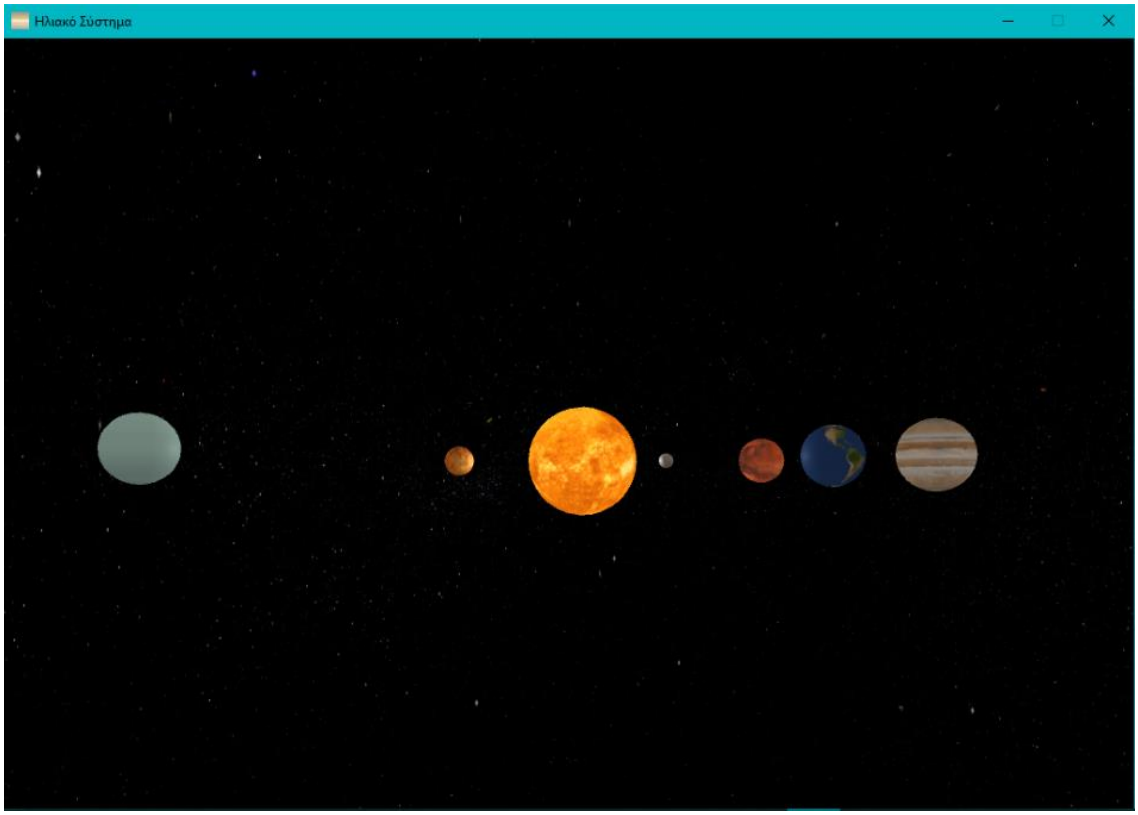
Πανεπιστήμιο Ιωαννίνων, Τμήμα Μηχανικών Υπολογιστών και Πληροφορικής
Ακαδημαϊκό Έτος 2021 – 2022

ΜΥΥ702-ΓΡΑΦΙΚΑ ΥΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΣΥΣΤΗΜΑΤΑ ΑΛΛΗΛΕΠΙΔΡΑΣΗΣ

ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΗ ΑΣΚΗΣΗ 2 – Unity 3D



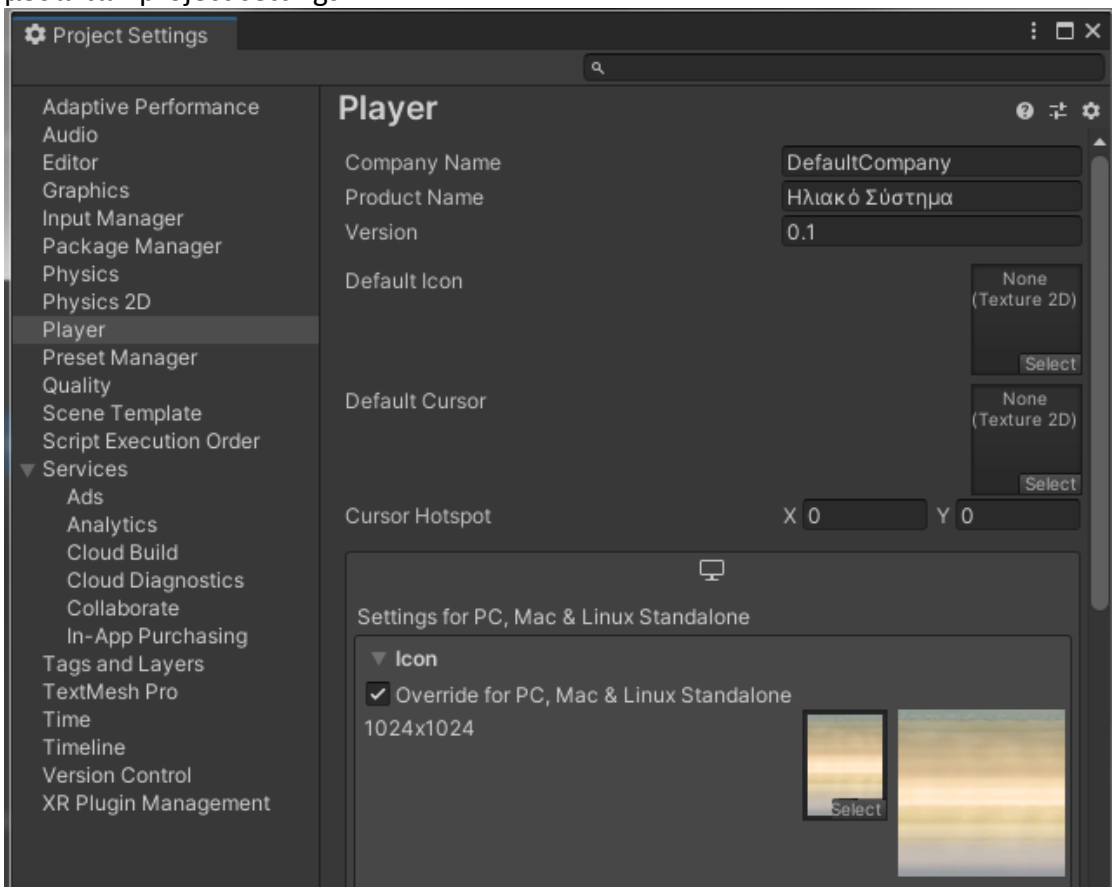
Νικολέττα Μπιντζηλαίου, 4435
Θωμαή Τσιαούση, 4510

Ερώτημα (i)

Στο script GameManager θέτουμε την ανάλυση σε 1024x768. Μπορούμε να κλείσουμε την εφαρμογή πατώντας το πλήκτρο Q.

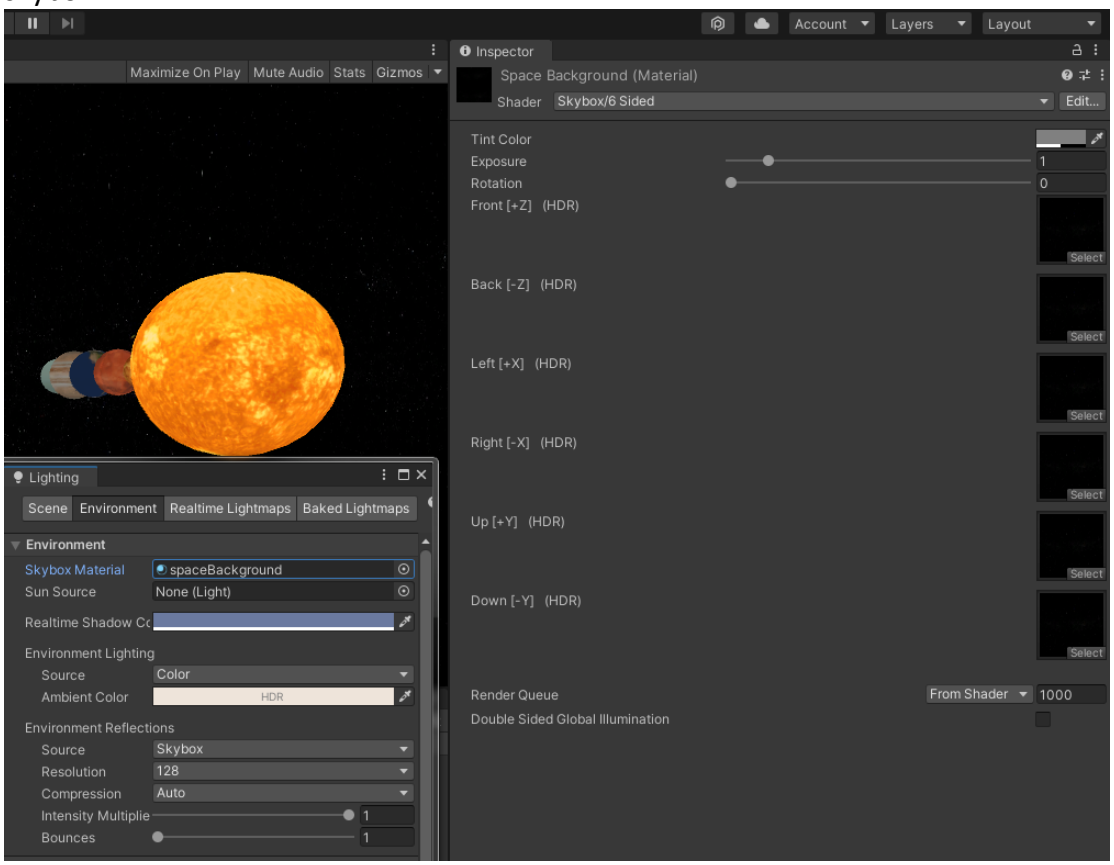
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class GameManager : MonoBehaviour
{
    void Start()
    {
        Screen.SetResolution(1024, 768, false);
    }

    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Q))
        {
            Application.Quit();
        }
    }
}
```

Θέτουμε στην εφαρμογή μας το τίτλο «Ηλιακό Σύστημα» και της βάζουμε εικονίδιο, μέσω των project settings.



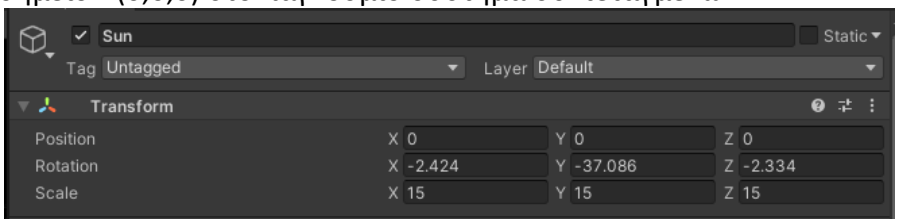
Bonus (c):

Μέσω την ιστοσελίδας που μας δόθηκε για τις υφές, βρήκαμε κατάλληλη εικόνα που να αντιπροσωπεύει το διάστημα και την επεξεργαστήκαμε. Φτιάξαμε ένα καινούργιο material με shader skybox/6sided και το «φορτώσαμε» στο material του skybox.

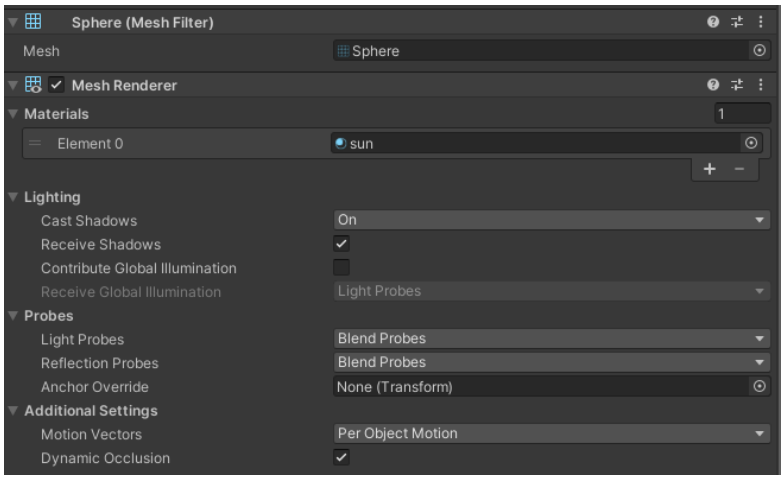


Ερώτημα (ii)

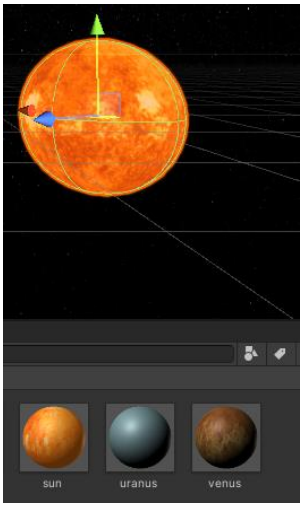
Η σφαίρα Sun αντιπροσωπεύει τον Ήλιο, έχει ακτίνα 15 και το κέντρο της είναι το σημείο Α(0,0,0) στο παγκόσμιο σύστημα συντεταγμένων.



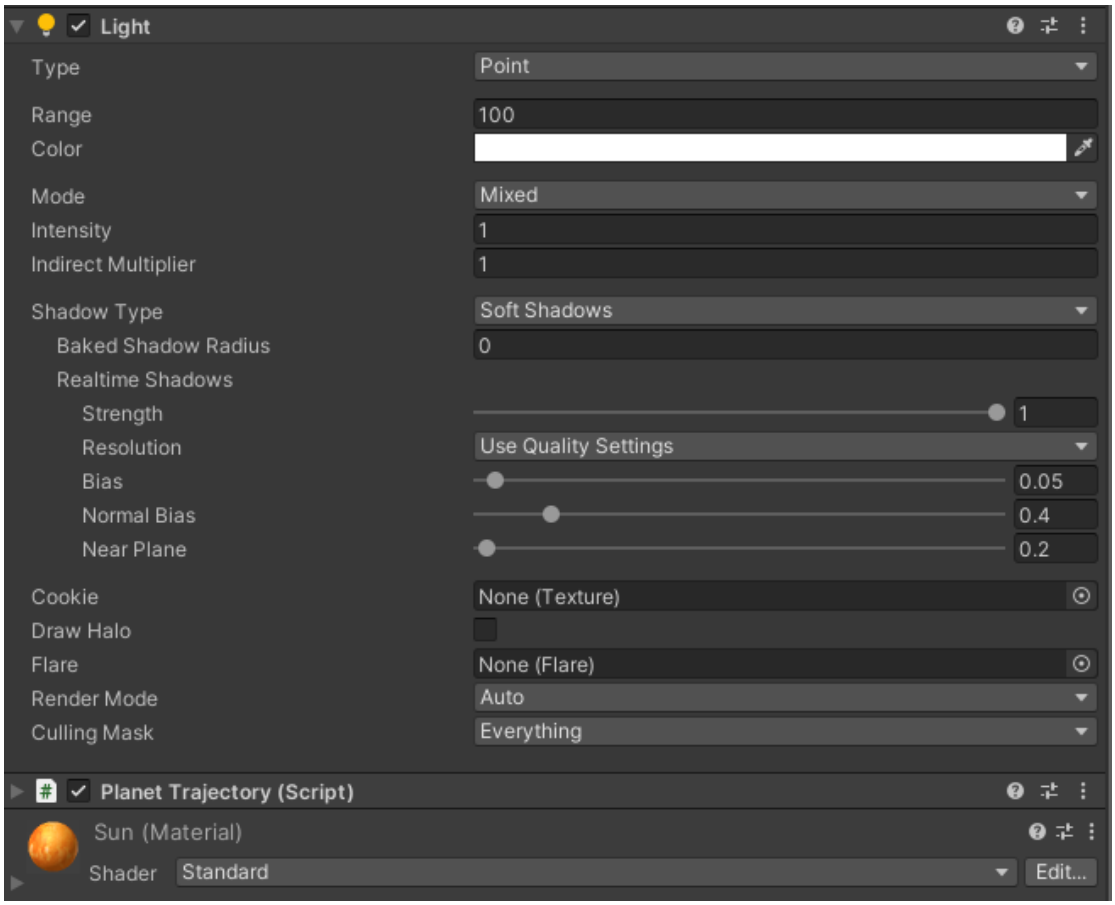
Στον Ήλιο φορτώνουμε το texture sun.jpg, βαζοντάς το σαν material στην σφαίρα.

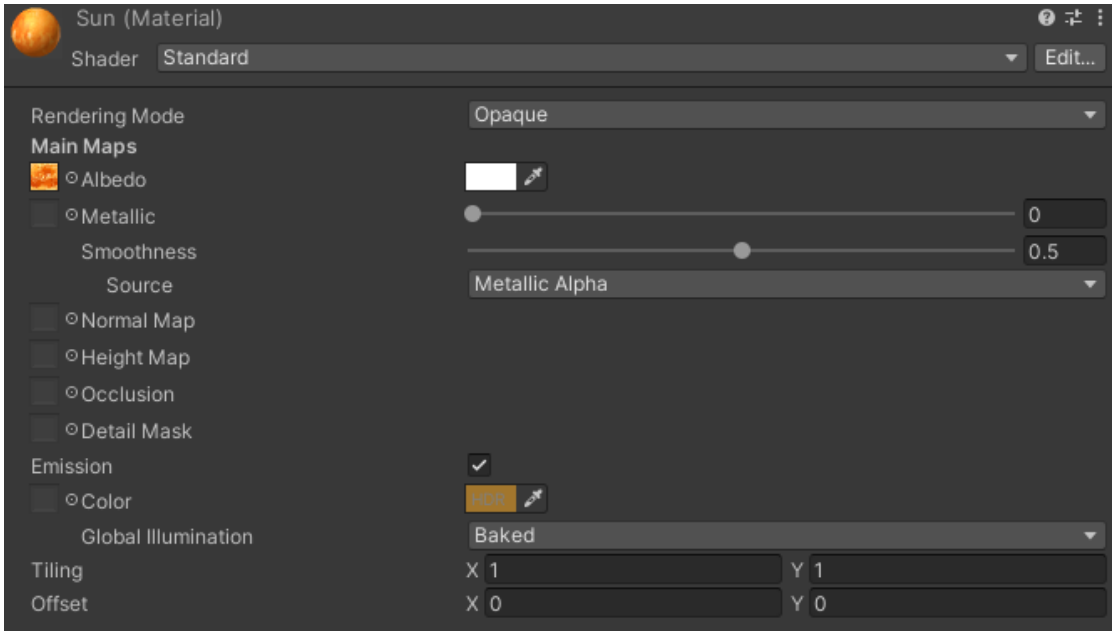


Ο



Ήλιος φωτίζει το σύστημα πλανητών μέσω του component του Light (Point) και αλλάζοντας το emission του material του.

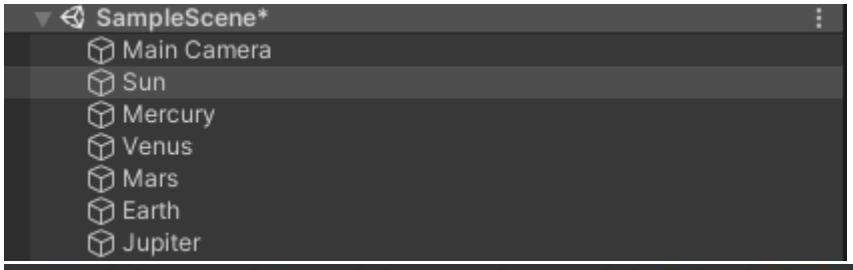




Ερώτημα (iii)

Εκτός του Ήλιου, το σύστημα περιλαμβάνει και 6 πλανήτες. Οι πλανήτες είναι σφαίρες με διαφορετικές ακτίνες, που κινούνται σε κυκλικές τροχιές γύρω από τον Ήλιο με ταχύτητα (velocity).

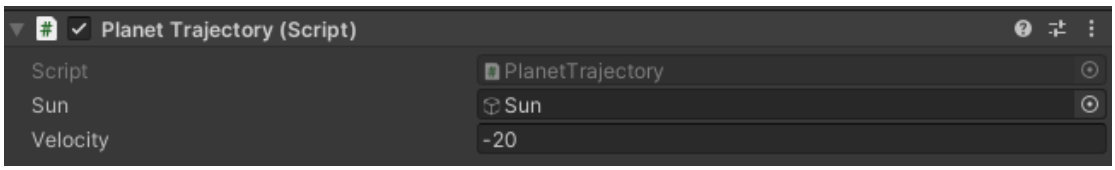
Έτσι, φορτώνουμε τις αντίστοιχες υφές στις σφαίρες, φτιάχνοντας τα κατάλληλα materials:



Υλοποιούμε τις τροχιές των πλανητών γύρω από τον Ήλιο μέσω του script PlanetTrajectory για κάθε GameObject. Συνδέουμε τον Ήλιο σε κάθε script, ώστε ο πλανήτης να περιστρέφεται γύρω από αυτόν με την ταχύτητα που θέλουμε.

```
transform.RotateAround(Sun.transform.position, Vector3.up, velocity * Time.deltaTime);
```

Π.χ. για τον Mercury, περιστροφή στη θέση που τον έχουμε στήσει στην σκηνή μας, γύρω από τον Sun, με ταχύτητα 20 και φορά αντίθετη του ρολογιού.

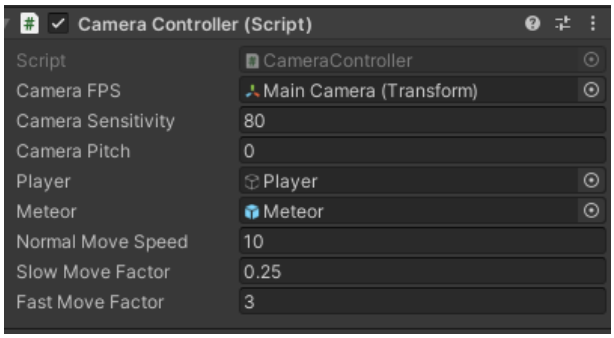


Bonus (b):
Με ακριβώς ανάλογο τρόπο υλοποιούμε κίνηση των πλανητών γύρω από τον εαυτό τους. Μέσω του script PlanetRotation, ο κάθε πλανήτης περιστρέφεται γύρω από τον y άξονά του.

```
if (planet != null)
{
    //rotation of planet around itself
    transform.RotateAround(planet.transform.position, planet.transform.up, velocity * Time.deltaTime)
}
```

Ερώτημα (iv)

Φτιάχνουμε μία κάψουλα, η οποία αντιπροσωπεύει τον «παίκτη» μας και τοποθετούμε πάνω της την κάμερα και το αντίστοιχο script, CameraController. Χρησιμοποιώντας και το ExtendedFlyCam που μας δώθηκε στο demo, παίρνουμε τη θέση του cursor του ποντικιού και τα πλήκτρα που πατάει ο χρήστης του παιχνιδιού/εφαρμογής.



```
void UpadateMouseLookAt()
{
    Vector2 mouse = new Vector2(Input.GetAxis("Mouse X"), Input.GetAxis("Mouse Y"));

    // y axis
    cameraPitch += mouse.y * cameraSensitivity * Time.deltaTime;
    cameraPitch = Mathf.Clamp(cameraPitch, -90.0f, 90.0f);  // to not flip upside down

    cameraFPS.localEulerAngles = Vector3.right * (-cameraPitch);
    // x axis
    Player.transform.Rotate(Vector3.up * mouse.x * cameraSensitivity * Time.deltaTime); // rotate on x axis

    if (Input.GetKey(KeyCode.LeftShift) || Input.GetKey(KeyCode.RightShift))
    {
        Player.transform.position += transform.forward * (normalMoveSpeed * fastMoveFactor) * Input.GetAxis("Vertical") * Time.deltaTime;
        Player.transform.position += transform.right * (normalMoveSpeed * fastMoveFactor) * Input.GetAxis("Horizontal") * Time.deltaTime;
    }
    else if (Input.GetKey(KeyCode.LeftControl) || Input.GetKey(KeyCode.RightControl))
    {
        Player.transform.position += transform.forward * (normalMoveSpeed * slowMoveFactor) * Input.GetAxis("Vertical") * Time.deltaTime;
        Player.transform.position += transform.right * (normalMoveSpeed * slowMoveFactor) * Input.GetAxis("Horizontal") * Time.deltaTime;
    }
    else
    {
        Player.transform.position += transform.forward * normalMoveSpeed * Input.GetAxis("Vertical") * Time.deltaTime;
        Player.transform.position += transform.right * normalMoveSpeed * Input.GetAxis("Horizontal") * Time.deltaTime;
    }
```

Επίσης, μέσα στο script αυτό, δημιουργούμε και τον μετεωρίτη στην κατάλληλη θέση και τον εκτοξεύουμε, όπως αναφέρουμε παρακάτω.

Ερώτημα (v)

Αρχικά, σε ένα script Meteor, ορίζουμε την ταχύτητα και την ακτίνα του μετεωρίτη μέσω μιας γεννήτριας τυχαίων αριθμών στο διάστημα d=[1.0,5.0] (Start()).

```
void Start()
{
    // Gets the local scale of game object
    Vector3 objectScale = transform.localScale;
    //generate random num for radius
    float randnum = Random.Range(1.0f, 5.0f);
    // Sets the local scale of game object
    transform.localScale = new Vector3(objectScale.x * randnum, objectScale.y * randnum, objectScale.z * randnum);
}
```

Στη συνέχεια, κάθε φορά που ο χρήστης πατάει το spacebar θα υλοποιείται ένας τέτοιος μετεωρίτης (σφαίρα) μέσα στο script για την κάμερα CameraController (Instantiate).

```
if ((Input.GetKeyDown(KeyCode.Space)))
{
    print("METEOR LAUNCHED");
    Instantiate(Meteor, transform.position, transform.rotation);
}
```

Έτσι, θα μπορεί να παίρνει την θέση του παίκτη και να τη χρησιμοποιεί στο Meteor για να κινηθεί στην ευθεία που κοιτάει η κάμερα. (Update()).

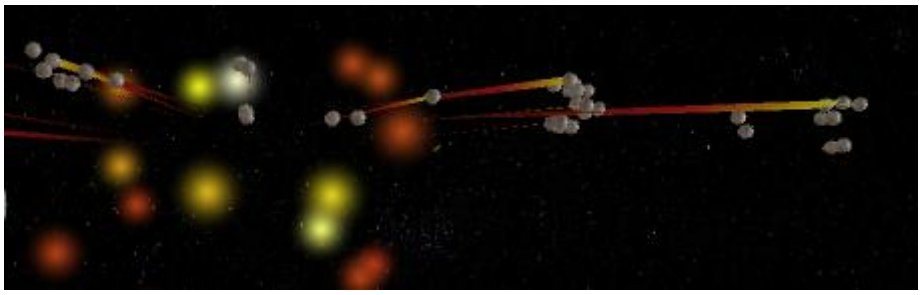
```
void Update()
{
    transform.position += transform.forward *speed*Time.deltaTime;

}
```

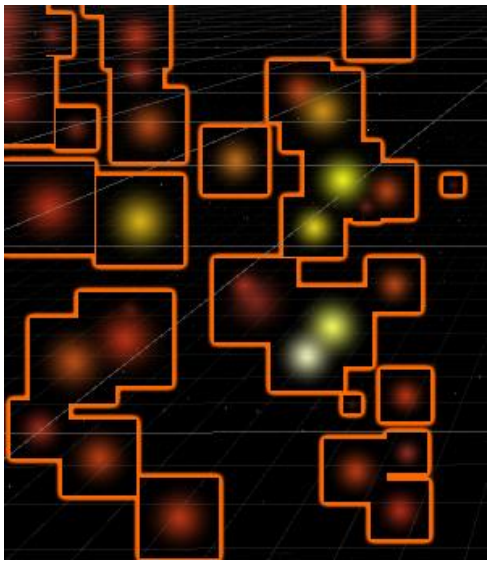
Τέλος, όταν ο μετεωρίτης συγκρούεται με έναν πλανήτη, τότε δημιουργείται «έκρηξη» όπου σπάνε τα δύο αυτά σώματα σε περισσότερα. Αυτό το επιτυγχάνουμε μέσω του script Collision. Συγκεκριμένα, στην περίπτωση της σύγκρουσης, πραγματοποιείται «έκρηξη» του πλανήτη (planetExplosion()) και «έκρηξη» του μετεωρίτη (meteorExplosion()). Σε κάθε μία από αυτές τις εκρήξεις/μεθόδους αντίστοιχα, τα σώματα που συγκρούστηκαν εξαφανίζονται και στη θέση τους δημιουργούνται πολλά άλλα μικρά prefabs πλανήτες (chuncks), τα οποία μετά από λίγο εξαφανίζονται.

Bonus (α):

Με τη χρήση particle system, προσθέσαμε ένα εφέ «φωτιάς» κατά την πρόσκρουση των σωμάτων (script Collision). Επίσης, το κάθε chunck αφήνει πίσω του ένα trail «φωτιάς».

Παράλληλα, κάθε φορά που γίνεται σύγκρουση μετεωρίτη με κάποιο πλανήτη, παίζει ο ήχος explosion.wav.





Παρατηρήσεις:

Υλοποιήσαμε σύστημα σκορ για την σύγκρουση μεταξύ μετεωρίτη και πλανήτη. Το σκορ αναγράφεται στην κάτω αριστερή γωνία της οθόνης του παίκτη, όπως βλέπει η κάμερα. Όταν ο παίκτης πετύχει όλους τους πλανήτες, εμφανίζεται το αντίστοιχο μήνυμα.