

Kaç-Kaç Top: ilk tam oyunumuz

Menüsü, oynanışı ve bitiş ekranı olan bir oyun.
Öğrendiklerimizin hepsi tek sketch'te.

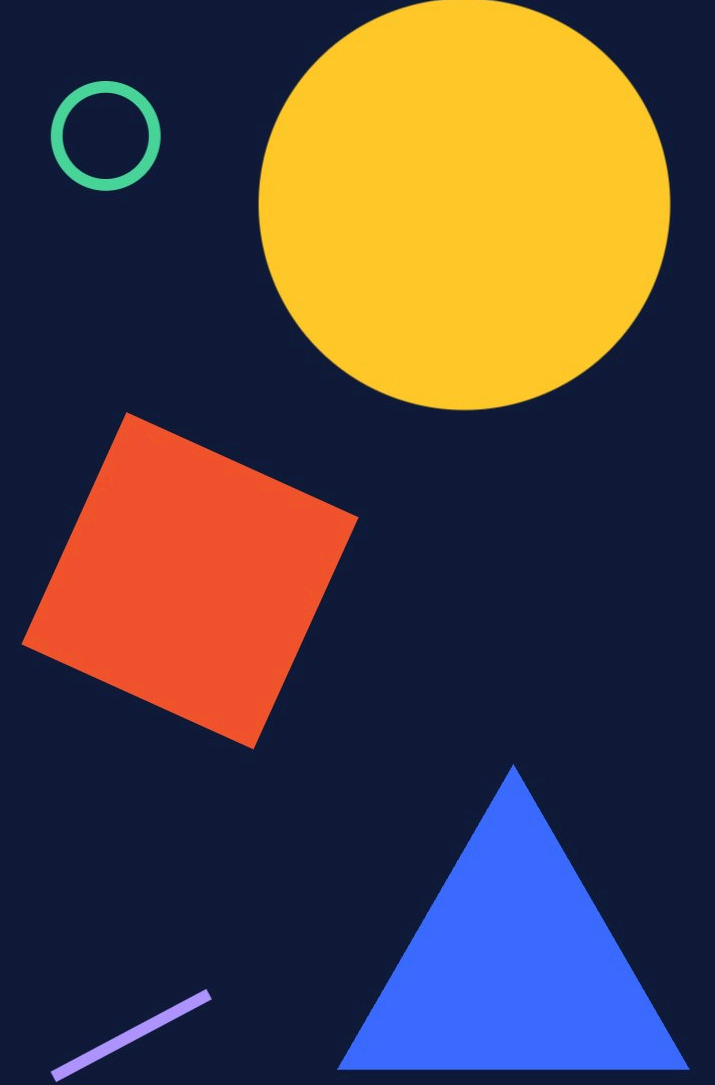
Kurallar

Durum tasarımı

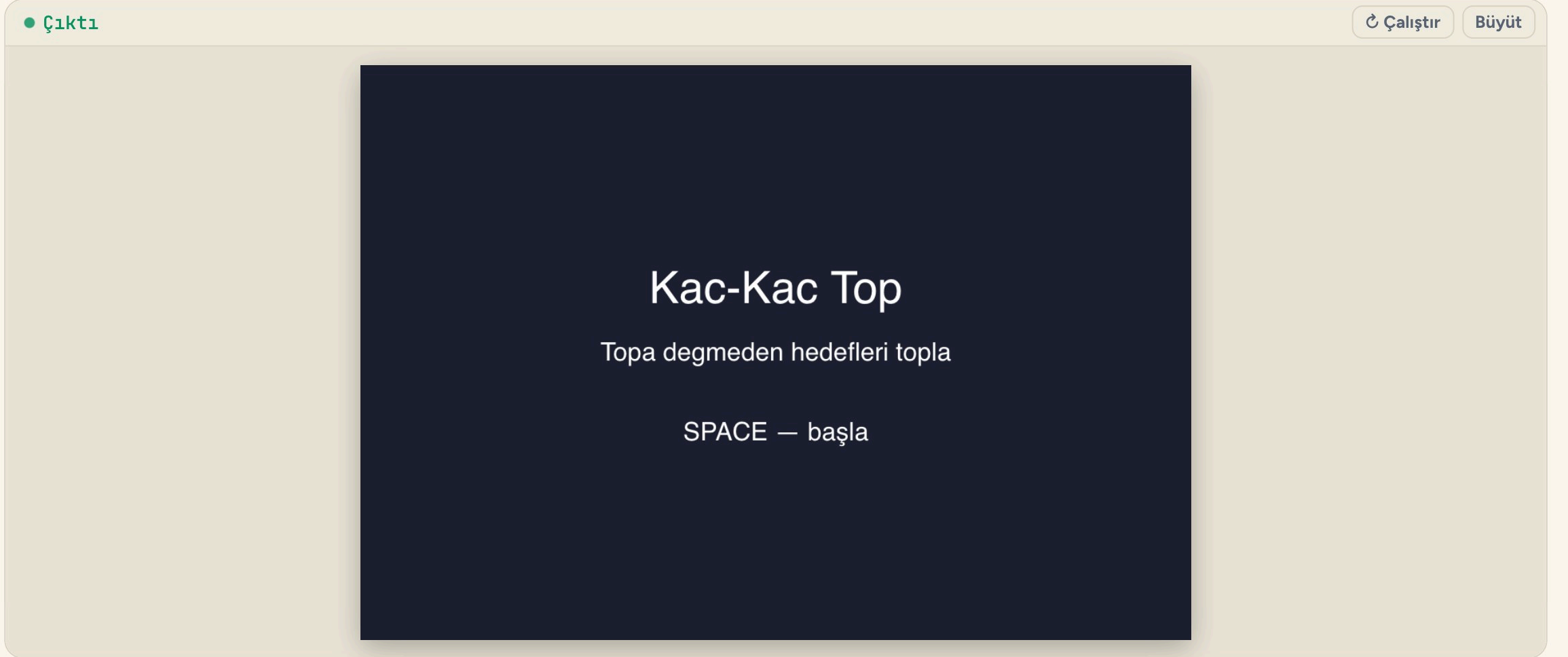
Kodu parça parça okumak

Geçişleri sınamak

Hafıza kartları projesi



Önce oynayalım



Tuvale tıklayın, SPACE'e basın. Ok tuşları ya da WASD ile yeşil hedefleri toplayın; turuncu topa değmeyin. Bitince R.

SORU

Az önce oynadığınız oyunda kaç farklı ekran gördünüz?

Her ekranda hangi tuşlar işe yarıyordu? Hangi olay sizi bir ekrandan diğerine götürdü?



Kurallar ve üç durum

1. **Menü:** "SPACE — başla" yazısı.
2. **Oyun:** kare oyuncu WASD ya da oklarla hareket eder; top duvarlardan sekerek dolaşır.
3. Oyuncu yeşil hedefe değince skor 1 artar; hedef başka bir yere taşınır.
4. Top oyuncuya değerse **bitiş** ekranı gelir; skor gösterilir.
5. Bitişte R ile menüye dönülür.

MENÜ → SPACE → OYUN → çarpışma →

BITİŞ

BITİŞ → R → MENÜ

Küçük bir tasarım kararı

Skor yalnızca hedefe dokununca artıyor. Köşede beklemek işe yaramaz; puan için topun yakınına gitmeyi göze almanız gerekir. Oyun tasarımında buna **risk ve ödül** dengesi denir.

Bu oyunda yeni hiçbir şey yok

Bölüm 2

Fonksiyonlar: `cizMenu`,
`guncelleOyun`, `cizOyun` ...
Güncelle ve çiz ayrımı.

Bölüm 3

`deltaTime` ile hız.
`keyIsDown` ile hareket,
`keyPressed` ile tek basış.
`constrain`.

Bölüm 4

Seken top.
`dist` ile temas testi.
`oyunDurumu` ve geçişler.

Yeni olan tek şey, bunların hepsini **bir araya getirmek**. Kodun tamamı 137 satır; parça parça okuyacağız.

Sabitler ve oyunun verisi

sketch.js · satır 1-15

Kopyala

Büyüt

```
1 const DURUM_MENU = "menu";
2 const DURUM_OYUN = "oyun";
3 const DURUM_BITIS = "bitis";
4
5 const BOYUT = 36;
6 const TOP_YARICAP = 20;
7 const HEDEF_YARICAP = 12;
8 const HIZ_OYUNCU = 200;
9 const UZAK_MESAFE = 90; // hedef oyuncuya bu kadar uzakta doğar
10
11 let oyunDurumu;
12 let oyuncuX, oyuncuY;
13 let topX, topY, hizX, hizY;
14 let hedefX, hedefY;
15 let skor;
```

- Üç durum sabiti.
- Boyutlar ve hızlar da sabit: oyunun "ayar düğmeleri". Değiştirip dengeyi deneyebilirsiniz.
- `oyunDurumu` : hangi ekrandayız?
- Oyuncu, top, hedef ve skor: oyunun verisi. Durumdan bağımsız değişkenlerde duruyor.

setup ve yönlendirici draw

sketch.js · satır 17-32

Kopyala

Büyüt

```
17 function setup() {  
18   createCanvas(520, 360);  
19   noStroke();  
20   oyunDurumu = DURUM_MENU;  
21 }  
22  
23 function draw() {  
24   if (oyunDurumu === DURUM_MENU) {  
25     cizMenu();  
26   } else if (oyunDurumu === DURUM_OYUN) {  
27     guncelleOyun();  
28     cizOyun();  
29   } else if (oyunDurumu === DURUM_BITIS) {  
30     cizBitis();  
31   }  
32 }
```

Oyun menüde başlar. `draw` yalnızca duruma bakar ve işi dağıtır.

Okuma sırası

Bir oyunun kodunu okurken hep `draw` 'dan başlayın. Üç daldan yalnızca biri, `oyunDurumu` 'nun o anki değerine göre çalışır.

Menü ekranı

sketch.js · satır 34-43

Kopyala

Büyüt

```
34 function cizMenu() {  
35   background(28, 32, 48);  
36   fill(255);  
37   textSize(28);  
38   textAlign(CENTER, CENTER);  
39   text("Kac-Kac Top", width / 2, height / 2 - 40);  
40   textSize(16);  
41   text("Topa degmeden hedefleri topla", width / 2, height / 2);  
42   text("SPACE — başla", width / 2, height / 2 + 50);  
43 }
```

● Çıktı

Çalıştır

Büyüt

Kac-Kac Top

Topa degmeden hedefleri topla

SPACE — başla

Arka plan, başlık, iki satır talimat. Burada hareket yok, çarpışma yok; yalnızca çizim.

guncelleOyun (1/2): oyuncu ve top

sketch.js · satır 45-61

Kopyala

Büyüt

```
45 function guncelleOyun() {
46   let dt = deltaTime / 1000;
47
48   if (keyIsDown(LEFT_ARROW) || keyIsDown(65)) oyuncuX -= HIZ_OYUNCU * dt;
49   if (keyIsDown(RIGHT_ARROW) || keyIsDown(68)) oyuncuX += HIZ_OYUNCU * dt;
50   if (keyIsDown(UP_ARROW) || keyIsDown(87)) oyuncuY -= HIZ_OYUNCU * dt;
51   if (keyIsDown(DOWN_ARROW) || keyIsDown(83)) oyuncuY += HIZ_OYUNCU * dt;
52
53   oyuncuX = constrain(oyuncuX, 0, width - BOYUT);
54   oyuncuY = constrain(oyuncuY, 0, height - BOYUT);
55
56   topX += hizX * dt;
57   topY += hizY * dt;
58   if (topX - TOP_YARICAP < 0) { topX = TOP_YARICAP; hizX = -hizX; }
59   if (topX + TOP_YARICAP > width) { topX = width - TOP_YARICAP; hizX = -hizX; }
60   if (topY - TOP_YARICAP < 0) { topY = TOP_YARICAP; hizY = -hizY; }
61   if (topY + TOP_YARICAP > height) { topY = height - TOP_YARICAP; hizY = -hizY; }
```

Çıktı

Çalıştır

Büyüt

Kac-Kac Top

Topa değmeden hedefleri topla

SPACE — başla

guncelle0yun (2/2): iki temas testi

sketch.js · satır 63-74

Kopyala

Büyüt

```
63 let px = oyuncuX + BOYUT / 2;  
64 let py = oyuncuY + BOYUT / 2;  
65  
66 if (dist(px, py, hedefX, hedefY) < HEDEF_YARICAP + BOYUT / 2) {  
67     skor += 1;  
68     hedefYerlestir();  
69 }  
70  
71 if (dist(px, py, topX, topY) < TOP_YARICAP + BOYUT / 2) {  
72     oyunDurumu = DURUM_BITIS;  
73 }  
74 }
```

Çıktı

Çalıştır

Büyüt

Kac-Kac Top

Topa değmeden hedefleri topla

SPACE — başla

Oyuncu bir kare; merkezi `(px, py)`. İki test de aynı kalıp: `dist` < yarıçapların toplamı. İlki skoru artırır, ikincisi durumu değiştirir.

**Top oyuncuya değdi ve
oyunDurumu = DURUM_BITIS atandı.
Aynı frame'de ekrana ne çizilir?**

sketch.js · satır 23-32

Kopyala

Büyüt

```
23 function draw() {  
24   if (oyunDurumu === DURUM_MENU) {  
25     cizMenu();  
26   } else if (oyunDurumu === DURUM_OYUN) {  
27     guncelleOyun();  
28     cizOyun();  
29   } else if (oyunDurumu === DURUM_BITIS) {  
30     cizBitis();  
31   }  
32 }
```

Oyun sahnesi, son bir kez

- 1 `draw` başladığında durum hâlâ `DURUM_OYUN` ; ikinci dala girilir.
- 2 `guncelleOyun()` çarpışmayı bulur ve durumu `DURUM_BITIS` yapar.
- 3 Aynı dalın devamı çalışır: `cizOyun()` çarpışma anını çizer.
- 4 Bir **sonraki** `draw` çağrısında üçüncü dal çalışır: `cizBitis()`.

Geçiş bir sonraki frame'de görünür

`if / else if` zincirinde dal bir kez seçilir. Durumu dalın ortasında değiştirmek, o frame'de başka bir dala atlatmaz.

Oyun ekranını çizmek

sketch.js · satır 76-88

Kopyala

Büyüt

```
76 function cizOyun() {  
77   background(22, 28, 42);  
78   fill(120, 230, 140);  
79   ellipse(hedefX, hedefY, HEDEF_YARICAP * 2, HEDEF_YARICAP * 2);  
80   fill(100, 200, 255);  
81   rect(oyuncuX, oyuncuY, BOYUT, BOYUT);  
82   fill(255, 190, 70);  
83   ellipse(topX, topY, TOP_YARICAP * 2, TOP_YARICAP * 2);  
84   fill(255);  
85   textSize(16);  
86   textAlign(LEFT, TOP);  
87   text("Skor: " + skor, 12, 12);  
88 }
```

- Sıra önemli: önce hedef, sonra oyuncu, en üstte top.
- Skor sol üstte: küçük bir HUD.
- Bu fonksiyon hiçbir şeyi **değiştirmez**; yalnızca değişkenlere bakıp ekrana aktarır.

Bitiş ekranını çizmek

sketch.js · satır 90-100

Kopyala

Büyüt

```
90 function cizBitis() {  
91   background(35, 25, 30);  
92   fill(255);  
93   textSize(26);  
94   textAlign(CENTER, CENTER);  
95   text("Oyun bitti", width / 2, height / 2 - 50);  
96   textSize(18);  
97   text("Skor: " + skor, width / 2, height / 2);  
98   textSize(14);  
99   text("R - menüye dön", width / 2, height / 2 + 50);  
100 }
```

- Farklı bir arka plan rengi: oyuncu ekranın değiştiğini hemen anlar.
- Skor, oyun ekranındaki aynı `skor` değişkeninden okunuyor.
- Son satır, oyuncuya bundan sonra ne yapabileceğini söylüyor. Her ekranda bir sonraki adım yazılı olmalı.

SORU

Yeni hedef tam oyuncunun üstünde doğarsa ne olur?

Hedef rastgele bir yere konuyor. Şans eseri oyuncunun durduğu yere denk gelirse?



hedefYerlestir : uzak bir yer bulana kadar dene

sketch.js · satır 102-116

Kopyala

Büyüt

```
102 function hedefYerlestir() {
103     let px = oyuncuX + BOYUT / 2;
104     let py = oyuncuY + BOYUT / 2;
105     let kenar = HEDEF_YARICAP + 10;
106     hedefX = random(kenar, width - kenar);
107     hedefY = random(kenar, height - kenar);
108     let deneme = 0;
109     // Güvenlik sınırı: UZAK_MESAFE tuvale göre çok büyük seçilirse uygun konum
110     // bulunamaz ve draw() içindeki bu döngü sketch'i dondurur.
111     while (dist(px, py, hedefX, hedefY) < UZAK_MESAFE && deneme < 50) {
112         hedefX = random(kenar, width - kenar);
113         hedefY = random(kenar, height - kenar);
114         deneme = deneme + 1;
115     }
116 }
```

Oyuncu hiçbir şey yapmadan puan alırdı. Bu yüzden hedef, oyuncuya `UZAK_MESAFE` 'den yakınsa yeniden çekiliyor.

while ve güvenlik sınırı

sketch.js

Kopyala

Büyüt

```
while (dist(px, py, hedefX, hedefY) < UZAK_MESAFE && deneme < 50) {
```

while 'ın doğal yeri

Kaç denemede uygun yer bulunacağı baştan belli değil. "Yeterince uzak bir yer bulana kadar dene."

deneme < 50 neden var?

UZAK_MESAFE tuvale göre çok büyük seçilirse uygun yer hiç bulunamaz. Sınır olmasa döngü hiç bitmez; draw o frame'i tamamlayamaz ve oyun donar.

Oyunlarda yaygın bir kalıp

Rastgele yerleştirme yapan her sistem (düşman doğurma, ödül koyma, harita üretme) bir deneme sınırı taşır. Yer bulunamazsa en son denenen konumla yetinilir.

Yeni tur: oyunuSifirLa

sketch.js · satır 118-127

Kopyala

Büyüt

```
118 function oyunuSifirLa() {  
119     oyuncuX = width / 2 - BOYUT / 2;  
120     oyuncuY = height / 2 - BOYUT / 2;  
121     topX = random(60, width - 60);  
122     topY = 50;  
123     hizX = random(-100, 100);  
124     hizY = random(80, 140);  
125     skor = 0;  
126     hedefYerlestir();  
127 }
```

- Oyuncu ortada başlar.
- Top üstte, rastgele bir x konumunda ve rastgele bir hızla doğar: her tur biraz farklı.
- Skor sıfırlanır, ilk hedef yerleştirilir.
- Yalnızca menüden oyuna geçerken çağrılır.

Geçişler: keyPressed

sketch.js · satır 129-137

Kopyala

Büyüt

```
129 function keyPressed() {  
130   if (oyunDurumu === DURUM_MENU && key === " ") {  
131     oyunuSifirla();  
132     oyunDurumu = DURUM_OYUN;  
133   }  
134   if (oyunDurumu === DURUM_BITIS && (key === "r" || key === "R")) {  
135     oyunDurumu = DURUM_MENU;  
136   }  
137 }
```

- Menüde SPACE: önce sıfırla, sonra oyuna geç.
- Bitişte R: menüye dön.
- Geçişlerin ikisi de tek satırlık atama.
- Başka durumlarda bu tuşlar hiçbir şey yapmaz.

Kodun tamamı geiş tablosunun evirisi

Mevcut durum	Olay	Yeni durum	Kodda nerede?
Menü	SPACE	Oyun	<code>keyPressed</code> · <code>oyunuSifirLa()</code> ile birlikte
Oyun	Hedefe temas	Oyun (deėişmez)	<code>guncelleOyun</code> · skor artar, <code>hedefYerlestir()</code>
Oyun	Topa temas	Bitiş	<code>guncelleOyun</code>
Bitiş	R	Menü	<code>keyPressed</code>

Geişler iki yerde olur

Tuşla tetiklenenler `keyPressed` içinde; oyunun kendi kuralından doğanlar (arpışma) `guncelleOyun` içinde.

Geçişleri deneyin

 20 dk

Oyunu açın ve tablodaki **olmaması gereken** şeyleri deneyin. Her biri için ne olduğunu yazın.

1. Menüdeyken ok tuşlarına basılı tutun, sonra SPACE'e basın. Oyuncu ortada mı başlıyor?
2. Menüdeyken R'ye basın.
3. Oyun sırasında SPACE'e basın. Skor sıfırlanıyor mu?
4. Bitiş ekranında SPACE'e basın. Bitişte ok tuşlarına basın.
5. Kodda `UZAK_MESAFE` 'yi 600 yapın. Oyun donuyor mu? Neden donmuyor?

Oyunu geliştirin

 45 dk

1. Her 5 puanda topun hızı %15 artsın.
2. En yüksek skoru tutun ve bitiş ekranında gösterin. İpucu: `oyunuSifirLa` onu sıfırlamamalı.
3. Oyuna `DURUM_DURAKLAT` ekleyin: P ile duraklat, P ile devam.
4. Bitiş ekranından R ile menüye, SPACE ile doğrudan yeni oyuna geçilebilsin.
5. **Zorlu:** ikinci bir top ekleyin. Topların konum ve hızlarını paralel dizilerde tutun; her 10 puanda yeni bir top gelsin.

Yapay zekâyla: bir oyundan durum makinesi çıkarmak

🕒 20–25 dk

1. **Önce siz:** bildiğiniz bir oyun seçin (satranç, Tetris, hafıza kartları...). En az üç durum bulun. Her biri için: ekranda ne görünür, hangi girdi işe yarar?
2. **Sorun:** “Şu oyunu durum makinesi gözüyle anlat: 3–5 durum, her durumda görünenler ve anlamlı girdiler, en az 5 satırlık geçiş tablosu. Durum makinesi olmasa hangi hatalar kolay çıkardı? Kod yazma.”
3. **Denetleyin:** yanlış ya da gereksiz bir durum var mı? Yapay zekâ bazen olmayan kurallar ya da fazladan durumlar ekler.
4. Kendi mini oyununuz için yanıtlayın: hangi üç durum şart, her biri ne işe yarar, neden bu üçü yeterli?

Yapay zekâyla: Mermaid diyagramı çizdirmek

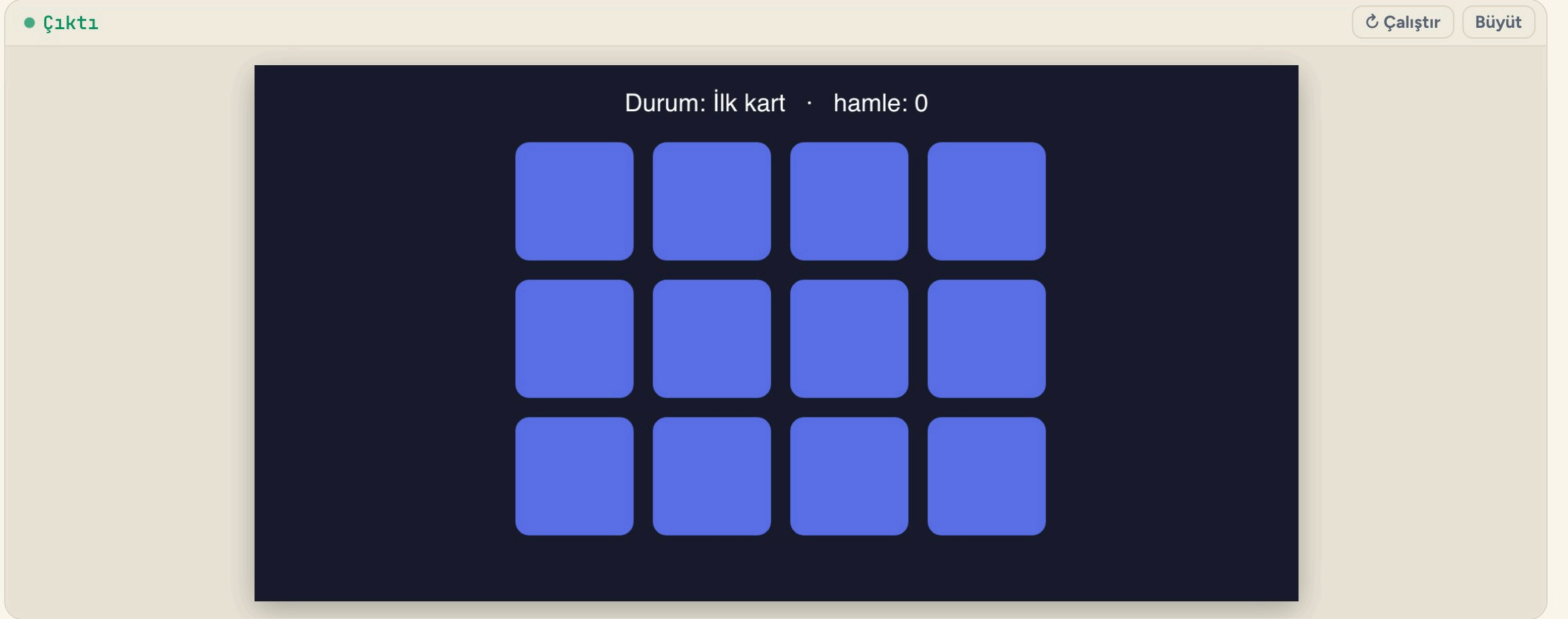
25-35 dk

1. Kaç-Kaç Top ya da kendi oyununuz için geçiş tablosunu yazın: en az dört satır ve bir **olmayan** geçiş (örneğin menüde R).
2. Tabloyu verip `stateDiagram-v2` metnine çevirtin. Başlangıç için `[*] --> ...` kullansın.
3. Metni `mermaid.live` adresinde açın. Çizilmezse ya da ok eksikse düzelttirin.
4. **Makyaj:** kutuları renklendirtin; `oyun --> bitis` okunu belirginleştirtin. Her değişikliği yeniden deneyin.
5. İsteğe bağlı: "Bu diyagramdaki her ok p5.js'te nereye yazılır: `draw` içine mi, `keyPressed` içine mi?"

Dikkat

Yapay zekâ bazen durum diyagramı yerine akış şeması (`flowchart`) üretir ya da olmayan bir duruma ok çizer. Önce doğru geçişler, sonra makyaj.

Sıradaki hedef: Hafıza kartları



Üstteki “Durum” yazısını izleyin: İlk kart → İkinci kart → Eşleşme kontrolü → yeniden İlk kart. Kontrol sırasında tıklamalar yok sayılıyor.

Hafıza kartları (1/2): plan ve kod

 60-90 dk

- 1. Mekaniği anlayın.** Bu oyuna İngilizcede Memory, Concentration ya da Matching pairs denir. Çevrimiçi bir örneğini birkaç dakika oynayın.
- 2. Önce diyagram.** Geçiş tablonuzu yazın, Mermaid ile çizin. İsterseniz başa bir Menü, sona bir Bitti ekleyin. Diyagram netleşmeden koda geçmeyin.
- 3. Kodu siz yazın.** Küçük bir ızgara: 4×3 , yani 6 çift. Önce 2×2 bile yeter.
 - Durum sabitleri ve `oyunDurumu`.
 - Kart değerleri ve açık / kapalı bilgisi için paralel diziler. Class yok.
 - `draw()` : duruma göre çiz. Kapalı kart gri, açık kartta sayı ya da renk.
 - `mousePressed()` : yalnızca uygun durumda tıklanan kartı aç.
 - Eşleşmezse kısa bekleme (`millis` ya da bir sayaç), sonra kartları kapat.

Hafıza kartları (2/2): takılınca ve kontrol

Yapay zekâdan onarım ve açıklama isteyin

“Şu sorun var: [ikinci kart açılmıyor]. Önce öğretmen gibi anlat: hata hangi durum geçişinde olabilir? Sonra yalnızca gerekli satırları düzelt; oyunu baştan yazma. En sonda neyi neden değiştirdiğini madde madde açıkla.”

Yazamadığınız tek bir parça için

“Class kullanmadan yalnızca `kartlariKaristir()` fonksiyonunu yaz: 6 çift (0–5, her biri iki kez) üretip karıştırın.”

Kontrol listesi

- Eşleşme kontrolü sırasında yeni kart tıklanabiliyor mu? Tıklama yok sayılmalı.
- Menüdeyken kart tıklanınca oyun bozuluyor mu?
- İkinci kart açılmadan “eşleşti mi?” soruluyor mu?
- Durum adları sabit mi, yoksa `"ilk"` ile `"İlk kart"` karışıyor mu?
- Yapay zekâ neyi gereksiz uzattı?

Oyunun tamamını yapay zekâya baştan yazdırmayın. Kaç-Kaç Top'taki `oyunDurumu` + `draw` dallanması + `oyunuSifirla` kalıbını yeniden kullanın.

Yapay zekâyla: eksik geçiş

 15-20 dk

1. Menüden oyuna geçişi olan ama bitişten menüye dönüşü “unutulmuş” bir iskelet yazın ya da yazdırın.
2. Yapay zekâyı hatayı buldurun. Teşhisi sizinkiyle aynı mı?
3. Düzeltmeyi kendi sketch’inizde uygulayın; üç ekranı da gezerek doğrulayın.

oyunuSifirLa() çağrısını keyPressed 'den silersek ilk oyunda ne olur?

sketch.js · satır 129-133

Kopyala

Büyüt

```
129 function keyPressed() {  
130   if (oyunDurumu === DURUM_MENU && key === " ") {  
131     oyunuSifirLa();  
132     oyunDurumu = DURUM_OYUN;  
133   }
```

A Hiçbir şey değişmez.

☒ Oyuncu, top ve hedef çizilemez: konumlarına hiç değer verilmedi.

c Oyun menüde kalır.

Bu değişkenler en üstte yalnızca tanımlandı; değerlerini oyunuSifirLa veriyor. Çağrılmazsa hepsi undefined kalır.

Hedef testini ve top testini yer değiştirsek, aynı frame'de ikisine birden değen oyuncuya ne olur?

- A** Oyun biter, puan alamaz.
- ☒ **B** İki sırada da hem puan alır hem oyun biter.
- C** Puan alır, oyun bitmez.

İki test bağımsız iki `if` ; arada `return` ya da `else` yok. Sıra değişse de ikisi de çalışır. "Ölürken puan almasın" isteseydik top testini öne alıp `return` eklerdik.

Oyuncu karesinin sol üst köşesi (100, 100), BOYUT 36. Merkezi nerede?

A (100, 100)

B (136, 136)

✓ (118, 118)

D (82, 82)

`rect` köşeden çizer; `dist` ise merkez ister. Bu yüzden kodda $px = oyuncuX + BOYUT / 2$.

Merak edenler için kaynaklar

WIKİPEDIA

Concentration (card game)

Hafıza kartları oyununun kuralları ve çeşitlemeleri.

[en.wikipedia.org/wiki/Concentration_\(card_game\)](https://en.wikipedia.org/wiki/Concentration_(card_game))

MERMAID

Mermaid Live Editor

Durum diyagramınızı metinle çizin.

mermaid.live

ROBERT NYSTROM

Game Programming Patterns: State

Durumlar çoğaldığında kodu nasıl düzenli tutarız?

gameprogrammingpatterns.com/state.html

PATT VIRA

Collisions Without a Physics Library!

Zorlu görevdeki çoklu toplar için.

pattvira.com/coding-tutorials/v/collisions-without-a-physics-library

Bugün ne öğrendik?

- ✓ Menü, oyun, bitiş: üç durum ve dört geçişle tam bir oyun akışı.
- ✓ `draw` yönlendiricidir; her durumun işi kendi fonksiyonunda.
- ✓ Tuşla gelen geçişler `keyPressed` içinde, oyun kuralından doğanlar `guncelleOyun` içinde.
- ✓ Durum değişikliği bir sonraki frame'de ekrana yansır.
- ✓ `oyunuSifirLa` her turu temiz bir başlangıçla kurar.
- ✓ `while` ile deneme yaparken bir güvenlik sınırı koymak oyunu donmaktan korur.

SIRADAKİ DERS

XOX

İki oyunculu, sıra tabanlı bir oyun; tahtası iki boyutlu bir dizi.

Sonraki sunum →

← Önceki

