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

Unity 3D İle Oyun Programlama

Bölüm 8: Düşmana Hasar Vermek

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

Artık bizi kovalayan zombiler var. Bu bölümde karakterin zombilere ateş etmesini sağlayacağız.

Can Eklemek

Düşmanlara hasar vermeden önce temel bir konuyu halletmeliyiz: onlara birer sağlık değeri vermeliyiz ki hasar uygulayabilelim. Sağlık bir tamsayı (integer) olacak ve örneğin ilk değeri 100 olacak. Düşman hasar aldıkça bu değer azalacak. Eğer sağlık 50'ye inerse anlayacağız ki düşman sağlığının yarısını kaybetmiş. Sağlık sıfıra inince ise düşman ölmüş olacak.

Sağlık (Health) için yeni bir script yazacağız. "Health" adında yeni bir C# scripti oluşturun:

```
01 using UnityEngine;
02 using System.Collections;
03
04 public class Health : MonoBehaviour
05 {
06     [SerializeField]
07     int _maximumHealth = 100;
08
09     int _currentHealth = 0;
10
11     void Start()
12     {
13         _currentHealth = _maximumHealth;
14     }
15 }
```

“_maximumHealth” değişkeni maksimum sağlık değerini belirliyor. Sağlığı en başta bu değere eşitliyoruz. Örneğin boss vari düşmanlar için bu değişkenin değerini 100 değil de daha yüksek bir değer yapabilir ve düşmanın daha zor ölmesini sağlayabiliriz.

“_currentHealth” ise mevcut sağlığın değerinin depolandığı değişkenimiz.

Yeni bir fonksiyon yazalım:

```
01 using UnityEngine;
02 using System.Collections;
03
04 public class Health : MonoBehaviour
05 {
06     [SerializeField]
07     int _maximumHealth = 100;
08
09     int _currentHealth = 0;
10
11     void Start()
12     {
13         _currentHealth = _maximumHealth;
14     }
15
16     public void Damage(int damageValue)
17     {
18         _currentHealth -= damageValue;
19     }
20 }
```

Damage (HasarVer) fonksiyonu sağlığı azaltmaya yarıyor. Düşmana hasar verirken bu fonksiyonu kullanacağız.

Şimdi düşmanlara Health scriptini verelim.

Project panelinden Enemy prefabını seçin. Component > Scripts > Health yolunu izleyerek Health scriptini prefaba verin. Artık Health scripti tüm Enemy klonlarına otomatik olarak eklendi.

Raycast Sistemini Kullanarak Ateş Etmek

Şimdi sıra düşmana hasar vermeye yarayan scripti yazmaya geldi! Basit bir silah scripti yazacağız. “RifleWeapon” adında yeni bir C# scripti oluşturun.

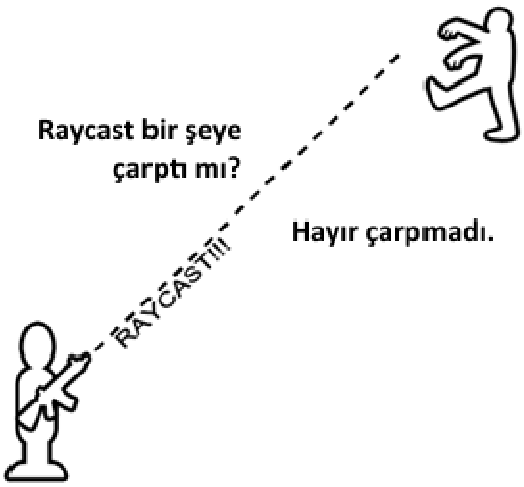
NOT: Scriptleri her zaman için "Scripts" klasöründe oluşturun ki Project paneli düzenli kalsın.

Health scriptindeki Damage fonksiyonunu çağıran script işte bu RifleWeapon scripti olacak.

Ateş ederken düşmanı vurup vurmadığımızı anlamak için “raycasting” denen bir teknolojiyi kullanacağız.

Raycasting sistemi, 3 boyutlu uzayda görünmez bir çizgi oluşturup bu çizginin bir objeyle temas edip etmediğine bakmaya yarar.

Bizim çizgimiz karakterin üzerinden başlayacak ve ileri yönde olacak. Bu çizginin bir düşmana çarpıp çarpmadığına bakacağız. Eğer çarpmışsa o düşmanın Health scriptindeki Damage fonksiyonunu çağıracağız.



Aslına bakacak olursanız 3D uzayda kurşun objeleri oluşturmayacağız. Raycast ile zaten o kurşunun nereye isabet edeceğini aşağı-yukarı hesaplıyoruz. Raycast işlemi anlık gerçekleşir, yani eğer hedefinizde bir düşman varsa düşman anında hasar alır. Video oyun terminolojisinde buna "*hitscan*" denir (bkz. http://en.wikipedia.org/wiki/Hitscan)

Adım adım ilerleyelim. Önce RifleWeapon'u şu şekilde güncelleyin:

```
01 using UnityEngine;
02 using System.Collections;
03
04 public class RifleWeapon : MonoBehaviour
05 {
06     void Update()
07     {
08         Ray mouseRay = Camera.main.ViewportPointToRay(new Vector3(0.5f, 0.5f, 0));
09         RaycastHit hitInfo;
10
11         if (Physics.Raycast(mouseRay, out hitInfo))
12         {
13             Debug.Log(hitInfo.transform.name);
14         }
15     }
16 }
```

Bu kod, ekranın ortasından dümdüz ileri doğru bir raycast oluşturur ve eğer raycast ışını bir objeye temas ederse onun ismini konsola yazdırır.

8. satırda Ray (ışın) türünde bir değişken oluşturuyoruz. Bu ışının başlangıç noktasını ekranın tam ortası (kameranın verdiği görüntünün tam ortası) olarak ayarlıyoruz. ViewportPointToRay fonksiyonu sadece bir Vector3 parametre alır. Bu parametrenin X değeri 0 olursa ışın ekranın solundan, 1 olursa sağından çıkar. Y değeri 0 olursa ışın ekranın alt kenarından, 1 olursa üst kenarından çıkar. Eğer iki değer de 0.5 olursa ışın ekranın ortasından çıkar. Vector3'ün Z değeri ise ışının başlangıç noktasının kameradan kaç birim uzakta olacağını belirler. 0 verdiğimiz için ışın tam kameranın olduğu noktadan çıkar. -1 yazsaydık ışının başlangıç noktası kameranın 1 metre berisi, +1 yazsaydık da 1 metre ilerisi olurdu.

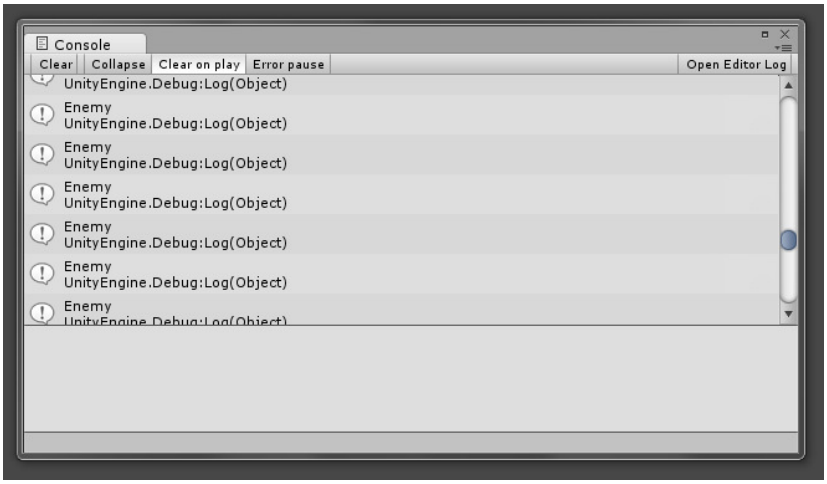
Raycast işlemi sonucu ışınımız bir objeyle temas ederse o objenin bilgilerini depolamak üzere bir değişken kullanılır ve bu değişkenin türü RaycastHit'tir. Kodun 9. satırında bu türde geçici bir değişken oluşturuyoruz.

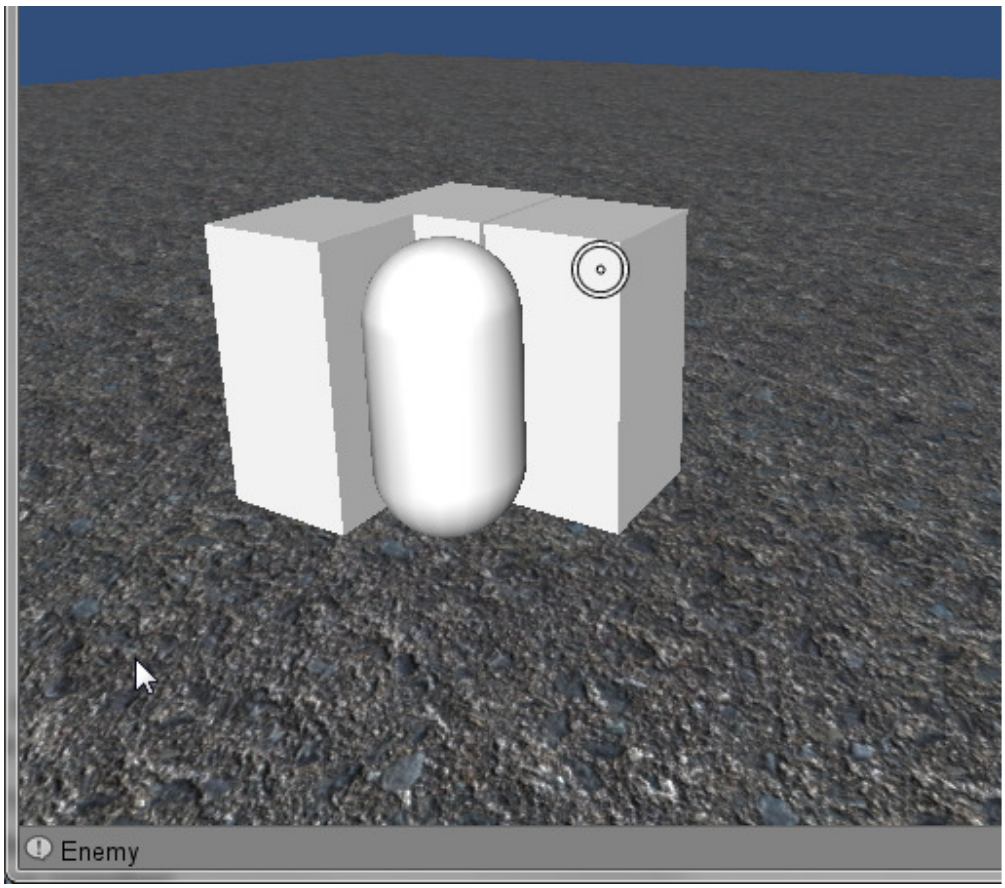
11. satır asıl işin yapıldığı satır. Burada Raycast fonksiyonunu çağırıyoruz. Eğer ışınımız bir objeye temas ederse Raycast fonksiyonu true döndürür yoksa false döndürür. Yani Raycast fonksiyonunu bir if'in içine koyarsak o if koşulu ancak ışın bir objeye temas edince gerçekleşir. İkinci parametrenin başında "**out**" diye bir anahtar kelime var. Bu kelime C#'ta kullanılır ve girilen parametrenin değerini fonksiyonun değiştirebileceğini belirler. Daha fazla bilgi için bkz. http://msdn.microsoft.com/en-us/library/ee332485.aspx

Eğer raycast ışını bir objeye temas ederse 13. satırdaki kod çalışıyor ve Debug.Log fonksiyonu ile konsola, ışının temas ettiği objenin ismi yazılıyor. Debug.Log fonksiyonu scriptlerinizde hata ayıklama yaparken çok kullanışlıdır. Burada kullandığımız hitInfo.transform komutu temas edilen objenin Transform component'ine erişmeye ve hitInfo.transform.name ise bu objenin ismine erişmeye yarar.

Konsola verilen output'ları görmek için Window > Console yaparak konsol penceresini açın.

Şimdi scripti Player objesine verin ve oyunu çalıştırın. Konsoldaki mesajları dikkatlice kontrol edin. İmlecinizin ucunda hangi obje varsa onun ismi konsolda yazmalı:





Görsel 8.1: En son gelen konsol output'unu ayrıca Unity'nin sol altındaki ufak bölmede de görebilirsiniz.

Raycast kodunu Update'in içine yazdığımız için her karede (frame) raycast işlemi yapılıyor. Bu hem gereksizdir hem de sistemi boş yere yormak demektir. Raycast işlemini sadece sol mouse tuşuna basınca, yani ateş edince yapalım. Bunun için scripti şu şekilde güncelleyin:

```
01 using UnityEngine;
02 using System.Collections;
03
04 public class RifleWeapon : MonoBehaviour
05 {
06     void Update()
07     {
08         if (Input.GetButtonDown("Fire1"))
09         {
10             Ray mouseRay = Camera.main.ViewportPointToRay(new Vector3(0.5f, 0.5f, 0));
11             RaycastHit hitInfo;
12
13             if (Physics.Raycast(mouseRay, out hitInfo))
14             {
15                 Debug.Log(hitInfo.transform.name);
16             }
17         }
18     }
19 }
```

Bir if koşulu oluşturup içinde GetButtonDown fonksiyonunu kullandık. GetButtonDown fonksiyonuna parametre olarak “Fire1” verirseniz sol mouse tuşuna tıklayınca fonksiyon true, öbür durumlarda false döndürür. **Böylece raycast işleminin sadece ateş edince gerçekleşmesini sağladık.**

ÇEVİRMEN EKLEMESİ: GetButtonDown("Fire1") komutu sadece **sol mouse tuşuna bastığınız ilk frame'de** true döndürür. Yani diyelim ki sol mouse tuşuna basılı tutuyorsunuz. GetButtonDown sadece tuşa basmaya başladığınız anda true döndürür. Ardından basılı tutsanız bile false döndürür. Ta ki tuştan elinizi çekip tekrar basana kadar.

Hasar Vermek

Raycast yapabildiğimize göre düşmana hasar verme kısmını artık halletmeye hazırız:

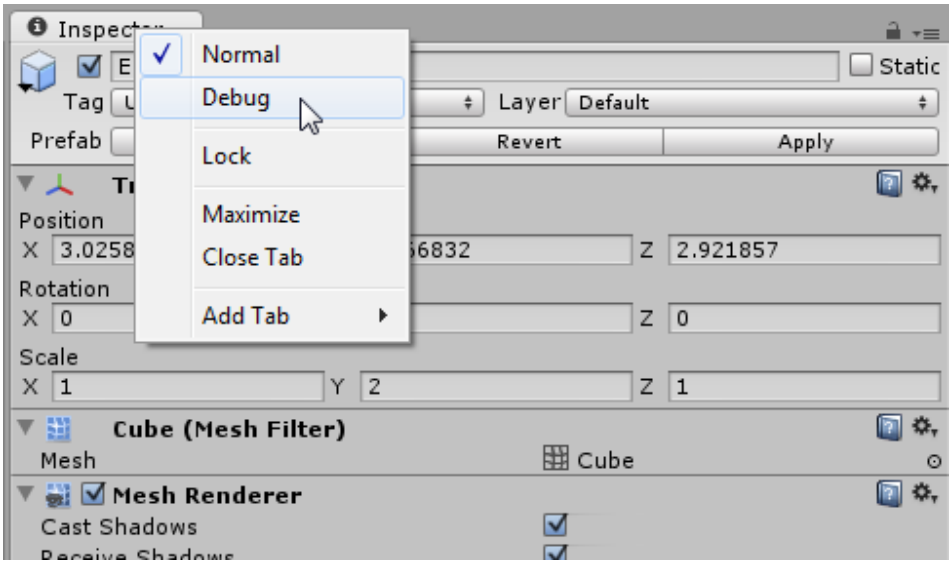
```
01 using UnityEngine;
02 using System.Collections;
03
04 public class RifleWeapon : MonoBehaviour
05 {
06     void Update()
07     {
08         if (Input.GetButtonDown("Fire1"))
09         {
10             Ray mouseRay = Camera.main.ViewportPointToRay(new Vector3(0.5f, 0.5f, 0));
11             RaycastHit hitInfo;
12
13             if (Physics.Raycast(mouseRay, out hitInfo))
14             {
15                 Health enemyHealth = hitInfo.transform.GetComponent<Health>();
16                 if (enemyHealth != null)
17                 {
18                     enemyHealth.Damage(50);
19                 }
20             }
21         }
22     }
23 }
```

Artık raycast'in çarptığı objenin ismini print etmek yerine o objenin Health component'ine erişiyoruz. Eğer objenin Health componenti yoksa (raycast mesela zemine çarptıysa) "*enemyHealth != null*" ifadesi false döndürür ve if'in içine girilmez. Ama eğer Health componenti varsa o zaman if'in içine girilir ve Health scriptindeki Damage fonksiyonu çağrılarak objeye 50 hasar verilir.

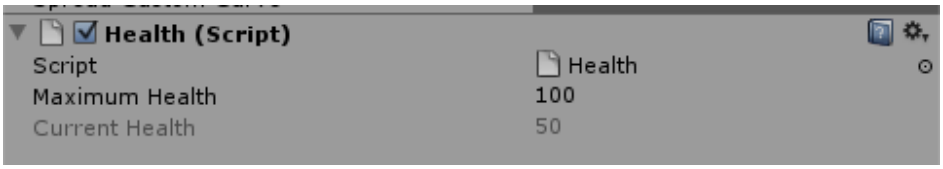
Bu koddaki sıkıntı verilen hasarın sabit (50) olması. Bu değeri bir değişken vasıtasıyla değiştirebilirsek daha güzel olur:

```
01 using UnityEngine;
02 using System.Collections;
03
04 public class RifleWeapon : MonoBehaviour
05 {
06     [SerializeField]
07     int _damageDealt = 50;
08
09     void Update()
10     {
11         if (Input.GetButtonDown("Fire1"))
12         {
13             Ray mouseRay = Camera.main.ViewportPointToRay(new Vector3(0.5f, 0.5f, 0));
14             RaycastHit hitInfo;
15
16             if (Physics.Raycast(mouseRay, out hitInfo))
17             {
18                 Health enemyHealth = hitInfo.transform.GetComponent<Health>();
19                 if (enemyHealth != null)
20                 {
21                     enemyHealth.Damage(_damageDealt);
22                 }
23             }
24         }
25     }
26 }
```

Scripti henüz test etmek zor çünkü düşmanların ölmesini daha ayarlamadık. Şu anda test etmenin tek yolu bir düşmana birkaç kez ateş etmek ve ardından o düşmanı seçip Inspector'u da Debug moduna alarak düşmanın sağlığını (Current Health) kontrol etmek. Inspector'a sağ tıklayıp Debug deyin:



Artık Inspector private değişkenlerin değerlerini de gösteriyor ama bu değerler koyu gri renkte gözüküyor ve elle değiştirilemiyor.



Şimdi oyunu çalıştırıp bir zombiye ateş edin ve zombinin canının azalıp azalmadığını kontrol edin. Azalmadıysa bir sıkıntı var demektir, bu bölümü baştan okuyup bir detay atlamadığınızdan emin olun.

Ölünce Olacakları Ayarlamak

Bir düşman ölünce bizi takip etmeyi bırakıp yere yığılmalı. Ama henüz yere yığılacak bir karakter modelimiz yok; o yüzden şu safhada ölen düşmanları direkt sahneden sileceğiz.

Bunun için ise Health scriptinde ilgili değişiklikleri yapalım:

```
01 using UnityEngine;
02 using System.Collections;
03
04 public class Health : MonoBehaviour
05 {
06     [SerializeField]
07     int _maximumHealth = 100;
08
09     int _currentHealth = 0;
10
11     void Start()
12     {
13         _currentHealth = _maximumHealth;
14     }
15
16     public void Damage(int damageValue)
17     {
18         _currentHealth -= damageValue;
19
20         if (_currentHealth <= 0)
21         {
22             Destroy(gameObject);
23         }
24     }
25 }
```

Artık sağlık sıfıra ulaştığı vakit objeyi siliyoruz. Silme işlemini Destroy fonksiyonu ile yapıyoruz.

Oyunu çalıştırın ve bir zombiye saldırın. Yeterince vurduğunuzda zombi puf diye yok olacak. İleride zombi öldüğünde yok olmak yerine yere yığılacak; şimdilik bununla idare edin.

Fare İmlecinin Görünmesini Engellemek

Ekranın ortasında bir nişangahımız var, fare imlecinin ortada dolaşıp kafamızı karıştırmasına gerek yok. Fare imlecinin hareket etmesini engellemek ve onu görünmez kılmak için bir kod bulunmakta.

RifleWeapon scriptini güncelleyelim:

```
01 using UnityEngine;
02 using System.Collections;
03
04 public class RifleWeapon : MonoBehaviour
05 {
06     [SerializeField]
07     int _damageDealt = 10;
08
09     void Start()
10     {
11         Screen.lockCursor = true;
12     }
13
14     void Update()
15     {
16         if (Input.GetButtonDown("Fire1"))
17         {
18             Ray mouseRay = Camera.main.ViewportPointToRay(new Vector3(0.5f, 0.5f, 0));
19             RaycastHit hitInfo;
20
21             if (Physics.Raycast(mouseRay, out hitInfo))
22             {
23                 Health enemyHealth = hitInfo.transform.GetComponent<Health>();
24                 if (enemyHealth != null)
25                 {
26                     enemyHealth.Damage(_damageDealt);
27                 }
28             }
29         }
30     }
31 }
32
```

Screen.lockCursor bir boolean'dır ve true yaparsanız imleç hem görünmez olur hem de hareket etmeyi keser.

Oyunu test ederseniz bunun işe yaradığını ama imlecin bir süre sonra tekrar belirdiğini göreceksiniz. Bu sorun sadece Unity editöründe böyle. Oyunu build ederseniz bu sorunu yaşamazsınız.

Escape tuşuna basınca bir menü çıktığını varsayalım. Bu durumda imleci geri görünür kılmamız gerekir. Bunu başarmak için şu düzenlemeyi yapın:

```
01 using UnityEngine;
02 using System.Collections;
03
04 public class RifleWeapon : MonoBehaviour
05 {
06     [SerializeField]
07     int _damageDealt = 10;
08
09     void Start()
10     {
11         Screen.lockCursor = true;
12     }
13
14     void Update()
15     {
16         if (Input.GetKey(KeyCode.Escape))
17         {
18             Screen.lockCursor = false;
19         }
20
21         if (Input.GetButtonDown("Fire1"))
22         {
23             Screen.lockCursor = true;
24             Ray mouseRay = Camera.main.ViewportPointToRay(new Vector3(0.5f, 0.5f, 0));
25             RaycastHit hitInfo;
26
27             if (Physics.Raycast(mouseRay, out hitInfo))
28             {
29                 Health enemyHealth = hitInfo.transform.GetComponent<Health>();
30                 if (enemyHealth != null)
31                 {
32                     enemyHealth.Damage(_damageDealt);
33                 }
34             }
35         }
36     }
37 }
38
```

Artık ESC tuşuna basınca imleç belirecek, sol mouse tuşuna basıp ateş ettiğimizde tekrar kaybolacak.

Özet Geçecek Olursak...

Bu bölümde hatırı sayılır derecede farklı şeyler gördük: raycast sistemi, objeyi yok etmek, konsola output vermek ve imleci gizlemek.