

Γραφικά Υπολογιστων και συστήματα αλληλεπίδρασης

Προγραμματιστική άσκηση 2 (Unity 3D)

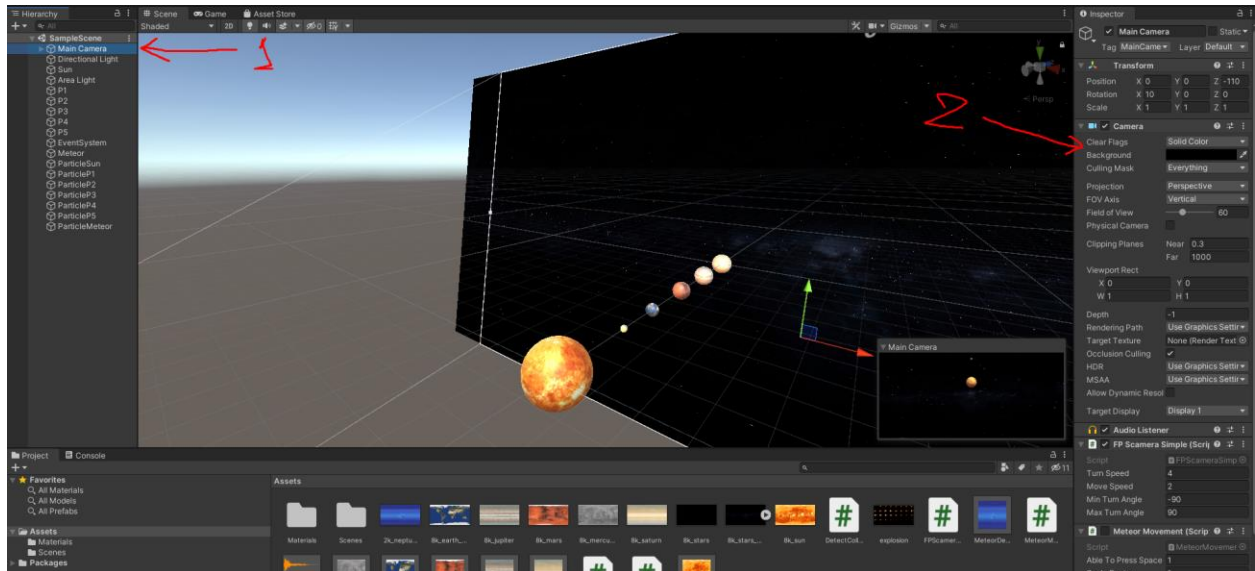
Νεφέλη-Ελένη Κατσιλέρου 4385

Χρήστος Καραγιαννίδης 4375

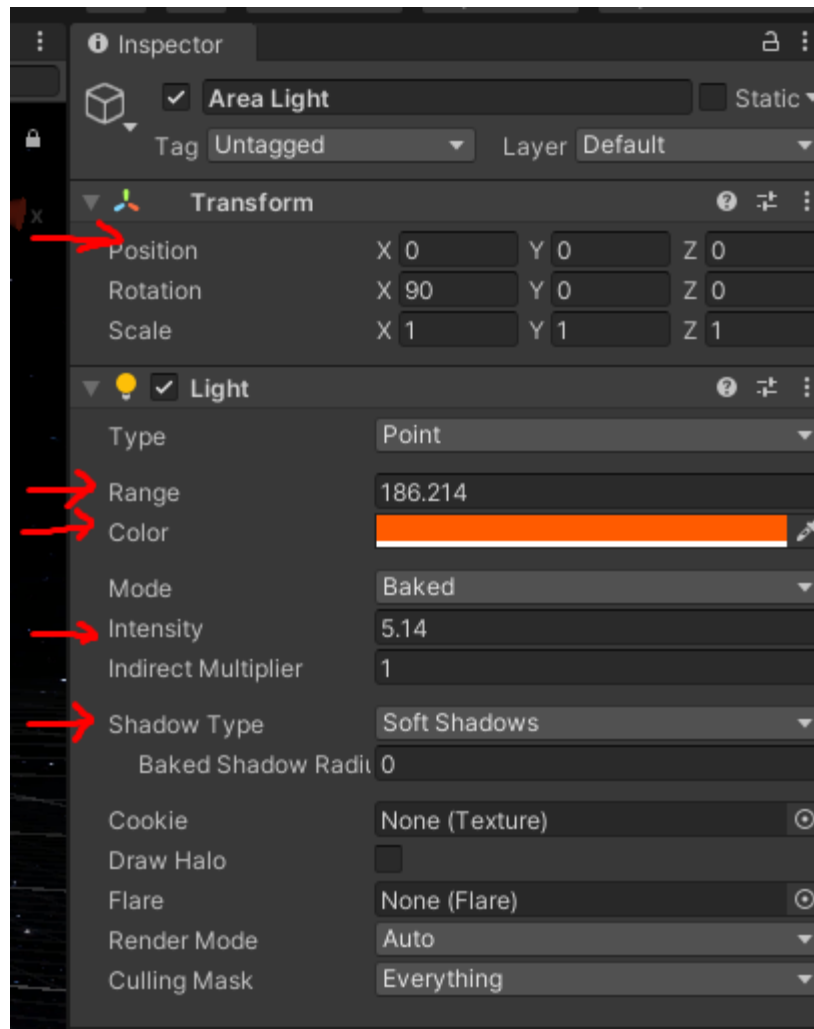
MD5: BAB53AA1B4A2CCD7F3821372B8FB888F

Σχολια:

- 1) Η εργασία έγινε σε λειτουργικό windows 10.
 - 2) Απο την στιγμή που βαλαμε τον ηχο κατα την εκρηξη μειωθηκε η ταχυτητα του μετεωριτη και δεν ξερουμε για ποιον λογο.
 - 3) ΚΑΤΑ ΤΗΝ ΕΝΑΡΞΗ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ ΑΚΟΥΓΕΤΑΙ Ο ΗΧΟΣ ΤΗΣ ΕΚΡΗΞΗΣ γιατι αν δεν ειχαμε το PlayOnAwake = true δεν επαιζε καθολου ο ηχος για καποιον λογο
 - 4) Ο ηχος εχει ενα μικρο delay απο την στιγμή που συγκρουονται οι πλανητες που κατα πασα πιθανοτητα ευθυνεται στο κακο audio file που εχει κενο χρονο στην αρχη.
-
- i) Για να φτιαξουμε την εφαρμογη πηγαμε στο περιβαλλον της unity File > Build Settings > PC,Mac & Linux Standalone > Player Settings... > Player. Εδω αλλαξαμε το ονομα στο πεδιο product name και κατω απο το tab resolution and presentation το full screen mode σε windowed και width/height σε 1024x768. Το background το αλλαξαμε πατωντας την Main Camera στο Hierarchy tab και διπλα στον Inspector αλλαξαμε το Background σε μαυρο (Ωστοσο εχουμε κανει και το μπονους c με το background διαστηματος οποτε δεν φαινεται τωρα).



- ii) Για να δημιουργήσουμε μια σφαίρα πηγαίνουμε στο GameObject > 3D object > Sphere το οποίο εμφανίζει μια σφαίρα (αρχικά τον ήλιο). Πατώντας πάνω της στο tab Inspector αλλάξαμε τα πεδία Position (για την θέση του Ηλίου) και Scale (για να μεγαλώσουμε την ακτίνα του)*. Για παράδειγμα για τον ήλιο το Position ήταν 0,0,0 και το Scale 30,30,30 γιατί αν το κάναμε 15,15,15 θα είχαμε ακτίνα 7.5 γιατί οι παραμετροί του Scale αναφέρονται στην διάμετρο. Για να βάλουμε texture κάναμε drag and drop την αντιστοιχη εικόνα .jpg στην περιοχή των assets και μετά drag and drop πάνω στον Ηλιο . (Σημείωση: Στον μετεωριτη βάλαμε το texture του Ποσειδωνα γιατί δεν βρήκαμε texture από μετεωριτη και γιατί ξεχωριζε αρκετά λόγω χρωματος) . Τέλος προσθέσαμε ένα Area light από το GameObject > light > area light αλλάξαμε το χρώμα του , το μετατοπίσαμε στο κέντρο του ηλίου (0,0,0) αλλάξαμε το range και το Intensity του και βάλαμε να κάνει cast soft

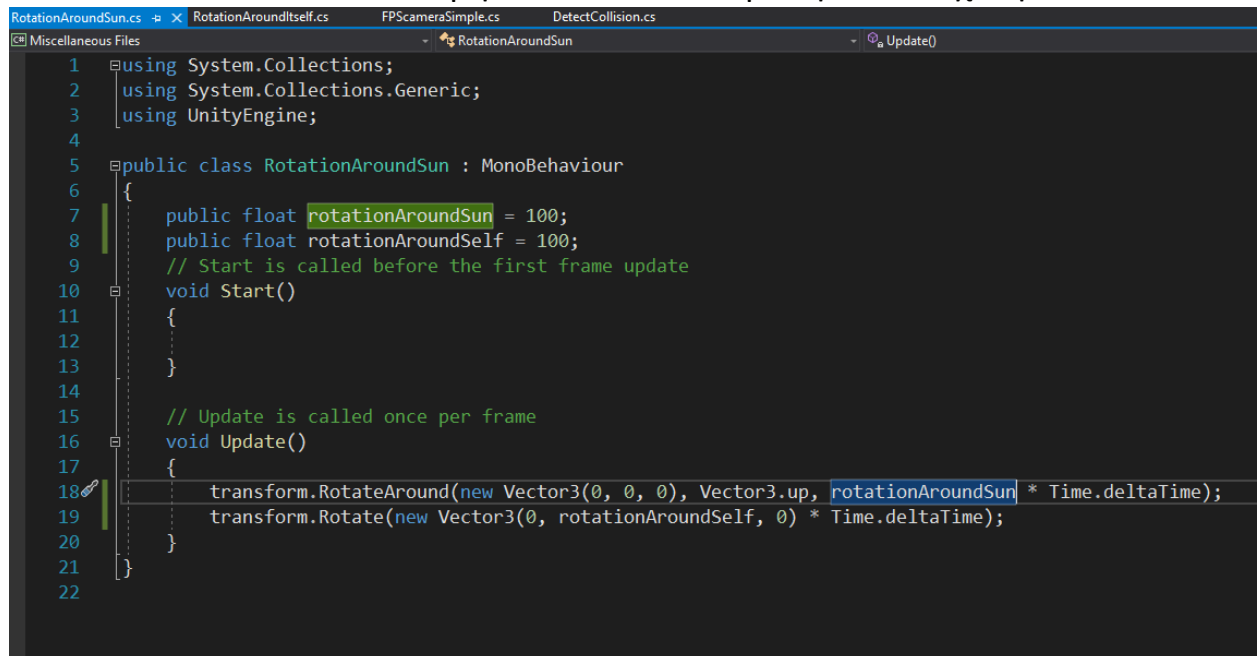


shadows.

- iii) Κανουμε την ιδια διαδικασια με το (ii) ερωτημα για οσους πλανητες χρειαζομασταν (7 συνολο : 1 Ηλιο 5 Πλανητες 1 Μετεωριτη). Δηλαδη
- P1 Scale (4,4,4)
 - P2 Scale (8,8,8)
 - P3 Scale (12,12,12)
 - P4 Scale (14,14,14)
 - P5 Scale (16,16,16)

Για τις κινήσεις των πλανητων φτιαξαμε ενα script με ονομα RotationAroundSun.cs στο οποιο κανουμε περιστροφη γυρω απο τον ηλιο με την transform.RotateAround με ταχυτητα rotationAroundSun

γυρω απο τον αξονα y . (Επισης κανουμε και την περιστροφη γυρω απο τον εαυτο τους με την transform.Rotate με ταχυτητα rotationAroundSelf γυρω απο τον αξονα y αλλα θα το αναφερουμε και παρακατω που θα μιλησουμε για τα bonus). Βαλαμε τις εντολες μας στην void update() γιατι θελουμε να γινονται με καθε αλλαγη frame (60 fps).Καναμε drag and drop το script σε καθε object/planet που θελουμε να περιστρεφεται (P1-P5) και αλλαξαμε απο το πεδιο Inspector τις τιμες του καθε πλανητη για τις μεταβλητες RotationAroundSun και RotationAroundSelf ωστε να μην κινουνται ολοι με την ιδια ταχυτητα.



```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class RotationAroundSun : MonoBehaviour
6 {
7     public float rotationAroundSun = 100;
8     public float rotationAroundSelf = 100;
9     // Start is called before the first frame update
10    void Start()
11    {
12    }
13
14    // Update is called once per frame
15    void Update()
16    {
17        transform.RotateAround(new Vector3(0, 0, 0), Vector3.up, rotationAroundSun * Time.deltaTime);
18        transform.Rotate(new Vector3(0, rotationAroundSelf, 0) * Time.deltaTime);
19    }
20
21 }
22
```

- iv) Για την υλοποιηση της καμερας βρηκαμε αρκετες ιδεες απο το google και υπηρχανε και πολλες υλοποιησεις 150+ γραμμων που δεν καταλαβαιναμε οποτε κρατησαμε οτι καταλαβαιναμε και μας ταιριαζε ωστε να βγει κατανοητα η κινηση με βαση αυτο που ζητειται στην εκφωνηση. Καναμε ενα script στο οποιο η κινηση γινεται με την χρηση ποντικιου για το «που κοιταει» η καμερα και με τα πληκτρα w,s κινειται κατα μηκος του αξονα που βλεπει και με τα a,d αριστερα και δεξια (καθετα στον αξονα που βλεπει). Μετα καναμε drag and drop το script στο gameObject Main camera.

```
RotationAroundSun.cs  RotationAroundItself.cs  FPScameraSimple.cs  DetectCollision.cs
Miscellaneous Files  FPScameraSimple  minTurnAngle

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class FPScameraSimple : MonoBehaviour
6  {
7      public float turnSpeed = 1.2f;
8      public float moveSpeed = 15.0f;
9      public float minTurnAngle = -90.0f;
10     public float maxTurnAngle = 90.0f;
11     private float rotX;
12     void Start()
13     {
14         turnSpeed = 1.2f;
15         moveSpeed = 15f;
16     }
17     void Update()
18     {
19
20         //kinisi me pontiki
21         float y = Input.GetAxis("Mouse X") * turnSpeed;
22         rotX += Input.GetAxis("Mouse Y") * turnSpeed;
23         rotX = Mathf.Clamp(rotX, minTurnAngle, maxTurnAngle);
24         transform.eulerAngles = new Vector3(-rotX, transform.eulerAngles.y + y, 0);
25         //kinisi me pliktrtologio
26         Vector3 dir = new Vector3(0, 0, 0);
27         dir.x = Input.GetAxis("Horizontal");
28         dir.z = Input.GetAxis("Vertical");
29         transform.Translate(dir * moveSpeed * Time.deltaTime);
30     }
31 }
32
33 }
```

- v) Με τον ίδιο τρόπο που καναμε ολους τους πλανητες φτιαξαμε και τον μετεωριτη και του βαλαμε και το texture. Φτιαξαμε ενα script για την κινηση του μετεωριτη (MeteorMovement.cs) και ενα για την ανιχνευση συγκρουσης(DetectCollision.cs) . Για την κινηση του μετεωριτη ελεγχουμε αν πατησει ο χρηστης το κουμπι space και αν ναι και ταυτοχρονα του επιτρεπουμε να πατησει το κουμπι space (μεταβλητη ableToPressSpace) τοτε :
- 1) κανουμε το ableToPressSpace = 0 ωστε να μη μπορει να ξαναπατησει μεχρι να ολοκληρωθει η κινηση
 - 2) διαμορφωνουμε την ακτινα του μετεωριτη κανοντας τον scale με

random τιμη στο [1,5].

3) κανουμε την μεταβλητη meteorMove = 1 (η οποια ξεκιναι την κινηση του μετεωριτη)

4) μετατοπιζουμε τον μετεωριτη στην θεση της καμερας

5) αποθηκευουμε την κατευθυνση που κοιταει η καμερα σε ενα διανυσμα με ονομα meteorDirection.

Οσο η μεταβλητη meteorMove ειναι ιση με 1 :

1) Ο μετεωριτης κινειται απο την θεση εναρξης(θεση της καμερας την στιγμη που πατηθηκε το space) προς την κατευθυνση που εχει αποθηκευμενη το διανυσμα meteorDirection (το που κοιτουσε η καμερα οταν πατηθηκε το space).

2) κραταμε με καθε μετατοπιση την θεση του μετεωριτη σε ενα διανυσμα meteorPosition

3) κραταμε στην μεταβλητη distanceFromSun την αποσταση του μετεωριτη απο τον ηλιο (ωστε να μην κινειται επ απειρο)

4) Οταν ο μετεωριτης μπει σε ακτινα 100 μοναδων απο τον ηλιο τοτε κανουμε την μεταβλητη teleiwnei_i_venzini ιση με 1 (η οποια ξεκιναι μια «αντιστροφη μετρηση» για το ποτε θα σταματησει ο μετεωριτης)

5) Οσο εχουμε οτι teleiwnei_i_venzini == 1 μειωνεται η μεταβλητη distanceLeft κατα 0.1 (ανα frame) δηλαδη μειωνεται η αποσταση που θα μπορεσει να διανυσει ο μετεωριτης μεχρι να σταματησει

6) οταν η μεταβλητη αυτη γινει <= 0 τοτε κανουμε την μεταβλητη meteorMove = 0 ωστε να σταματησει η κινηση του

μετεωριτη, επιτρεπουμε στον χρηστη να ξαναπατησει το space (για να ξαναξεκινησει η κινηση του μετεωριτη) , μετατοπιζουμε τον μετεωριτη στο κεντρο του ηλιου , τον κανουμε scaleDown ωστε να μην κανει scaleup μονο , κανουμε reset το distanceLeft και την μεταβλητη teleiwnei_i_venzini = 0. Πρακτικα κανουμε reset ολες τις μεταβλητες που σχετιζονται με την κινηση ωστε να μπορεσει να επαναληφθει.

7) Αν η αποσταση του μετεωριτη απο τον ηλιο ξεφυγει γιατι ο χρηστης «στοχευσε» προς την αλλη μερια τοτε εχουμε βαλει ενα ανω οριο 1000 μοναδων ωστε να μην κινειται επ απειρο.

```

6 {
7     public int ableToPressSpace = 1;
8     public float scaleFactor = 1;
9     public int meteorMove = 0;
10    public float distanceLeft = 220f;
11    public Vector3 MeteorPosition;
12    public Vector3 meteorDirection;
13    public static Vector3 SunPosition = new Vector3(0.0f,0.0f,0.0f);
14    public float distanceFromSun = 0.0f;
15    public int teleiwnei_i_venzini = 0;
16
17    // Start is called before the first frame update
18    void Start()
19    {
20    }
21
22
23    // Update is called once per frame
24    void Update()
25    {
26
27        if (Input.GetKey(KeyCode.Space) && ableToPressSpace == 1)
28        {
29            ableToPressSpace = 0;
30            scaleFactor = Random.Range(1.0f, 5.0f);
31            transform.localScale += new Vector3(scaleFactor,scaleFactor,scaleFactor);
32            meteorMove = 1;
33            transform.position = Camera.main.transform.position;
34            meteorDirection = Camera.main.transform.forward;
35            DetectCollision.inSun = 0;
36        }
37
38        if (meteorMove ==1)
39        {
40            transform.position = transform.position + meteorDirection * 0.1f;
41            MeteorPosition = GameObject.Find("Meteor").transform.position;
42
43            distanceFromSun = Vector3.Distance(MeteorPosition, SunPosition);
44            if (distanceFromSun < 100) ...
49            if (teleiwnei_i_venzini == 1) ...
53            if (distanceLeft <= 0.0)
54            {
55                meteorMove = 0;
56                ableToPressSpace = 1;
57                DetectCollision.inSun = 1;
58                transform.position = SunPosition;
59                transform.localScale -= new Vector3(scaleFactor, scaleFactor, scaleFactor);
60                distanceLeft = 220f;
61                teleiwnei_i_venzini = 0;
62            }
63            if (distanceFromSun > 1000)
64            {
65                teleiwnei_i_venzini = 1;
66            }
67        }
68    }
69

```

Για την ανίχνευση της συγκρούσης βάλαμε σε κάθε πλανήτη , στον μετεωριτη και στον ήλιο , έναν collider πατώντας στον inspector του κάθε gameObject Add Component > sphere collider . Στο αρχείο DetectCollision.cs χρησιμοποιήσαμε την συνάρτηση OnTriggerEnter() που ανιχνεύει συγκρούσεις ανάμεσα σε 2 colliders. Σε αυτό το σημείο ανιχνευόνταν 2 συγκρούσεις του μετεωριτη με τον ήλιο (1 όταν αυτός πηγαινε στο κέντρο του και 1 όταν οντως συγκρουόνταν με αυτόν) .Γι αυτό φτιάξαμε στο αρχείο MeteorMovement μια public static μεταβλητή με όνομα inSun η οποία γίνεται 1 πριν μετατοπιστεί ο μετεωριτης στο κέντρο του ήλιου (ώστε να απορριψουμε την μια συγκρούση) . Όταν λοιπόν ο collider του μετεωριτη συγκρούεται με κάποιον άλλο collider κάνουμε «print» ότι ανιχνευτήκε η συγκρούση και κάνουμε deactivate τους 2 πλανήτες θέτοντας το πεδίο SetActive = false. Αρα μέχρι στιγμής έχουμε ανίχνευση συγκρούσης και εξαφάνιση των πλανητών.


```

6 public class DetectCollision : MonoBehaviour
7 {
8     public static int inSun = 1;
9     public Text scoreText ;
10    public int counter = 0;
11    public GameObject ParticleSun;
12    public GameObject ParticleP1;
13    public GameObject ParticleP2;
14    public GameObject ParticleP3;
15    public GameObject ParticleP4;
16    public GameObject ParticleP5;
17    public GameObject ParticleMeteor;
18    public int SunActive = 1;
19    public int P1Active = 1;
20    public int P2Active = 1;
21    public int P3Active = 1;
22    public int P4Active = 1;
23    public int P5Active = 1;
24    public int MeteorActive = 1;
25    private AudioSource boom;
26
27
28
29    void Start()...
41    private void OnTriggerEnter(Collider other)
42    {
43        if (!(inSun == 1))
44        {
45            Debug.Log("hit");
46            counter++;
47            boom.Play();
48            other.gameObject.SetActive(false);
49            gameObject.SetActive(false);
50            scoreText.text = counter.ToString();
51            if(other.gameObject.name == "Sun" && SunActive == 1)...
58            if (other.gameObject.name == "P1" && P1Active == 1)...
65            if (other.gameObject.name == "P2" && P2Active == 1)...
72            if (other.gameObject.name == "P3" && P3Active == 1)...
79            if (other.gameObject.name == "P4" && P4Active == 1)...
86            if (other.gameObject.name == "P5" && P5Active == 1)...
93            Invoke("setActiveMeteor", 3f);
94
95
96
97        }
98    }
99

```

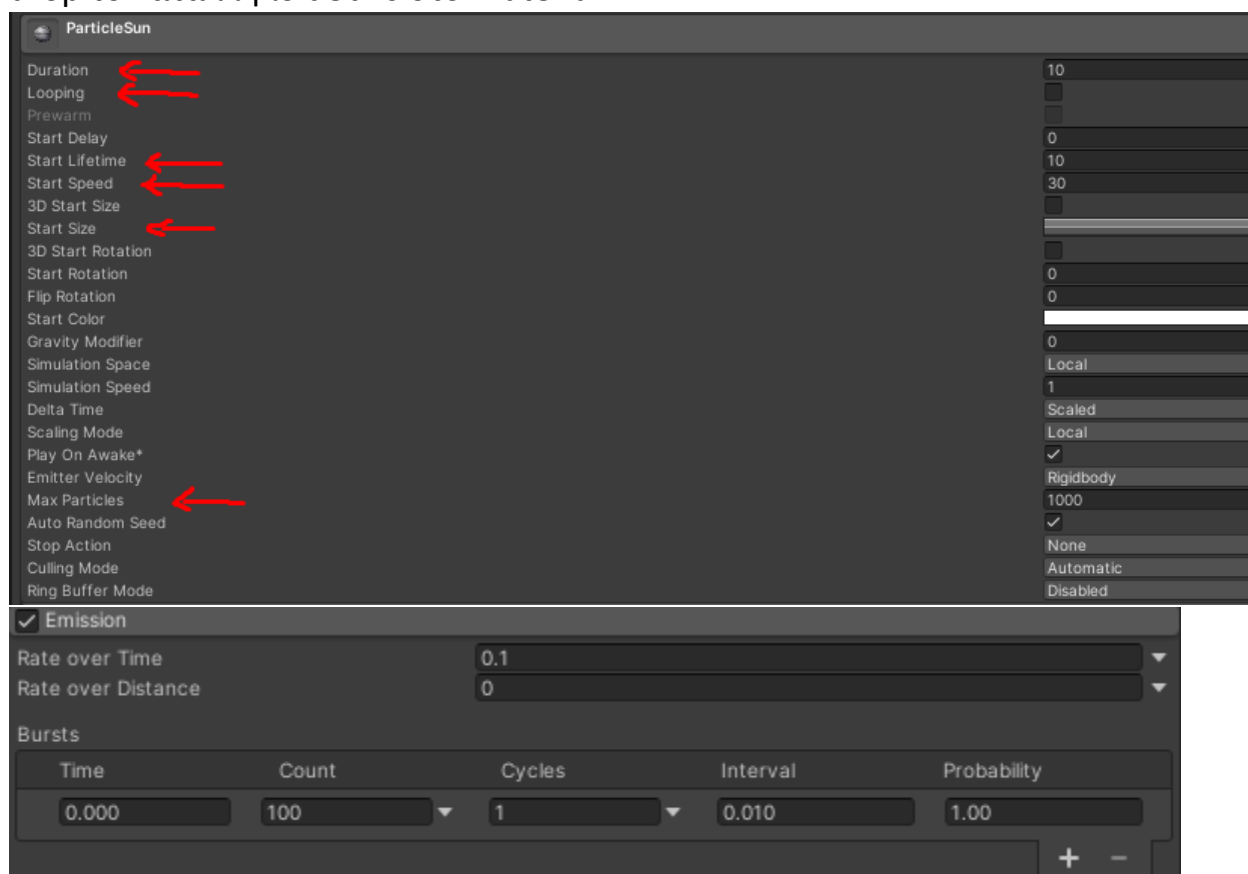
Για την εκρηξη σε θραυσματα φτιαξαμε 7 materials (ενα για καθε πλανητη) στο πεδιο με τα assets πατωντας δεξι κλικ >create > material και κανοντας drag and drop την εικονα .jpg του καθε πλανητη στο καθε material στο πεδιο albedo του παραθυρου Inspector και αλλαξαμε τον shader απο standard σε sprites> default. Ετσι δημιουργησαμε τα 7 αρχεια με την ονομασια debris (πχ SunDebris, P1Debris ...). Μετα πατησαμε GameObject > effects > particle System και δημιουργησαμε 7 ιδια (σε παραμετρους) particle systems το καθενα να αντιπροσωπευει τον καθε πλανητη και να εχει το καταλληλο debris material. Αλλαξαμε τα πεδια

Duration , looping , Start Lifetime, Start Speed , Start Size, Max Particles
Emission : rate over Time, Bursts

Shape : Shape

Collision : Type , Mode , Dampen, Bounce

Renderer : RenderMode (σε mesh) , Mesh (σε sphere) , καναμε drag and drop το καταλληλο debris στο Material.



✓ Shape

Shape Sphere

Radius 1

Radius Thickness 1

Arc 360

Mode Random

Spread 0

Texture None (Texture 2D)

Position X 0 Y 0 Z 0

Rotation X 0 Y 0 Z 0





Scale X 1 Y 1 Z 1

Align To Direction ☐

Randomize Direction 1

Spherize Direction 0

Randomize Position 0

✓ Collision

Type World

Mode 3D

Dampen 0.6

Bounce 0.2

Lifetime Loss 0

Min Kill Speed 0

Max Kill Speed 10000

Radius Scale 1

Collision Quality High

Collides With Everything

Max Collision Shapes 256

Enable Dynamic Colliders ☒

Collider Force 0

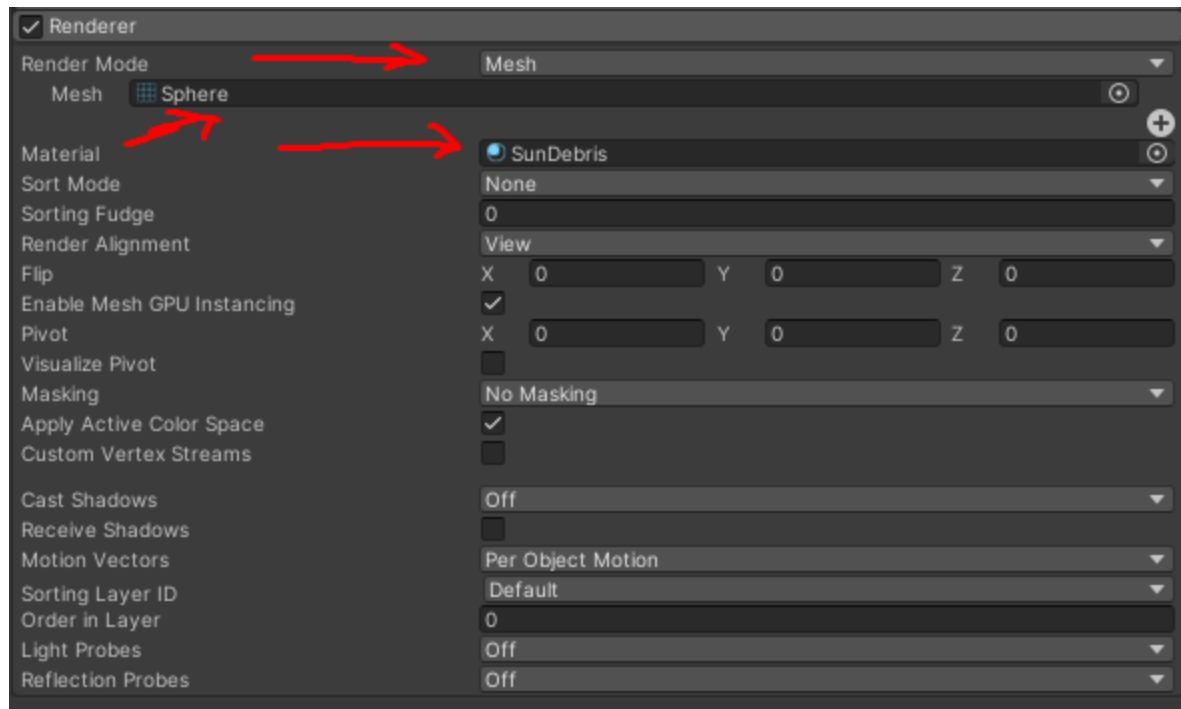
Multiply by Collision Angle ☒

Multiply by Particle Speed ☐

Multiply by Particle Size ☐

Send Collision Messages ☐

Visualize Bounds ☐



Ετσι ειχαμε πλεον 7 Particle Systems που κανουν μια «εκρηξη» με μικροτερα κομματια του καθε πλανητη. Στο αρχειο DetectCollision.cs στην συναρτηση start() (εκτελειται μονο στην αρχη) καναμε deactivate ολα τα particle systems (γιατι εχουμε το PlayOnAwake πεδιο = true αρα παιζουν με το που γινουν activate). Επιπλεον μεσα στην OnTriggerEnter() ελεγχουμε σε ποιον πλανητη ανηκει ο collider με τον οποιον συγκρουστηκε ο μετεωριτης και αναλογα κανουμε activate τα particles εκρηξης των 2 αυτων πλανητων (μετεωριτη – άλλου) και κανουμε μια μεταβλητη με Ονομα ____Active (πχ SunActive, P1Active ...) = 0 ωστε να μην ξαναενεργοποιηθουν τα particles παρολο που εχει ηδη εξαφανιστει ο πλανητης απο το προηγουμενο βημα. Επειτα καλουμε την συναρτηση setActiveMeteor() με την χρηση της Invoke() (ωστε να γινει μετα απο 3sec) για να ξανακανει activate τον μετεωριτη

και deactivate το particle system του (ωστε να μπορεί να ξαναγίνει η εκρηξη) .

Καναμε drag and drop τα 2 αρχεία (detectCollision.cs και MeteorMovement) στο gameObject του μετεωριτη.

```
29 void Start()  
30 {  
31     ParticleSun.SetActive(false);  
32     ParticleP1.SetActive(false);  
33     ParticleP2.SetActive(false);  
34     ParticleP3.SetActive(false);  
35     ParticleP4.SetActive(false);  
36     ParticleP5.SetActive(false);  
37     ParticleMeteor.SetActive(false);  
38     boom = GetComponent();  
39 }
```

```
51 if(other.gameObject.name == "Sun" && SunActive == 1)
52 {
53     Debug.Log("hit sun");
54     ParticleSun.SetActive(true);
55     ParticleMeteor.SetActive(true);
56     SunActive = 0;
57 }
58 if (other.gameObject.name == "P1" && P1Active == 1)
59 {
60     Debug.Log("hit P1");
61     ParticleP1.SetActive(true);
62     ParticleMeteor.SetActive(true);
63     P1Active = 0;
64 }
65 if (other.gameObject.name == "P2" && P2Active == 1)
66 {
67     Debug.Log("hit P2");
68     ParticleP2.SetActive(true);
69     ParticleMeteor.SetActive(true);
70     P2Active = 0;
71 }
72 if (other.gameObject.name == "P3" && P3Active == 1)
73 {
74     Debug.Log("hit P3");
75     ParticleP3.SetActive(true);
76     ParticleMeteor.SetActive(true);
77     P3Active = 0;
78 }
79 if (other.gameObject.name == "P4" && P4Active == 1)
80 {
81     Debug.Log("hit P4");
82     ParticleP4.SetActive(true);
83     ParticleMeteor.SetActive(true);
84     P4Active = 0;
85 }
86 if (other.gameObject.name == "P5" && P5Active == 1)
87 {
88     Debug.Log("hit P5");
89     ParticleP5.SetActive(true);
90     ParticleMeteor.SetActive(true);
91     P5Active = 0;
92 }
93 Invoke("setActiveMeteor", 3f);
```

```

100 void setActiveMeteor()
101 {
102 
103     gameObject.SetActive(true);
104     ParticleMeteor.SetActive(false);
105 }

```

vi) BONUS:

- a) Για τον ηχο κατα την εκρηξη των πλανητων κατεβασαμε ενα audio file το βαλαμε στα assets και μετα στο αρχειο DetectCollision καναμε μια μεταβλητη boom με τον ηχο , καναμε assign το audio file με drag and drop απο τα assets στον inspector και τελος σε καθε συγκρουση βαλαμε το boom.Play()
- b) Στο αρχειο RotationAroundSun.cs με την χρηση της transform.Rotate() κανουμε rotate τον καθε πλανητη γυρω απο τον ευατο του με ξεχωριστη ταχυτητα απο αυτη της περιστροφης γυρω απο τον ηλιο.
- c) Δεξι κλικ στην main camera > UI > canvas. Στον καινουριο canvas που δημιουργησαμε στο πεδιο canvas component αλλαξαμε το render mode σε Screen space – Camera, render camera βαλαμε με drag and drop την main camera μας και μεγαλωσαμε το plane distance. Δεξι κλικ στον canvas > UI > Image στο πεδιο Image component βαλαμε με drag and drop στο source image την εικονα με τα αστερια απο τα assets μας.
- f) Δεξι κλικ στον canvas που φτιαξαμε στο (bonus c) ερωτημα > UI > Text. Στο Inspector στο πεδιο text γραψαμε 0 και αλλαξαμε font size και position. Μεσα στο αρχειο detectCollision.cs φτιαξαμε μια μεταβλητη που περιεχει το text μας και εναν counter που μετραει ποσους πλανητες χτυπησαμε . Καθε φορα που χτυπαμε εναν πλανητη το counter αυξανεται κατα 1 και μετα παμε στο πεδιο text του scoretext μας και γραφουμε οτι τιμη εχει η μεταβλητη counter (χρησιμοποιουμε την ToString γιατι ο counter ειναι int). Τελος καναμε drag and drop το text απο το Hierarchy tab στο score text πεδιο του DetectCollision script που ειναι attached στο Meteor

GameObject.