

Klavye: kontrol oyuncuya geçiyor

Basılı tutunca yürümek, bir kez basınca zıplamak. İkisi aynı şey değil.

keyIsDown

keyPressed

key ve keyCode

ASCII

Kontrol edilen kare

SORU

Bir oyunda yürümek ile zıplamak arasında klavye açısından ne fark var?

Sağ oka basılı tutuyorsunuz: karakter yürümeye devam ediyor.
Boşluğa basılı tutuyorsunuz: karakter durmadan zıplıyor mu?



İki katman: durum ve olay

Klavye girdisi

Oyuncunun tuşa basarak programa ilettiği kontrol bilgisidir.

Girdi	Sürekli (basılı tutma)	Tek seferlik (basma anı)
Fare	<code>mouseIsPressed</code> , <code>mouseX</code> / <code>mouseY</code>	<code>mousePressed()</code>
Klavye	<code>keyIsDown(...)</code>	<code>keyPressed()</code>

Tanıdık bir ayrım: Daire Tıklama'da skoru `mouseIsPressed` ile değil `mousePressed()` ile saymıştık. Klavyede de aynı iki katman var.

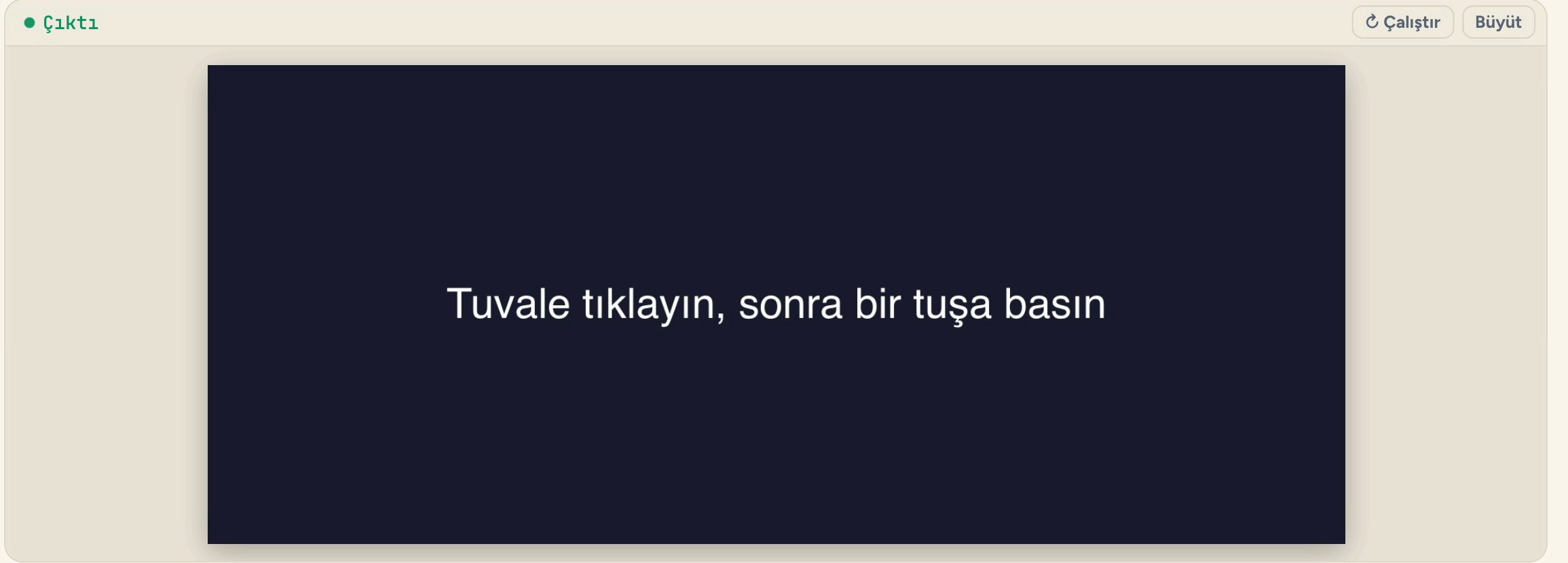
Üç araç, üç farklı an

Araç	Ne zaman çalışır?	Tipik kullanım
<code>draw()</code> + <code>keyIsDown()</code>	Tuş basılı olduğu sürece, her frame	Sürekli yürüme, hızlanma
<code>keyPressed()</code>	Tuşa basıldığı an , bir kez	Zıplama, ateş, menü seçimi, turu yenileme
<code>keyReleased()</code>	Tuş bırakıldığı an , bir kez	Şarj biriktirip bırakınca ateş etme

Adlar sabit

`keyPressed` ve `keyReleased`, p5.js'in kendisinin çağırdığı olay fonksiyonlarıdır. `setup` ve `draw` gibi tam olarak böyle yazılmalıdır.

Önce tuvale tıklayın, sonra bir tuşa basın



Tuşlar çalışmıyorsa

Tuval “odakta” değildir. Önizlemeye bir kez tıklayın; yoksa tuşlar tarayıcıya ya da editöre gider.

keyIsDown() : şu an basılı mı?

keyIsDown(tuş)

Verilen tuş o frame'de basılıysa `true`, değilse `false` döndürür. `draw()` içinde çağrılır.

Tuş	Yazımı
Sol ok	<code>keyIsDown(LEFT_ARROW)</code>
Sağ ok	<code>keyIsDown(RIGHT_ARROW)</code>
Yukarı ok	<code>keyIsDown(UP_ARROW)</code>
Aşağı ok	<code>keyIsDown(DOWN_ARROW)</code>
Harf (A, D)	<code>keyIsDown(65)</code> , <code>keyIsDown(68)</code>

Ok tuşları için hazır sabitler var. Harf tuşlarında `keyIsDown` bir sayı ister; nedenini birazdan göreceğiz.

Ok tuşlarıyla yatay hareket

sketch.js · satır 11-25

Kopyala

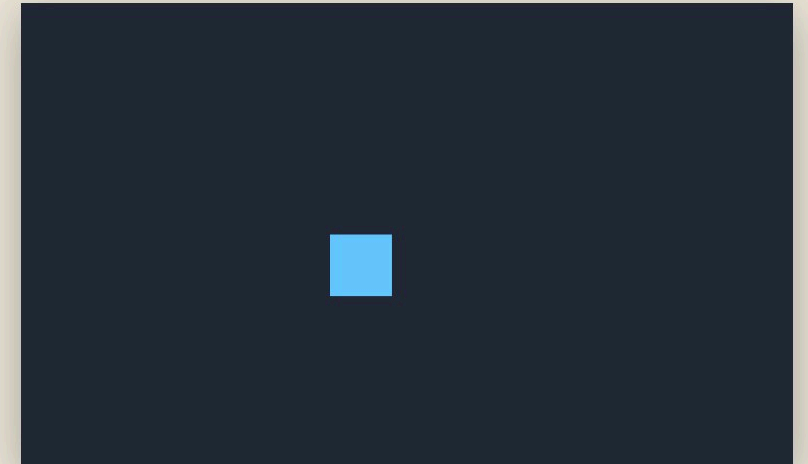
Büyüt

```
11 function draw() {  
12   background(35, 40, 55);  
13  
14   let dt = deltaTime / 1000;  
15  
16   if (keyIsDown(RIGHT_ARROW)) {  
17     kareX = kareX + HIZ * dt;  
18   }  
19   if (keyIsDown(LEFT_ARROW)) {  
20     kareX = kareX - HIZ * dt;  
21   }  
22  
23   fill(100, 200, 255);  
24   rect(kareX, kareY, KARE_BOYUT, KARE_BOYUT);  
25 }
```

● Çıktı

Çalıştır

Büyüt



Tuvale tıklayıp sağ ve sol oka basılı tutun. Hız yine `deltaTime` ile: saniyede 180 piksel.

SORU

Aynı satırı `keyPressed()` içine yazarsak kare nasıl hareket eder?

sketch.js

Kopyala

Büyüt

```
function keyPressed() {  
  if (keyCode === RIGHT_ARROW) {  
    kareX = kareX + 20;  
  }  
}
```

Sağ oka basılı tutuyorsunuz...

Akıcı yürümez; her basışta bir adım sıçrar

Tuşa basılı tutuyorum,
neden yürümüyor?



oyuncu

Bana bir kez haber
verildi. Bir adım attım.



keyPressed()

Sık yapılan hata

`keyPressed()` "tuşla ilgili her şeyin yazıldığı yer" gibi görünür. Ama yalnızca basma anında bir kez tetiklenir.

Kural

Sürekli hareket her zaman `draw()` içinde, `keyIsDown()` ile.

Dikey hareket ve çapraz gitmek

sketch.js

Kopyala

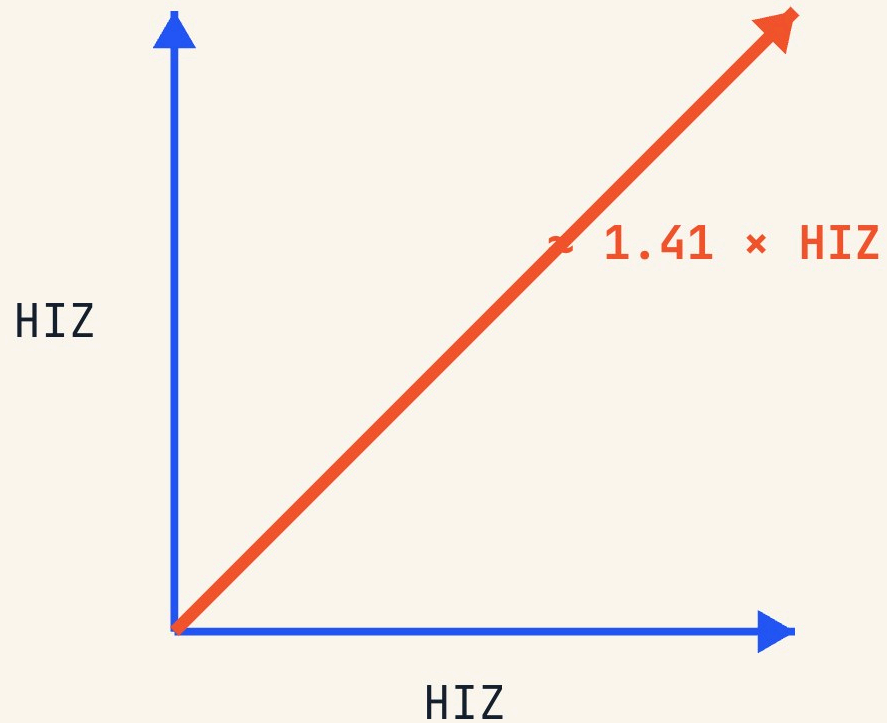
Büyüt

```
1  if (keyIsDown(DOWN_ARROW)) {  
2    kareY = kareY + HIZ * dt;  
3  }  
4  if (keyIsDown(UP_ARROW)) {  
5    kareY = kareY - HIZ * dt;  
6  }
```

Dört ayrı `if` birbirinden bağımsızdır. Sağ ve yukarı aynı anda basılıysa ikisi de çalışır: kare çapraz gider. `else if` yalnızca tek yön seçilecekse uygundur.

Basılı tuşlar	x	y	Etki
Yalnızca →	+	0	Yatay
Yalnızca ↑	0	-	Dikey
→ ve ↑	+	-	Çapraz, sağ üst

Çapraz koşmak neden daha hızlı?



İki ekseninde de **HIZ** kadar ilerleyince köşegen, Pisagor'a göre $\sqrt{2} \approx \mathbf{1.41 \text{ kat}}$ daha uzun olur.

Doom 1993

Hem ileri hem yana basarak koşmak (strafe-running) düz koşmaktan belirgin biçimde hızlıydı. Hız rekortmenleri hâlâ bu yöntemi kullanır.

Rekabetçi oyunlarda hız **normalize edilir**: iki bileşen birim vektöre göre ölçeklenir, sonra **HIZ** ile çarpılır. Basit örneklerimizde buna gerek yok.

keyPressed() ve key

sketch.js

Kopyala

Büyüt

```
1 function keyPressed() {  
2   if (key === "r" || key === "R") {  
3     // turu sıfırla  
4   }  
5 }
```

key

Basılan tuşun **karakter** değerini tutan değişken: 'a', 'R', ' ' (boşluk). Yalnızca olay fonksiyonu çalıştığı anda anlamlıdır.

"r" ile "R" farklı karakterlerdir. Caps Lock açık olabilir; ikisini de kontrol edin.

keyPressed() ile mousePressed() aynı mantığı izler: olay bir kez tetiklenir. Zıplama, tek atış, menüde seçim gibi "her basışta bir eylem" işlerine uygundur.

Boşluk tuşu: rengi değiştir

sketch.js

Kopyala

Düzenle

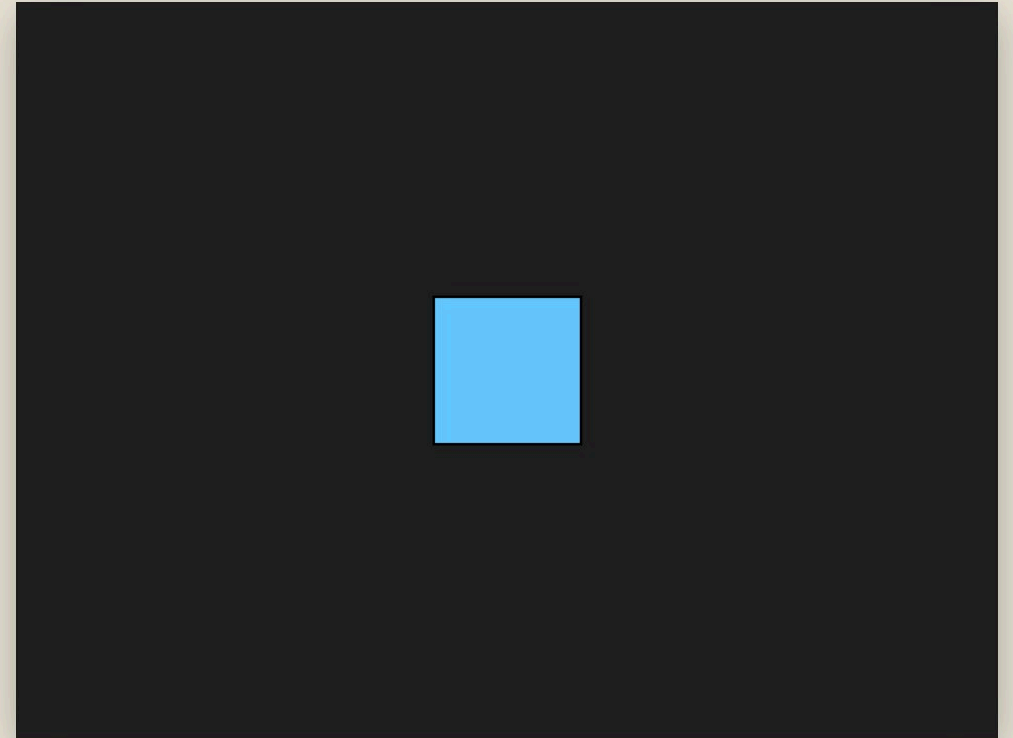
Büyüt

```
1 let kareRengi;
2
3 function setup() {
4   createCanvas(400, 300);
5   kareRengi = color(100, 200, 255);
6 }
7
8 function draw() {
9   background(30);
10  fill(kareRengi);
11  rect(width / 2 - 30, height / 2 - 30, 60, 60);
12 }
13
14 function keyPressed() {
15   if (key === " ") {
16     if (red(kareRengi) > 200) {
17       kareRengi = color(100, 200, 255);
18     } else {
19       kareRengi = color(255, 120, 80);
20     }
21   }
22 }
```

Çıktı

Çalıştır

Büyüt



Tuvale tıklayıp boşluğa basın. Basılı tutsanız da renk bir kez değişir. `red()` bir rengin kırmızı bileşenini okur.

keyReleased() : basılı tut, şarj et, bırak, ateşle

● Çıktı

Çalıştır

Büyüt

Tuvale tıklayın · BOŞLUK'a basılı tutun, sonra bırakın



keyIsDown → şarj: 0 · keyReleased → ateş · ekrandaki mermi: 0

Mega Man'ın şarjlı atışı gibi: basılı tutma `keyIsDown`, bırakma anı `keyReleased`. Mermiler de bir dizide duruyor.

Kontrol şeması: bir uzay gemisi

Tuş	Yöntem	Eylem
Ok / WASD	<code>keyIsDown</code> + <code>draw()</code>	Sürekli hareket
SPACE	<code>keyPressed()</code>	Tek mermi
R	<code>keyPressed()</code>	Turu sıfırla
ESC	<code>keyPressed()</code>	Menüye dön

WASD nereden geldi?

1990'ların sonunda Quake oyuncularını sol eli WASD'ye, sağ eli fareye koymaya başladı. Düzeni yaygınlaştıran isim, turnuva şampiyonu Dennis "Thresh" Fong'du. Bugün neredeyse her PC oyununun varsayılanı.

Ok tuşlarını da WASD'yi de birlikte desteklerseniz farklı alışkanlıklara sahip oyuncular aynı oyunu oynayabilir.

Tek seferlik eylem için neden `keyPressed()` ?

Doğrudan `keyPressed()`

Niyet açıkça görünür: “bu tuşa basılınca şu olsun.” p5.js, “önceki frame’de basılı değildi, şimdi basılı” karşılaştırmasını sizin yerinize yapar.

Alternatif: `keyIsDown` + bayrak

`draw()` içinde bir “önceki durum” değişkeni tutarak aynı şey elle yapılabilir. Bazı ileri senaryolarda gerekir; ama fazladan değişken ve karşılaştırma ister.

Basit ve orta düzey oyunlarda doğrudan `keyPressed()` tercih edilir.

SORU

keyIsDown(65) yazınca A tuşu algılanıyor. 65 nereden geliyor?

Bilgisayar "A" harfini aslında nasıl saklıyor olabilir?



Her karakterin bir numarası var: ASCII

ASCII tablosu — yazdırılabilir karakterler (32–126)

0–31 denetim kodlarıdır (görünmez); Unicode'un ilk 128 kodu bu tabloyla aynıdır

32 SPC	33 !	34 "	35 #	36 \$	37 %	38 &	39 '	40 (	41)	42 *	43 +	44 ,	45 -	46 .	47 /
48 0	49 1	50 2	51 3	52 4	53 5	54 6	55 7	56 8	57 9	58 :	59 ;	60 <	61 =	62 >	63 ?
64 @	65 A	66 B	67 C	68 D	69 E	70 F	71 G	72 H	73 I	74 J	75 K	76 L	77 M	78 N	79 O
80 P	81 Q	82 R	83 S	84 T	85 U	86 V	87 W	88 X	89 Y	90 Z	91 [	92 \ \	93]	94 ^	95 _
96 ,	97 a	98 b	99 c	100 d	101 e	102 f	103 g	104 h	105 i	106 j	107 k	108 l	109 m	110 n	111 o
112 p	113 q	114 r	115 s	116 t	117 u	118 v	119 w	120 x	121 y	122 z	123 {	124 	125 }	126 ~	

 Rakamlar (48–57)

 Büyük harf (65–90)

 Küçük harf (97–122)

Örnek: 'A' → 65 · 'a' → 97 · '0' → 48 · boşluk → 32

ASCII: American Standard Code for Information Interchange. 0–31 görünmeyen denetim kodları, 32–126 yazdırılabilir karakterler.

Tabloyu okumak

Aralık	Karakterler	Anlamı
48 ... 57	'0' ... '9'	Rakamlar
65 ... 90	'A' ... 'Z'	Büyük harfler
97 ... 122	'a' ... 'z'	Küçük harfler
32	boşluk	Space

sketch.js

Kopyala

Büyüt

```
'A'.charCodeAt(0) // 65  
'a'.charCodeAt(0) // 97
```

Harf ve rakam tuşlarında `key` doğrudan karakteri verir: `'A'`, `'w'`. Numarasını görmek için `charCodeAt`.

Küçük bir ayrntı

Rakam `'5'` ile sayı 5 aynı şey değil: karakterin numarası 53.

key mi, keyCode mu?

key

Hangi karakter? Harf, rakam ve sembol tuşlarında okunabilir bir karakter verir: `'a'`, `'5'`, `' '`.

keyCode

Hangi fiziksel tuş? Basılan tuşun sayısal kodu. Ok tuşları, SHIFT, CTRL gibi karakteri olmayan tuşlar bununla tanınır.

Harf için `key === "w"`; ok tuşu için `keyCode === LEFT_ARROW`. p5.js ok tuşlarının sayılarını ezberlemeyin diye sabitleri hazır verir.

SORU

Sol okun `keyCode` 'u 37. ASCII tablosunda 37 ise `%` işareti.

Sol oka basınca ekrana `%` mi yazılıyor? Bu bir çelişki mi?



Aynı numara, farklı binalar

Tuş	keyCode	p5.js sabiti	ASCII'de?
Sol ok	37	LEFT_ARROW	Yok
Yukarı ok	38	UP_ARROW	Yok
Sağ ok	39	RIGHT_ARROW	Yok
Aşağı ok	40	DOWN_ARROW	Yok

Ok tuşları birer **karakter** değil, özel tuştur; ASCII'de karşılıkları yoktur. 37, tarayıcıların ayrı bir tuş-numara listesinden gelir.

Benzetme

İki farklı apartmanda “daire 37” olabilir. Aynı numara, farklı binalar. Sol ok $\neq$ %.

Ezberlemeyin

Kodda her zaman `LEFT_ARROW` yazın.
`console.log(keyCode)` yalnızca merak ve hata ayıklama içindir.

LEFT ile LEFT_ARROW aynı şey değil

`keyIsDown(LEFT)` **çalışmaz**

`LEFT`, `RIGHT`, `UP`, `DOWN` **hizalama** sabitleridir: `textAlign(LEFT)`. Değerleri metindir (`'left'`). `keyIsDown` ise sayı bekler.

Harfler için de sayı gerekir: A 65, D 68, W 87, S 83.

Tuş	Tipik kullanım
Oklar	<code>keyIsDown(LEFT_ARROW)</code> ya da <code>keyCode === LEFT_ARROW</code>
Harf, sürekli	<code>keyIsDown(65)</code>
Harf, olayda	<code>key === "w" key === "W"</code>

Olayda ok ve harf: konsola bakın

sketch.js

Kopyala Düzenle Büyüt

```
1 function setup() {  
2   createCanvas(200, 100);  
3 }  
4  
5 function keyPressed() {  
6   if (keyCode === UP_ARROW) {  
7     console.log("Yukari ok");  
8   }  
9   if (key === "w" || key === "W") {  
10    console.log("W tusu");  
11  }  
12 }
```

Çıktı

Çalıştır Büyüt

Gri tuvale tıklayın, sonra ↑ ve W'ye basın. Mesajlar tuvalde değil, altındaki konsolda görünür.

WASD ve okları birleştirmek

sketch.js

Kopyala

Büyüt

```
1  if (keyIsDown(LEFT_ARROW) || keyIsDown(65)) {  
2    kareX = kareX - HIZ * dt;  
3  }  
4  if (keyIsDown(RIGHT_ARROW) || keyIsDown(68)) {  
5    kareX = kareX + HIZ * dt;  
6  }
```

`||` ile “sol ok **ya da** A basılıysa” deriz. İki alışkanlık, tek hareket.

Görünmeyen karakterler: `\n` ve `\t`

Yazımı	ASCII	Adı	Ne yapar?
<code>\n</code>	10	LF (line feed)	Yeni satıra geçer
<code>\t</code>	9	TAB (sekme)	Yatayda birkaç boşluk kadar ilerler
boşluk	32	Space	Görünür boşluk

Kaçış dizisi (escape sequence)

Ters eğik çizgi `\` "sonraki karakter özel anlam taşıyor" demektir. `\n` iki karakterle yazılır ama bellekte tek bir satır sonu karakteridir.

0–31 arasındaki denetim kodları ekrana harf gibi basılmaz; programda böyle yazılır.

Çok satırlı HUD yazısı

Yazdırılamayan karakterler — text() çıktısı

\n satır sonu · \t sekme (tab) — ASCII denetim kodları (görünmez)

```
// sketch
text("Skor: 3\nCan: 5", 24, 40);
text("A\tB\tC", 24, 100);

// \n = yeni satır (ASCII 10)
// \t = sekme / tab (ASCII 9)
```

Kaynak kod (string içinde)

Tuval (text çıktısı)

Skor: 3
Can: 5

A B C

Ekranda görünen

Kaçış dizileri (escape)

String içinde yazılan \n ve \t, ekranda «görünmez» karakter üretir: satır kırar veya yatay boşluk bırakır.

p5.js text() bunları işler. Sütun hizası kritikse \t yerine ayrı text() çağrıları veya sabit x konumları daha güvenilirdir.

ASCII: LF (\n) = 10 · TAB (\t) = 9 · boşluk = 32

Solda kaynak kod, sağda tuvaldeki sonuç. Skor ve can gibi alt alta yazılar için \n pratiktir.

İki küçük uyarı

Sütunlar önemliyse

`\t` 'nin genişliği yazı tipine göre değişir. Tuvalde düzgün sütun için ayrı `text(..., x, y)` çağrıları daha öngörülebilirdir. Konsolda `console.log("skor\t" + skor)` iş görür.

Girinti ile karıştırmayın

Kodda satırı içeri almak için bastığınız boşluk ve Tab da aynı karakterlerdir (32 ve 9). Ama onlar editördeki düzen içindir; metnin içindeki `\n` ve `\t` ise ekrana yazılan içeriktir.

Unicode: ASCII'nin ötesi

- ASCII yalnızca Latin alfabesinin dar bir dilimini kapsar.
- **Unicode**, dünyadaki bütün yazı sistemlerini (ğ, ş, emoji...) tek bir numara uzayında toplar.
- İlk 128 kodu ASCII ile aynıdır: 'A' ikisinde de 65.

Türkçe karakterli değişken adı

`let skorŞuAn = 0;` çoğu tarayıcıda çalışır. Yine de adlarda ASCII harfleri tercih ediyoruz (`skorSuAn`): karakter bozulması riski azalır, İngilizce kaynaklarla eşleşmek kolaylaşır, klavye düzeni farkı sorun çıkarmaz.

Oyuncuya gösterilen yazılarda

Türkçe karakter doğaldır: `text("Kazandınız!")` . Kısıt ekran yazılarında değil, adlardadır.

Klavye ile kontrol edilen kare (1/2)

sketch.js · satır 13-34

Kopyala

Büyüt

```
13 function draw() {  
14   background(28, 32, 48);  
15   let dt = deltaTime / 1000;  
16  
17   if (keyIsDown(LEFT_ARROW) || keyIsDown(65)) {  
18     oyuncuX = oyuncuX - HIZ * dt;  
19   }  
20   if (keyIsDown(RIGHT_ARROW) || keyIsDown(68)) {  
21     oyuncuX = oyuncuX + HIZ * dt;  
22   }  
23   if (keyIsDown(UP_ARROW) || keyIsDown(87)) {  
24     oyuncuY = oyuncuY - HIZ * dt;  
25   }  
26   if (keyIsDown(DOWN_ARROW) || keyIsDown(83)) {  
27     oyuncuY = oyuncuY + HIZ * dt;  
28   }  
29  
30   oyuncuX = constrain(oyuncuX, 0, width - BOYUT);  
31   oyuncuY = constrain(oyuncuY, 0, height - BOYUT);  
32  
33   fill(90, 180, 255);  
34   rect(oyuncuX, oyuncuY, BOYUT, BOYUT);
```

Çıktı

Çalıştır

Büyüt

Ok / WASD: hareket | R: merkeze sıfırla
Fare: 0, 0



Klavye ile kontrol edilen kare (2/2)

sketch.js · satır 36-49

Kopyala

Büyüt

```
36 fill(220);
37 textSize(14);
38 textAlign(LEFT, TOP);
39 text("Ok / WASD: hareket | R: merkeze sıfırla", 12, 12);
40 text("Fare: " + floor(mouseX) + ", " + floor(mouseY), 12, 32);
41 }
42
43 function keyPressed() {
44   if (key === "r" || key === "R") {
45     oyuncuX = width / 2 - BOYUT / 2;
46     oyuncuY = height / 2 - BOYUT / 2;
47   }
48 }
```

Çıktı

Çalıştır

Büyüt

Ok / WASD: hareket | R: merkeze sıfırla
Fare: 0, 0

Hareket `draw` içinde ve sürekli; R ile sıfırlama `keyPressed` içinde ve tek seferlik. İki katman aynı sketch'te.

Bu örnekteki yardımcılar

`constrain(deger, min, max)`

Değeri alt ve üst sınır arasında tutar. Kare tuvalin dışına taşmaz.

`color(r, g, b)`

Bir renk değeri üretir; değişkende saklanır. `red()`, `green()`, `blue()` bileşenlerini okur.

`floor(n)`

Aşağı yuvarlar. Burada fare koordinatını tam sayı göstermek için.

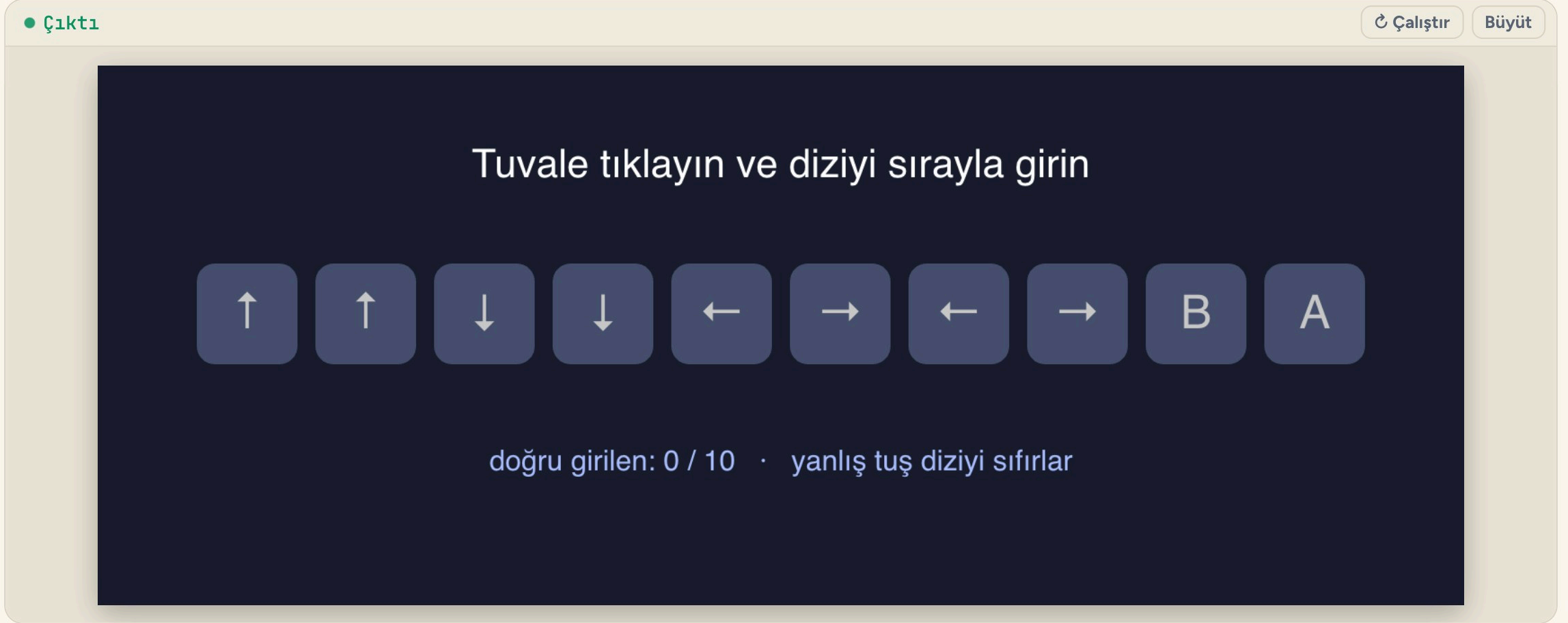
`constrain`, konumu sınırdaki kilitleyen bir engeldir. Duvara çarpınca yön değiştiren **sekme** tepkisini bir sonraki bölümde açık `if` ile yazacağız.

Bir şey çalışmıyorsa

Sorun	Olası neden	Denenecek
Tuşlar hiç çalışmıyor	Tuval odakta değil	Önizlemeye tıkla
Hareket çok yavaş ya da hızlı	<code>HIZ</code> ya da <code>deltaTime</code>	<code>HIZ * dt</code> formülünü kontrol et
R tuşu yanıt vermiyor	<code>keyPressed</code> tanımsız	Fonksiyon adının yazımına bak
Kare ekrandan çıkıyor	<code>constrain</code> eksik	Sınır satırını ekle
Çapraz hareket fazla hızlı	x ve y birlikte değişiyor	Basit oyunda sorun değil; istenirse normalize et
Bir tuş bazen çalışıyor	Klavye düzeni, büyük/küçük harf	<code>keyCode</code> ya da iki harfi birden kontrol et

İlk soru hep aynı: bu davranış **sürekli** mi olmalı, **tek seferlik** mi?

Tarihin en ünlü tuş dizisi: Konami Kodu



1986'da Gradius'un NES sürümünde ortaya çıktı; Contra'da 30 can verdiği için efsaneleşti. Kodu bir dizide, `keyPressed` içinde tek tek kontrol ediliyor.

Yapay zekâyla: kontrol şeması tasarlayın

 25-30 dk

1. Basit bir oyun fikri seçin: uzay gemisi, top, kare.
2. Hangi tuşlar sürekli (`keyIsDown`), hangileri tek seferlik (`keyPressed`) olacak? Tabloya yazın.
3. Tablonuzu yapay zekâya verin: "Her tuş için `keyIsDown` mu `keyPressed` mi kullanmalıyım? Kısa gerekçe ver."
4. Öneriyi az önceki kare örneğiyle karşılaştırın.

İki satırlık karar

 20-25 dk

1. Sürekli basılı tutulan hareket (WASD) ile tek seferlik eylem (SPACE ile ateş ya da zıplama) için hangi aracı seçerdiniz? Tablo gibi iki satırda yazın.
2. Ok tuşları için `LEFT_ARROW` kullanımını not edin.
3. Yapay zekâya sorun: “p5.js’te `keyIsDown`, `keyPressed` ve `keyCode` ne zaman kullanılır?” Yanıtı bugünkü kontrol tablosuyla karşılaştırın.

Kod zamanı

 35 dk

1. Kare örneğinde `Shift` basılıyken hız iki katına çıksın. İpucu: `keyIsDown(SHIFT)`.
2. Sayılara basınca karenin rengi değişsin: 1 kırmızı, 2 yeşil, 3 mavi.
3. **Ateş:** boşluğa her basışta karenin üstünden bir mermi çıksın ve yukarı gitsin. Mermileri bir dizide tutun; ekrandan çıkanı silin.
4. **Zorlu:** çapraz giderken hız artmasın. İki yön birden basılıysa hızı 0.707 ile çarpın.

Oyuncu boşluğa 2 saniye basılı tuttu. `zipla()` kaç kez çağrılır?

sketch.js

Kopyala

Büyüt

```
1 function draw() {  
2   if (keyIsDown(32)) {  
3     zipla();  
4   }  
5 }
```

A 1

B 2

☒ Yaklaşık 120

D 0

60 FPS × 2 saniye. Zıplama tek seferlik bir eylemdir; yeri `keyPressed()`.

Hangisi sol ok tuşunu doğru yakalar?

A `keyIsDown(LEFT)`

B `key === "left"`

☒ `keyIsDown(LEFT_ARROW)`

D `keyIsDown("%")`

`LEFT` hizalama sabitidir. Ok tuşlarının karakteri yoktur; `keyCode` ve `LEFT_ARROW` ile tanınırlar.

`text("A\nB\tC", 20, 40)` tuvalde nasıl görünür?

- ☐ A Tek satırda: A\nB\tC
- ☒ İki satır: üstte A; altta B, biraz boşluk, C
- ☐ C Üç satır: A, B, C

`\n` satır kırar, `\t` yatayda sekme boşluğu bırakır.

Merak edenler için kaynaklar

P5.JS REFERANSI

keyIsDown()

Resmî tanım ve tuş kodlarıyla örnekler.

p5js.org/reference/p5/keyIsDown

WIKİPEDIA

ASCII

Tablonun tamamı ve tarihçesi.

en.wikipedia.org/wiki/ASCII

WIKİPEDIA

Konami Code

Kodun hikâyesi ve geçtiği oyunların listesi.

en.wikipedia.org/wiki/Konami_Code

UNICODE CONSORTIUM

The Unicode Standard

Bütün yazı sistemlerini kapsayan standardın resmî sitesi.

home.unicode.org

Bölüm 3'ü bitirdik

- ✓ `keyIsDown()` basılı olduğu her frame doğrudur: sürekli hareket, `draw` içinde.
- ✓ `keyPressed()` basma anında, `keyReleased()` bırakma anında bir kez çalışır.
- ✓ `key` karakteri, `keyCode` tuşun sayısal kodunu verir.
- ✓ Oklar için `LEFT_ARROW` gibi sabitler; `LEFT` hizalama içindir.
- ✓ Karakterler sayılarla saklanır: ASCII ve onu kapsayan Unicode. `\n` satır kırar, `\t` sekme koyar.
- ✓ `constrain` değeri iki sınır arasında tutar.

SIRADAKİ BÖLÜM

Bölüm 4: Çarpışmalar ve oyun durumları

Top duvardan sekecek, oyun menüden başlayacak.

Bölüm 4'e geç →

← Önceki