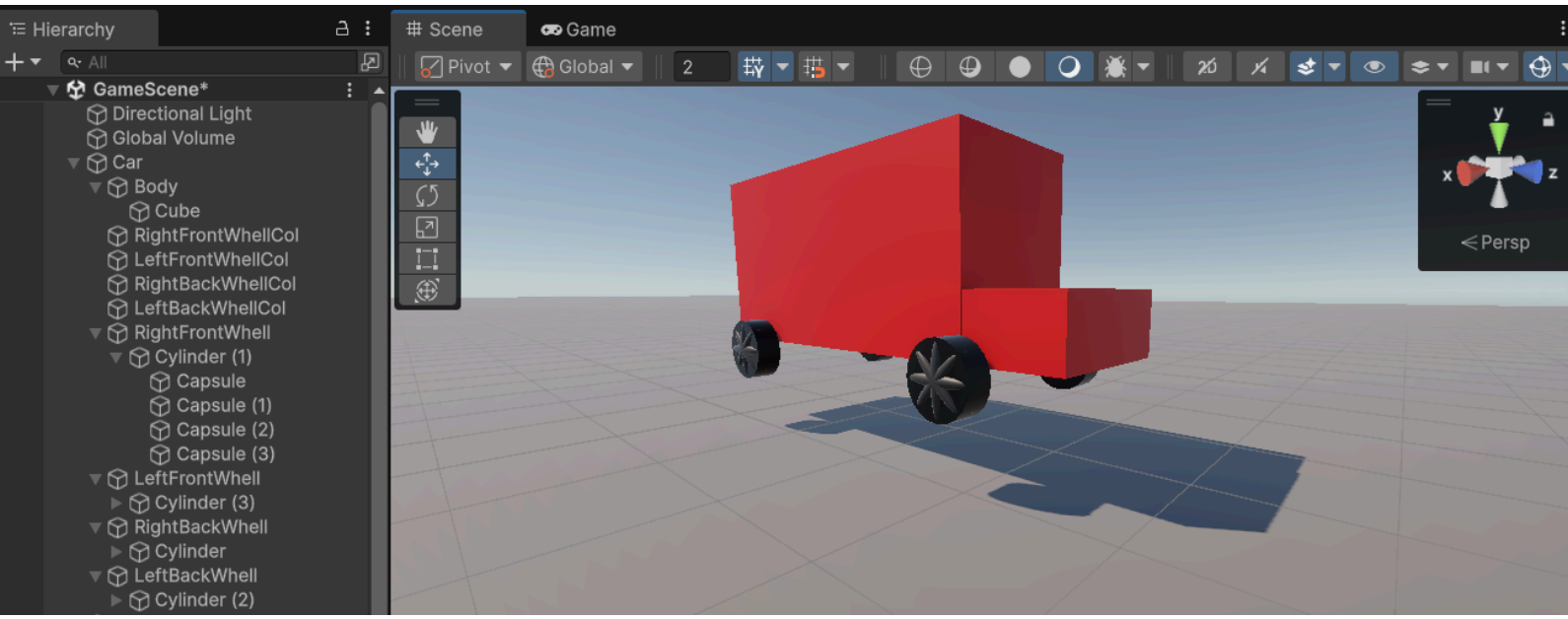
Yapacaklarımız : Gerçek bir araba gibi davranacak, özelleştirilebilir ve kamera açıları değiştirilebilir bir araç sistemi hazırlamak. 4 Adet Script yacağız. Bu Scriptler şu şekilde olacak.

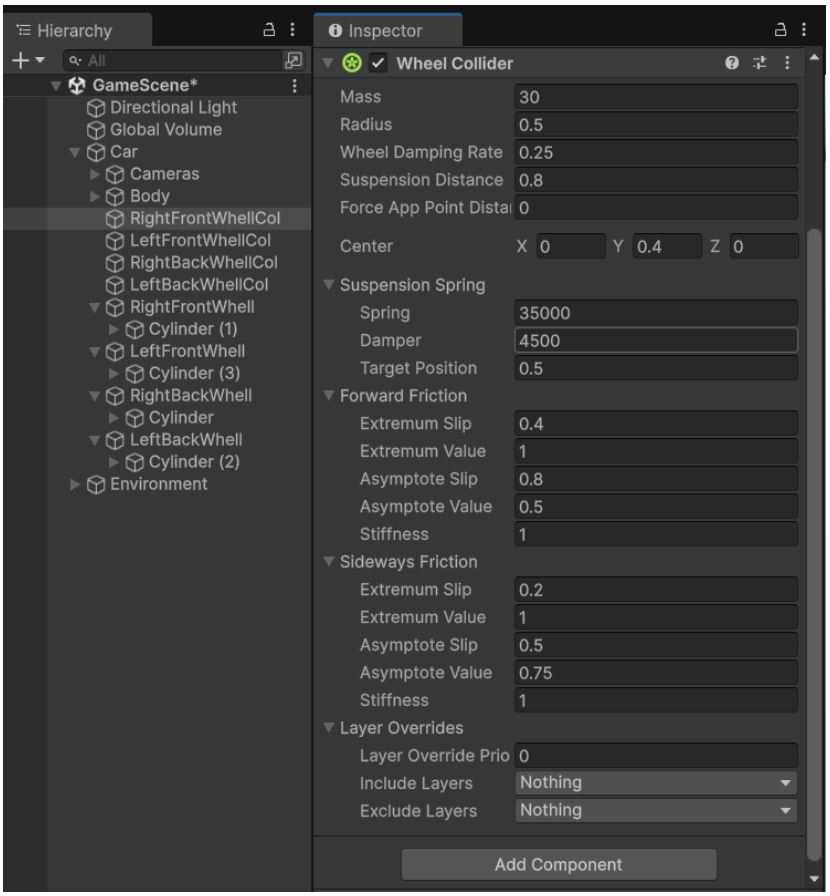
1. CarController : Arabamızın hareket sistemini kontrol edecek.
2. WhellSC : Arabamızın tekerleklerinin dönüşünü ayarlaycak.
3. VehicleStats : Bu Scriptimiz bir ScriptibleObject olacak. Amacı da aracımızın özelliklerini (Max Speed, Motor Torque, Brake) belirlemek.
4. CameraSwitcher : Kamera açılarını ayarlamak için kullanacağımız Script.

Adım 1 : ilk olarak kendi arabamızı tasarlıyoruz. Burada nasıl bir tararım yapacağınız tamamen size kalmış bir durum. ancak arabanın kemik yapısı önemli bir ayrıntı.

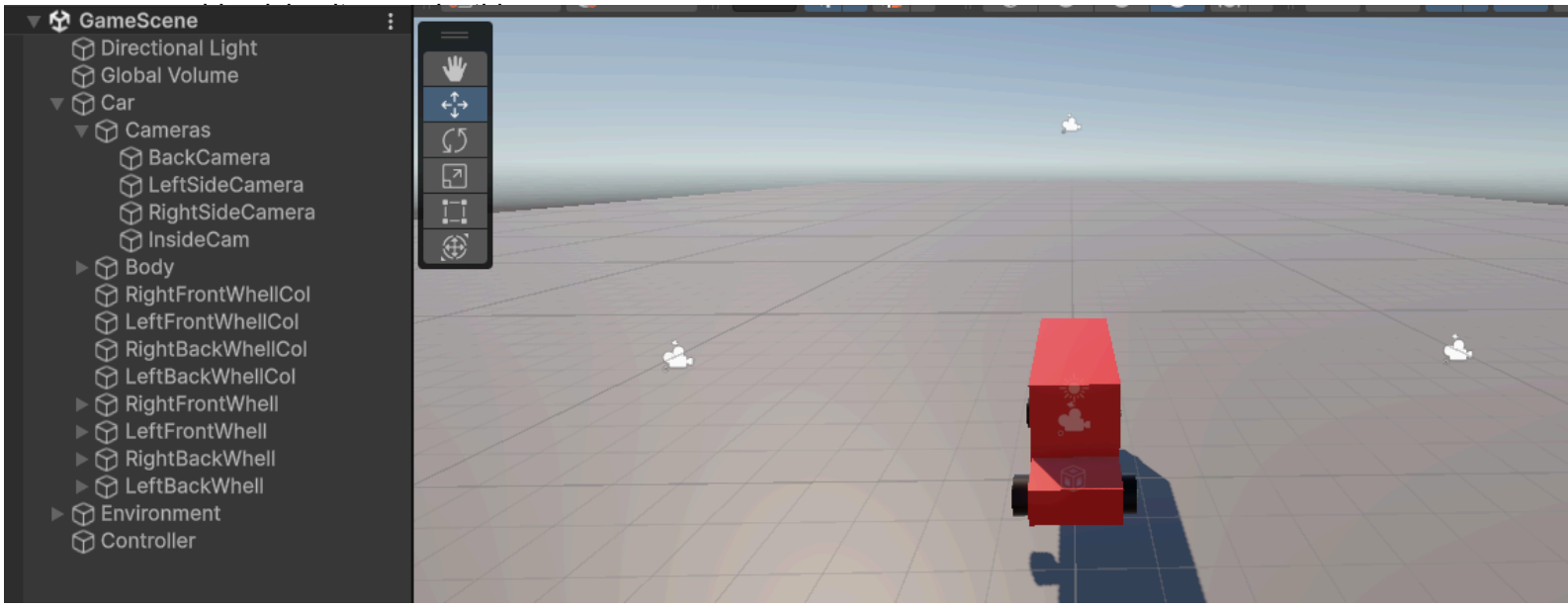


Görmüş olduğunuz arabanın tamamı Car obejsinin altında. Car objesinin altında ilk olarak Body var. Buradaki Body arabanın gövdesi oluyor. istersek kapı, cam, ayna gibi ekstra yeni yerler oluşturacaksak bu bölümün altında olmak zorunda. Ardından tekerleklerin Colliderları geliyor. Bu 4 objenin içerisinde sadece WhellColiider komponenti var. Ancak bu komponenin tam olarak tekerlek ile aynı pozisyonda olması gerekiyor. Ardından da 4 adet daha tekerlek objelerimiz var. Buradaki en üstteki objelerin içerisi boş duruyor. tekerleklerin görünümünü oluşturacak objeleri bu objelerin altında olması gerekiyor. Örneğin ben ilk olarak bir silindir ekleyip yatay yönde çevirdim ki tekerlek gibi gözüksün. Ardından daha güzel gözükmesi için jant gibi gözükecek bir tasarım yaptım. Bu sayede tekerleklerin döndüğünü de doğrulayabileceğiz. Bu tekerleğin görünüşü belirleyen objeler default olarak collider ile gelirler. Bizim Colliderımız zaten diğer objede WhellCollider şeklinde olduğu için tekerleğin görünüşünü belirleyen objelerden colliderları kaldırıyoruz.

Adım 2 : Tekerleklere WheelCollider eklemek.
Bu Collider sayesinde artık bir tekerlek sistemimiz olmuş oluyor. Burada süspansiyon ayarlarımızı, sürtünme kuvvetlerini vs. Birçok ayarı değiştirebiliyoruz. Dilerseniz default gelen ayarları kullanabilirsiniz. Yada sağda görmüş olduğunuz ayarları da kullanabilirsiniz, zaten sadece birkaç ayarı farklı. Bu WheelCollider componentini tekerlek objelerimizin içerisine değil RightFrontWhellCol gibi boş objelerin içerisine atmamız gerektiğini unutmayın. Ayrıca bu Collider objelerinin konumlarının da kendi araba tekerleklerimizin konumları ile aynı olması gerektiğini de tekrardan hatırlatmak istiyorum.



Adım 3 : Birden fazla kamera açısı için birden fazla kamera objesi oluşturup kendi araba

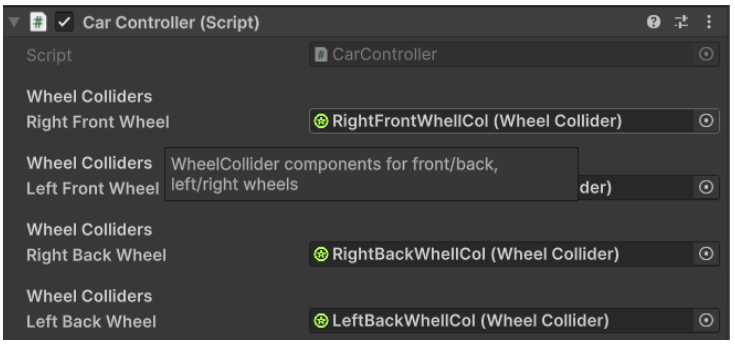


Buradaki yerleşimler sizin keyfinize kalmış. ancak kameraların araba objenizin altında olması gerektiğini unutmayın. yoksa kameralar araba ile hareket etmezler. çünkü bir kamera takip sistemi kullanmıyoruz. Kameraların araba ile hareket etmesinin sebebi parent-child ilişkisi.

Adım 4 : Scriptlerin yazılması

1. CarController :

Bu Scriptimizi aracımızın en kökünde bulunan objemize atacağız. Ve objenin mutlaka bir RigidBody komponenti olmasını istiyorum. Böyle durumlarda RequireComponent kullanabiliyoruz. Yani herhangi bir komponentin omasını mecburi hale getirebiliriz. Onu da bu şekilde yapacağız : [RequireComponent(typeof(Rigidbody))] Bu kodu Sınıf tanımlamasından önce yazıyoruz. Ardından sınıfımızı tanımlıyoruz ve Unity üzerinden atamaları yapabilmek için kullandığımız [SerilaizeField] ve public değerleri istediğimiz türde değişkenler ve sınıfları tanımlıyoruz. Burada ekstra olarak bu değişkenler için bir Header veya bir Tooltip yani tanımlama girebiliriz. Bu sayede birisi Unity editör üzerinde faresini o değişkenin üzerine getirirse değişkenin ne işe yaradığını anlayabilir. Sağda görmüş olduğunuz gibi. Ardından da diğer gerekli değişkenleri tanımlıyoruz.



Sırada Awake Fonksiyonu var. Bu fonksiyon Start ile aynı mantık da çalışıyor ancak Start çalışmadan önce çalışıyor. Burada aracımızın daha stabil olabilmesi için aracımızın ağırlık merkezini 30 santim aşağıya çekiyoruz : rb.centerOfMass = new Vector3(0f, -0.3f, 0f);

Input.GetAxis("Horizontal") 'in bize verdiği değer biz a tuşuna bastığımız zaman 0 dan başlayarak 1 e doğru ilerleyen bir float değer. Bu ilerlemeyi oldukça hızlı yapıyor. Bizim bunu kullanmamız ve Input.GetKey kullanmamızın amacı da basar basmaz tüm gücü uygulamak yerine kısa bir süre içerisinde tüm gücü uygulayabilecek hale gelmesini sağlamak.

FixedUpdate in içerisinde de arabamızın hareket işlemlerini gerçekleştiriyoruz. Bu işlemleri direkt gerçekleştirmek yeriene ilk olarak şu anda aracın sahip olduğu değerler hesaplanıyor. currentSpeed, maxSpeed, desiredMotorTorque gibi değerler. Ardından firene basılıyor mu diye bir kontrol yapıyoruz ve peşinden de Bunları uygulayacak fonksiyonları yazıp çağırıyoruz. Bu fonksyionlar da :

ApplySteering() : Aracımızın direksiyonun açısını hesaplıyor. Bu açıyı horizontalInput ile yani A ve D tuşlarını kullanarak ve bu açının ne kadar hızlı değişmesini ayarladığımız activeStats.steerSpeed değişkenleri ile ayarlıyor.

ApplyBrakes(brakeTorque) : Bütün tekerleklere whellCollider ile hazır olarak gelen Wheel.brakeTorque işlemini uyguluyor.

ApplyDriveTorque(desiredMotorTorque) : ilk olarak bütün tekerleklerin torkunu 0 yapıyor. Ardından da olması gerektiği gibi atamalarını gerçekleştiriyor. Burada activeStats.driveType kullanıyor. Bunun amacı da araba önden çekişli mi, arkadan itişli mi, 4x4 mü ? Buna göre hangi tekerleklerin döneceğini ayarlıyor ve torklarını aktif tekerlek sayısına göre katsayılar ile çarpıyor.

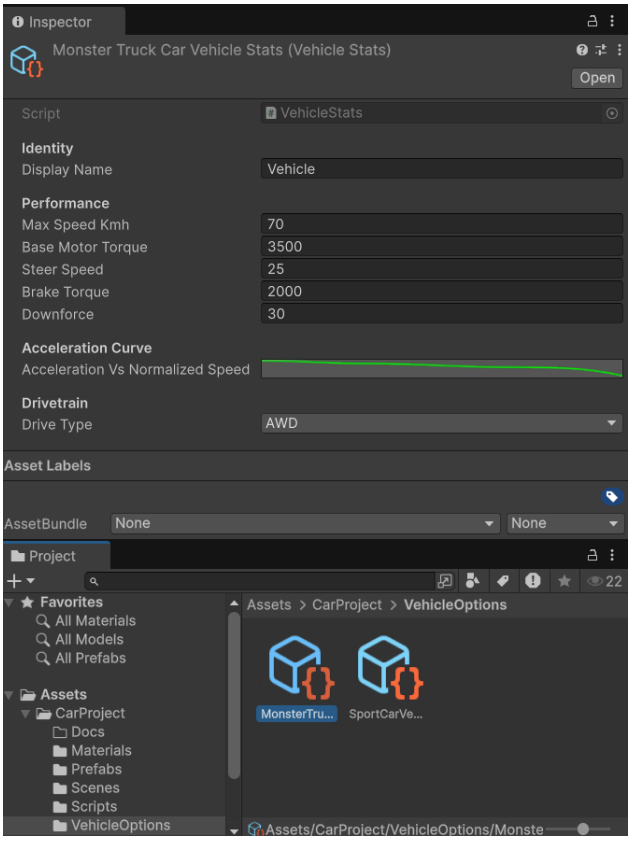
ApplyDownforce(currentSpeed) : Aracımız hızlandıkça aracı geriye doğru iten bir kuvvet simüle ediyoruz bunun amacı da aslında sürtünme gibi bir mantık elde etmek.

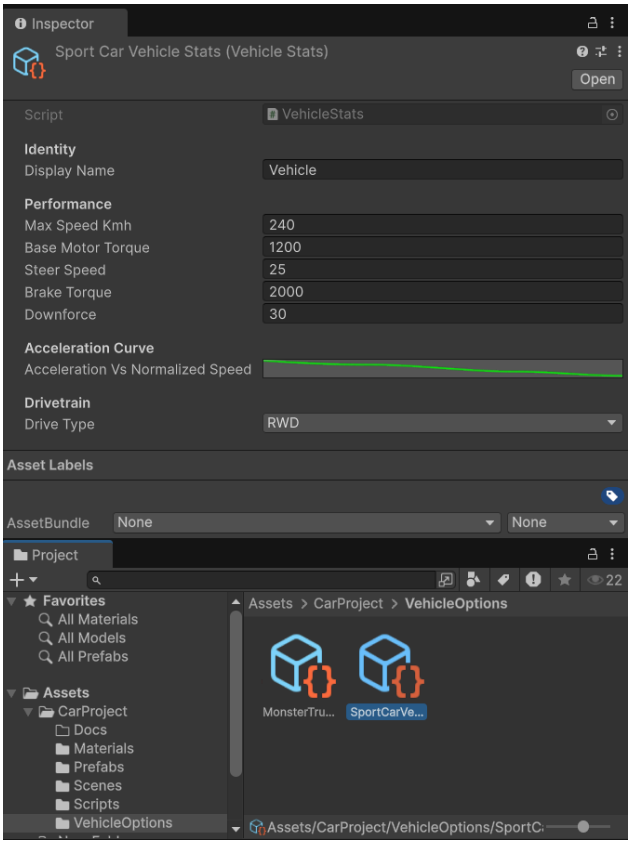
Şeklinde.

2. WhellSC : Bu Scriptin amacı tekerleklerimizin dönemsini ve ön tekerlekler için açıların ayarlanamsını sağlamak. Bunu zaten WheelColliderlarda yaptık. Bu yüzden aslında yapmamız gereken tek işlem Tekerleklerin görsellerindeki açı ve rotasyonların wheelCollider ile aynı olmasını sağlamak. Bunun için de 10 satırlık Basit bir kod yazıyoruz.

```
using UnityEngine;

Unity Betiği (4 varlık başvurusu) | 0 başvuru
public class WhellSC : MonoBehaviour
{
    public WheelCollider whellCollider;
    public Transform whellMesh;

    Unity İletisi | 0 başvuru
    void Update()
    {
        if (whellCollider == null || whellMesh == null) return;

        Vector3 pos;
        Quaternion rot;
        whellCollider.GetWorldPose(out pos, out rot);
        whellMesh.position = pos;
        whellMesh.rotation = rot;
    }
}
```

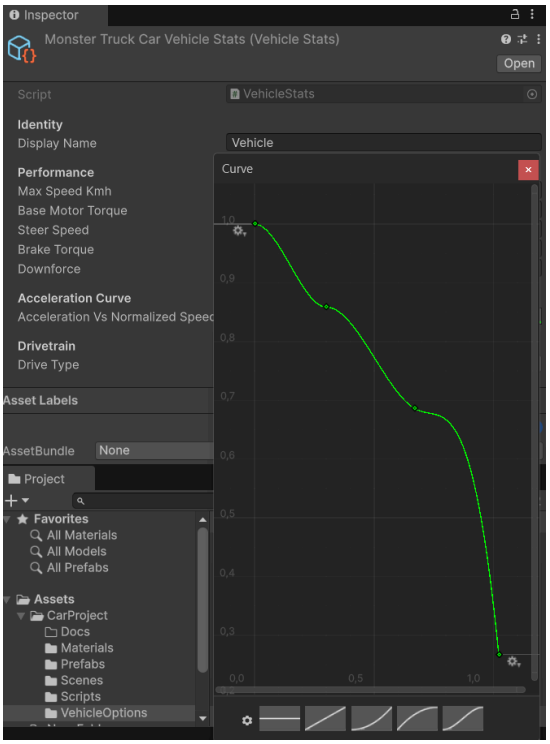
3. VehicleStats : Bu Scripti bir ScriptableObject olarak ayarlayacağız. Bu yüzden ilk olarak ScriptableObject nedir buna değineceğim. ScriptableObject, Objesini oluşturabileceğimiz Script demektir. Peki ne işe yararlar ? Aslında yaptıkları şey özellikler saklamaktır. Bizim burada kullanım amacımız da birden fazla araç için farklı özellik objeleri oluşturmak. Örneğin maksimum hızı fazla ancak torku az olduğu için yavaş hızlanan bir Spor araba. Yada torku fazla ancak maksimum hızı düşük olan bir Ralli arabası özellikleri ayarlama. Kafanızda daha iyi canlanması için 2 adet fotoğraf ekledim. Görmüş olduğunuz Soldaki fotoğraf Master Truck Car Vehicle Stats. Bu araç özelliğinin max hızı 70 ancak torku 3500. Sport Car Vehicle Stats da ise maksimum hızı 240 iken torku 1200. Bu da yavaş hızlanan ancak maksimum hızı fazla bir spor araba örneği.





Tabiki buradaki değerleri verirken matematiksel hesaplamaları oldukça göz önünde bulundurmamız gerekiyor. Örneğin maksimum hızı 240 diye 240 km/h hızına kadar uluşmak zorunda değil. Eğer DownForce değerini yüksek verirseniz aracın torku düşün olduğu için rüzgarın sürtünme kuvvetine dayanamayarak 240 a hiçbir zaman ulaşamıyor olabilir. Bu ve benzeri işlemler için burada ayarlayacağımız değerler oldukça kritik. Buradaki SteerSpeed değeri bir A ve D ye bastığımız zaman ön tekerleklerin ne kadar hızlı sağa sola döneceğini belirliyor. BrakeTorque de frenleme gücümüzü belirliyor. Drive Type da AWD 4x4, RWD arkadan itişli ve FWD ise önden çekişli anlamına geliyor. Bu ScriptableObject kodundan bir obje oluşturmak için project menüsünde herhangi bir yere sağ tıklayarak Racing bölümünden vehicle Stats a geliyoruz. Bu yolu belirleyen şey de bizim kodda yazdığımız : [CreateAssetMenu(fileName = "VehicleStats", menuName = "Racing/Vehicle Stats", order = 0)] kısmı. Kodun yazılması kısmında ise diğer kodlarımızdan farklı olarak MonoBehaviour sınıfını değil de ScriptableObject sınıfını inherit ediyoruz. Ardından da değişken ve fonksiyon tanımlama işlemleri yapabiliyoruz.

*Karışık kısım (opsiyonel):
accelerationVsNormalizedSpeed diye bir değerimiz var ve bu bir grafik aslına bakarsanız. Eğer grafiğe tıklarsam sağda görmüş olduğunuz gibi bir ekran karşıma çıkıyor. Burada bu şekilde bir şey kullanmamızın sebebi aracımız hızlandıkça torktan düşmesi. Biz bunu aracın hızı 0 ken torku maksimumdur şeklinde yaptık ancak bu da tam doğru sayılmaz. Gerçek araçlarda tork değeri devir düşükken ve yüksek iken azdır. En çok torkun olduğu yer maksimum motor devrini 3/2 si gibi bir değerdir. örneğin maksimum motor devri 13 bin devir olan bir motorun maksimum tork verecei nokta muhtemelen 7-8 bin devirlerdir. Biz bunu burada sadece maksimum hız ile ilişkilendirdik ancak bunu her viteste motor devrine göre ayarlamak daha doğru bir yaklaşım olacaktır.



4. CameraSwitcher : Bu Scriptin yapacağı temel şey şu. Biden fazla kameramız olduğu için kullanılan kamera objesinin aktifliğini açacak diğer tüm kameraların aktifliğini kapatacak. Bunu yapması için de bir tuş belirleyeceğiz. Şu anlık C dedik bu tuşa. bir sayacımız var ve bu sayaç her C ye bastığımızda artıyor. Sayacımız kamera sayısını geçtiği zaman hata almamız için activeIndex = (activeIndex + 1) % cameras.Length şeklinde bir atama kullandık. Bu sayede sürekli döngü içinde kalabiliyor. Tüm kameraları kapatıp sadece gerekli kamerayı açan kodumuz da burası :

```
void ApplyActive()
{
    if (cameras == null) return;
    for (int i = 0; i < cameras.Length; i++)
    {
        if (cameras[i] != null) cameras[i].enabled = (i == activeIndex);
    }
}
```