MYY702
Γραφικά Υπολογιστών και Συστήματα Αλληλεπίδρασης
Διδάσκων: Ιωάννης Φούντος

Αναφορά Προγραμματιστικής Άσκησης 2
Unity 3D

Ομάδα:
Κονδυλία Βέργου 4325
Παναγιώτης Βουζαλής 2653

Χειμερινό Εξάμηνο 2021

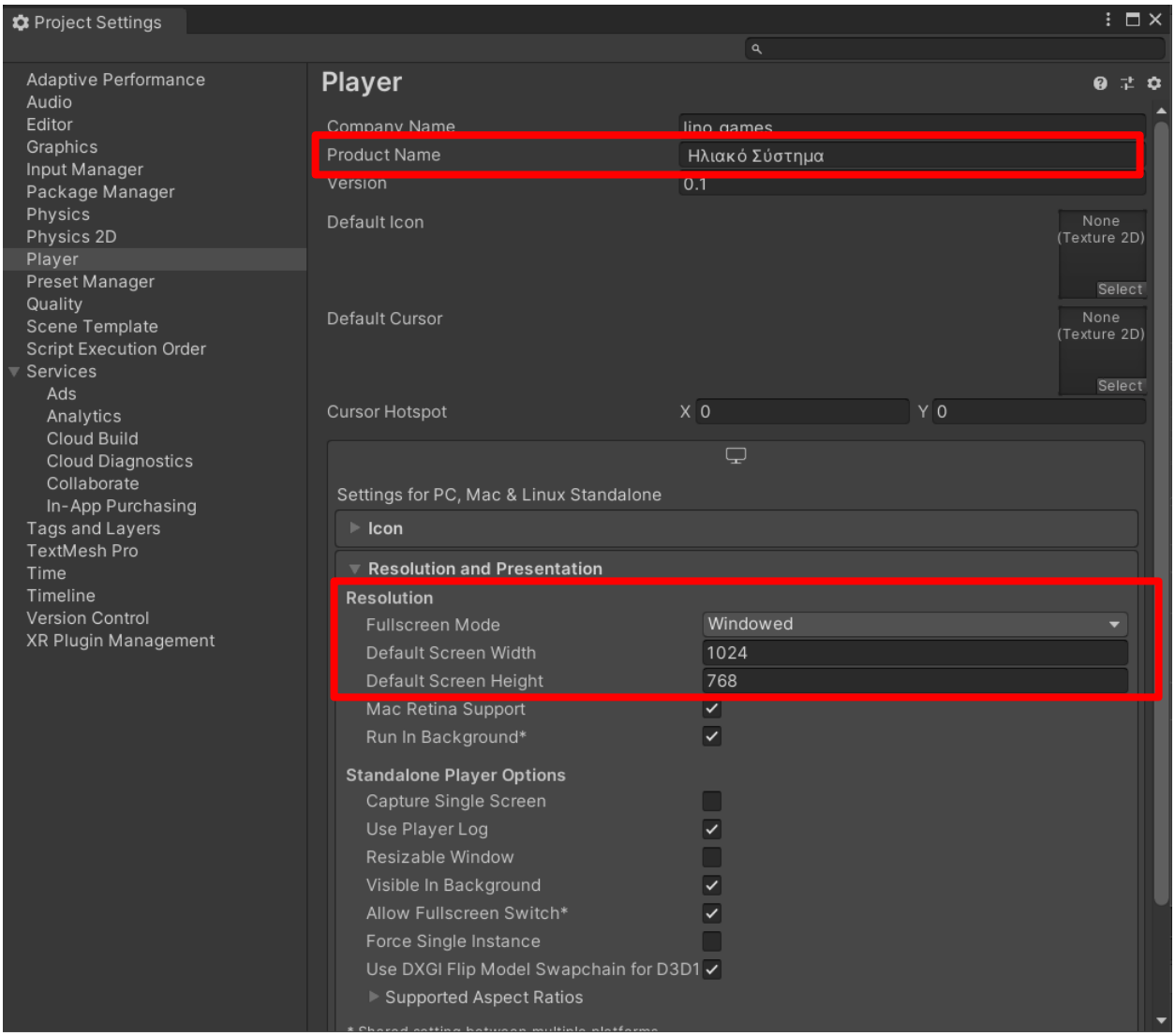
Περιεχόμενα

Η άσκηση υλοποιήθηκε σε Unity 3D σε Windows Host.

Ερώτημα (i)

Ανάλυση & Τίτλος Παραθύρου

Για να φτιάξουμε ένα παράθυρο με ανάλυση 1024x768 και τίτλο «Ηλιακό Σύστημα», πήγαμε στις ρυθμίσεις του παραθύρου της Unity: `Edit -> Project Settings -> Player` και κάναμε τα παρακάτω



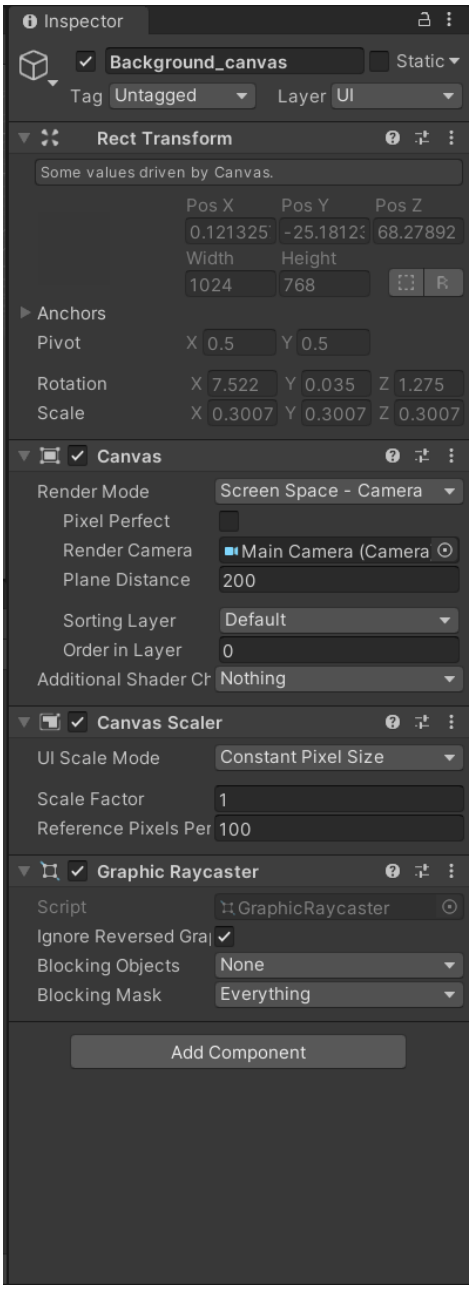
Background (bonus c)

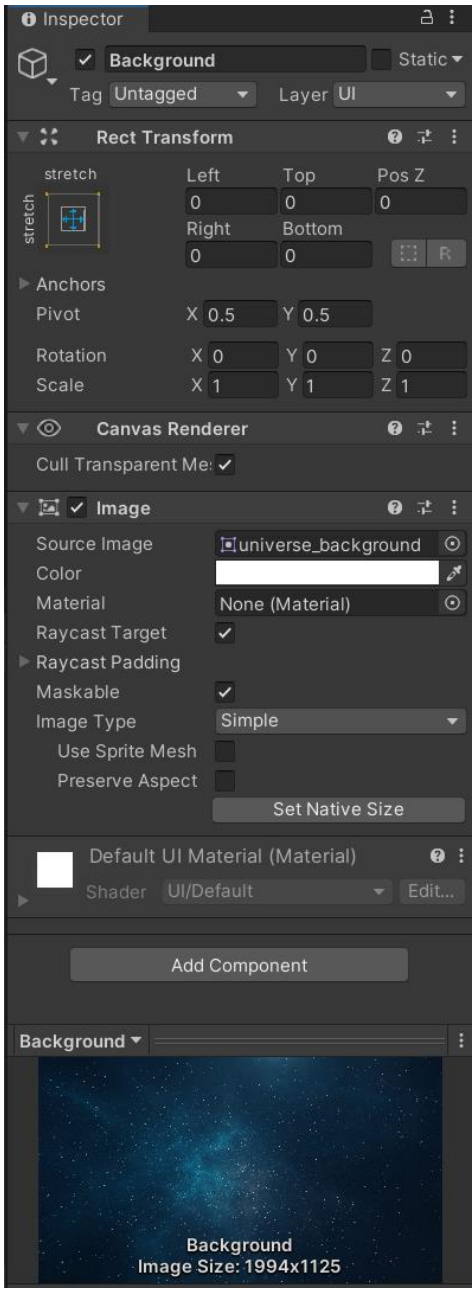
Για να βάλουμε ως background φωτογραφία από το διάστημα, προσθέσαμε αρχικά στη σκηνή ένα Canvas, το Background_canvas (`Δεξί κλικ -> UI -> Canvas`)

Μέσα στο Background_canvas, δημιουργήσαμε ένα Image, το Background (Δεξί κλικ -> UI -> Image), στο οποίο φορτώσαμε την εικόνα του διαστήματος

Ακολουθήσαμε τις οδηγίες του site: **Background of MainCamera (Unity C#)**

Παρακάτω παρατίθενται screenshots του Inspector του Background_canvas και του Background

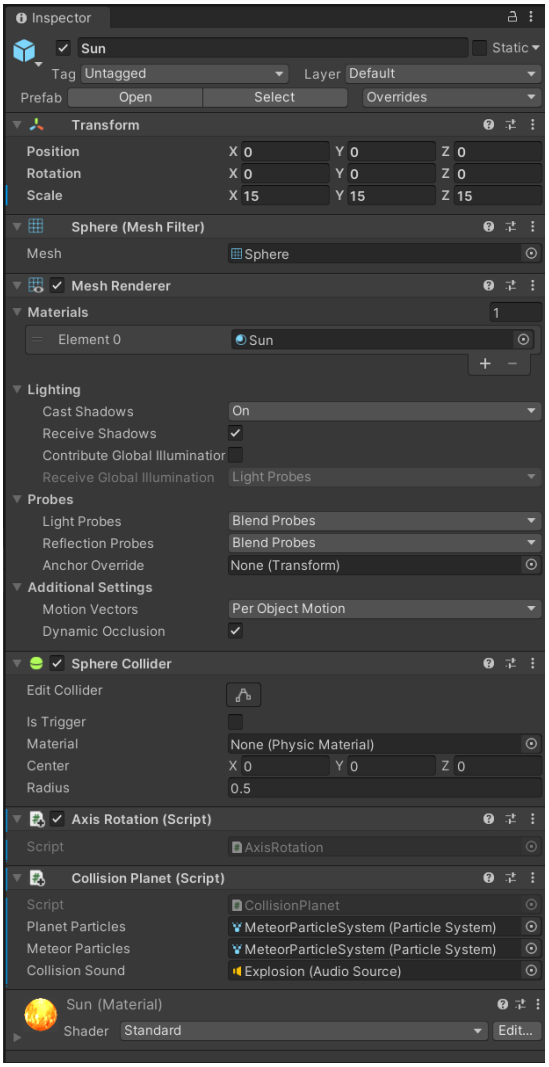




Ερώτημα (ii)

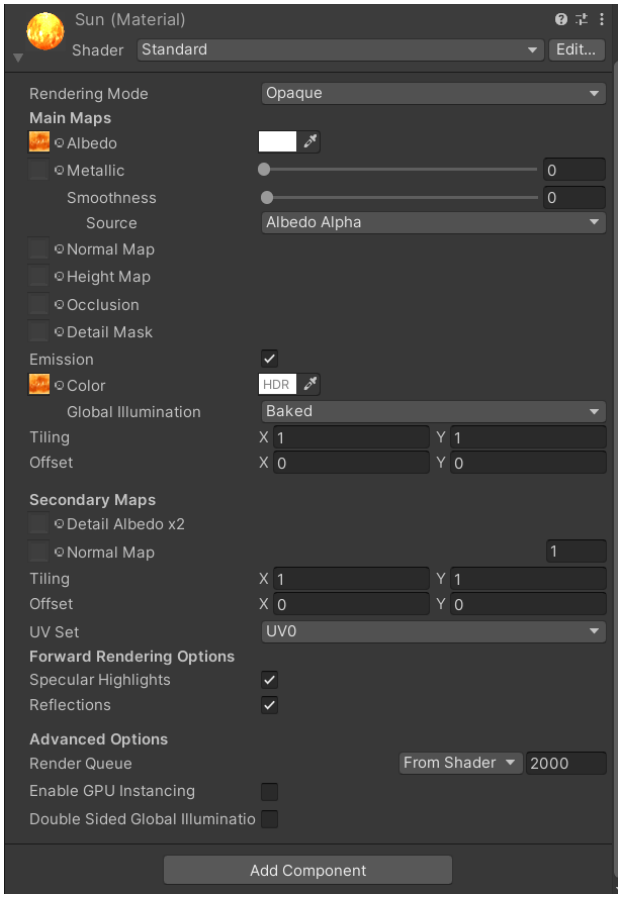
Φόρτωση & Φωτισμός Ήλιου

Για την φόρτωση του Ήλιου, καθώς και των πλανητών στην συνέχεια, χρησιμοποιήσαμε τα prefabs από το tutorial: learn.unity.com/tutorial/zoe-prepare-your-scene



Το κέντρο του Ήλιου είναι στο position: (0,0,0)

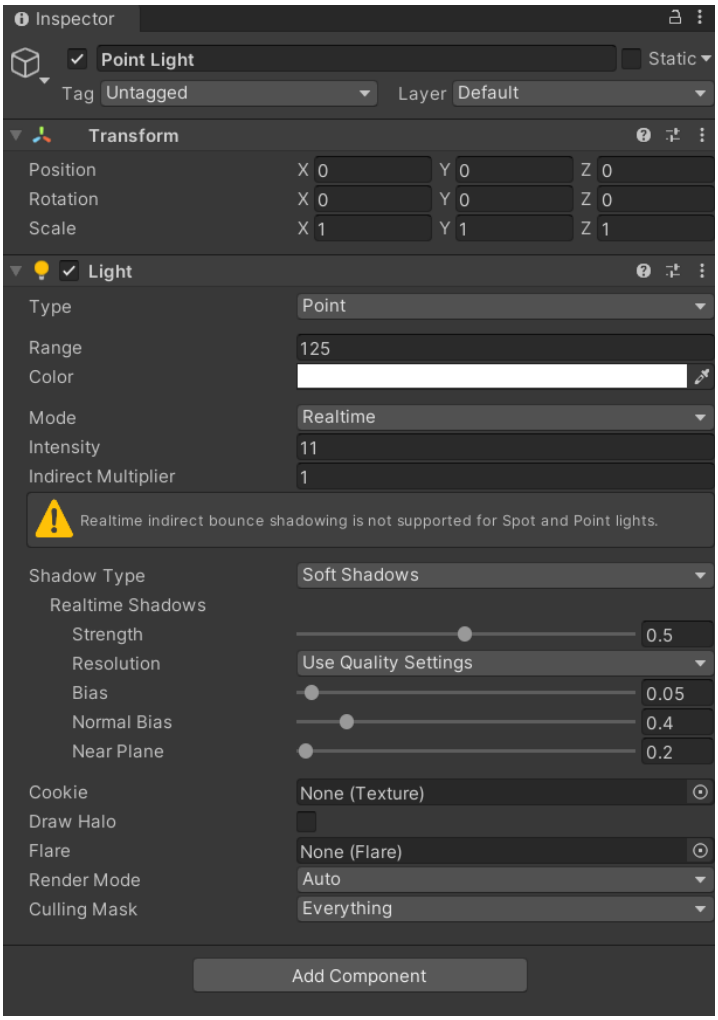
Κάναμε την ακτίνα του Ήλιου 15 με scale (15,15,15)



Ο ίδιος ο Ήλιος είναι φωτεινός, καθώς ενεργοποιήσαμε το Emission στο Material, όπως φαίνεται στον Inspector

Ο Ήλιος φωτίζει τους υπόλοιπους πλανήτες μέσω ενός Point Light (`Δεξί κλικ -> Light -> Point Light`)

Το τοποθετήσαμε στο κέντρο του Ηλίου και του δώσαμε τις ρυθμίσεις που φαίνονται στο screenshot

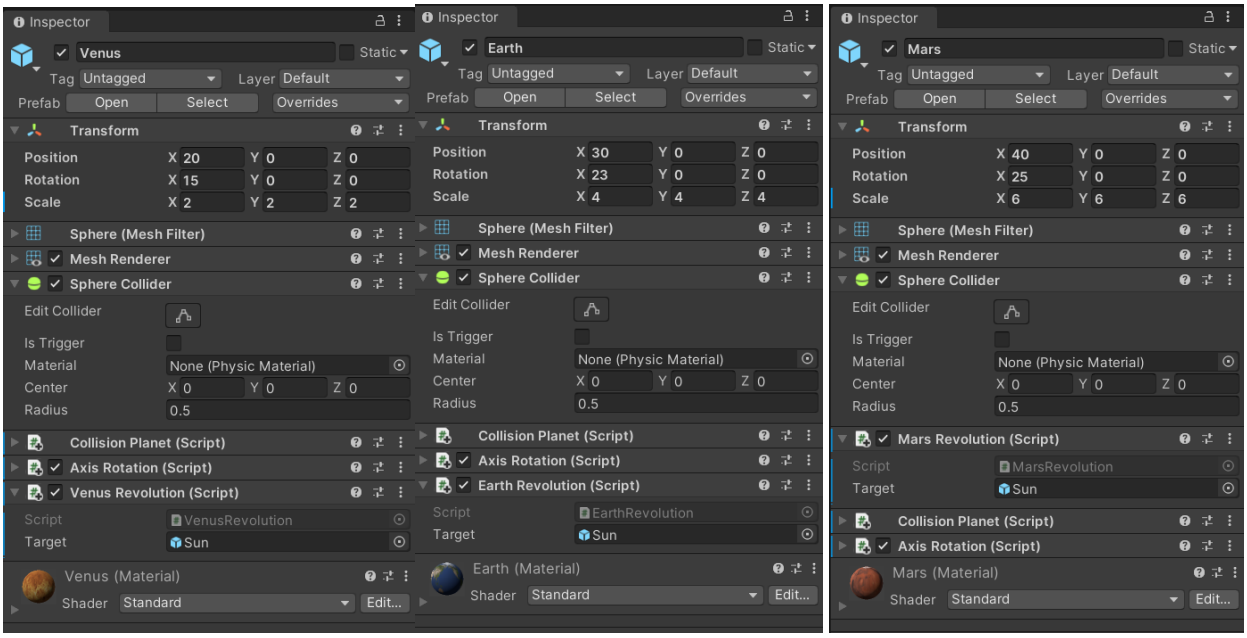


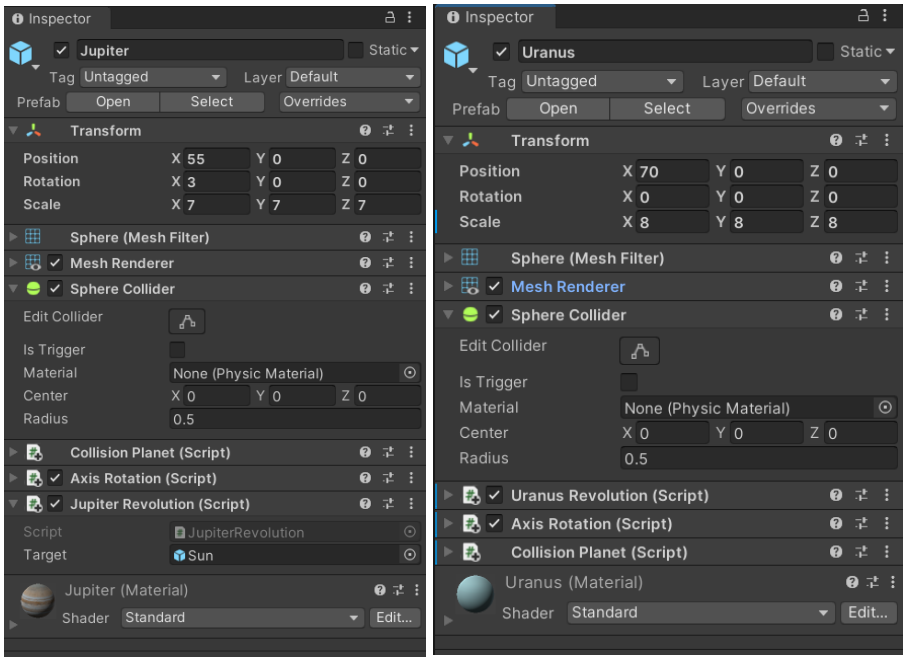
Ερώτημα (iii)

Φόρτωση 5 πλανητών

Προσθέσαμε 5 πλανήτες (Venus, Earth, Mars, Jupiter, Uranus) από τα prefabs του tutorial: learn.unity.com/tutorial/zoe-prepare-your-scene

Για να τους δώσουμε τις ακτίνες που μας ζητάει η εκφώνηση, κάναμε scale με την αντίστοιχη ακτίνα. Ακολουθούν screenshots από τον inspector:

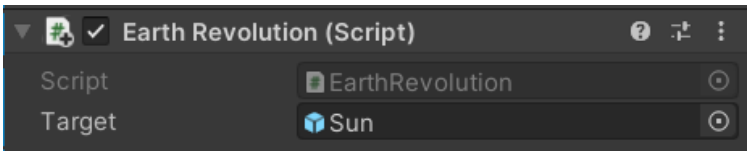




Κίνηση γύρω από τον Ήλιο

Η κίνηση γύρω από τον Ήλιο για τον κάθε πλανήτη γίνεται με τα script Revolution

Κάθε φορά που εφαρμόζουμε το script περνάμε ως GameObject Target τον Ήλιο, όπως φαίνεται στο παρακάτω screenshot:



Παρακάτω έχουμε βάλει τον κώδικα για το EarthRevolution και το MarsRevolution. Για τους άλλους 3 πλανήτες είναι ακριβώς ο ίδιος κώδικας με την μόνη διαφορά την ταχύτητα περιστροφής, δηλαδή το τρίτο όρισμα της RotateAround

`EarthRevolution.cs`

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

public class EarthRevolution : MonoBehaviour
{
    // Assign a GameObject in the Inspector to rotate around
    public GameObject target;

    // Update is called once per frame
    void Update()
    {
        // Spin the object around the target at XX degrees/second.
        transform.RotateAround(target.transform.position, Vector3.up, 22.5f *
Time.deltaTime);
    }
}
```

MarsRevolution.cs

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

public class MarsRevolution : MonoBehaviour
{
    // Assign a GameObject in the Inspector to rotate around
    public GameObject target;

    // Update is called once per frame
    void Update()
    {
        // Spin the object around the target at XX degrees/second.
        transform.RotateAround(target.transform.position, Vector3.up, -25f
* Time.deltaTime);
    }
}
```

Κίνηση γύρω από τον εαυτό τους (bonus b)

Για να κινούνται οι πλανήτες γύρω από τον εαυτό τους, εφαρμόσαμε το AxisRotation script, το οποίο ουσιαστικά τους περιστρέφει γύρω από τον y άξονα.

AxisRotation.cs

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

public class AxisRotation : MonoBehaviour
{
    // Update is called once per frame
    void Update()
    {
        transform.Rotate(0, 0.1f, 0);
    }
}
```

Ερώτημα (iv)

First Person Camera

Για να φτιάξουμε την First Person Camera ακολουθήσαμε τις οδηγίες των links:

- FIRST PERSON MOVEMENT in Unity - FPS Controller
- Creating a Spectator Camera in Unity
- Moving the Camera Up and Down by pressing Key
- Moving in the direction of mouse

Επομένως, έχουμε ένα GameObject, τον FirstPersonPlayer

Στον FirstPersonPlayer εφαρμόζουμε το script PlayerMovement, ώστε να μπορούμε να κινούμαστε με πλήκτρα

- **W**, πάει μπροστά
- **S**, πάει πίσω
- **A**, πάει αριστερά
- **D**, πάει δεξιά
- **W + κοιτάζει η κάμερα προς τα πάνω**, πάει μπροστά και πάνω
- **W + κοιτάζει η κάμερα προς τα κάτω,** πάει μπροστά και κάτω
- **S + κοιτάζει η κάμερα προς τα πάνω,** πάει πίσω και κάτω
- **S + κοιτάζει η κάμερα προς τα κάτω,** πάει σπίσω και πάνω
- **Q**, πάει γρήγορα προς τα πάνω
- **E**, πάει γρήγορα προς τα κάτω

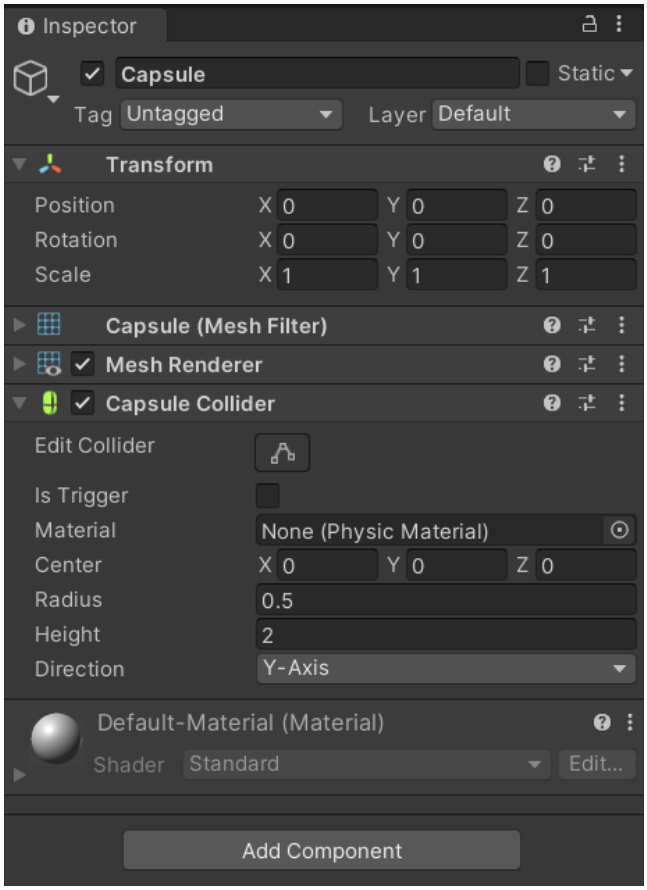
PlayerMovement.cs

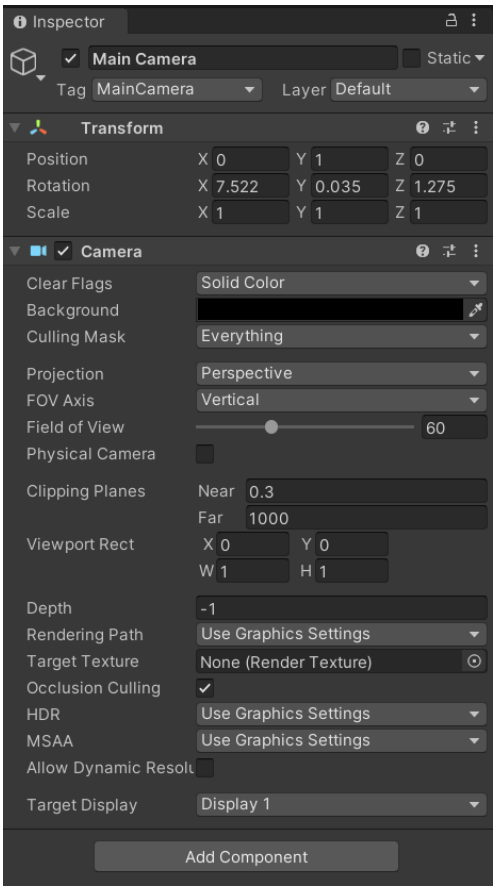
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class PlayerMovement : MonoBehaviour
{

    public CharacterController controller;
    public Transform lookTarget;
    public float speed = 12f;
    public float verticalSpeed = 200f;
    public float mouseSensitivity = 3f;
    float yaw;
    float pitch;

    void Start() {
        Cursor.lockState = CursorLockMode.Locked;
        // force player to look at the target object
        transform.LookAt(lookTarget.transform);
    }

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

        // Get input
        float x = Input.GetAxis("Horizontal");
        float z = Input.GetAxis("Vertical");

        // Movement according to WASD keys
        Vector3 move = transform.right * x + transform.forward * z;
        controller.Move(move * speed * Time.deltaTime);

        // Mouse look + Movement when W,D are pressed
        yaw += Input.GetAxis("Mouse X") * mouseSensitivity;
        pitch -= Input.GetAxis("Mouse Y") * mouseSensitivity;
        pitch = Mathf.Clamp(pitch, -90f, 90f);
        transform.rotation = Quaternion.Euler(pitch, yaw, 0);
```

```
// Move Camera up and down with Q,E
        if (Input.GetKey(KeyCode.Q))
        {
            transform.position += Vector3.up * verticalSpeed *
Time.deltaTime;
        }else if (Input.GetKey(KeyCode.E))
        {
            transform.position += Vector3.down * verticalSpeed *
Time.deltaTime;
        }

    }
}
```

Επιπλέον, ο FirstPersonPlayer έχει ως σώμα μια Capsule και βλέπει με την Main Camera. Η Main Camera έχει ως παιδί το Gun, το οποίο θα αναλύσουμε στην επόμενη ενότητα.

Εξαιτίας του σώματος-κάψουλας ο παίχτης συγκρούεται με τους πλανήτες, όταν πέφτει πάνω τους. Δεν μπορεί δηλαδή να περάσει από μέσα τους.





Ερώτημα (iv)

Δημιουργία Μετεωρίτη

Για τον μετεωρίτη δημιουργήσαμε ένα prefab με βάση αυτά που είχε το tutorial learn.unity.com/tutorial/zoe-prepare-your-scene

και χρησιμοποιήσαμε κώδικα από:

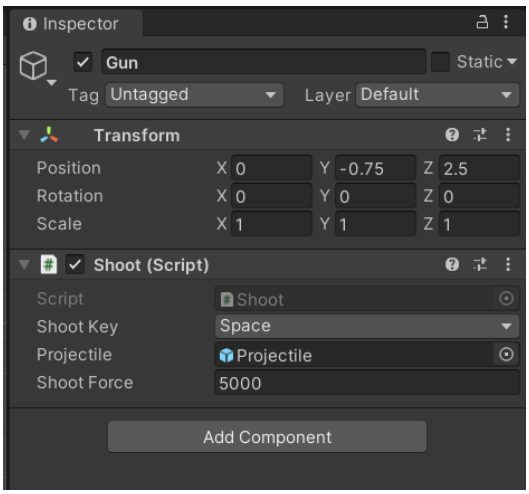
- το tutorial Shoot a projectile with Instantiate
- τις απαντήσεις στο How to fire projectile in direction character is facing?

για να δημιουργήσουμε το script Shoot

Shoot.cs

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

public class Shoot : MonoBehaviour
{
    public KeyCode shootKey = KeyCode.Space;
    public GameObject Projectile;
    public float shootForce;
    private Vector3 scaleChange;

    // Get random scale number in range [1.0,5.0]
    public float getRandomNumberInRange()
    {
        float max = 5.0f;
        float min = 1.0f;

        System.Random rnd = new System.Random();
        double value = rnd.NextDouble() * (max - min) + min;
        return (float)value;
    }
```

```
// Update is called once per frame
    void Update()
    {
        if (Input.GetKeyDown(shootKey))
        {
            // Create Projectile
            GameObject shot = GameObject.Instantiate(Projectile,
transform.position, transform.rotation);

            // Scale Projectile
            float radius = getRandomNumberInRange();
            scaleChange = new Vector3(radius, radius, radius);
            shot.transform.localScale = scaleChange;

            // Move Projectile
            shot.GetComponent<Rigidbody>().AddForce(transform.forward *
shootForce);

            // Destroy the projectile after 3 seconds
            Destroy(shot, 3f);
        }
    }
}
```



Εφαρμόσαμε το script στο Gun και τοποθετήσαμε ως projectile το αντίστοιχο prefab.

Τοποθετήσαμε το Gun πιο μπροστά το σώμα-κάψουλα του FirstPersonPlayer. Λάβαμε αυτήν την απόφαση, έτσι ώστε να μην χτυπά ο μετεωρίτης πάνω στο σώμα του παίχτη σε δυο περιπτώσεις.
Η πρώτη περίπτωση είναι, αν βρισκόταν σε περίεργη γωνία κατά την εκτόξευση.
Η δεύτερη περίπτωση είναι, αν ο ίδιος ο μετεωρίτης ήταν πολύ μεγάλος, όταν γεννιόταν και ακουμπούσε κατευθείαν το σώμα του παίχτη

Αν ο μετεωρίτης συνεχίζει να ζει μετά από 3 δευτερόλεπτα από την δημιουργία του, τότε καταστρέφεται αυτόματα.

Σύγκρουση μετεωρίτη με πλανήτη

Για να ελέγξουμε την σύγκρουση του μετεωρίτη με κάποιον από τους 5 πλανήτες ή τον Ήλιο, φτιάξαμε το CollisionPlanet script.

Το εφαρμόζουμε στους 5 πλανήτες και τον Ήλιο.

`CollisionPlanet.cs`

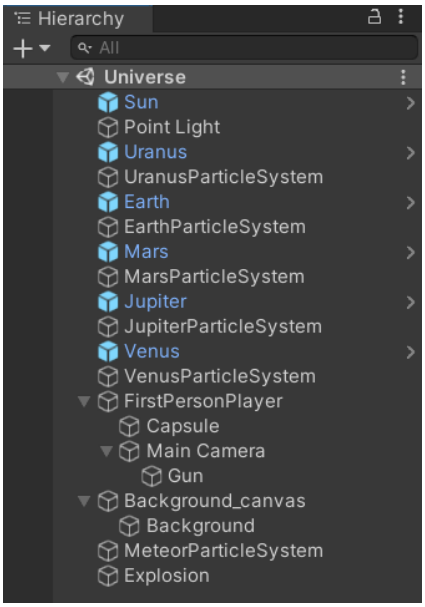
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CollisionPlanet : MonoBehaviour
{
    public ParticleSystem planetParticles;
    public ParticleSystem meteorParticles;
    public AudioSource collisionSound;

    void OnCollisionEnter(Collision collision)
    {
        //destroy meteor on impact
        collisionSound.Play();
        meteorParticles.transform.position =
collision.gameObject.transform.position;
        meteorParticles.Play();
        GameObject.Destroy(collision.gameObject);

        //destroy the gameObject(planet) if its not the Sun
        if (gameObject.name != "Sun")
        {
            planetParticles.Play();
            GameObject.Destroy(gameObject);
        }
    }
}
```

Particles κατά την έκρηξη

Όταν ο μετεωρίτης συγκρούεται με έναν πλανήτη, τότε γίνεται έκρηξη όπου σπάνε τα δύο αυτά σώματα σε περισσότερα και μετά από λίγο εξαφανίζονται και τα δύο. Σε περίπτωση που ο μετεωρίτης συγκρουστεί μόνο με τον Ήλιο, τότε το παραπάνω γίνεται μόνο στον μετεωρίτη.

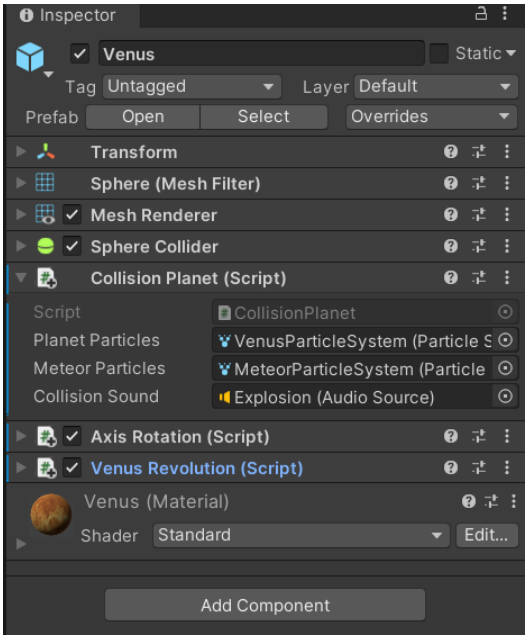
Εφαρμόσαμε την παραπάνω έκρηξη με particle system, εμπνευσμένοι από το video [Unity Object Collision and Sound and Particle System]

Δημιουργήσαμε 5 particle systems για καθένα από τους πλανήτες και 1 particle system για τον μετεωρίτη. Εφαρμόσαμε τις κατάλληλες ρυθμίσεις, ώστε να σπάνε σε κομματάκια τα σώματα.



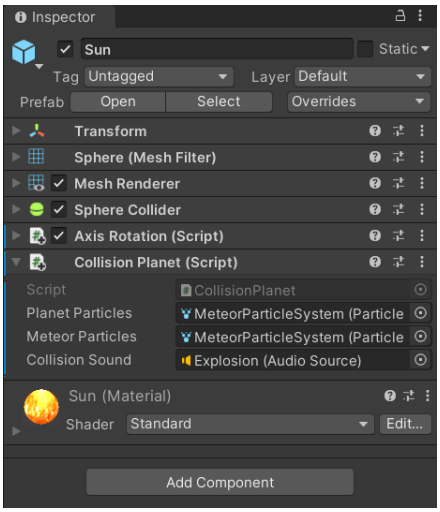
Όταν γίνεται η σύγκρουση, στο script CollisionPlanet, μεταφέρουμε το MeteorParticleSystem στην θέση που έχει ο μετεωρίτης εκείνη την στιγμή και το αφήνουμε να παίξει.

Για τους πλανήτες έχουμε εφαρμόσει σε κάθε Particle System το αντίστοιχο Revolution, έτσι ώστε να κινούνται στην ίδια θέση με τον πλανήτη. Με την σύγκρουση ο πλανήτης εξαφανίζεται και το αντίστοιχο Particle System κάνει play.



Σε έναν πλανήτη, πχ το Venus, το script CollisionPlanet εφαρμόζεται με:

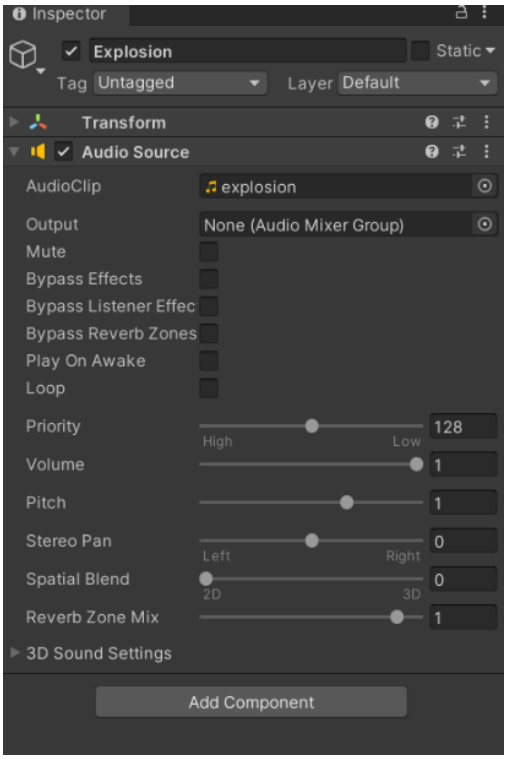
- Planet Particles: το Particle System του ίδιου του πλανήτη, πχ VenusParticleSystem
- Meteor Particles: το MeteorParticleSystem



Στον Ήλιο που δεν έχει Particle System βάζουμε ως Planet Particles ξανά τα MeteorParticleSystem

Εφέ ήχου κατά την έκρηξη (μισό bonus a)

Στο CollisionPlanet script έχουμε και έναν Collision Sound στο οποίο φορτώνουμε το Audio Source “Explosion”



Για να δημιουργήσουμε το audio source στο project μας, κάναμε Δεξί κλικ -> Audio -> Audio Source και προσθέσαμε στο AudioClip ένα explosion.mp3 αρχείο

Επιπλέον, για να ακούμε τον ήχο, όταν παίζει, χρειάστηκε να προσθέσουμε στο FirstPersonPlayer το component Audio Listener

Checksums

File path and name: Project2_Universe.zip

MD5: B4E84AF98516D9B6F629C1FA9EC354FD

SHA-1: AD926555656B133CB1BD203D5B558E877B8E6CB7

SHA-256: A7D39C1E2AF5F24CA10052B9E6320D4A3BEFF569C54C0122601ECFF30AC96BAE

SHA-512:
30F250647C035CB51A899713CF70B27493AE1C3F09A447A7C83859D958AAA94AB489AF114B3
C70D8C3060E34C9A511317F593E96160A686D0C9449D2D687B1CB

RIPEMD: 9E24F9A9F04C786553232C46B79C7F28D865FD9E