

deltaTime: her bilgisayarda aynı hız

Hareketi frame sayısına değil, geçen gerçek süreye bağlayacağız.

Sorun: $x += 3$

deltaTime nedir?

Piksel / saniye

Hareketli top

SORU

Aynı oyun, iki bilgisayar. Birinde karakter iki kat hızlı koşuyor.

Kod aynı, oyun aynı. Fark nereden geliyor olabilir?

İpucu: geçen dersin son konusu.



x += 3 hızı frame sayısına bağlar

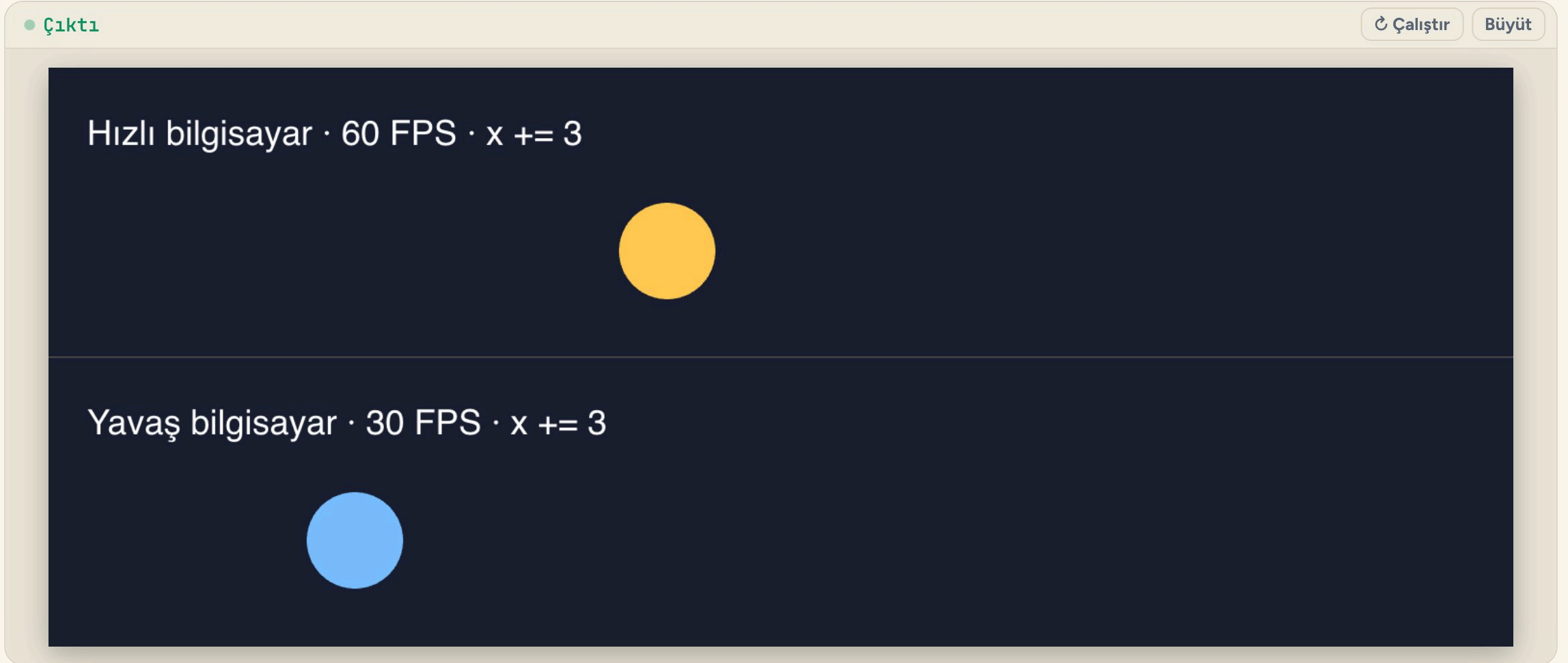
Bilgisayar	FPS	Saniyede yol
Hızlı	~60	$60 \times 3 = 180$ piksel
Yavaş	~30	$30 \times 3 = 90$ piksel

Her frame 3 piksel ekliyoruz. Saniyede kaç frame üretilirse o kadar yol alınır.

Sonuç

Aynı kod, farklı frame hızında farklı oyun hızı üretir. Oyuncunun deneyimi bilgisayardan bilgisayara değişir.

İki bilgisayarı yan yana koysak



İki topun kodu da aynı: `x += 3`. Alttaki bilgisayar yalnızca yarısı kadar frame üretebiliyor.

Bu hata gerçek oyunlarda da yaşandı

Dark Souls II PC

60 FPS'te oynayanların silahları, 30 FPS'te oynayanlara göre iki kat hızlı yıpranıyordu. Yıpranma hesabı frame sayısına bağlıydı.

Skryim, Fallout 4

Fizik hesabı frame hızına bağlı. Çok yüksek FPS'te eşyalar masalardan fırlar, karakterler tuhaf davranır.

Space Invaders 1978

Uzaylılar azaldıkça işlemcinin yükü hafifliyor, oyun hızlanıyordu. Bu "hata" oyunun en sevilen özelliği oldu.

Bugünkü konu, profesyonel stüdyoların da zaman zaman takıldığı bir konu.

Hızlı nesnelerde sorun büyür: mermi

Frame başına küçük adım: çarpar



Frame başına büyük adım: üstünden atlar



Mermi frame başına 15 piksel gitsin:

- 30 FPS'te saniyede **450** piksel.
- 60 FPS'te saniyede **900** piksel.

Çarpışma bile değişir

Adım büyüdükçe mermi ince bir engelin içinden "atlayarak" geçebilir. Hata bazı bilgisayarlarda görünür, bazılarında hiç çıkmaz.

Programcıların en meşhur cümlesi

Oyun çok hızlı, oynanmıyor!



oyuncu • 144 Hz monitör

Tuhaf... Bende gayet normal.



geliştirici • 60 Hz dizüstü

"Benim bilgisayarımda
çalışıyordu."



çözüm değil

Fizik tabanlı ya da ağ üzerinden eşlenen oyunlarda bu fark kabul edilemez.

SORU

Hareketi frame sayısına bağlamayalım. Peki neye bağlayalım?

Bir araba “saatte 60 km” gider; “motor devri başına 2 metre” demeyiz. Hızın doğal birimi ne?



Geçen süreye: `deltaTime`

`deltaTime`

Bir önceki `draw()` çağrısından bu yana geçen süre, **milisaniye** cinsinden. p5.js otomatik günceller; bizim atamamız gerekmez.

FPS	<code>deltaTime</code>
60	≈ 16.7 ms
30	≈ 33.3 ms
10	≈ 100 ms

Frame'ler seyrekleşince her frame'in "süresi" uzar. Bu süreyi biliyorsak, o frame'de ne kadar yol alınacağını hesaplayabiliriz.

Yol = hız × süre

HIZ

Saniyede kaç piksel?

120 piksel / saniye

× süre

Bu frame kaç saniye sürdü?

deltaTime / 1000

= yol

Bu frame'de kaç piksel?

≈ 2 piksel

sketch.js

Kopyala

Büyüt

```
dtSaniye = deltaTime / 1000           // ör. 16.7 / 1000 ≈ 0.0167 s
konumDegisimi = HIZ * dtSaniye        // ör. 120 * 0.0167 ≈ 2 px bu frame'de
```

Kritik birim

deltaTime milisaniyedir, saniye değil. 1 saniye = 1000 ms olduğu için önce 1000'e bölünür.

Sayılarla izleyelim

İfade	Örnek değer	Anlamı
HIZ	120	Saniyede 120 piksel
deltaTime	≈ 16.7 (ms)	Son frame'in süresi; henüz saniye değil
deltaTime / 1000	≈ 0.0167 (s)	Aynı süre, saniye cinsinden
HIZ * deltaTime / 1000	≈ 2 piksel	Bu frame'deki kayma

Okunabilir alışkanlık

Kopyala

Büyüt

```
let dt = deltaTime / 1000; // bir kez saniyeye çevir
topX = topX + HIZ * dt;
```

Aynı top, bu kez `deltaTime` ile

sketch.js · satır 12-29

Kopyala

Büyüt

```
12 function draw() {  
13   background(30, 35, 50);  
14  
15   let dt = deltaTime / 1000; // ms → saniye  
16   topX = topX + HIZ * dt; // HIZ * deltaTime / 1000 ile aynı  
17  
18   if (topX > width + TOP_YARICAP) {  
19     topX = -TOP_YARICAP;  
20   }  
21  
22   fill(255, 200, 80);  
23   ellipse(topX, topY, TOP_YARICAP * 2, TOP_YARICAP * 2);  
24  
25   fill(255);  
26   textSize(14);  
27   textAlign(LEFT, TOP);  
28   text("FPS: " + nf(frameRate(), 0, 1), 10, 10);  
29 }
```

● Çıktı

Çalıştır

Büyüt

FPS: 59.9



“Düzenle” ile `frameRate(60)` değerini 30, sonra 10 yapın. Top takılarak ilerler ama aynı sürede aynı yolu alır.

iki yöntem yan yana

Yöntem	60 FPS (dt $\approx$ 16.7 ms)	30 FPS (dt $\approx$ 33.3 ms)	Saniyede toplam
<code>topX + 3</code>	+3 px / frame	+3 px / frame	180 px ve 90 px
<code>topX + 120 * dt</code>	+2 px / frame	+4 px / frame	$\approx$ 120 px, ikisinde de

İşin sırrı

Frame süresi kısalınca frame başına eklenen piksel azalır ama frame sayısı artar. Süre uzayınca tersi olur. Saniyedeki toplam aynı kalır.

SORU

/ 1000 yazmayı unutursak ne olur?

sketch.js

Kopyala

Büyüt

```
x = x + 100 * deltaTime;
```

deltaTime yaklaşık 16.7 iken bir frame'de kaç piksel eklenir?

Top ekrandan fırlar



SANİYEDE 100 PİKSEL İSTEMİŞTİM

$100 \times 16.7 \approx 1670$ piksel. Her frame'de.

En sık tuzak

Yanlış birimle çarpmak. Top saniyede 100 değil, on binlerce piksel yol alır.

Doğrusu

Kopyala

Büyüt

```
let dt = deltaTime / 1000;  
x = x + 100 * dt;
```

Birim özeti: HIZ piksel/saniye $\cdot$ $\text{deltaTime} / 1000$ saniye $\cdot$ çarpım piksel.

SORU

Oyun açıkken başka bir sekmeye geçtiniz. Beş saniye sonra döndünüz.

Sekme arka plandayken `draw` duraklar. Döndüğünüz ilk frame'de `deltaTime` kaç olur? Top ne yapar?



deltaTime çok büyüyebilir

İki frame arasındaki süre birkaç **saniyeye** çıkar. Top tek frame'de yüzlerce piksel atlar; bir duvarın içinden bile geçebilir.

Üst sınır koymak (clamp)

Kopyala

Büyüt

```
let dt = min(deltaTime, 50) / 1000;  
topX = topX + HIZ * dt;
```

Tek frame'de en fazla 50 ms sayılır.

Oyunlarda

Çoğu oyun bu sınırı koyar. Yoksa oyunu duraklatıp dönen oyuncu, karakterini haritanın dışında bulabilir.

Bu dersteki örneklerde

Doğrudan `deltaTime` yeterli. Sekmeden dönünce ani bir sıçrama görürseniz bu beklenen davranıştır.

Hareketli top: yatay kayma + dikey salınım

sketch.js · satır 1-7

Kopyala

Büyüt

```
1 let topX = 0;
2 const TOP_YARICAP = 22;
3 const HIZ_X = 150;           // yatay: piksel / saniye
4 const SALINIM_GENISLIGI = 40;
5 const SALINIM_HIZI = 2;      // salınım faz hızı (radyan/saniye yaklaşık kullanım)
6
7 let zaman = 0; // birikimli saniye (salınım fazı için)
```

1. Top sürekli sağa gider; tuvalden çıkınca soldan yeniden girer.
2. Dikey konum sinüs eğrisiyle salınır.
3. Ekranda anlık FPS ve `deltaTime` gösterilir.

zaman değişkeni

Her frame `dtSaniye` kadar artan, birikimli bir saat. Salınımın fazı bu saate bağlanacak.

Hareketli top: draw

sketch.js · satır 15-36

Kopyala

Büyüt

```
15 function draw() {  
16   background(25, 32, 48);  
17  
18   let dtSaniye = deltaTime / 1000;  
19   topX = topX + HIZ_X * dtSaniye;  
20   zaman = zaman + dtSaniye;  
21  
22   if (topX > width + TOP_YARICAP) {  
23     topX = -TOP_YARICAP;  
24   }  
25  
26   let topY = height / 2 + sin(zaman * SALINIM_HIZI) * SALINIM_GENISLIGI;  
27  
28   fill(255, 190, 70);  
29   ellipse(topX, topY, TOP_YARICAP * 2, TOP_YARICAP * 2);  
30  
31   fill(220);  
32   textSize(14);  
33   textAlign(LEFT, TOP);  
34   text("FPS: " + nf(frameRate(), 0, 1), 12, 12);  
35   text("deltaTime: " + nf(deltaTime, 0, 1) + " ms", 12, 32);  
36 }
```

Çıktı

Çalıştır

Büyüt

FPS: 59.9
deltaTime: 16.7 ms



Neden `frameCount` değil de `zaman` ?

`sin(frameCount * 0.05)`

Faz frame sayısına bağlı. 30 FPS'te salınım yarı hızda olur; az önceki tutarsızlık dikey ekseninde geri gelir.

`sin(zaman * SALINIM_HIZI)`

`zaman` geçen gerçek süredir. Salınımın hızı da FPS'ten bağımsız olur; tıpkı yatay hareket gibi.

Deney

`setup` içinde `frameRate(30)` deneyin. Topun yatay hızı gözle aynı kalmalı; yalnızca akıcılık değişir. Sonra `topX = topX + 5` satırını geçici olarak geri getirip farkı gözleyin.

İki toplu karşılaştırma

 20 dk

1. **topA** frame başına sabit gitsin: `topAX = topAX + 4`.
2. **topB** zamana bağlı gitsin: `topBX = topBX + HIZ * deltaTime / 1000` (örneğin `HIZ = 240`).
3. İkisi de tuvalden çıkınca soldan yeniden girsin; farklı renklerde olsunlar.
4. Ekranda `frameRate()` ve "A: frame", "B: deltaTime" etiketlerini gösterin.
5. `frameRate(60)` ve `frameRate(30)` 'u sırayla deneyin. Hangisinin temposu değişiyor? Bir iki cümleyle not edin.
6. **Bonus:** `guncelle` ve `ciz` işlerini ayrı fonksiyonlara bölün.

Yapay zekâyla: siz yazın, o ölçsün

 35-45 dk

1. Önce iki toplu sketch'i **kendiniz** yazıp çalıştırın. Yapay zekâya oyunun tamamını yazdırmayın.
2. Sketch'inizi yapıştırıp ölçüm isteyin: "Mekaniği bozma. Her iki top için son 1 saniyede kat edilen yatay yolu ölçüp ekranda göster. 30 ve 60 FPS için beklenen farkı kısa bir tabloyla yaz. Class kullanma."
3. Kontrol edin: A'nın saniyedeki yolu FPS ile değişiyor mu, B'ninki sabit mi? Yapay zekâ `deltaTime` 'ı iki kez mi böldü, 1000'i mi unuttu?
4. İsteğe bağlı cilâ: tuşla 30 / 60 geçişi, `min(deltaTime, 50)` tavanı, hafif iz. Sonra "neleri değiştirdin, madde madde açıkla" deyin.

Yapay zekâyla: sınırlı deltaTime

 15-20 dk

1. Sekmeden dönünce `deltaTime` 'ın çok büyüdüğü senaryoyu kendi cümlelerinizle yazın.
2. Yapay zekâdan `min(deltaTime, 50)` benzeri bir sınırlama (clamp) iskeleti isteyin.
3. Kendi top örneğinizde sınırlı ve sınırsız sürümü deneyin; başka sekmeye geçip dönerek farkı gözleyin.

Kod zamanı

 25 dk

1. Topun hızını saniyede 300 piksel yapın. Tuvali kaç saniyede geçmeli? Ölçün.
2. **Geri sayım:** 10 saniyeden geriye sayan bir sayaç yazın. `frameCount` kullanmayın; `deltaTime` ile biriktirin.
3. **Hızlanan top:** hız her saniye 50 piksel/saniye artsın.
4. **Zorlu:** top iki kenar arasında sekerek gidip gelsin; hareket `deltaTime` ile olsun, sınır `min(deltaTime, 50)` ile korunsun.

Dört soru ve bir köprü

1

VeriDurumu hangi
değişkenler tutuyor?

2

GüncellemeHer
frame'de ne
değişiyor?

3

ÇizimGörünenler
nereden geliyor?

4

KuralHangi koşul neyi
tetikliyor?

Köprü cümlesi: “Şimdi **X** biliyorum; bundan sonra **Y** için buna ihtiyacım var.” Bugünkü konu için doldurun.

HIZ = 200 piksel/saniye, bu frame 20 ms sürdü. Top kaç piksel ilerler?

A 200

B 20

☒ **C** 4

D 4000

20 ms = 0.02 saniye. $200 \times 0.02 = 4$ piksel.

Kod $x += 3$. Oyuncunun monitörü 144 Hz ve oyun 144 FPS'te çalışıyor. Saniyede kaç piksel?

A 180

B 144

☒ **C** 432

D 3

$144 \times 3 = 432$. Siz 60 FPS'te 180 piksel/saniye için tasarlamıştınız; oyun bu oyuncuda 2,4 kat hızlı.

Hangisi her bilgisayarda saniyede 150 piksel ilerletir?

A`x = x + 150;`**B**`x = x + 150 * deltaTime;`☒`x = x + 150 * deltaTime / 1000;`**D**`x = x + 150 / deltaTime;`

Hız (piksel/saniye) × süre (saniye) = yol (piksel). `deltaTime` milisaniye olduğu için 1000'e bölünür.

Merak edenler için kaynaklar

P5.JS REFERANSI

deltaTime

Resmî tanım ve kısa bir örnek.

p5js.org/reference/p5/deltaTime

WIKİPEDIA

Delta timing

Kavramın oyun programlamadaki yeri ve tarihçesi (İngilizce).

en.wikipedia.org/wiki/Delta_timing

CONSTRUCT

Delta-time and framerate independence

Aynı fikir başka bir oyun motorunda: konu p5.js'e özgü değil.

construct.net/en/tutorials/delta-time-framerate-2

THE CODING TRAIN

The Bouncing Ball

Zorlu görevdeki seken top için iyi bir başlangıç videosu.

[thecodingtrain.com > 3-conditionals > 2-bouncing](https://thecodingtrain.com/3-conditionals/2-bouncing)

Bugün ne öğrendik?

- ✓ `x += n` FPS'e bağımlıdır: frame sayısı değişince hız değişir.
- ✓ `deltaTime` : son frame'den bu yana geçen süre, milisaniye.
- ✓ Hız piksel/saniye olarak yazılır: `x += HIZ * deltaTime / 1000`.
- ✓ `/ 1000` unutulursa top ekrandan fırlar.
- ✓ Sekmeden dönünce `deltaTime` çok büyür; `min(deltaTime, 50)` ile sınırlandırılabilir.
- ✓ Salınım gibi zamana bağlı her şey için `frameCount` yerine biriken `zaman`.

SIRADAKİ DERS

Tek top yetmez: yüz top lazım

Diziler: birçok değeri tek bir adla tutmak.

Sonraki sunum →

Tüm sunumlar