



Crash++

Βήματα και Υλοποίηση

Μία εφαρμογή επαυξημένης πραγματικότητας στο μάθημα Αλληλεπίδραση Ανθρώπου-Μηχανής.

I. Vuforia

Αρχικά επιλέγουμε το image target που θα χρησιμοποιηθεί για την εφαρμογή. Εφόσον βασίζεται σε ένα χαρακτήρα ενός βιντεοπαιχνιδιού, ανάλογο θα είναι και το image target.



Εικόνα 1 Toad Village, Naughty Dog

Δημιουργούμε ένα λογαριασμό το [Developer.vuforia.com](https://developer.vuforia.com) και φτιάχνουμε ένα Development Key. Έπειτα υλοποιούμε μια βάση δεδομένων με την εικόνα μας. Παρατηρούμε την αξιολόγηση. Αφού είναι πάνω από 4 αστέρια, τότε δεν θα έχουμε πρόβλημα στην ανίχνευση του εικονικού στόχου μας. Κατεβάζουμε τη βάση δεδομένων.

HomePricingDownloadsLibraryDevelopSupport

License ManagerTarget Manager

License Manager

Create a license key for your application.

Get Development Key

Buy Deployment Key

Name	Type	Status ▾	Date Modified
Crash BandiGOOD	Develop	Active	Nov 21, 2018 13:04

Last updated: Today 3:14 PM

Refresh

Εικόνα 2 Δημιουργία Development Key

HomePricingDownloadsLibraryDevelopSupport

License ManagerTarget Manager

Target Manager

Use the Target Manager to create and manage databases and targets.

Add Database

Database	Type	Targets	Date Modified
CrashBandicootDB	Device	1	Nov 21, 2018 13:05
CrashDB	Cloud	1	Nov 21, 2018 13:03
Fraps	Device	6	Nov 15, 2018 14:34

Last updated: Today 3:18 PM

Refresh

Εικόνα 3 Δημιουργία Βάσης Δεδομένων

[Target Manager](#) > [CrashBandicootDB](#)

CrashBandicootDB [Edit Name](#)

Type: Device

Targets (1)

Add Target

Download Database (All)

<input type="checkbox"/>	Target Name	Type	Rating	Status ▾	Date Modified
<input type="checkbox"/>	 toad-village	Single Image	★★★★★	Active	Nov 21, 2018 13:05

Last updated: Today 03:19 PM [Refresh](#)

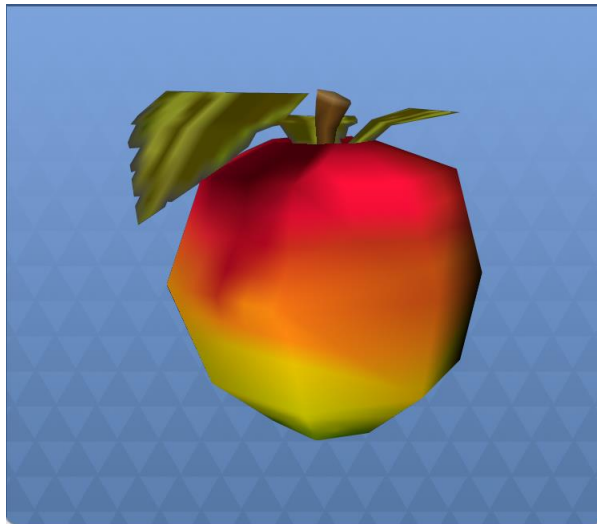
Εικόνα 4 Αξιολόγηση Image Target

II. Resources

Από την ιστοσελίδα models-resource.com κατεβάζουμε τα μοντέλα που θα ενσωματώσουμε στην εφαρμογή μας. Θα χρειαστούμε 2 μοντέλα του Crash για την εμφάνιση και περιστροφή του, και ένα μοντέλο για το μήλο – μπάλα που θα εκτοξεύουμε.



Εικόνες 5 & 6 , Crash, Crash Team Racing & Crash Wrath of Cortex, Naughty Dog

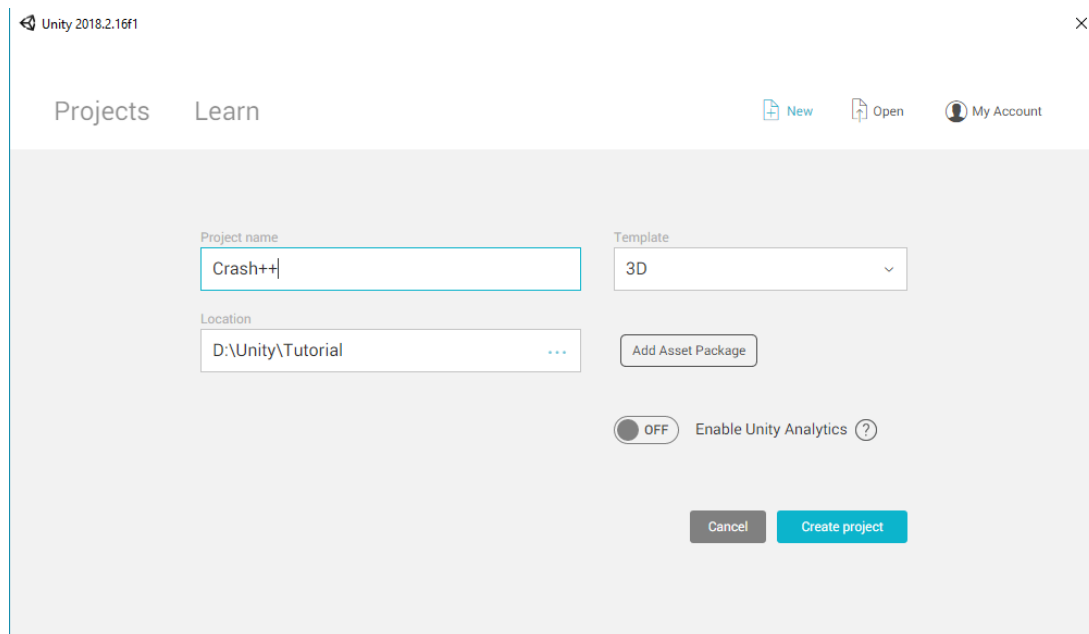


Εικόνα 7 Wumpa Fruit , Crash Wrath of Cortex, Naughty Dog

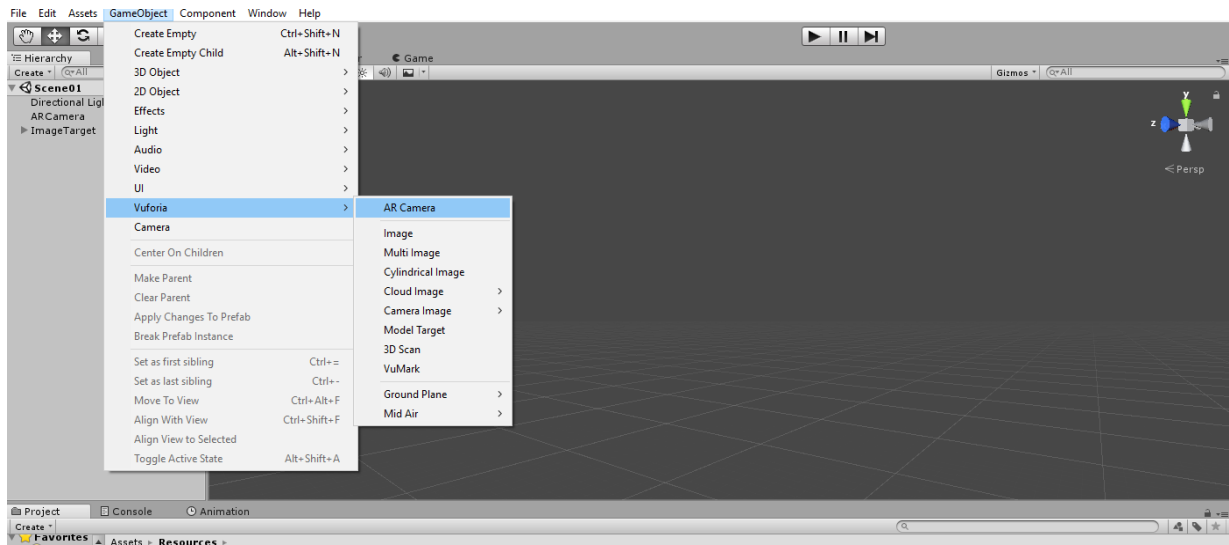
Από την ιστοσελίδα [sounds-resource](https://sounds-resource.com) κατεβάζουμε το πακέτο με το sound effect που θα χρειαστούμε.

III. Εισαγωγή Unity

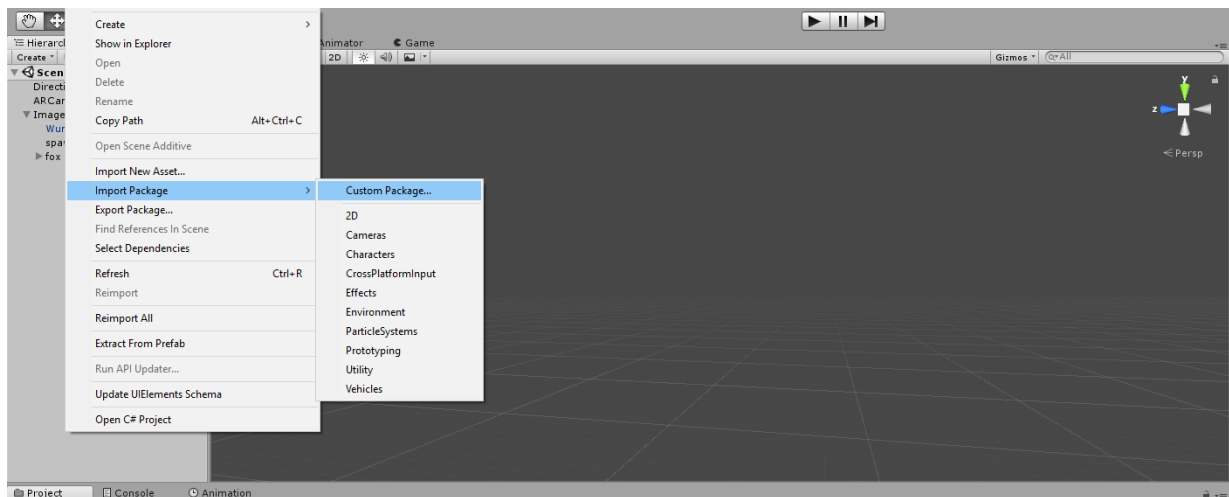
Κατεβάζουμε και εγκαθιστούμε το [Unity](#). Δημιουργούμε νέο project και εισάγουμε τα αρχεία και τη βάση δεδομένων του Vuforia. Στο import package επιλέγουμε το αρχείο που κατεβάσαμε από τη σελίδα του Vuforia.



Εικόνα 8 Δημιουργία Project

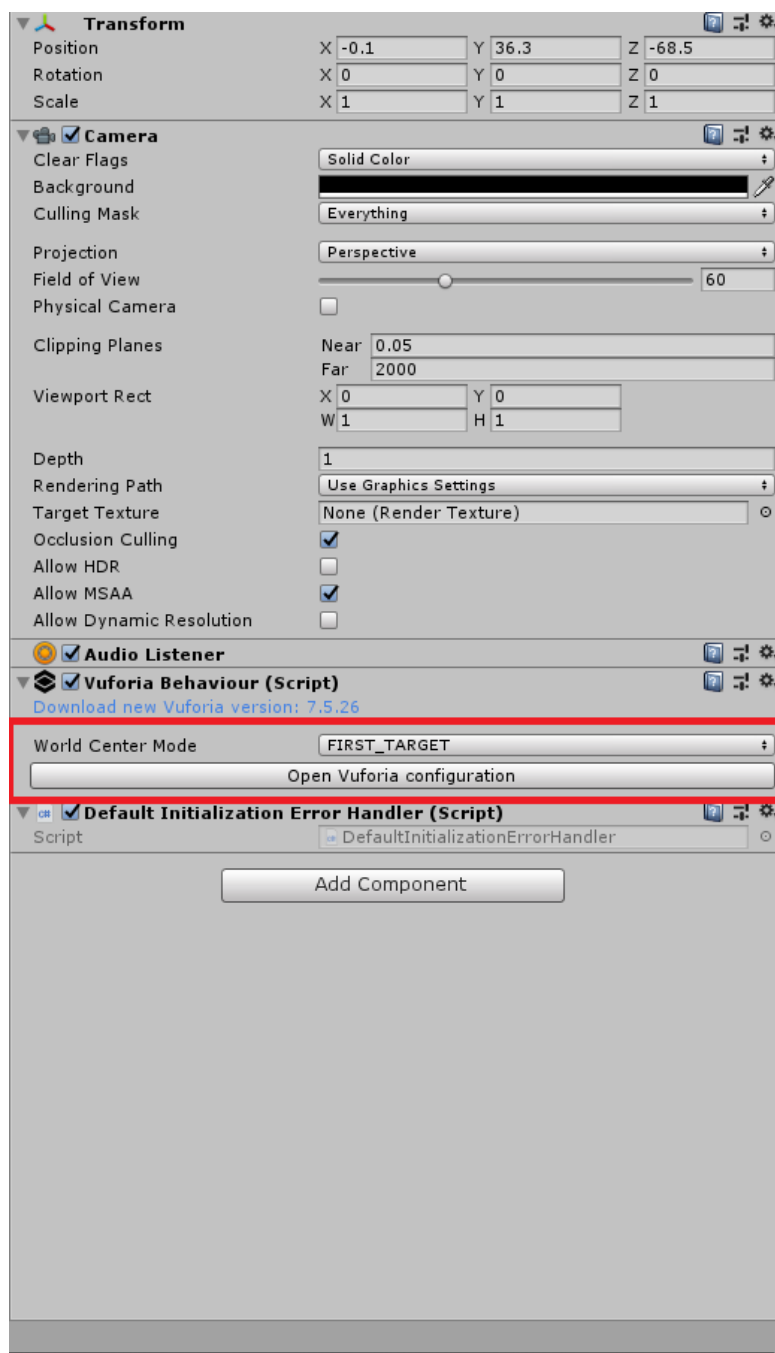


Εικόνα 9 Προσθήκη AR Camera



Εικόνα 10 Εισαγωγή Βάσης Δεδομένων Vuforia

Στη νέα μας σκηνή, επιλέγουμε το στοιχείο ARCamera και θέτουμε στο World Center Mode την επιλογή "First Target" έτσι ώστε η βαρύτητα να ισχύει με βάση το imagetarget. Στη συνέχεια πατάμε "Open Vuforia Configuration" και προσθέτουμε το Development Key, αυτό που δημιουργήσαμε στην ιστοσελίδα του Vuforia. Παράλληλα, επιλέγουμε ως device : Handheld.



Εικόνα 11 World Center Mode

[Open Library Article](#)

App License Key


Delayed Initialization ☐

Camera Device Mode

Max Simultaneous Tracked Images

Max Simultaneous Tracked Objects

Load Object Targets on Detection ☐

 Front camera support is deprecated and will be removed in a future Vuforia release.


Camera Direction

Mirror Video Background

▼ **Digital Eyewear**

Device Type


▼ **Databases**

 Databases will be automatically loaded and activated if its TrackingBehaviour is enabled on scene load.

CrashBandicootDB


▼ **Video Background**

Enable video background ☒

Video Background Shader 

Number Divisions

Overflow geometry

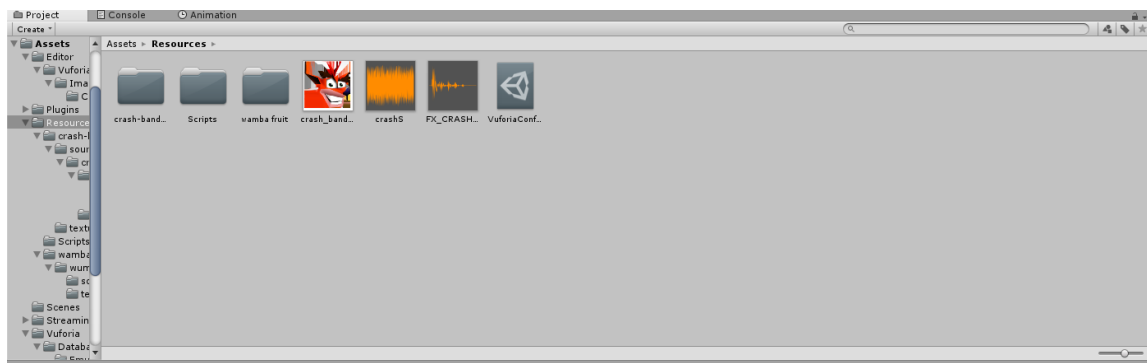
Matte Shader 

▼ **Device Tracker**

Track Device Pose ☐

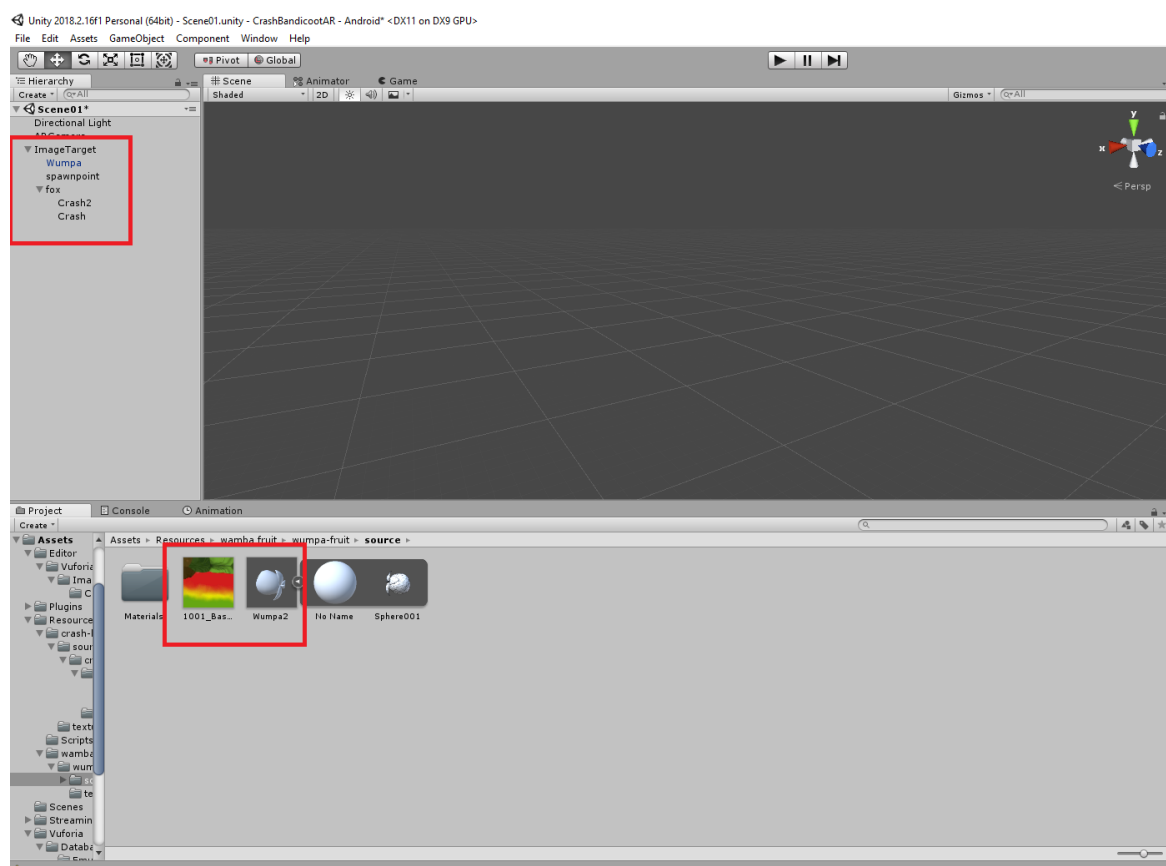
Εικόνα 12 Εισαγωγή License Key

Κάνουμε unzip, Drag 'n Drop όλα τα resources μας μέσα στο Unity



Εικόνα 12 Περιβάλλον Resources

- I. Δημιουργούμε ένα κενό αντικείμενο «fox» και το τοποθετούμε μέσα στο image target.
- II. Τοποθετούμε μέσα στο image target το μοντέλο του wumba fruit, drag n drop το texture του επάνω στο μοντέλο.
- III. Τοποθετούμε στο fox τα μοντέλα του Crash, βάζουμε τα απαραίτητα textures επάνω στα μοντέλα.

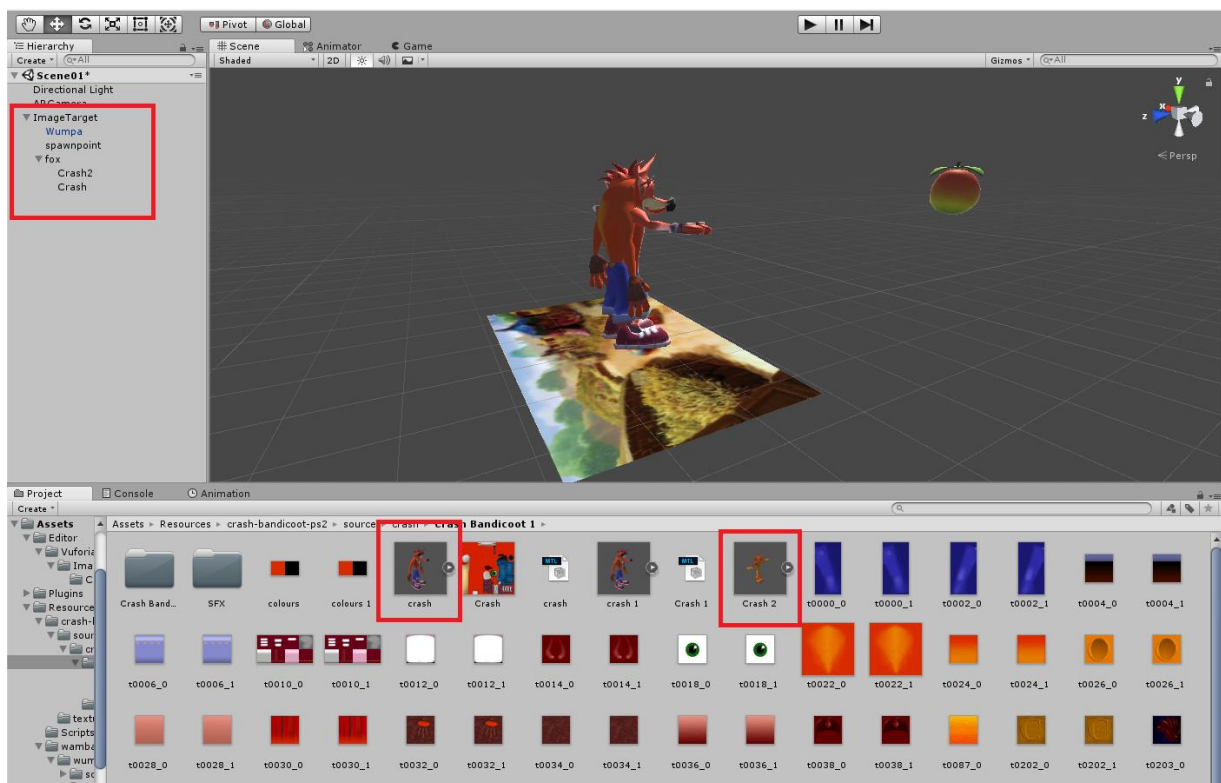


Εικόνα 14 Εισαγωγή Wumba Fruit

Παρατηρώντας ότι το texture του Crash Team Racing διαφέρει χρωματικά από το πρώτο,ανοίγουμε με έναν Image Editor το texture του και το χρωματίζουμε ανάλογα.



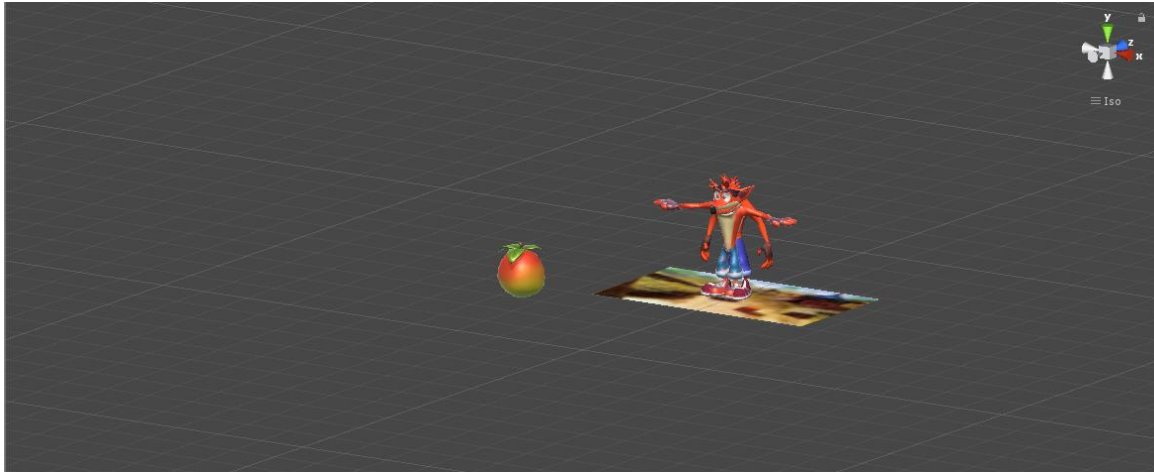
Εικόνες 14 & 15 Texture Πριν και Μετά.



Εικόνα 16 Εισαγωγή Crash

Μετακινήσεις μέσα στη σκηνή :

- Μοντέλο Crash Wrath of Cortex στο κέντρο του image target
- Μοντέλο Crash Team Racing στην ίδια ακριβώς περιοχή με το πρώτο
- Wumpa Fruit στην ίδια ευθεία, λίγο πιο ψηλά από τα υπόλοιπα μοντέλα



Εικόνα 17 Σκηνή Εφαρμογής

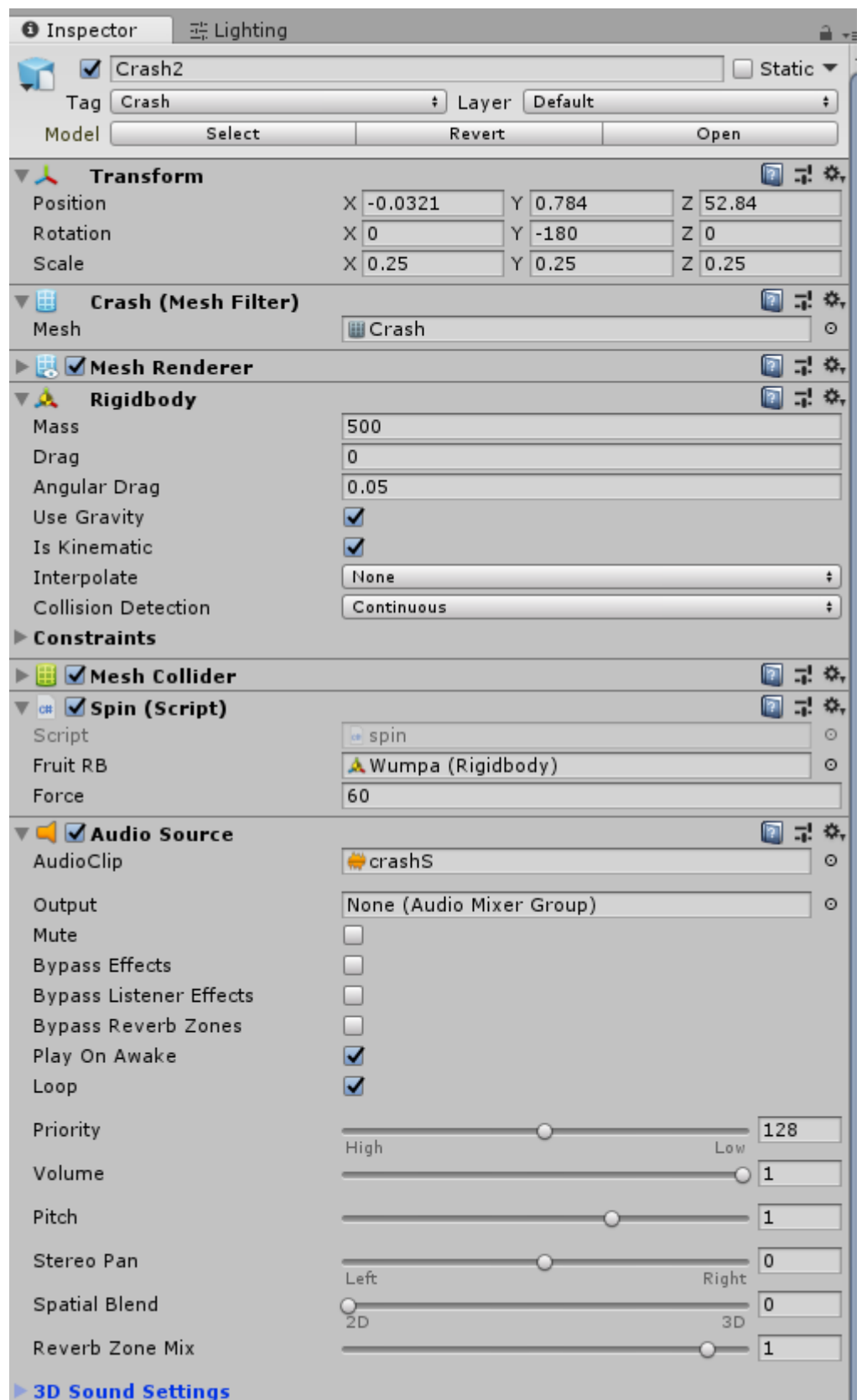
IV. Ιδιότητες Στοιχείων

Το μοντέλο Crash Wrath of Cortex (Crash) και το Wumpa Fruit (Wumpa) θα είναι τα πρώτα που εμφανίζονται με την αναγνώριση του image target, ενώ το Crash Team Racing (Crash2) με το άγγιγμα της οθόνης, όπου θα εξαφανίζεται το πρώτο μοντέλο. Οπότε η βασική αλληλεπίδραση θα είναι μεταξύ των μοντέλων Wumpa και Crash.

Τώρα θα προσθέσουμε μερικές ιδιότητες στα μοντέλα μας.

Για το Crash2 :

- I. Add Component – Rigidbody
έτσι ώστε να δέχεται επιδράσεις physics.
Use Gravity ✓
Is Kinematic ✓
Collision Detection : Continuous
- II. Add Component – Audio Source
Drag n Drop το αρχείο ήχου που θα παίζει όταν το μοντέλο είναι ενεργό στο Audio Clip.
Loop ✓
- III. Scale – X:0.25 Y:0.25 Z:0.25
Το μέγεθος πρέπει να είναι ανάλογος με τα υπόλοιπα στοιχεία.

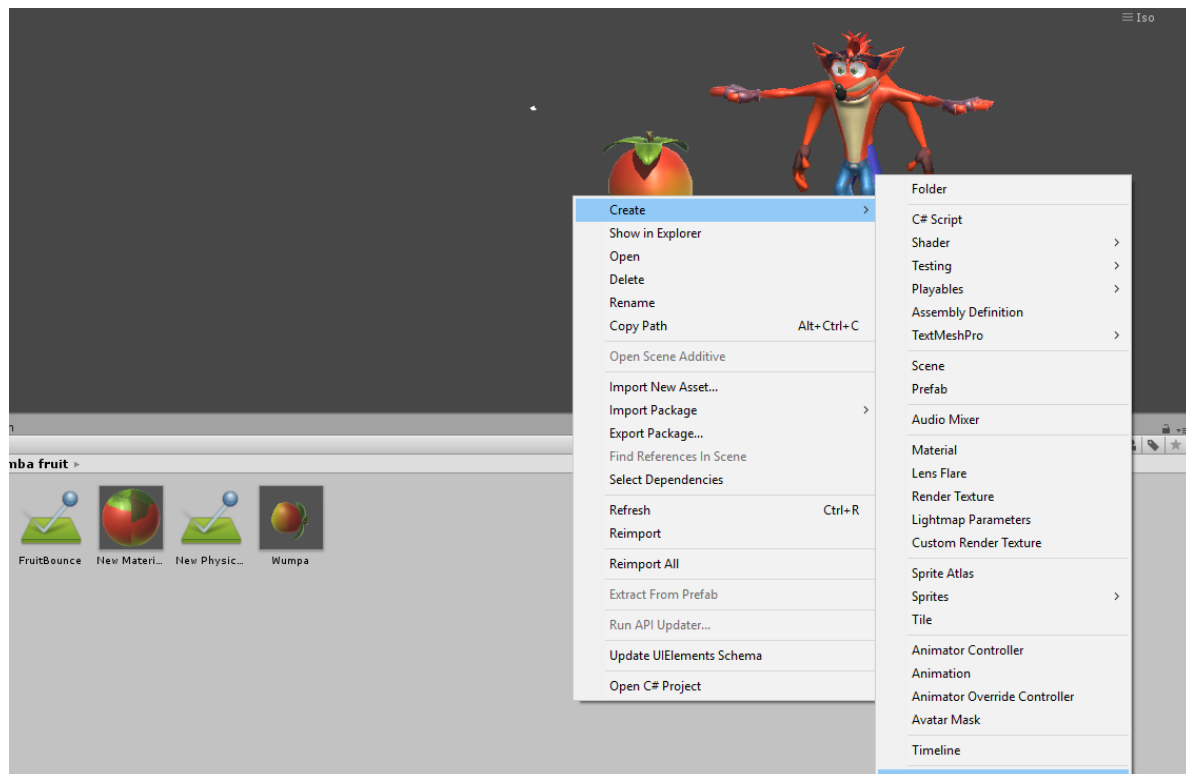


Εικόνα 18 Ιδιότητες Crash2

Για το Wumpa :

- I. Scale – X: 0.6 Y:0.6 Z:0.6
Αναλογία Μεγέθους
- II. Add Component – Rigidbody
Επιδράσεις Physics
Use Gravity ✓
Is Kinematic ✓
(Όσο είναι Kinematic, δεν επηρεάζεται από τις δυνάμεις τις βαρύτητας)
Collision Detection : Continuous
- III. Add Component – Sphere Collider

Σε αυτό το σημείο θα δημιουργήσουμε ένα νέο στοιχείο για την αντίδραση του Wumpa στην κρούση του με το Crash2. Δεξί κλικ στο περιβάλλον Resources, Create, Physics Material.



Εικόνα 19 Εισαγωγή Physics Material

Το ονομάζουμε FruitBounce και του θέτουμε τις εξής τιμές :

Dynamic Fiction : 0.4

Ισχύς Αδράνειας

Static Fiction : 0.5

Ευκολία Μετατόπισης

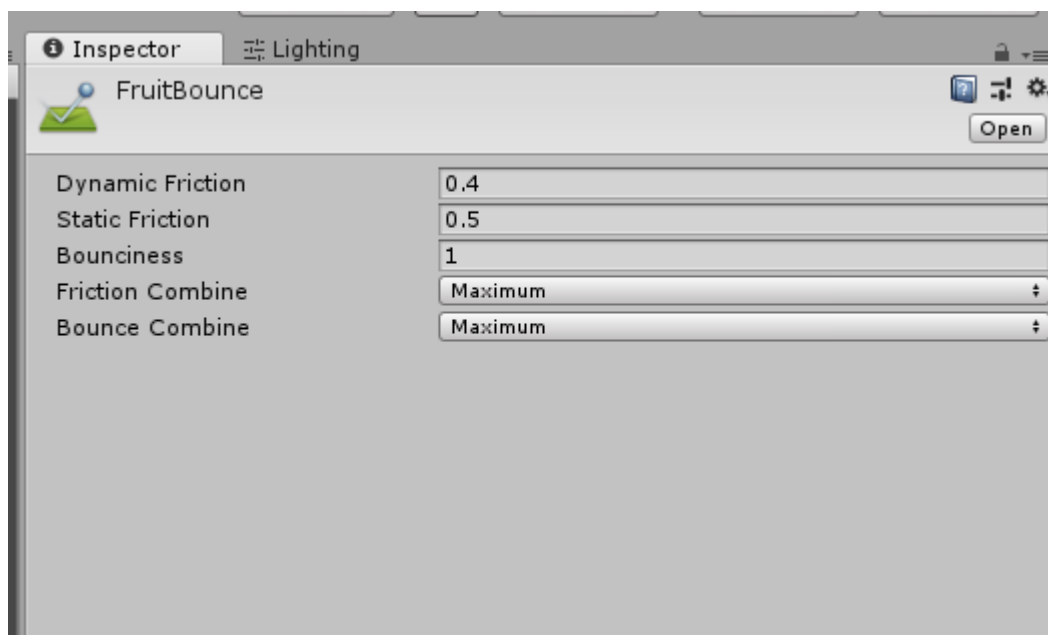
Bounciness : 1

Επίπεδο Αναπήδηση

Friction Combine : Maximum

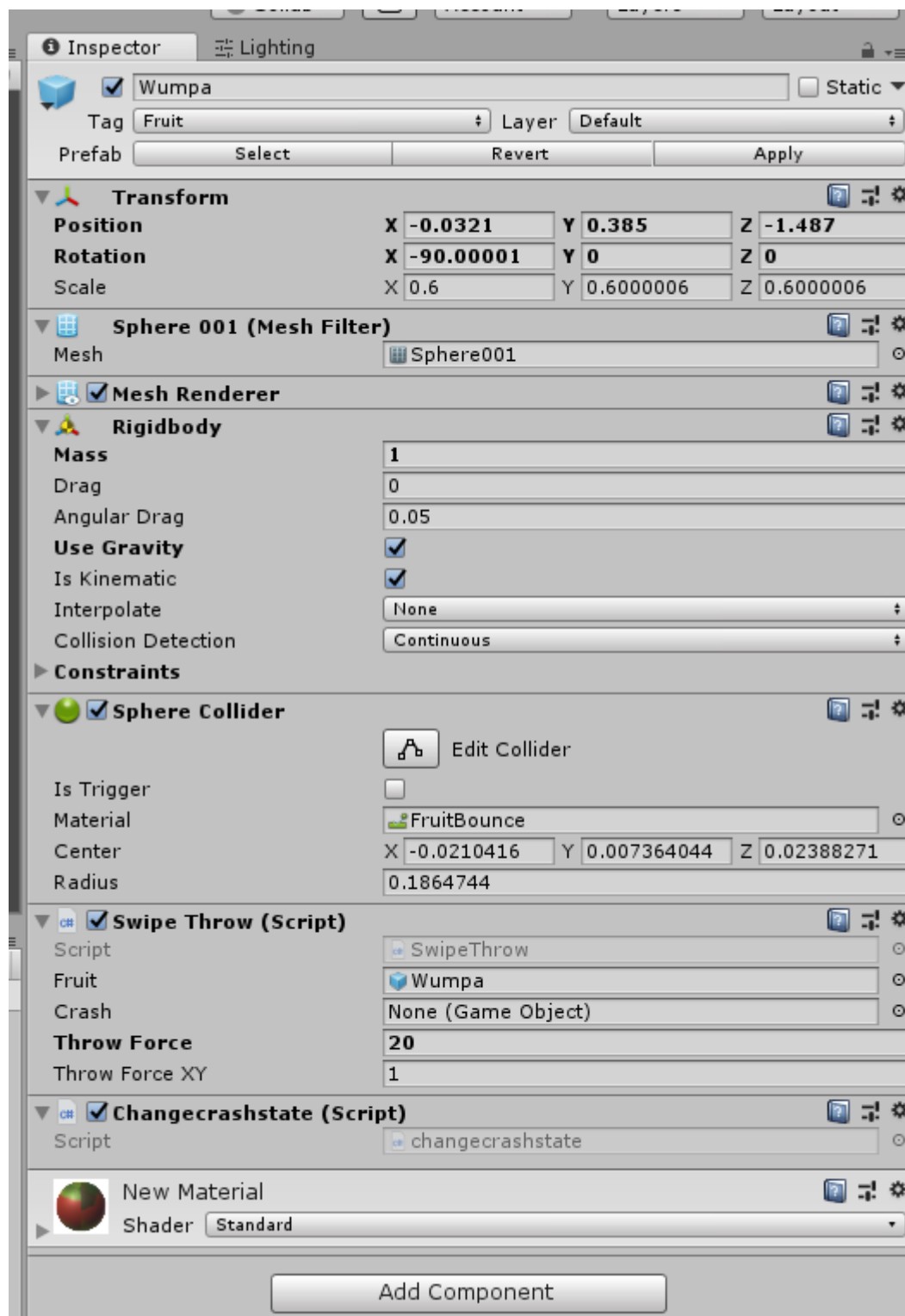
Bounce Combine : Maximum

Δράσεις συγκρουόμενων αντικειμένων



Εικόνα 20 FruitBounce

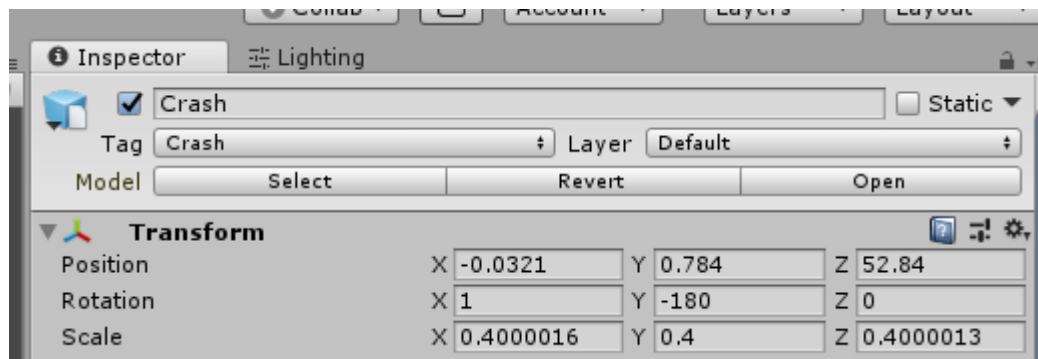
Το τοποθετούμε στο Material του Sphere Collider



Εικόνα 21 Ιδιότητες Wumpa

Για το Crash :

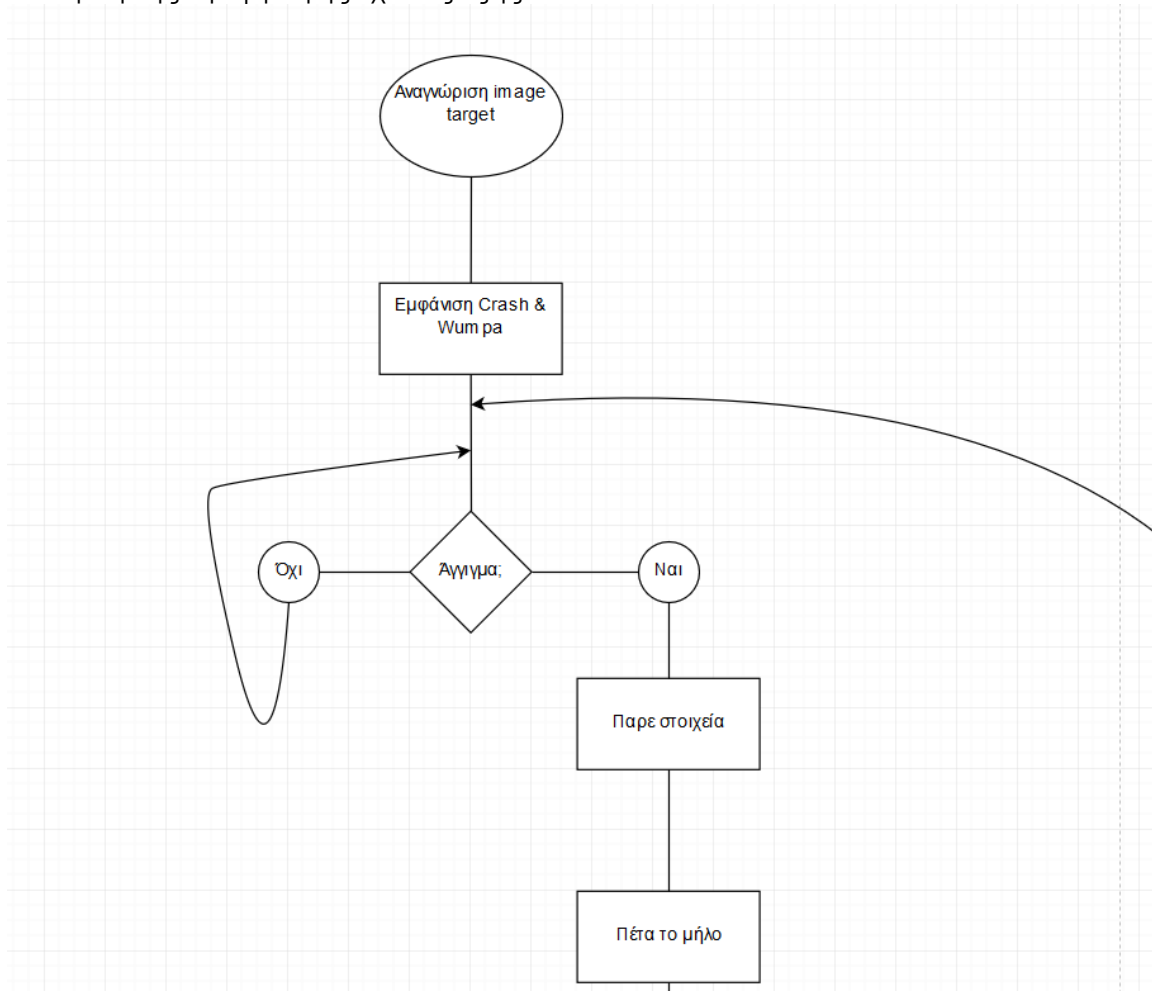
- I. Scale : X: 0.4 Y:0.4 Z:0.4
Αναλογία



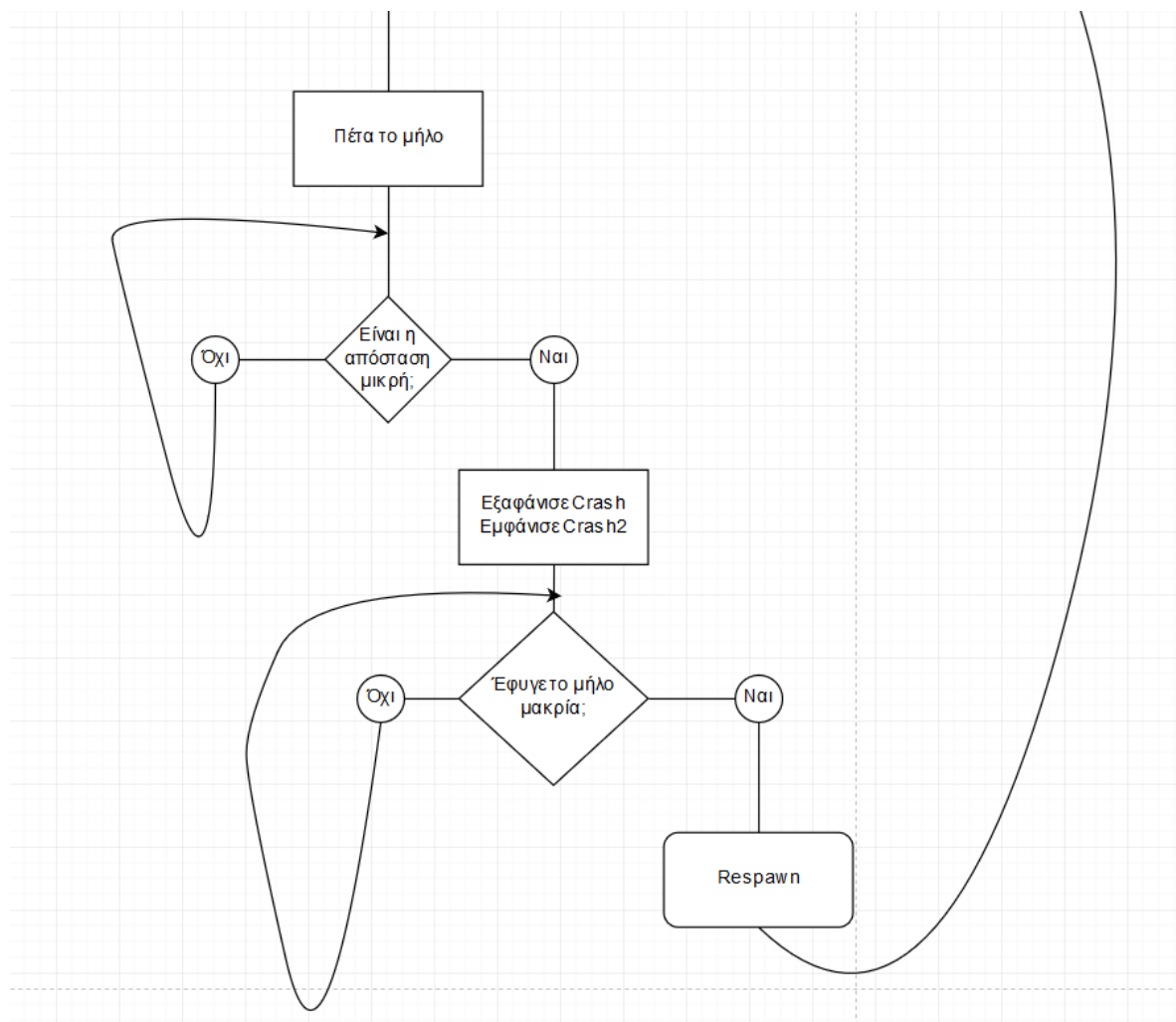
Εικόνα 22 Ιδιότητες Crash

V. Scripts

Η λογική της εφαρμογής έχει ως εξής:



Εικόνα 22α Διάγραμμα Crash++



Εικόνα 22β Διάγραμμα Crash++

[Διάγραμμα Online](#)

Script 1 : SwipeThrow

Σε αυτό το script θα υλοποιήσουμε τη λειτουργία πετάγματος με την κίνηση επάνω στην οθόνη. Το Script θα είναι του Crash2, οπότε κάνουμε Add Component, Add Script Θα χρειαστούμε:

3 μεταβλητές τύπου Vector2, μεταβλητές δηλαδή με δύο πεδία.

startpos – αρχική θέση κίνησης

endpos – τελική θέση κίνησης

direction – κατεύθυνση μπάλας

5 μεταβλητές float

touchTimeStart – ώρα εκκίνησης

touchTimeFinish – ώρα τέλους κίνησης

timeInterval – Διάρκεια

throwForce – Ορμή

throwForceXY – ορμή στους άξονες X,Y

1 μεταβλητή bool

flag – αν έχει πραγματοποιηθεί κίνηση

2 public μεταβλητές GameObject

fruit - Για το Wumpa Fruit

crash - Για το Crash2

1 μεταβλητή Rigidbody

rb – για το rigidbody του wumpa ώστε να μπορεί να δέχεται επιρροές physics

void Start()

Θέσε στο rb το Rigidbody του αντικειμένου που έχει το script

void Update()

με τη μέθοδο `transform.Rotate(10,0,0)` θέτουμε την περιστροφή του wumpa στον άξονα X.

Στη συνέχεια ελέγχουμε αν έχει υπάρξει κάποια κίνηση επάνω στην οθόνη. Αν έχει, τότε παίρνουμε τη θέση που ξεκίνησε και τη χρονική στιγμή.

Επόμενος έλεγχος αν έχει υπάρξει και σταματήσει μια κίνηση. Αν ναι, τότε παίρνουμε τη χρονική στιγμή, αφαιρούμε την από την αρχική και κρατάμε το αποτέλεσμα ως διάρκεια. Επίσης κρατάμε την τελική θέση, την οποία θα αφαιρέσουμε από την αρχική για να βρούμε την κατεύθυνση της κίνησης.

Έπειτα, θέτουμε την επιλογή Kinematic σε false με τη μέθοδο `isKinematic` του Rigidbody έτσι ώστε να δέχεται δυνάμεις. Προσθέτουμε μια δύναμη στο wumpa μέσω του rigidbody του χρησιμοποιώντας τη μέθοδο `AddForce` και πεδία την αρνητική κατεύθυνση * `throwForce XY` για τους άξονες X,Y και `throwForce / timeInterval` στον άξονα Z έτσι ώστε η ταχύτητα να είναι ανάλογη με την καθυστέρηση της κίνησης. Εφόσον έχει πραγματοποιηθεί μια κίνηση, θέτουμε στο `flag = true`

Τέλος, αν η απόσταση μεταξύ του crash και του wumpa είναι μηδαμινή, και φυσικά έχει υπάρξει κάποια κίνηση, τότε πρόσθεσε *10 τη δύναμη στον άξονα Z για να φαίνεται η δύναμη με την οποία ο Crash χτυπάει το μήλο. Τέλος αλλάζουμε το `flag` σε false, έτσι ώστε να μην επαναληφθεί η τελευταία προσθήκη.


```

public class SwipeThrow : MonoBehaviour {
    Vector2 startPos, endPos, direction;
    float touchTimeStart, touchTimeFinish, timeInterval;
    Rigidbody rb;
    bool flag = false;
    public GameObject fruit;
    public GameObject crash;

    //allakse ti metavliti apo to unity
    [SerializeField]
    float throwForce = 140f;

    //allakse ti metavliti apo to unity
    [SerializeField]
    float throwForceXY = 1f;

    // Use this for initialization
    void Start () {
        rb = GetComponent<Rigidbody> ();
    }

    // Update is called once per frame
    void Update()
    {
        //me tin parodo tou kathe frame,gyrna to fruit ston aksona X
        transform.Rotate(10, 0, 0);

        //an agkikse othoni,pare ti thesi kai to xrono
        if (Input.touchCount > 0 && Input.GetTouch(0).phase == TouchPhase.Began)
        {
            touchTimeStart = Time.time;
            startPos = Input.GetTouch(0).position;
        }

        //afou stamatisi h kinisi,pare to xrono kai ti thesi
        if (Input.touchCount > 0 && Input.GetTouch(0).phase == TouchPhase.Ended)
        {
            touchTimeFinish = Time.time;

```

Εικόνα 23α Κώδικας SwipeThrow

```

//afou stamatisi h kinisi,pare to xrono kai ti thesi
if (Input.touchCount > 0 && Input.GetTouch(0).phase == TouchPhase.Ended)
{
    touchTimeFinish = Time.time;

    //diarkeia
    timeInterval = touchTimeFinish - touchTimeStart;

    endPos = Input.GetTouch(0).position;

    //katefthinsi
    direction = startPos - endPos;

    //to fruit tha epireazetai pleon apo ti varitita
    rb.isKinematic = false;

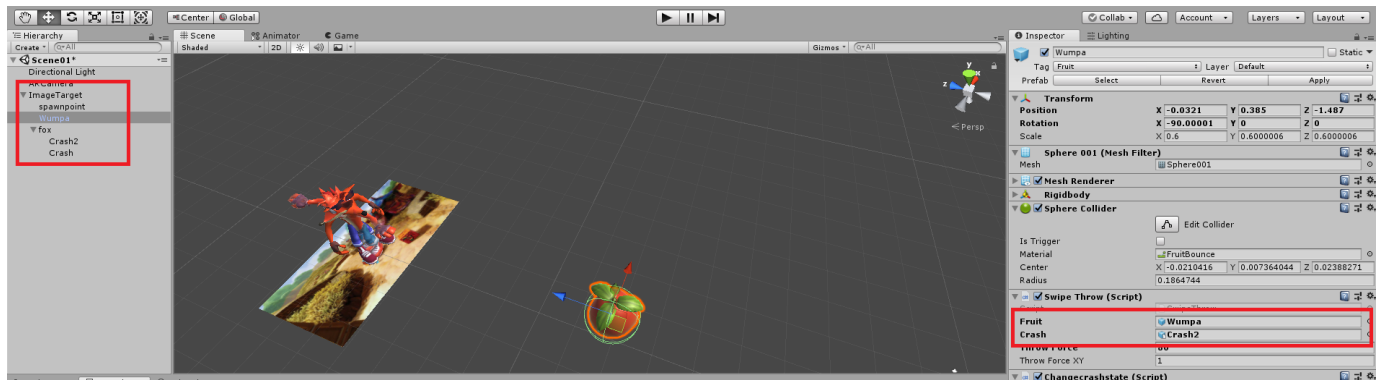
    //vale mia dinami sto frouto analogi me ta parapanw
    rb.AddForce(-direction.x*throwForceXY,-direction.y*throwForceXY, throwForce/timeInterval);

    //exei pragmatopoihthei kinisi
    flag = true;
}
Debug.Log(Vector3.Distance(crash.transform.position, fruit.transform.position));
//an exei yparksei kinisi kai h apostasi Wumpa kai Crash einai midamini
if (Vector3.Distance(crash.transform.position, fruit.transform.position) < 28 && flag)
{
    //prothese mia akomi dinami gia na fainetai megaliteri h ormi stin krousi
    rb.AddForce(throwForceXY,throwForceXY, 10* throwForce / timeInterval);
    flag = false;
}
}

```

Εικόνα 23β Κώδικας SwipeThrow

Στο περιεχόμενο του Crash2 θα έχει δημιουργηθεί το script μας με 3 μεταβλητές τις οποίες μπορούμε να πειράξουμε εμείς μέσα από το Unity. Έτσι, κάνουμε drag n drop κάθε στοιχείο από την αριστερή ιεραρχία στο ανάλογο ονομαστικά πεδίο.



Εικόνα 24 Περιεχόμενο SwipeThrow()

Script 2 : ChangeCrashState

Τώρα, θα προγραμματίσουμε την αλλαγή των μοντέλων του Crash.
Θα χρειαστούμε:

2 μεταβλητές GameObject
Τα οποία θα είναι τα μοντέλα μας.

void Start()

Βρίσκουμε τις αναφορές των στοιχείων σκηνής μας και τις εφαρμόζουμε ανάλογα με τις ονομασίες των μεταβλητών μας. Εφόσον αυτή η μέθοδος εκτελείται με τη φόρτωση της σκηνής, θέτουμε το δεύτερο μοντέλο του Crash(Crash2) σε στάδιο απενεργοποίησης έτσι ώστε να μην είναι εμφανές.

void Update()

Συγκρίνουμε, σε κάθε frame, την απόσταση μεταξύ μοντέλου Crash και wumpa fruit. Αν το wumpa fruit έχει μετατοπιστεί αρκετά τότε απενεργοποίησε το μοντέλο του πρώτου Crash,και ενεργοποίησε το Crash2.

```
public class changeCrashState : MonoBehaviour {
    GameObject Crash;
    GameObject Crash2;
    // Use this for initialization
    void Start ()
    {
        Crash = GameObject.Find("Crash");
        Crash2 = GameObject.Find("Crash2");
        //apenergopoihse ton Crash2,afou to wumpa einai makria
        Crash2.SetActive(false);
    }

    // Update is called once per frame
    void Update ()
    {
        //an h apostasi metaksi crash kai wumpa einai mikri,tote energopoihse ton crash 2
        //apenergopoihse ton crash
        if (Vector3.Distance(GameObject.FindWithTag("Crash").transform.position,
            GameObject.FindWithTag("Fruit").transform.position) < 40 )
        {
            Crash.SetActive(false);
            Crash2.SetActive(true);
        }
    }
}
```

Εικόνα 25 ChangeCrashState

Script 3 : Spin

Τελειώσαμε με τα Scripts του Wumpa και προχωράμε σε αυτά του Crash2. Σε αυτό το Script θα δώσουμε ένα animation στο Crash2, έτσι ώστε να θυμίζει στρόβιλο. Θα χρειαστούμε :

1 μεταβλητή τύπου float

Για την ταχύτητα περιστροφής.

void Update()

Σε κάθε frame, περιστρέψε το Crash2 στον άξονα Y με ταχύτητα που όρισες πριν.

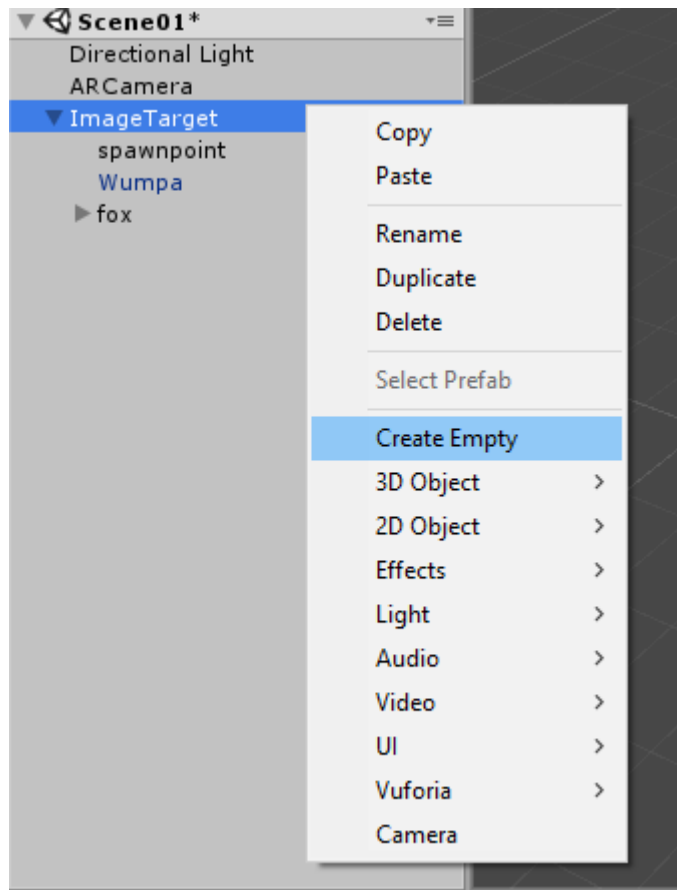
```
public class spin : MonoBehaviour {  
    //taxitita svouras  
    float speed = 60;  
  
    // Use this for initialization  
    void Start ()  
    {  
    }  
  
    // Update is called once per frame  
    void Update () {  
        //gyrna ston aksona Y synexws  
        transform.Rotate(0, speed, 0);  
    }  
}
```

Εικόνα 26 Spin

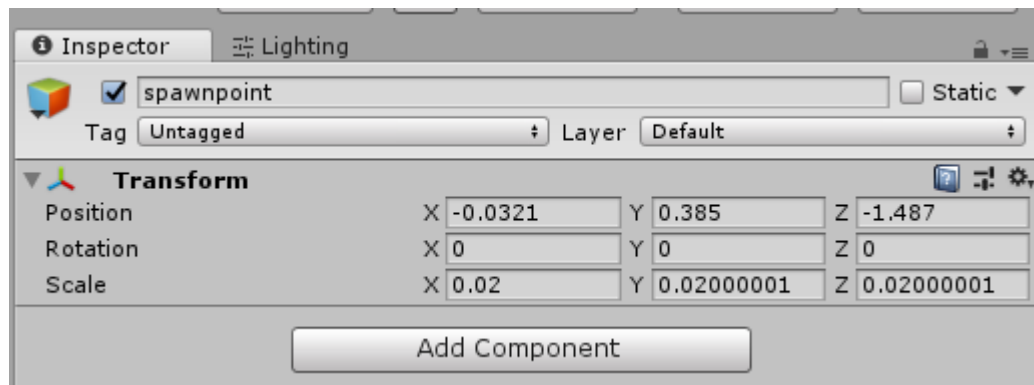
Script 4 : Spawn

Τέλος, θα προγραμματίσουμε την αναγέννηση του Wumpa Fruit.

Αρχικά θα πρέπει να δημιουργήσουμε ένα άδειο αντικείμενο, το οποίο απλά θα υπάρχει στην αρχική θέση του Wumpa Fruit και θα χρησιμοποιείται σαν σημείο αναφοράς για την επανεμφάνιση του τελευταίου. Δεξί κλικ στο Image Target, Create Empty. Θέτουμε την ίδια ακριβώς τοποθεσία με το Wumpa.



Εικόνα 27 Δημιουργία άδειου αντικειμένου



Εικόνα 28 Ιδιότητες SpawnPoint

Για το Script,θα χρειαστούμε :

2 Μεταβλητές Transform

για τις θέσεις spawnpoint και Wumpa Fruit

1 Μεταβλητή Rigidbody για το rigidbody του WumpaFruit

void Update()

Αν το μήλο έχει απομακρυνθεί πολύ από τον Crash, κάλεσε τη συνάρτηση spawnF

void spawnF()

Θέσε το rigidbody του Wumpa να μην είναι πλέον Kinematic, έτσι ώστε να αιωρείται.

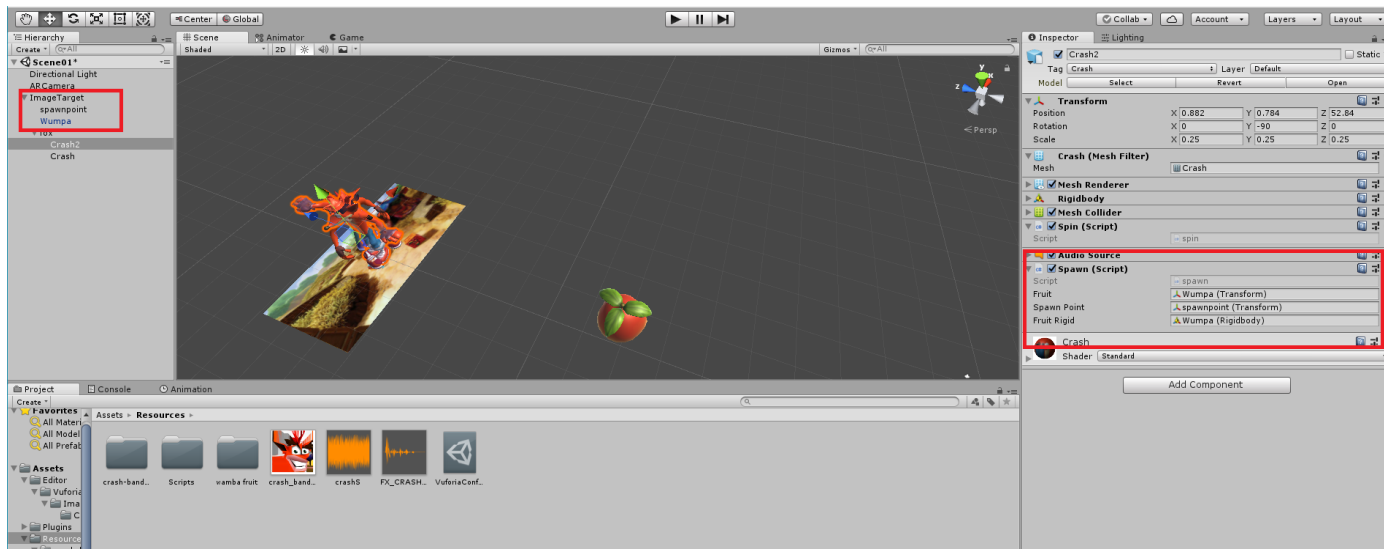
Η θέση του Wumpa Fruit να είναι η ίδια με του SpawnPoint.

Η περιστροφή ας είναι Default.

```
public class spawn : MonoBehaviour {  
  
    [SerializeField] private Transform fruit;  
    [SerializeField] private Transform spawnPoint;  
    [SerializeField] private Rigidbody fruitRigid;  
  
    void Update()  
    {  
        //αν το wumpa exei fugei makria,kane respawn  
        if (Vector3.Distance(transform.position,fruit.position) > 150)  
        {  
            spawnF();  
        }  
    }  
  
    void spawnF()  
    {  
        //to wumpa den epireazetai pleon apo varitita  
        fruitRigid.isKinematic = true;  
  
        //to wumpa tha gyrisei sto position tou spawnpoint  
        fruit.transform.position = spawnPoint.transform.position;  
  
        //to wumpa tha exei default rotation  
        fruit.transform.rotation = Quaternion.identity;  
    }  
}
```

Εικόνα 29 Spawn

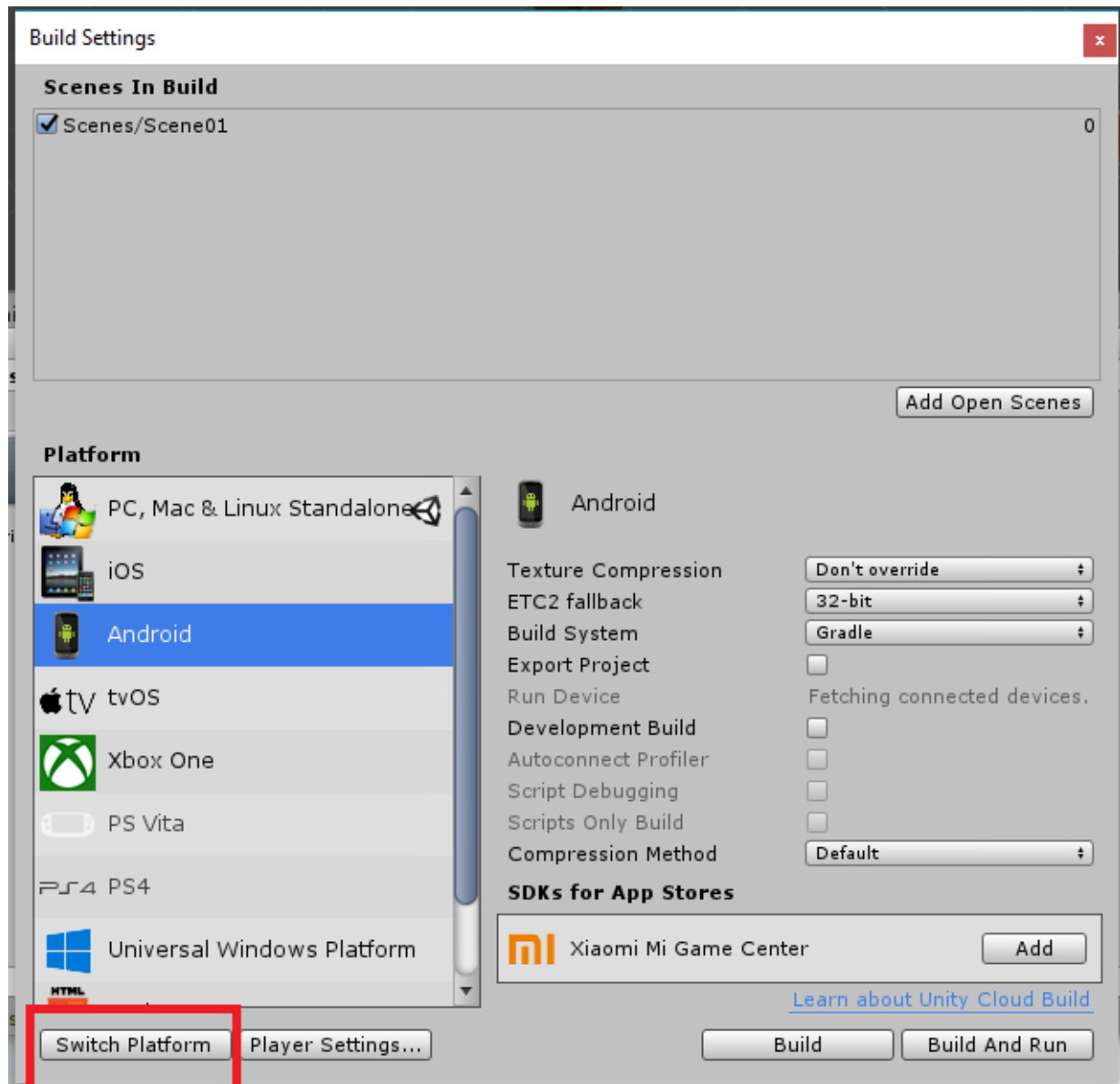
Εφόσον οι μεταβλητές μας είναι SerializeField, πρέπει να τα κάνουμε drag n drop στο περιβάλλον της Unity.



Εικόνα 30 Spawn σε Unity

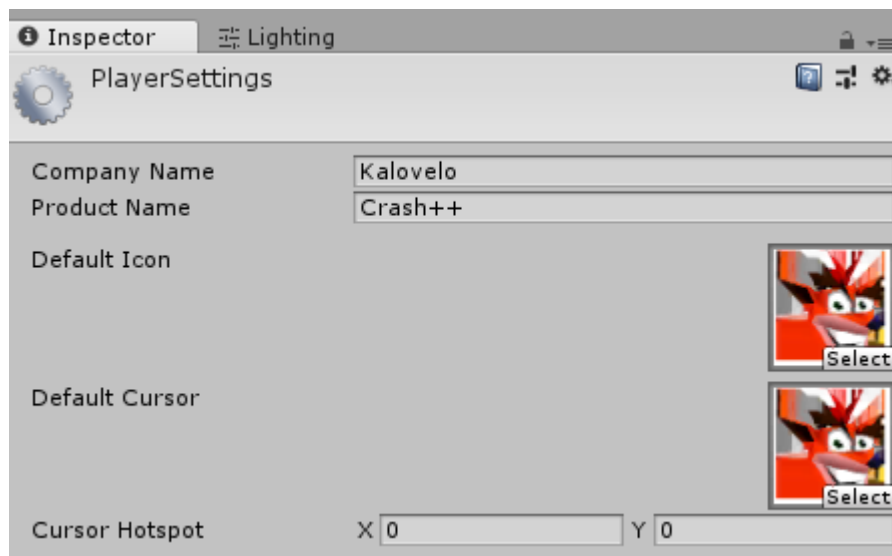
VI. Ολοκλήρωση

Για να ολοκληρώσουμε την εφαρμογή μας, πηγαίνουμε file , Build Settings. Εκεί επιλέγουμε Android, Switch Platform έτσι ώστε το export να είναι σε εφαρμογή android.



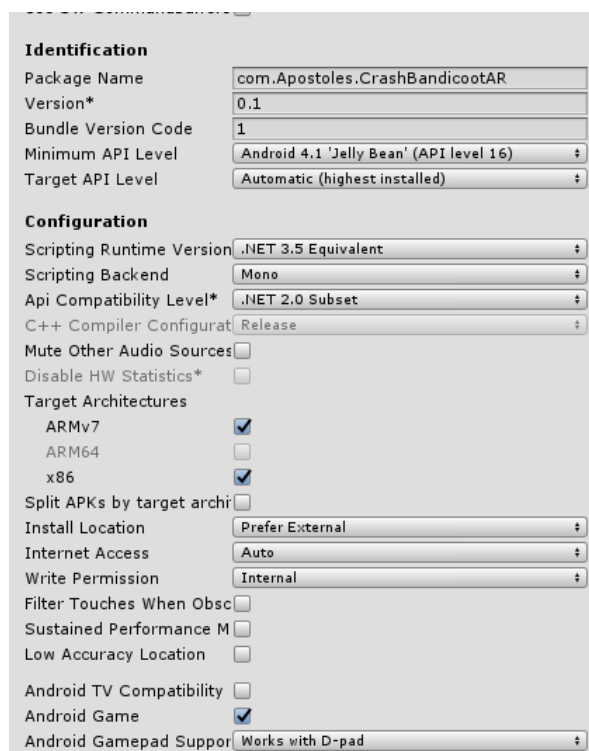
Εικόνα 31 Επιλογή Platform

Έπειτα player Settings έτσι ώστε να προσθέσουμε πληροφορίες για την εφαρμογή μας. Στο Company name βάζουμε το όνομα μας. στο Product Name όνομα της εφαρμογής. Προσθέτουμε ένα αντιπροσωπευτικό icon.



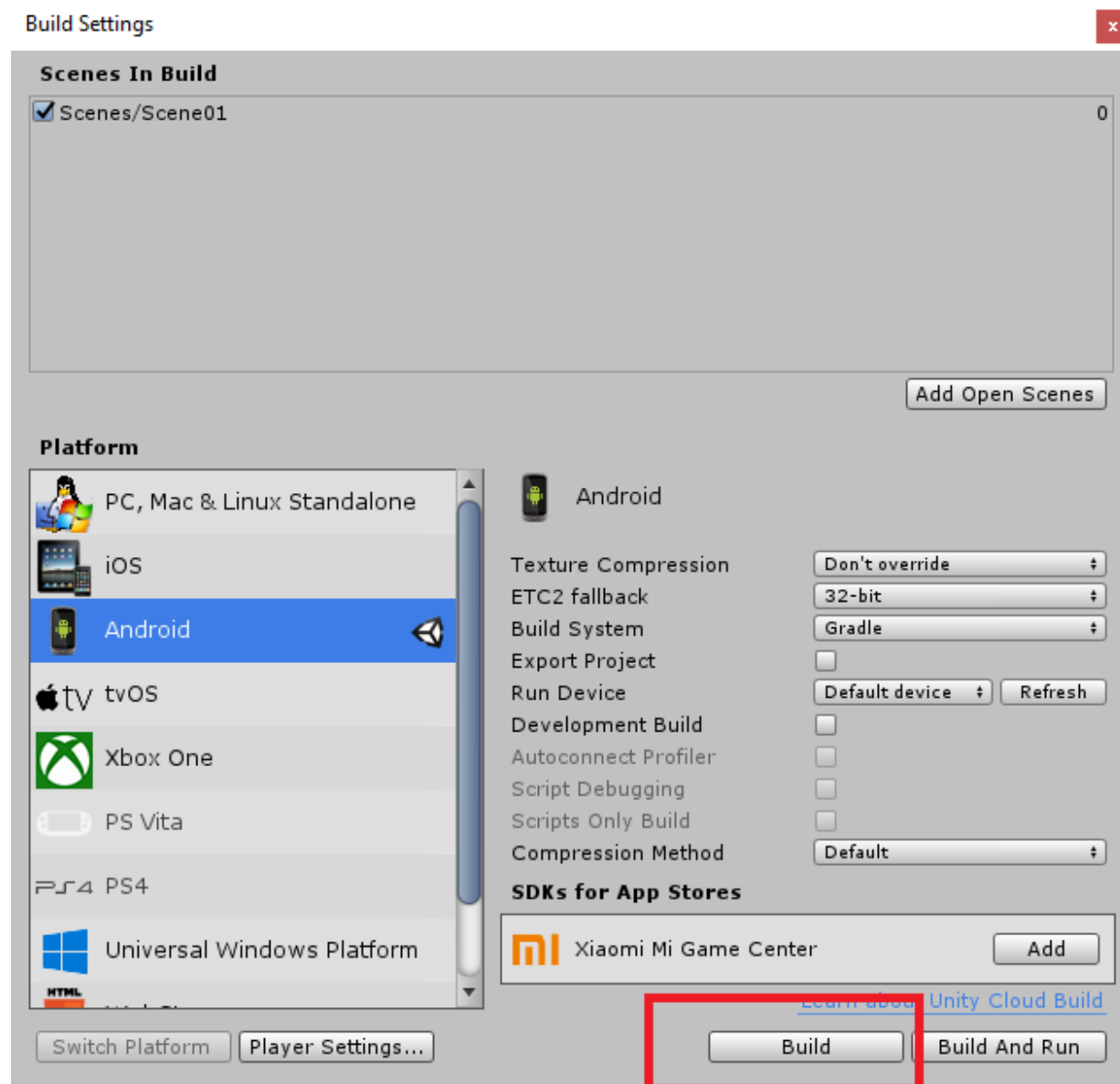
Εικόνα 32 Πληροφορίες Εφαρμογής

Στο other Settings Αλλάζουμε το Package name και αλλάζουμε την επιλογή Android TV Compatibility σε false



Εικόνα 33 Other Settings

Πίσω στο παράθυρο με τα build Settings, πατάμε στο Build και το αρχείο apk είναι έτοιμο για εγκατάσταση από τη συσκευή Android μας!



Εικόνα 34 Ολοκλήρωση Εφαρμογής