

Durum makinesi: oyun şu an hangi ekranda?

Menü, oyun ve bitiş ekranlarını birbirine karıştırmadan yönetmek.

Durum ve geçiş

Diyagram çizmek

oyunDurumu

Duruma göre draw ve girdi

Geçiş tablosu

SORU

Bir oyunu açtınız. Karakter hemen kořmaya mı başlıyor?

Önce ne görüyorsunuz? Oyun bitince ne görüyorsunuz? Bu ekranların her birinde aynı tuşlar aynı işi mi yapıyor?



Durum makinesi

Durum makinesi (state machine)

Sistemin her an **yalnızca bir durumda** olduğu; belirli olaylarla (tuş, tıklama, süre, çarpışma) başka durumlara geçebildiği yapıdır.

Durum	Ekranda ne görünür?	Girdi ne işe yarar?
Menü	Başlık, "Başla" talimatı	SPACE → oyunu başlat
Oyun	Oyuncu, hedef, top, skor	WASD hareket; çarpışma → bitiş
Bitiş	Skor, "Tekrar dene"	R → menüye dön

Durum oyunun aşamasıdır. **Geçiş** (transition), bir olay olunca durum değişkeninin güncellenmesidir.

Üç durum, üç geçiş

Durum Makinesi — Menü, Oyun, Bitiş Geçişleri



Tek aktif durum: oyunDurumu

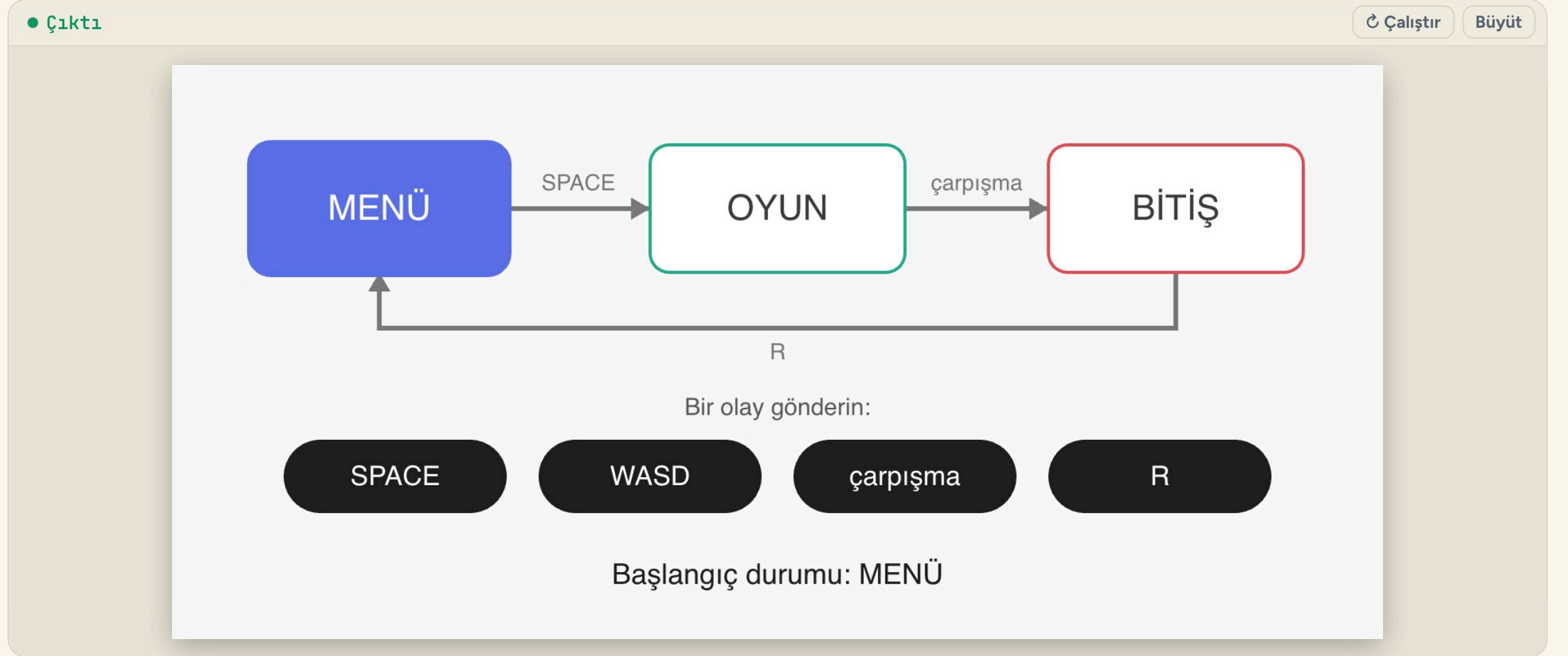
Menü → SPACE → oyun başlar · Oyun → çarpışma → bitiş · Bitiş → R → menü

Menüde WASD → oyuncu hareket etmez (guncelleOyun çağrılmaz)

Bitişte tıklama → hamle işlenmez (mousePressed durum kontrolü)

Kutular durumlar, oklar geçişler, okun üstündeki yazı o geçişi tetikleyen olay.

Olay düğmelerine tıklayın



Menüdeyken WASD'ye ya da R'ye basın: hiçbir şey olmaz. Bir olay, yalnızca o durum için tanımlıysa geçiş yaptırır.

SORU

Bunu iki boolean ile yazsak olmaz mı?

sketch.js

Kopyala

Büyüt

```
let menuAcik = true;  
let oyunAcik = false;
```

Bir hata sonucu ikisi de `true` olursa ekranda ne görürsünüz?

Tek değişken, her an tek değer

Ayrı boolean'ların riski

İkisi birden `true` olursa `draw` aynı frame'de hem menüyü hem sahneyi çizer. Girdinin hangi bloğa gideceği belirsizleşir.

Tek `oyunDurumu`

Her an yalnızca bir değer taşır. Geçiş tek bir atama satırıdır.

Yaklaşım	Örnek	Ne zaman yeterli?
Tek boolean	<code>oyunBitti</code>	Yalnızca devam / bitti
Metin durum	<code>"menu"</code> , <code>"oyun"</code> , <code>"bitis"</code>	Birkaç aşamalı akış
Ayrı boolean'lar	<code>menuAcik</code> , <code>oyunAcik</code>	Önerilmez: çakışma riski

Daire Tıklama'daki `oyunBitti` aslında iki durumlu küçük bir makineydi: oynuyor / bitti.

Trafik ışığı: geçişi süre tetikler



Kırmızı → Sarı → Yeşil → Sarı → Kırmızı. Burada olay bir tuş değil; sürenin dolması.

Turnike: geçersiz olay yok sayılır

Turnike / geçiş kapısı

Kart geçersizse veya kimse geçmezse başa döner — itmek Kilitli'yi açmaz



Kilitli

Kart okutulursa
"Doğrulanıyor"a geçer.

Doğrulanıyor

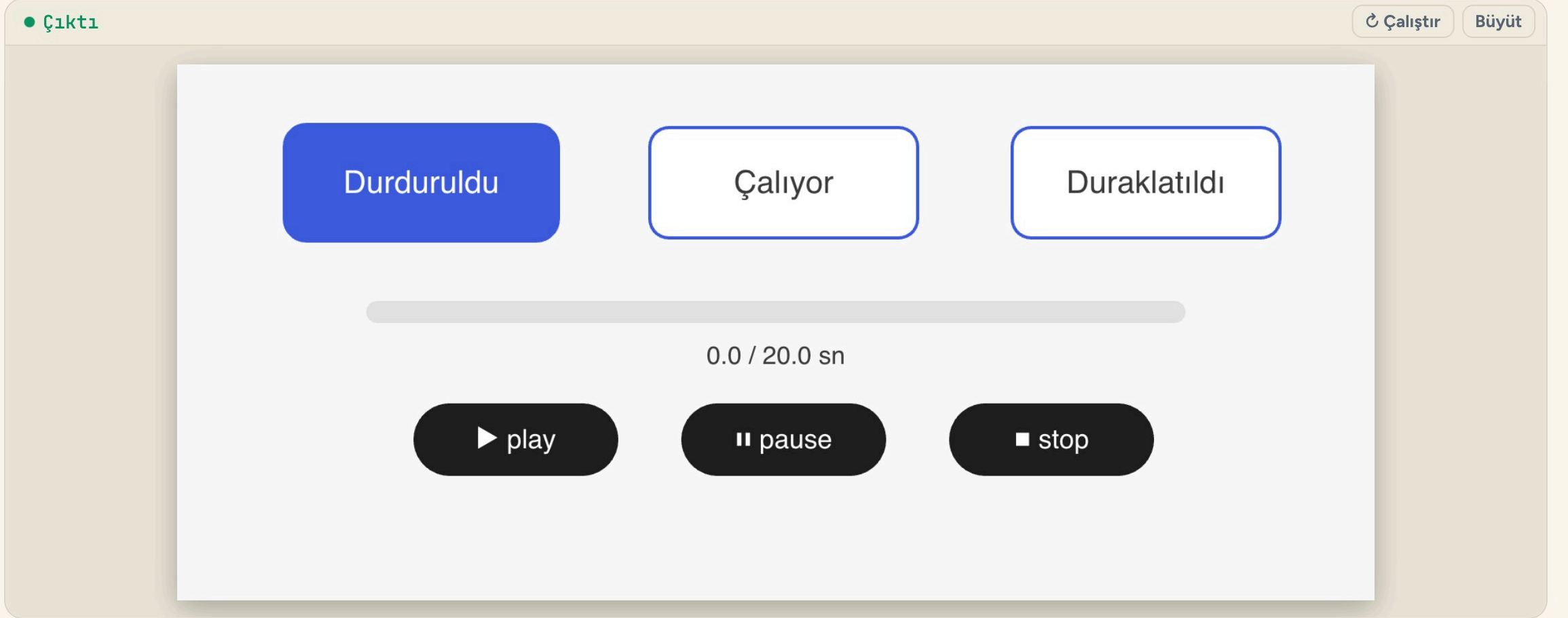
Kart geçerliyse "Açık",
geçersizse yeniden "Kilitli".

Açık

Biri geçince ya da 5 saniye
dolunca "Kilitli".

Kilitliyken turnikeyi itmek bir olay değildir; sistem olduğu yerde kalır. Menüde WASD'nin hiçbir şey yapmamasıyla aynı mantık.

Medya oynatıcı: düğmelere tıklayın



Durduruldu ↔ Çalışıyor ↔ Duraklatıldı. Oyuna “duraklat” eklemek istediğinizde de kalıp aynıdır: yeni bir durum, birkaç yeni geçiş.

Pac-Man'ın hayaletleri de birer durum makinesi



Aynı hayalet, durumuna göre bambaşka davranıyor: kovalarken size doğru, korkmuşken sizden uzağa. Geçiş bir olay (güç hapı) ve bir süre tetikliyor.

Durum makinesi yalnızca menü için değil

Pac-Man hayaletleri

Kovala, dağıl, korkmuş,
yenmiş. Her modda
hedef farklı.

Mario

Küçük, büyük, ateşli.
Mantar ve çiçek geçişi
tetikler; düşmana
değmek geri döndürür.

Düşman yapay zekâsı

Devriye gez →
oyuncuyu gör → kovala
→ yaklaş → saldır →
kaybet → devriyeye dön.

Kapı, sandık, tuzak

Kapalı → açılıyor → açık.
Animasyon bitince geçiş
olur.

Ortak nokta: nesne bir anda tek durumdadır ve **neye izin verildiği** o duruma bağlıdır.

Bilgisayar biliminde yeri

Sonlu durum makinesi (FSM)

Finite state machine: sınırlı sayıda durum, olaylarla tanımlı geçişler. Biçimsel bir modeldir. Derleyiciler, ağ protokolleri, arayüz ekranları ve gömülü sistemler aynı fikri kullanır.

Nesne yönelimli tasarımla ilişkisi

Durum sayısı arttıkça `if / else if` zinciri uzar. İleride her durumu ayrı bir sınıf yapacağız; her biri kendi `guncelle()` ve `ciz()` metodunu taşıyacak. Buna **State deseni** denir.

Bu derste otomat teorisinin matematiğine girmiyoruz. Metin sabitli basit sürüm hem yeterli hem okunması kolay.

Diyagramı kendiniz çizin

Kutu

Durum

Ok

Geçiş

Ok yazısı

Olay

Kâğıt yeterlidir. Dijital için ücretsiz araçlar da var:

Araç	Adres	Not
Mermaid Live	mermaid.live	Metinden otomatik şekil
diagrams.net	app.diagrams.net	Sürükle bırak
Excalidraw	excalidraw.com	El çizimi hissi, hızlı taslak

Amaç sanat değil

“Hangi olay hangi duruma götürür?” sorusunu tek bakışta yanıtlamak. Kod yazarken bu diyagram, `keyPressed()` ve `draw()` dallarının kontrol listesi olur.

Mermaid: diyagramı metinle yazmak

Mermaid

Büyüt

```

1 stateDiagram-v2
2   [*] --> menu
3   menu --> oyun : SPACE
4   oyun --> bitis : carpisma
5   bitis --> menu : R

```

Satır	Anlamı
stateDiagram-v2	Bu bir durum diyagramıdır
[*] --> menu	Siyah nokta başlangıç; ilk durum <code>menu</code>
menu --> oyun : SPACE	<code>menu</code> iken SPACE → <code>oyun</code>
oyun --> bitis : carpisma	<code>oyun</code> iken çarpışma → <code>bitis</code>
bitis --> menu : R	<code>bitis</code> iken R → <code>menu</code>

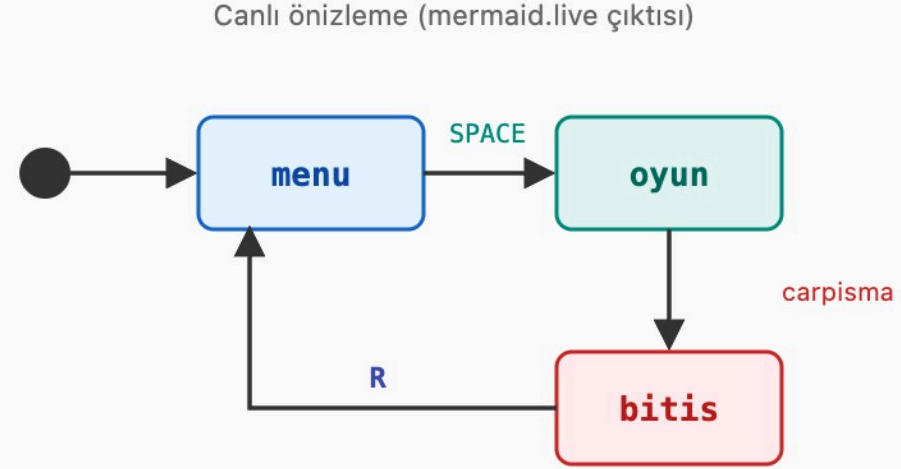
Kalıp: `A --> B : olay`. Metni `mermaid.live` adresine yapıştırınca şekil kendiliğinden oluşur. Metin olduğu için Git'te saklamak da kolaydır.

Kaynak metin ve oluşan diyagram

Mermaid stateDiagram — kod ve çıktı

```
// mermaid.live
stateDiagram-v2
[*] --> menu
menu --> oyun : SPACE
oyun --> bitis : carpisma
bitis --> menu : R
```

Kaynak (metin)



Üretilen diyagram

[*] = başlangıç · A --> B : olay = A durumundan B'ye geçiş (ok etiketi = olay adı)

Aynı akış kitaptaki Görsel 1 ile uyumludur; Mermaid metni kopyala-yapıştır ile mermaid.live'da açılır.

Yapay zekâya geçiş tablonuzu verip “bunu stateDiagram-v2 olarak yaz” diyebilirsiniz. Çıkan metni mutlaka Live Editor’da açıp okları kontrol edin.

Satranç: sıra bir durumdur

Satranç

Sıra tabanlı: beyaz / siyah / oyun bitti



Beyazın sırası

Yalnızca beyaz taş oynanır.

Siyahın sırası

Yalnızca siyah taş oynanır.

Oyun bitti

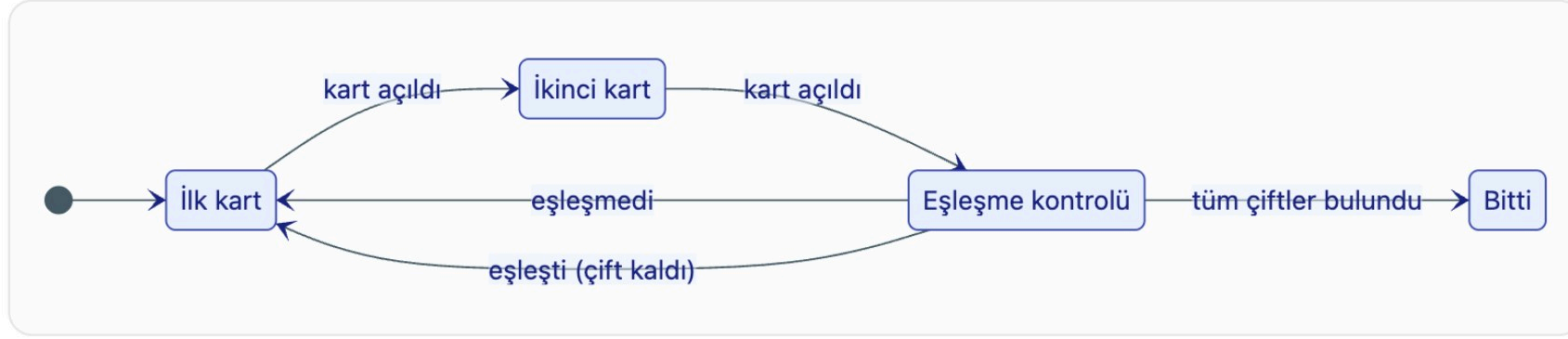
Mat ya da pat. Hamle kabul edilmez.

Hamle oyunu bitirmiyorsa sıra karşıya geçer; mat ya da pat üretiyorsa "Oyun bitti".

Hafıza kartları: kaçınıcı kartı açıyoruz?

Hafıza kartları

İlk kart → ikinci kart → eşleşme kontrolü → (tekrar veya bitti)



İlk kart

Bir kart açılır; ikinci seçim beklenir.

İkinci kart

Başka bir kart açılır; iki yüz görünür.

Eşleşme kontrolü

Kısa bekleme. Eşleşirse açık kalır, değilse kapanır.

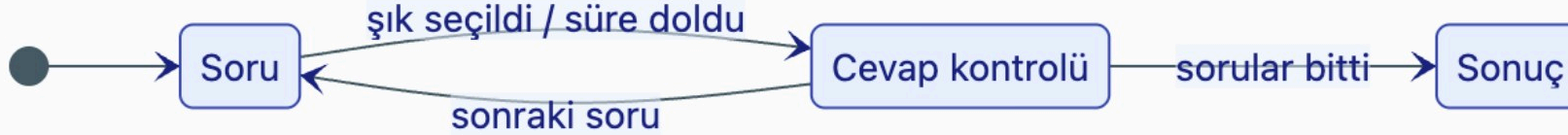
Bitti

Bütün çiftler bulundu.

Bilgi yarışması: 10 soru da olsa 50 soru da olsa üç durum

Bilgi yarışması

Soru → cevap kontrolü → sonraki soru veya sonuç



Soru

Soru metni, şıklar ve süre.

Cevap kontrolü

Doğru / yanlış geri bildirimi.
Şıklar kilitli.

Sonuç

Sorular bitince toplam puan.

Soru sayısı durum sayısını artırmaz. Kaçıncı soruda olduğumuz ayrı bir değişkende tutulur.

Tetris: kontrol ne zaman kapalı?

Tetris

Parça düşüyor → satır temizleme → yeni parça; tepe dolunca bitti



Parça düşüyor

Oklarla hareket ve döndürme.

Satır temizleme

Dolu satırlar silinirken kontrol kapalı.

Oyun bitti

Yeni parça doğduğu yere sığmıyor: tepe dolmuş.

Bir ok kendi kutusuna dönüyor: parça yere oturdu, dolu satır yok, yeni parça geliyor. Durum aynı kalır.

Sıradaki oyunumuz XOX'ta da aynı omurga var

- **Menü:** kurallar. SPACE ile başla.
- **X'in sırası:** yalnızca X, boş bir hücreye konur.
- **O'nun sırası:** yalnızca O konur.
- **Bitiş:** üçlü ya da berabere. R ile menüye dön.

Kodda biraz farklı tutacağız

X ve O sırası kavram olarak ayrı durumlar. Kodda ise tek bir `DURUM_OYUN` içinde, `sira = "X"` / `"O"` değişkeniyle tutulacak. Menü / oyun / bitiş omurgası aynı kalacak.

Oyun döngüsünde durum

- 1 Mevcut durumu oku: `oyunDurumu`.
- 2 Duruma göre **güncelle**: hareket ve çarpışma yalnızca oyun durumundayken.
- 3 Duruma göre **çiz**: menü metni ya da oyun sahnesi.
- 4 Olay fonksiyonlarında (`keyPressed`) duruma göre **geçiş** uygula.

Durumu saklamak: sabitler ve tek değişken

sketch.js

Kopyala

Büyüt

```
1 const DURUM_MENU = "menu";
2 const DURUM_OYUN = "oyun";
3 const DURUM_BITIS = "bitis";
4
5 let oyunDurumu = DURUM_MENU;
```

sketch.js

Kopyala

Büyüt

```
if (oyunDurumu === DURUM_OYUN) {
  // yalnızca oyun aşamasında çalışır
}
```

Sabit	Anlamı
DURUM_MENU	Başlangıç, ana menü
DURUM_OYUN	Aktif oyun
DURUM_BITIS	Oyun bitti ekranı

Durum değerleri metin olarak tutulur. Karşılaştırmalarda `===` kullanılır.

SORU

SPACE'e basıyorsunuz, oyun başlamıyor. Hata mesajı da yok.

sketch.js

Kopyala

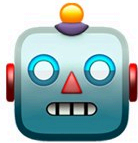
Büyüt

```
let oyunDurumu = "menü";

function keyPressed() {
  if (oyunDurumu === "menu" && key === " ") {
    oyunDurumu = "oyun";
  }
}
```

"menü" ile "menu" farklı metinlerdir

"menü" eşit mi
"menu"?



JavaScript

Hayır. false. Şikâyet
edecek bir şey yok.



hata mesajı yok

Sessiz hata

Karşılaştırma yalnızca `false` döner. Sonuç: nedeni belirsiz bir "hiçbir şey olmuyor".

Sabitlerin asıl kazancı

`DURUM_MENU` yerine `DURUM_MENÜ` yazarsanız JavaScript tanımsız ad için **hata verir** ve sorunu doğrudan gösterir.

Yeni tur: her şeyi başa al

sketch.js

Kopyala

Büyüt

```
1 function oyunuSifirLa() {  
2   skor = 0;  
3   oyuncuX = width / 2;  
4   oyuncuY = height / 2;  
5   topX = random(50, width - 50);  
6   topY = 40;  
7   hizX = random(-120, 120);  
8   hizY = 80;  
9 }
```

Skor, oyuncunun konumu, topun hızı gibi oyun verileri durumdan bağımsız değişkenlerde tutulur. Durum yalnızca **hangi kodun çalışacağını** belirler.

Neden bir fonksiyon?

Oyun her başladığında aynı sıfırlama gerekir. Tek bir yerde durursa bir değişkeni unutma riski azalır. `random` sayesinde her tur biraz farklı başlar.

Duruma göre draw()

sketch.js

Kopyala

Büyüt

```
1 function draw() {  
2   if (oyunDurumu === DURUM_MENU) {  
3     cizMenu();  
4   } else if (oyunDurumu === DURUM_OYUN) {  
5     guncelleOyun();  
6     cizOyun();  
7   } else if (oyunDurumu === DURUM_BITIS) {  
8     cizBitis();  
9   }  
10 }
```

Fonksiyon

Görevi

cizMenu()

Arka plan, başlık, talimat

guncelleOyun()

Hareket, çarpışma, skor; yalnızca oyun aşamasında

cizOyun()

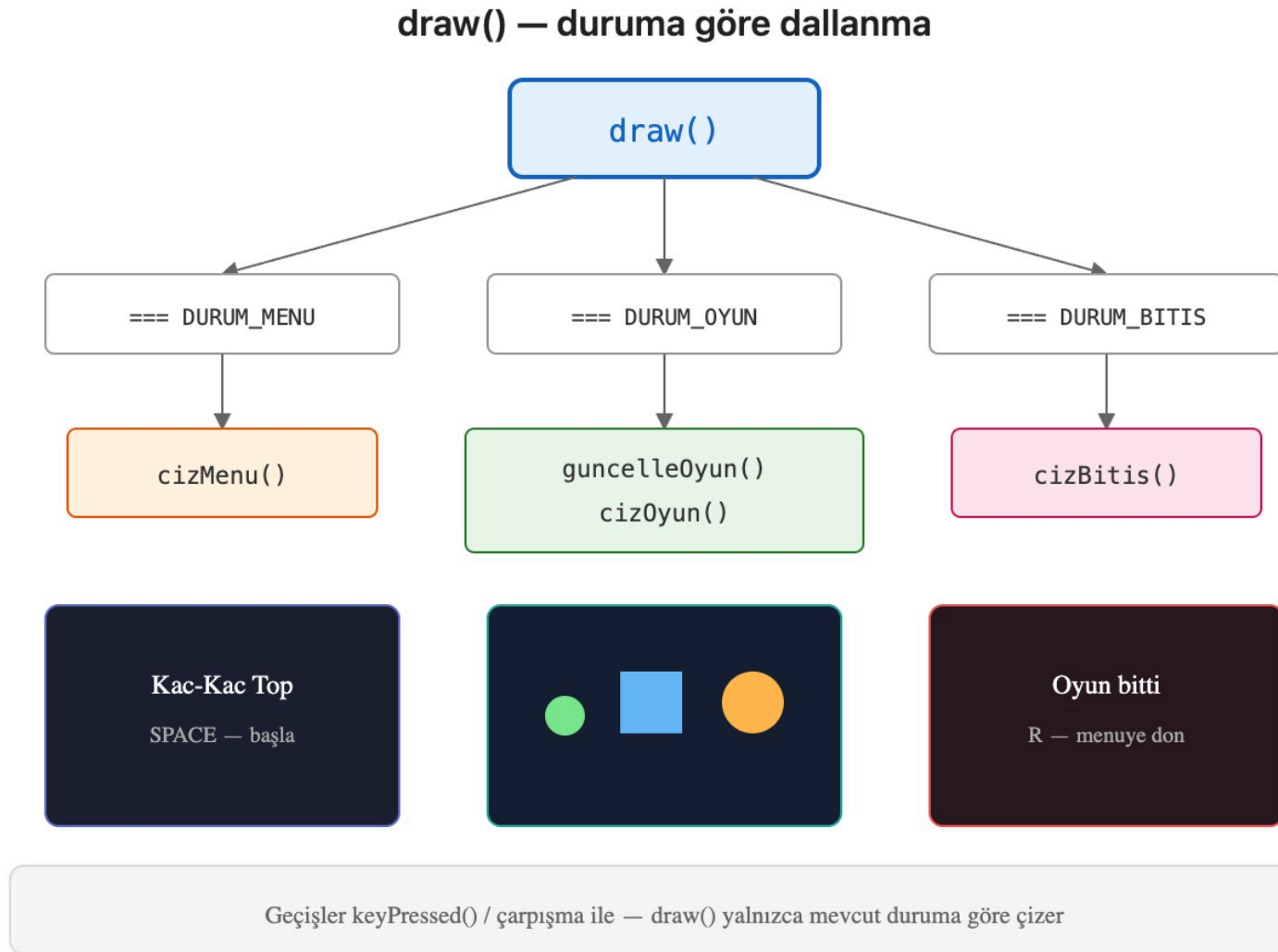
Oyuncu, top, arayüz

cizBitis()

Son skor, yeniden başlatma talimatı

draw artık bir yönlendirici: duruma bakıyor ve işi ilgili fonksiyonlara bırakıyor.

Her durumda başka bir ekran



Hangi durumda ne çalışır?

Durum	<code>draw()</code> içinde	<code>keyPressed()</code>	Sürekli girdi (<code>keyIsDown</code>)
DURUM_MENU	<code>cizMenu()</code>	SPACE → oyuna geç	Yok sayılır
DURUM_OYUN	<code>guncelleOyun()</code> + <code>cizOyun()</code>	—	WASD / ok tuşları
DURUM_BITIS	<code>cizBitis()</code>	R → menüye dön	Yok sayılır

Girdi de duruma bağlıdır. Aynı tuş farklı aşamalarda farklı iş yapabilir ya da bilinçli olarak yok sayılır.

Geçişler `keyPressed()` içinde

sketch.js

Kopyala

Büyüt

```
1 function keyPressed() {  
2   if (oyunDurumu === DURUM_MENU && key === " ") {  
3     oyunuSifirla();  
4     oyunDurumu = DURUM_OYUN;  
5   }  
6   if (oyunDurumu === DURUM_BITIS && (key === "r" || key === "R")) {  
7     oyunDurumu = DURUM_MENU;  
8   }  
9 }
```

- Her `if` iki şeye birden bakıyor: **hangi durumdayız** ve **hangi tuş**.
- SPACE ve R tek seferlik olaylar: yerleri `keyPressed`.
- Oyun içi sürekli hareket `keyIsDown` ile `guncelleOyun()` içinde kalır.

Fare için de aynı

`mousePressed()` içinde de duruma bakın. Yoksa bitiş ekranındaki bir tıklama oyun mantığını tetikleyebilir.

Geçiş tablosu: önce kâğıtta

Mevcut durum	Olay / koşul	Yeni durum	Ek işlem
DURUM_MENU	SPACE basıldı	DURUM_OYUN	oyunuSifirla()
DURUM_OYUN	Oyuncu hedefe değdi	DURUM_OYUN (değişmez)	Skor artar, hedefYerlestir()
DURUM_OYUN	Oyuncu topa çarptı	DURUM_BITIS	—
DURUM_BITIS	R basıldı	DURUM_MENU	—

Her olay durum değiştirmez

Hedef toplamak skoru artırır ama oyun sürer. Durum, yalnızca tabloda yeni bir durum yazan satırlarda değişir.

Tablo hazırsa kod hazırdır

`draw()` ve `keyPressed()`, tablodaki satırların doğrudan çevirisine dönüşür. Tablosuz yazınca genellikle bir geçiş unutulur.

Yapay zekâyla: durum diyagramı

 15-20 dk

1. Üç durumu ve en az dört geçişi kâğıda çizin.
2. Yapay zekâdan aynı diyagramı Mermaid metni olarak isteyin. `mermaid.live` adresinde açın. Eksik ya da hatalı bir geçiş var mı?
3. “Duraklat” eklemek isterseniz hangi yeni durum ve geçişler gerekir? Sorun, sonra kendi diyagramınıza ekleyin.

Kod zamanı

 40 dk

1. **İskelet:** üç durumlu bir sketch yazın. Her durumda farklı arka plan rengi ve ortada durumun adı. SPACE ile oyuna, R ile menüye. Oyundan bitişe geçiş için şimdilik B tuşu.
2. Oyun durumundayken bir kare ok tuşlarıyla hareket etsin; menüde ve bitişte hareket etmesin.
3. `DURUM_DURAKLAT` ekleyin: oyundayken P ile duraklat, yeniden P ile devam.
4. Trafik ışığını durum makinesiyle yazın; süreyi `deltaTime` ile sayın.
5. **Zorlu:** fareyi kovalayan bir hayalet. Tıklayınca 4 saniye korksun ve kaçsın.

Oyun bitiş ekranında. Oyuncu SPACE'e bastı. Ne olur?

sketch.js

Kopyala

Büyüt

```
1 function keyPressed() {  
2   if (oyunDurumu === DURUM_MENU && key === " ") {  
3     oyunuSifirla();  
4     oyunDurumu = DURUM_OYUN;  
5   }  
6   if (oyunDurumu === DURUM_BITIS && (key === "r" || key === "R")) {  
7     oyunDurumu = DURUM_MENU;  
8   }  
9 }
```

A Oyun yeniden başlar

B Menüye dönülür

☒ **C** Hiçbir şey olmaz

D Hata verir

İlk **if** menü durumunu, ikincisi R tuşunu istiyor. İkisi de tutmuyor. Bitişte SPACE tanımlı bir olay değil.

Hangisi bir geçiş değildir?

- ☐ A Menüde SPACE'e basınca oyunun başlaması
- ☒ B Oyun sırasında hedef toplayıp skorun 1 artması
- ☐ C Oyuncunun topa çarpıp bitiş ekranına düşmesi
- ☐ D Bitişte R'ye basınca menüye dönülmesi

Skor arttı ama durum hâlâ "oyun". Geçiş, durum değişkeninin değer değiştirmesidir.

Menüde W'ye basılı tutunca oyuncu hareket ediyor. Hata en çok nerede aranır?

A `keyPressed()` içinde

☒ `draw()` içinde: `guncelleOyun()` durum kontrolünün dışında çağrılıyor

C `setup()` içinde

D Sabitlerin yazımında

Sürekli hareket `guncelleOyun()` içindedir. O fonksiyon yalnızca `DURUM_OYUN` iken çağrılmalı.

Merak edenler için kaynaklar

ROBERT NYSTROM

Game Programming Patterns: State

Durum makinesinin oyun mimarisindeki yeri; ücretsiz okunabilen bir kitap bölümü.

gameprogrammingpatterns.com/state.html

MERMAID

Mermaid Live Editor

Diyagram metnini yapıştırın, şekil oluşsun.

mermaid.live

EXCALIDRAW

El çizimi hissiyle diyagram

Hızlı taslak için.

excalidraw.com

DIAGRAMS.NET

draw.io

Sürükle bırak ile durum kutuları ve oklar.

app.diagrams.net

Bugün ne öğrendik?

- ✓ Durum makinesi: sistem her an tek bir durumdadır; olaylar geçişleri tetikler.
- ✓ Ayı boolean'lar yerine tek bir `oyunDurumu` değişkeni.
- ✓ Durum adları için `const` sabitleri: yazım hatası sessiz kalmaz.
- ✓ `draw` duruma göre dallanır; girdi de duruma göre işlenir ya da yok sayılır.
- ✓ Her olay durum değiştirmez; geçiş, durum değişkenine yapılan atamadır.
- ✓ Önce diyagram ve geçiş tablosu, sonra kod.

SIRADAKİ DERS

Kaç-Kaç Top

Menüsü, oyunu ve bitiş ekranı olan ilk tam oyunumuz.

Sonraki sunum →

← Önceki

