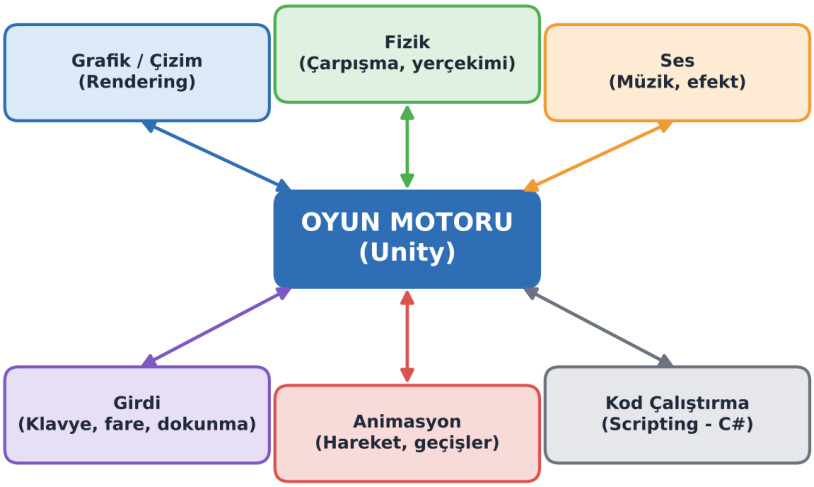
SİNOP ÜNİVERSİTESİ

Meslek Yüksekokulu | Bilgisayar Programcılığı

OYUN PROGRAMLAMA I

Vize Sınavı Ders Notları



Şekil 1: Bir oyun motorunun temel bileşenleri

Hazırlayan: Öğr. Gör. İsmail Umut DURAN

2026 Güz Dönemi

Konu Başlıkları

Bu ders notu vize sınavı kapsamındaki yedi konuyu içerir. Notlarda kod yazılmamıştır; kavramlar günlük hayattan benzetimlerle ve şemalarla açıklanmıştır. Kodlama konusuna geçtiğimizde buradaki kavramlara tekrar döneceğiz.

Bölüm	Konu
1	Oyun motoru nedir, ne amaçla kullanılır?
2	Unity Hub ve Unity kurulumu
3	2D oyun türleri
4	Sprite, Tileset, Tilemap, Tile Palette; Collider, Rigidbody ve Trigger
5	Sprite animasyonları: Animation penceresi, parametreler, geçişler
6	Sprite Editor ve Sprite Slicer (bölme)
7	Nesne tabanlı programlama ve oyun motoru ilişkisi (C# ve Unity)
8	Vize öncesi hızlı kontrol listesi ve terim sözlüğü

> **İPUCU | Nasıl çalışmalı?**
> Her bölümün başındaki hedefi oku, benzetimleri kendi cümlelerinle anlatmaya çalış, bölüm sonundaki tekrar sorularını kapalı kitap cevaplamayı dene.
>
> Unity bilgisayarında kuruluysa her bölümdeki adımları uygulayarak tekrar et; vizede ekrandaki pencerelerin ve menülerin yerini bilmek işini kolaylaştırır.

1. Oyun Motoru Nedir?

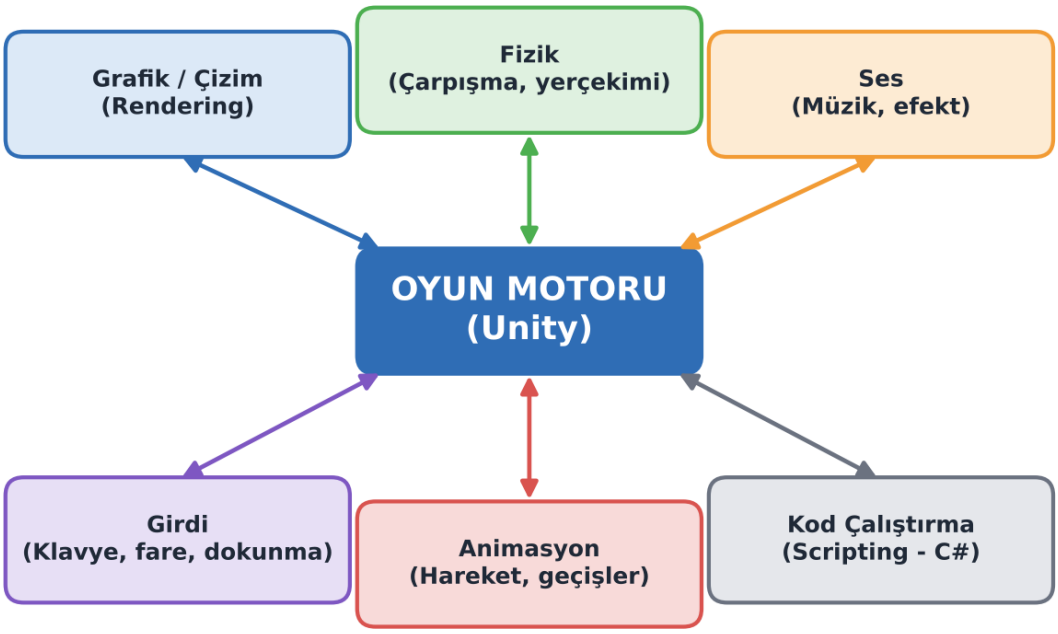
> **BİLGİ | Bu bölümün hedefi**
> Oyun motorunun ne olduğunu, hangi parçalardan oluştuğunu ve neden kullanıldığını açıklayabilmek.

1.1 Tanım

Oyun motoru (game engine), oyun yapımında tekrar tekrar ihtiyaç duyulan temel işlerin (ekrana çizim, fizik, ses, kullanıcı girdisi, animasyon, kod çalıştırma) hazır hâlde sunulduğu yazılım çatısıdır. Oyun geliştirici bu işleri sıfırdan yazmak yerine motorun hazır araçlarını kullanır ve zamanını oyunun kendisine, yani kurallarına, görsellerine ve hikâyesine ayırır.

> **BENZETİM | Hazır mutfak**
> Bir restoran açmak istediğini düşün. Ocağı, fırını, buzdolabını ve tezgâhı kendin üretmek zorunda olsaydın yemek yapmaya hiç başlayamazdın. Donanımlı bir mutfak kiraladığında ise sadece menüyü (yani oyunu) tasarlarsın.
>
> **Oyun motoru = donanımlı mutfak, oyun = senin hazırladığın menü.**

1.2 Oyun motorunun temel bileşenleri



Şekil 1: Oyun motorunun ana bileşenleri birbiriyle sürekli haberleşir

Bileşen	Ne yapar?	2D oyundan örnek
Grafik (Rendering)	Görselleri ekrana çizer, kamerayı ve katmanları yönetir	Karakterin ve haritanın ekranda görünmesi

Bileşen	Ne yapar?	2D oyundan örnek
Fizik	Yerçekimi, çarpışma ve hareket kurallarını hesaplar	Karakterin zemine basması, duvara çarpması
Ses	Müzik ve ses efektlerini çalar	Zıplama sesi, arka plan müziği
Girdi (Input)	Klavye, fare, joystick, dokunmatik ekran girişlerini okur	Ok tuşuna basınca karakterin yürümesi
Animasyon	Nesnelerin hareket ve durum değişimlerini oynatır	Yürüme, bekleme, zıplama animasyonları
Kod çalıştırma (Scripting)	Geliştiricinin yazdığı oyun kurallarını çalıştırır	Puanın artması, düşmanın davranışı

1.3 Oyun motoru neden kullanılır?

- **Zaman kazandırır:** Fizik, çizim, ses gibi altyapı işleri hazırdır.
- **Görsel editör sunar:** Sahneyi kod yazmadan fareyle kurabilirsin.
- **Çoklu platform:** Aynı proje Windows, Android, iOS, web ve konsol için derlenebilir.
- **Hazır içerik ve topluluk:** Asset Store, hazır paketler, eğitimler ve forumlar vardır.
- **Ekip çalışmasına uygundur:** Programcı, grafiker ve tasarımcı aynı projede çalışabilir.

1.4 Popüler oyun motorları

Motor	Kod dili	Öne çıkan yönü
Unity	C#	Hem 2D hem 3D; mobil oyunlarda çok yaygın; geniş topluluk ve Asset Store
Unreal Engine	C++ ve Blueprint (görsel kodlama)	Yüksek görsel kalite; büyük 3D projelerde tercih edilir
Godot	GDScript, C#	Açık kaynak, hafif; 2D desteği güçlü
GameMaker	GML	2D oyunlara odaklı, öğrenmesi kolay
Construct	Kod yazmadan (olay tabanlı)	Tarayıcı tabanlı, hızlı prototip için uygun

> **BİLGİ | Motor ile oyun aynı şey değildir**
> Motor bir **araçtır**, oyun ise o araçla yapılmış **üründür**. Aynı motorla bambaşka oyunlar yapılabilir; aynı oyun farklı motorlarda da yapılabilir.

1.5 Bu derste neden Unity?

- Kullandığı C# dili, nesne tabanlı programlamayı öğrenmek için çok uygundur.

- 2D ve 3D oyunlar aynı programda yapılabilir; sonraki derslerde (3D modelleme, artırılmış gerçeklik) aynı araca devam ederiz.
- Sektörde, özellikle mobil oyun ve simülasyon alanında yaygındır; iş ilanlarında sık aranır.
- Öğrenciler için ücretsiz kişisel kullanım seçeneği vardır (belirli bir gelir sınırına kadar).

Tekrar Soruları

1. Oyun motoru nedir? Kendi cümlelerinle ve bir benzetim kullanarak açıkla.
2. Oyun motorunun altı temel bileşenini say ve her birine 2D oyundan örnek ver.
3. Oyun motoru kullanmanın üç faydasını yaz.
4. Unity ile Unreal Engine arasındaki iki farkı belirt.

2. Unity Hub ve Unity Kurulumu

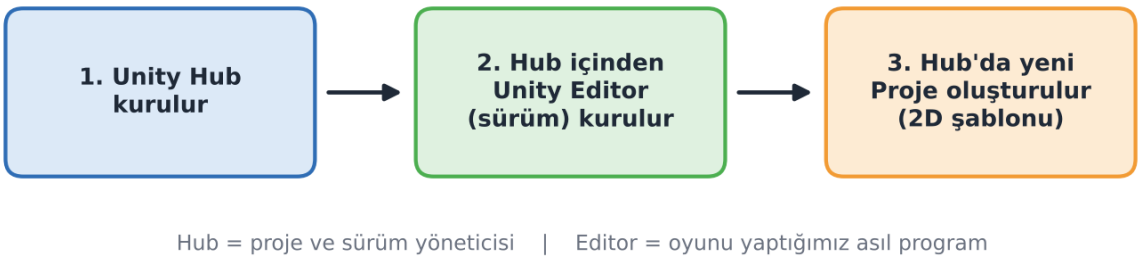
> **BİLGİ | Bu bölümün hedefi**
> Unity Hub ile Unity Editor arasındaki farkı anlamak, kurulumu yapabilmek, ilk 2D projeyi açmak ve editör arayüzünü tanımak.

2.1 Unity Hub nedir?

Unity Hub, Unity'nin yönetim programıdır. Kendisi oyun yapılan yer değildir; hangi projelerin ve hangi Unity sürümlerinin bilgisayarda kurulu olduğunu yönetir, yeni proje açmamızı sağlar. Oyunu geliştirdiğimiz asıl program **Unity Editor**'dür.

> **BENZETİM | Kütüphane görevlisi**
> Unity Hub, kütüphane görevlisi gibidir: hangi kitaplar (projeler) var, hangi baskılar (Editor sürümleri) rafta, onları bilir ve istediğini sana verir. Okuma (oyun yapma) işi ise masada, yani Editor'de yapılır.

2.2 Kurulum adımları



Şekil 2: Kurulum sırası: Hub, Editor, proje

1. Unity'nin resmî sitesinden **Unity Hub**'ı indir ve kur.
2. Hub'ı aç; **Unity ID** hesabınla giriş yap (hesabın yoksa ücretsiz oluşturabilirsin).
3. Lisans olarak ücretsiz **Personal** lisansı etkinleştir.
4. Sol menüden **Installs** (Kurulumlar) sekmesine gir, **Install Editor**'e tıkla.
5. Önerilen **LTS** (uzun süreli destek) sürümünü seç.
6. Gerekli modülleri işaretle (aşağıdaki tabloya bak) ve kurulumu başlat. İndirme birkaç GB olabilir, sabırlı ol.
7. **Projects** sekmesine dön, **New Project**'e tıkla, **2D** şablonunu seç, proje adını ve konumunu belirleyip **Create**'e bas.

Modül	Ne işe yarar?
Windows Build Support	Oyunu Windows bilgisayarlar için derlemek (genellikle varsayılan gelir)
Android Build Support	Oyunu Android telefonlar için derlemek (mobil oyun yapacaksan)
WebGL Build Support	Oyunu tarayıcıda çalışır hâle getirmek
Visual Studio Community	C# kodlarını yazacağımız kod editörü

DİKKAT | Sürüm uyumu

Bir proje hangi Editor sürümüyle oluşturulduysa aynı sürümle açılması en sağlıklısıdır. Okuldaki bilgisayarlarla evdeki bilgisayarda farklı sürüm kullanırsan Unity sürüm yükseltme uyarısı verir. Projeyi taşımadan önce sürüm numarasını not al.

2.3 2D şablonu

Yeni proje açarken karşına şablonlar çıkar. 2D oyun için **2D** adı geçen şablon seçilir (Unity sürümüne göre "2D (Built-In Render Pipeline)" veya "Universal 2D" şeklinde görünebilir). Bu şablon kamerayı 2D'ye uygun ayarlar ve 2D çalışma araçlarını (Sprite Editor, Tilemap, 2D fizik) hazır getirir.

2.4 Unity Editor arayüzü

Pencere	Görevi	Benzetim
Scene	Sahneyi kurduğumuz, nesneleri yerleştirdiğimiz çalışma alanı	Film seti
Game	Oyunun kameradan nasıl göründüğü; Play ile test edilir	Kamera görüntüsü
Hierarchy	Sahnedeki tüm nesnelerin listesi	Oyuncu listesi
Project	Projedeki dosyalar: görseller, sesler, scriptler	Malzeme deposu
Inspector	Seçili nesnenin özellikleri ve bileşenleri	Nesnenin kimlik kartı
Console	Hata ve uyarı mesajları	Hata tutanağı

İPUCU | Sahne düzenini hatırla

Scene penceresinde değişiklik yapmak oyunu çalıştırmaz; test etmek için **Play** düğmesine basılır. **Play modunda yapılan değişiklikler oyundan çıkınca kaybolur.** Bu yüzden Play açıkken ayar yapma, önce durdur.

Sahneyi kaydetmek için **Ctrl+S** kullan. Sadece bir sahneyi değil projeyi de düzenli kaydet.

2.5 Projeyi başka bilgisayara taşıma

Unity proje klasörü çok yer kaplayabilir. Başka bir bilgisayara taşırken sadece **Assets**, **Packages** ve **ProjectSettings** klasörlerini kopyalamak yeterlidir. **Library** klasörü Unity tarafından yeniden üretilir, taşımana gerek yoktur.

2.6 Sık karşılaşılan sorunlar

Sorun	Olası çözüm
Hub'da kurulu Editor görünmüyor	Installs sekmesinde Locate ile Editor'ün kurulu olduğu klasörü göster
Proje farklı sürümde açılıyor uyarısı	Aynı sürümü kur veya yükseltmeyi kabul et (önce projenin yedeğini al)
Kurulum çok yavaş	Modülleri gereğinden fazla seçme; ağ bağlantısını ve disk alanını kontrol et
Script açılmıyor / kod editörü yok	Visual Studio Community modülünü kur, Edit > Preferences > External Tools'ta seç

Tekrar Soruları

1. Unity Hub ile Unity Editor arasındaki fark nedir?
2. Yeni bir 2D proje açma adımlarını sırayla yaz.
3. LTS sürüm ne demektir ve neden tercih edilir?
4. Scene, Game, Hierarchy, Project ve Inspector pencerelerinin görevlerini yaz.
5. Play modunda yapılan değişikliklere ne olur?

3. 2D Oyun Türleri

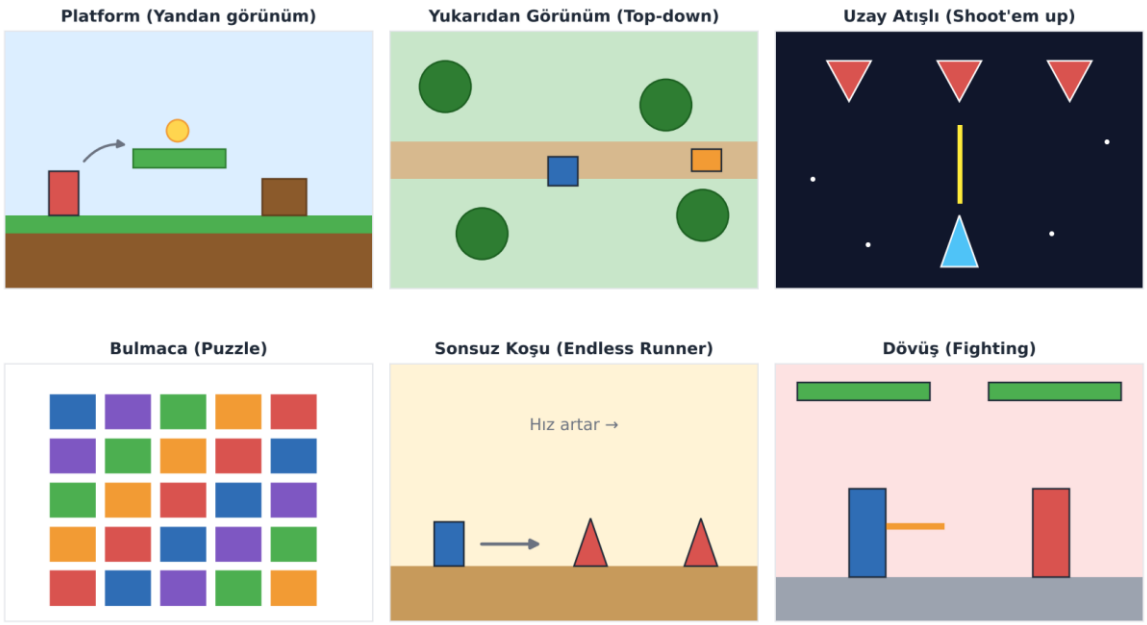
> **BİLGİ | Bu bölümün hedefi**
> 2D oyunun ne olduğunu, başlıca 2D oyun türlerini ve her türün temel özelliklerini ayırt edebilmek.

3.1 2D oyun nedir?

2D (iki boyutlu) oyunlarda dünya yalnızca **genişlik (X)** ve **yükseklik (Y)** ekseninde yer alır; derinlik (Z) kavramı yoktur. Görseller düz resimlerdir (sprite) ve kamera dünyaya karşıdan bakar.

Özellik	2D oyun	3D oyun
Eksen	X ve Y	X, Y ve Z
Görsel yapı	Düz resimler (sprite)	Üç boyutlu modeller
Yapım maliyeti	Genellikle daha düşük	Genellikle daha yüksek
Donanım ihtiyacı	Düşük	Daha yüksek
Örnek	Mario, Celeste	Minecraft, GTA

3.2 Başlıca 2D oyun türleri



Şekil 3: Altı temel 2D oyun türünün sadeleştirilmiş görünümü

Tür	Kamera / bakış	Temel özellik	Örnek oyun
Platform	Yandan	Zıplama, platformlar, engeller, toplanabilir nesneler	Super Mario Bros, Celeste
Yukarıdan görünüm (Top-down)	Kuşbakışı	Dört yöne hareket, keşif, görev; RPG ve macera oyunlarında yaygın	The Legend of Zelda (klasik), Stardew Valley

Tür	Kamera / bakış	Temel özellik	Örnek oyun
Uzay atışlı (Shoot'em up)	Genelde yukarıdan, ekran kayar	Sürekli ateş etme, dalga dalga düşman, mermiden kaçma	Galaga, Space Invaders
Bulmaca (Puzzle)	Sabit tahta	Eşleştirme, mantık, zihinsel çözüm	Tetris, Candy Crush
Sonsuz koşu (Endless Runner)	Yandan, otomatik kayar	Karakter durmadan koşar, oyuncu zıplar veya eğilir; hız giderek artar	Chrome Dino, Jetpack Joyride
Dövüş (Fighting)	Yandan, sabit arena	İki karakter karşılıklı, can çubuğu, kombolar	Street Fighter II, Mortal Kombat (ilk oyunlar)

Diğer türler

- **Metroidvania:** Büyük bağlantılı harita, kazanılan yeteneklerle yeni alanlara erişme (Hollow Knight).
- **Kule savunma (Tower Defense):** Yol boyunca gelen düşmanlara karşı savunma yapıları kurma (Plants vs. Zombies).
- **Görsel roman (Visual Novel):** Hikâye ve seçimlere dayalı, az hareketli yapı.
- **Roguelike:** Rastgele üretilen haritalar, ölünce baştan başlama.

3.3 Tür seçimi ve Unity'de kullanılacak araçlar

Tür	Unity'de en çok kullanılan özellikler
Platform	Tilemap (zemin), Rigidbody 2D, Collider, sprite animasyonu
Top-down	Tilemap, Collider, dört yönlü animasyon, Trigger (kapı, eşya)
Shoot'em up	Hareketli arka plan, mermi ve düşman nesneleri, Trigger veya Collider çarpışması
Puzzle	Izgara düzeni, nesne seçme ve yer değiştirme, puan sistemi
Endless Runner	Sürekli kayan zemin, rastgele engel üretimi, Collider
Dövüş	Animator (çok sayıda durum), Collider ile vuruş alanları, can sistemi

> **İPUCU | Proje fikri seçerken**
> İlk oyununu küçük tut. Tek ekranlık bir platform oyunu veya basit bir top-down keşif oyunu, bu dersin araçlarıyla (Tilemap, Collider, Animator) yapılabilecek en uygun başlangıç projeleridir.

Tekrar Soruları

1. 2D oyun ile 3D oyun arasındaki üç farkı yaz.
2. Platform ve top-down oyunlarının kamera açılarını karşılaştır.
3. Endless Runner türünün temel özelliği nedir? Bir örnek ver.

4. Verilen bir oyun tanımına bakarak türünü belirle: “Karakter yandan görünür, sürekli koşar, engellere takılmamak için zıplar.”

4. Sprite, Tileset, Tilemap, Collider ve Rigidbody

> **BİLGİ | Bu bölümün hedefi**
> 2D haritayı Tile Palette ile oluşturabilmek; Collider, Rigidbody ve Trigger kavramlarını farkıyla birlikte açıklayabilmek.

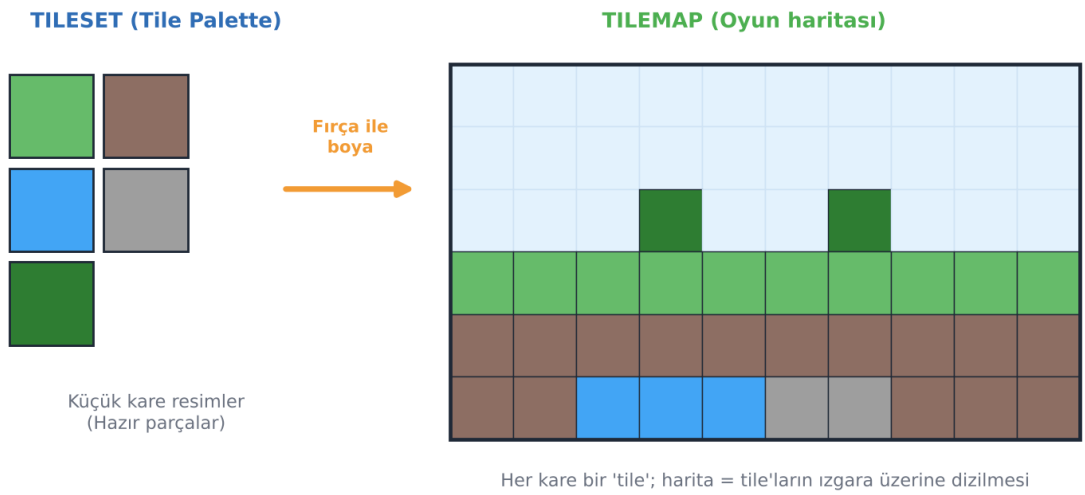
4.1 Sprite nedir?

Sprite, 2D oyunda kullanılan tek parça görsel nesnedir: karakter, düşman, ağaç, altın, düğme gibi. Unity'ye bir PNG resmi eklediğimizde, görselin türünü **Sprite (2D and UI)** olarak ayarlarsak Unity onu sahnede kullanabilir.

Ayar (Inspector)	Anlamı
Texture Type: Sprite (2D and UI)	Resmin 2D oyunda sprite olarak kullanılacağını belirtir
Sprite Mode: Single / Multiple	Tek resim mi, yoksa içinde birçok kare olan bir sprite sheet mi (Bölüm 6)
Pixels Per Unit (PPU)	Kaç pikselin Unity'de 1 birim uzunluk olacağını belirler (varsayılan 100)
Filter Mode: Point (no filter)	Piksel sanatı (pixel art) görsellerin bulanık görünmemesi için kullanılır
Compression: None	Pikselleri sıkıştırmadan korur; piksel sanatında net görüntü verir

4.2 Tile, Tileset ve Tilemap

Büyük bir oyun haritasını tek bir dev resim olarak çizmek yerine, haritayı **küçük kare parçaların** (tile) ızgara üzerine dizilmesiyle oluşturabiliriz. Zemin, çim, su, taş, duvar gibi parçaların her biri bir tile'dır. Tüm bu parçaların toplandığı görsele **tileset** denir.



Şekil 4: Tile'lar paletten seçilir ve Tilemap üzerine boyanır

Kavram	Benzetim	Unity'deki karşılığı
Sprite	Tek bir LEGO parçasının resmi	Tek parça 2D görsel
Tile	Plakaya takılabilen tek LEGO parçası	Bir sprite'tan üretilmiş harita parçası (Tile asset)
Tileset	LEGO parça kutusu	Tile'ların toplandığı görsel; bölünmüş sprite sheet
Tile Palette	Ressamın boya paleti	Boyamak için tile'ların seçildiği pencere
Tilemap	LEGO zemin plakası / tuval	Tile'ların yerleştirildiği harita katmanı
Grid	Kareli defter	Tilemap'in dayandığı kare ızgara (hücre boyutu varsayılan 1x1)

BENZETİM | Kareli defter ve boyama

Kareli bir defterin her karesine istediğin rengi boyadığını düşün. Defter **Grid**, renkli kalemlerin durduğu kutu **Tile Palette**, boyadığın sayfa **Tilemap**, her kare ise bir **tile**'dır. Kareyi silip başka renk boyayabilirsin; haritayı değiştirmek bu kadar kolaydır.

4.3 Tile Palette ile harita yapma

1. Tileset görselini projeye ekle. **Texture Type: Sprite**, **Sprite Mode: Multiple** yap. **Filter Mode: Point** ve **Compression: None** seç, sonra Apply'a bas.
2. **Sprite Editor**'ü açıp görseli kare boyutuna göre **Slice** et (Bölüm 6).
3. Hierarchy'de sağ tıkla: **2D Object > Tilemap > Rectangular**. Unity bir **Grid** nesnesi ve onun içinde bir **Tilemap** oluşturur.

4. **Window > 2D > Tile Palette** ile paleti aç. **Create New Palette** ile yeni palet oluşturup bir klasöre kaydet.
5. Bölünmüş sprite'ları (veya tüm görseli) palet penceresine sürükle. Unity, her tile için bir dosya kaydetmek üzere klasör sorar; bir klasör seç.
6. Paletten bir tile seç, **Paint (fırça)** aracıyla Scene penceresinde ızgaraya tıklayarak haritayı boya.
7. Zemin, arka plan ve süsler için **ayrı Tilemap katmanları** oluştur ve **Order in Layer** değerleriyle hangi katmanın üstte görüneceğini belirle.

Palet aracı	Ne yapar?
Select (Seç)	Haritadaki alanı seçer
Move (Taşı)	Seçili tile'ları taşır
Paint (Boya)	Tek tek tile yerleştirir
Box Fill (Kutu doldur)	Sürüklediğin dikdörtgen alanı doldurur
Flood Fill (Boya kovası)	Bitişik aynı alanı tek tıkla doldurur
Picker (Pipet)	Haritadan bir tile'ı alıp fırça yapar
Erase (Sil)	Tile'ı siler

DİKKAT | Yanlış katmana boyama
Palet penceresinin üstünde **Active Tilemap** seçeneği vardır. Boyama, hangi Tilemap seçiliyse onun üstüne yapılır. Zemini yanlışlıkla arka plan katmanına boyamamak için boyamadan önce bu seçime bak.

4.4 Collider nedir?

Sprite sadece bir **resimdir**; Unity'ye “bu nesne katıdır, içinden geçilemez” demez. Nesnenin fiziksel bedenini tanımlayan bileşene **Collider** (çarpışma alanı) denir. Collider görünmez bir kılıf gibidir; oyunda görünmez ama Scene penceresinde yeşil çizgilerle gösterilir.

BENZETİM | Hayalet ve beden
Sprite, nesnenin **görüntüsüdür**; Collider ise **bedenidir**. Bedeni olmayan bir hayalet duvarın içinden geçer. Collider eklediğinde nesne artık fiziksel olarak vardır ve diğer nesneler ona çarpar.

Collider türü	Şekil	Ne zaman kullanılır?
Box Collider 2D	Dikdörtgen	Kutu, platform, düz zemin, çoğu karakter
Circle Collider 2D	Daire	Top, madeni para, yuvarlak nesneler

Collider türü	Şekil	Ne zaman kullanılır?
Capsule Collider 2D	Kapsül	İnsan veya hayvan karakterler (köşe takılmalarını azaltır)
Polygon Collider 2D	Sprite'ın şeklini izleyen çokgen	Düzensiz şekilli nesneler
Edge Collider 2D	Çizgi	Eğimli yüzeyler, ince sınırlar
Tilemap Collider 2D	Tile'ların şeklini izler	Tilemap ile yapılan zemin ve duvarlar

Tilemap ile yaptığın zeminin üstünde karakterin yürüyebilmesi için Tilemap nesnesine **Tilemap Collider 2D** bileşeni eklenir (Inspector > Add Component).

4.5 Rigidbody 2D nedir?

Rigidbody 2D, nesneyi Unity'nin fizik motoruna dâhil eden bileşendir. Rigidbody olan nesne yerçekiminden etkilenir, kuvvet alabilir, itilebilir ve çarpışmalara tepki verir.

> **BENZETİM | Ağırlık iksiri**
> Sahnede asılı duran bir balon gibi bir nesne düşün. Ona Rigidbody 2D eklemek, ona "ağırlık" kazandırmak gibidir: artık yere düşer, zıplar, savrulur.

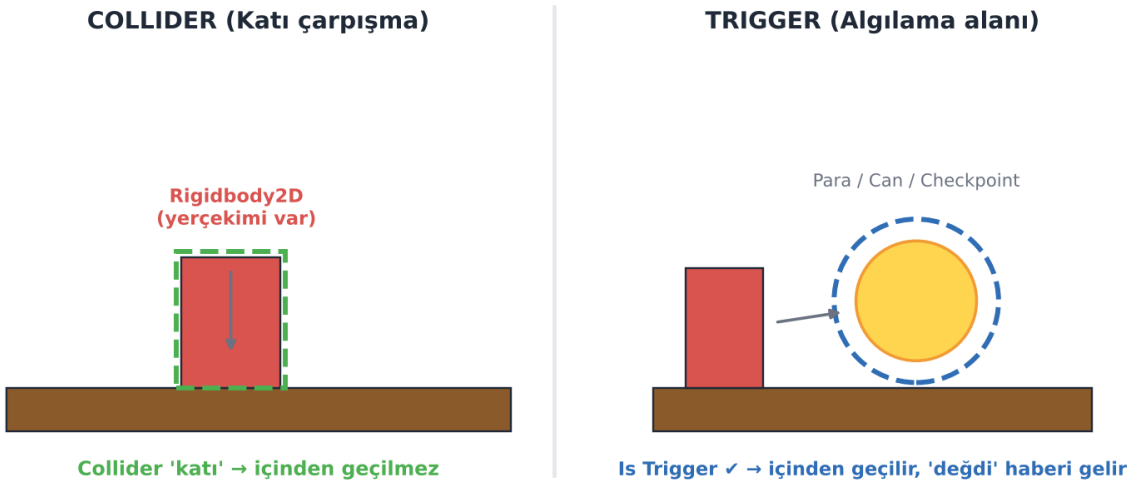
Body Type	Özellik	Örnek
Dynamic	Yerçekimi ve kuvvetlerden tam etkilenir	Oyuncu karakteri, düşen kutu
Kinematic	Fizikten etkilenmez, kod veya animasyonla hareket ettirilir; diğerlerini iter	Hareketli platform, asansör
Static	Hiç hareket etmez	Sabit zemin, duvar

Önemli ayar	Ne işe yarar?
Mass (Kütle)	Nesnenin ağırlığı; diğerleriyle çarpışmada ne kadar itileceğini etkiler
Gravity Scale	Yerçekiminin şiddeti; 0 ise nesne düşmez, 2 ise iki kat hızlı düşer
Linear Damping / Drag	Hareketin yavaşlamasını sağlayan direnç
Constraints > Freeze Rotation Z	Karakterin çarpınca dönmesini (devrilmesini) engeller

> **DİKKAT | Çarpışma için iki şart**
> İki nesnenin birbirine çarpabilmesi için **ikisinde de Collider 2D** bulunmalı ve **en az birinde Rigidbody 2D** olmalıdır. Karakterde Collider yoksa zeminden düşer; zeminde Collider yoksa karakter içinden geçip düşer.

4.6 Trigger (tetikleyici) nedir?

Collider bileşeninde **Is Trigger** kutusu işaretlenirse nesne katı olmaktan çıkar: diğer nesneler içinden geçebilir, fakat Unity "bir nesne bu alana girdi" bilgisini bize haber verir. Böylece alanı bir **algılama sensörü** olarak kullanırız.



Şekil 5: Collider katı bir engeldir, Trigger ise bir algılama alanıdır

Özellik	Collider (Is Trigger kapalı)	Trigger (Is Trigger açık)
Fiziksel engel mi?	Evet, içinden geçilmez	Hayır, içinden geçilir
Çarpışma tepkisi (itme, durma)	Var	Yok
Çarpışma/giriş haberi	Çarpışma olayı olarak gelir	Alana giriş, kalış, çıkış olayı olarak gelir
Kullanım yeri	Zemin, duvar, platform, düşmanla temas	Altın, kalp, checkpoint, tuzak alanı, düşmanın görüş alanı

BENZETİM | Duvar ve fotosel

Collider = duvar: yürümeye çalışırsan çarpar, geçemezsin. **Trigger = marketin otomatik kapısındaki fotosel**: içinden geçersin, ama sensör seni görür ve kapıyı açar. Yani geçişe izin verir ama haberdar olur.

DİKKAT | Zemin ve Trigger

Rigidbody 2D olan bir nesnenin Collider'ı Trigger yapılırsa nesne artık zemine çarpmaz ve **yerçekimiyle aşağı düşer**. Karakterin ana Collider'ını Trigger yapma; algılama için ayrı, ikinci bir Collider ekle.

Kodlamaya geçince bu kavramlar nasıl kullanılacak?

İleride kodlamaya geçtiğimizde Unity, çarpışma ve tetiklenme anlarında kodumuza haber verecek. Örneğin karakter altına değdiğinde puanı artırmak, tuzağa değdiğinde canı azaltmak için bu haberi kullanacağız. Şimdilik önemli olan, hangi durumda Collider, hangi durumda Trigger seçileceğini bilmektir.

Tekrar Soruları

1. Sprite, tile, tileset, tilemap ve tile palette kavramlarını bir cümleyle tanımla.
2. Tile Palette ile harita yapma adımlarını sırayla yaz.
3. Collider nedir? Sprite ile Collider arasındaki fark nedir?
4. Rigidbody 2D'nin üç Body Type'ını örnekle açıkla.
5. Collider ve Trigger arasındaki farkı üç maddede yaz ve her birine uygun bir oyun örneği ver.
6. Karakter zemine basmak yerine içinden geçip düşüyor. Hangi bileşenlerin eksik olabileceğini yaz.

5. Sprite Animasyonları

> **BİLGİ | Bu bölümün hedefi**
> Sprite karelerinden animasyon oluşturmak, Animator'da durumlar arası geçişi parametrelerle kontrol edebilmek.

5.1 Sprite animasyonu nasıl çalışır?

Sprite animasyonu, aynı karakterin birbirinden az farklı birkaç resminin **hızlıca art arda gösterilmesiyle** hareket illüzyonu yaratmaktır. Çizgi film ile aynı mantıktır: kareler yeterince hızlı değiştiğinde göz bunu hareket olarak algılar.

> **BENZETİM | Çizgi film defteri**
> Defterin köşelerine küçük bir adamın her sayfada biraz farklı pozda çizildiğini düşün. Sayfaları hızlıca çevirdiğinde adam yürür. Her sayfa bir **kare (frame)**, çevirme hızı ise **kare hızıdır (Samples/FPS)**.

5.2 Üç temel kavram

Kavram	Dosya / bileşen	Görevi	Benzetim
Animation Clip	.anim dosyası	Tek bir hareketin kare kare kaydı (Idle, Run, Jump)	Film şeridi
Animator Controller	.controller dosyası	Hareketler (durumlar) arasındaki geçiş kurallarının şeması	Trafik yönlendirme planı / makas düzeneği
Animator	GameObject üzerindeki bileşen	Controller'ı nesnede çalıştırır	Film makinesi

5.3 Animasyon yapma adımları

Başlamadan önce sprite sheet'in **Slice** edilmiş olması gerekir (Bölüm 6).

1. **Window > Animation > Animation** ile Animation penceresini aç (kısayol: Ctrl+6).
2. Hierarchy'de animasyon verilecek nesneyi (örn. Player) seç.
3. Animation penceresinde **Create** düğmesine bas ve klip dosyasına ad ver (örn. Player_Idle). Unity otomatik olarak bir **Animator Controller** oluşturur ve nesneye **Animator** bileşeni ekler.
4. Project penceresinden animasyonun karelerini (sprite'ları) seçip Animation penceresindeki **zaman çizelgesine** sürükle.
5. Kare hızını **Samples** değerinden ayarla. Sprite animasyonlarında genelde 8 ile 12 arası yeterlidir.
6. Animasyonun tekrar tekrar oynaması için Project'te klibi seçip Inspector'da **Loop Time**'ı işaretle.
7. **Play** (önizleme) düğmesiyle animasyonu izle.

8. Her hareket için (Idle, Run, Jump) ayrı klip oluştur. Yeni klip için Animation penceresinin sol üstündeki klip menüsünden **Create New Clip** seçilir.

> **İPUCU | Hızlı yol**
> Project penceresinde animasyonun tüm sprite'larını seçip doğrudan **Scene** penceresine sürüklersen Unity nesneyi, animasyon klibini ve Animator ayarlarını senin yerine otomatik oluşturur. Klibi sonra düzenleyebilirsin.

Animation penceresi öğesi	Ne işe yarar?
Zaman çizelgesi (Timeline)	Karelerin zaman içindeki sırasını gösterir
Keyframe	Çizelgedeki bir kare noktası; o anda hangi sprite'ın görüneceğini belirler
Samples	Saniyedeki kare sayısı (animasyon hızı)
Add Property	Animasyona hangi özelliğin dahil olacağını belirler (örn. Sprite Renderer > Sprite)
Preview / Record	Önizleme ve kayıt modu

5.4 Animator penceresi ve durumlar (State)

Window > Animation > Animator ile Animator penceresini açarız. Burada her animasyon klibi bir **durum (state)** olarak görünür. Karakterin "şu anda hangi hareketi yaptığını" bu durum belirler. Turuncu renkli durum, oyun başladığında ilk çalışan **varsayılan durumdur** (Entry okuyla bağlanır; genelde Idle).

- **Entry:** Animator'ın başlangıç noktası.
- **Any State:** Hangi durumda olursak olalım geçiş yapılabilen özel durum (örneğin ölme animasyonu için kullanışlı).
- **Exit:** Çıkış noktası.
- **Make Transition:** Durumların üzerine sağ tıklayıp geçiş oku çizmek için kullanılır.

5.5 Animator parametreleri

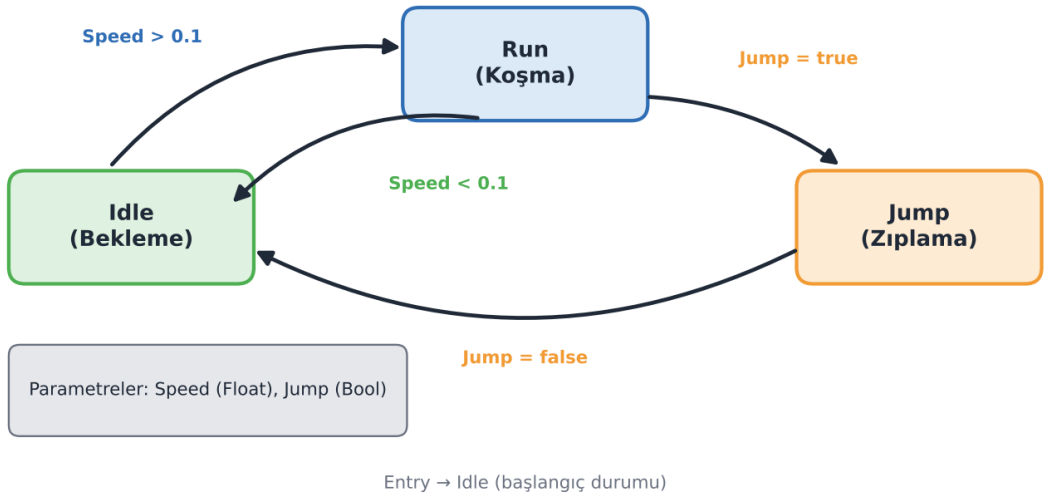
Parametre, Animator'a dışarıdan verilen bir bilgi değeridir. Animator geçişe bu değerlere bakarak karar verir. Parametreler Animator penceresinin sol üstündeki **Parameters** sekmesinden eklenir.

Tür	Tuttuğu değer	Örnek	Benzetim
Float	Ondalıklı sayı	Speed (hız)	Arabanın hız göstergesi
Int	Tam sayı	WeaponLevel (silah seviyesi)	Asansör kat numarası
Bool	Doğru / yanlış	IsGrounded, IsJumping	Lamba düğmesi (açık/kapalı)

Tür	Tuttuğu değer	Örnek	Benzetim
Trigger	Tek seferlik sinyal	Attack, Hurt	Zil: basınca bir kez çalar, sonra kendi söner

5.6 Geçişler (Transition) ve koşullar (Condition)

Transition, bir durumdan diğerine geçiş okudur. **Condition** ise bu geçişin gerçekleşmesi için sağlanması gereken şarttır. Koşul olmazsa geçiş, animasyon bitince kendiliğinden olur.



Şekil 6: Parametreye göre durumlar arası geçiş örneği

> **BENZETİM | Trafik polisi**
> Animator bir trafik polisi, parametreler ise yoldaki sensörlerdir. **Speed** sensörü 0.1'in üstüne çıkınca polis arabayı Idle yolundan Run yoluna yönlendirir. Sensör tekrar düşerse arabayı geri çevirir. Geçiş kuralları polisin elindeki kuralları belirler.

Geçiş oluşturma adımları

1. Animator penceresinde kaynak durum (örn. Idle) üzerine sağ tıkla ve **Make Transition** seç.
2. Çıkan oku hedef duruma (örn. Run) tıklayarak bağla.
3. Oka tıkla; Inspector'da **Conditions** bölümündeki **+** ile koşul ekle.
4. Koşulda parametreyi (örn. Speed), karşılaştırma işlemini (Greater/Less) ve eşik değerini (örn. 0.1) seç.
5. Geri dönüş için ters yönde ikinci bir geçiş çiz ve koşulunu ters ver (örn. Speed Less 0.1).

Transition ayarı	Anlamı	Öneri
Has Exit Time	Geçişin önce mevcut animasyonun bitmesini beklemesi	Anlık tepki (zıplama, koşma) için kapat
Transition Duration	İki animasyon arası harmanlama süresi	Sprite animasyonlarında 0 yap
Conditions	Geçiş şartı (parametre + koşul + değer)	Her geçişe en az bir koşul ekle

İPUCU | Kod yazmadan test etme

Oyunu **Play** ile çalıştır, Animator penceresini açık tut ve **Parameters** bölümünden değerleri elle değiştir (örn. Speed'i 1 yap, Jump'ı işaretle). Geçişlerin doğru çalışıp çalışmadığını kod yazmadan görebilirsin. Kodlamaya geçince bu değerleri script değiştirecek.

5.7 Sık yapılan hatalar

Sorun	Olası neden / çözüm
Animasyon hiç oynamıyor	Nesnede Animator bileşeni yok veya Controller atanmamış
Animasyon bir kez oynayıp duruyor	Klipte Loop Time işaretli değil
Geçiş gecikmeli oluyor	Has Exit Time açık kalmış; kapat, Transition Duration'ı 0 yap
Karakter sürekli aynı animasyonda kalıyor	Koşullar yanlış veya ters; parametre değerlerini kontrol et
Animasyon çok hızlı veya yavaş	Samples değerini ayarla
Sprite'lar titriyor veya zıplıyor	Karelerin Pivot noktaları tutarsız (Bölüm 6)

Tekrar Soruları

1. Animation Clip, Animator Controller ve Animator bileşeni arasındaki farkı açıkla.
2. Sprite animasyonu yapma adımlarını sırayla yaz.
3. Dört Animator parametre türünü ve her birine örnek ver.
4. Idle'dan Run'a geçiş için nasıl bir koşul yazılır? Geri dönüş için koşul nasıl olur?
5. Has Exit Time ne işe yarar? Hangi durumlarda kapatılır?

6. Sprite Editor ve Sprite Slicer

> **BİLGİ | Bu bölümün hedefi**
> Bir sprite sheet'i Sprite Editor'de doğru boyutlarda bölebilmek ve pivot noktasını ayarlayabilmek.

6.1 Sprite sheet nedir?

Bir animasyonun veya bir tile setinin bütün karelerini ayrı ayrı dosyalar hâlinde tutmak yerine, tek bir resim dosyasında yan yana dizeriz. Bu tek resme **sprite sheet** denir. Düzenli olur, dosya sayısı azalır ve oyun daha verimli çalışır.

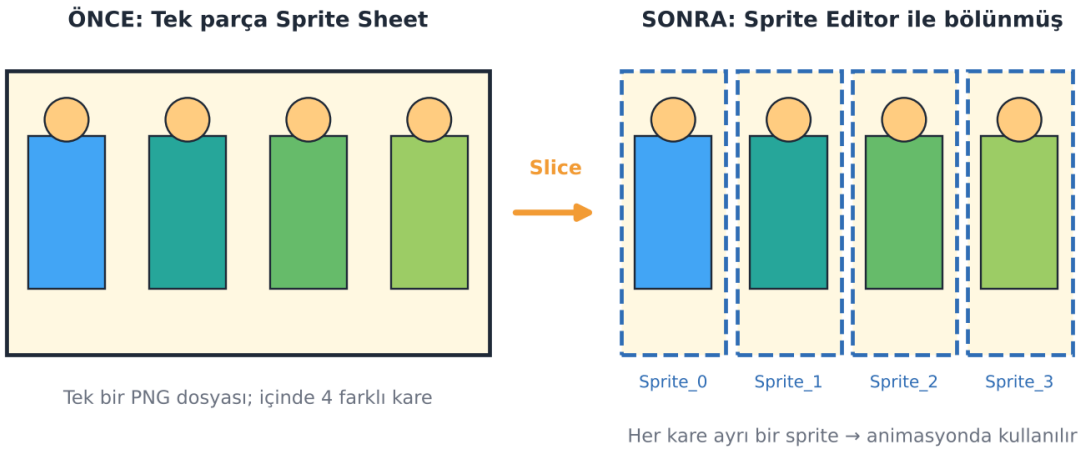
Unity bu resmi önce tek parça olarak görür. Karelerin ayrı ayrı kullanılabilmesi için resmin **bölünmesi (slice)** gerekir. Bu işi **Sprite Editor** yapar.

6.2 Bölmeden önce import ayarları

1. Project penceresinde sprite sheet görselini seç.
2. Inspector'da **Texture Type: Sprite (2D and UI)** olduğundan emin ol.
3. **Sprite Mode: Multiple** yap (içinde birden fazla kare olduğu için).
4. Piksel sanatıysa **Filter Mode: Point (no filter)** ve **Compression: None** seç.
5. **Apply** düğmesine bas.

Sprite Mode	Ne zaman?
Single	Resim tek bir sprite ise (tek bir ağaç, tek bir düğme)
Multiple	Resim içinde birden çok kare/sprite varsa (animasyon sheet'i, tileset)

6.3 Sprite Editor ile bölme (Slice)



Şekil 7: Slice işlemi: tek resim, ayrı ayrı sprite karelerine dönüşür

1. Inspector'da **Sprite Editor** düğmesine tıkla.
2. Açılan pencerenin sol üstünde **Slice** menüsünü aç.
3. **Type** alanından bölme yöntemini seç (aşağıdaki tabloya bak).
4. Gerekirse **Pivot** değerini ayarla.
5. **Slice** düğmesine bas; Unity kareleri çerçevelerle gösterir.
6. Sağ üstteki **Apply** düğmesine basarak işlemi kaydet. Apply'a basmazsan bölme kaybolur.

Slice türü	Nasıl çalışır?	Ne zaman kullanılır?
Automatic	Karelerin arasındaki boş alanlara bakarak otomatik böler	Kareler arası boşluk varsa ve kare boyutları farklıysa
Grid By Cell Size	Verdiğin piksel boyutuna (örn. 32x32) göre ızgaraya böler	Kare boyutunu biliyorsan (en güvenilir yöntem)
Grid By Cell Count	Verdiğin satır ve sütun sayısına böler	Kaç satır/sütun olduğunu biliyorsan

BİLGİ | Hesaplama örneği

Resmin boyutu **256 x 128 piksel**, her kare **32 x 32** ise: yatayda 256 / 32 = **8 sütun**, dikeyde 128 / 32 = **4 satır** olur. Toplam 8 x 4 = **32 sprite** elde edilir.

Grid By Cell Size seçip **Pixel Size** alanına 32 ve 32 yazmak yeterlidir.

6.4 Pivot noktası

Pivot, sprite'ın “tutma noktasıdır”. Nesnenin konumu, dönüşü ve ölçeği bu noktaya göre hesaplanır. Slice sırasında **Pivot** seçeneğiyle merkez (Center), alt (Bottom) gibi bir konum seçilebilir.

BENZETİM | Karton ve raptiye

Bir kartonu duvara raptiyeyle tuttuğunu düşün. Raptiye nereye batarsa karton oraya göre döner. Karakterin pivotu ayaklarda ise karakter ayaklarından zemine basar; pivot karakterin göbeğinde ise zeminle hizalamak zorlaşır.

DİKKAT | Animasyonda pivot tutarlılığı

Bir animasyonun bütün karelerinde pivot aynı noktada olmalıdır. Aksi hâlde animasyon oynarken karakter **titrer veya yerinden zıplar**. Karakterler için genelde **Bottom (alt)** veya **Center (merkez)** seçilir.

6.5 Sık karşılaşılan sorunlar

Sorun	Olası neden / çözüm
Kareler kaymış, kesik görünüyor	Hücre boyutu yanlış girilmiş veya sheet'te kenar boşluğu (Padding / Offset) var
Boş kareler de sprite olmuş	Automatic bölmede boş kareleri tutma seçeneğini kapat veya bu sprite'ları sil
Bölme sonrası değişiklik yok	Apply'a basılmamış
Sprite Editor düğmesi görünmüyor	Sprite Mode Multiple değil veya Texture Type Sprite değil
Görsel bulanık görünüyor	Filter Mode Point (no filter) ve Compression None olarak ayarla

Bölme tamamlandığında Project penceresinde görselin yanındaki ok işaretiyle açılan listede **Sprite_0, Sprite_1, Sprite_2 …** şeklinde ayrı sprite'lar görürsün. Bunları Tile Palette'e (Bölüm 4) veya Animation penceresine (Bölüm 5) sürükleyerek kullanırız.

Tekrar Soruları

1. Sprite sheet nedir ve neden kullanılır?
2. Sprite Mode'ta Single ile Multiple arasındaki fark nedir?
3. 512 x 256 piksellik bir sheet'i 64 x 64 karelere bölersek kaç sprite elde ederiz? Hangi Slice türünü kullanırız?
4. Pivot nedir? Animasyonda pivotların tutarsız olması hangi soruna yol açar?
5. Slice işleminden sonra neden Apply düğmesine basmak gerekir?

7. Nesne Tabanlı Programlama ve Oyun Motoru

BİLGİ | Bu bölümün hedefi

Nesne tabanlı programlamanın temel kavramlarını anlamak; Unity'deki GameObject, bileşen ve script yapısını bu kavramlarla ilişkilendirebilmek. Bu bölümde kod yok; kavramlar benzetimlerle anlatılır.

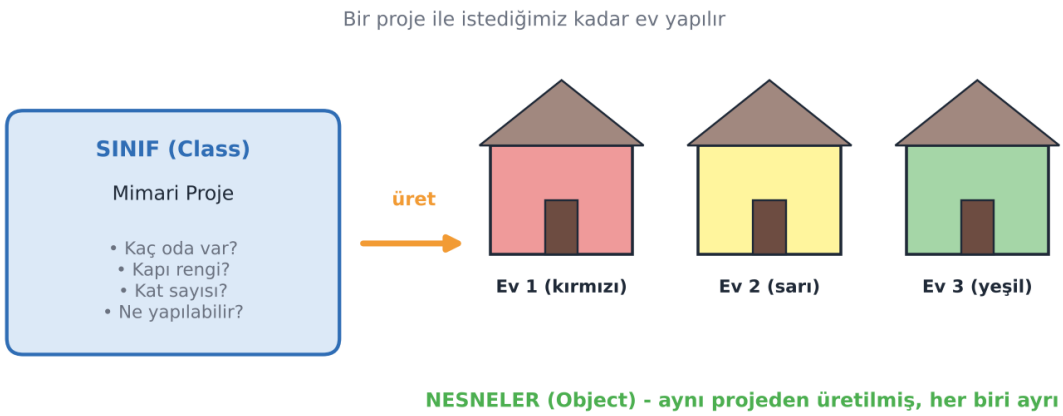
7.1 Neden önce bu kavramlar?

Unity'deki her şey bir **nesnedir**: karakter, düşman, zemin, kamera, ışık, hatta boş bir yönetici. Kodlamaya geçmeden önce "nesne" ve "sınıf" fikrini anlarsak, Unity'nin çalışma mantığı ve ileride yazacağımız kodlar çok daha anlaşılır olur.

7.2 Nesne tabanlı programlama nedir?

Nesne tabanlı programlama (OOP, Object-Oriented Programming), yazılımı gerçek dünyadaki nesneler gibi düşünerek tasarlama yaklaşımıdır. Her nesnenin **özellikleri** (sahip olduğu bilgiler) ve **davranışları** (yapabildiği işler) vardır. C# bu yaklaşımı kullanan bir programlama dilidir.

Sınıf (Class) ve Nesne (Object)



Şekil 8: Sınıf bir tasarım şablonudur; nesneler ondan üretilen gerçek örneklerdir

BENZETİM | Mimari proje ve evler

Bir mimar, evin projesini bir kez çizer: kaç oda olacak, kapı nerede, kaç kat. Bu proje **sınıftır (class)**. Projeye bakarak üç farklı ev yapılabilir: kırmızı, sarı ve yeşil. Her ev ayrı bir **nesnedir (object)**; hepsi aynı projeden doğmuştur ama her birinin rengi, sakini, adresi farklıdır.

Kurabiye kalıbı da aynı fikirdir: tek bir kalıp (sınıf), çok sayıda kurabiye (nesne).

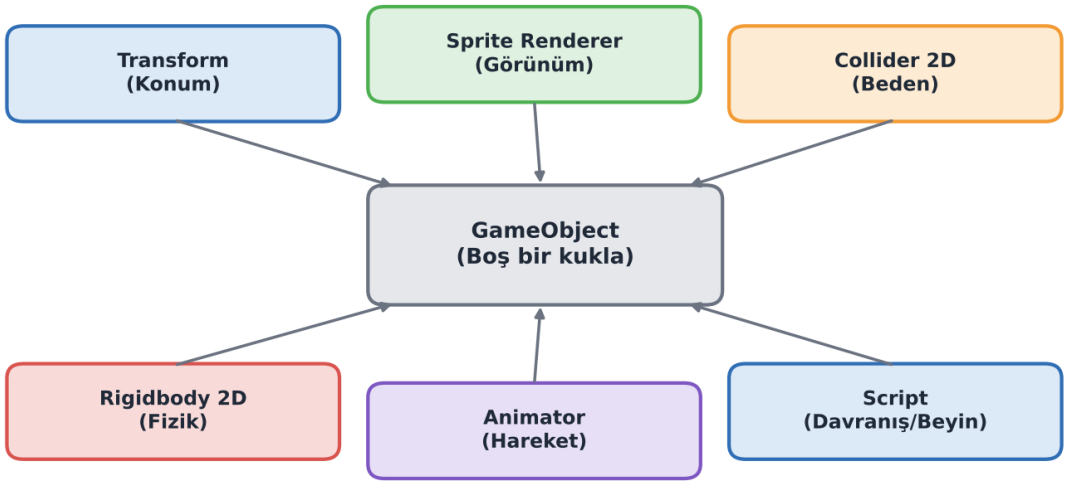
Kavram	Günlük hayat	Oyun örneği
Sınıf (Class)	Araba tasarım şablonu	"Düşman" tasarımı

Kavram	Günlük hayat	Oyun örneği
Nesne (Object)	Ali'nin kırmızı arabası	Sahnedeki 3. zombi
Özellik (Attribute / Field)	Renk, hız, yakıt miktarı	Can, hız, hasar
Davranış (Method)	Hızlan, fren yap, korna çal	Saldır, koş, öl

Aynı "Düşman" sınıfından 5 düşman üretirsek, hepsinin "can" özelliği vardır ama her birinin canı **kendine aittir**: birine vurduğunda diğerinin canı azalmaz.

7.3 Unity'de nesne: GameObject ve bileşenler

Unity'de sahnedeki her şey bir **GameObject**'tir. GameObject kendi başına boş bir kukladır; ona **bileşen (component)** takarak özellik ve yetenek kazandırırız.



Şekil 9: GameObject boştur; bileşenler ona görünüm, beden, fizik ve davranış kazandırır

BENZETİM | Boş kukla ve takılan parçalar

Boş bir robot düşün. Göz takarsan görür, motor takarsan hareket eder, beyin takarsan karar verir. Parçaları değiştirerek robotu istediğin gibi yapılandırırsın. **GameObject = boş robot, Component = takılan parçalar.**

Bileşen	Kazandırdığı şey
Transform	Konum, dönüş, ölçek (her GameObject'te vardır)
Sprite Renderer	Ekranda sprite olarak görünme
Collider 2D	Fiziksel beden, çarpışma
Rigidbody 2D	Fizik: yerçekimi, kuvvet
Animator	Animasyon oynatma
Script (C#)	Özel davranış, oyun mantığı

BİLGİ | Inspector = kimlik kartı

Bir nesneyi seçtiğinde Inspector'da gördüğün her değer (konum, hız, kütle, Gravity Scale…) o nesnenin bir **özelliğidir**. Bu değerleri değiştirmek, nesnenin özelliklerini değiştirmektir.

7.4 C# ve Unity ilişkisi

Başlangıçta en sık karıştırılan konu şudur: **Unity bir oyun motorudur, C# ise bir programlama dilidir.** Unity sahneyi ve araçları sağlar; C# ise nesnelere ne yapacaklarını söylediğimiz dildir.

BENZETİM | Film seti ve senaryo

Unity bir film stüdyosudur: set, kamera, ışık, dekor hazırdır. **GameObject'ler** oyunculardır. **Bileşenler** oyuncunun kostümü ve ekipmanıdır. **C# script'i** ise oyuncunun elindeki **senaryodur**: "şu olunca koş, şu olunca zıpla, şu olunca bağır." **C# dili** senaryonun yazıldığı dildir, Türkçe veya İngilizce gibi.

Senaryo olmadan da set kurulur ve oyuncu dekorda durur; ama film (oyun) ancak senaryoyla canlanır.

Film seti	Unity'de karşılığı
Film stüdyosu	Unity Editor (motor)
Oyuncu	GameObject
Kostüm ve ekipman	Bileşenler (Sprite Renderer, Collider, Rigidbody…)
Senaryo	C# script
Senaryonun yazıldığı dil	C# programlama dili
Sahne / bölüm	Scene

Bir C# script'i aslında bir **sınıftır**. Bu script'i bir GameObject'e eklediğimizde, o sınıf nesne üzerinde bir **bileşen** olarak çalışır. Yani "senaryoyu oyuncuya vermek" demek, sınıfı nesneye bağlamak demektir.

7.5 OOP'nin dört temel ilkesi

İlke	Anlamı	Günlük hayat	Oyun örneği
Kapsülleme (Encapsulation)	Nesnenin iç ayrıntılarını gizleyip sadece gerekli kısmı dışarıya açmak	TV kumandası: iç devreyi bilmeden düğmeye basarsın	Karakterin canı doğrudan değil, "hasar al" davranışıyla değişir
Soyutlama (Abstraction)	Karmaşık ayrıntıları atlayıp sadece önemli kısmı modellemek	Araba kullanırken sadece direksiyon, gaz ve fren	"Düşman" sınıfı sadece can, hız ve saldırıyı içerir

İlke	Anlamı	Günlük hayat	Oyun örneği
Kalıtım (Inheritance)	Bir sınıfın başka bir sınıfın özelliklerini miras alması	Çocuk, anne babadan bazı özellikleri alır	Zombi ve İskelet, "Düşman" sınıfının can ve hız özelliklerini miras alır
Çok biçimlilik (Polymorphism)	Aynı komutun farklı nesnelerde farklı sonuç vermesi	"Ses çıkar" komutu: kedi miyavlar, köpek havlar	"Saldır": kılıçlı vurur, okçu ok atar, büyücü büyü yapar

7.6 Prefab: hazır nesne şablonu

Unity'de aynı nesneden sahnede çok sayıda kullanmak isteyebiliriz (50 tane altın, 10 tane düşman). Bunun için nesneyi bir **Prefab** olarak kaydederiz. Prefab, hazır bir **şablon nesnedir**; ondan istediğimiz kadar kopya üretiriz.

> **BENZETİM | Kurabiye kalıbı**
> Prefab'ı kurabiye kalıbı gibi düşün. Kalıbı bir kez hazırlarsın, sonra istediğin kadar kurabiye çıkarırsın. Kalıbın şeklini değiştirirsen bundan sonra çıkan (ve bağlı olan) tüm kurabiyeler değişir. Yani 50 altının hepsini tek tek düzeltmezsin, Prefab'ı düzeltirsin.

> **BİLGİ | Dikkat: birebir aynı değil**
> Prefab, sınıfla birebir aynı şey değildir. Sınıf kodda tanımlanan bir tasarımdır; Prefab ise Unity'de hazırlanmış, bileşenleriyle birlikte kayıtlı bir nesne şablonudur. İkisinin ortak fikri "bir kez tasarla, çok kez kullan" yaklaşımıdır.

7.7 Sonraki aşama

Kodlamaya geçtiğimizde şu fikir sürekli karşımıza çıkacak: **bir sınıf yazarız, onu bir GameObject'e bileşen olarak ekleriz ve nesnenin özelliklerini değiştirerek davranışını yönlendiririz.** Bu bölümdeki kavramları rahatça hayal edebiliyorsan, kod yazarken ne yaptığını anlamak çok daha kolay olacaktır.

Tekrar Soruları

1. Sınıf ve nesne arasındaki farkı bir günlük hayat örneğiyle açıkla.
2. Aynı sınıftan üretilen iki nesnenin özellik değerleri neden farklı olabilir?
3. GameObject ve bileşen (component) kavramlarını açıkla. Bir oyuncu karakterine hangi bileşenler eklenebilir?
4. Unity ile C# arasındaki ilişkiyi film seti benzetimiyle anlat.
5. OOP'nin dört temel ilkesini say ve her birine kendi örneğini yaz.
6. Prefab nedir? Neden işimizi kolaylaştırır?

8. Vize Öncesi Hızlı Kontrol

8.1 Bunları yapabiliyor olmalısın

Konu	Kontrol: Yapabiliyor musun?
Oyun motoru	Tanımını, bileşenlerini ve kullanım amaçlarını anlatabilir, en az üç motor sayabilirim
Unity kurulumu	Hub ve Editor farkını açıklayabilir, kurulum ve proje açma adımlarını sıralayabilirim
Editor arayüzü	Scene, Game, Hierarchy, Project, Inspector görevlerini söyleyebilirim
2D oyun türleri	Altı türü kamera açısı ve örnekle ayırt edebilirim
Sprite ve Tilemap	Sprite, tile, tileset, tilemap ve palette'i açıklayıp harita yapma adımlarını sıralayabilirim
Collider / Rigidbody	Tür ve işlevlerini, çarpışmanın iki şartını açıklayabilirim
Trigger	Collider ile farkını ve kullanım yerlerini söyleyebilirim
Animasyon	Clip, Controller ve Animator farkını bilirim; parametre türlerini ve geçiş koşulunu açıklayabilirim
Sprite Editor	Slice türlerini, hücre sayısı hesabını ve pivot kavramını açıklayabilirim
OOP	Sınıf, nesne, özellik, davranış, GameObject, bileşen, C# ve Unity ilişkisini benzetimle anlatabilirim

8.2 Terim sözlüğü

Türkçe	İngilizce / Unity terimi	Kısa anlamı
Oyun motoru	Game Engine	Oyun yapımı için hazır altyapı sunan yazılım
Sahne	Scene	Oyunun bir bölümü; nesnelerin yerleştirildiği alan
Oyun nesnesi	GameObject	Sahnedeki her türlü nesne
Bileşen	Component	Nesneye özellik kazandıran parça
Sprite	Sprite	2D görsel nesne
Sprite sayfası	Sprite Sheet	Çok sayıda kareyi içeren tek resim
Karo / parça	Tile	Haritayı oluşturan küçük kare parça
Harita katmanı	Tilemap	Tile'ların dizildiği katman
Çarpışma alanı	Collider	Nesnenin fiziksel bedeni
Katı cisim	Rigidbody	Nesneyi fizik motoruna dahil eden bileşen
Tetikleyici	Trigger	Katı olmayan algılama alanı

Türkçe	İngilizce / Unity terimi	Kısa anlamı
Animasyon klibi	Animation Clip	Tek bir hareketin kare kayıtları
Durum	State	Animator'daki bir hareket durumu
Geçiş	Transition	İki durum arasındaki ok
Parametre	Parameter	Animator'a verilen bilgi değeri
Şablon nesne	Prefab	Tekrar kullanılabilir hazır nesne
Sınıf	Class	Nesne üretmek için tasarım şablonu
Nesne	Object	Sınıftan üretilmiş gerçek örnek

Vize sınavında başarılar dilerim.