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

Unity 3D İle Oyun Programlama

Bölüm 4: Player Scriptine 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 bölümle birlikte oyunumuzu yapmaya başlıyoruz. Yapacağımız oyun TPS bakış açısı ile oynanan bir zombi-shooter olacak.

Bu dersin genel olarak amacı programlamaya odaklanmak. Bu yüzden oyunumuzda hep hazır modeller, kaplamalar, ses dosyaları vb. kullanacağız. Örneğin resimde gördüğünüz karakter ve zombi modelleri Unity Asset Store'daki ücretsiz modeller arasından alınmıştır.

Dersin bu bölümünde oyuncuyu (player) oluşturacak ve klavye ile hareket ettirmeye çalışacağız.

“TheGame” adında yeni bir proje oluşturun.

Oyun Alanı

Önce karakterimizin yürüyeceği zemini oluşturalım. Bir küp oluşturun (GameObject > Create Other > Cube) ve onu (0, 0, 0) koordinatlarına yerleştirin (Position değerlerini değiştirerek). Scale değerini (50, 1, 50) yaparak küp objesini boyutlandırın.

Küpün ismini “Ground” olarak değiştirin.

Şimdi GameObject > Create Other > Directional Light ile sahneye ışık ekleyin ve ışığı (30, -30, 0) şeklinde döndürün.

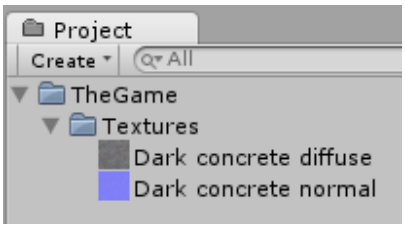
Zemin gri renkte güzel durmuyor. Onun üzerine bir kaplama atalım. Bunun için Winrar arşivinin oradaki Images klasöründe yer alan “Dark concrete diffuse.png” ve “Dark concrete normal.png” resimlerini projenize import edin.

Project Panelini Düzenli Tutmak

İstediğimiz şeye en hızlı şekilde erişmenin en iyi yolu Project panelini klasörler aracılığıyla düzenli tutmak.

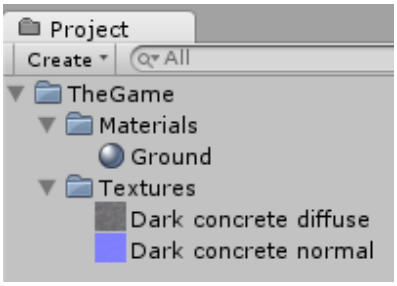
Project panelindeki Create butonunu kullanarak “TheGame” adında bir klasör (folder) oluşturun.

Klasörün içine “Textures” adında başka bir klasör oluşturup az önce import ettiğiniz kaplama asset'lerini bu klasöre taşıyın.

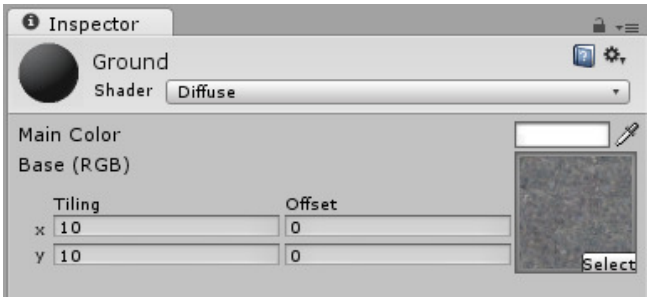


Zemine Kaplama Atamak

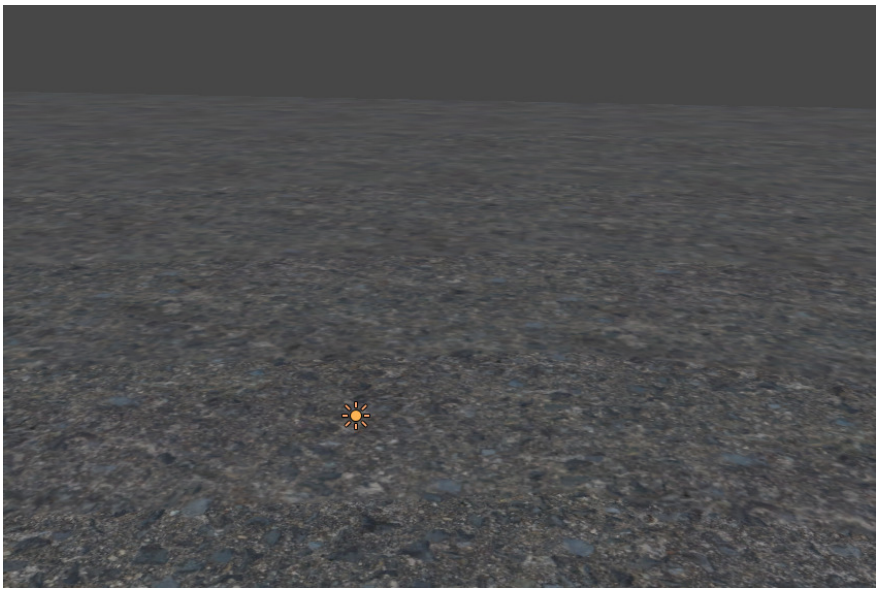
“TheGame” klasörünün içinde “Materials” adında başka bir klasör daha oluşturun. Bu klasörü seçip içinde yeni bir materyal (material) oluşturun, ismini "Ground" yapın.



Ground materyalini seçip Texture kısmına değer olarak “Dark concrete diffuse” texture'sini verin. Tiling değerinin X ve Y'sini 10 yapın.

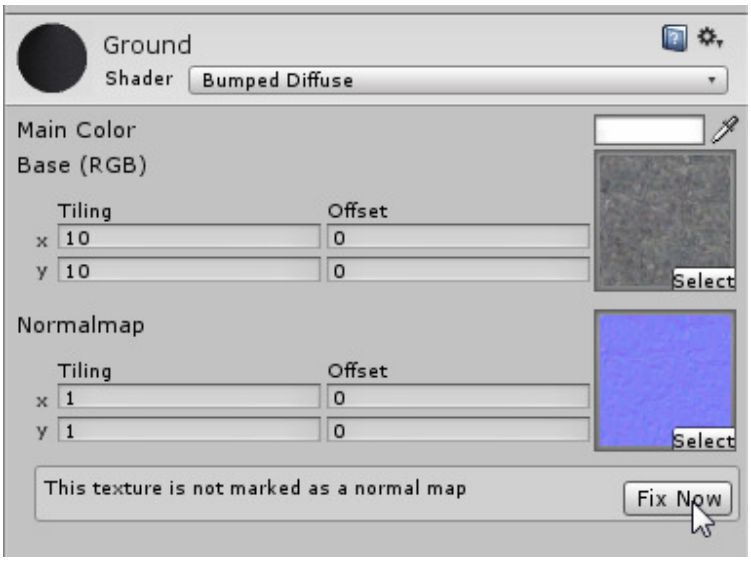


Şimdi bu materyali zemine atayalım. Bunun için zemini seçip Inspector'unda yer alan Mesh Renderer component'inin Materials kısmındaki Element 0'a değer olarak Ground materyalini verin.



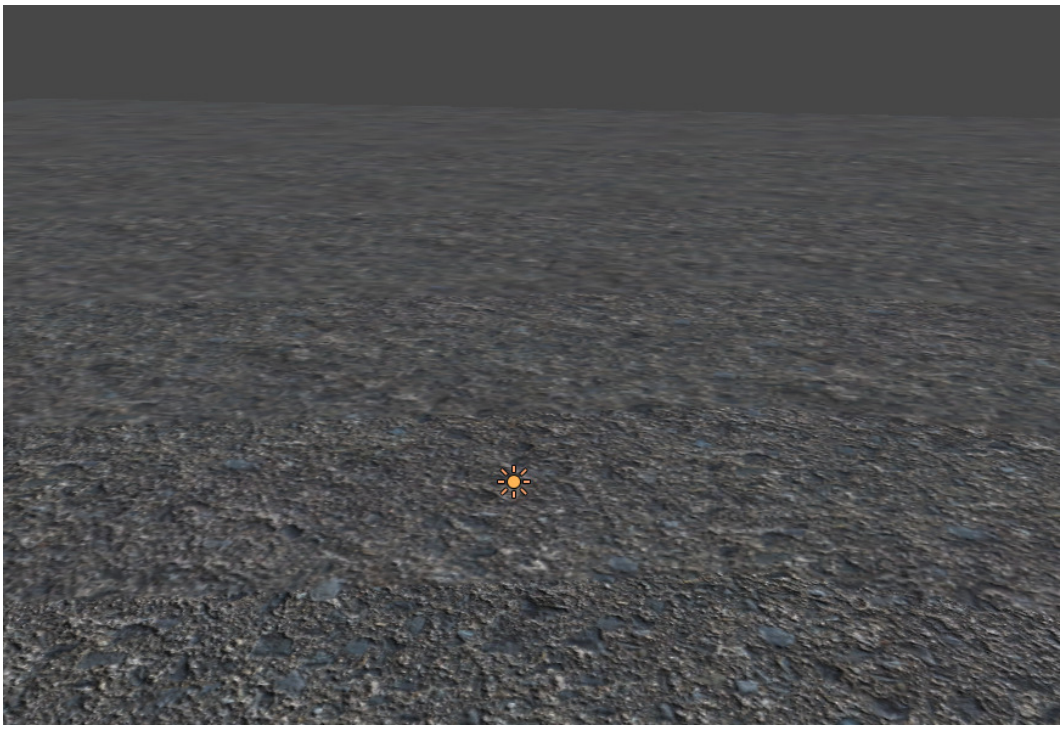
Şimdi materyalin kullandığı shaderı değiştirerek normal-map desteklemesini sağlayacağız. Ground materyalini seçip shader'ını “Bumped Diffuse” olarak değiştirin. “Dark concrete normal” texture'sini Normalmap kısmına değer olarak verin.

"This texture is not marked as a normal map" şeklinde bir uyarı verecek. Yanındaki “Fix Now”a tıklayın.

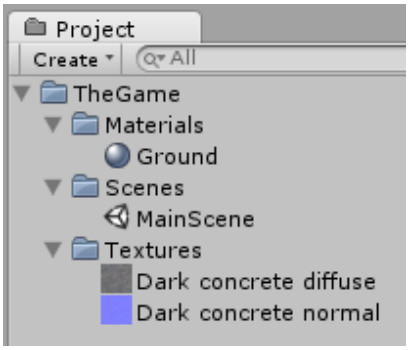


Şimdi Normalmap'in Tiling'ini de 10'a 10 yapın.

Sonuç olarak zemin girintili çıkıntılı gibi, daha gerçekçi duracak:



Oyunu kaydetmek için şimdi iyi bir zaman olabilir. Ctrl + S ile sahneyi kaydetmek istediğinizde nereye kaydetmeyi istediğinizi soracak. "TheGame" klasörünün içinde "Scenes" adında yeni bir klasör oluşturun ve bu klasörün içine sahneyi "MainScene" adıyla kaydedin.



Character Controller

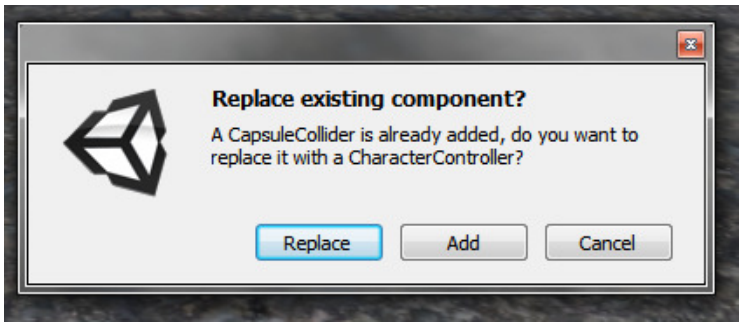
Unity'de bir karakteri hareket ettirmenin kolay yolu Character Controller componenti kullanmaktır. Bu component tek başına bir işe yaramaz ama ona komut veren bir script olunca işte o zaman akan sular durur.

Şimdilik karakterimiz için karmaşık bir 3D model değil basit bir kapsül kullanalım. Yeni bir kapsül oluşturup (GameObject > Create Other > Capsule) (0, 1.75, 0) pozisyonuna yerleştirin.

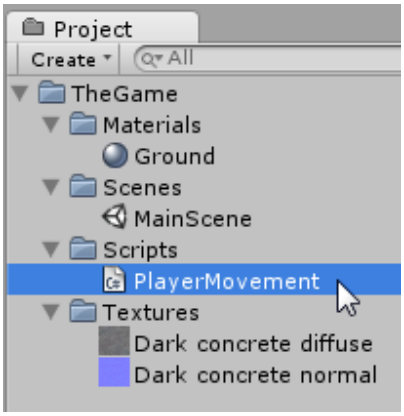
Kapsüle “Player” ismini verin.

Şimdi kapsüle Component > Physics > Character Controller ile Character Controller verin. Eğer bir uyarı penceresi çıkarsa “Replace”e basın.

ÇEVİRMEN EKLEMESİ: Eğer böyle bir uyarı penceresi bende çıkmadığı gibi sizde de çıkmazsa Player objesinin Inspector'undan Capsule Collider'ın sağındaki dişli ikonuna tıklayın ve Remove Component seçeneğini seçin. Yoksa karakter hareket ederken garip bir şekilde uçmaya başlıyor.



Oyuncuyu hareket ettirmek için bir script yazmamız lazım. "TheGame"in içinde "Scripts" klasörü oluşturun. Bu klasörde “PlayerMovement” adında bir C# scripti oluşturun.



PlayerMovement scriptini Player objesine component olarak atayın. Sonra scripti açın.

Keyboard'dan Input Almak

Scripti şu şekilde güncelleyin:

```
01 using  UnityEngine;
02 using  System.Collections;
03
04 public class  PlayerMovement : MonoBehaviour
05 {
06       public  float  h = 0;
07
08       // Use this for initialization
09       void  Start()
10       {
11
12       }
13
14       // Update is called once per frame
15       void  Update()
16       {
17            h = Input.GetAxis("Horizontal");
18       }
19 }
```

Değişkenimizin public olmasını sıkıntı etmeyin, o değişkeni geçici olarak kullanıyoruz. Değişkeni public yapmamızın sebebi değerini kolayca Inspector'dan takip edebilmek.

Oyunu çalıştırın. Klavyeden sol ya da sağ ok tuşuna basılı tutup Inspector'daki H'nin değerini kontrol edin. Değeri bastığınız tuşa göre 0.0'dan 1.0'a ya da -1.0'a kayacaktır. Dilerseniz ok tuşları yerine A ve D tuşlarını da kullanabilirsiniz.

Dışarıdan input almanın bir yolunu gördünüz. Input.GetAxis, hangi ok tuşuna bastığınıza göre -1.0'dan 1.0'a kadar bir değer döndürür. “Horizontal” (yatay) anahtar kelimesi kullandığımız için döndürülen değer sadece sağ ve sol ok tuşlarından (ve A-D tuşlarından) etkileniyor.

Character Controller'a Erişmek

Şimdilik Input kodunu silelim. Scriptin son halinin şöyle olduğundan emin olun:

```
01 using UnityEngine;
02 using System.Collections;
03
04 public class PlayerMovement : MonoBehaviour
05 {
06     CharacterController _controller;
07
08     // Use this for initialization
09     void Start()
10     {
11         _controller = GetComponent<CharacterController>();
12     }
13
14     // Update is called once per frame
15     void Update()
16     {
17     }
18 }
```

Karakteri hareket ettiren component Character Controller; yazmakta olduğumuz script ise Character Controller'a gerekli olan input'ları sağlıyor. Bu scriptten Character Controller'a erişmek için Character Controller component'ini _controller isimli bir değişkende tutuyoruz.

Değişkene değerini Start fonksiyonunda veriyoruz. Burası önemli. Yazdığınız class MonoBehaviour class'ını extend ediyorsa sakın ola yazdığınız class'ın constructor'ını kullanmayın / ellemeyin / kurcalamayın. Yoksa sebebini bir türlü anlamadığınız hatalarla karşılaşabilirsiniz.

GetComponent fonksiyonu, MonoBehaviour class'ı tarafından sunulmakta. Kendisi, bir component'e erişmek için kullanılır. Player objesine bir Character Controller componenti vermiştik; burada GetComponent ile de o component'e erişip onu değişkenimize atıyoruz.

Karakteri Hareket Ettirmek

Şimdi Input kodunu fonksiyona geri ekleyelim ve onu kullanarak karakteri hareket ettirelim:

```
01 using UnityEngine;
02 using System.Collections;
03
04 public class PlayerMovement : MonoBehaviour
05 {
06     CharacterController _controller;
07
08     // Use this for initialization
09     void Start()
10     {
11         _controller = GetComponent<CharacterController>();
12     }
13
14     // Update is called once per frame
15     void Update()
16     {
17         Vector3 direction = new Vector3(Input.GetAxis("Horizontal"), 0, Input.GetAxis("Vertical"));
18         _controller.Move(direction);
19     }
20 }
```

Input.GetAxis'i kullanarak input alıyoruz. Parametre olarak “Horizontal” yazınca sağ-sol ok tuşlarının, "Vertical" yazınca ileri-geri ok tuşlarının döndürdüğü değeri alıyoruz. Ardından bu değerleri bir Vector3 değişkeninde depoluyoruz.

Vector3 yapısı (struct) x, y ve z değerlerine sahiptir. 3 boyutlu uzaydaki pek çok şeyi (pozisyon, eğim, boyut...) ifade etmek için Vector3 ideal yapı tipidir.

“Horizontal” input'u Vector3'ün X değerine, “Vertical” input'u da Vector3'ün Z değerine atıyoruz. Y değerine ise 0 veriyoruz çünkü Y ekseni dikey hareketi temsil eder ve karakterimiz sadece ileri-geri, sağa-sola hareket edecek. Henüz zıplama gibi bir durumumuz yok.

18. satırda Character Controller component'inin Move fonksiyonunu kullanıyoruz. Bu fonksiyon karakteri hareket ettirir. Vector3 değişkenimizi fonksiyona parametre olarak veriyoruz. Böylece karakteri sağa-sola ve ileri-geri hareket ettirebiliyoruz.

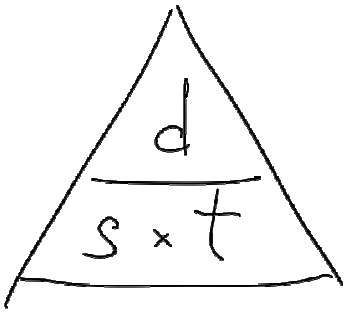
Şimdi oyunu çalıştırıp karakteri ok tuşlarıyla ya da WASD tuşlarıyla hareket ettirmenin keyfini yaşayabilirsiniz.

Saniye Cinsinden Hesap Yapmak

Önceki bölümde bahsettiğim gibi Update içinde sabit sayılarla işlem yapmak iyi değil. Çünkü oyunun hızına göre Update'in çalışma sıklığı değişir. O yüzden hesaplarımızı bir saniyeyi baz alarak yapmalıyız:

```
18         _controller.Move(direction * Time.deltaTime);
```

Fizikten bildiğiniz üzere Hız (speed) = Mesafe (distance) / Zaman (time). 18. satırda hızı zaman ile çarpıyoruz ve haliyle karakterin gittiği mesafeyi ayarlamış oluyoruz. (Sonraki sayfaya bakınız...)



Görsel 4.1: Basit bir fizik kanunu. Lise derslerinizden hatırlarsınız belki.

ÇEVİRMEN EKLEMESİ: Time.deltaTime'ın ne işe yaradığını size hatırlatmak istedim. Update fonksiyonunda Time.deltaTime kullanmadan bir işlem yaparsanız o işlemde adı geçen değerler, Update'in çağrılma sıklığına göre güncellenir. Bu tavsiye edilmiyor, çünkü oyunun yavaş ve hızlı bilgisayarlarda birebir aynı şekilde kullanıcı deneyimi sunmasını engelliyor. Eğer ki Update fonksiyonunda işlem yaparken üzerinden geçtiğiniz değerleri Time.deltaTime ile çarparsanız o zaman artık yaptığınız işlem bir karede (frame) değil bir saniyede gerçekleşecek şekilde anlam kazanır, kullanıcı deneyimi bilgisayardan bilgisayara değişmez. İşte bu, tavsiye edilen yöntem.

Hızı Ayarlayabilmek

Şimdi sorunumuz karakterin çok yavaş hareket etmesi. Input.GetAxis, -1.0'dan 1.0'a bir değer döndürüyor. Yani objeyi saniyede 1 metre hareket ettirebiliyoruz. Bu değeri bir başka değerle çarparak hızı ayarlayabilmeliyiz. Kodu şu şekilde güncelleyin:

```
01 using UnityEngine;
02 using System.Collections;
03
04 public class PlayerMovement : MonoBehaviour
05 {
06     CharacterController _controller;
07
08     [SerializeField]
09     float _moveSpeed = 5.0f;
10
11     // Use this for initialization
12     void Start()
13     {
14         _controller = GetComponent<CharacterController>();
15     }
16
17     // Update is called once per frame
18     void Update()
19     {
20         Vector3 direction = new Vector3(Input.GetAxis("Horizontal"), 0, Input.GetAxis("Vertical"));
21         Vector3 velocity = direction * _moveSpeed;
22         _controller.Move(velocity * Time.deltaTime);
23     }
24 }
```

Artık direction'ın x, y ve z değerlerini _moveSpeed ile çarpıyoruz. Bu sayede, eğer _moveSpeed 5 ise -1.0'dan 1.0'a olan değer yelpazesi -5.0'dan 5.0'a genişlemiş oluyor. Player'ı böylece saniyede 1 değil 5 metre ilerletebiliyoruz.

Karakteri Zıplatmak (Jump)

Space tuşuna basınca karakterin zıplamasını sağlayalım:

```
01 using UnityEngine;
02 using System.Collections;
03
04 public class PlayerMovement : MonoBehaviour
05 {
06     CharacterController _controller;
07
08     [SerializeField]
09     float _moveSpeed = 5.0f;
10
11     [SerializeField]
12     float _jumpSpeed = 20.0f;
13
14     [SerializeField]
15     float _gravity = 1.0f;
16
17     float _yVelocity = 0.0f;
18
19     // Use this for initialization
20     void Start()
21     {
22         _controller = GetComponent<CharacterController>();
23     }
24
25     // Update is called once per frame
26     void Update()
27     {
28         Vector3 direction = new Vector3(Input.GetAxis("Horizontal"), 0, Input.GetAxis("Vertical"));
29         Vector3 velocity = direction * _moveSpeed;
30
31         if (_controller.isGrounded)
32         {
33             if (Input.GetButtonDown("Jump"))
34             {
35                 _yVelocity = _jumpSpeed;
36             }
37         }
38         else
39         {
40             _yVelocity -= _gravity;
41         }
42
43         velocity.y = _yVelocity;
44
45         _controller.Move(velocity * Time.deltaTime);
46     }
47 }
```

Birkaç yeni değişken ekledik: “_jumpSpeed” ne kadar yükseğe zıplayabileceğinizi belirler. "_gravity" ise yerçekiminin gücünü belirler. “_yVelocity” değişkeni objenin dikey eksendeki (Y) hızını depolar ve farkettiğiniz üzere değeri Inspector'da gözükmez çünkü objenin mevcut karedeki (frame) dikey hızını Inspector'dan değiştirmek çok saçma olurdu.

31. satırda kullandığımız isGrounded fonksiyonu Character Controller'ın yere değip değmediğini döndürür. Eğer karakter yere değiyorsa ancak o zaman zıplama işlemini gerçekleştiriyoruz.

33. satırdaki Input.GetButtonDown fonksiyonu bir butona basılıp basılmadığını kontrol etmemize yarar. Parametre olarak “Jump” yazarsak spacebar tuşuna basılıp basılmadığına bakılır. Eğer basmışsak karakterin dikey hızını (_yVelocity) aniden zıplama gücüne eşitliyoruz, böylece karakter zıplıyor.

Eğer karakter havadaysa (isGrounded false döndürürse) onun Y eksenindeki hızını yerçekimi kadar azaltıyoruz. Böylece obje önce yavaşlayıp duruyor, sonra aşağı düşmeye başlıyor (Y hızı negatif olduğunda).

“_yVelocity” değişkeninin değerini 43. satırda velocity değişkeninin y parametresine veriyoruz. Hatırlayın: Move fonksiyonunda velocity değişkenini kullanıyoruz. Az önce velocity'nin y parametresinin değerini değiştirdiğimiz için de obje dikey eksende hareket edebiliyor.

Özet Geçecek Olursak...

Bu bölümde kullanıcıdan input almayı ve bu input'u kullanarak karakteri hareket ettirmeyi gördük. Ayrıca klasörler oluşturarak Project paneline biraz çeki düzen verdik.