**Orijinal Kaynak:** http://forum.unity3d.com/threads/unity-lesson-1-draft.103421/

# Unity 3D İle Oyun Programlama

# Bölüm 6: Düşman Yapay Zekasına Giriş


**ÇEVİRİ: Süleyman Yasir KULA**

http://yasirkula.com

Ferdinand Joseph Fernandez

Chief Technological Officer, Dreamlords Digital Inc.

Admin and Co-founder, Unity Philippines Users Group


September 2011

Bu bölümde düşman yapay zekası ile uğraşacağız. Şimdilik yapay zekanın yaptığı tek şey bizi kovalamak olacak.

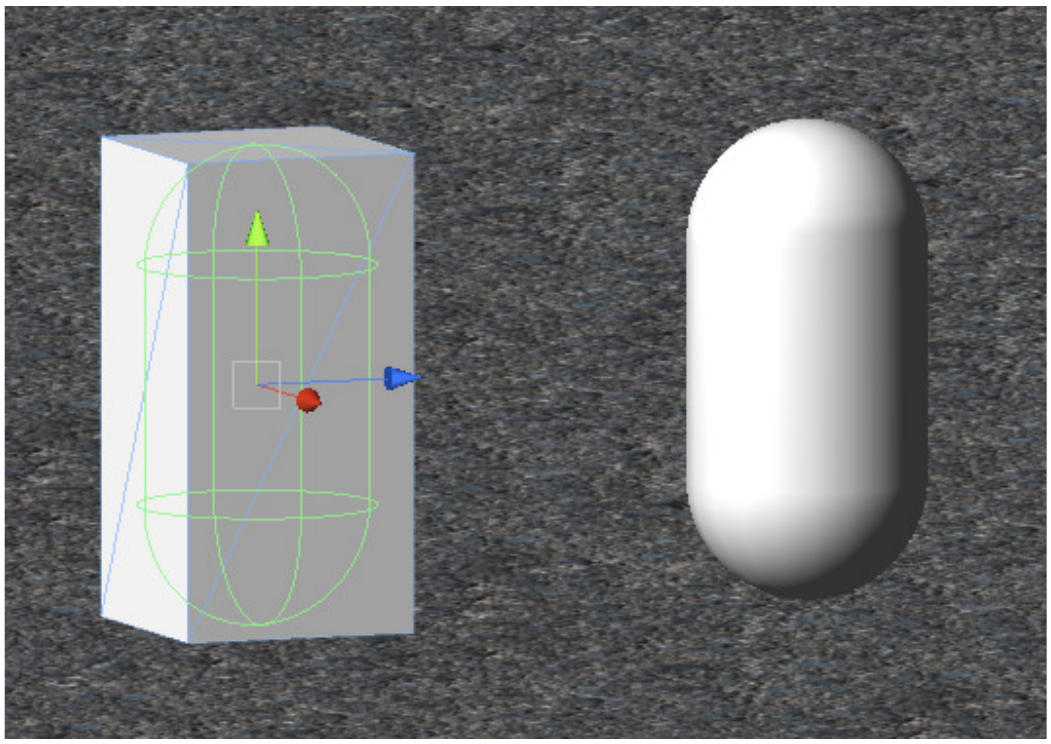
# Düşman Oluşturmak

Şu an için düşman yerine temsili küp objeleri kullanacağız.

GameObject > Create Other > Cube ile küp oluşturup (1, 2, 1) şeklinde boyutlandırın (scale).

Player'da olduğu gibi, düşmanın hareket etmesini de Character Controller ile sağlayacağız. O yüzden küp objesine bir Character Controller verin (Component > Physics > Character Controller).

Şimdi objeye “Enemy” (düşman) adını verin. (**ÇEVİRMEN EKLEMESİ:** Tıpkı Player'da yaptığımız gibi, **Enemy** objesinin **Box Collider**'ını da **Remove Component** ile silin. Zira Character Controller'ın kendi görünmez collider'ı var.)



*Görsel 6.1: Düşman objesinin ebatları Player ile aşağı yukarı aynı.*

# Scripti Yazmaya Başlamak

“EnemyMovement” diye bir C# scripti oluşturun.

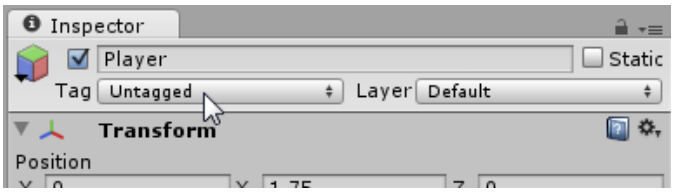
Düşman bizi takip edeceği için haliyle nerede olduğumuzu bilmelidir. Pozisyonumuzu Player'ın Transform componenti depoluyor. Bu component'e düşman objelerinden sık sık erişeceğimiz için bu componenti tutan bir değişken oluşturalım:

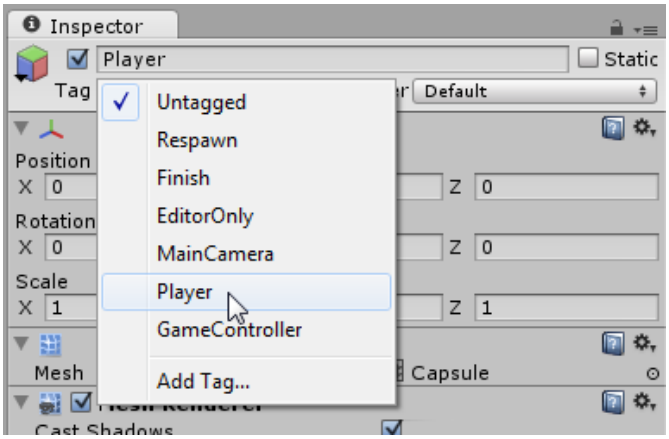
```
01 using  UnityEngine;
02 using  System.Collections;
03
04 public class  EnemyMovement : MonoBehaviour
05 {
06        Transform _player;
07
08        void  Start()
09        {
10        }
11
12        void  Update()
13        {
14        }
15 }
```

"_player” değişkeni Player objesinin Transform component'ini içinde tutacak. Bu değişkene değer vermek için kullanabileceğimiz çeşitli yollar var. Biz "tag" (etiket) sistemini kullanacağız.

# Tag Kullanımı

Hierarchy'den Player'ı seçin. Yukarıda Tag adında bir değer yer almakta. Varsayılan değeri “Untagged” olarak verilmiş.




“Untagged”e tıklayınca karşınıza seçenekler gelecek. Bunlar arasından “Player”ı seçin. Artık objemizin tag'ı "Player" oldu.

EnemyMovement script'ini şu şekilde güncelleyin:

```
01 using  UnityEngine;
02 using  System.Collections;
03
04 public class  EnemyMovement : MonoBehaviour
05 {
06        Transform _player;
07
08        void  Start()
09        {
10              GameObject playerGameObject = GameObject.FindGameObjectWithTag("Player");
11              _player = playerGameObject.transform;
12        }
13
14        void  Update()
15        {
16        }
17 }
```

Player objesini bizim için Unity'nin bulmasını istiyoruz. FindGameObjectWithTag fonksiyonu parametre olarak bir string alır ve bu tag'a sahip objeyi döndürür. Parametre olarak tahmin edeceğiniz üzere "Player" verdik. Artık bu fonksiyon Player objesini bulup playerGameObject değişkenine atacak.

Sonraki satırda playerGameObject'in, yani Player objesinin Transform component'ine erişip onu _player değişkenine veriyoruz.

# Character Controller'a Erişmek

Nasıl Player objemizin CharacterController component'ine script ile erişiyorsak düşman için de aynısını yapacağız. Bunun için EnemyMovement scriptini güncelleyelim:

```
01 using  UnityEngine;
02 using  System.Collections;
03
04 public class  EnemyMovement : MonoBehaviour
05 {
06        CharacterController _controller;
07        Transform _player;
08
09        void  Start()
10        {
11              GameObject playerGameObject = GameObject.FindGameObjectWithTag("Player");
12              _player = playerGameObject.transform;
13
14              _controller = GetComponent<CharacterController>();
15        }
16
17        void  Update()
18        {
19        }
20 }
```

# Oyuncuyu Kovalamak

Düşmanı hareket ettirmek için de Move fonksiyonunu kullanacağız. Ama bu sefer input'u klavyeden almayacağız. Bir formül kullanarak zombinin gideceği yönü kendimiz hesaplayacağız:

```
01 using UnityEngine;
02 using System.Collections;
03
04 public class EnemyMovement : MonoBehaviour
05 {
06     CharacterController _controller;
07     Transform _player;
08
09     void Start()
10     {
11         GameObject playerGameObject = GameObject.FindGameObjectWithTag("Player");
12         _player = playerGameObject.transform;
13
14         _controller = GetComponent<CharacterController>();
15     }
16
17     void Update()
18     {
19         Vector3 direction = _player.position - transform.position;
20
21         _controller.Move(direction * Time.deltaTime);
22     }
23 }
```

Öncelikle scripti test edelim. Scripti düşman objesine verin ve oyunu çalıştırın. Zombinin sizi kovalaması lazım.

Hemen hemen tüm işi 19. satırda yapıyoruz. Player ve Enemy arasındaki yön vektörünü buluyoruz. Bunu yapmanın yolu da Player'ın pozisyonundan Enemy'nin pozisyonunu çıkarmak. Böylece bu iki obje arasındaki yön vektörünü elde ediyoruz ve vektörün ucu Player'a doğru bakıyor. Eğer Enemy'nin pozisyonundan Player'ınkini çıkarsaydık okun ucu Enemy'e bakacaktı:

$$\Delta x = x_2 - x_1$$

Bir başka deyişle:

**YÖN = HEDEF KONUM - KENDİ KONUMUM**

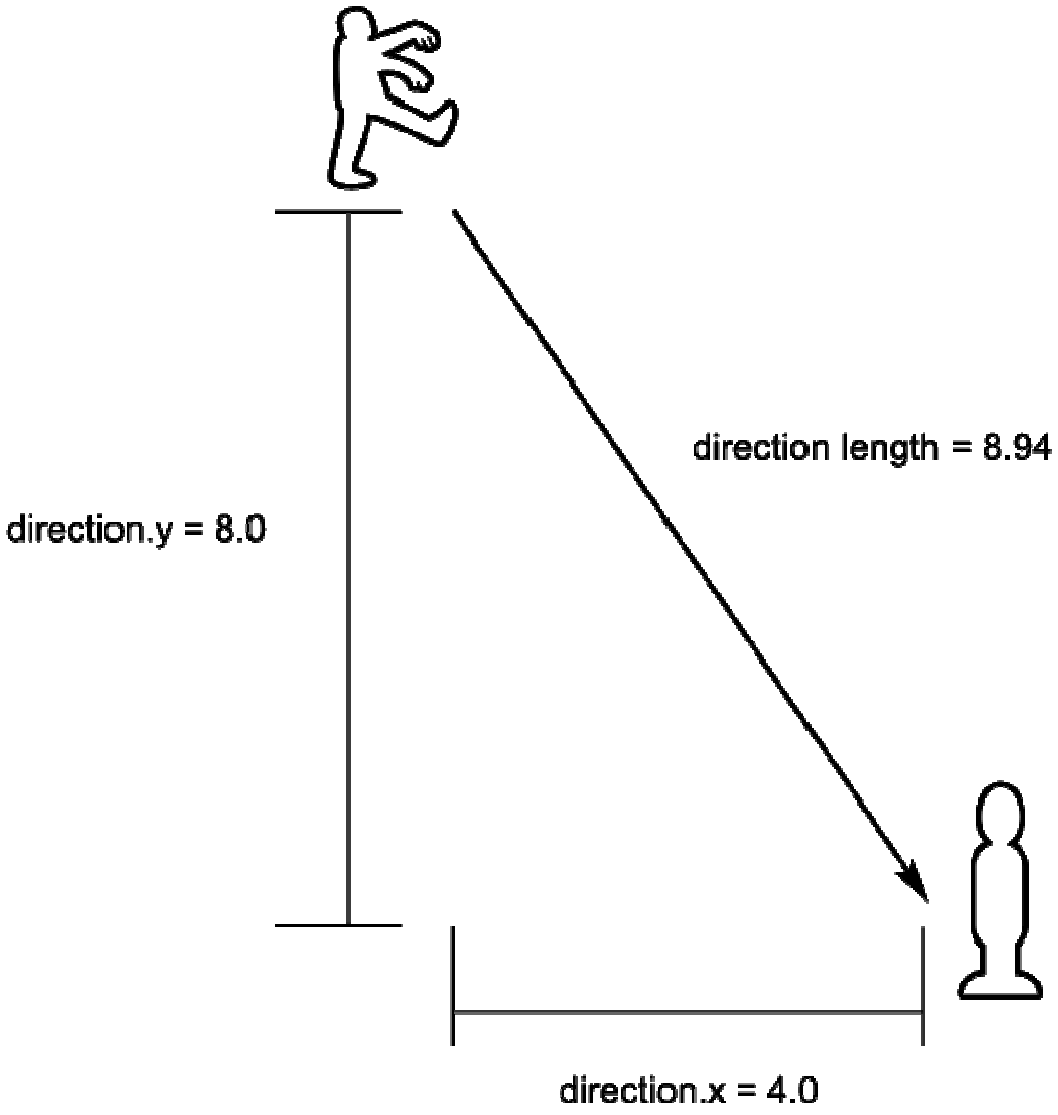
Bu formül sayesinde Player'ı kovalamak için gerekli vektörü elde ediyoruz.

19. satırda "KENDİ KONUMUM" olarak transform.position'ı, "HEDEF KONUM" olarak da _player.position'ı kullanıyoruz. Bu iki değişken birer Vector3 (yani x, y ve z değerlerine sahip). Bu yüzden elde ettiğimiz veri de Vector3 türünde oluyor.

# Vektörü Normalize Etmek

Düşman oyuncuya yaklaştıkça yavaşlıyor. Çünkü düşman ile oyuncu arasındaki yön vektörünün boyu (length) gitgide kısalıyor.

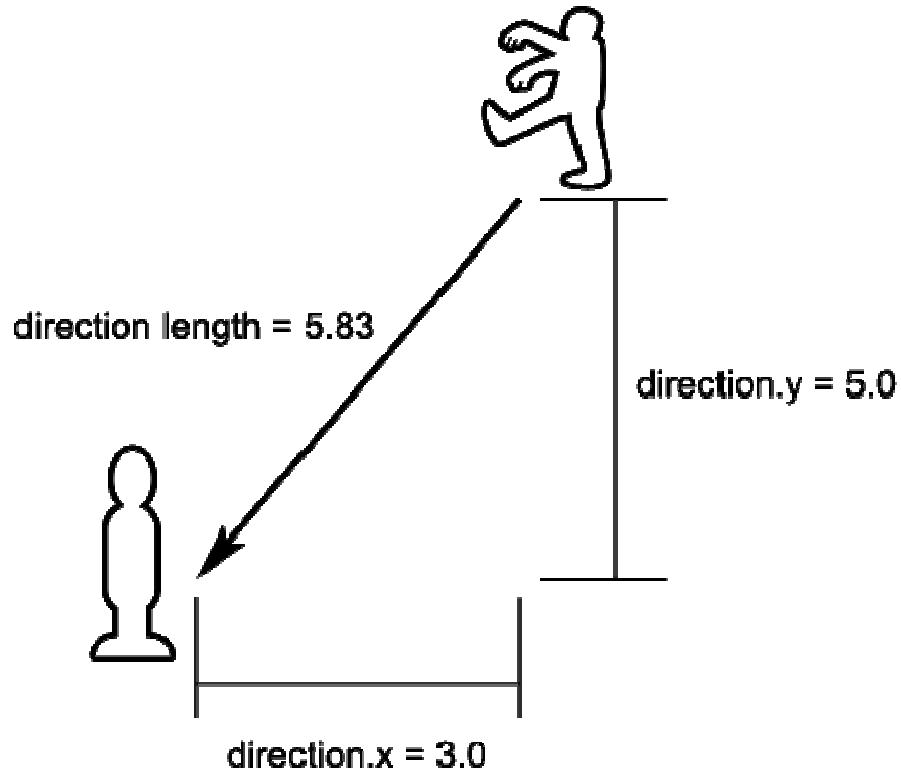
Şu örneği inceleyelim:




Yön vektörünün (direction) görselleştirilmesi

Bu örnekte düşman ile oyuncu arasındaki yön vektörünün uzunluğu (length) 8.94 birim.

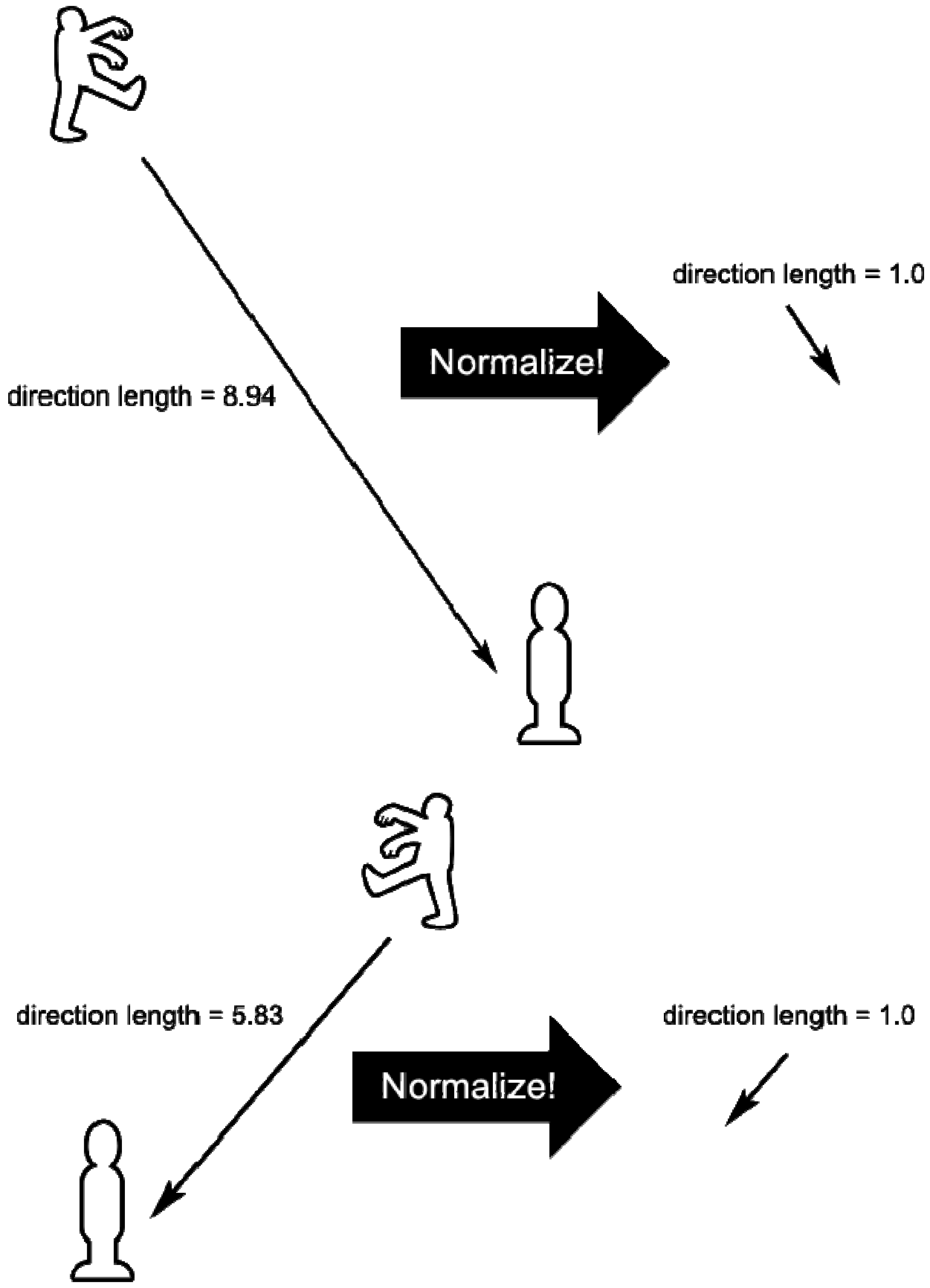
Şimdi sıradaki örneğe bakalım. Düşmanın bize daha yakın olduğu bir senaryo bu. Yön vektörü daha da kısalmış durumda. Hatırlarsanız Move fonksiyonuna parametre olarak bu yön vektörünü verdik. Yani yön vektörü kısaldıkça zombi daha yavaş hareket etmeye başlıyor.




Biz böyle olsun istemiyoruz. Zombilerimiz hep aynı hızda hareket etsinler. Bunun için de bir şekilde yön vektörünün uzunluğunu belli bir değere sabitlemeliyiz. Bu işleme Normalize deniyor.

**Bir vektörü Normalize etmek onun uzunluğunu 1.0 yapar. Ama vektörün yönü aynı kalır!**

Üstteki iki örnekteki yön vektörlerini normalize edersek ne olur birlikte görelim:


direction length = 1.0
Normalize!
direction length = 8.94
direction length = 5.83
direction length = 1.0
Normalize!

Gördüğünüz gibi Normalize edilen vektörün uzunluğu 1.0 oluyor ama yönü aynı kalıyor.

Unity'de Normalize için hazır bir fonksiyon mevcut. Biz de bu fonksiyondan faydalanalım:

```
01 using UnityEngine;
02 using System.Collections;
03
04 public class EnemyMovement : MonoBehaviour
05 {
06     CharacterController _controller;
07     Transform _player;
08
09     void Start()
10     {
11         GameObject playerGameObject = GameObject.FindGameObjectWithTag("Player");
12         _player = playerGameObject.transform;
13
14         _controller = GetComponent<CharacterController>();
15     }
16
17     void Update()
18     {
19         Vector3 direction = _player.position - transform.position;
20         direction.Normalize();
21
22         _controller.Move(direction * Time.deltaTime);
23     }
24 }
```

20. satırda Vector3 class'ının Normalize fonksiyonunu kullanıyoruz. Böylece vektörümüz Normalize ediliyor. Normalize etmenin bir başka yolu da şu:

```
20         direction = direction.normalized;
```

“**normalized**” değişkeni ise vektörün kendisini değiştirmez ama Normalize edilmiş halini döndürür. Biz de döndürülen bu değeri istediğimiz değişkene atabiliriz (üstteki kodda yine direction vektörüne atıyoruz).

Oyunu çalıştırınca zombinin yavaşladığını ama hızının sabitlendiğini göreceksiniz. Yavaşlamanın sebebi artık yön vektörünün uzunluğunun hep 1.0 olması. Yani saniyede 1 metre hareket ediyoruz. Bir hız değişkeni ekleyerek düşmanı hızlandıralım derim ben.

# Düşmana Hız Vermek

EnemyMovement'ı düzenleyelim:

```
01 using UnityEngine;
02 using System.Collections;
03
04 public class EnemyMovement : MonoBehaviour
05 {
06     CharacterController _controller;
07     Transform _player;
08
09     [SerializeField]
10     float _moveSpeed = 5.0f;
11
12     void Start()
13     {
14         GameObject playerGameObject = GameObject.FindGameObjectWithTag("Player");
15         _player = playerGameObject.transform;
16
17         _controller = GetComponent<CharacterController>();
18     }
19
20     void Update()
21     {
22         Vector3 direction = _player.position - transform.position;
23         direction.Normalize();
24
25         Vector3 velocity = direction * _moveSpeed;
26
27         _controller.Move(velocity * Time.deltaTime);
28     }
29 }
```

Player scriptinde yaptığımız işlemin aynısını yaptığımız için açıklama yapmama gerek yok. Düşmana çok hız vermeyin ki kaçma imkanınız olsun!

# Özet Geçecek Olursak...

Bu bölümde düşmanı oluşturup onu hareket ettirmekle kalmadık; aynı zamanda vektör matematiğine de basit bir giriş yaptık. Ayrıca tag (etiket) sistemi ile uğraştık.