ΗΛΙΑΚΟ ΣΥΣΤΗΜΑ

ΤΣΙΟΥΡΗ ΑΓΓΕΛΙΚΗ 3354

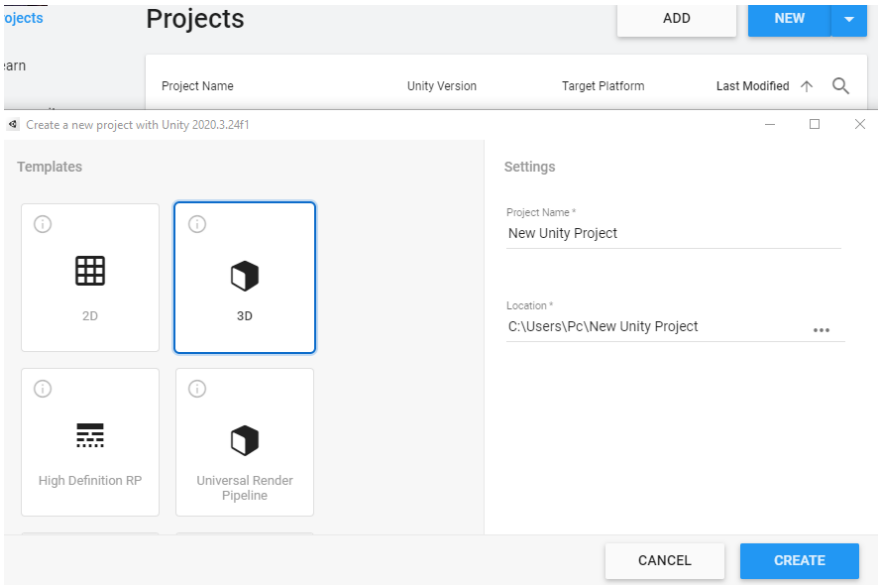
https://drive.google.com/file/d/1i2EKs8Yc1x-flXQsMh_JpztIQy79mpXn/view?usp=sharing

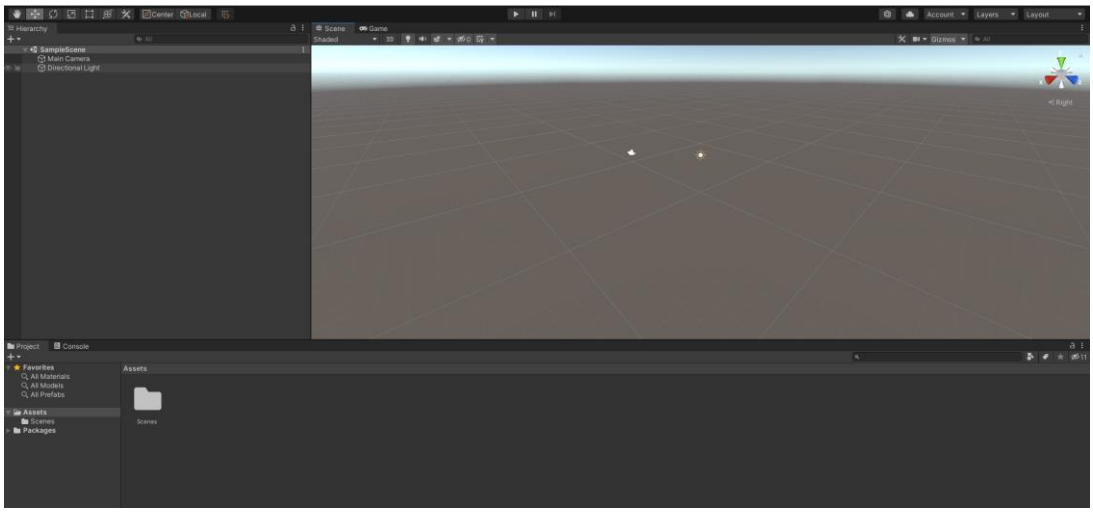
ΠΕΡΙΓΡΑΦΗ

Η διαδραστική αυτή εφαρμογή αφορά την προσομοίωση του ηλιακού συστήματος με τη χρήση της πλατφόρμας **Unity 3D**. Πιο συγκεκριμένα περιλαμβάνει τον Ήλιο και διάφορους κινούμενους πλανήτες γύρω του με τον χρήστη να προσπαθεί να τους πετύχει εκτοξεύοντας μετεωρίτες.

Η κατασκευή του παιχνιδιού αποτελείται από το γραφικό σχεδιασμό των αντικειμένων (GameObjects) που χρειαζόμαστε και από αρχεία κώδικα (scripts) σε γλώσσα C# ,μέσω του Visual Studio, για την συμπεριφορά αυτών των αντικειμένων. Μας δίνει δηλαδή την δυνατότητα να προσθέσουμε ιδιότητες και χαρακτηριστικά στα αντικείμενα.

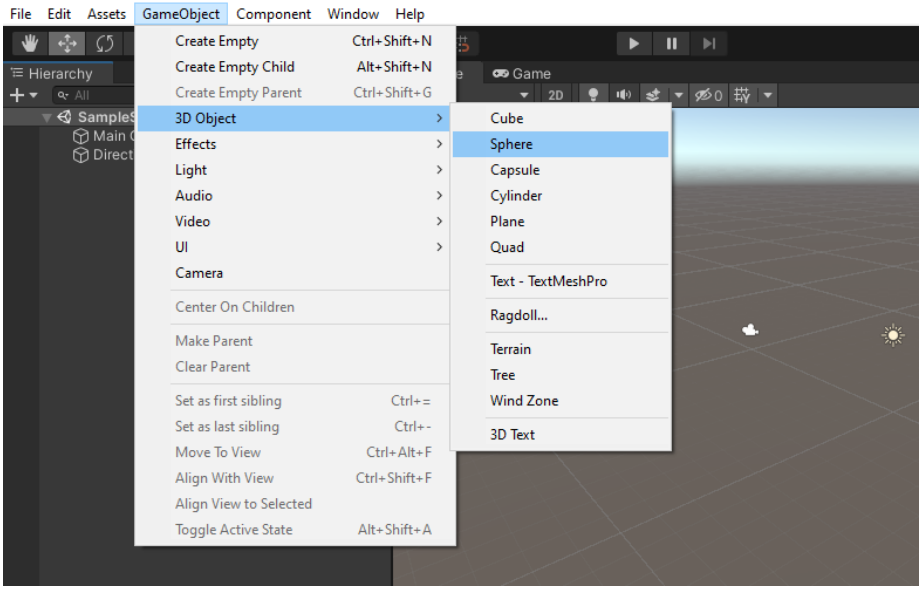
Ακολουθώντας τα βήματα όπως αναφέρονται στο σχετικό βίντεο του εργαστηρίου δημιούργησα ένα νέο Project 3D ,μια άδεια σκηνή όπου αυτόματα προσθέτει μια κάμερα και μια πηγή φωτός.



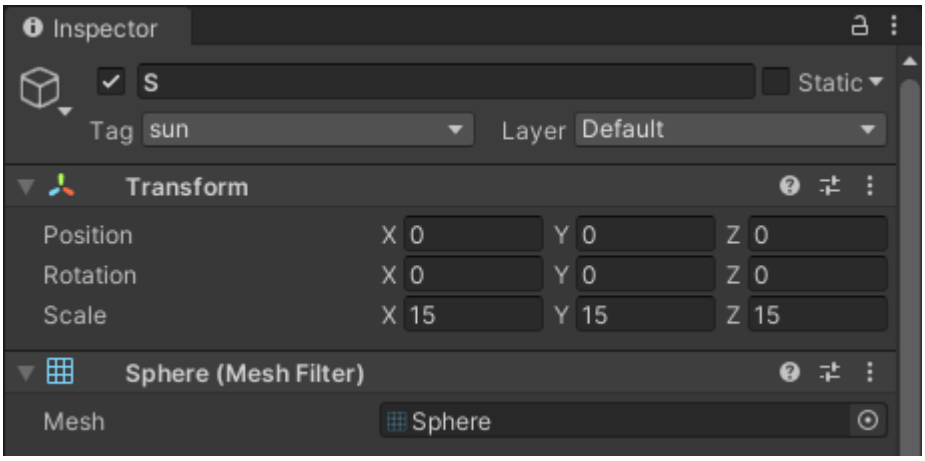


ΣΧΕΔΙΑΣΜΟΣ ΚΑΙ ΥΛΟΠΟΙΗΣΗ ΕΦΑΡΜΟΓΗΣ

- Για την δημιουργία του πρώτου αντικειμένου (Ήλιος) προσθέτω μια σφαίρα όπως φαίνεται στην εικόνα.



Για να αλλάξω τα στοιχεία της σφαίρας, όταν επιλέγω το αντικείμενο από το Hierarchy, στα **δεξιά** εμφανίζεται ένα tab Inspector που περιέχει τις ιδιότητες της.
Το πρώτο πεδίο S είναι το όνομα του αντικειμένου, στο Transform.position ορίζω το κέντρο (0,0,0) και Transform.scale για την ακτίνα θέτω x,y,z 15. Επίσης κάτι που θα μας χρειαστεί αργότερα είναι ένα Tag για να ξεχωρίζουμε το αντικείμενο. Στο συγκεκριμένο όρισα ένα καινούριο -Add tag- (από το κάτω βέλος που υπάρχει δίπλα στο πεδίο) και το ονόμασα sun.

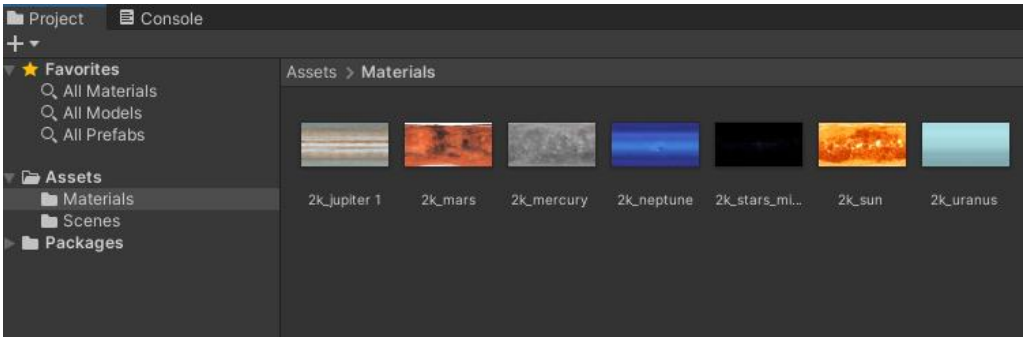


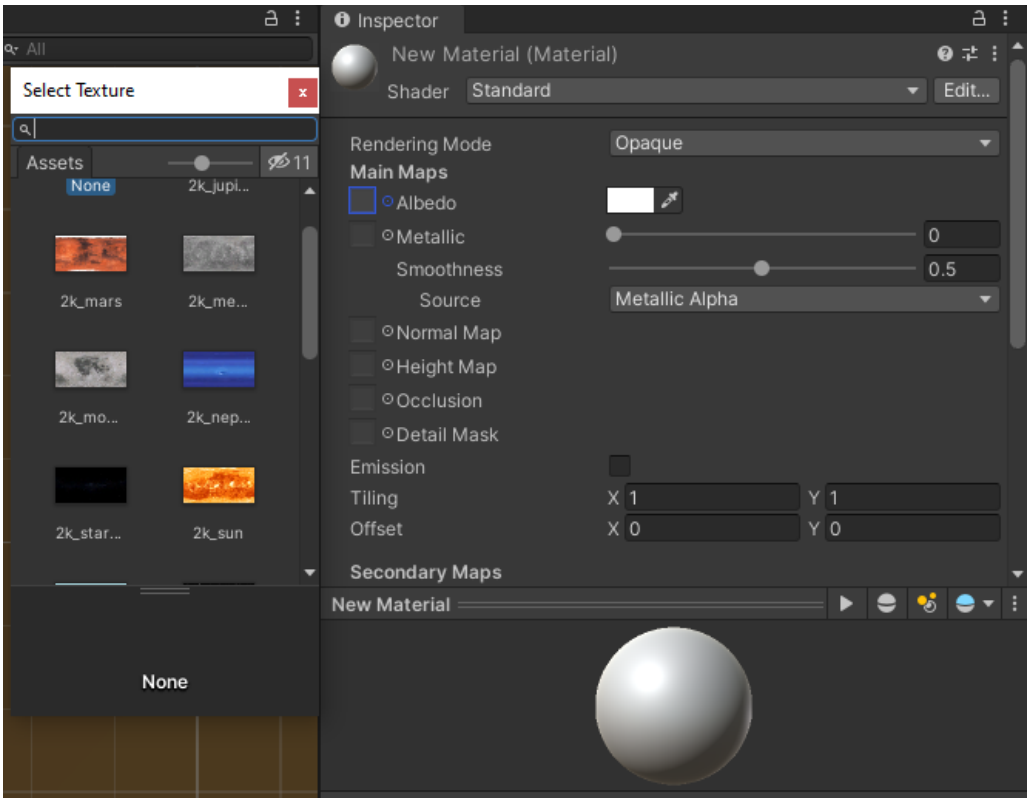
Με τον ίδιο τρόπο δημιούργησα άλλες 5 σφαίρες (πλανήτες) :

P1 r=2, P2 r=4 , P3 r=6, P4 r=7, P5 r=8.
Έπρεπε να τοποθετηθούν σωστά για να μην ακουμπάνε μεταξύ τους οι κυκλικές τροχιές, ομόκεντροι κύκλοι με κέντρο το (0,0,0) και να βρίσκονται στο ίδιο επίπεδο. Τα κέντρα τους στο επίπεδο xz. Το y παρέμεινε 0 και έθεσα κατάλληλα τα x , z για να έχω τον μικρότερο

πλανήτη πιο κοντά στον Ήλιο και ο μεγαλύτερος να είναι ο πιο απομακρυσμένος. Στους πλανήτες έβαλα το ίδιο Tag X.

- Το επόμενο βήμα ήταν να προσθέσω **texture** σε κάθε αντικείμενο. Αφού αποθήκευσα αυτά που ήθελα από https://www.solarsystemscope.com/textures/ , τα πρόσθεσα στα assets σε ένα φάκελο materials όπως φαίνεται παρακάτω και εκεί δημιούργησα material όπου στον inspector στην επιλογή albedo επέλεξα ποιο texture θέλω. Με drag and drop εφαρμόζω το Material σε κάθε object.





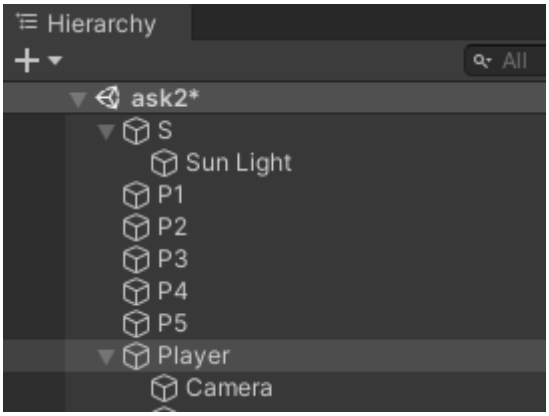
- **Lighting**

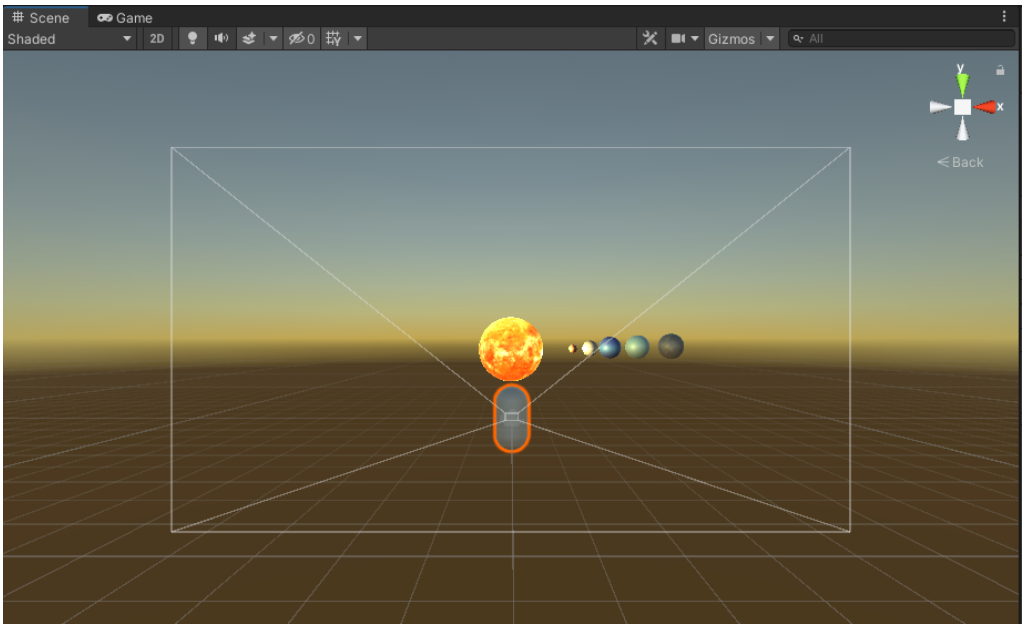
Για τον Ήλιο ,για να κάνω να φωτίζει το υπόλοιπο σύστημα πλανητών, έσβησα το light που είχε δημιουργηθεί αυτόματα και έβαλα ένα τύπου Point (Sun Light) το οποίο το έκανα child* του S και με 0,0,0 είναι στο κέντρο του Ήλιου και εξαρτάται από αυτόν. Στον inspector του Sun Light ενεργοποίησα το Light, άλλαξα το χρώμα για να είναι φυσικό σε σχέση με το Material που έχω χρησιμοποιήσει και μεγάλωσα το range και το intensity για να φτάνει το φως σε όλους τους πλανήτες και να έχω το επιθυμητό αποτέλεσμα. Στο Mode επέλεξα mixed. Επίσης για να τον κάνω πιο φωτεινό και να δείχνει σαν πηγή φωτός της σκηνής ενεργοποίησα το emission στον Inspector του sunmaterial και στο χρώμα πρόσθεσα ξανά το ίδιο texture.

*με drag and drop πάνω στο αντικείμενο που θέλω για parent στην στήλη Hierarchy

- **Κάμερα**

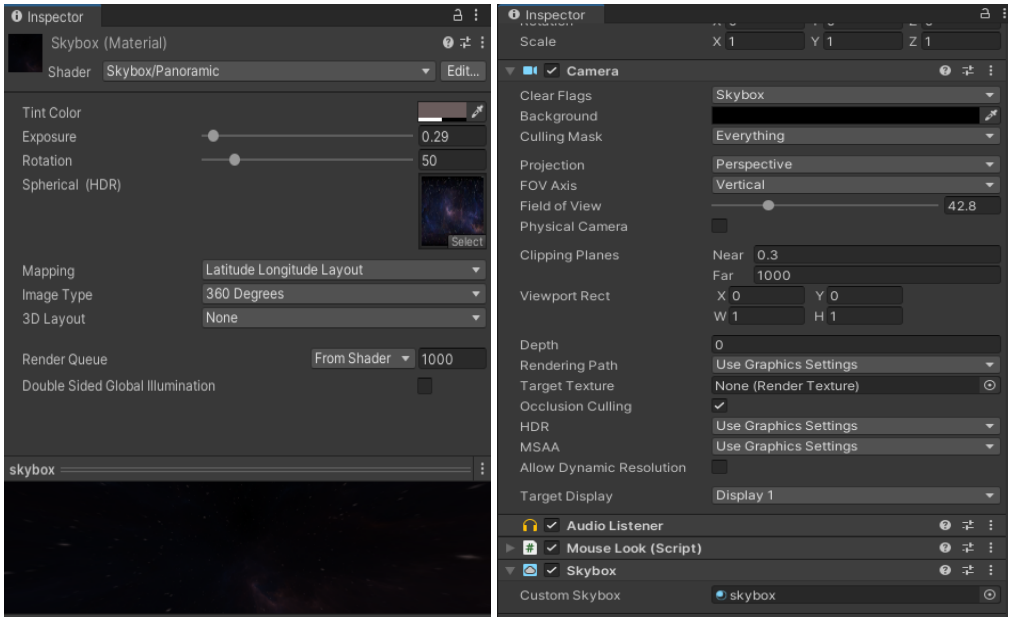
  Δημιούργησα ένα καινούργιο αντικείμενο κάψουλα, το ονόμασα Player και έκανα παιδί του την κάμερα έτσι ώστε όταν κινείται να πηγαίνει και αυτή μαζί. Κινώντας το πάνω στην σκηνή διαπίστωσα πως για να μπορώ να έχω πλήρη εικόνα του συστήματος μου και να βρίσκομαι στην ίδια ευθεία με το κέντρο του Ήλιου μια βολική θέση ήταν (0,0,-100).





  Η κάμερα έχει τοποθετηθεί στο κέντρο του Player (0,0,0).

- Για **background** στην σκηνή μου έφτιαξα material ,όπως για τους πλανήτες, όμως στον Inspector shader επέλεξα skybox/Panoramic και εκεί έβαλα το texture που διάλεξα.Στον Inspector της κάμερας πρόσθεσα το Skybox Component το οποίο ενεργοποίησα και έβαλα το material που δημιούργησα.Για να φανεί όμως αυτό άλλαξα και την επιλογή στο Clear Flags από Solid Color σε Skybox.

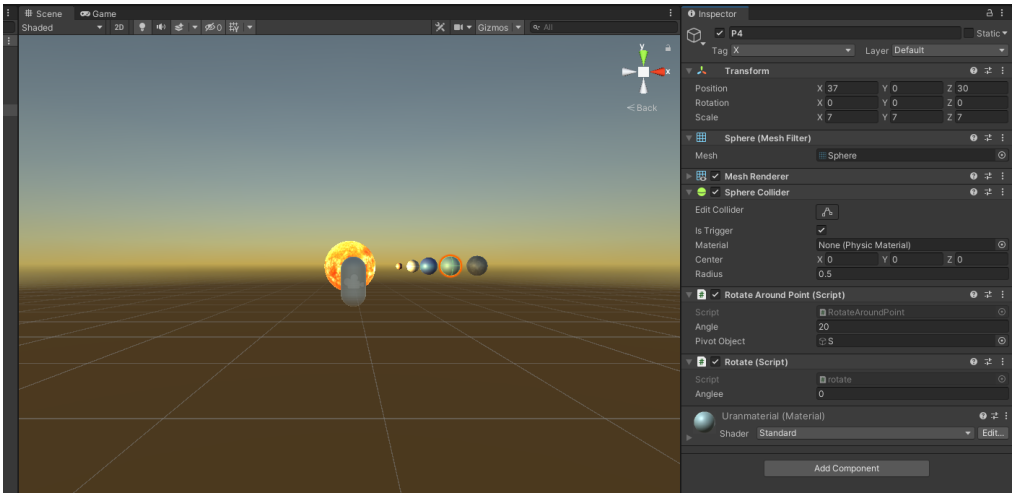


- **Κίνηση πλανητών**

Η πρώτη κίνηση που έφτιαξα ήταν γύρω από τον Ήλιο. Στα assets δημιούργησα ένα C# script RotateAroundPoint. Το μόνο που χρειάστηκα ήταν 2 πεδία την γωνία και το αντικείμενο S γύρω από το οποίο θα περιστρέφεται κάθε πλανήτης. Με την μέθοδο **RotateAround** στην transform κλάση θα γίνει αυτή η περιστροφή δίνοτας 1η παράμετρο την **θέση**(position) του αντικειμένου S , 2η τον **άξονα** γύρω από τον οποίο θα γίνει η περιστροφή ,εγώ ήθελα γύρω από τον **y** γι'αυτό έβαλα (0,1,0) και 3η την ταχύτητα δηλαδή την γωνία που θα ορίσω πολλαπλασιασμένη με το Time.deltaTime που είναι ο χρόνος που πέρασε από το προηγούμενο frame ,αυτό κάνει και την κίνηση πιο ομαλή.

Επίσης έκανα ένα script rotateY για την κίνηση των πλανητών **γύρω από τον εαυτό τους**.Με παρόμοια μέθοδο την **Rotate** με παράμετρο ένα διάνυσμα που καθορίζει τον άξονα γύρω από τον οποίο θα περιστρέφεται εγώ επέλεξα πάλι τον άξονα y οπότε στην θέση αυτή είναι το πεδίο που όρισα ως γωνία πολλαπλασιασμενο με το Time.deltaTime.

Και τα δύο τα πρόσθεσα σε κάθε πλανήτη όπου έβαλα τυχαίες τιμές σε κάθε γωνία για την κίνηση γύρω από τον Ήλιο και ίδια γωνία για την κίνηση γύρω από τον εαυτό τους.



- **First Person Movement**

  Για να έχω κίνηση σε όλη την σκηνή χρειάστηκαν δύο διαφορετικά script **PlayerMovement** και **mouseLook**.

  Το πρώτο και βασικό για να μπορέσω να χειριστώ το αντικείμενο και να δώσω κίνηση είναι να προσθέσω στον player **Character Controller Component.**

  Στο **PlayerMovemen**t γίνεται η κίνηση μέσω του πληκτρολογίου είτε με τα βελάκια είτε με τα adsw. Πιο συγκεκριμένα με τα δεξί/αριστερό ή d/a κινείται ο player στον άξονα x θετικά αρνητικά αντίστοιχα. Για το input αυτό χρησιμοποίησα την GetAxis που επιστρέφει -1 ή 0 ή 1 (0 είναι όταν δεν πατάω κάποιο πλήκτρο, -1 στο αριστερό και κάτω βέλος ,1 στο δεξί και πάνω βέλος). Δημιουργώ ένα διάνυσμα από τα διανύσματα κατεύθυνσης right και forward πολλαπλασιασμένα με τις μεταβλητές από τα input (δηλαδή αν έχω πατήσει το δεξί βέλος μόνο το διάνυσμα μου θα είναι (1,0,0). Η μετακίνηση γίνεται με την μέθοδο move στο αντικείμενο που έχω ορίσει πεδίο τύπου CharacterController με παράμετρο την κατεύθυνση(το διάνυσμα) πολλαπλασιασμένο με την ταχύτητα και το Time.deltaTime.

  Στο **mouseLook** γίνεται η περιστροφή με την χρήση του ποντικιού όπου το input GetAxis, η κατεύθυνση του ποντικιού, πολλαπλασιάζεται με το Time.deltaTime και το mouseSensitivity που έχω ως πεδίο. Επίσης σαν πεδίο πρόσθεσα την transform κλάση του player για να μπορώ να κάνω μετατροπές εφόσον το script αυτό είναι στην camera. Η κίνηση του ποντικιού πάνω κάτω είναι περιστροφή γύρω από τον άξονα x και αριστερά δεξιά είναι γύρω από τον y.

  - Το rotate γύρω από τον άξονα y γίνεται με τον player και είναι ίδιο τρόπο με το rotate των πλανητών γύρω από τον εαυτό τους με την διαφορά ότι εδώ δεν υπάρχει σταθερή ταχύτητα αλλά κινείται ανάλογα με το input mouseX.
    (*Vector3up είναι το* (0,1,0))

  - Το rotate γύρω από τον άξονα x γίνεται με την κάμερα. Χρησιμοποίησα την localRotation γιατί η περιστροφή της είναι σχετική με την περιστροφή του parent player. Η Quaternion.Euler επιστρέφει την περιστροφή γύρω από τον x κατά xRotation μοίρες.

- **Δημιουργία τυχαίων αντικειμένων**

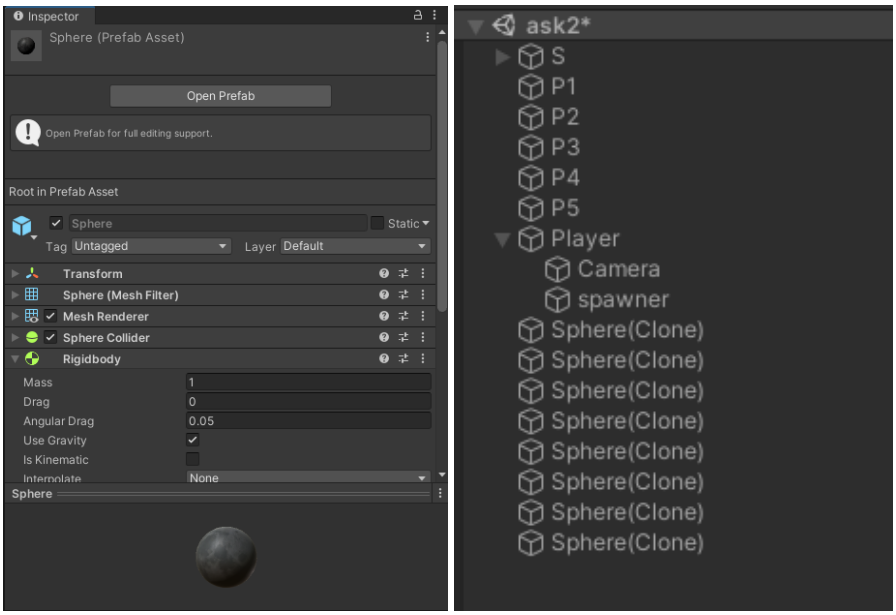
  Για να μπορώ να δημιουργώ αντικείμενα που θα εμφανίζονται με κάποια συγκεκριμένη πράξη του χρήστη και να μην είναι ορατά πριν απ΄αυτή ο τρόπος είναι να φτιάξω Prefab Asset. Στο συγκεκριμένο παιχνίδι ήθελα να εκτοξεύω μετεωρίτες πατώντας το spacebar. Όπως στην αρχή δημιούργησα μία σφαίρα την οποία έβαλα στα assets και την έσβησα από την στήλη hierarchy. Χωρίς καμία διαφορά από τα προηγούμενα, πρόσθεσα και εκεί material με το texture που διάλεξα.

- **Spawner**

  Η εκτόξευση των μετεωριτών γίνεται από το σημείο της κάμερας. Οπότε δημιούργησα ένα κενό μέσα στον Player το οποίο ονόμασα spawner και η θέση του είναι στο κέντρο της κάψουλας (όπως και η κάμερα). Του πρόσθεσα και το mouseLook script για να ακολουθεί την κίνηση από το ποντίκι.

- **Εκτόξευση**

  Μέσα στο spawner έκανα το spawner script που πραγματοποιεί την εκτόξευση των αντικειμένων. Σαν πεδίο έβαλα το object που θέλουμε να εκτοξεύεται (prefab) , την ταχύτητα και για να μην έχω ίδιο μέγεθος, με την Random.Range δίνονται διαφορετικές τιμές στο size. Κάθε φορά που ο χρήστης πατά το πλήκτρο spacebar δημιουργείται ένας κλώνος του object με την Instantiate (με ίδια θέση και κατεύθυνση) αλλά διαφορετικού μεγέθους transform.localScale (με το size που έχει βγει τυχαία). Αφού πρόσθεσα rigidbody component στο prefab sphere με την transform.forward επί την ταχύτητα γίνεται η κίνηση.
  Αριστερά στο hierarchy για κάθε αντικείμενο που εκτοξεύεται εμφανίζεται sphere(clone).



- **Σύγκρουση**

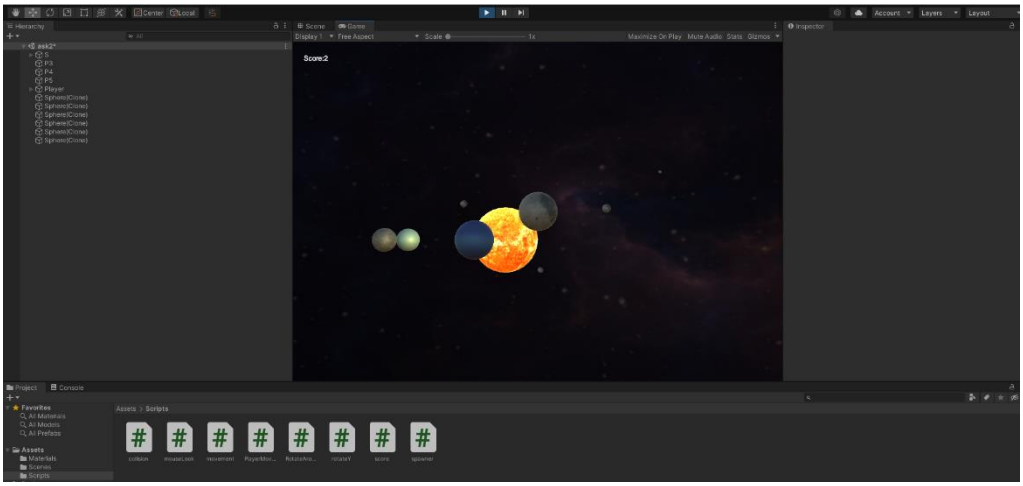
  Πρώτο βήμα ήταν η ενεργοποίηση του sphere collider στα αντικείμενα που ήθελα να ανιχνεύεται η σύγκρουση άρα σε όλους τους πλανήτες (και τον Ήλιο) και στο prefab.
  Η σύγκρουση γίνεται στο **collision script**. Η μέθοδος OnTriggerEnter ανιχνεύει αν κάποιο άλλο collider μπαίνει στο collider του object (μετεωρίτη). Εδώ χρησιμοποιήθηκαν τα tags που είχα ορίσει στην αρχή. Οπότε υπάρχουν δύο περιπτώσεις :

  - Αν το tag είναι X άρα πλανήτης τότε εξαφανίζονται (destroy) και ο πλανήτης και ο μετεωρίτης και το σκορ (θα αναφερθώ παρακάτω) αυξάνεται
  - Αν το tag είναι sun τότε εξαφανίζεται μόνο ο μετεωρίτης.

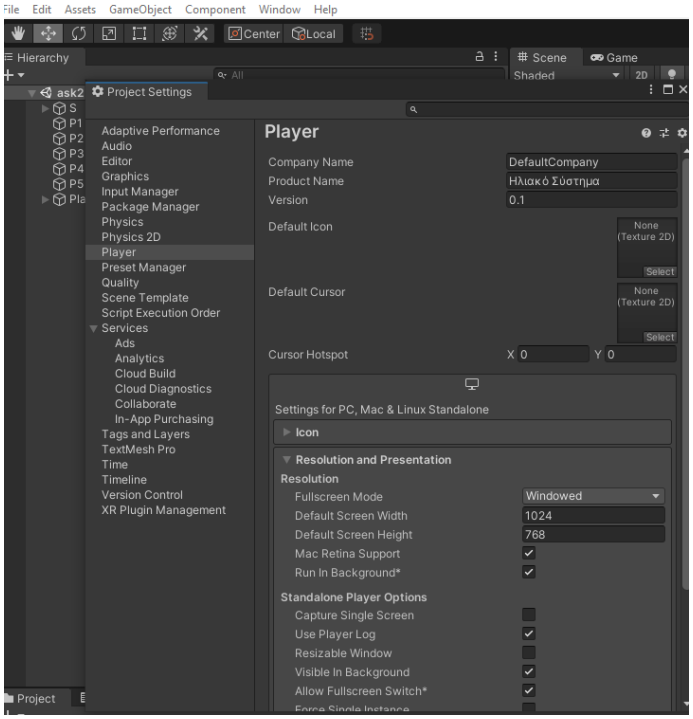
  Σβήνονται και από το hierarchy που ανέφερα παραπάνω.

- **Σκορ**

  Στο score script χρησιμοποίησα την GUI.label για να εμφανίσω κείμενο στην οθόνη και την μεταβλητή points static(για να μπορεί να χρησιμοποιηθεί στο collision). Με τις παραμέτρους του rect ορίζουμε που θέλουμε να εμφανίζεται το label score,το πλάτος και το ύψος.



Για να φτιάξω το εκτελέσιμο αρχείο αποθήκευσα την σκηνή μου και στο File->Build Settings έκανα build and run για windows. Στο edit->Project Settings έδωσα το όνομα και στο resolution and presentation επέλεξα να εμφανίζεται σαν παράθυρο με ανάλυση 1024x768.Το background ρυθμίζεται από το splash image.



-GAME OVER-