

Adım adım XOX

İki boyutlu dizi, fare tıklaması ve durum makinesi tek bir oyunda buluşuyor.

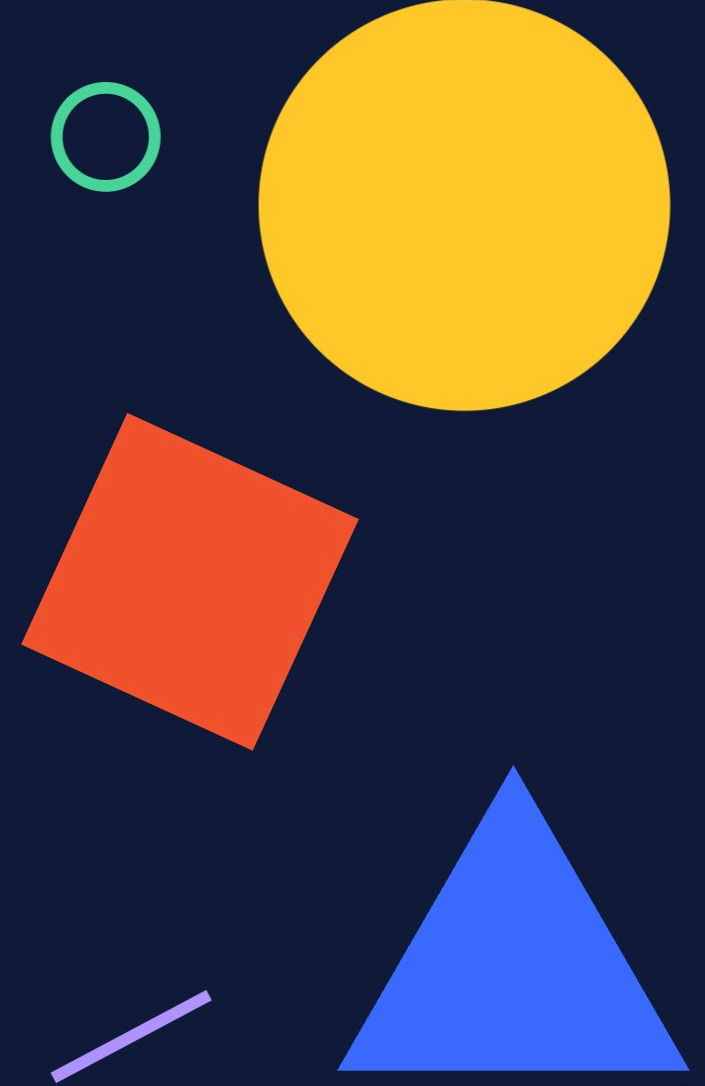
İki boyutlu dizi

Tahta ve taşlar

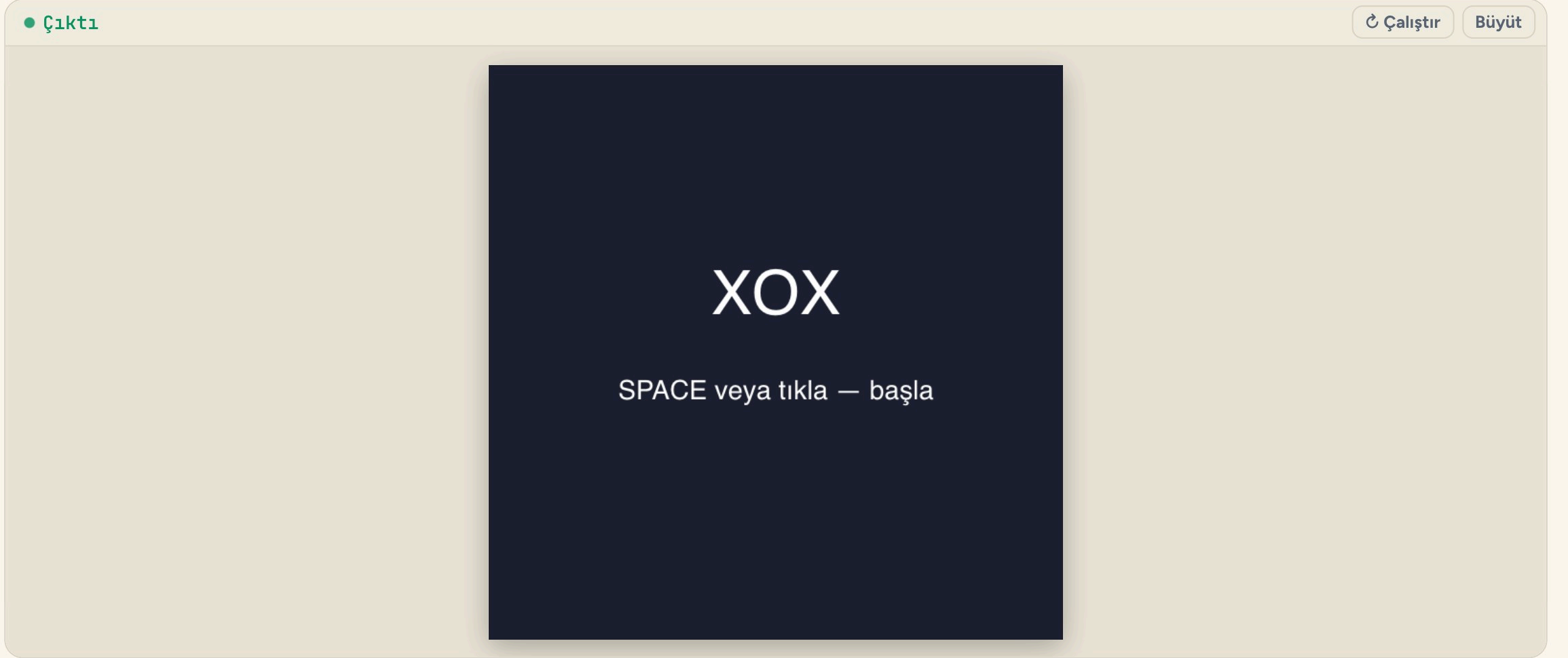
Tıklamadan hücreye

Kazanan ve beraberlik

Varyantlar



Önce bir el oynayalım



Tıklayarak başlayın; sırayla X ve O koyun. Dolu bir kareye tıklamayı deneyin. Oyun bitince tuvale tıklayıp R'ye basın.

XOX'un kuralları

XOX (Tic-Tac-Toe)

İki oyuncunun sırayla 3×3 ızgaraya X ve O koyduğu oyun. Aynı işaret yatay, dikey ya da çapraz üç hücreyi doldurursa o oyuncu kazanır. Bütün hücreler dolup kazanan çıkmazsa oyun berabere biter.

Kural	Açıklama
Tahta	3 satır $\times$ 3 sütun = 9 hücre
Sıra	Oyuncular sırayla oynar; X başlar
Geçerli hamle	Boş bir hücreye tıklamak
Geçersiz hamle	Dolu hücre ya da tahta dışı: yok sayılır
Kazanma	Aynı işaretle üç hücre: 8 olası hat
Beraberlik	9 hücre dolu, kazanan yok

SORU

Dokuz hücrenin içinde ne olduğunu program nerede tutacak?

Dokuz ayrı değişken mi? Dokuz elemanlı tek bir dizi mi? “Orta satırın sağ hücresi” dediğimizde hangisine bakacağız?



İki boyutlu dizi: dizilerin dizisi

sketch.js

Kopyala

Büyüt

```
1 let tahta = [  
2   ["", "", ""],  
3   ["", "", ""],  
4   ["", "", ""]  
5 ];
```

Dış dizinin üç elemanı var; her biri üç elemanlı bir dizi, yani bir **satır**.

Erişim hep aynı sırayla: `tahta[satır][sutun]`. **Önce satır, sonra sütun.**

İfade	Anlamı
<code>tahta[0]</code>	İlk satır (üç elemanlı dizi)
<code>tahta[1][2]</code>	2. satır, 3. sütun
<code>""</code>	Boş hücre
<code>"X"</code> / <code>"0"</code>	Dolu hücre: o karedeki taş

Satır ve sütun indeksleri

İki Boyutlu Dizi — tahta[satır][sütun]

		sütunlar		
		0	1	2
satırlar	0	X [0][0]	[0][1]	O [0][2]
	1	[1][0]	X [1][1]	[1][2]
	2	O [2][0]	[2][1]	[2][2]

```
let tahta = [  
  ["X", "", "O"],  
  ["", "X", ""],  
  ["O", "", ""]  
];
```

Boş hücre: "" (boş string)

Dolu hücre: "X" veya "O"

tahta[1][1] → "X"

tahta[0][1] === "" → true

İlk indeks satır, ikinci sütun

Her hücre boş metin ya da "X" / "O" tutar. İlk indeks satır, ikinci indeks sütun.

Fareyi hücrelerin üstünde gezdirin

● Çıktı

Çalıştır

Büyüt

	sütun 0	sütun 1	sütun 2
satır 0	X [0][0]	 [0][1]	O [0][2]
satır 1	 [1][0]	X [1][1]	 [1][2]
satır 2	O [2][0]	 [2][1]	 [2][2]

Bir hücrenin üstüne gelin

`tahta[1]` bütün bir satırı, `tahta[1][2]` o satırdaki tek bir hücreyi verir.

iki farklı “kare”

Frame

`draw()` fonksiyonunun bir kez çalışması. Oyun döngüsünün bir turu.

Hücre (kare)

Izgaradaki tek bir oyun alanı. Bu sunumda “kare” yalnızca tahta hücresi anlamında.

Boyutlar

`tahta.length` satır sayısını (3), `tahta[0].length` sütun sayısını verir. XOX hep 3×3 olduğu için döngülerde 0–2 aralığı yeter.

SORU

Aynı tahtayı 9 elemanlı düz bir dizide tutsak ne kaybederiz?

sketch.js

Kopyala

Büyüt

```
let tahta = ["", "", "", "", "", "", "", "", ""];
```

"2. satır, 3. sütun" hangi indeks? Peki hemen altındaki hücre?

Izgara dünyasının doğal dili

Düz dizide her seferinde bir formül gerekir:

`satir * 3 + sutun`. Izgara büyüdükçe hata ve okuma yükü artar.

`tahta[satir][sutun]`

"Burada ne var?" sorusunu doğrudan sorar.

- Komşular: `satir ± 1`, `sutun ± 1`.
- Kazanma çizgileri: aynı satır, aynı sütun, çapraz.
- "Bu sütunda aşağı düş" gibi kurallar.

Hepsi aynı indeks diline oturur. Birçok oyun dünyası bir satır-sütun ızgarasıdır: her hücre tek bir yanıt tutar.

Hücrede ne tutuluyor? (1/2)

XOX 3×3

Boş, X ya da O.

Satranç, dama 8×8

Boş ya da taşın türü.

Mayın Tarlası 9×9

Kapalı, açık, mayın,
bayrak.

Connect Four 6×7

Boş ya da renkli taş. Taş
sütunda aşağı kayar.

Sıra hep aynı: **önce tahtayı diziyle modelle, sonra çiz, sonra kuralları uygula.** XOX bu sıranın en küçük tam projesi.

Hücrede ne tutuluyor? (2/2)

Tetris 20 × 10

Boş ya da dolu blok.

2048, Candy Crush

Sayı, renk ya da boş.

Pac-Man labirenti

Yol, duvar, nokta, kapı.
Platform oyunlarındaki
karo haritalar (tile map)
da böyle.

Sudoku 9 × 9

1–9 arası bir sayı ya da
boş.

Üç ve daha fazla boyut

Bazen üçüncü bir indeks işe yarar: `katman[z][y][x]` ile Minecraft tarzı blok dünyası gibi. Ayrı bir sihir değil; ızgara fikrinin doğal uzantısı. Çoğu iki boyutlu oyun tek bir matris ve birkaç yardımcı diziyle ilerler.

Veri: sabitler, değişkenler, sıfırlama

sketch.js

Kopyala

Büyüt

```

1 const DURUM_MENU = "menu";
2 const DURUM_OYUN = "oyun";
3 const DURUM_BITIS = "bitis";
4 const HUCRE = 100;
5
6 let oyunDurumu;
7 let tahta;
8 let sıra;      // "X" veya "0" – şu an kim oynuyor
9 let sonuc;     // kazanan ("X"/"0") veya "beraberlik"

```

sketch.js

Kopyala

Büyüt

```

1 function oyunuSifirla() {
2   tahta = [
3     ["", "", ""],
4     ["", "", ""],
5     ["", "", ""]
6   ];
7   sıra = "X";
8   sonuc = "";
9 }

```

Değişken	Görevi
oyunDurumu	Menü, oyun ya da bitiş
tahta	3 × 3 hücre verisi
sıra	Sıradaki oyuncunun işareti
sonuc	Bitiş ekranında gösterilecek sonuç

createCanvas(300, 300) ve HUCRE = 100 : üç hücre tuvali tam kaplar. oyunuSifirla() tahtayı boşaltır, sırayı X'e verir.

Çizimi dört fonksiyona bölmek

Fonksiyon	Sorumluluğu
<code>cizTahtaCizgileri()</code>	Arka plan ve ızgara çizgileri
<code>taslariCiz()</code>	Dolu hücreleri dolaşır, her biri için <code>tasCiz</code> 'i çağırır
<code>tasCiz(x, y, boyut, deger)</code>	Tek bir hücreye X ya da O yazar
<code>cizOyunEkrani()</code>	Tahta + taşlar + sıra yazısı

Tanıdık bir yapı: Bölüm 2'deki `tahtaCiz` ve `hucresCiz`. O zaman "ileride XOX'ta kullanacağız" demiştik.

Tahta çizgileri ve tek bir taş

cizTahtaCizgileri()

Kopyala

Büyüt

```
1 function çizTahtaCizgileri() {  
2   background(245, 242, 235);  
3   stroke(40);  
4   strokeWeight(4);  
5   line(HUCRE, 0, HUCRE, height);  
6   line(HUCRE * 2, 0, HUCRE * 2, height);  
7   line(0, HUCRE, width, HUCRE);  
8   line(0, HUCRE * 2, width, HUCRE * 2);  
9   noStroke();  
10 }
```

tasCiz()

Kopyala

Büyüt

```
1 function tasCiz(x, y, boyut, deger) {  
2   textAlign(CENTER, CENTER);  
3   textSize(boyut * 0.55);  
4   if (deger === "X") {  
5     fill(200, 50, 50);  
6   } else {  
7     fill(50, 80, 200);  
8   }  
9   text(deger, x + boyut / 2, y + boyut / 2);  
10 }
```

Dört `line` : iki dikey, iki yatay. Taş, hücrenin ortasına `textAlign(CENTER, CENTER)` ile yazılan bir harf; X kırmızı, O mavi.

ADIM 2

Diziyi gez, dolu hücreleri çiz

sketch.js · satır 8-22

Kopyala

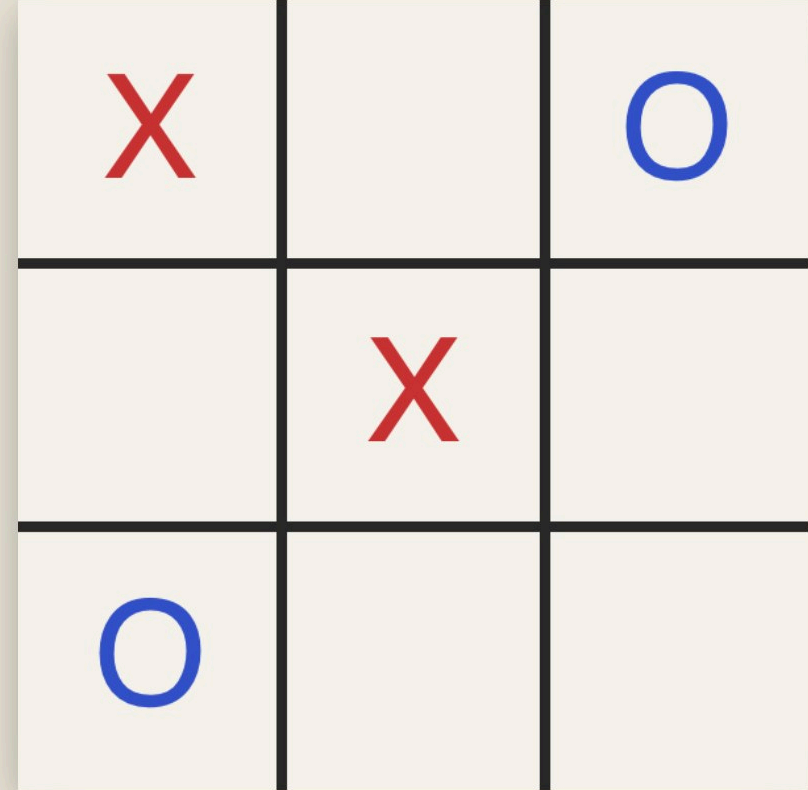
Büyüt

```
8 function draw() {  
9   cizTahtaCizgileri();  
10  taslariCiz();  
11 }  
12  
13 function taslariCiz() {  
14   for (let satir = 0; satir < 3; satir = satir + 1) {  
15     for (let sutun = 0; sutun < 3; sutun = sutun + 1) {  
16       let deger = tahta[satir][sutun];  
17       if (deger !== "") {  
18         tasCiz(sutun * HUCRE, satir * HUCRE, HUCRE, deger);  
19       }  
20     }  
21   }  
22 }
```

Çıktı

Çalıştır

Büyüt



Veri ile görüntü ayrı: çizim her frame dizinin o anki hâlini yansıtır. Kodu kopyalayıp dizideki bir `""` yerine `"O"` yazın; çizim kendiliğinden değişir.

SORU

Oyuncu (150, 220) pikseline tıkladı. Hangi hücre?

Fare bize piksel veriyor; dizimiz ise satır ve sütun istiyor. Hücreler 100 piksel. Pikselden indekse nasıl geçeriz?

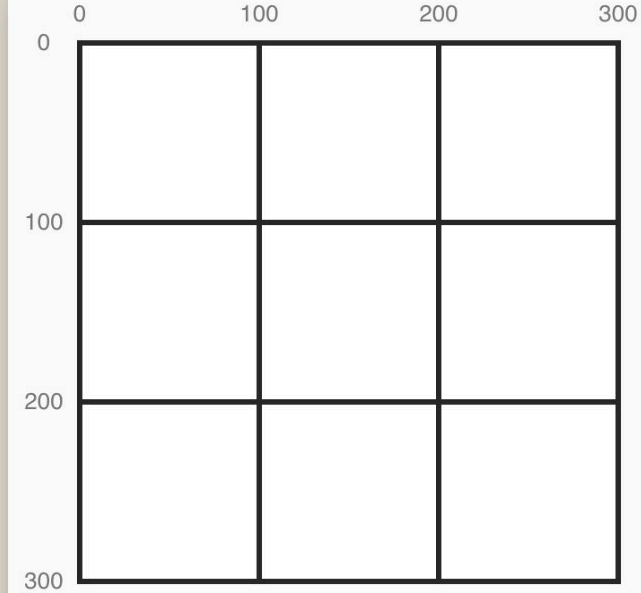


Böl ve aşağı yuvarla

● Çıktı

Çalıştır

Büyüt



Fareyi tahtanın üstünde gezdirin

sketch.js

Kopyala

Büyüt

```
let sutun = floor(mouseX / HUCRE);  
let satir = floor(mouseY / HUCRE);
```

Bir tıklamanın yolculuğu

Adım	Kontrol	Sonuç
1	Menüde miyiz?	Evetse <code>oyunuSifirla()</code> , oyuna geç ve çık
2	Oyun durumunda mıyız?	Değilse <code>return</code>
3	<code>satir</code> ve <code>sutun</code> 0–2 aralığında mı?	Değilse <code>return</code>
4	Hücre boş mu?	Doluysa <code>return</code>
5	Hamleyi yaz	<code>tahta[satir][sutun] = sıra</code>
6	Kazanan ya da beraberlik var mı?	Varsa bitişe geç
7	Sırayı diğer oyuncuya ver	$X \leftrightarrow O$

İlk dört adım birer **kapı bekçisi**: geçersiz her tıklama erken `return` ile elenir. Asıl iş tek satır.

mousePressed (1/2): bekçiler ve hamle

sketch.js · satır 1-20

Kopyala

Büyüt

```
1 function mousePressed() {
2   if (oyunDurumu === DURUM_MENU) {
3     oyunuSifirla();
4     oyunDurumu = DURUM_OYUN;
5     return;
6   }
7   if (oyunDurumu !== DURUM_OYUN) {
8     return;
9   }
10
11   let sutun = floor(mouseX / HUCRE);
12   let satir = floor(mouseY / HUCRE);
13   if (satir < 0 || satir > 2 || sutun < 0 || sutun > 2) {
14     return;
15   }
16   if (tahta[satir][sutun] !== "") {
17     return;
18   }
19
20   tahta[satir][sutun] = sıra;
```

- Menüde tıklamak oyunu başlatır.
- Bitiş ekranında tıklama yok sayılır: “duruma göre girdi”.
- Tahta dışı ve dolu hücre elenir.
- Geriye kalan geçerli bir hamledir: sıradaki işaret diziye yazılır.
- Çizim için hiçbir şey yapmıyoruz; bir sonraki `draw` yeni taşı kendisi çizecek.

mousePressed (2/2): sonuç ve sıra

sketch.js · satır 22-39

Kopyala

Büyüt

```
22 let kazanan = kazananKontrol();
23 if (kazanan !== "") {
24     sonuc = kazanan;
25     oyunDurumu = DURUM_BITIS;
26     return;
27 }
28 if (tahtaDoluMu()) {
29     sonuc = "beraberlik";
30     oyunDurumu = DURUM_BITIS;
31     return;
32 }
33
34 if (sira === "X") {
35     sira = "O";
36 } else {
37     sira = "X";
38 }
39 }
```

- Her hamleden sonra **önce** kazanan, **sonra** beraberlik kontrol edilir.
- Oyun bittiyse sıra değiştirilmez; `return` ile çıkılır.
- Oyun sürüyorsa sıra `if / else` ile X ve O arasında değişir.

SORU

Üçlü olmanın kaç farklı yolu var?

Yatay, dikey, çapraz... Hepsini sayın. Program bunların hepsine tek tek bakmak zorunda mı?



Sekiz kazanma hattı

● Çıktı

Çalıştır

Büyüt

[0][0]	[0][1]	[0][2]
X	X	X
[2][0]	[2][1]	[2][2]

2 / 8

Yatay · satır 1

 $[1,0] - [1,1] - [1,2]$ $3 \text{ yatay} + 3 \text{ dikey} + 2 \text{ çapraz} = 8 \text{ hat}$

Kazanma koşulu: üç hücrede aynı işaret **ve** o hücrelerin boş olmaması.

kazananKontrol : iki döngü, iki if

sketch.js

Kopyala

Büyüt

```
1 function kazananKontrol() {  
2   for (let i = 0; i < 3; i = i + 1) {  
3     if (tahta[i][0] !== "" && tahta[i][0] === tahta[i][1] && tahta[i][1] === tahta[i][2]) {  
4       return tahta[i][0];  
5     }  
6   }  
7   for (let j = 0; j < 3; j = j + 1) {  
8     if (tahta[0][j] !== "" && tahta[0][j] === tahta[1][j] && tahta[1][j] === tahta[2][j]) {  
9       return tahta[0][j];  
10    }  
11  }  
12  if (tahta[0][0] !== "" && tahta[0][0] === tahta[1][1] && tahta[1][1] === tahta[2][2]) {  
13    return tahta[0][0];  
14  }  
15  if (tahta[0][2] !== "" && tahta[0][2] === tahta[1][1] && tahta[1][1] === tahta[2][0]) {  
16    return tahta[0][2];  
17  }  
18  return "";  
19 }
```

kazananKontrol nasıl okunur?

ilk for

Üç yatay satır: `tahta[i][0]`, `[i][1]`, `[i][2]`.

ikinci for

Üç dikey sütun: `tahta[0][j]`, `[1][j]`, `[2][j]`.

iki if

İki çapraz. Kazanan yoksa `""` döner.

Fonksiyon kazananın işaretini (`"X"` ya da `"0"`) döndürür. İlk bulduğu üçlüde `return` ile çıkar; geri kalan hatlara bakmaz.

Sekiz hat, dört kontrol

Döngü değişkeni `i` satırı, `j` sütunu gezdiği için üç yatay ve üç dikey hat ikişer satıra sığıyor.

SORU

Koşuldaki `!== ""` kısmını silersek ne olur?

sketch.js

Kopyala

Büyüt

```
if (tahta[i][0] === tahta[i][1] && tahta[i][1] === tahta[i][2]) {  
  return tahta[i][0];  
}
```

Oyunun en başında tahta nasıl görünüyor?

Üç boş hücre de “birbirine eşit”



TEBRİKLER! KAZANAN:

Boş tahtada üst satır: "" == "" == "". Üçlü sayılır ve fonksiyon "" döndürür.

Şansımıza oyun bozulmaz, ama...

Dönen değer "" olduğu için “kazanan yok” gibi görünür. Fakat fonksiyon ilk satırda çıktığı için diğer hatlara **hiç bakmaz**. Üst satır boşken alt satırdaki gerçek üçlü görülmez.

Kural

Üç hücre birbirine eşit **ve** boş değil.

Beraberlik ve bitiş ekranı

tahtaDoluMu()

Kopyala

Büyüt

```
1 function tahtaDoluMu() {  
2   for (let satir = 0; satir < 3; satir = satir + 1) {  
3     for (let sutun = 0; sutun < 3; sutun = sutun + 1) {  
4       if (tahta[satir][sutun] === "") {  
5         return false;  
6       }  
7     }  
8   }  
9   return true;  
10 }
```

cizBitisEkrani()

Kopyala

Büyüt

```
1 function cizBitisEkrani() {  
2   background(35, 30, 40);  
3   fill(255);  
4   textAlign(CENTER, CENTER);  
5   textSize(22);  
6   if (sonuc === "beraberlik") {  
7     text("Beraberlik!", width / 2, height / 2 - 20);  
8   } else {  
9     text("Kazanan: " + sonuc, width / 2, height / 2 - 20);  
10  }  
11  textSize(14);  
12  text("R - menüye dön", width / 2, height / 2 + 30);  
13 }
```

Tek bir boş hücre bulunca `false` döner ve aramayı bırakır. Tahta dolu ve kazanan yoksa beraberlik. Bitiş ekranı `sonuc` 'a bakıp mesajı seçer.

XOX'un geiş tablosu

Mevcut durum	Olay / kořul	Yeni durum	Ek iřlem
DURUM_MENU	SPACE ya da tıklama	DURUM_OYUN	oyunuSifirla()
DURUM_OYUN	Geerli tıklama, kazanan var	DURUM_BITIS	sonuc = kazanan
DURUM_OYUN	Geerli tıklama, tahta dolu	DURUM_BITIS	sonuc = "beraberlik"
DURUM_OYUN	Geerli tıklama, oyun sryor	DURUM_OYUN	Sıra X ↔ O
DURUM_OYUN	Geersiz tıklama	DURUM_OYUN	—
DURUM_BITIS	R basıldı	DURUM_MENU	—

Omurga Ka-Ka Top ile aynı. X ve O sırası ayrı birer durum deėil; oyun durumunun iinde `sıra` deėiřkeninde tutulan alt bilgi.

setup ve yönlendirici draw

sketch.js · satır 11-25

Kopyala

Büyüt

```
11 function setup() {  
12   createCanvas(300, 300);  
13   textFont("sans-serif");  
14   oyunDurumu = DURUM_MENU;  
15 }  
16  
17 function draw() {  
18   if (oyunDurumu === DURUM_MENU) {  
19     cizMenu();  
20   } else if (oyunDurumu === DURUM_OYUN) {  
21     cizOyunEkrani();  
22   } else if (oyunDurumu === DURUM_BITIS) {  
23     cizBitisEkrani();  
24   }  
25 }
```

Çıktı

Çalıştır

Büyüt



XOX

SPACE veya tıkla — başla

Menü ve oyun ekranı

sketch.js · satır 27-45

Kopyala

Büyüt

```
27 function cizMenu() {
28   background(28, 32, 48);
29   fill(255);
30   textAlign(CENTER, CENTER);
31   textSize(32);
32   text("XOX", width / 2, height / 2 - 30);
33   textSize(14);
34   text("SPACE veya tıkla — başla", width / 2, height / 2 + 20);
35 }
36
37 function cizOyunEkrani() {
38   cizTahtaCizgileri();
39   taslariCiz();
40   fill(30);
41   textSize(14);
42   textAlign(LEFT, TOP);
43   noStroke();
44   text("Sıra: " + sıra, 8, 8);
45 }
```

Çıktı

Çalıştır

Büyüt



Klavye: başlat ve menüye dön

sketch.js · satır 165-183

Kopyala

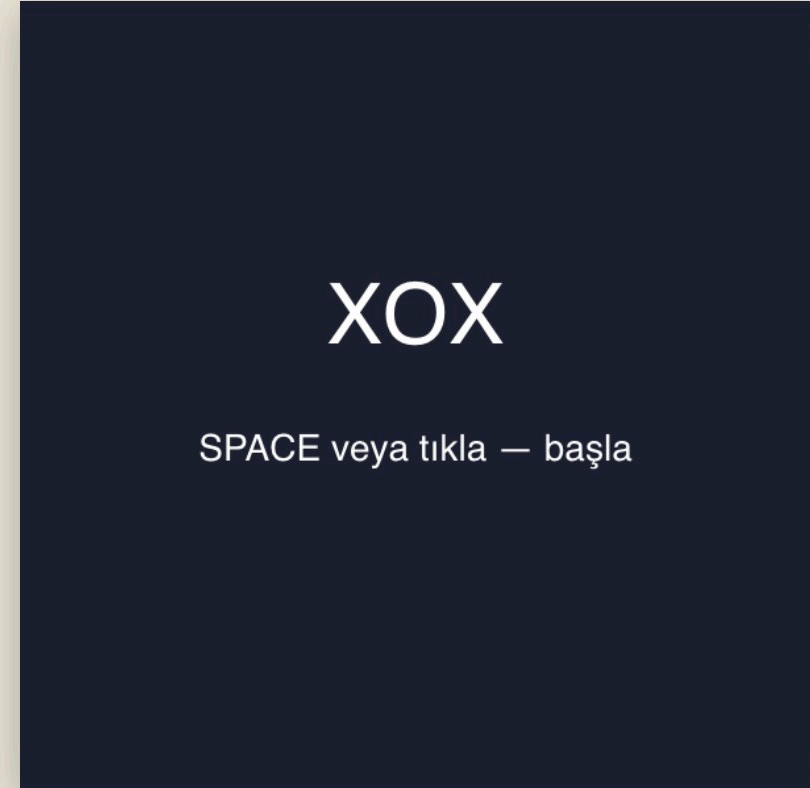
Büyüt

```
165 function oyunuSifirla() {
166   tahta = [
167     [ "", "", "" ],
168     [ "", "", "" ],
169     [ "", "", "" ]
170   ];
171   sirasira = "X";
172   sonuc = "";
173 }
174
175 function keyPressed() {
176   if (oyunDurumu === DURUM_MENU && key === " ") {
177     oyunuSifirla();
178     oyunDurumu = DURUM_OYUN;
179   }
180   if (oyunDurumu === DURUM_BITIS && (key === "r" || key === "R")) {
181     oyunDurumu = DURUM_MENU;
182   }
183 }
```

● Çıktı

Çalıştır

Büyüt



Menüden oyuna hem SPACE hem tıklama ile geçilir; ikisi de `oyunuSifirla()` 'yı çağırır. Hamleler fareyle, menü geçişleri klavyeyle.

Bu projede her şey var

Bölüm 1

Koordinat, `text`, `if`
`/ else`,
`mousePressed`,
`return`.

Bölüm 2

İç içe `for`, modüler
çizim fonksiyonları,
oyun döngüsü.

Bölüm 3

Diziler, `floor`,
`keyPressed`.

Bölüm 4

Durum makinesi,
geçiş tablosu, iki
boyutlu dizi.

Sırada ne var?

Aynı `tahta[satır][sutun]` iskeletiyle bilgisayar rakibi (Minimax), Connect Four, Yılan ve Mayın Tarlası. Onlar İleri Teknikler kitabının konusu.

Yapay zekâyla: ızgara oyunları

 15-25 dk

1. **Önce siz:** bildiğiniz iki oyun seçin. Her biri için bir cümle: “Bu oyunda ızgaranın bir hücresi ne tutuyor?”
2. **Sorun:** “Oyun programlamada iki boyutlu dizi neden sık kullanılır? En az 6 bilinen oyunda tahta nasıl ızgara gibi düşünülür, hücrede ne tutulur? Tek boyutlu diziyle modellemek ne zaman zahmetli olur? Üç boyutlu dizinin işe yaradığı bir iki örnek ver. Kod yazma.”
3. **Doğrulayın:** verdiği örneklerden ikisini kısa bir aramayla kontrol edin. Uydurma oyun ya da yanlış tahta boyutu var mı?
4. Kendi cümlelennizle yazın: yapmak istediğiniz küçük bir oyunda ızgara kullanır mısınız? Hücrede ne tutarsınız?

Yapay zekâyla: XOX varyantlarını keşfedin

 15-20 dk

1. Çevrimiçi bir iki el klasik XOX oynayın; Wikipedia'daki varyant listesine göz atın.
2. **Sorun:** "Elimde klasik 3×3 XOX var. Bu iskeletle kodlanabilecek en az 4 varyantı anlat. Hangisi yalnızca sembol değiştirmekle biter, hangisi kazanma kontrolünü ya da tahta boyutunu değiştirmeyi gerektirir? Kod yazma; Minimax anlatma."
3. Bir varyant seçin. Kâğıda, kodda nelerin değişeceğini en az iki maddeyle yazın. Bakılacak yerler: `sıra` ve taş metinleri, `tasCiz`, `kazananKontrol`, `createCanvas` ve `HUCRE`, bitiş mesajı.

Kendi XOX varyantınızı yazın

50-80 dk

Seçenek	Ne değişir?	Zorluk
A · Sembol	X ve O yerine A / B, ★ / ● ya da iki kısa kelime. Menü ve bitiş yazıları uyumlu.	Kolay
B · Ters XOX	Üçlü yapan kaybeder . Bitiş yazısı ve sonuc mantığı tersine döner.	Orta
C · 4 × 4	Tahta 4 × 4; yine üç aynı işaret kazandırır. HUCRE ve döngü sınırları büyür.	Orta-zor
D · Serbest işaret	Oyuncu önce taşı seçer (1 → X, 2 → O), sonra boş hücreye tıklar.	Orta-zor

- 1. Plan:** hangi sabit ve değişken değişir? **kazananKontrol** aynı mı kalır? Bitişte ne yazmalı?
- 2. Kod:** kitaptaki XOX'un bir kopyasında önce klasik oyunun çalıştığını görün; sonra yalnızca gereken satırları değiştirin.
- 3. Takılınca:** yapay zekâdan önce açıklama, sonra yalnızca gerekli satırların düzeltmesini isteyin.

Yapay zekâsız deneyin

 20 dk

1. Üç ızgara oyunu seçin. Her biri için yazın: satır $\times$ sütun tahmini ve hücrede tutulan veri. Hangisinde `tahta[satir][sutun]` doğrudan işe yarar?
2. XOX kopyasında X ve O yerine kendi iki sembolünüzü kullanın; menü ve bitiş yazılarını uyumlu yapın. Bir tam tur oynayın.
3. Bu değişiklik `kazananKontrol` 'e neden hiç dokunmadan çalışıyor?
4. Ters XOX için `mousePressed` ve bitiş dalında neyi tersine çevireceğinizi üç maddeyle planlayın.

HUCRE = 100. Oyuncu (299, 0) pikseline tıkladı. Hangi hücre?

A `tahta[2][0]`

✓ `tahta[0][2]`

C `tahta[3][0]`

D `tahta[0][3]`

sutun = $\text{floor}(299 / 100) = 2$, satir = $\text{floor}(0 / 100) = 0$. Dizide önce satır yazılır: `tahta[0][2]`, sağ üst köşe.

Sıra değiştirme satırlarını kazanan kontrolünün önüne alırsak ne bozulur?

A Hiçbir şey; sıra önemli değil.

B Kazanan hiç bulunamaz.



Kazanan doğru bulunur ama ekranda "Sıra" yazısı gereksiz yere değişir; ileride `sıra` 'ya güvnen kod yanlış oyuncuyu gösterir.

`kazananKontrol` tahtaya bakıyor, `sıra` 'ya değil; o yüzden kazanan doğru çıkar. Ama "kazanan = sıra" diye yazılmış bir sürümde yanlış oyuncu kazanırdı. Sıra, oyun sürüyorsa değişmeli.

`tahta[1].length` ve `tahta.length` sırasıyla neyi verir?

- ☐ A Satır sayısı ve sütun sayısı
- ☒ B Bir satırdaki sütun sayısı ve satır sayısı
- ☐ C İkisi de toplam hücre sayısı: 9

`tahta` satırların dizisi: uzunluğu satır sayısı. `tahta[1]` tek bir satır: uzunluğu o satırdaki hücre sayısı.

Merak edenler için kaynaklar

PATT VİRA

2D Array

İki boyutlu dizileri görsel örneklerle anlatıyor.

pattvira.com/intro-to-creative-coding/2d-array

WIKİPEDIA

List of tic-tac-toe variants

Varyant projesi için fikir kaynağı.

en.wikipedia.org/wiki/List_of_tic-tac-toe_variants

XKCD 832

Tic-Tac-Toe

Her durumda en iyi hamleyi gösteren tek sayfalık harita.

Bilgisayar rakibi yazmayı düşünenlere.

xkcd.com/832

ÇEVİRİMİÇİ OYUN

playtictactoe.org

Klasik kuralları birkaç el deneyin.

playtictactoe.org

Kitap 1'i bitirdik

- ✓ İki boyutlu dizi: `tahta[satır][sutun]`. Önce satır, sonra sütun.
- ✓ Izgara oyunlarının ortak sırası: veriyi modelle, çiz, kuralları uygula.
- ✓ Pikselden hücreye: `floor(mouseX / HÜCRE)` ve `floor(mouseY / HÜCRE)`.
- ✓ Geçersiz tıklamalar erken `return` ile elenir.
- ✓ Sekiz kazanma hattı; hücreler eşit **ve** boş değil. Önce kazanan, sonra beraberlik.
- ✓ Menü, oyun, bitiş: aynı durum makinesi omurgası.

KİTAP 1 TAMAMLANDI

Bir tuvalden tam bir oyuna

Gri bir kareyle başladık; menüsü, kuralları ve kazananı olan bir oyunla bitirdik.

Tüm sunumlar

← Önceki