true);
        }
    }
    public void detachFromParent()      // για να πέσει το αρχικό δείγμα από τον δίσκο
στο κόσκινο
    {
        traysRocks.transform.parent = null;
        traysRocks.GetComponent<Rigidbody>().useGravity = true;
        traysRocks.SetActive(false);
        sievesRocks.SetActive(true);
        putRocksOnSieve(true);
     }
    public void detach16ForBucket() // για να πέσει το δείγμα από τον δίσκο στον κουβά
    {
        resetWeight();
        ghostRocks16after.transform.parent = null;
        ghostRocks16after.GetComponent<Rigidbody>().useGravity = true;
        ghostRocks16after.SetActive(false);
        rocks16OnBucket.SetActive(true);
        nextSieve8.GetComponent<ghosts>().chooseNumberSieve(sieve8);
    }
    public void detach8ForBucket()
    {
        resetWeight();
        ghostRocks8after.transform.parent = null;
        ghostRocks8after.GetComponent<Rigidbody>().useGravity = true;
        ghostRocks8after.SetActive(false);
        rocks8OnBucket.SetActive(true);
        nextSieve4.GetComponent<ghosts>().chooseNumberSieve(sieve4);
    }
    public void detach4ForBucket()
```

```
    {
        resetWeight();
        ghostRocks4after.transform.parent = null;
        ghostRocks4after.GetComponent<Rigidbody>().useGravity = true;
        ghostRocks4after.SetActive(false);
        rocks4OnBucket.SetActive(true);
        nextSieve2.GetComponent<ghosts>().chooseNumberSieve(sieve2);
    }
    public void detach2ForBucket()
    {
        resetWeight();
        ghostRocks2after.transform.parent = null;
        ghostRocks2after.GetComponent<Rigidbody>().useGravity = true;
        ghostRocks2after.SetActive(false);
        rocks2OnBucket.SetActive(true);
        nextSieve1.GetComponent<ghosts>().chooseNumberSieve(sieve1);
    }
    public void detach1ForBucket()
    {
        ghostRocks1after.transform.parent = null;
        ghostRocks1after.GetComponent<Rigidbody>().useGravity = true;
        ghostRocks1after.SetActive(false);
        rocks1OnBucket.SetActive(true);
        nextSieve05.GetComponent<ghosts>().chooseNumberSieve(sieve05);
    }
    public void detach05ForBucket()
    {
        resetWeight();
        ghostRocks05after.transform.parent = null;
        ghostRocks05after.GetComponent<Rigidbody>().useGravity = true;
        ghostRocks05after.SetActive(false);
        rocks05OnBucket.SetActive(true);
        nextSieve025.GetComponent<ghosts>().chooseNumberSieve(sieve025);

    }
    public void detach025ForBucket()
    {
        resetWeight();
        ghostRocks025after.transform.parent = null;
        ghostRocks025after.GetComponent<Rigidbody>().useGravity = true;
        ghostRocks025after.SetActive(false);
        rocks025OnBucket.SetActive(true);
    }
   public IEnumerator rotateForBucket()
    {

        rotationBucket = rotateSpeed * Time.deltaTime;
        rotatingBucket = true;
        while (rotationdownBucket > rotationBucket)
        {
            rotationdownBucket -= rotationBucket;
            yield return new WaitForSeconds(0.01F);
            transform.Rotate(0, 0, rotationBucket);
        }
        if (currentSieve == sieve16)
        {
          detach16ForBucket();
```

```
        }
       else if (currentSieve == sieve8)
        {
            detach8ForBucket();
        }
       else if (currentSieve == sieve4)
        {
            detach4ForBucket();
       }
        else if (currentSieve == sieve2)
        {
           detach2ForBucket();
        }
        else if (currentSieve == sieve1)
        {
           detach1ForBucket();
        }
        else if (currentSieve == sieve05)
        {
           detach05ForBucket();
        }
        else if (currentSieve == sieve025)
        {
           detach025ForBucket();
        }
        while (rotationupBucket > rotationBucket)
        {
            rotationupBucket -= rotationBucket;
            yield return new WaitForSeconds (0.01F);
            transform.Rotate(0, 0, -rotationBucket);
        }
        rotatingBucket = false;
        gamerBucket = false;
        resetRotationForBucket ();
    }
    public IEnumerator rotate()
    {
        rotation = rotateSpeed * Time.deltaTime;
        rotating = true;

        while(rotationdown > rotation)
        {
            rotationdown -= rotation;
            yield return new WaitForSeconds(0.01F);
            transform.Rotate(0, 0, rotation);
        }
        detachFromParent();              // για να πέσει το δείγμα από τον δίσκο στο
κόσκινο 31.5
        while(rotationup > rotation)
        {
            rotationup -= rotation;
            yield return new WaitForSeconds (0.01F);
            transform.Rotate(0, 0, -rotation);
         }
        rotating = false;
        gamerHasTray(false);
        resetRotate();
```

```
    }
    public void resetRotate()
    {
        rotationdown = 40;
        rotationup = 40;
        rotation = 0;
    }
    public void resetRotationForBucket ()
    {
        rotationdownBucket = 40;
        rotationBucket = 0;
        rotationupBucket = 40;
}
    void Update()
    {
        if (gamerTray == true && rotating == false && Input.GetKey(KeyCode.Space))
        {
            StartCoroutine(rotate());
        }
        if    (    gamerBucket    ==    true    &&    rotatingBucket    ==    false    &&
Input.GetKey(KeyCode.Space) )
        {
            StartCoroutine(rotateForBucket());
        }
    }


}
```

(κώδικας 17)

```
public class ghosts : MonoBehaviour
{
     // Use this for initialization
    void Start()
    {

    }
    public void gamerHasSieve(bool gamersHands)
    {
        gamerSieve = gamersHands;
    }
    public void chooseNumberSieve(GameObject _current)  // για να πέσει το δείγμα
από το αντίστοιχο κόσκινο στον δίσκο
    {
        currentSieve = _current;
        if ( _current == sieve31)
        {
            gamerHasSieve(false);

        }
        else if (_current == sieve16)
        {
            gamerHasSieve(true);
        }
```

```
        else if(_current == sieve8)
        {
            gamerHasSieve(true);
        }
        else if(_current == sieve4)
        {
            gamerHasSieve(true);
        }
        else if (_current == sieve2)
        {

            gamerHasSieve(true);
        }
        else if(_current == sieve1)
        {
            gamerHasSieve(true);
        }
        else if(_current == sieve05)
        {
            gamerHasSieve(true);
        }
        else if (_current == sieve025)
        {
            gamerHasSieve(true);
        }
    }
  public void detachFromParent16()        // για να πέσει το δείγμα από το κόσκινο
16 στον δίσκο
        sievesRocks16.transform.parent = null;
        sievesRocks16.GetComponent<Rigidbody>().useGravity = true;
        sievesRocks16.SetActive(false);
        ghostRocks16after.SetActive(true);
        weight16.updateRealWeight(rocksWeight16);
    }
    public void detachFromParent8()         // για να πέσει το δείγμα από το κόσκινο
8 στον δίσκο
        sievesRocks8.transform.parent = null;
        sievesRocks8.GetComponent<Rigidbody>().useGravity = true;
        sievesRocks8.SetActive(false);
        ghostRocks8after.SetActive(true);
        weight8.updateRealWeight(rocksWeight8);
    }
    public void detachFromParent4()
        sievesRocks4.transform.parent = null;
        sievesRocks4.GetComponent<Rigidbody>().useGravity = true;
        sievesRocks4.SetActive(false);
        ghostRocks4after.SetActive(true);
        weight4.updateRealWeight(rocksWeight4);
    }
    public void detachFromParent2()
        sievesRocks2.transform.parent = null;
        sievesRocks2.GetComponent<Rigidbody>().useGravity = true;
        sievesRocks2.SetActive(false);
        ghostRocks2after.SetActive(true);
        weight2.updateRealWeight(rocksWeight2);
    }
    public void detachFromParent1()
```

```
        sievesRocks1.transform.parent = null;
        sievesRocks1.GetComponent<Rigidbody>().useGravity = true;
        sievesRocks1.SetActive(false);
        ghostRocks1after.SetActive(true);
        weight1.updateRealWeight(rocksWeight1);
    }
    public void detachFromParent05()
        sievesRocks05.transform.parent = null;
        sievesRocks05.GetComponent<Rigidbody>().useGravity = true;
        sievesRocks05.SetActive(false);
        ghostRocks05after.SetActive(true);
        weight05.updateRealWeight(rocksWeight05);
    }
    public void detachFromParent025()
        sievesRocks025.transform.parent = null;
        sievesRocks025.GetComponent<Rigidbody>().useGravity = true;
        sievesRocks025.SetActive(false);
        ghostRocks025after.SetActive(true);
        weight025.updateRealWeight(rocksWeight025);
    }
public IEnumerator rotate()
    {
        rotationSieve = rotateSpeed * Time.deltaTime;
        rotating = true;
        while (rotationdownSieve > rotationSieve)
        {
            rotationdownSieve -= rotationSieve;
            yield return new WaitForSeconds(0.01F);
            transform.Rotate(0, rotationSieve, 0);
        }
        if( currentSieve == sieve16)
        {
            detachFromParent16();
        }
        else if (currentSieve == sieve8)
        {
           detachFromParent8();
        }
        else if (currentSieve == sieve4)
        {
           detachFromParent4();
        }
        else if(currentSieve == sieve2)
        {
           detachFromParent2();
        }
        else if (currentSieve == sieve1)
        {
           detachFromParent1();
        }
        else if (currentSieve == sieve05)
        {
           detachFromParent05();
        }
        else if (currentSieve == sieve025)
        {
           detachFromParent025();
```

```
        }
        while (rotationup > rotationSieve)
        {
            rotationup -= rotationSieve;
            yield return new WaitForSeconds(0.01F);
            transform.Rotate(0, -rotationSieve, 0);
        }
        rotating = false;
        gamerHasSieve(false);
    }
    void Update()
    {
        if (gamerSieve == true && rotating == false && Input.GetKey(KeyCode.Space))
        {
             StartCoroutine(rotate());
        }
    }
}
```

(κώδικας 18)

Υπεύθυνη Δήλωση Συγγραφέα:

Δηλώνω ρητά ότι, σύμφωνα με το άρθρο 8 του Ν.1599/1986, η παρούσα εργασία αποτελεί αποκλειστικά προϊόν προσωπικής μου εργασίας, δεν προσβάλλει κάθε μορφής δικαιώματα διανοητικής ιδιοκτησίας, προσωπικότητας και προσωπικών δεδομένων τρίτων, δεν περιέχει έργα/εισφορές τρίτων για τα οποία απαιτείται άδεια των δημιουργών/δικαιούχων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον και πληρούν τους κανόνες της επιστημονικής παράθεσης.