
**Πανεπιστήμιο Ιωαννίνων**

**Πολυτεχνική Σχολή**

**Τμήμα Μηχανικών Η/Υ και Πληροφορικής**

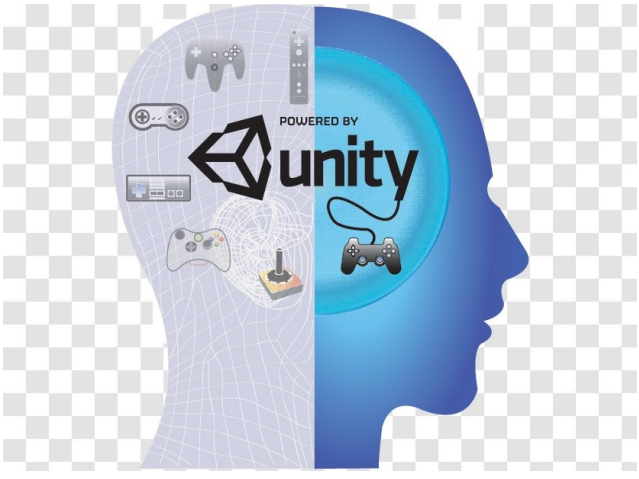
**Προπτυχιακό Μάθημα: «Γραφικά Υπολογιστών και Συστήματα Αλληλεπίδρασης»**

**Δεύτερη Προγραμματιστική Άσκηση**

**<u>Όνομα Φοιτητών – Α.Μ.:</u>**

**Παναγιώτης Ντουνέτας - 2781**

**Γεώργιος Κρομμύδας – 3260**




*ΙΩΑΝΝΙΝΑ,*

*2020*

# Μέρος-1ο ~ Γενική Περιγραφή Άσκησης:

Στόχος της άσκησης ήταν η υλοποίηση ενός 3D-Κύβου , όπου στο εσωτερικό του βρίσκεται μία κόκκινη σφαίρα η οποία έχει την δυνατότητα να δέχεται texture appliance. Επίσης ζητείται κίνηση FP (first person) κάμερας , καθώς και κίνηση της σφαίρας. Τέλος, με εντολή από το πληκτρολόγιο δημιουργούνται τυχαία αντικείμενα στο εσωτερικό του κύβου και κινούνται επίσης τυχαία στο χώρο αυτό.

Έχουν υλοποιηθεί όλα τα βασικά ερωτήματα της άσκησης (τυχόν παραλήψεις-σχόλια-διορθώσεις θα αναφερθούν λεπτομερώς στο τέλος) και τα Bonus ερωτήματα 3 και 5.

# Μέρος-2ο ~ Υλοποίηση-Στάδια Υλοποίησης:

**Ερώτημα (i):**

Η τελική ανάλυση του παραγόμενου αρχείου (Συγκρουόμενα.exe) είναι 1024x768 σε Window Mode. Ο κύβος χρωματίζεται κανονικά με 0.5 διαφάνεια (έχει χρησιμοποιηθεί fade και όχι transparency στο material του κύβου(γιατί το πρότειναν αρκετοί στο Internet) και οι rgba τιμές έχουν αλλαχθεί σε range 0 έως 1 από 0 έως 255). Το λευκό background έχει γίνει set από το πεδίο της κάμερας Background.

**Ερώτημα (ii):**

Με το πάτημα του spacebar εμφανίζονται από το (0,0,0) τα νέα αντικείμενα (Το πως ακριβώς γίνεται αυτό και από που πραγματικά εμφανίζονται θα εξηγηθεί παρακάτω) με τυχαίο χρώμα και τυχαία πορεία κίνησης. Επίσης οι πηγές φωτισμού (3) τοποθετήθηκαν πειραματικά ώστε να φαίνεται ο χρωματισμός του κύβου και των αντικειμένων στο εσωτερικό του.

**Ερώτημα (iii):**

Η SPH (κόκκινη σφαίρα) κινείται με τα ζητούμενα πλήκτρα στο εσωτερικό του κύβου και αντιδρά στις συγκρούσεις.

**Ερώτημα (iv):**

Η FP (first person) κάμερα κινείται στο χώρο με τα πλήκτρα W A S D και Q E για αυξομείωση ύψους.

**Ερώτημα (v):**

Φορτώνεται στη σφαίρα μία εικόνα μορφής μπλε κύματος που βρήκαμε από το google.

**Bonus Ερώτημα (iii):**

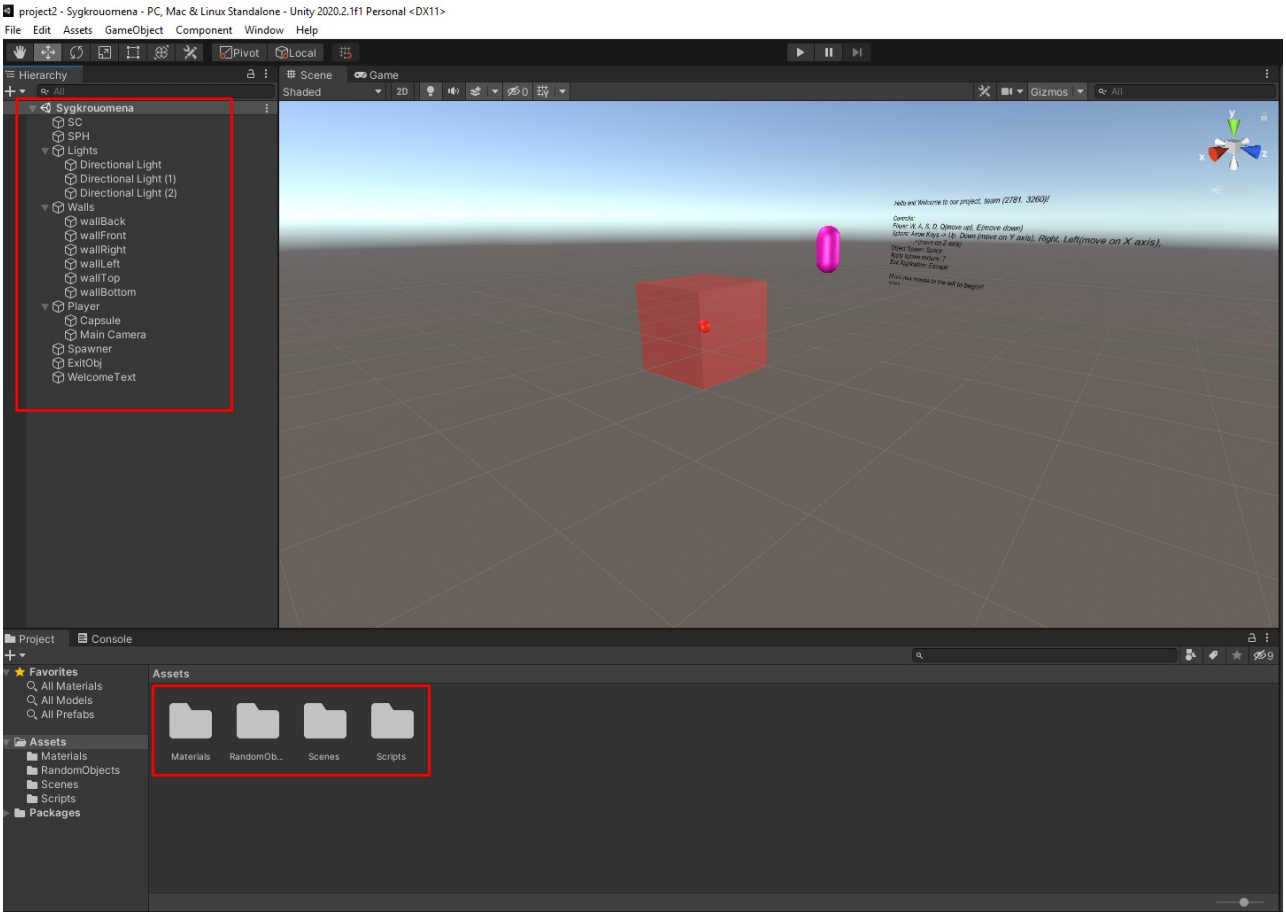
Τα νέα αντικείμενα αντιδρούν μεταξύ τους σωστά και έχουν bounciness.

**Bonus Ερώτημα (v):**

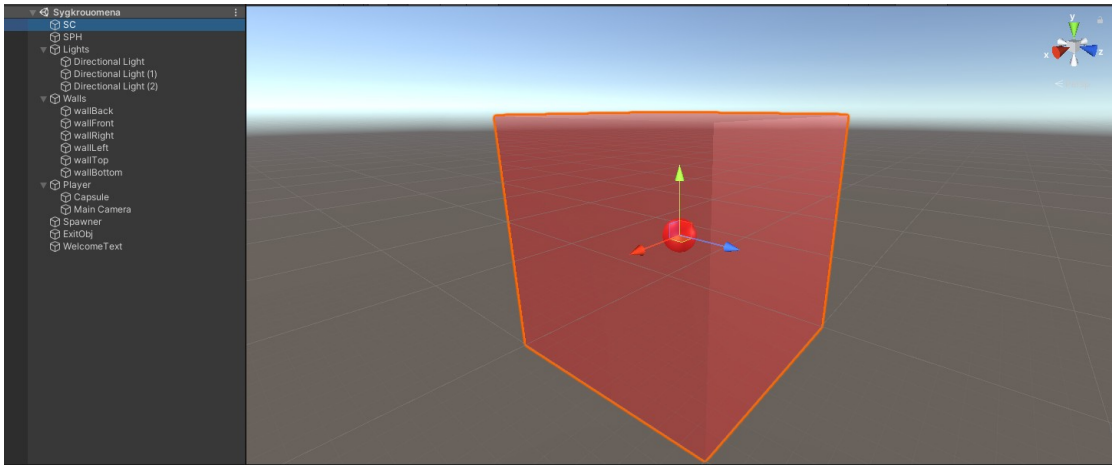
Η κάψουλα που είναι ο Player έχει τη δυνατότητα να συγκρούεται κανονικά με όλα τα αντικείμενα ή να απενεργοποιηθεί αυτό το στοιχείο ( το πως θα γίνει αυτό θα αναλυθεί παρακάτω).

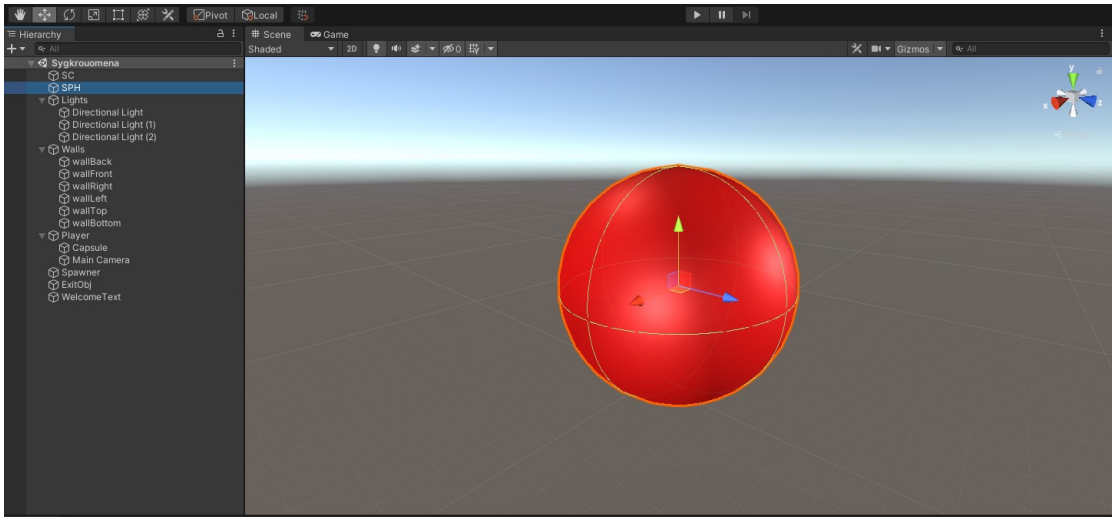
# Μέρος-3ο ~ Το πρόγραμμα, Scripts και ανάλυση λειτουργίας

Παρακάτω φαίνονται όλα τα περιεχόμενα της εργασίας:



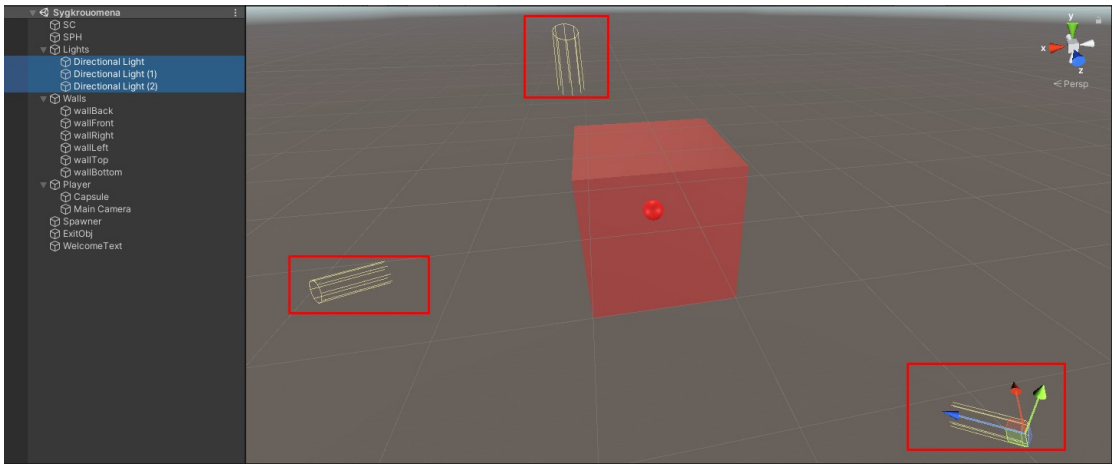

**SC:** Είναι ο ζητούμενος κύβος που οριοθετεί την δράση των παραγόμενων αντικειμένων και της σφαίρας. Το αρχικό του χρώμα είναι ένα διαφανές κόκκινο που έχει εφαρμοστεί μέσω material, το οποίο material στη συνέχεια επεξεργαζόμαστε για την απόδοση τυχαίων χρωμάτων στον κύβο κάθε φορά.





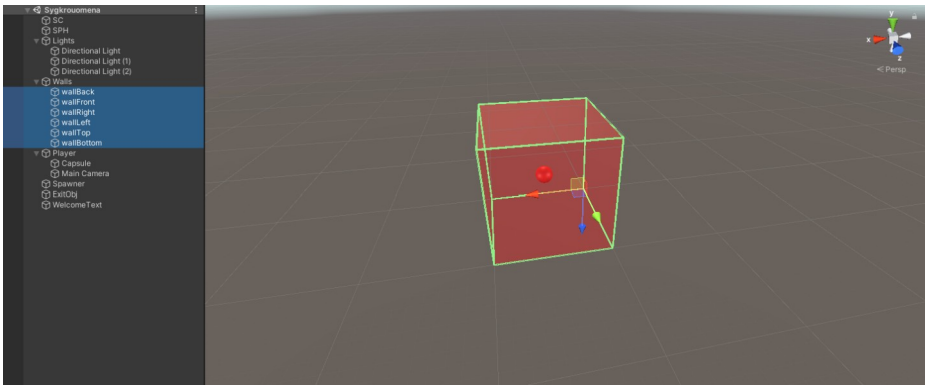

**SPH:** Η ζητούμενη κόκκινη σφαίρα που κινείται στο χώρο. Όπως και στον κύβο, έτσι και στη σφαίρα, εφαρμόζεται ένα κόκκινο χρώμα με τη διαφορά ότι είναι έντονο και δεν μεταβάλλεται το χρώμα αργότερα αλλά ολόκληρο το material.

**Lights:** Τα directional lights είναι οι φωτισμοί που έχουν τοποθετηθεί τελείως πειραματικά στο χώρο από διάφορες δοκιμές (Παρατήρηση: Κατά τη διάρκεια της εκτέλεσης αν ο παίκτης μεταβεί και "κοιτάξει" τον κύβο από το κάτω μέρος, η πλευρά μαυρίζει λόγω έλλειψης φωτισμού και ο λόγος που δεν προστέθηκε παραπάνω πηγή είναι σαν λογική ότι η κάμερα σε εκείνο το σημείο βρίσκεται "κάτω από το έδαφος").

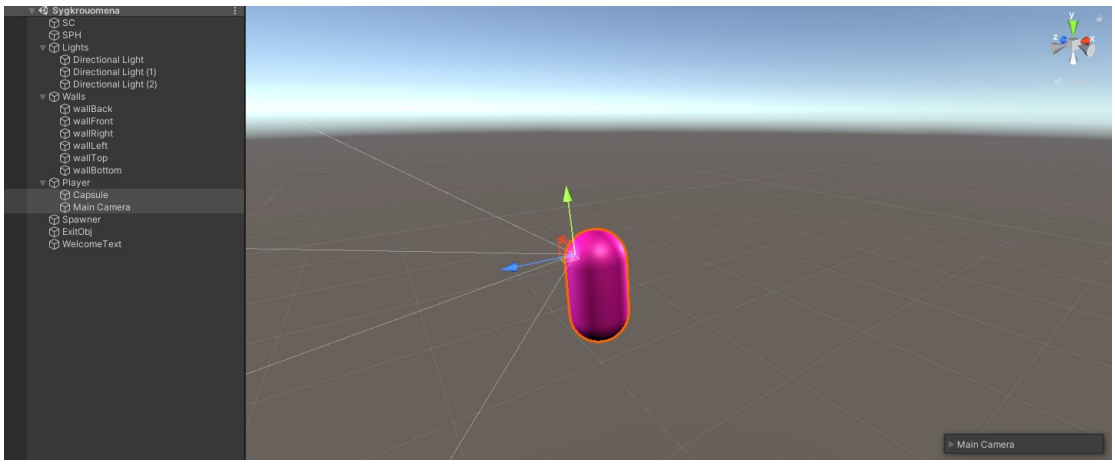



**Walls**: Αποτελούν τους colliders του κύβου. Ο απλός default collider του κύβου ανίχνευε τις συγκρούσεις από έξω προς τα μέσα ενώ εμείς θέλουμε και τις 2 περιπτώσεις. Αφαιρέσαμε τον αρχικό collider και το αντικαταστήσαμε με 6 νέους box colliders ανεξάρτητους από τον αρχικό μας κύβο. Έχουν πάχος 1 (στο Box Collider→ Size→ Z=1). Ο πρώτος που δημιουργήθηκε ήταν ο wallBack και οι υπόλοιποι 5 είναι

αντίγραφα αυτού με σωστές αποστάσεις και περιστροφές. (Παρατήρηση: εισέρχονται κατά 0.5 στο εσωτερικό του κύβου).




**Player:** Αναπαριστά τον παίκτη και πάνω του είναι η κάμερα. (Παρατήρηση: η κάμερα βρίσκεται στο ύψος του κεφαλιού και κατά προσέγγιση στη θέση των ματιών για να είναι όσο πιο αληθοφανές. Επίσης η θέση τους έγινε πειραματικά μέσω των εργαλείων του GameObject→ Align with View και Move To View.)

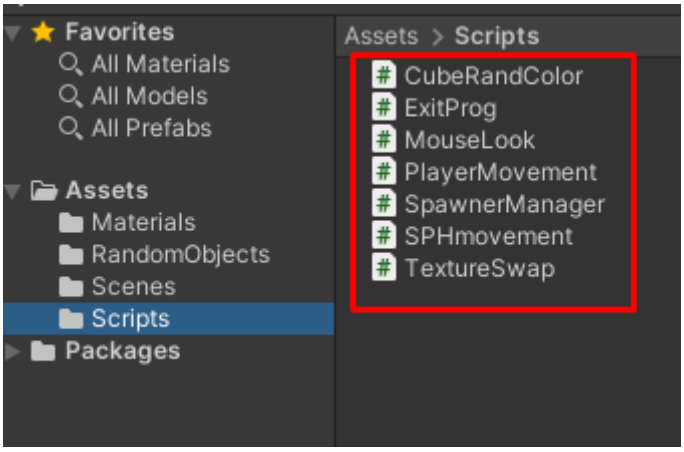



**Spawner:** Κενό αντικείμενο το οποίο δημιουργήθηκε για την επικόλληση του script **SpawnerManager:** για τα μικρά νέα αντικείμενα.

**ExitObj:** Κενό αντικείμενο που περιέχει το script για το τερματισμό.

**WelcomeText:** Ενα σύντομο κείμενο εντός του προγράμματος.

## Scripts:




**CubeRandColor:** Περιέχει τον κώδικα για την παραγωγή τυχαίου χρώματος για τον κύβο. Υλοποιούνται όλα στο 1ο frame (συνάρτηση **Start**) εφόσον δεν μας απασχολεί ο κύβος και το χρώμα του δεν μεταβάλλονται κατά της διάρκεια του προγράμματος. Η συνάρτηση δέχεται 3 τυχαία χρώματα και τα εφαρμόζει στο πεδίο **Color** του **CubeMaterial** σαν τα RGB ενώ το **Alpha** μένει σταθερό στο 0.5. Ύστερα το material στο **materials[0]** που είναι το πεδίο **Materials** του του κύβου.

**ExitProg:** Πατώντας Escape τερματίζει την εφαρμογή ( δηλαδή το παραγόμενο .exe).

**MouseLook:** Το αρχείο που είναι υπεύθυνο για την κίνηση της κάμερας μέσω του ποντικιού. Κάνουμε expose στην Unity το πεδίο **mouseSens** ώστε να αλλάζουμε πειραματικά την ταχύτητα χωρίς να ανοίγουμε κάθε φορά το script. Στην **Start** τοποθετούμε τον κέρσορα στο κέντρο της οθόνης και τον εξαφανίζουμε. Μετά, στην **Update** (για κάθε frame), αρχικοποιούνται οι συναρτήσεις που δέχονται το **input** από το ποντίκι για οριζόντια και κάθετη κίνηση. Επίσης αυτό πολλαπλασιάζεται με 2 πράγματα:

**1.** Την ταχύτητα του ποντικιού (γνωστό και ως sensitivity), ώστε να δίνεται η επιθυμητή ταχύτητα περιστροφής προς τις οριζόντιες κατευθύνσεις.

**2**. Την συνάρτηση της Unity **Time.deltaTime**, ώστε να μην επηρεάζεται η ταχύτητα του ποντικιού από τα υψηλά ή χαμηλά frames που μπορεί να έχει η εφαρμογή (Λίγα frames → χαμηλή ταχύτητα και αντίστροφα).

Η μέθοδος **Mathf.Clamp(xRotation, -90f, 90f)** εξασφαλίζει ότι η κάμερα έχει περιθώριο κίνησης από 90 έως -90 μοίρες στον άξονα y'y με την λογική ότι ο παίκτης μπορεί να κοιτάξει από τον ουρανό ως το έδαφος, και η κάμερα δεν εκτελεί πλήρη περιστροφή στον y'y. Παραθέτω το βίντεο που βοήθησε στην κατασκευή αυτού του αρχείου (https://www.youtube.com/watch?v=_QajrabyTJc). (Παρατήρηση: Ο παίκτης κρατώντας πατημένο το W αλλάζει κανονικά κατεύθυνση κουνώντας το ποντίκι αριστερά – δεξιά όχι όμως προς τα κάτω ή πάνω.)

**PlayerMovement:** Εκθέτουμε στη Unity το πεδίο της ταχύτητας του παίκτη για δοκιμαστικούς λόγους. Ύστερα, στο 1ο frame συνδέουμε τον **CharacterController** του **Player** με μια μεταβλητή (**plController**) για να χρησιμοποιηθεί στο τέλος για την απόδοση του μετασχηματισμού σε κάθε frame. Αναθέτουμε πλήκτρα κίνησης τα οποία ουσιαστικά δίνουν "κατεύθυνση" στο αντικείμενο μετασχηματισμού (της εντολής Move).

**SpawnManager:** Εκθέτουμε τον πίνακα **spwdObjs** που περιέχει τα αντικείμενα που θα διαλέγονται τυχαία για δημιουργία στο εσωτερικό του κύβου. Ο σκοπός αυτή τη φορά είναι να δώσουμε, έστω και σαν hardcoding μέγεθος 3 στον πίνακα και σε κάθε θέση να κάνουμε drag and drop από την Unity και τον φάκελο **RandomObjects** τα αντικείμενα ένα σε κάθε κενή θέση του **spwdObjs**. Οπότε στο πάτημα του Spacebar έχουμε:

→ Ένας τυχαίος αριθμός (πεδίο **index**) από το 0 έως το 2 ( στην περίπτωση της άσκησης) που θα χρειαστεί λίγο παρακάτω σαν όρισμα.

→ Μέσω του **Vector3 randScale** , δημιουργία τυχαίου μεγέθους κάποιου αντικειμένου.

Τώρα στην μεταβλητή obj αποθηκεύεται το νέο αντικείμενο σαν:

**Instantiate(spwdObjs[index], startpos , Quaternion.identity)**

όπου: **spwdobjs[index]** → Διαλέγει κύβο/κύλινδρο/σφαίρα

**startpos** → Η θέση που θα βγει το αντικείμενο ορίζεται από το μέγεθος του και από μια ελάχιστη απομάκρυνση από τον **collider** του κύβου ώστε να μην αρχικοποιείται κανένα σχήμα και να μην εφάπτεται στον κύβο διότι είδαμε ότι μπορεί να προκαλέσει προβλήματα.

**Quaternion.identity** → Το αντικείμενο δεν θα κάνει αρχική περιστροφή στον χώρο.

Εφόσον το αντικείμενο είναι έτοιμο το κάνουμε **resize (transform)** στο σωστό **scale(μέγεθος).**

Μετά του αποδίδουμε τυχαίο χρώμα με παρόμοια διαδικασία όπως και στον μεγάλο κύβο της σκηνής. Τέλος , δημιουργούμε ένα **vector3** το οποίο κρατάει την διεύθυνση κίνησης, η οποία δημιουργείται τυχαία. Στα 3 τυχαία αντικείμενα δώσαμε **Rigidbody** ώστε να περνάμε εκεί την ταχύτητα στο πεδίο velocity. Έτσι περνάμε στο **Rigidbody** το **velocity** του αντικειμένου, το αποτέλεσμα της ταχύτητας επί την διεύθυνση.

**SPHmovement:** Παρόμοια με την κίνηση του παίκτη που εξαρτόμαστε από τον **CharacterController**, αυτή τη φορά με μικρές προσθήκες. Σε αυτή την περίπτωση κάνουμε **camera transformation** ώστε από όποια πλευρά και αν κοιτάξουμε τον κύβο η σφαίρα κινείται σωστά με τα ίδια κουμπιά στις σωστές διευθύνσεις. Αντίθετα χωρίς αυτό, για παράδειγμα αν κοιτούσαμε τον κύβο από την πίσω όψη τα πλήκτρα θα έδιναν σωστό **Input** αλλά οπτικά λάθος αποτέλεσμα διότι η σφαίρα θα έκανε τις ανάποδες κινήσεις από ότι αν κοιτούσαμε την μπροστινή όψη.

**TextureSwap:** Λειτουργεί σαν "διακόπτης on/off" για το texture της σφαίρας. Η Boolean τιμή μας επιτρέπει σε κάθε frame να ελέγχουμε αν πατήθηκε το πλήκτρο για την αλλαγή του **material**. Το simple παραπέμπει στο αρχικό κόκκινο χρώμα, ενώ το **textured** στην προσθήκη.

## Assets:

Όλα τα περιεχόμενα του προγράμματος βρίσκονται ταξινομημένα ανάλογα με τη χρήση/είδος τους σε υποφακέλους για καλύτερη διαχείρηση.

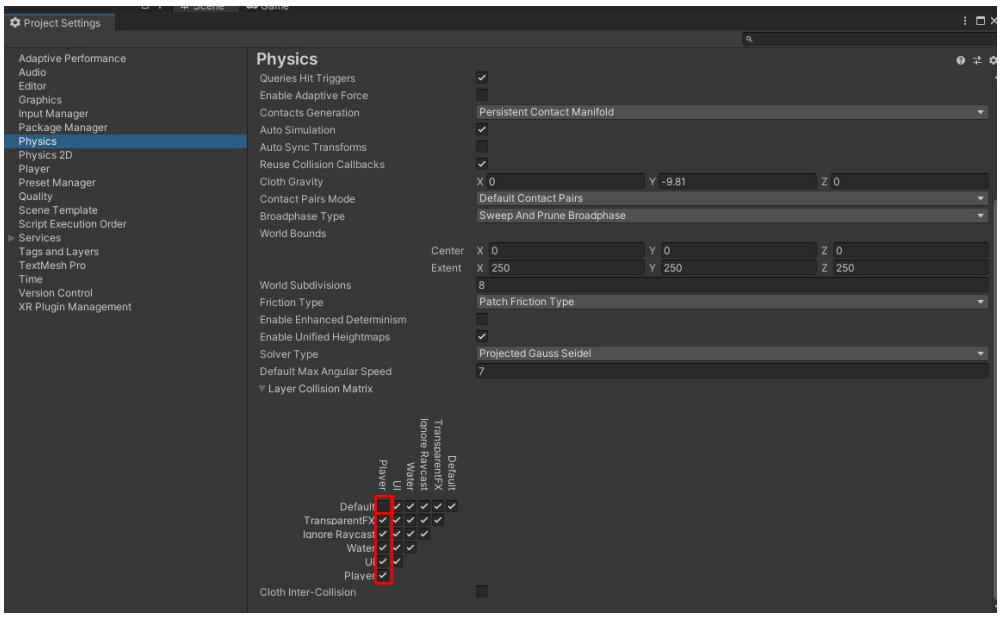
## Μέρος-4ο ~ Παρατηρήσεις-Σχόλια-Ειδικές Συνθήκες Λειτουργίας:

- **Παρατήρηση**: Όταν ο παίκτης εισέρχεται στον κύβο, ο τελευταίος εξαφανίζεται. Ξέρουμε ότι αυτό οφείλεται στα normals που αντιστρέφονται.

- **Παρατήρηση**: Η SPH (σφαίρα), όταν υπάρχουν πολλά αντικείμενα στο χώρο ενώ συγκρούεται κανονικά με αυτά όπως και τα αντικείμενα μεταξύ τους, μερικές φορές η σφαίρα μετατοπίζεται εξ αιτίας της σύγκρουσης. Είχαμε δει μια λύση η οποία ήταν να αυξήσουμε κατά πάρα πολύ τη μάζα της , αλλά δεν διόρθωσε το πρόβλημα.

- **Παρατήρηση**: Ξεκινώντας την εφαρμογή η κάμερα χάνει το αρχικό της Point Of View εξ αιτίας του reset που γίνεται από το script του MouseLook. Έτσι στην αρχή του προγράμματος, έχουμε βάλει ένα μικρό 3D text σαν mini tutorial με τους χειρισμούς.

- **Συνθήκη λειτουργίας**: Για το bonus ερώτημα 5, όπου ο παίκτης – κάψουλα μπορεί να κάνει collision ή όχι με τον κύβο, δημιουργήσαμε ένα ξεχωριστό Layer στον παίκτη ονόματος “Player”. Πηγαίνοντας στο Edit→Project Settings→Physics , έχουμε κάνει disbale το κουτάκι για collision μεταξύ default/player, οπότε αν θέλετε να το δείτε κάνετε tick το κουτάκι και πλέον ο παίκτης και ο κύβος συγκρούονται κανονικά. Εικόνα παρακάτω:



- **<u>Συνθήκη λειτουργίας</u>**: Το texture φορτώνεται “manually”. Δηλαδή, αν θέλετε να δοκιμάσετε άλλη εικόνα, την κάνετε drag and drop στο φάκελο Materials και ύστερα στο material SphereTexture αλλάζετε την εικόνα από το Albedo πεδίο.
- **<u>Παρατήρηση</u>**: Έχει προστεθεί ειδικό material “Bouncy” με αναπήδηση 0.98 , γιατί είδαμε στο internet ότι είναι η βέλτιστη τιμή που πρότειναν για bounciness.

Το .zip αρχείο είναι χωρισμένο σε Executables και Source Code.

Ο φάκελος Executables περιέχει x32 και x64 version.

Για να τρέξετε σε Unity τον source code, στο .zip αρχείο:

Src Folder→project2

Έκδοση Unity: 2020.2.1f1

OS: Widows 10 Enterprise

Παραθέτουμε επίσης τα Google Drive Link Και MD5 (για παν ενδεχόμενο):

<u>Google Drive Link:</u>

https://drive.google.com/file/d/1a0a8Y-FGCgxQtjNM9im5Z3QcQe5TN1WM/view?usp=sharing

<u>MD5 Checksum</u>: 7CDAB4A978C53A68CDF620494704743E

***<u>Ευχαριστούμε πολύ για το χρόνο σας!!!</u>***