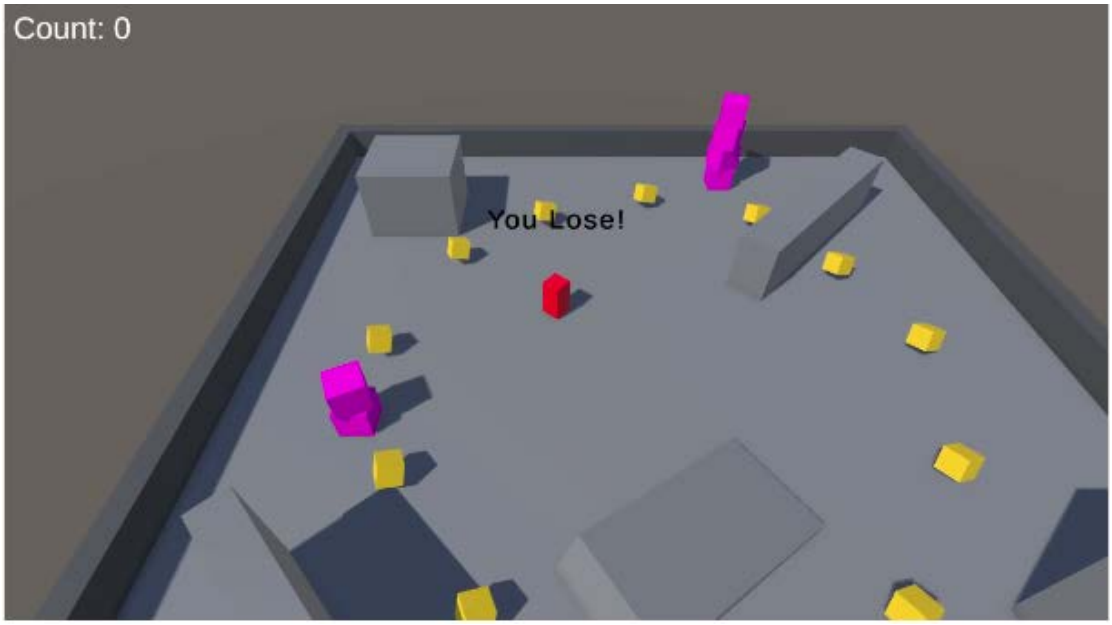
Προσθήκη AI Navigation

1. Επισκόπηση

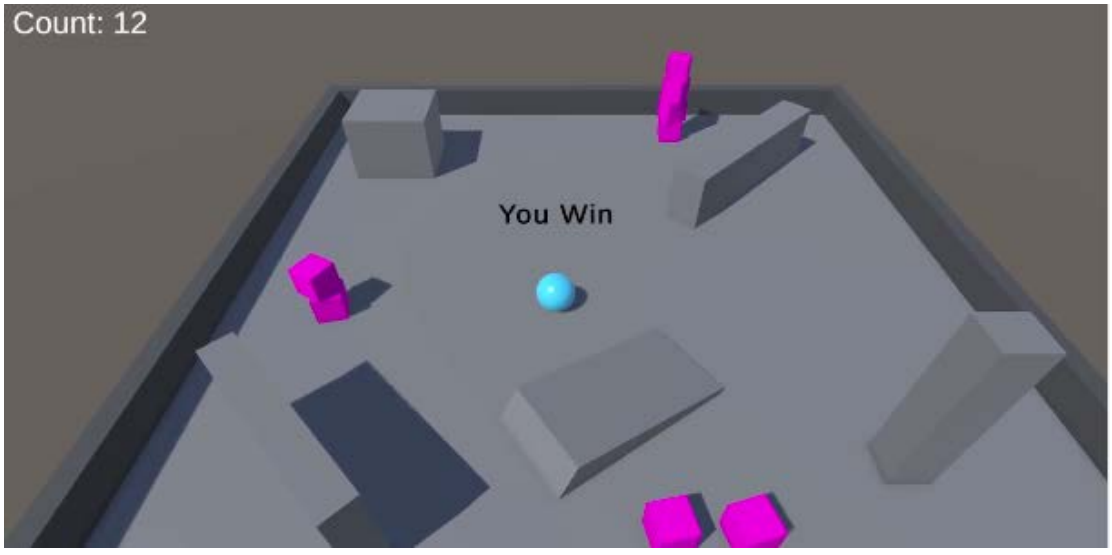
Σε αυτό το σεμινάριο, θα:

- Ρυθμίστε ένα βασικό περιβάλλον Unity 3D με GameObjects παίκτη και εχθρού.
- Εφαρμόστε το NavMesh για πλοήγηση εχθρού που βασίζεται σε AI.
- Προσθέστε στατικά και δυναμικά εμπόδια για να προκαλέσετε τον παίκτη.
- Συνθήκες νίκης και απώλειας κώδικα χρησιμοποιώντας scripts Unity.

Μέχρι το τέλος αυτού του σεμιναρίου, το παιχνίδι σας θα μοιάζει κάπως έτσι εάν ο παίκτης χάσει



Και το παιχνίδι σας θα μοιάζει κάπως έτσι εάν ο παίκτης κερδίσει:



2. Δημιουργήστε έναν εχθρό

1. Δημιουργήστε ένα κενό Enemy GameObject.

- Δημιουργήστε ένα κενό GameObject και μετονομάστε το σε "Enemy".
- Επαναφέρετε reset το στοιχείο **Transform** του **Enemy** GameObject για να το τοποθετήσετε στην αρχή.

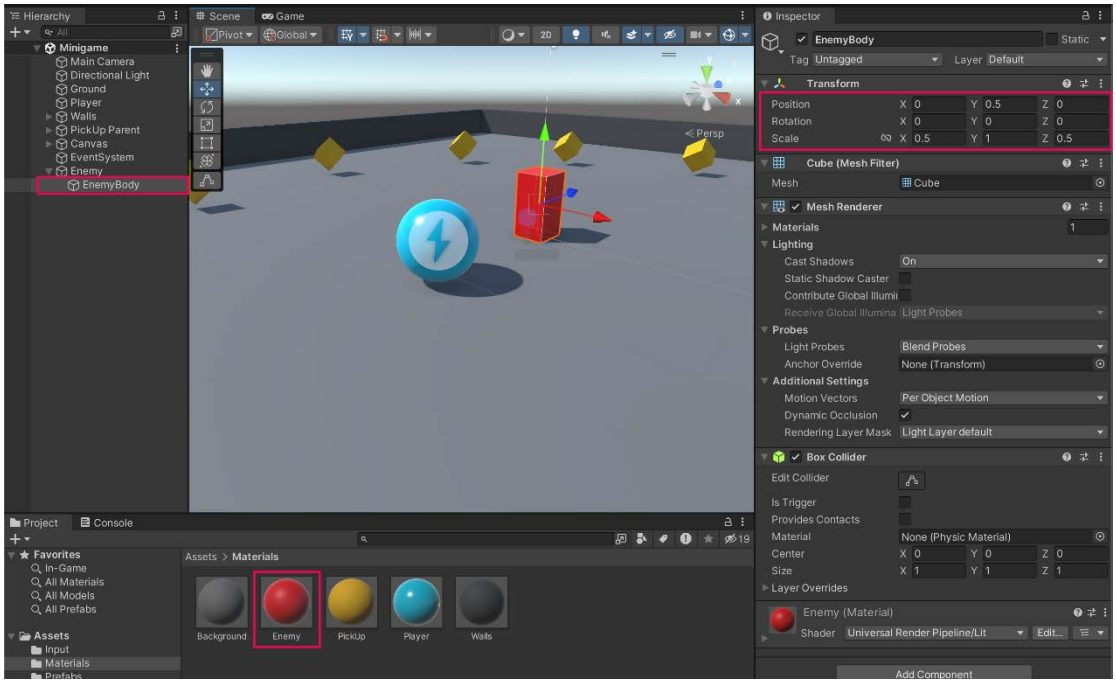
2. Προσθέστε έναν κύβο για το σώμα του εχθρού.

- Στο παράθυρο **Ιεραρχία** , κάντε δεξί κλικ στο **Enemy** GameObject και επιλέξτε **3D Object** > **Cube** και μετά μετονομάστε το σε "EnemyBody".
- Ορίστε τις τιμές **Κλίμακα** σε **0,5, 1, 0,5** έτσι ώστε να είναι πιο ψηλό ορθογώνιο σχήμα.
- Ρυθμίστε τη **Θέση** του **EnemyBody** σε **0, 0,5, 0** για να το τοποθετήσετε πάνω από το **Ground** GameObject.

3. Δημιουργήστε ένα υλικό για τον εχθρό.

- Κάντε δεξί κλικ μέσα στο φάκελο **Υλικά** και επιλέξτε **Δημιουργία** > **Υλικό** και μετά μετονομάστε το υλικό σε "Εχθρός".
- Ρυθμίστε το χρώμα **Base Map του υλικού** σε όποιο χρώμα θέλετε για **το Enemy** GameObject σας.
- Σύρετε και αποθέστε το υλικό **του Enemy** στο **EnemyBody** GameObject στην προβολή **σκηνής** ή στο παράθυρο **Ιεραρχία** .

Σημείωση: Για ευκολότερη επεξεργασία, ίσως θέλετε να απομακρύνετε λίγο το **Enemy GameObject.** Θυμηθείτε να μετακινήσετε το γονικό **Enemy** GameObject και όχι μόνο το **EnemyBody** παιδί GmeObject.



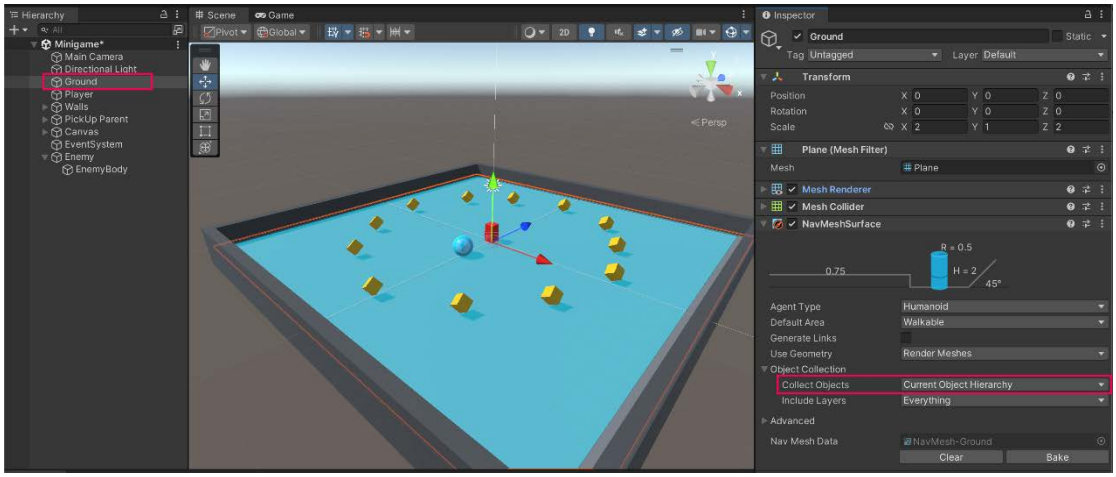
3. Δημιουργείστε (Bake) ένα NavMesh

1. Δημιουργείστε-Bake ένα NavMesh στο έδαφος GameObject.

- Επιλέξτε το **Ground** GameObject.
- Στο παράθυρο **του Inspector** , επιλέξτε **Προσθήκη στοιχείου** > **NavMeshSurface** .
- Επιλέξτε **Bake** στο στοιχείο **NavMeshSurface** .

2. Διαμορφώστε τι να συμπεριλάβετε στην επιφάνεια NavMesh.

- Στο παράθυρο **Inspector** , στις ρυθμίσεις στοιχείων **NavMeshSurface** , χρησιμοποιήστε το δίπλωμα (τρίγωνο) για να αναπτύξετε τη μονάδα **Object Collection .**
- Ανοίξτε το αναπτυσσόμενο μενού ιδιοτήτων **Συλλογή αντικειμένων (Object Collection)**και επιλέξτε **Ιεραρχία τρέχοντος αντικειμένου (Current Object Hierarchy**.
- Επιλέξτε ξανά το κουμπί **Bake** .
- Τα συλλεκτικά αντικείμενα **PickUp** GameObject θα πρέπει τώρα να αγνοηθούν στο NavMesh.



1. Κάντε τον Enemy Agent NavMesh.

- Επιλέξτε το **Enemy** GameObject στο παράθυρο **Hierarchy** .
- Στο παράθυρο **Επιθεωρητής** , επιλέξτε **Add Component** > **Nav Mesh Agent**. **Προσθήκη στοιχείου** > **Πράκτορας πλέγματος πλοήγησης** .
- Ορίστε την ιδιότητα **Ταχύτητα σε μια τιμή περίπου 2,5** .

2. Προσθέστε ένα νέο σενάριο EnemyMovement.

- Με επιλεγμένο το **Enemy** GameObject στο παράθυρο **Hierarchy** , επιλέξτε **Add Component** > **New script** στο παράθυρο **Inspector** .
- Ονομάστε το νέο σας σενάριο "EnemyMovement".
- Στο παράθυρο **Project** , μετακινήστε το σενάριο από τον ριζικό φάκελο **Assets** στον φάκελο **Scripts** .
- Ανοίξτε το νέο σενάριο για επεξεργασία.

3. Προσθέστε μια μεταβλητή για αναφορά στο στοιχείο Nav Mesh Agent.

- Προσθέστε την ακόλουθη δήλωση "using" στην κορυφή του σεναρίου, ακριβώς κάτω από τη γραμμή χρησιμοποιώντας το UnityEngine:

  using UnityEngine.AI;

- Πάνω από τη συνάρτηση **Start** , προσθέστε τις ακόλουθες δύο νέες μεταβλητές:

  public Transform player;

  private NavMeshAgent navMeshAgent;

- Στη συνάρτηση **Start** , προσθέστε την ακόλουθη γραμμή κώδικα για να εκχωρήσετε τη μεταβλητή **NavMeshAgent** :

  navMeshAgent = GetComponent<NavMeshAgent>();

4. Ορίστε τον προορισμό του Enemy GameObject ως θέση του Player GameObject.

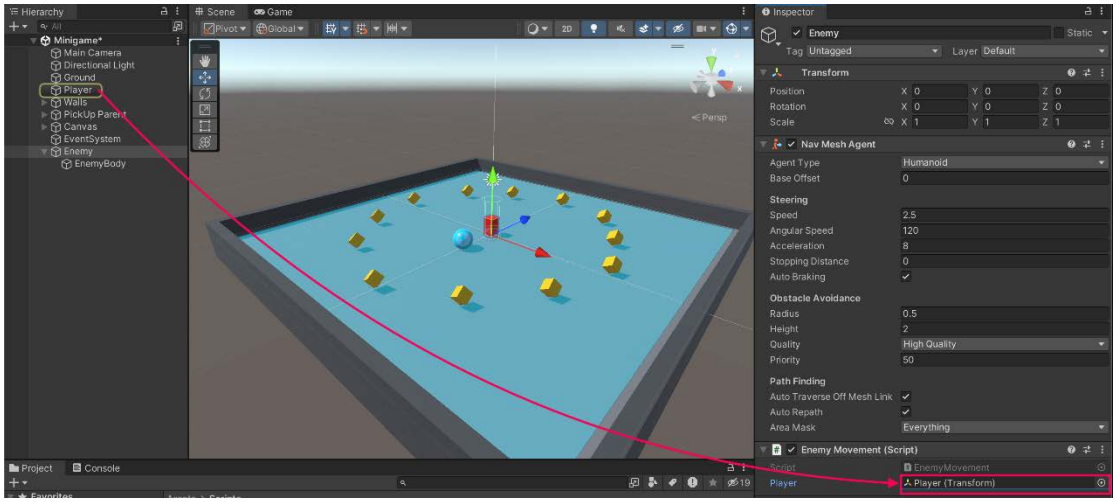
- Στη συνάρτηση **Update** , προσθέστε τον ακόλουθο κώδικα για να ενημερώσετε τον προορισμό του **Enemy** GameObject στο **Player** GameObject:

```
if (player != null)
{
    navMeshAgent.SetDestination(player.position);
}
```

5. Εκχωρήστε τη μεταβλητή του player.

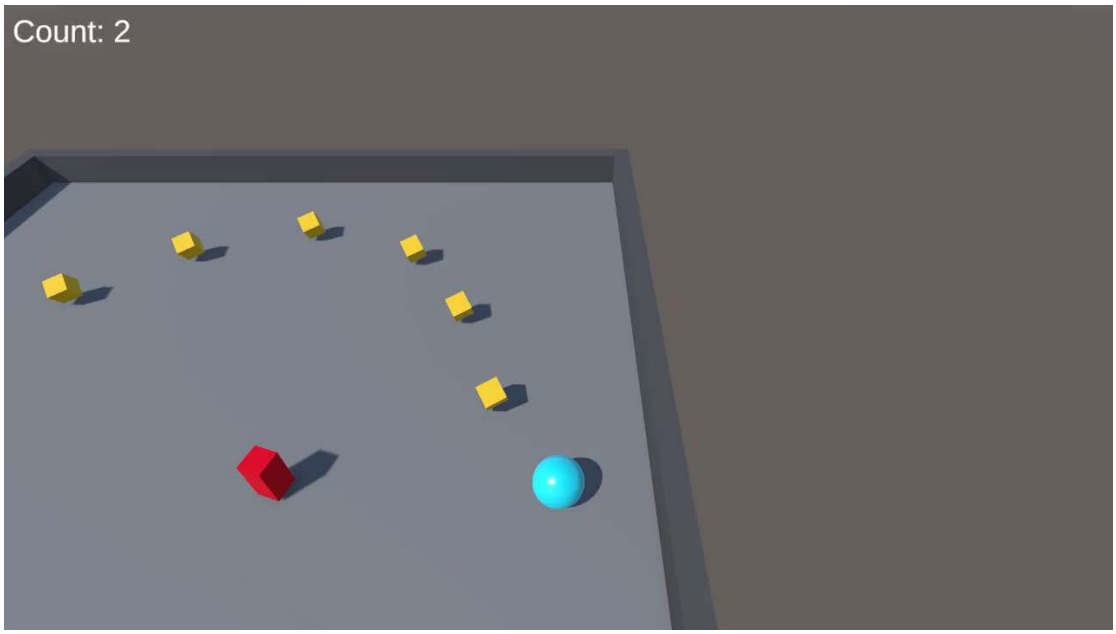
- Αποθηκεύστε το σενάριο και επιστρέψτε στο Unity Editor.
- Επιλέξτε το **Enemy** GameObject στο παράθυρο **Hierarchy** και, στη συνέχεια, σύρετε το **Player** GameObject στην υποδοχή **Player** του στοιχείου σεναρίου **EnemyMovement** στο παράθυρο **Inspector** .

Σημαντικό: Αυτό το βήμα είναι πολύ σημαντικό και εύκολο να το χάσετε. Εάν δεν κάνετε αυτό το βήμα, θα δείτε ένα σφάλμα **NullReferenceException** στο παράθυρο της **Κονσόλας** και το παιχνίδι σας δεν θα λειτουργήσει.



6. Δοκιμάστε το παιχνίδι.

- Αποθηκεύστε τη σκηνή και τρέξτε το παιχνίδι.
- Το **Enemy** GameObject θα πρέπει τώρα να κυνηγήσει το **Player** GameObject.
- Εάν δεν λειτουργεί, μπορείτε να δείτε τα πλήρη δείγματα σεναρίου στο τέλος αυτού του σεμιναρίου.



5. Δημιουργήστε στατικά εμπόδια

. Δημιουργήστε μια ποικιλία από εμπόδια.

- Δημιουργήστε ένα νέο κύβο GameObject και, στη συνέχεια, χρησιμοποιήστε τα εργαλεία κίνησης, περιστροφής και κλίμακας για να το μετατρέψετε σε ένα ενδιαφέρον εμπόδιο.
- Επαναλάβετε αυτή τη διαδικασία για να δημιουργήσετε επιπλέον εμπόδια διαφορετικών σχημάτων και μεγεθών.
- Φροντίστε να μετατρέψετε τουλάχιστον ένα αντικείμενο σε ράμπα για να δοκιμάσετε τις ικανότητες του **Enemy** GameObject σας για αναρρίχηση σε πλαγιά.

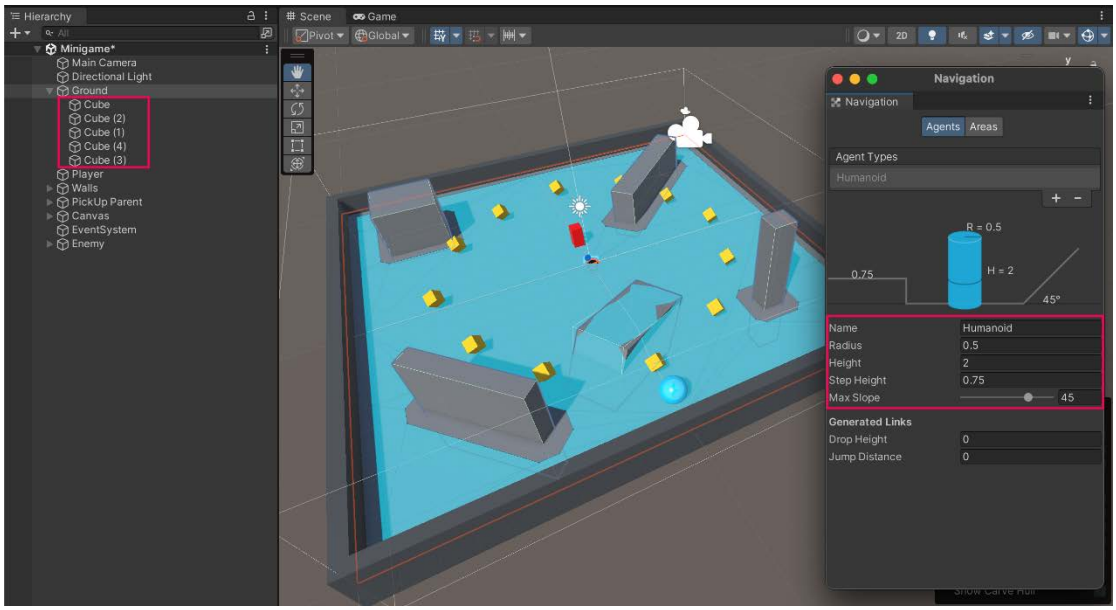
2. Συμπεριλάβετε τα εμπόδια στο στοιχείο NavMesh Surface.

- Στο παράθυρο **Hierarchy** , σύρετε κάθε εμπόδιο GameObject στο **Ground** GameObject για να τα κάνετε child GameObjects.

- Επιλέξτε το **Ground** GameObject και επιλέξτε ξανά **Bake** στο στοιχείο **NavMesh Surface** για να αναδημιουργήσετε το NavMesh με την ενημερωμένη διαμόρφωση εμποδίου.
- Θα πρέπει τώρα να δείτε την περιοχή γύρω από τα εμπόδια που είναι σκαλισμένα από την μπλε επικάλυψη **NavMesh Surface** στη σκηνή.

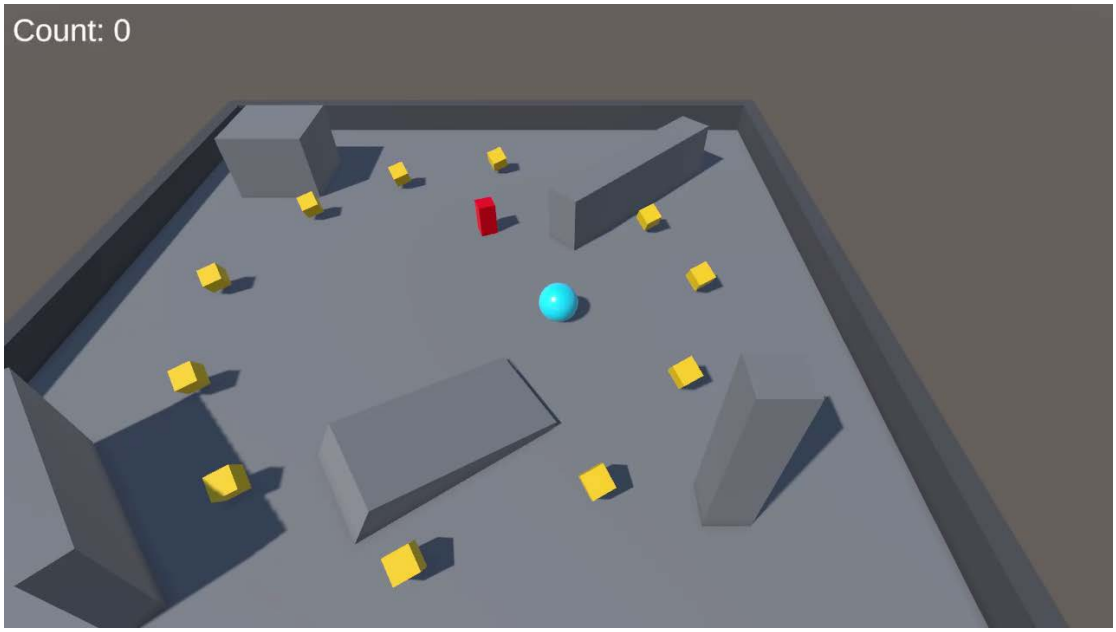
3. Προσαρμόστε τις ρυθμίσεις πράκτορα.

- Στο στοιχείο **NavMesh Surface** , από το αναπτυσσόμενο μενού **Agent Type** , επιλέξτε **Open Agent Settings** . Εναλλακτικά, μπορείτε να επιλέξετε **Window** > **AI** > **Navigation(Παράθυρο** > **AI** > **Πλοήγηση.)**
- Η αύξηση του ύψους του βήματος θα επιτρέψει στον Agent να ανέβει σε υψηλότερες επιφάνειες.
- Η αύξηση της μέγιστης κλίσης θα επιτρέψει στον πράκτορα να ανέβει σε πιο απότομους λόφους.



4. Δοκιμάστε το παιχνίδι σας

- Δοκιμάστε το παιχνίδι σας και πειραματιστείτε με διαφορετικές ρυθμίσεις πρακτόρων.



6. Δημιουργήστε δυναμικά εμπόδια

1. Δημιουργήστε έναν ελαφρύ, κινούμενο κύβο.

- Δημιουργήστε ένα νέο κύβο GameObject και μετονομάστε το σε "DynamicBox".
- Κλιμακώστε και τοποθετήστε το κουτί σύμφωνα με τις προτιμήσεις σας στη σκηνή.
- Δημιουργήστε ένα νέο υλικό με το όνομα "Dynamic Obstacle", δώστε του ένα χρώμα και αντιστοιχίστε το στον νέο κύβο.

- Στο παράθυρο **Inspector** , προσθέστε ένα νέο στοιχείο **Rigidbody και, στη συνέχεια, μειώστε τη μάζα MASS** σε περίπου **0.1** , ώστε να μπορεί να μετακινηθεί πιο εύκολα.
    - Δοκιμάστε τη σκηνή και παρατηρήστε ότι το **Enemy** GameObject περνά κατευθείαν μέσα από το δυναμικό εμπόδιο.

2. Κάντε τον κύβο εμπόδιο NavMesh.

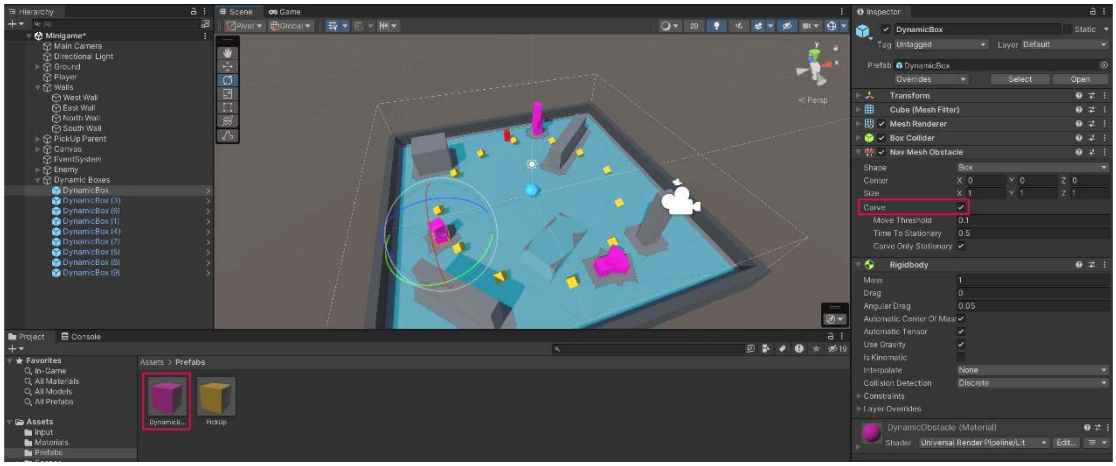
- Επιλέξτε το **DynamicBox** GameObject και, στη συνέχεια, αναζητήστε και προσθέστε το στοιχείο **NavMesh Obstacle** .
- Ενεργοποιήστε την επιλογή **Carve** στο στοιχείο **NavMesh Obstacle** .

3. Μετατρέψτε το σε προκατασκευασμένο.

- Σύρετε το **DynamicBox** GameObject από το παράθυρο **Hierarchy** στον φάκελο **Prefabs** στο παράθυρο **Project** . Αυτό θα δημιουργήσει μια προκατασκευή του **DynamicBox** GameObject.

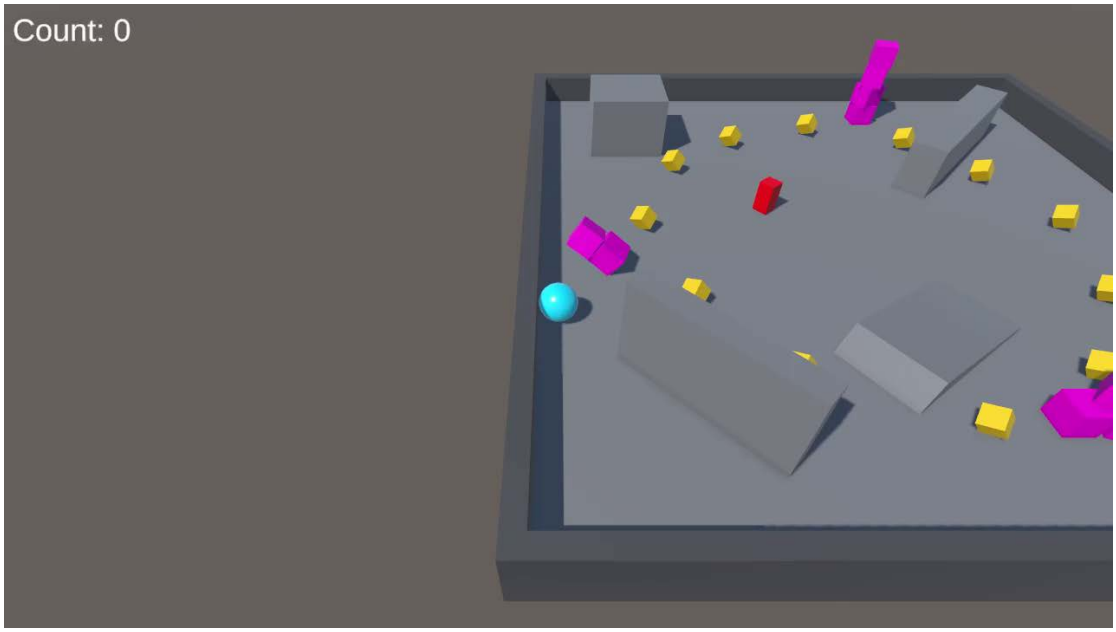
- Αντιγράψτε και διασκορπίστε τα προκατασκευασμένα στιγμιότυπα **του DynamicBox** στην περιοχή παιχνιδιού όπως θέλετε.

Σημείωση: Μπορεί να θέλετε να δημιουργήσετε ένα κενό γονικό αντικείμενο για να κρατάτε όλα αυτά τα **DynamicBox GameObjects για να διατηρείτε το παράθυρο Ιεραρχίας** σας τακτοποιημένο και οργανωμένο.



4. Δοκιμάστε το παιχνίδι σας.

- Πειραματιστείτε με διαφορετικές στοίβες δυναμικών εμποδίων για να φτιάξετε την πιο διασκεδαστική περιοχή παιχνιδιού.



7. Θέστε τις προϋποθέσεις νίκης και ήττας

1. Προσθέστε μια συνθήκη απώλειας που καταστρέφει τον παίκτη και λέει "Χάνεις!"

- Ανοίξτε το σενάριο **του PlayerController .**
- Προσθέστε την ακόλουθη νέα συνάρτηση **OnCollisionEnter** πριν από την τελική αγκύλη στο σενάριο:

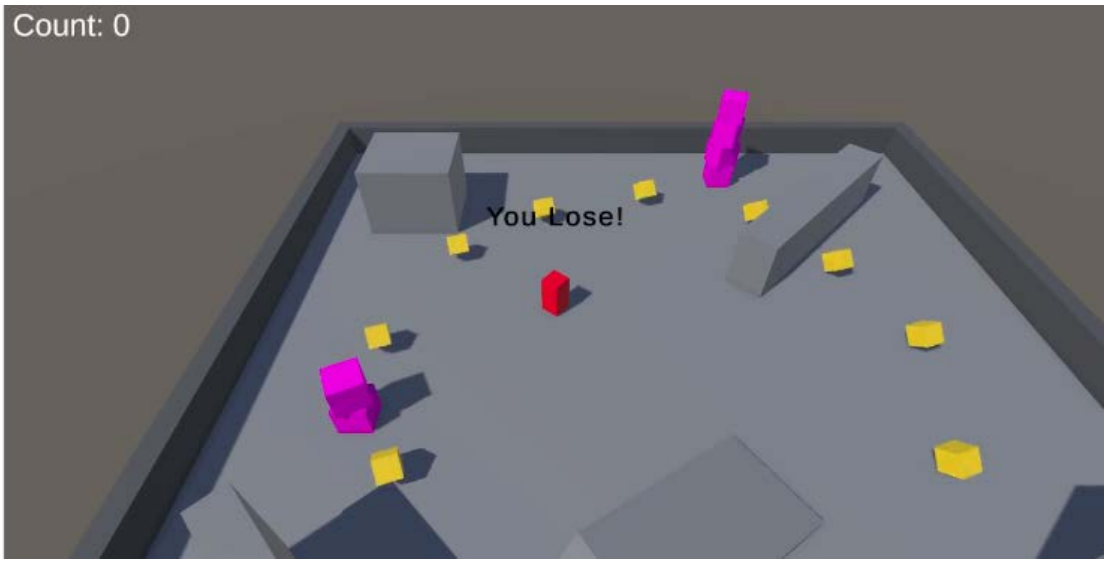
```
private void OnCollisionEnter(Collision collision)
{
  if (collision.gameObject.CompareTag("Enemy"))
  {
    // Destroy the current object
    Destroy(gameObject);
    // Update the winText to display "You Lose!"
    winTextObject.gameObject.SetActive(true);
    winTextObject.GetComponent<TextMeshProUGUI>().text = "You Lose!";
  }
}
```

2. Προσθέστε μια ετικέτα "Enemy" στο EnemyBody GameObject.

- Αποθηκεύστε το σενάριο και επιστρέψτε στο Unity Editor.
- Επιλέξτε το **EnemyBody** GameObject στο παράθυρο **Hierarchy** .
- Στο παράθυρο **Επιθεωρητής** , εντοπίστε το αναπτυσσόμενο μενού **Tag Ετικέτα** και επιλέξτε **Add Tag Προσθήκη ετικέτας** .
- Επιλέξτε το κουμπί **Προσθήκη** (**+**) για να δημιουργήσετε μια νέα ετικέτα και να την ονομάσετε "Enemy". Βεβαιωθείτε ότι η χρήση κεφαλαίων αντιστοιχεί σε αυτό που γράψατε στον κώδικά σας.
- Αποθηκεύστε τη νέα ετικέτα.
- Στο παράθυρο **Inspector** , επιλέξτε ξανά το **EnemyBody** GameObject και ορίστε την ετικέτα του στη νέα ετικέτα **Enemy** . Βεβαιωθείτε ότι έχετε

εφαρμόσει την ετικέτα στο **EnemyBody child** παιδί GameObject και όχι στο **Enemy parent** γονικό GameObject.

- Εκτελέστε το παιχνίδι για δοκιμή — τώρα, όταν το **Enemy** GameObject συγκρούεται με τον παίκτη, το **Player** GameObject καταστρέφεται και το ενημερωμένο κείμενο εμφανίζει "Χάνεις!"



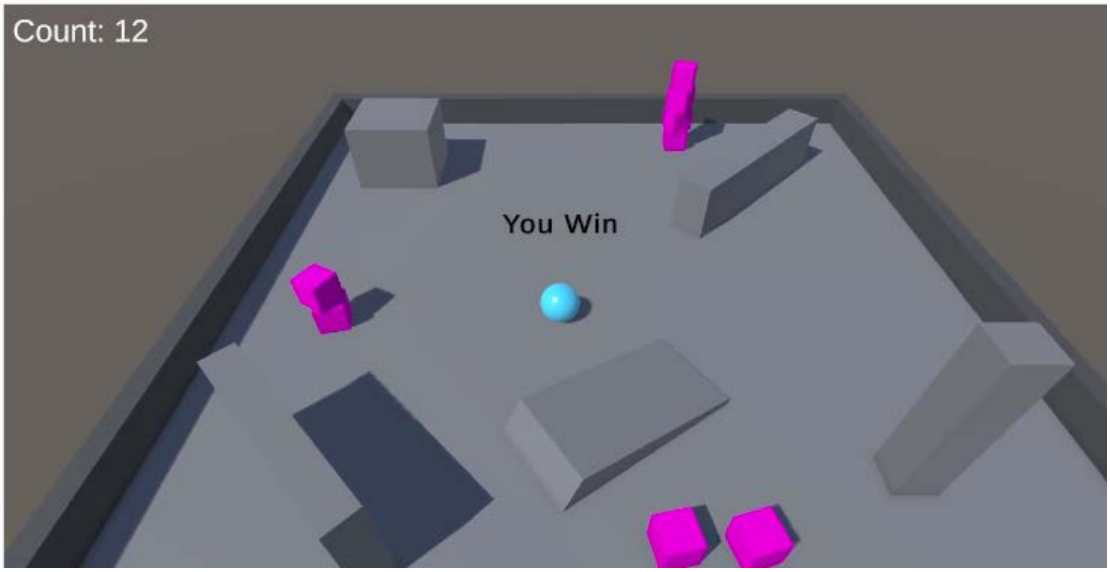
3. Καταστρέψτε το Enemy GameObject όταν ο παίκτης κερδίζει.

- Ανοίξτε ξανά το σενάριο **του PlayerController .**
- Στη συνάρτηση SetCountText, προσθέστε την ακόλουθη γραμμή κώδικα για να καταστρέψετε το **Enemy** GameObject:

```
Destroy(GameObject.FindGameObjectWithTag("Enemy"));
```

4. Δοκιμάστε το ολοκληρωμένο παιχνίδι σας.

- Αποθηκεύστε το σενάριο και επιστρέψτε στο πρόγραμμα επεξεργασίας.
- Εκτελέστε το παιχνίδι για να δοκιμάσετε την κατάσταση νίκης. Όταν συλλέγετε όλα τα **PickUp** GameObjects, το **Enemy** GameObject καταστρέφεται.



Τελικά δείγματα σεναρίου

PlayerController.cs

```
using UnityEngine;
using UnityEngine.InputSystem;
using TMPro;

public class PlayerController : MonoBehaviour
{
 // Rigidbody of the player.
 private Rigidbody rb;

 // Variable to keep track of collected "PickUp" objects.
 private int count;

 // Movement along X and Y axes.
 private float movementX;
 private float movementY;

 // Speed at which the player moves.
 public float speed = 0;

 // UI text component to display count of "PickUp" objects
collected.
 public TextMeshProUGUI countText;

 // UI object to display winning text.
 public GameObject winTextObject;

 // Start is called before the first frame update.
 void Start()
    {
```

```
 // Get and store the Rigidbody component attached to the
player.
        rb = GetComponent<Rigidbody>();

 // Initialize count to zero.
        count = 0;

 // Update the count display.
        SetCountText();

 // Initially set the win text to be inactive.
        winTextObject.SetActive(false);
    }

 // This function is called when a move input is detected.
 void OnMove(InputValue movementValue)
    {
 // Convert the input value into a Vector2 for movement.
        Vector2 movementVector = movementValue.Get<Vector2>();

 // Store the X and Y components of the movement.
        movementX = movementVector.x;
        movementY = movementVector.y;
    }

 // FixedUpdate is called once per fixed frame-rate frame.
 private void FixedUpdate()
    {
 // Create a 3D movement vector using the X and Y inputs.
        Vector3 movement = new Vector3 (movementX, 0.0f,
movementY);

 // Apply force to the Rigidbody to move the player.
        rb.AddForce(movement * speed);
    }


 void OnTriggerEnter(Collider other)
    {
 // Check if the object the player collided with has the
"PickUp" tag.
 if (other.gameObject.CompareTag("PickUp"))
        {
 // Deactivate the collided object (making it disappear).
            other.gameObject.SetActive(false);

 // Increment the count of "PickUp" objects collected.
            count = count + 1;

 // Update the count display.
```

```
            SetCountText();
        }
    }

 // Function to update the displayed count of "PickUp" objects
collected.
 void SetCountText()
    {
 // Update the count text with the current count.
        countText.text = "Count: " + count.ToString();

 // Check if the count has reached or exceeded the win
condition.
 if (count >= 12)
        {
 // Display the win text.
            winTextObject.SetActive(true);

 // Destroy the enemy GameObject.

Destroy(GameObject.FindGameObjectWithTag("Enemy"));
        }
    }

private void OnCollisionEnter(Collision collision)
{
 if (collision.gameObject.CompareTag("Enemy"))
    {
 // Destroy the current object
        Destroy(gameObject);

 // Update the winText to display "You Lose!"
        winTextObject.gameObject.SetActive(true);
        winTextObject.GetComponent<TextMeshProUGUI>().text =
"You Lose!";

    }

}


}
```

EnemyMovement.cs

```
using UnityEngine;
using UnityEngine.AI;

public class EnemyMovement : MonoBehaviour
{
```

```
 // Reference to the player's transform.
 public Transform player;

 // Reference to the NavMeshAgent component for pathfinding.
 private NavMeshAgent navMeshAgent;

 // Start is called before the first frame update.
 void Start()
    {
 // Get and store the NavMeshAgent component attached to this
object.
        navMeshAgent = GetComponent<NavMeshAgent>();
    }

 // Update is called once per frame.
 void Update()
    {
 // If there's a reference to the player...
 if (player != null)
        {
 // Set the enemy's destination to the player's current
position.
            navMeshAgent.SetDestination(player.position);
        }
    }
}
```