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

Unity 3D İle Oyun Programlama

Bölüm 13: Oyunu Kazanmak

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 oyuna bir amaç vereceğiz; kendimizi sadece zombi öldürmekle sınırlamayacağız.

Bir kurtarma helikopteri diyelim ki 3 dakika sonra bulunduğumuz bölgede olacak. Bu süre zarfında bize saldıran zombileri püskürtmek zorundayız. Helikoptere bindiğimiz zaman ise oyunu kazanmış olacağız.

İlk Rötuşlar

Önce basit bir script ile başlayalım. Oyun başladıktan 3 dakika sonra oyunu direkt kazanmış olalım; henüz helikopterle uğraşmayalım.

"GameManager" diye bir C# scripti oluşturun:

```
01 using  UnityEngine;
02 using  System.Collections;
03
04 public class  GameManager : MonoBehaviour
05 {
06       [SerializeField]
07       float  _secondsToWin = 180.0f;
08
09       void  Start()
10       {
11       }
12 }
13
```

_secondsToWin değişkeni oyun başladıktan kaç saniye sonra oyunu kazanacağımızı depolayan değişkendir.

Şimdi Unity'nin Invoke fonksiyonunu kullanacağız. Invoke, scriptte yer alan bir fonksiyonu belli bir miktar saniye sonra çalıştırmaya (yani geciktirmeli çalıştırmaya) yarar.

```
01 using UnityEngine;
02 using  System.Collections;
03
04 public class  GameManager : MonoBehaviour
05 {
06       [SerializeField]
07       float  _secondsToWin = 180.0f;
08
09       void  Start()
10       {
11             Invoke("CheckWin", _secondsToWin);
12             Debug.Log("Time started: "  + Time.time);
13       }
14
15       void  CheckWin()
16       {
17             Debug.Log("Its "  + Time.time + ". Checking winning conditions now...");
18       }
19 }
20
```

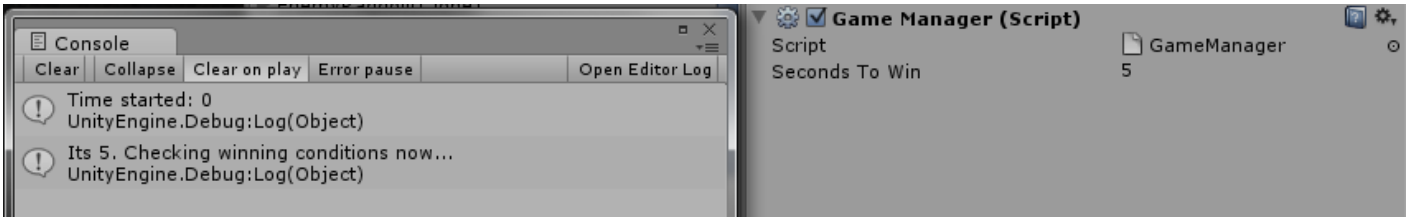
Start'ın içinde Invoke fonksiyonunu kullanıyoruz. İlk parametreye geciktirmeli çalıştırmak istediğimiz fonksiyonun ismini, ikinci parametreye de bu fonksiyonu kaç saniye sonra çalıştırmak istediğimizi giriyoruz.

Invoke'a girdiğiniz fonksiyonun public olmasına gerek yok. Örneğin bizim örneğimizdeki CheckWin fonksiyonu bir private fonksiyon.

Kodu test etme zamanı. Yeni bir empty gameobject oluşturun ve ismini “GameManager” olarak değiştirin. Ardından GameManager scriptini bu objeye verin.

“Seconds To Win” değişkeninin değerini Inspector'dan 5 saniyeye düşürün. Test ederken gidip de 3 dakika boş boş beklemeyelim. Sonra bu değeri tekrar 180 yapabilirsiniz.

Oyunu çalıştırın. 5 saniye sonra konsolda bir mesaj belirecek:



Konsola Debug.Log fonksiyonu ile test amaçlı yazılar yazdırdık. Artık onları gerçek bir kodla değiştirebiliriz:

```
01 using UnityEngine;
02 using System.Collections;
03
04 public class GameManager : MonoBehaviour
05 {
06     [SerializeField]
07     float _secondsToWin = 180.0f;
08
09     static bool _hasPlayerWon = false;
10
11     void Start()
12     {
13         Invoke("CheckWin", _secondsToWin);
14     }
15
16     void CheckWin()
17     {
18         _hasPlayerWon = true;
19     }
20 }
21
```

Oyunu kazanıp kazanmadığımızı depolayan static bir değişken oluşturduk: _hasPlayerWon. Bu değişken static olduğu için ona rahatça erişebileceğiz.

Diğer scriptlerden bu scriptteki private bir değişkene ulaşamayız. O halde private olan _hasPlayerWon'a diğer scriptlerden erişebilmek için public bir "**property**" oluşturalım (property ile ilgili detaylı bilgi için bkz: http://unity3d.com/learn/tutorials/modules/intermediate/scripting/properties):

```
01 using UnityEngine;
02 using System.Collections;
03
04 public class GameManager : MonoBehaviour
05 {
06     [SerializeField]
07     float _secondsToWin = 180.0f;
08
09     static bool _hasPlayerWon = false;
10
11     public static bool HasPlayerWon { get{ return _hasPlayerWon; } }
12
13     void Start()
14     {
15         Invoke("CheckWin", _secondsToWin);
16     }
17
18     void CheckWin()
19     {
20         _hasPlayerWon = true;
21     }
22 }
23
```

Oyun Bitince Zombilerin Spawn Olmasını Durdurmak

EnemySpawnManager scriptini düzenleyin:

```
01 using UnityEngine;
02 using System.Collections;
03
04 public class EnemySpawnManager : MonoBehaviour
05 {
06     static int _livingZombies = 0;
07     static public void OnEnemyDeath()
08     {
09         --_livingZombies;
10     }
11
12     [SerializeField]
13     GameObject _enemyToSpawn;
14
15     [SerializeField]
16     float _spawnDelay = 1.0f;
17
18     [SerializeField]
19     int _enemyLimit = 30;
20
21     float _nextSpawnTime = -1.0f;
22
23     [SerializeField]
24     LayerMask _spawnLayer;
25
26     void Update()
```

```
27      {
28          if (GameManager.HasPlayerWon)
29          {
30              Destroy(this);
31              return;
32          }
33
34          if (Time.time >= _nextSpawnTime && _livingZombies < _enemyLimit)
35          {
36              Vector3 edgeOfScreen = new Vector3(1.25f, Random.value, 8.0f);
37              if (Random.value > 0.5f)
38              {
39                  edgeOfScreen.x = -0.25f;
40              }
41
42              Ray ray = Camera.main.ViewportPointToRay(edgeOfScreen);
43              RaycastHit hit;
44              if (Physics.Raycast(ray, out hit, Mathf.Infinity, _spawnLayer.value))
45              {
46                  Vector3 placeToSpawn = hit.point;
47                  Quaternion directionToFace = Quaternion.identity;
48                  Instantiate(_enemyToSpawn, placeToSpawn, directionToFace);
49                  _nextSpawnTime = Time.time + _spawnDelay;
50                  ++_livingZombies;
51              }
52          }
53      }
54 }
55
```

Oyunu kazandığımızda Destroy fonksiyonu ile EnemySpawnManager scriptini siliyoruz. Böylece yeni zombilerin spawn olmasının önüne geçiyoruz.

Ekranda Oyun Bitti Mesajı Göstermek

Oyun kazanıldığı zaman oyuncuya bunu ekrandaki bir mesaj ile bildirelim, tıpkı oyunu kaybedince ekranda game over yazısı yazdırdığımız gibi...

CombatGui scriptini açıp güncelleyin:

```
01 using UnityEngine;
02 using System.Collections;
03
04 public class CombatGui : MonoBehaviour
05 {
06      Health _playerHealth;
07
08      [SerializeField]
09      Texture2D _gameOverImage;
10
11      [SerializeField]
12      Texture2D _winImage;
13
14      void Start()
15      {
```

```
16          GameObject playerGameObject = GameObject.FindGameObjectWithTag("Player");
17          _playerHealth = playerGameObject.GetComponent<Health>();
18      }
19
20      void OnGUI()
21      {
22          if (_playerHealth.IsDead)
23          {
24              float x = (Screen.width - _gameOverImage.width) / 2;
25              float y = (Screen.height - _gameOverImage.height) / 2;
26              GUI.DrawTexture(new Rect(x, y, _gameOverImage.width, _gameOverImage.height),
27                                      _gameOverImage);
28          }
29          else if (GameManager.HasPlayerWon)
30          {
31              float x = (Screen.width - _winImage.width) / 2;
32              float y = (Screen.height - _winImage.height) / 2;
33              GUI.DrawTexture(new Rect(x, y, _winImage.width, _winImage.height), _winImage);
34          }
35      }
36 }
37
```

Game over yazısını ekrana yazdırırken kullandığımız kodun çok benzerini kullandık. Anlatacak fazladan birşey yok.

Winrar arşivinin olduğu yerdeki "Gui" klasörünün içinde "WinScreen.png" adında bir resim dosyası var. Onu projenizdeki "Gui" klasörüne import edin. Ardından dosyayı Unity'de seçin ve Texture Type'ı “GUI (Editor \ Legacy)” olarak değiştirin (Apply butonuna basmayı unutmayın).

Şimdi CombatGui objesindeki "Win Image"a değer olarak "WinScreen"i verin ve oyunu çalıştırın:

Belki henüz dikkat etmediniz ama aslında GameManager scriptimizde ufak bir bug var: eğer oyuncu _secondsToWin kadar saniye geçmeden önce ölürse yine de CheckWin fonksiyonu çalıştırıldığı zaman sanki oyunu kazanmışız gibi _hasPlayerWon'un değeri true oluyor.

CheckWin'de önce player'ın hayatta olup olmadığını kontrol etmeliyiz:

```
01 using UnityEngine;
02 using System.Collections;
03
04 public class GameManager : MonoBehaviour
05 {
06     [SerializeField]
07     float _secondsToWin = 180.0f;
08
09     static bool _hasPlayerWon = false;
10
11     public static bool HasPlayerWon { get{ return _hasPlayerWon; } }
12
13     Health _playerHealth;
14
15     void Start()
16     {
17         Invoke("CheckWin", _secondsToWin);
18
19         GameObject playerGameObject = GameObject.FindGameObjectWithTag("Player");
20         _playerHealth = playerGameObject.GetComponent<Health>();
21     }
22
23     void CheckWin()
24     {
25         if (!_playerHealth.IsDead)
26         {
27             _hasPlayerWon = true;
28         }
29     }
30 }
31
```

Player'ın Health component'ini _playerHealth değişkeninde tutuyoruz ve CheckWin fonksiyonu çağrılınca önce player'ın yaşayıp yaşamadığını kontrol ediyoruz. Sadece eğer player hayattaysa _hasPlayerWon'un değerini true yapıyoruz.

Skor Yapmak

Öldürdüğümüz zombi sayısını oyun bitince ekranda göstermeye ne dersiniz?

Ölen zombi sayısını tutan bir script gerekli bize. Bunun için “PlayerStats” adında yeni bir C# scripti oluşturun. **Scripti Player objesine verin** ve ardından kodu şöyle düzenleyin:

```
01 using UnityEngine;
02 using System.Collections;
03
04 public class PlayerStats : MonoBehaviour
05 {
06     int _zombiesKilled = 0;
07     public int ZombiesKilled { get{ return _zombiesKilled; } set { _zombiesKilled = value; } }
08 }
09
```

Her zombi öldürüşümüzde _zombiesKilled'in değerini 1 artırmalıyız. Bunu yapmak için Health scriptini güncelleyelim:

```
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     override public string ToString()
12     {
13         return _currentHealth + " / " + _maximumHealth;
14     }
15
16     public bool IsDead { get{ return _currentHealth <= 0; } }
17
18     Renderer _renderer;
19
20     PlayerStats _playerStats;
21
22     void Start()
23     {
24         _renderer = GetComponentInChildren<Renderer>();
25         _currentHealth = _maximumHealth;
26
27         GameObject player = GameObject.FindGameObjectWithTag("Player");
28         _playerStats = player.GetComponent<PlayerStats>();
29     }
30
31     public void Damage(int damageValue)
32     {
33         _currentHealth -= damageValue;
34
35         if (_currentHealth < 0)
36         {
37             _currentHealth = 0;
38         }
39
40         if (_currentHealth == 0)
41         {
42             Animation a = GetComponentInChildren<Animation>();
43             a.Stop();
44
```

```
45                  if (tag == "Player")
46                  {
47                      Destroy(GetComponent<PlayerMovement>());
48                      Destroy(GetComponent<PlayerAnimation>());
49                      Destroy(GetComponent<RifleWeapon>());
50                  }
51                  else // its an enemy
52                  {
53                      _playerStats.ZombiesKilled++;
54                      EnemySpawnManager.OnEnemyDeath();
55                      Destroy(GetComponent<EnemyMovement>());
56                      Destroy(GetComponentInChildren<EnemyAttack>());
57                  }
58
59                  Destroy(GetComponent<CharacterController>());
60
61                  Ragdoll r = GetComponent<Ragdoll>();
62                  if (r != null)
63                  {
64                      r.OnDeath();
65                  }
66              }
67          }
68
69          void Update()
70          {
71              if (IsDead && !_renderer.isVisible)
72              {
73                  Destroy(gameObject);
74              }
75          }
76 }
77
```

Bir zombi ölünce yapılacak şeyleri Health scriptinde yazmıştık. Bu yüzden bu kodu da Health scriptinin içine yazıyoruz. Yaptığımız şey basit: eğer ölen kişi bir zombi ise (player değilse) PlayerStats scriptindeki _zombiesKilled değişkenini 1 artırıyoruz (bunu yaparken ZombiesKilled adındaki property'i kullanıyoruz).

Artık öldürülen zombi sayısını ekrana yazdırabiliriz.

CombatGui scriptini güncelleyin:

```
01 using UnityEngine;
02 using System.Collections;
03
04 public class CombatGui : MonoBehaviour
05 {
06      Health _playerHealth;
07      PlayerStats _playerStats;
08
09      [SerializeField]
10      Texture2D _gameOverImage;
11
12      [SerializeField]
13      Texture2D _winImage;
14
```

```
15      void Start()
16      {
17          GameObject playerGameObject = GameObject.FindGameObjectWithTag("Player");
18          _playerHealth = playerGameObject.GetComponent<Health>();
19          _playerStats = playerGameObject.GetComponent<PlayerStats>();
20      }
21
22      void OnGUI()
23      {
24          if (_playerHealth.IsDead)
25          {
26              float x = (Screen.width - _gameOverImage.width) / 2;
27              float y = (Screen.height - _gameOverImage.height) / 2;
28              GUI.DrawTexture(new Rect(x, y, _gameOverImage.width, _gameOverImage.height),
29                                      _gameOverImage);
30              GUI.Label(new Rect(x + 100, y + 280, 100, 100), "Zombies killed: " + _playerStats.
31                              ZombiesKilled);
32          }
33          else if (GameManager.HasPlayerWon)
34          {
35              float x = (Screen.width - _winImage.width) / 2;
36              float y = (Screen.height - _winImage.height) / 2;
37              GUI.DrawTexture(new Rect(x, y, _winImage.width, _winImage.height), _winImage);
38              GUI.Label(new Rect(x + 100, y + 280, 100, 100), "Zombies killed: " + _playerStats.
39                              ZombiesKilled);
40          }
41      }
42 }
43
```

PlayerGui scriptinde nasıl ekrana sağlığımızı yazdırıyorsak burada da benzer şekilde GUI.Label kullanarak öldürdüğümüz zombi sayısını ekrana yazdırıyoruz.

Helikopteri Oyuna Eklemek

Belli bir süre geçince oyunu otomatik olarak kazanmanın devri geçti artık. 3 dakikalık süre oyunu kazanmak için tetikleyici olmayacak ama kurtarma helikopterinin yere iniş yapması için tetikleyici olacak. Helikopter yere inip de kapılarını açtığında ona binebileceğiz ve işte o zaman oyunu kazanmış olacağız.

İlk önce 3D helikopter modelini projemize import edelim. Bunun için Winrar arşivinin olduğu yerdeki Helicopter klasörünü kopyalayın ve projenizin olduğu yerdeki "Models" klasörünün içine yapıştırın.

Project panelinden Helicopter modelini tutup sahnenize sürükleyin. Helikopter, karakterin yanında çok küçük kalabilir. Bunu çözmek için modelin Scale Factor'ünü 0.06 yapmayı deneyin.

ÇEVİRMEN EKLEMESİ: Helicopter modelini seçin. Inspector'dan "**Rig**" sekmesine geçiş yapın ve "**Animation Type**"ı "Generic"ten "**Legacy**"e çevirin.

Helikoptere Collider Vermek

Şu anda helikopterin içinden geçebiliyoruz. Bunu birlikte çözelim.

“Helicopter Collider” adında yeni bir empty gameobject oluşturun ve bu objeyi sahnedeki Helicopter objesinin bir child'ı yapın (Bunu yapmanın kısa yolu **Hierarchy** panelinden **Helicopter** objesini seçmek ve menü barından **GameObject-Create Empty Child** yolunu izlemektir). Sonrasında "Helicopter Collider"ın pozisyonunu (0,0,0) yapın.

Ardından Component > Physics > Box Collider yolunu izleyerek “Helicopter Collider”a box collider verin.

Shift tuşuna basılı tutun ve collider'ın kenarlarında beliren noktalardan tutup sürükleyerek onu, helikopterin gövdesini kaplayacak şekilde boyutlandırın (ben Collider'ın Center'ını (0, 2.3, -0.3) ve Size'ını (2, 3, 6.5) yapınca güzel durdu):

Şimdi “Helicopter Collider” objesini CTRL + D ile klonlayın ve klonu, collider'ını kuyruğu kapatacak şekilde ayarlayın (Center'ı (0, 2.7, -6.5) ve Size'ı (1, 1, 6.5) yapınca güzel oturuyor):

"Helicopter Collider"ı bir kere daha klonlayın ve bu sefer collider'ı helikopterin burnunu kapatacak şekilde boyutlandırın (objeyi döndürmeniz gerekebilir)(ben objenin X ekseninde rotation'ını 36 derece yaptım ve collider'ın Center'ını (0, 3.5, 1.48), Size'ını (2, 1, 3) yaptım):

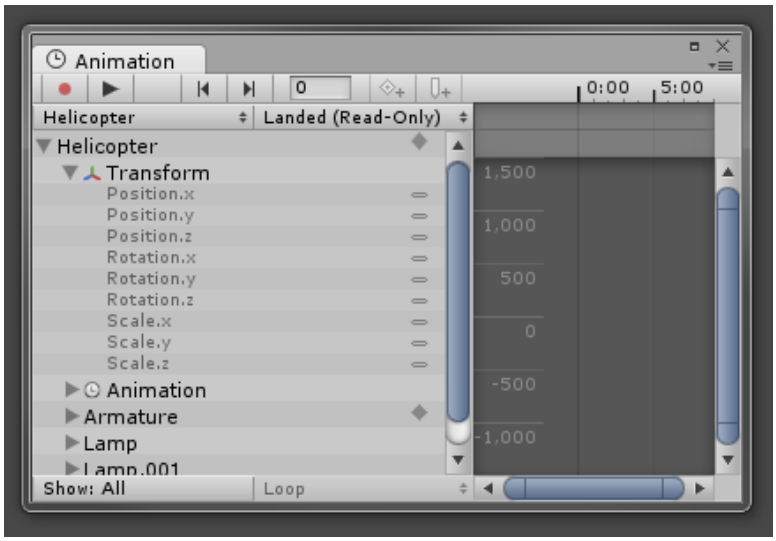
Oyunu çalıştırın. Artık helikopterin içinden geçememeniz lazım.

Helikopterin İniş Yapmasını Ayarlamak

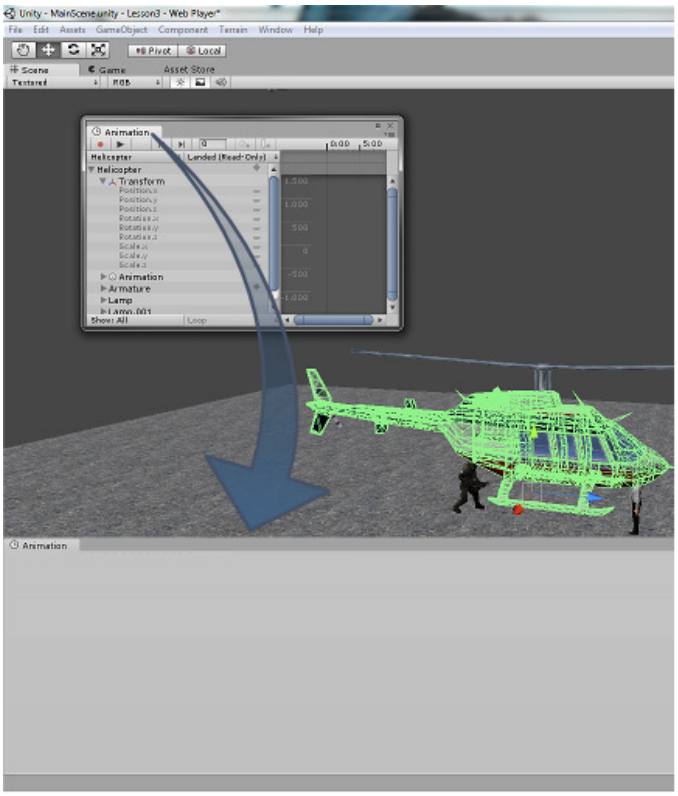
Bunun için kendimiz basit bir animasyon yapacağız.

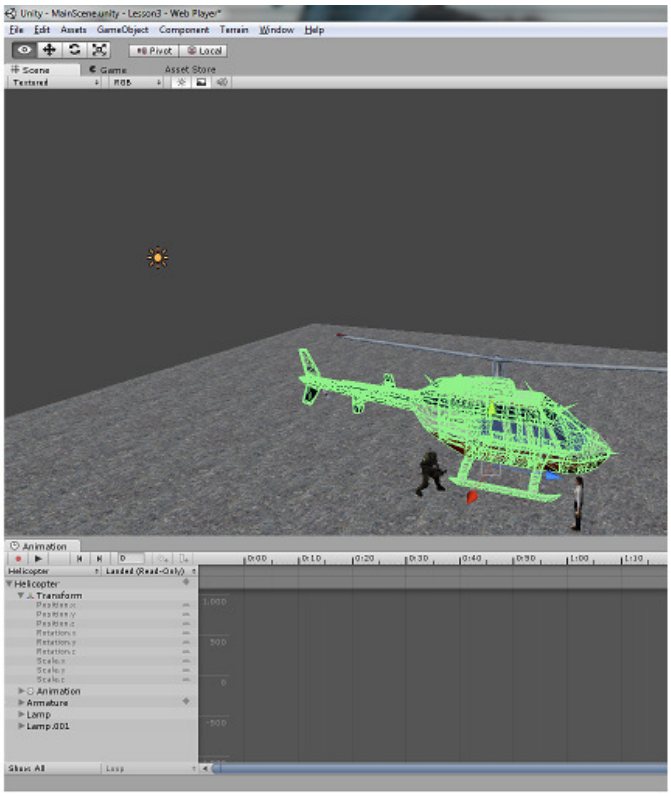
Animation View'ı Kullanmak

Unity'de animasyon oluşturmak için Animation View kullanılır. Window > Animation ile ya da Ctrl+6 ile bu pencereyi açın:



Animation penceresini başlığından (Animation yazan sekme) tutun ve ekranın alt kısmına sürükleyin. Böylece Animation penceresi ekranın alt kısmına sabitlenecek:

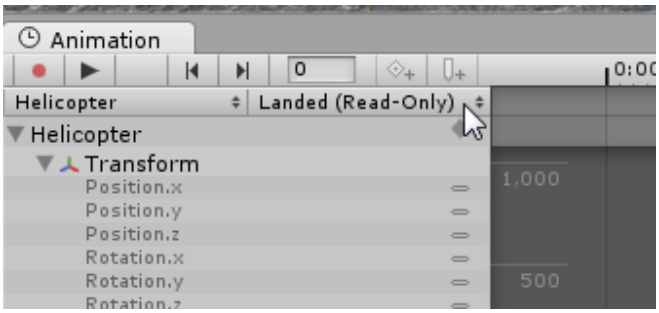




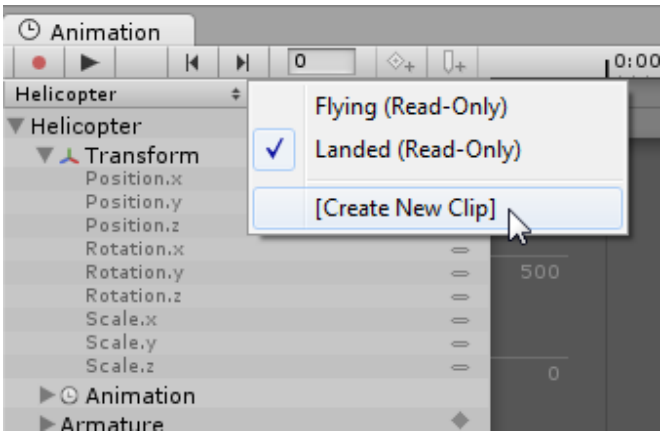
Yeni Bir Animasyon Oluşturmak

Helicopter objesini Hierarchy panelinden seçin. Parent objeyi seçtiğinizden emin olun, yanlışlıkla bir child objeyi seçmeyin.

Animation penceresinde “Landed (Read-Only)” ya da "Flying (Read-Only)" yazan bir yer var. Bu, seçili olan animasyonu temsil eder.



Yeni bir animasyon oluşturmak için onun üzerine tıklayıp “[Create New Clip]”i seçin:



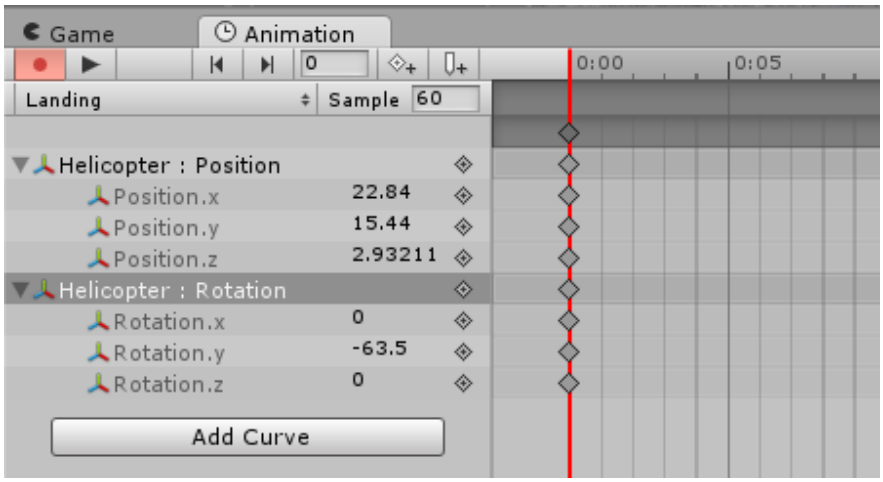
Animasyonu TheGame/Models/Helicopter/ klasörünün içine “Landing.anim” adıyla kaydedin.

ÇEVİRMEN EKLEMESİ: Bende nedense "[Create New Clip]" seçeneği yoktu. Bu yüzden işleri elle hallettim. Eğer sizde de yoksa **Project** panelinden **Helicopter klasörünü** seçin ve **Create-Animation** yolunu izleyin. Animasyona "**Landing**" adını verin. Son olarak **Hierarchy**'den **Helicopter objesini** seçin. **Animation component**'indeki **Animations'ın Size'ını** 3 yapın ve "**Element 2**"ye değer olarak "**Landing**" animasyonunu verin. Artık **Animation penceresinde** Landing animasyonu da gözükecek.

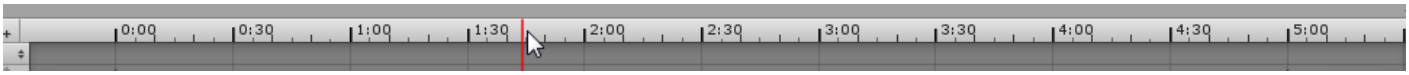
Animasyonumuzu oluşturmaya hazırız. Kırmızı daire ikonuna sahip butona tıklayın. Buton kırmızı renge bürünecek (ayrıca Play butonu da kırmızı olacak).

Helikopteri havada bir yere taşıyın. Oyunun başında helikopter burada yer alacak.

Animation penceresinde gri noktalar belirecek. Bu noktalar *keyframe* oluyor. Objenin o özelliklerinin animasyona dahil edildiğini belirtiyorlar:



Animation penceresinin tepesindeki zaman çizelgesini gördünüz mü? Onda istediğiniz yere tıklayın. Kırmızı çizgi oraya zıplayacak. Bu kırmızı çizgi o an animasyonun hangi karesinde olduğumuzu belirlemekte.



Eğer mouse'nin orta tekerleğini çevirirseniz bu zaman çizelgesine zoom yaparsınız. “5:00” zamanı çizelgede belirene kadar zoom-out yapın. “5:00”a tıkladığımızda animasyonun 5. saniyesine gitmiş oluyoruz.

5. saniyedeyken helikopteri zeminde istediğiniz bir yere indirin.

Helikopterimizin iniş yapması toplam 5 saniye sürüyor. Zaman çizelgesinde bir yere tıklayarak mevcut frame'i değiştirdikçe helikopterin animasyonunun gerçekleştiğini göreceksiniz. İsterseniz daha çok keyframe oluşturarak daha detaylı bir animasyon yaparsınız. Benim için 2 keyframe yeterli.

Kırmızı daire ikonlu butona tekrar tıklayarak aniamsyon modundan çıkın.

ÇEVİRMEN EKLEMESİ: Helikoptere animasyon verirken **Animation** component'ini kullanıyoruz. Bu, eski (**Legacy**) animasyon sistemi; artık sizin de bildiğiniz üzere **Mecanim** sistemi kullanılıyor. Animation component'inde kullanılan animasyonların **"Legacy" animasyon** olarak belirtilmesi lazım yoksa Unity "*The AnimationClip 'Landing' used by the Animation component 'Helicopter' must be marked as Legacy.*" uyarısı verir. Bunu yapmanın yolu basit: **Project** panelinden **Landing** animasyonunu seçin. **Inspector**'a sağ tıklayıp "**Debug**" moduna geçiş yapın ve "**Animation Type**"ın değerini 2'den 1'e çevirin. Inspector'a tekrar sağ tıklayıp "**Normal**" moda geri geçiş yapın. Bu kadar!

Helikopter Scripti

Helikopterin sahip olduğu animasyonları kontrol etmek için bir script yazalım.

Yeni bir C# scripti oluşturup ismini “Helicopter” yapın. Scripti Helicopter objesine component olarak verin. Ardından kodu şöyle düzenleyin:

```
01 using UnityEngine;
02 using System.Collections;
03
04 public class Helicopter : MonoBehaviour
05 {
06     void Start()
07     {
08         animation["Flying"].wrapMode = WrapMode.Loop;
09         animation["Landed"].wrapMode = WrapMode.ClampForever;
10         animation["Landing"].wrapMode = WrapMode.ClampForever;
11     }
12 }
```

Helikopterin üç animasyonu var: “Flying” animasyonu pervaneyi çevirmeye yararken “Landed” animasyonu helikopterin kapılarını açmaya yarar. Az önce oluşturduğumuz “Landing” animasyonu ise helikopteri yere indirmeye yarıyor.

"Flying" animasyonunun wrapMode'unu Loop yapıyoruz. Bunun anlamı bu animasyon bittiği zaman başa saracak ve oynamaya devam edecek (loop yapacak). Diğer iki animasyonun wrapMode'unu ise ClampForever yaptık. Bunun anlamı ise o animasyonlar bittikten sonra animasyonun son frame'inde kalacaklar, tekrar başa sarmayacaklar. Helikopterin yere indikten sonra tekrar havaya ışınlanması garip dururdu herhalde.

Oyunun başında, pervanelerin dönmesini sağlayan “Flying” animasyonunun oynamasını sağlayalım:

```
01 using UnityEngine;
02 using System.Collections;
03
04 public class Helicopter : MonoBehaviour
05 {
06     void Start()
```

```
07      {
08          animation["Flying"].wrapMode = WrapMode.Loop;
09          animation["Landed"].wrapMode = WrapMode.ClampForever;
10          animation["Landing"].wrapMode = WrapMode.ClampForever;
11
12          animation.Blend("Flying", 1.0f, 0.01f);
13      }
14 }
```

Animation component'inin **Blend()** fonksiyonunu kullandık. Bu fonksiyon iki animasyonun bir arada oynamasını sağlar (tıpkı karakterin ileri-sağa koşarken ileri koşma ve sağa koşma animasyonlarının beraber oynaması gibi). Eğer Animation component'inin Play() fonksiyonunu kullansaydık diğer animasyonlar otomatik olarak durdurulur ve sadece "Flying" animasyonu oynardı. Biz de ne "Landed" animasyonunu ne de "Landing" animasyonunu oynatamazdık.

Blend fonksiyonunda iki parametre daha var. İlk parametre olan 1.0f değeri animasyonun diğer animasyonlar ile ne kadar harmanlanacağını belirler. 1.0f değeri verirsek animasyon olduğu gibi harmanlanır. İkinci parametre olan 0.01f ise animasyonun kaç saniye içinde harmanlanacağını belirtir. 0.01f verdiğimiz için animasyon anında diğer animasyonlar ile harmanlanır ama eğer 5.0f gibi bir değer verseydik yavaş yavaş harmanlanacaktı (Ben bu parametreleri değiştirince bir fark gözlemleyemedim açıkçası).

ÇEVİRMEN EKLEMESİ: Oyunu test ederseniz helikopterin kapılarının açıldığını göreceksiniz ama biz öyle bir komut vermedik. Bunun sebebi **Animation** component'inde "**Play Automatically**" seçeneğinin seçili olması. Onun başındaki işareti kaldırın.

Şimdi "Flying" animasyonu ile “Landing” animasyonunu harmanlayalım ve böylece helikopterin pervaneleri dönerken aynı zaman yere iniş yapmasını sağlayalım:

```
01 using UnityEngine;
02 using System.Collections;
03
04 public class Helicopter : MonoBehaviour
05 {
06      void Start()
07      {
08          animation["Flying"].wrapMode = WrapMode.Loop;
09          animation["Landed"].wrapMode = WrapMode.ClampForever;
10          animation["Landing"].wrapMode = WrapMode.ClampForever;
11
12          animation.Blend("Landing", 1.0f, 0.01f);
13          animation.Blend("Flying", 1.0f, 0.01f);
14      }
15 }
```

Oyunu test edin. İki animasyonun birlikte oynaması lazım.

Helikopter yere indiğinde ise kapılarının açılmasını, yani "Landed" animasyonunun devreye girmesini istiyoruz.

"Landed" animasyonunun sadece "Landing" animasyonu son frame'ine ulaştığında (helikopter yere indiğinde) oynatılmasını sağlayalım:

```
01 using UnityEngine;
02 using System.Collections;
03
04 public class Helicopter : MonoBehaviour
```

```
05 {
06       void  Start()
07       {
08             animation["Flying"].wrapMode = WrapMode.Loop;
09             animation["Landed"].wrapMode = WrapMode.ClampForever;
10             animation["Landing"].wrapMode = WrapMode.ClampForever;
11
12             animation.Blend("Landing", 1.0f, 0.01f);
13             animation.Blend("Flying", 1.0f, 0.01f);
14       }
15
16       void  Update()
17       {
18             if  (animation["Landing"].normalizedTime >= 1.0f)
19             {
20                   animation.Blend("Landed", 1.0f, 0.01f);
21             }
22       }
23 }
```

18. satırda "Landing" animasyonunun normalizedTime'ının 1.0'dan büyük olup olmadığına bakıyoruz. normalizedTime, animasyonun hangi safhasında olduğumuzu belirler. Değeri 0.0 ise animasyon ilk frame'de, 1.0 ise son frame'de, 0.5 ise tam ortadaki frame'dedir.

Eğer "Landing" animasyonunun sonuna gelmişsek 20. satırdaki kod ile "Landed" animasyonunu diğer animasyonlarla harmanlıyoruz (Blend).

Oyunu çalıştırın. Helikopter yere inince kapıları açılacak.

Helikopterin İstediğimiz Zaman İnmesini Sağlamak

Şu anda oyun başlayınca helikopter yere inmeye başlıyor. "Landing" animasyonunu oyunun başında durdurmalı (pause) ve yeterince zaman geçtikten sonra oynatmalıyız:

```
01 using  UnityEngine;
02 using  System.Collections;
03
04 public class  Helicopter : MonoBehaviour
05 {
06       void  Start()
07       {
08             animation["Flying"].wrapMode = WrapMode.Loop;
09             animation["Landed"].wrapMode = WrapMode.ClampForever;
10             animation["Landing"].wrapMode = WrapMode.ClampForever;
11
12             animation.Blend("Landing", 1.0f, 0.01f);
13             animation.Blend("Flying", 1.0f, 0.01f);
14
15             animation["Landing"].speed = 0;
16       }
17
18       public  void  Land()
19       {
20             animation["Landing"].speed = 1;
21       }
22
23       void  Update()
```

```
24      {
25          if (animation["Landing"].normalizedTime >= 1.0f)
26          {
27              animation.Blend("Landed", 1.0f, 0.01f);
28          }
29      }
30 }
```

Animasyonun speed'ini (hız) 0.0 yaptığımız zaman animasyon duruyor ve geri 1.0 yaptığımız zaman oynamaya devam ediyor. Eğer Land() fonksiyonunda hızı 2.0 yapsaydık animasyon iki kat hızlı oynar, -1.0 yapsaydık baştan sona değil sondan başa oynardı.

Land fonksiyonu public bir fonksiyon. Helikopterin yere ne zaman inmesini istiyorsak bu public fonksiyonu o zaman çağırmamız yeterli.

GameManager scriptini açın ve içeriğinin büyük kısmını silin:

```
01 using UnityEngine;
02 using System.Collections;
03
04 public class GameManager : MonoBehaviour
05 {
06      static bool _hasPlayerWon = false;
07
08      public static bool HasPlayerWon { get{ return _hasPlayerWon; } }
09 }
```

Geriye sadece _hasPlayerWon değişkeni ve HasPlayerWon property'si kalsın.

Helicopter scriptini açıp şöyle güncelleyin:

```
01 using UnityEngine;
02 using System.Collections;
03
04 public class Helicopter : MonoBehaviour
05 {
06      [SerializeField]
07      float _secondsToLand = 180.0f;
08
09      void Start()
10      {
11          animation["Flying"].wrapMode = WrapMode.Loop;
12          animation["Landed"].wrapMode = WrapMode.ClampForever;
13          animation["Landing"].wrapMode = WrapMode.ClampForever;
14
15          animation.Blend("Landing", 1.0f, 0.01f);
16          animation.Blend("Flying", 1.0f, 0.01f);
17
18          animation["Landing"].speed = 0;
19
20          Invoke("Land", _secondsToLand);
21      }
22
23      public void Land()
24      {
25          animation["Landing"].speed = 1;
26      }
27
28      void Update()
```

```
29      {
30          if (animation["Landing"].normalizedTime >= 1.0f)
31          {
32              animation.Blend("Landed", 1.0f, 0.01f);
33          }
34      }
35 }
```

Invoke() fonksiyonu ile _secondsToLand saniye sonra Land fonksiyonunu çağırıyoruz, yani helikoptere inmesini söylüyoruz.

Helikoptere Ulaşınca Oyunu Kazanmak

Geriye oyuncunun helikoptere ulaşınca oyunu kazanması kaldı. Bunun için “DoorsRight” adında yeni bir empty gameobject oluşturup bunu Helicopter objesinin child objesi yapın.

DoorsRight'ın pozisyonunu (0,0,0) yapın. Ardından DoorsRight'a box collider verin (Component > Physics > Box Collider).

Box Collider'ı helikopterin sağ taraftaki kapılarını dışarıdan kaplayacak şekilde konumlandırın/boyutlandırın (Center (1, 2, 1) ve Size (1, 2, 4) güzel duruyor). Oyuncu bu collider ile temas edince oyunu kazanacak.

Şimdi Inspector'dan Box Collider'daki “Is Trigger” seçeneğini işaretleyin ki karakter bu collider ile temas edince içinden geçebilsin.

Oyuncunun kapıdaki bu collider ile temas edip etmediğini anlamak için yeni bir C# scripti oluşturun. İsmini “HelicopterDoors” yapın ve kodu düzenleyin:

```
01 using UnityEngine;
02 using System.Collections;
03
04 public class HelicopterDoors : MonoBehaviour
05 {
06      void OnTriggerEnter(Collider c)
07      {
08          if (c.transform.root.tag == "Player")
09          {
10          }
11      }
12 }
```

Yaptığımız şey basit: "Is Trigger"ı işaretli olan collider'ımız ile temas eden objenin tag'ının "Player" olup olmadığına bakıyoruz. Eğer Player ise oyunu kazanacağız.

Kazanma işlemi için önce GameManager scriptinde şu değişikliği yapın:

```
01 using UnityEngine;
02 using System.Collections;
03
04 public class GameManager : MonoBehaviour
05 {
06     static bool _hasPlayerWon = false;
07
08     public static bool HasPlayerWon { get{ return _hasPlayerWon; } }
09
10     public static void WinPlayer()
11     {
12         _hasPlayerWon = true;
13     }
14 }
```

HelicopterDoors scriptine geri dönün ve eğer Player ile temas etmişsek WinPlayer fonksiyonunu çağırın:

```
01 using UnityEngine;
02 using System.Collections;
03
04 public class HelicopterDoors : MonoBehaviour
05 {
06     void OnTriggerEnter(Collider c)
07     {
08         if (c.transform.root.tag == "Player")
09         {
10             GameManager.WinPlayer();
11         }
12     }
13 }
```

HelicopterDoors scriptini DoorsRight objesine verip oyunu test edin. Sağ kapıdaki collider ile temas edince oyunu kazanacaksınız. Sol kapı için yeni bir obje oluşturup (DoorsLeft) ona da "Is Trigger"lı Collider ile HelicopterDoors scriptini vermeyi size bırakıyorum.

Görsel 13.1: Oyuncu helikopterin kapısındaki collider ile temas edince oyunu kazanıyor.

Özet Geçecek Olursak...

Bu bölümde bir fonksiyonu Invoke ile istediğimiz kadar saniye sonra çalıştırmayı ve Unity'de basit animasyonlar oluşturmayı gördük.

Oyunumuz aşağı yukarı tamamlandı. Sonraki derslerde oyuna bazı çerez eklemeler yapacağız; ana menü gibi...

ÇEVİRMEN EKLEMESİ: Bence artık sahnede yer alan üç tane zombi klonunu silin. Zaten oyun ilerledikçe spawn oluyorlar, o üçünün orada durması gereksiz.