Μετακίνηση του παίκτη

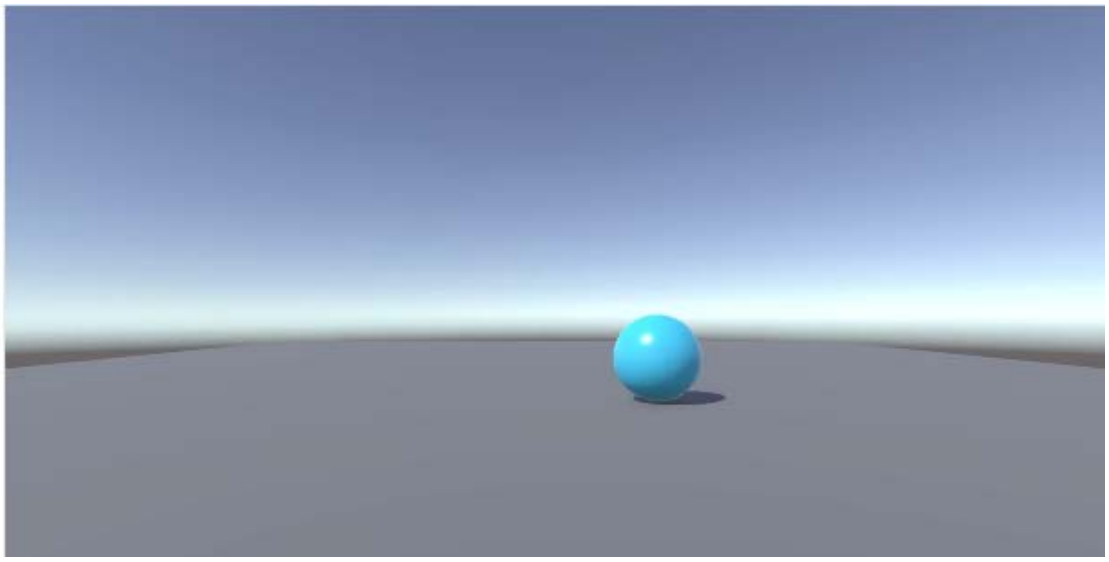
https://learn.unity.com/tutorial/66f2d394edbc2a011dded49a?uv=6&projectId=5f158f1bedbc2a0020e51f0d#

1. Επισκόπηση

Σε αυτό το σεμινάριο, θα:

- Προσθέστε ένα Rigidbody στη σφαίρα του Player, ώστε να μπορείτε να χρησιμοποιήσετε τη μηχανή φυσικής Unity
- Γράψτε το δικό σας σενάριο PlayerController σε C#, για να κάνετε τη σφαίρα να ανταποκρίνεται στην είσοδο του προγράμματος αναπαραγωγής
- Δοκιμάστε το σενάριο σας και διορθώστε τα σφάλματα

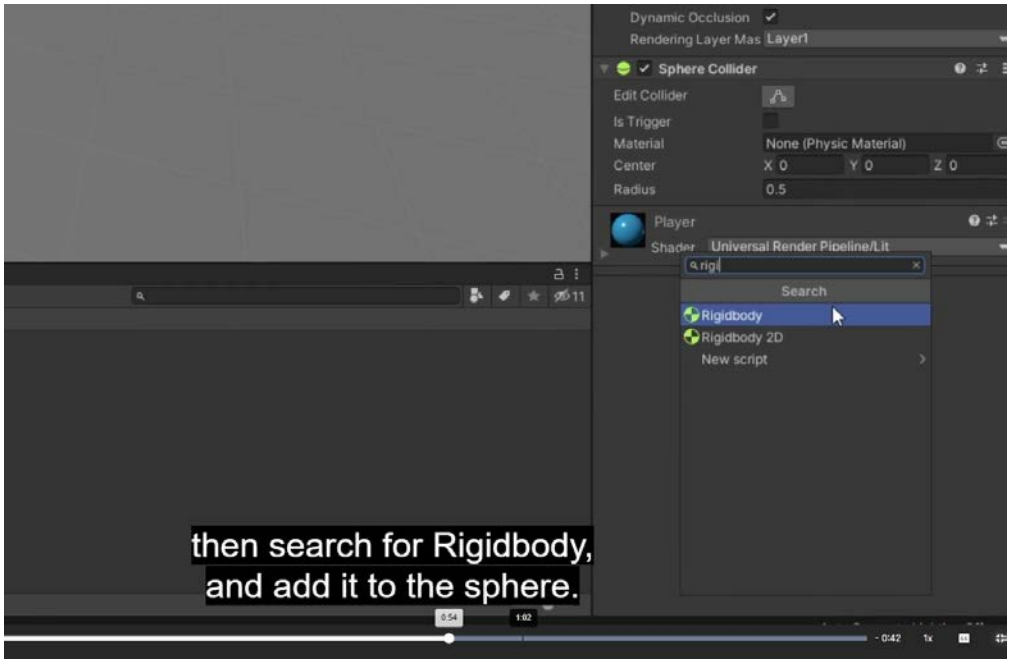
Μέχρι το τέλος αυτού του σεμιναρίου, το παιχνίδι σας θα μοιάζει κάπως έτσι:



Για να χρησιμοποιήσετε αυτό το σεμινάριο, θα χρειαστείτε ένα έργο Unity με μια σκηνή που περιέχει τη μπάλα και την επιφάνεια στην οποία κυλάει. Εάν δεν τα έχετε ήδη δημιουργήσει, ξεκινήστε με το σεμινάριο Ρύθμιση του παιχνιδιού .

2. Προσθέστε ένα Rigidbody στο Player

Ακολουθήστε τις οδηγίες βίντεο ή κειμένου παρακάτω για να προσθέσετε ένα στοιχείο **Rigidbody** στο **Player** GameObject



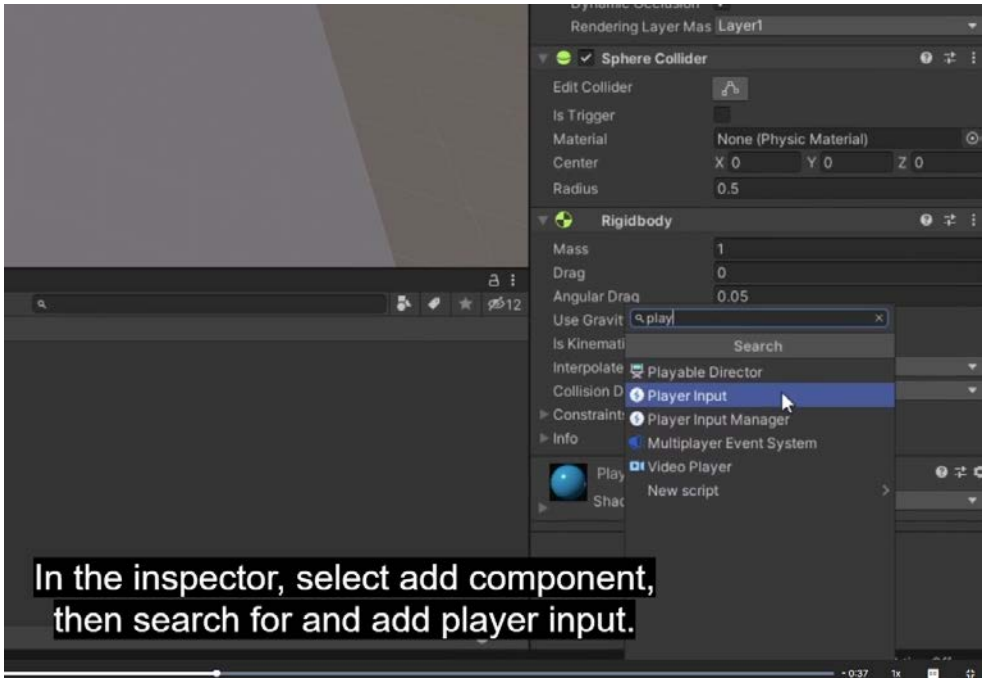
Προσθέστε ένα στοιχείο Rigidbody.

- Επιλέξτε το **Player** GameObject στο παράθυρο **Hierarchy** .
- Στο παράθυρο **Inspector** , επιλέξτε **Προσθήκη στοιχείου** , στη συνέχεια αναζητήστε "Rigidbody" και προσθέστε το στοιχείο **Rigidbody** στο **Player** GameObject.

Σημείωση : Βεβαιωθείτε ότι έχετε επιλέξει **Rigidbody** και όχι **Rigidbody 2D** .

3. Προσθέστε ένα στοιχείο Player Input

Ακολουθήστε τις παρακάτω οδηγίες βίντεο ή κειμένου για να προσθέσετε το στοιχείο **Player Input** :



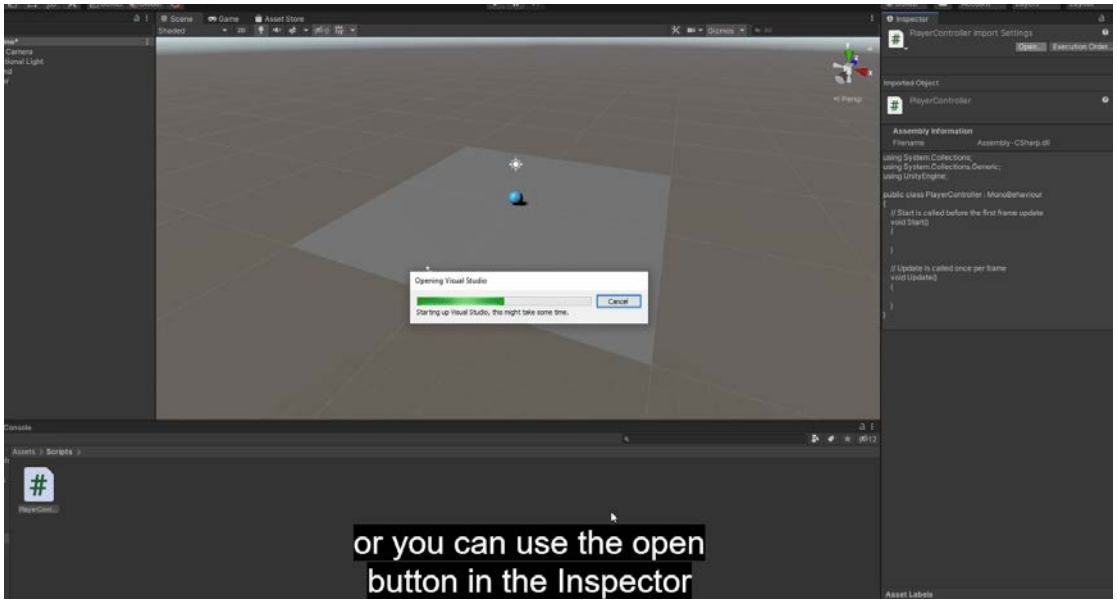
1. Προσθέστε το στοιχείο Player Input

- Επιλέξτε το **Player** GameObject στο παράθυρο **Hierarchy** .

- Στο παράθυρο **Επιθεωρητής** , προσθέστε το στοιχείο **Player Input**

4. Δημιουργήστε ένα νέο σενάριο

Ακολουθήστε τις παρακάτω οδηγίες βίντεο ή κειμένου για να δημιουργήσετε ένα νέο σενάριο **PlayerController** και να το ανοίξετε σε ένα πρόγραμμα επεξεργασίας σεναρίων:



1. Δημιουργήστε μια νέα δέσμη ενεργειών PlayerController.

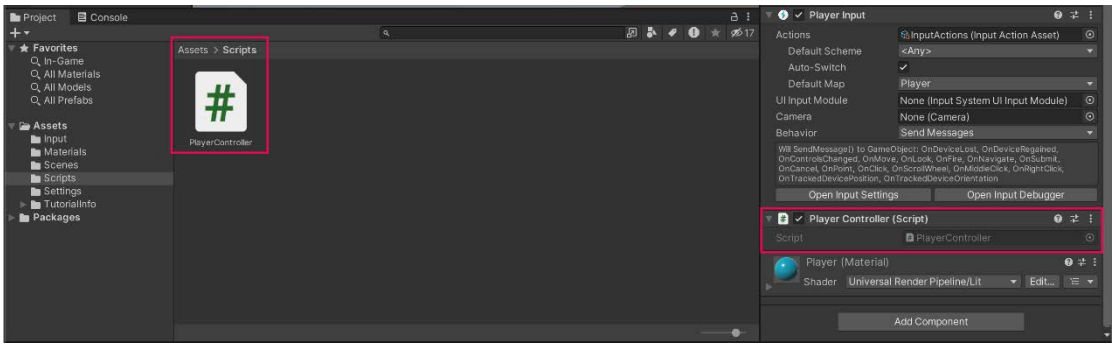
- Στο παράθυρο **Project** , δημιουργήστε έναν νέο φάκελο με το όνομα "Scripts".

- Με επιλεγμένο το **Player** GameObject, επιλέξτε **Add Component** > **New Script και,** στη συνέχεια, ονομάστε τη νέα δέσμη ενεργειών "PlayerController".

- Το στοιχείο σεναρίου που δημιουργήθηκε θα βρίσκεται στο ριζικό επίπεδο του φακέλου **Assets** από προεπιλογή. Μετακινήστε το νέο στοιχείο σεναρίου **PlayerController** στο φάκελο **Scripts** .

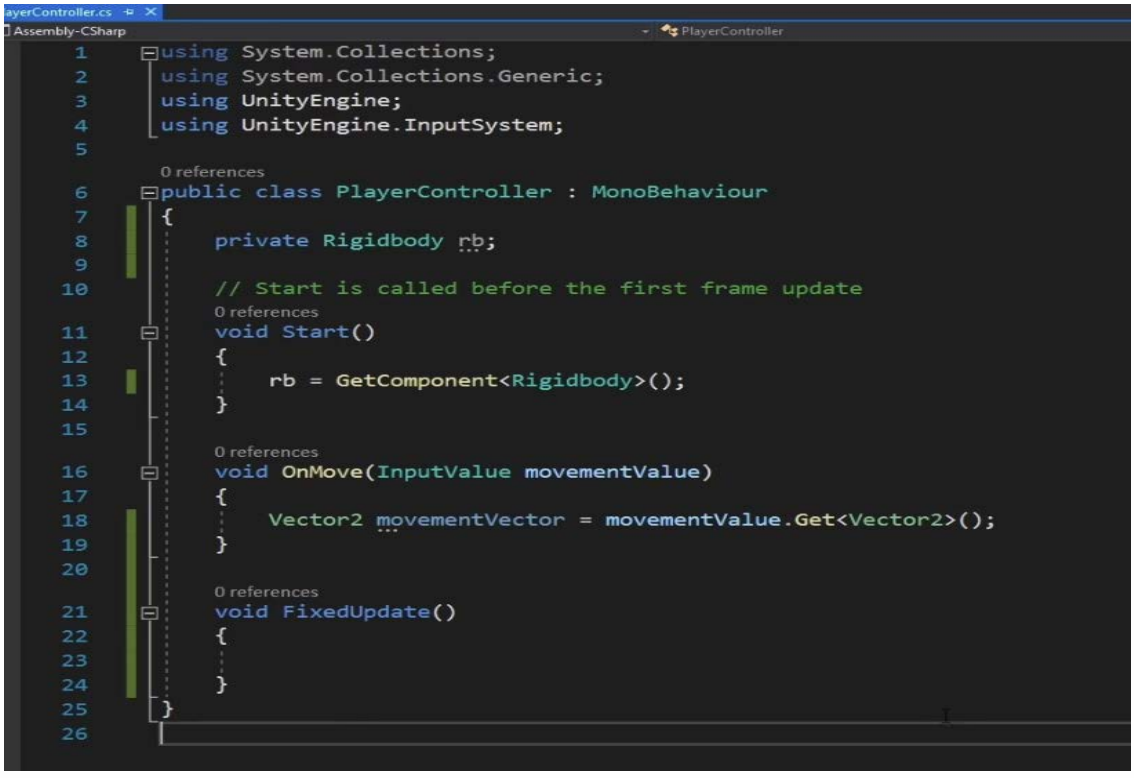
2. Ανοίξτε το σενάριο σε ένα πρόγραμμα επεξεργασίας σεναρίων.

- Κάντε διπλό κλικ στο στοιχείο σεναρίου στο παράθυρο **Project** για να το ανοίξετε στο πρόγραμμα επεξεργασίας σεναρίων που προτιμάτε, συνήθως στο Visual Studio.



5. Γράψτε τη δήλωση συνάρτησης OnMove

Ακολουθήστε τις παρακάτω οδηγίες βίντεο ή κειμένου για να διαγράψετε τη λειτουργία Ενημέρωση, να προσθέσετε τον χώρο ονομάτων InputSystem και να προσθέσετε τη συνάρτηση OnMove:



1. Διαγράψτε τη λειτουργία Update.

- Εάν δεν έχετε ήδη ανοιχτό το σενάριο **του PlayerController** , ανοίξτε το τώρα.
- Διαγράψτε ολόκληρη τη λειτουργία **Update** — δεν θα τη χρειαστείτε.

2. Προσθέστε τον χώρο ονομάτων InputSystem.

- Προσθέστε την ακόλουθη γραμμή κώδικα στην κορυφή του σεναρίου κάτω από τους υπάρχοντες χώρους ονομάτων:

```
using UnityEngine.InputSystem;
```

3. Προσθέστε τη λειτουργία OnMove.

- Κάτω από τη συνάρτηση **Start** , αλλά πριν από τον τελικό σγουρό άγκιστρο, προσθέστε μια νέα γραμμή και προσθέστε τον ακόλουθο κώδικα:

```
void OnMove (InputValue movementValue)
{


}
```

6. Εφαρμόστε δεδομένα εισόδου στον Player

Ακολουθήστε τις παρακάτω οδηγίες βίντεο ή κειμένου για να ορίσετε το διάνυσμα κίνησης, να αντιστοιχίσετε μια νέα μεταβλητή Rigidbody και να προσθέσετε μια συνάρτηση FixedUpdate:

1. Ορίστε το διάνυσμα κίνησης.

- Στο χώρο μέσα στη συνάρτηση **OnMove** , προσθέστε την ακόλουθη γραμμή κώδικα:

```
Vector2 movementVector = movementValue.Get<Vector2>();
```

2. Εκχωρήστε μια νέα μεταβλητή Rigidbody.

- Πάνω από τη συνάρτηση **Start** , προσθέστε την ακόλουθη γραμμή κώδικα για να δηλώσετε μια νέα μεταβλητή:

```
private Rigidbody rb;
```

- Στη συνέχεια, μέσα στη συνάρτηση **Start** , προσθέστε την ακόλουθη γραμμή κώδικα για να εκχωρήσετε την τιμή της μεταβλητής:

```
rb = GetComponent <Rigidbody>();
```

3. Προσθέστε μια συνάρτηση FixedUpdate.

- Δημιουργήστε μια νέα συνάρτηση που ονομάζεται **FixedUpdate** κάτω από τη συνάρτηση **Έναρξη** προσθέτοντας τον ακόλουθο κώδικα:

```
private void FixedUpdate()
{


}
```

7. Εφαρμόστε δύναμη στον Player

Ακολουθήστε τις παρακάτω οδηγίες βίντεο ή κειμένου για να προσθέσετε δύναμη στον παίκτη, να προσθέσετε μεταβλητές κίνησης x και y, να ορίσετε αυτές τις μεταβλητές, να ολοκληρώσετε την κλήση **AddForce** και να δοκιμάσετε το παιχνίδι σας

Προσθέστε δύναμη στον Player.

- Στο σώμα της συνάρτησης **FixedUpdate** , προσθέστε τον ακόλουθο κώδικα:

```
rb.AddForce(movementVector);
```

- Εάν το πρόγραμμα επεξεργασίας εμφανίσει ένα σφάλμα, μπορείτε να το αγνοήσετε προς το παρόν.

2. Προσθέστε μεταβλητές κίνησης x και y.

- Κάτω από τη μεταβλητή **Rigidbody** που δημιουργήσατε νωρίτερα, προσθέστε δύο ακόμη ιδιωτικές μεταβλητές που ονομάζονται moveX και moveY τύπου float:

```
private float movementX;

private float movementY;
```

3. Αντιστοιχίστε τις μεταβλητές κίνησης.

- Στη συνάρτηση **OnMove** , προσθέστε τις ακόλουθες δύο γραμμές κώδικα:

```
movementX = movementVector.x;

movementY = movementVector.y;
```

4. Ολοκληρώστε την κλήση AddForce.

- Στη συνάρτηση **FixedUpdate** , προσθέστε την ακόλουθη νέα γραμμή στο επάνω μέρος της συνάρτησης:

Vector3 movement = new Vector3 (movementX, 0.0f, movementY);

- Αναθεωρήστε τη δεύτερη γραμμή κώδικα στο **FixedUpdate** για να συμπεριλάβετε τη μεταβλητή κίνησης:

rb.AddForce(movement);

5. Δοκιμάστε το παιχνίδι.

- Το **Player** GameObject θα πρέπει να μετακινηθεί, αλλά αργά — θα το διορθώσετε στο επόμενο βήμα.

8. Διορθώστε την ταχύτητα κίνησης του Player

Ακολουθήστε τις οδηγίες βίντεο ή κειμένου παρακάτω για να δηλώσετε μια νέα μεταβλητή ταχύτητας, να πολλαπλασιάσετε τη δύναμη με την ταχύτητα, να αυξήσετε την ταχύτητα στο παράθυρο **Επιθεωρητής** και να δοκιμάσετε το παιχνίδι:

1. Δηλώστε μια νέα μεταβλητή ταχύτητας.

- Δημιουργήστε μια νέα δημόσια μεταβλητή float στο επάνω μέρος του σεναρίου:

public float speed = 0;

2. Πολλαπλασιάστε τη δύναμη με την ταχύτητα.

- Στη συνάρτηση **FixedUpdate** , αναθεωρήστε την κλήση **AddForce** για να συμπεριλάβετε τη μεταβλητή ταχύτητας:

rb.AddForce(movement * speed);

3. Αυξήστε την ταχύτητα στο παράθυρο Inspector.

Εφόσον η ταχύτητα είναι **0** , ο παίκτης δεν θα κινηθεί.

Στο στοιχείο **PlayerController** , ορίστε την τιμή ταχύτητας στο **10** και δοκιμάστε το παιχνίδι σας.

4. Δοκιμάστε το παιχνίδι.

Το **Player** GameObject θα πρέπει τώρα να κινείται με πιο λογική ταχύτητα.

Μπορείτε να ρυθμίσετε την ταχύτητα προς τα πάνω ή προς τα κάτω εάν θέλετε.

Τελικό σενάριο

Εάν το σενάριό σας δεν λειτουργεί όπως αναμένεται, παρακάτω είναι ένα παράδειγμα του πώς θα πρέπει να μοιάζει ο κώδικάς σας σε αυτό το σημείο. Τα σχόλια έχουν προστεθεί για να γίνει ο κώδικας πιο ευανάγνωστος.

PlayerController.cs

```
using System.Collections;
using System.Collections.Generic;
using Unity.VisualScripting;
using UnityEngine;
using UnityEngine.InputSystem;

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

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

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

 // Start is called before the first frame update.
 void Start()
    {
 // Get and store the Rigidbody component attached to the
player.
        rb = GetComponent<Rigidbody>();
    }
```

```
// 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);
    }

}
```