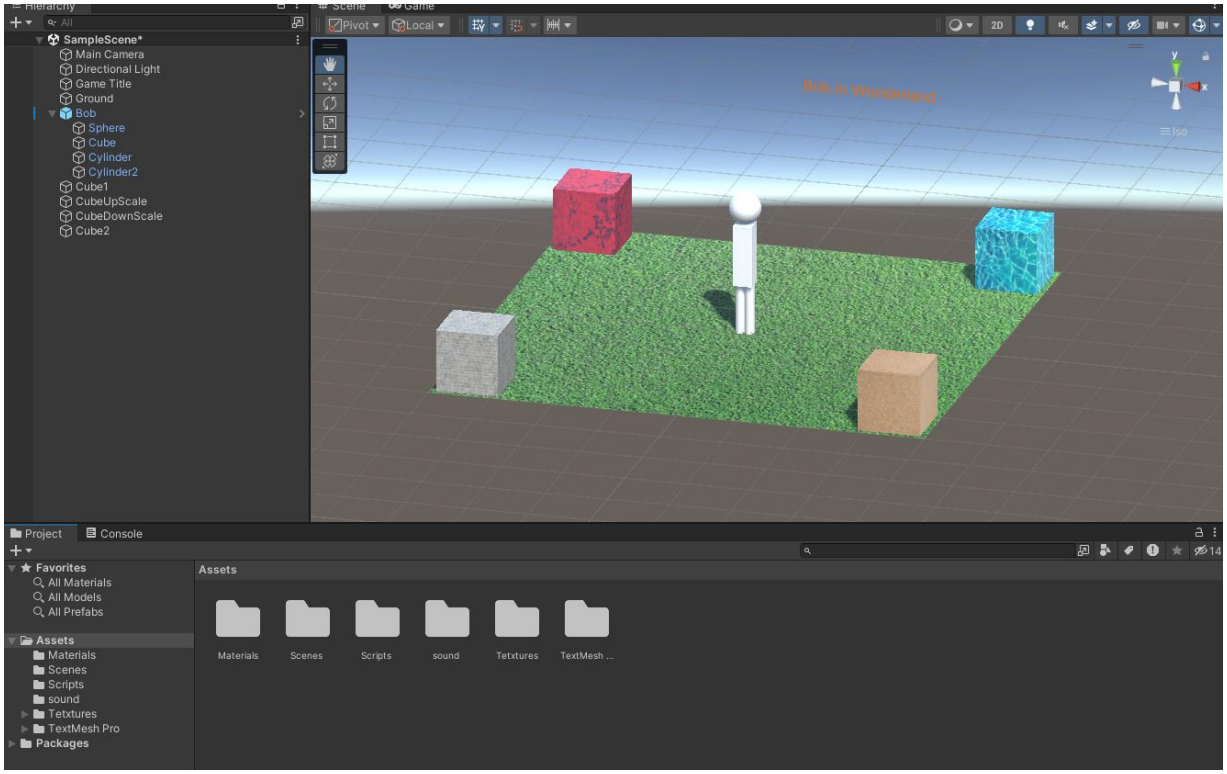
# Unity Bob in Wonderland

Vasileios Skarafigkas

Ημερομηνία: **20/12/2023**

## Περιεχόμενα

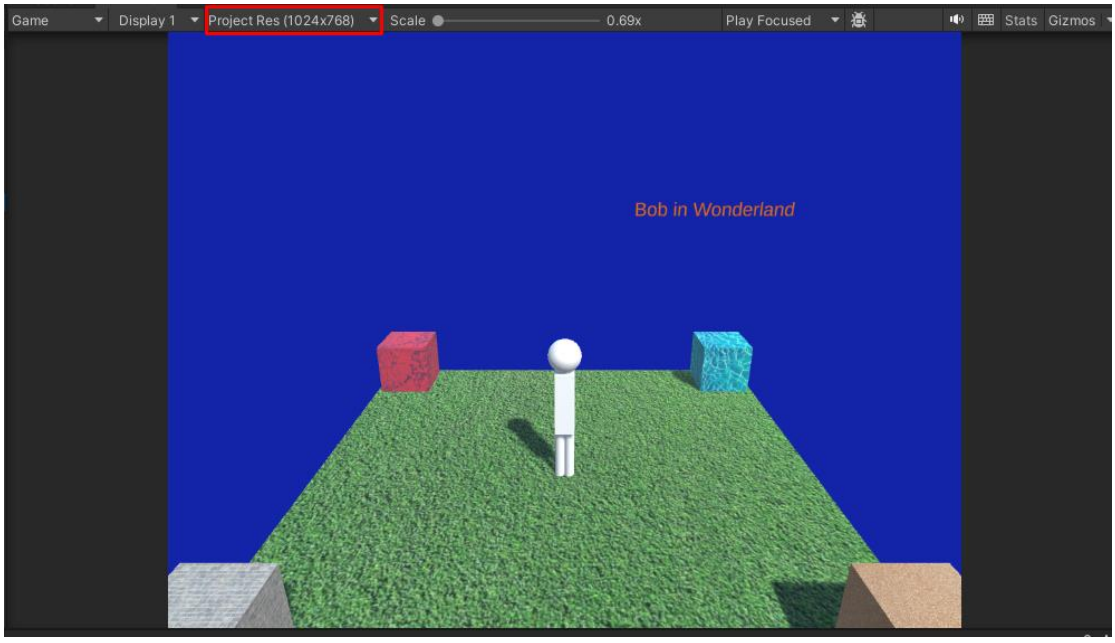
# WorkSpace




Έχουμε προσθέσει στο project φακέλους για τα script που θα χρησιμοποιηθούν στην άσκηση, ένα φάκελο sound που έχει τον ήχο για την σύγκρουση, έναν φάκελο με τα textures των κύβων και ένα φάκελο material που βρίσκεται μέσα ο Bob που δημιουργήσαμε. Τέλος στα δεξιά έχουμε την ιεραρχία της εφαρμογής. Όπως ο bob, οι κύβοι , το έδαφος, η κάμερα και ένα για τον τίτλο.

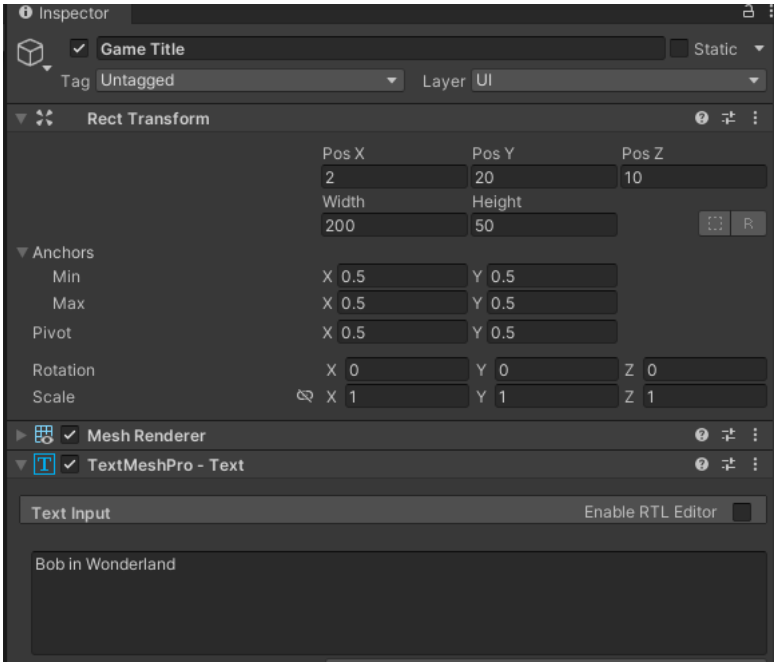
# Παράθυρο και Αντικείμενα

## Ανάλυση οθόνης



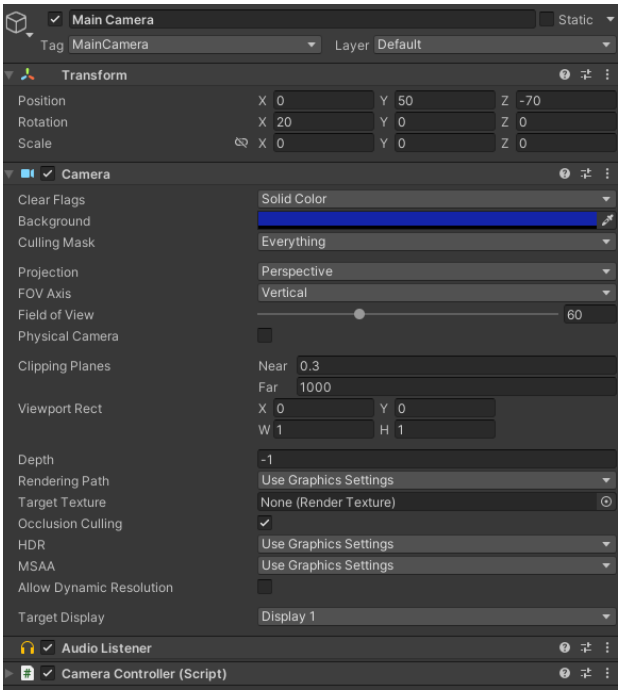
Η εφαρμογή ρυθμίζεται να τρέχει σε ανάλυση 1024x768. Αυτό γίνεται μέσω του Game View στο Unity όπου μπορούμε να επιλέξουμε την ανάλυση προβολής από τα διαθέσιμα presets.

## Τίτλος Εφαρμογής



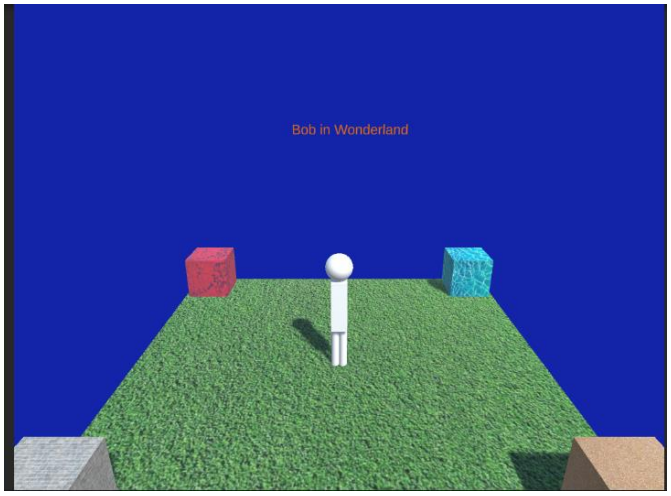

**Τίτλος Εφαρμογής**: Έχουμε προσθέσει έναν τίτλο στην εφαρμογή, "Bob in Wonderland", ο οποίος φαίνεται στην επάνω γωνία της προβολής μέσω ενός UI element, TextMesh Pro. Στο screenshot φαίνεται και η ακριβής θέση της κάμερας.

## Φόντο και Κάμερα

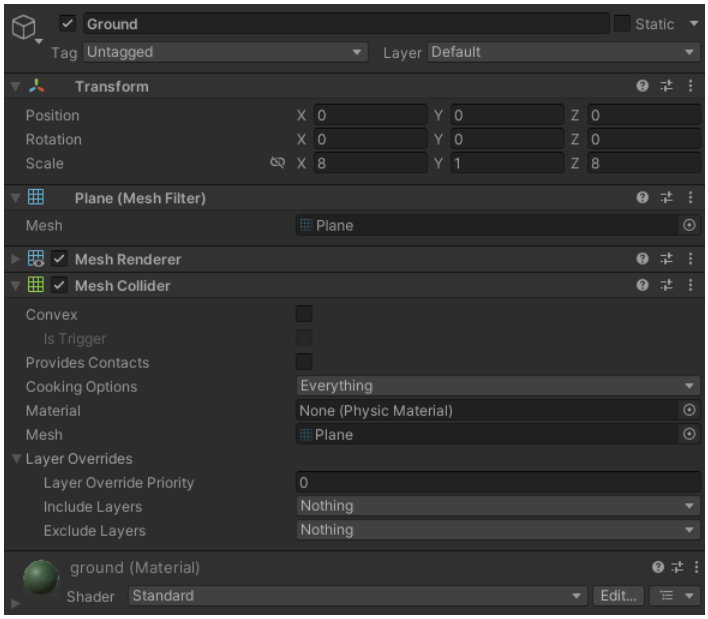



**Χρώμα Υποβάθρου και Κάμερα**: Το Main Camera GameObject έχει ρυθμιστεί να έχει ως χρώμα υποβάθρου το μπλε σκούρο, το οποίο γίνεται μέσω του component

Camera στο Unity, όπου μπορούμε να ορίσουμε το χρώμα του Clear Flags. Επίσης έχουμε αρχικοποιήσει την κάμερα να βλέπει σε σημείο που είναι διακριτός ο τίτλος και το τερέν που παίζουμε. Τέλος έχουμε 1 component για να ακούγονται οι ήχοι και ένα script “Camera Controller” για να κινούμε την κάμερα που θα αναλύσουμε παρακάτω.




## Έδαφος



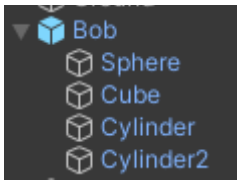

**Δημιουργία Επιφάνειας**: Ξεκινάμε δημιουργώντας μια επιφάνεια (Plane) στο Unity. Αυτό μπορεί να γίνει πατώντας δεξί κλικ στο παράθυρο της ιεραρχίας (Hierarchy) και επιλέγοντας 3D Object -> Plane.

**Ρυθμίσεις Transform**: Οι διαστάσεις του Plane στο Transform component πρέπει να ρυθμιστούν έτσι ώστε να αντιστοιχούν σε ένα τετράγωνο 80x80. Στο Unity, η προεπιλεγμένη μονάδα μέτρησης είναι το μέτρο, οπότε για να δημιουργήσουμε ένα έδαφος 80x80, θα πρέπει να ρυθμίσουμε το Scale του Plane σε X: 8 και Z: 8, εφόσον το προεπιλεγμένο Plane έχει μέγεθος 10x10.

**Υφή Εδάφους**: Για να προσθέσουμε μια υφή στο έδαφος, πρέπει πρώτα να έχουμε ένα αρχείο υφής (texture) στα Assets του project. Στη συνέχεια, δημιουργούμε ένα νέο Material και τοποθετούμε την υφή στο slot Albedo του Material. Το Material αυτό το σύρουμε και το αφήνουμε (drag and drop) πάνω στο Plane για να εφαρμοστεί η υφή.

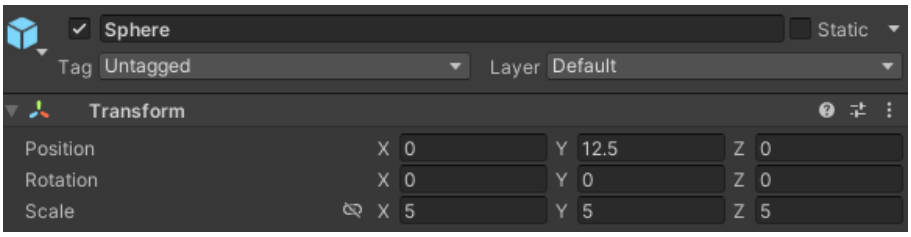
**Ρυθμίσεις Collider**: Το Plane πρέπει να έχει έναν Mesh Collider για να μπορεί ο Bob και οι κύβοι να αλληλεπιδρούν φυσικά με την επιφάνεια. Εφόσον ο Bob και οι κύβοι έχουν δικούς τους colliders, ο Mesh Collider του εδάφους διασφαλίζει ότι θα υπάρχει ανίχνευση συγκρούσεων.

## Bob

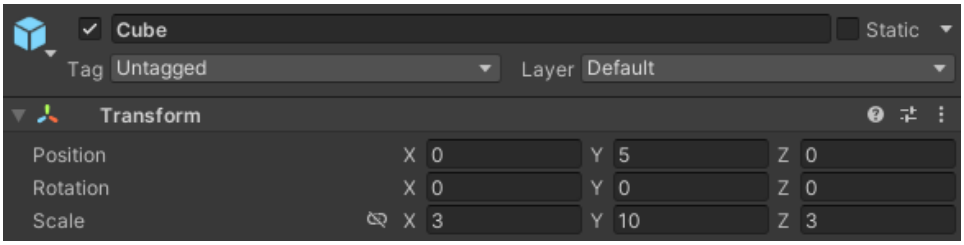



**Δομή του Bob**: Ο Bob αποτελείται από μία σφαίρα για το κεφάλι με κλιμάκωση 5, ένα παραλληλεπίπεδο για το σώμα με κλιμάκωση (3,10,3), και δύο κυλίνδρους για τα πόδια με κλιμάκωση (1.5,4,1.5). Τώρα οι θέσεις των αντικειμένων είναι ανάλογες του scale με σκοπό να φτιάξουμε τον Bob. Συγκεκριμένα:

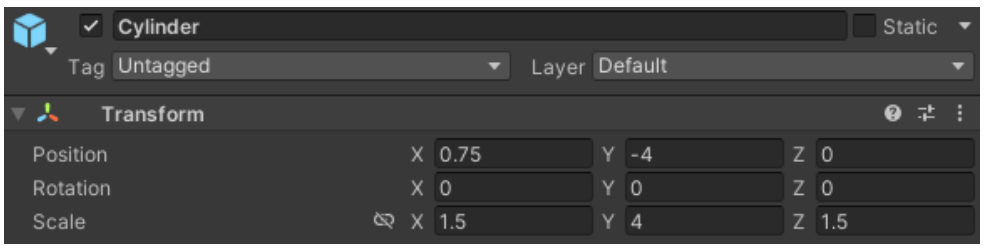
A) Κεφάλι(y.position) = πόδι(y.scale) + σώμα(y.sclae) + κεφάλι(y.scale)/2. Έτσι βρήκαμε το ύψος του κεφαλιού.

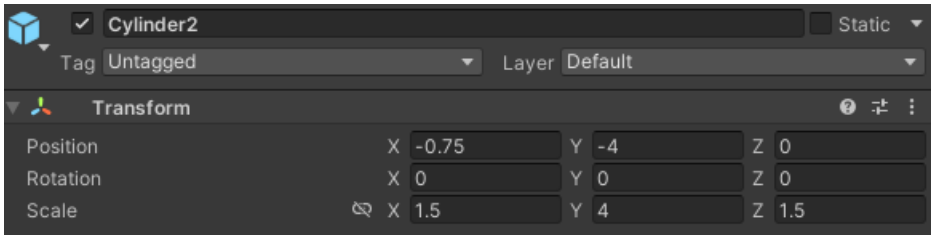


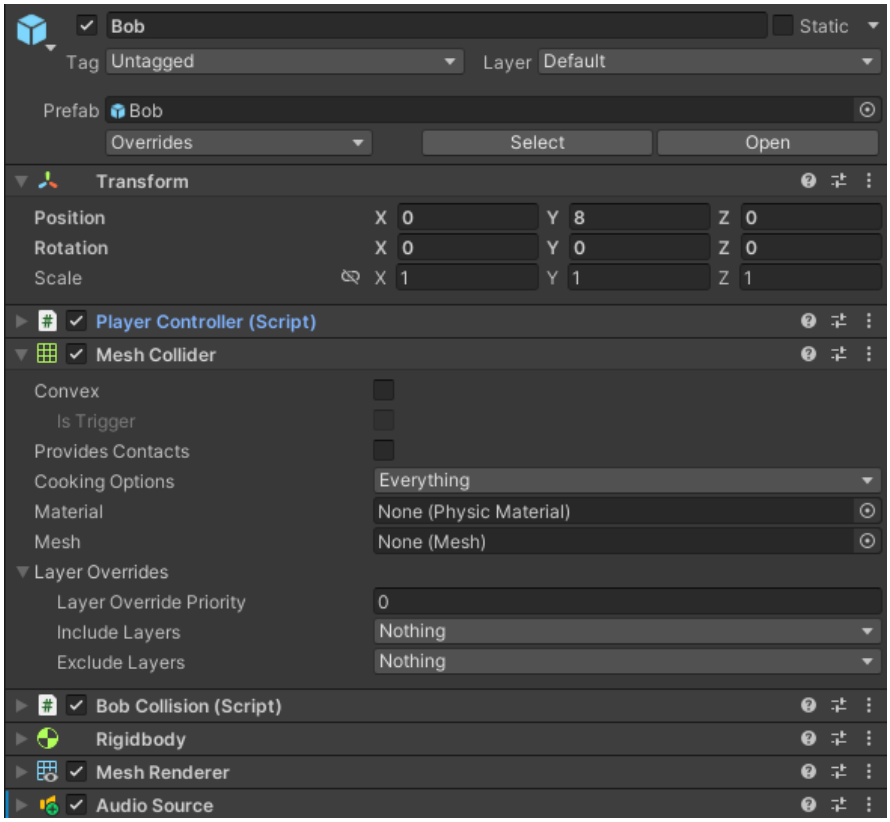

B) Σώμα(y.position) = σώμα(y.scale)/2 γιατί είναι το μεσαίο object του σώματος άρα ισαπέχει.




C) Πόδια στον χ.position = 0.75 και -0.75 γιατί ( |0.75| + |-0.75| ) + 1.5(x.scale) = 3 το οποίο ισούται με το scale του κορμού στον άξονα x.






**Ομαδοποίηση των στερεών αντικειμένων:** Όλα τα στερεά αντικείμενα που αποτελούν τον Bob έχουν ομαδοποιηθεί, σε ένα γονικό GameObject ονομαζόμενο "Bob". Αυτό επιτρέπει στον Bob να κινείται και να μετασχηματίζεται ως μία μονάδα.

**Το script BobCollision** διαχειρίζεται τις συγκρούσεις του Bob με τους κύβους, αλλάζοντας το υλικό (Material) του Bob και προσαρμόζοντας την κλιμάκωσή του ανάλογα με το αν πρόκειται για μεγέθυνση ή σμίκρυνση.

**Ήχος Συγκρούσεων**: Έχει προστεθεί ένας **AudioSource** στο GameObject του Bob για την αναπαραγωγή ήχου κατά τη σύγκρουση με τους κύβους.

**Κίνηση του Bob**: Το script **PlayerController** διαχειρίζεται την κίνηση του Bob στο επίπεδο.

## Κίνηση Bob

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


public class PlayerController : MonoBehaviour
{
    private float[] speedLevels = new float[] { 2.0f, 5.0f, 10.0f, 15.0f, 20.0f }; // Διαβαθμίσεις ταχύτητας
    private int currentSpeedLevel = 1; // Τρέχον επίπεδο ταχύτητας
    public float moveSpeed; // Ταχύτητα κίνησης του Bob

    private Vector3 minBounds = new Vector3(-40f, 0f, -40f); // Ελάχιστα όρια του εδάφους
    private Vector3 maxBounds = new Vector3(40f, 0f, 40f); // Μέγιστα όρια του εδάφους

    void Update()
    {
        // Αλλαγή ταχύτητας με τα πλήκτρα 1-5
        for (int i = 0; i < speedLevels.Length; i++)
        {
            if (Input.GetKeyDown((i + 1).ToString()))
            {
                currentSpeedLevel = i;
                moveSpeed = speedLevels[currentSpeedLevel];
            }
        }

        float xMovement = Input.GetKey(KeyCode.A) ? -1f : Input.GetKey(KeyCode.D) ? 1f : 0f;
        float zMovement = Input.GetKey(KeyCode.S) ? -1f : Input.GetKey(KeyCode.X) ? 1f : 0f;
        float yMovement = Input.GetKey(KeyCode.W) ? 1f : Input.GetKey(KeyCode.E) ? -1f : 0f;

        Vector3 moveDirection = new Vector3(xMovement, yMovement, zMovement).normalized;

        if (moveDirection.magnitude > 1f)
        {
            moveDirection.Normalize();
        }

        Vector3 newPosition = transform.position + moveDirection * moveSpeed * Time.deltaTime;

        newPosition.x = Mathf.Clamp(newPosition.x, minBounds.x, maxBounds.x);
        newPosition.z = Mathf.Clamp(newPosition.z, minBounds.z, maxBounds.z);

        transform.position = newPosition;
    }
}
```

### Ταχύτητα

**Διαβαθμίσεις Ταχύτητας**: Καθορίζουμε έναν πίνακα **speedLevels** που περιέχει πέντε διαφορετικές ταχύτητες. Αυτό επιτρέπει στον παίκτη να επιλέξει από διάφορα επίπεδα ταχύτητας για τον χαρακτήρα Bob.

**Εναλλαγή Ταχύτητας**: Μέσω της ενσωμάτωσης ενός loop στην **Update** function, ο χρήστης μπορεί να αλλάξει την ταχύτητα του Bob πατώντας τα πλήκτρα 1-5. Αυτό επιτυγχάνεται με τον έλεγχο για το πάτημα των αντίστοιχων πλήκτρων και την αντιστοίχιση της ταχύτητας με τον τρέχοντα επιλεγμένο επίπεδο ταχύτητας.

### Κίνηση και Όρια επιπέδου

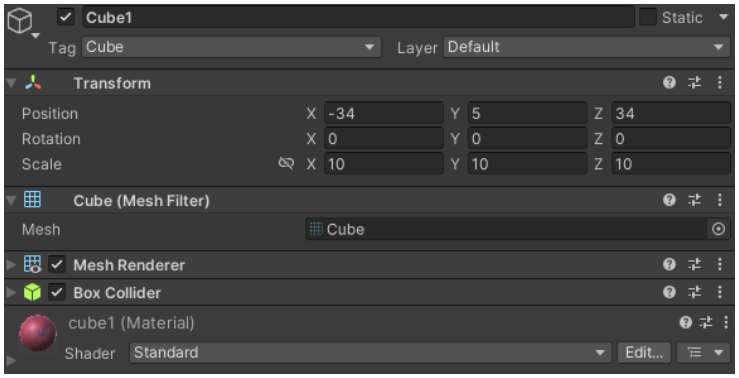
**Κίνηση Χαρακτήρα**: Το script αντιλαμβάνεται ποια πλήκτρα έχουν πατηθεί (A, D, S, X, W, E) και μετακινεί τον Bob στους άξονες x, y, z αντίστοιχα. Η κίνηση κανονικοποιείται με τη χρήση της **.normalized**, ώστε ο Bob να μην κινείται πιο γρήγορα όταν κινείται διαγώνια.

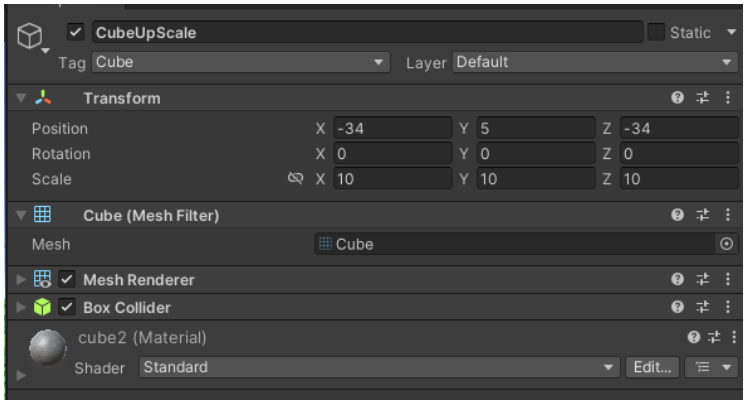
**Όρια Κίνησης**: Χρησιμοποιούμε την **Mathf.Clamp** για να διασφαλίσουμε ότι ο Bob δεν θα κινηθεί εκτός των ορίων του εδάφους και ότι δεν θα κινηθεί κάτω από το επίπεδο του εδάφους, ακόμη και όταν αλλάζει κλίμακα. Οι μεταβλητές **minBounds** και **maxBounds** καθορίζουν τα όρια στα οποία μπορεί να κινηθεί ο Bob και η αρχική θέση του Bob στον άξονα y καταγράφεται και χρησιμοποιείται για τον υπολογισμό του ελάχιστου επιτρεπτού y.

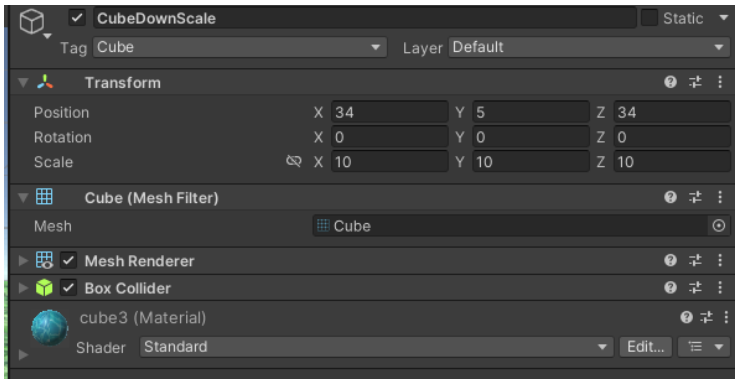
**Ενημέρωση Θέσης Χαρακτήρα**: Η νέα θέση του Bob υπολογίζεται με βάση την κατεύθυνση και την ταχύτητα κίνησης, και στη συνέχεια ενημερώνεται με την κατάλληλη εφαρμογή των όριων.

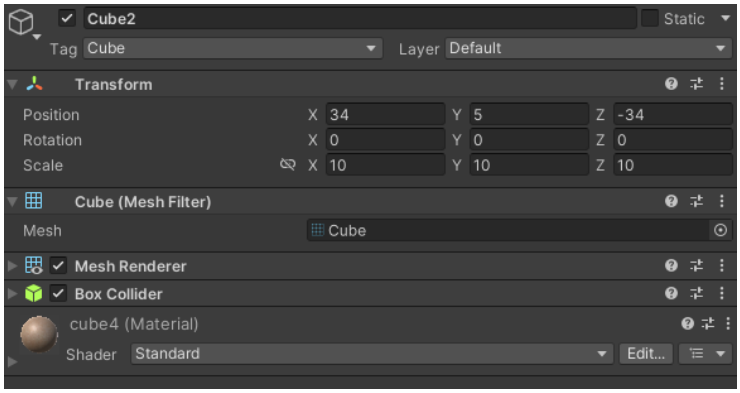
## Κύβοι και Σύγκρουση

### Τοποθέτηση και σχηματισμός κύβων



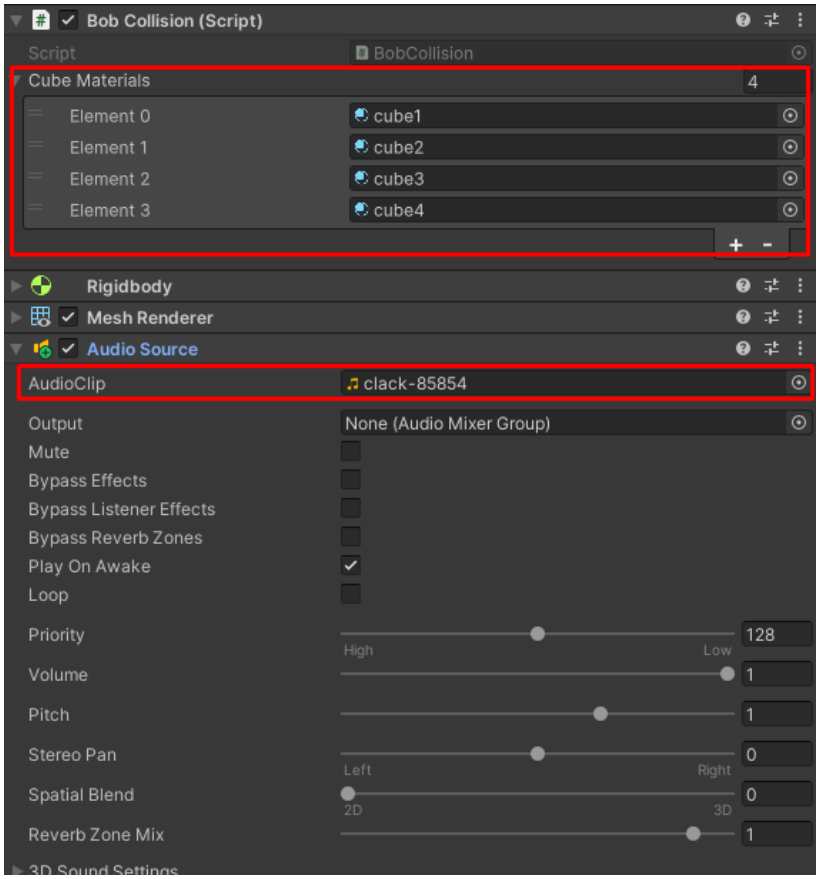
**Στοίχιση των Κύβων**: Στις τέσσερις γωνίες του εδάφους, έχουν τοποθετηθεί τέσσερις κύβοι με διαστάσεις 10x10x10. Αυτοί οι κύβοι έχουν οριστεί να είναι ακίνητοι, όπως φαίνεται από τα screenshots, όπου οι τοποθεσίες τους έχουν οριστεί σε σταθερά σημεία με θετικές και αρνητικές τιμές στους άξονες x και z, ενώ ο άξονας y είναι σταθερός στην τιμή 5 για να είναι πάνω από το έδαφος αφού το scale είναι ίσο με 10 .

**Αλλαγή Υφών και Μεγέθους του Bob**: Κάθε κύβος έχει ανατεθεί μια μοναδική υφή (Material), και δύο από αυτούς έχουν ειδικές λειτουργίες: ο "CubeUpScale" και ο "CubeDownScale". Όταν ο Bob ακουμπάει έναν από αυτούς τους δύο κύβους, αλλάζει το μέγεθός του με κλιμάκωση αντίστοιχα με το όνομα του κύβου.

## Διαχείριση Σύγκρουσης

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

public class BobCollision : MonoBehaviour
{
    public Material[] cubeMaterials; // Μια λίστα με τα Materials των κύβων
    private AudioSource audioSource; // Πηγή ήχου για την αναπαραγωγή ήχων συγκρούσεων

    void Start()
    {
        // Αποκτήστε πρόσβαση στο AudioSource component
        audioSource = GetComponent<AudioSource>();
    }

    void OnCollisionEnter(Collision collision)
    {
        if (collision.gameObject.CompareTag("Cube"))
        {
            foreach (Renderer renderer in GetComponentsInChildren<Renderer>())
            {
                renderer.material = collision.gameObject.GetComponent<Renderer>().material;
            }

            if (collision.gameObject.name == "CubeUpScale")
            {
                transform.localScale *= 1.1f;
            }
            else if (collision.gameObject.name == "CubeDownScale")
            {
                transform.localScale *= 0.9f;
            }

            // Αναπαραγωγή ήχου συγκρούσεων
            if (audioSource != null && !audioSource.isPlaying)
            {
                audioSource.Play();
            }
        }
    }
}
```

**Script BobCollision**: Το script **BobCollision** που είναι επισυναπτόμενο στο GameObject του Bob διαχειρίζεται τις συγκρούσεις. Όταν ο Bob συγκρούεται με έναν από τους κύβους, το script ελέγχει την ετικέτα (Tag) του GameObject που συγκρούεται και αλλάζει το Material του Bob σε αυτό του κύβου, χρησιμοποιώντας την **GetComponentInChildren<Renderer>()** για να βρει όλα τα Renderers που ανήκουν στον Bob.

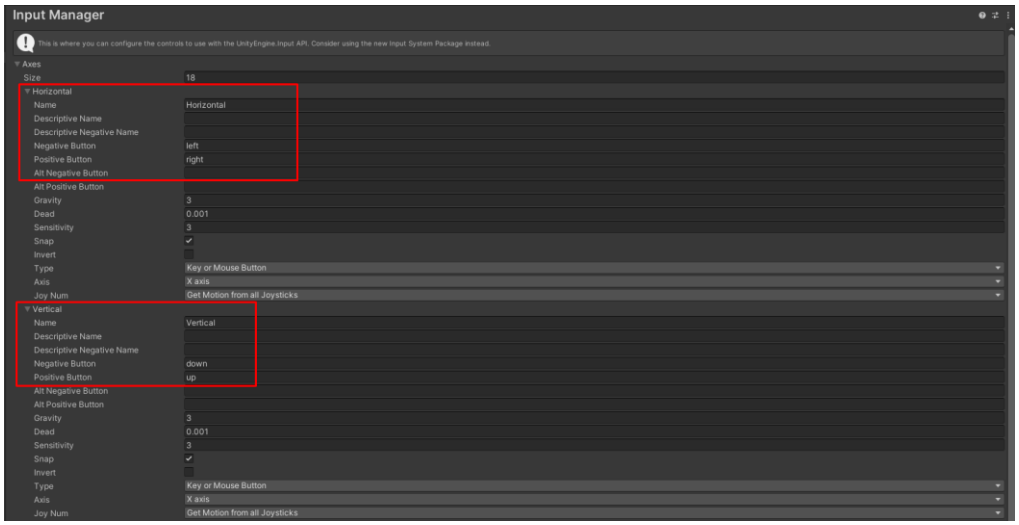



**Αναπαραγωγή Ήχου**: Το script επίσης διαθέτει ένα **AudioSource** component, το οποίο χρησιμοποιείται για να παίζει έναν ήχο όταν ο Bob ακουμπάει έναν κύβο. Ελέγχεται αν το **audioSource** δεν παίζει ήδη έναν ήχο πριν αρχίσει να παίζει τον νέο ήχο, ώστε να μην υπάρχει επικάλυψη ήχων.

**Κύβοι στο Unity**: Στο Unity Editor, κάθε κύβος έχει προστεθεί ως ξεχωριστό GameObject με Box Collider και μια συγκεκριμένη υφή (Material). Οι κύβοι έχουν μετονομαστεί ανάλογα με τις λειτουργίες τους, όπως "CubeUpScale" και "CubeDownScale", και τοποθετηθεί στις αντίστοιχες γωνίες του εδάφους.

# Κάμερα

## Κουμπιά αξόνων




Στο Unity Editor, οι αντιστοιχίσεις των κινήσεων με τα πλήκτρα έχουν αλλάξει μέσω του Input Manager για να αντιστοιχίζουν μόνο τα βελάκια με τις εντολές **Horizontal** και **Vertical**, αφαιρώντας έτσι τα προκαθορισμένα πλήκτρα w, s, d, a. Αυτό επιτυγχάνεται με την τροποποίηση των Negative και Positive Buttons στις αντίστοιχες εγγραφές του Input Manager, όπως φαίνεται στο παρεχόμενο screenshot.

## Ανάλυση

```
using UnityEngine;

public class CameraController : MonoBehaviour
{
    private float speed = 40f; // Κοινή ταχύτητα κίνησης και περιστροφής
    private Transform ground; // Μεταβλητή για το Transform του εδάφους

    void Start()
    {
        // Αναζητούμε το GameObject του εδάφους με όνομα "Ground" στο Unity Editor
        GameObject groundObject = GameObject.Find("Ground");
        if (!groundObject)
        {
            Debug.LogError("Ground object not found in the scene. Please ensure there is a GameObject named 'Ground'.");
            this.enabled = false; // Απενεργοποίηση του script αν δεν βρεθεί το ground
            return;
        }
        ground = groundObject.transform;
    }


    void Update()
    {
        // Κίνηση προς τα εμπρός και πίσω στον άξονα z της κάμερας
        float forwardMovement = Input.GetAxis("Vertical");
        transform.Translate(0, 0, forwardMovement * speed * Time.deltaTime);

        // Κίνηση πάνω και κάτω στον παγκόσμιο χώρο (στον άξονα y)
        if (Input.GetKey(KeyCode.Plus) || Input.GetKey(KeyCode.Equals) || Input.GetKey(KeyCode.KeypadPlus))
        {
            transform.Translate(0, speed * Time.deltaTime, 0, Space.World);
        }
        if (Input.GetKey(KeyCode.Minus) || Input.GetKey(KeyCode.KeypadMinus))
        {
            transform.Translate(0, -speed * Time.deltaTime, 0, Space.World);
        }

        // Περιστροφή γύρω από τον άξονα y του ground χρησιμοποιώντας το Horizontal axis
        float rotation = Input.GetAxis("Horizontal") * (-speed) * Time.deltaTime;
        transform.RotateAround(ground.position, Vector3.up, rotation);
    }
}
```

1. Εισαγωγή Κίνησης:
    - Χρήση της **Input.GetAxis("Vertical")** για να ελέγχουμε την κίνηση της κάμερας προς τα εμπρός και πίσω κατά μήκος του άξονα z, με χρήση των βελάκιων πάνω/κάτω.
    - Η κίνηση εφαρμόζεται με την **transform.Translate**, και η ταχύτητα κίνησης ελέγχεται από την μεταβλητή **speed**.
2. Περιστροφή Κάμερας:
    - Περιστροφή γύρω από τον άξονα x του αντικειμένου **ground** (το εδάφος), χρησιμοποιώντας την **Input.GetAxis("Horizontal")** για ανάγνωση των βελάκιων αριστερά/δεξιά.
    - Η περιστροφή εφαρμόζεται με την **transform.RotateAround**, και η ταχύτητα περιστροφής ελέγχεται επίσης από την μεταβλητή **speed**.
3. Κίνηση Κάθετα:
    - Χρήση των πλήκτρων <+> και <-> για την κίνηση της κάμερας πάνω και κάτω στον παγκόσμιο χώρο, κατά μήκος του άξονα y.
    - Η κίνηση κατά μήκος του άξονα y εφαρμόζεται με την **transform.Translate** στον παγκόσμιο χώρο (Space.World).

# Ομάδα και Περιβάλλον Εργασίας

## Περιβάλλοντα

i) Windows 11
ii) Visual Studio Community 2022
iii) Unity 2022.3.15f1
iv) .Exe folder: executable
v) Md5: 6970107E3004F957A3896B1F884EC055
vi) Link Drive: https://drive.google.com/drive/folders/17cFTSNSGqTOzkDTqQlkhQaB7eauYGfcN?usp=drive_link

## Ομάδα

Δεν αντιμετωπίσαμε ιδιαίτερη δυσκολία καθώς υπάρχουν πολλά tutorial στο διαδίκτυο για την unity. Η κίνηση και η δημιουργία όπως και το collision ήταν σχετικά εύκολα. Η δυσκολία που αντιμετωπίσαμε ήταν η κάμερα για τον τρόπο κίνησής της καθώς δεν την κατανοήσαμε εξαρχής. Χρησιμοποιήσαμε και το tutorial από το εργαστήριο κυρίως για την κάμερα αλλά και για την δημιουργία του εκτελέσιμου.

## Παράδειγμα Εκτέλεσης