

Beş yuvalı envanter, bir avuç balon

Boyutu baştan belli diziler ve birçok nesneyi birlikte yöneten paralel diziler.

Sabit kapasite

tepe ve BOS

Mini envanter

Balon sahnesi

splice

SORU

Oyunlarda envanter neden hep sınırlı?

Çantada 5 yuva var, altıncı eşyayı alamıyorsunuz. Geçen ders gördüğümüz `push` ile sınırsız ekleyebiliyorduk. Sınırı kim, nasıl koyuyor?



İki yol var: uzayan liste, sabit yuvalar

Dinamik

`push` / `pop` Eleman gelince dizi uzar, gidince kısalır. Eleman yoksa yuva da yoktur. Uzunluk baştan bilinmiyorsa: tahmin geçmişi, kule listesi.

Sabit

Sabit kapasite Yuva sayısı baştan bellidir ve değişmez. Boş yuvada özel bir “boş” işareti durur. Kaç yuvanın dolu olduğu ayrı bir sayaçta tutulur.

Sabit model, belleği ve döngü sınırını öngörülebilir kılar. Tasarım açısından da bir kuraldır: “en fazla 5 eşya”.

Can göstergesi: beş yuva, kaç dolu?

● Çıktı

Çalıştır

Büyüt



Sol yarıya tıkla: can kaybet · Sağ yarıya tıkla: can kazan

MAKS = 5 · dolu = 3 · yuva sayısı hiç değişmiyor

Zelda'daki kalpler, Minecraft'ın 9 yuvalı çubuğu, bir silahın şarjörü: hepsinde yuva sayısı sabit, değişen yalnızca doluluk.

Fikir dört adımda

- 1 Üst sınırı sabitle: `MAKS`.
- 2 Diziyi `MAKS` uzunluğunda oluştur ve `B0S` ile doldur.
- 3 **Ekle:** `yigin[tepe] = deger; tepe++`. `push()` yok.
- 4 **Çıkar:** `tepe--`; değeri oku; yuvaya yeniden `B0S` yaz. `pop()` yok.

Yığın mantığı metotlara bağlı değil: “sona ekle, sondan çıkar” davranışı indeks ve atamayla da kurulur. Birçok dilde hazır `push` / `pop` zaten yoktur.

tepe : sıradaki boş yuvayı gösteren parmak

kilic	iksir	anahtar	BOS	BOS
0	1	2	3 ▲ tepe	4

`MAKS = 5 · tepe = 3 · envanter.length` her zaman 5

tepe iki şeyi birden söyler

Kaç yuva dolu (3) ve yeni eşyanın nereye yazılacağı (indeks 3).

BOS

"Bu yuvada bir şey yok" demek için seçtiğimiz özel değer: `-1`, `""` gibi.

Ekleme: `yiginEkle`

sketch.js · satır 1-14

Kopyala

Büyüt

```
1 const MAKS = 5;
2 const BOS = -1; // geçerli tahmin 1-10 olduğu için -1 «boş» demektir
3
4 let yigin = [BOS, BOS, BOS, BOS, BOS]; // length her zaman 5
5 let tepe = 0;
6
7 function yiginEkle(deger) {
8   if (tepe >= MAKS) {
9     return false; // yer yok
10  }
11  yigin[tepe] = deger;
12  tepe = tepe + 1;
13  return true;
14 }
```

Önce sınıra bakıyor: yer yoksa `false` döner ve hiçbir şey eklemez. Yer varsa `tepe` 'nin gösterdiği yuvaya yazar, sonra `tepe` 'yi bir ilerletir.

Çıkarma: yiginCikar

sketch.js · satır 16-24

Kopyala

Büyüt

```
16 function yiginCikar() {  
17   if (tepe === 0) {  
18     return BOS; // boş  
19   }  
20   tepe = tepe - 1;  
21   let eski = yigin[tepe];  
22   yigin[tepe] = BOS;  
23   return eski;  
24 }
```

1. Yığın boşsa BOS döner.
2. tepe 'yi bir geri al: artık son dolu yuvayı gösteriyor.
3. Oradaki değeri bir kenara al.
4. Yuvaya yeniden BOS yaz.
5. Aldığın değeri döndür.

Adım adım izleyelim

Adım	İşlem	yigin	tepe
Başlangıç	—	[-1, -1, -1, -1, -1]	0
1	yiginEkle(7)	[7, -1, -1, -1, -1]	1
2	yiginEkle(3)	[7, 3, -1, -1, -1]	2
3	yiginCikar() → 3	[7, -1, -1, -1, -1]	1

Yalnızca dolu yuvaları yaz

Kopyala

Büyüt

```
for (let i = 0; i < tepe; i = i + 1) {  
  text(yigin[i], 20, 40 + i * 20);  
}
```

yigin.length hep MAKS . Dolu eleman sayısı tepe . Döngü bu yüzden $i < \text{tepe}$ ile sınırlanır.

SORU

Skor listesinde boş yuvayı 0 ile işaretleseک olur mu?

Oyuncu gerçekten 0 puan aldıysa ne olacak?



Boş işareti, geçerli bir değerle çakışmamalı

Tahmin 1–10 arası

BOS = -1 uygun. -1 hiçbir zaman geçerli bir tahmin olamaz.

Skor 0 olabilir

O zaman 0 “boş” anlamına gelemmez; yine -1 gibi imkânsız bir değer seçilir.

Metin listeleri

Çoğunlukla "" ya da "BOS" kullanılır.

tepe tutmak istemezseniz “ilk BOS yuvaya yaz” diye tarayabilirsiniz. Fikir aynıdır; doluluk hesabı döngüyle yapılır.

Hangisi ne zaman?

Yaklaşım	Boyut	Boş yer	Ne zaman uygun?
<code>push</code> / <code>pop</code>	<code>length</code> değişir	Eleman yoksa yuva da yok	Uzunluk baştan bilinmiyor; geçmiş listesi
Sabit kapasite + <code>BOS</code> + <code>tepe</code>	<code>length == MAKS</code> , sabit	Yuva var, değeri <code>BOS</code>	Envanter, can, sabit düşman havuzu, sınırlı geri al

Kule kurma ve skor geçmişi gibi örnekleri `push` / `pop` ile yazmak kısa ve okunaklıdır. Sabit model “en fazla N yer” kuralını koda gömmek için bilinmelidir.

Başka dillerde bu bir tercih değil, zorunluluk

JavaScript'te diziler dinamik

`push` ile büyür, `pop` ile küçülür. Doğru boyutu baştan seçmek çoğu zaman bir **tasarım tercihidir**.

C, C++, Pascal ve gömülü sistemlerde

Dizinin boyutu tanımlanırken belirlenir, sonradan büyütülemez. Küçük seçilirse **taşma** (overflow) ve hatalı bellek yazımı, büyük seçilirse bellek israfı olur.

Oyun motorlarında

“En fazla 20 adım geri al”, “en fazla 3 açık menü”, “ekranda en fazla 100 mermi” gibi üst sınırlar çok yaygındır. Kapasite baştan kesilir ki oyun ortasında bellek sürprizi olmasın. Bu fikre **nesne havuzu** (object pool) denir.

Mini envanter: tıkla ekle, sol alttan çıkar

sketch.js · satır 42-50

Kopyala

Büyüt

```
42 function mousePressed() {  
43   if (mouseX < 40 && mouseY > 160) {  
44     yiginCikar();  
45     return;  
46   }  
47   // Sıradaki eşyayı ekle (döngüsel)  
48   let ad = esyalar[tepe % esyalar.length];  
49   yiginEkle(ad);  
50 }
```

Çıktı

Çalıştır

Büyüt

Sol tık: eşya ekle | Sol üst kutu: çıkar | length = 5
tepe = 0

[0] (boş)
[1] (boş)
[2] (boş)
[3] (boş)
[4] (boş)

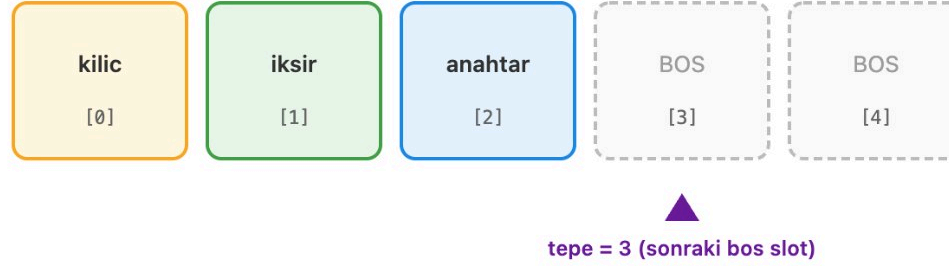
ÇIKAR

Tuvale birkaç kez tıklayın, sonra ÇIKAR kutusuna basın. `length` hep 5 kalıyor; değişen yalnızca `tepe`. Altıncı eşyayı eklemeyi deneyin.

Listeyi yuva kutularına çevirin

25 dk

Sabit envanter — MAKS = 5, tepe = 3
envanter = ["kilic", "iksir", "anahtar", BOS, BOS]



1. `draw` içindeki metin listesini kaldırın; yan yana 5 dikdörtgen yuva çizin.
2. Dolu yuva: dolu kutu ve eşyanın adı. Boş yuva: soluk çerçeve ve "BOS".
3. `tepe` indeksinin altına küçük bir üçgen koyun. `tepe == MAKS` iken "dolı" yazın.
4. `yiginEkle`, `yiginCıkar` ve `mousePressed` 'e dokunmayın; yalnızca çizimi deęiştirin.

SORU

Ekranda beş balon olacak. Her biri oval ve farklı boyda.

Bir balonu tarif etmek için kaç sayıya ihtiyacınız var? Bunları kaç dizide tutarsınız?



Dört paralel dizi, aynı indeks

balonX

120	300	480
0	1	2

balonY

120	180	100
0	1	2

balonRX · balonRY

32 · 42	50 · 32	24 · 36
0	1	2

Konum için `balonX` ve `balonY`; boyut için yatay ve dikey yarıçap: `balonRX`, `balonRY`.

sketch.js

Kopyala

Büyüt

```
ellipse(balonX[i], balonY[i],
        balonRX[i] * 2, balonRY[i] * 2);
```

`ellipse` çap ister: genişlik = $2 \times rx$, yükseklik = $2 \times ry$. Daire için rx ile ry eşittir.

Hedef: balon sahnesi

sketch.js

Kopyala

Düzenle

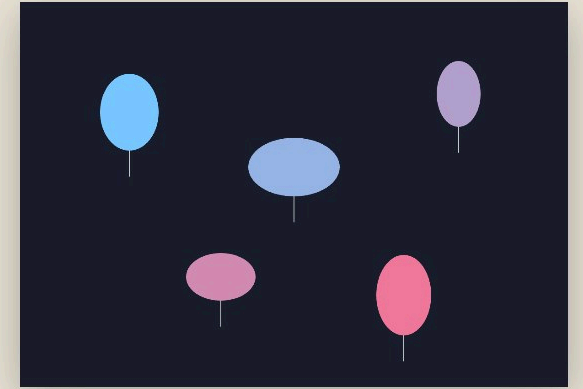
Büyüt

```
1 let balonX = [120, 300, 480, 220, 420];
2 let balonY = [120, 180, 100, 300, 320];
3 let balonRX = [32, 50, 24, 38, 30];
4 let balonRY = [42, 32, 36, 26, 44];
5
6 function setup() {
7   createCanvas(600, 420);
8 }
9
10 function draw() {
11   background(26, 30, 44);
12
13   for (let i = 0; i < balonX.length; i = i + 1) {
14     stroke(200);
15     line(balonX[i], balonY[i] + balonRY[i], balonX[i], balonY[i] + balonRY[i] + 28);
16     noStroke();
17     fill(120 + i * 30, 200 - i * 20, 255 - i * 25);
18     ellipse(balonX[i], balonY[i], balonRX[i] * 2, balonRY[i] * 2);
19   }
20 }
```

Çıktı

Çalıştır

Büyüt



Bu, birazdan sizin yazacağınız sahnenin bir örneği. Her diziye birer sayı ekleyin: altıncı balon belirir.

Paralel dizilerde ekleme ve çıkarma hep birlikte

sketch.js

Kopyala

Büyüt

```
1 balonX.push(mouseX);  
2 // balonY, balonRX, balonRY unutuldu!
```

Diziler farklı uzunlukta kalır

Döngü, yeni balon için `balonY[i]` 'yi okumaya çalışır ve `undefined` bulur.

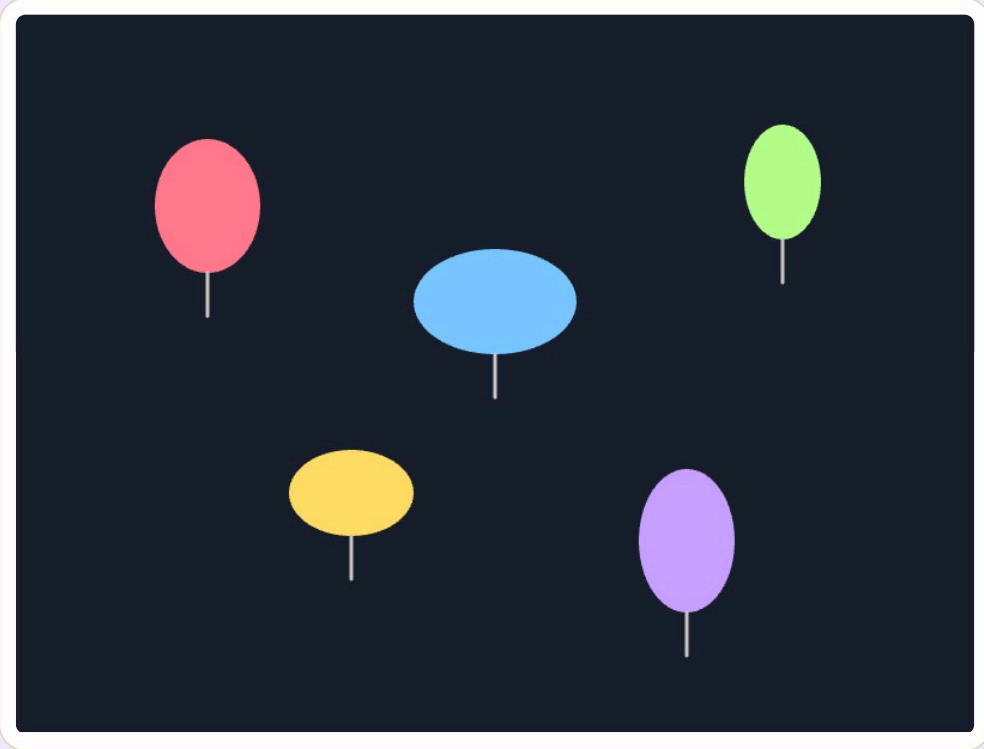
Kural

Bir diziye ne yapıyorsanız hepsine yapın: dört `push`, dört `pop`.

Sık karşılaşılan dizi hataları

Belirti	Olası neden	Çözüm
Ekranda <code>undefined</code>	Geçersiz indeks: üç elemanlı dizide <code>d[3]</code>	<code>i < dizi.length</code> koşulunu kontrol et
Son satırda fazladan boş değer	<code>i <= length</code> ile sınır aşımı	<code>i < length</code> kullan
<code>push</code> sonrası liste görünmüyor	<code>draw()</code> içinde dolaşma yok	<code>for</code> ile ekrana yazdır
Balonlar üst üste ya da garip boyda	Paralel dizide indeks kayması	X, Y, RX, RY'yi aynı <code>i</code> ile oku
Bir balon "kaybolmuş" gibi	Yalnızca bir diziye <code>push</code> / <code>pop</code>	Ekleme-çıkarma bütün dizilerde birlikte

Balon sahnenizi çizin

 20 dk

1. `balonX`, `balonY`, `balonRX`, `balonRY` dizilerini kurun. İsterseniz bir de renk dizisi.
2. `for` içinde `ellipse` ile çizin. Bonus: kısa bir `line` ile ip.
3. **Kâğıtta:** yalnızca `balonX.push(...)` yapıp diğer üç diziyi unutan arkadaşınızın ekranında ne bozulur? Bir cümleyle yazın.

Örnek olay: kule yığını

Mini kule sketch'inde üç kule kuruldu, sonra bir kez GERİ AL'a basıldı. Dizilerde ne kaldı, ekranda kaç kule görünüyor?

1. `kuleX[0]` hangi kuleyi temsil eder: ilk kurulanı mı, son kurulanı mı?
2. GERİ AL neden her iki dizide `pop` ister?
3. `length == 0` iken `pop` çağırmak neden risklidir?

SORU

Ortadaki balonu patlatmak istiyoruz. `pop()` işe yarar mı?

`pop` yalnızca sondaki elemanı çıkarıyor. Oyuncu ise istediği balona tıklıyor.



`splice(i, 1)` : aradan bir eleman çıkar

120	300	480	220
0	1	2	3

↓ `balonX.splice(1, 1)`

120	480	220
0	1	2

sonrakiler bir yuva sola kayar • `length` 3 olur

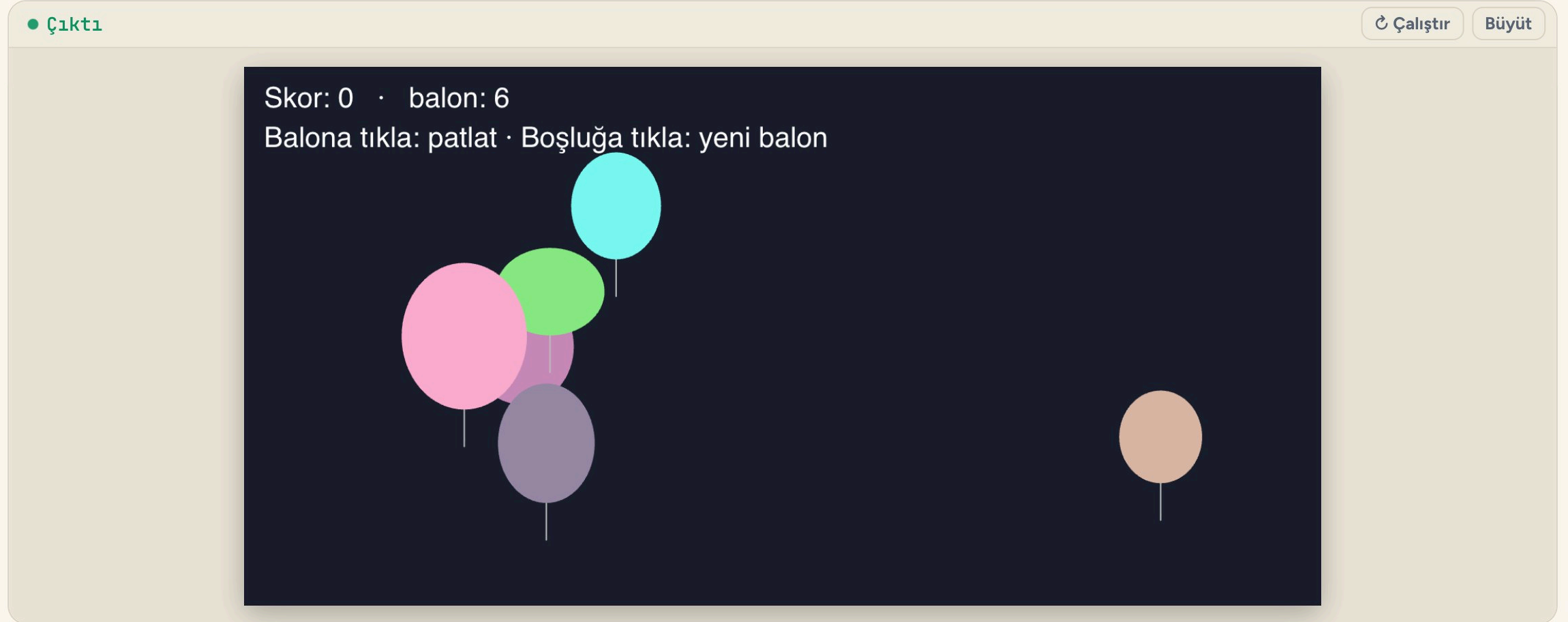
`dizi.splice(i, 1)`

İndeksi `i` olan elemanı çıkarır; `length` bir azalır. Boşluk kalmaz.

Paralel dizilerde

Aynı `i` için her dizide `splice(i, 1)` çağrılır. Biri unutulursa indeksler kayar.

Balon Patlat: hedeflediğimiz oyun



Balona tıklayınca o balon bütün dizilerden `splice` ile çıkar; boşluğa tıklayınca bütün dizilere `push` edilir. Birazdan bunu siz yazacaksınız.

Balon Patlat (1/3): mekaniği siz yazın

🕒 45-60 dk

1. Paralel dizileri ve birkaç başlangıç balonunu kurun. Class ya da nesne kullanmayın.
2. `setup` / `draw`: arka plan, skor, `for` + `ellipse`.
3. `mousePressed` akışı:
 - Tıklanan nokta hangi balonun içinde? İndeksini bulun.
 - Bulunduysa: bütün dizilerde `splice(i, 1)`; skoru artırın.
 - Bulunamadıysa: yeni bir balonu bütün dizilere `push` edin.

İskelet: vurulan balonun indeksini bulmak · satır 2-8

Kopyala

Büyüt

```
2 let vurus = -1;
3 for (let i = 0; i < balonX.length; i = i + 1) {
4   if (noktaElipsIcindeMi(mouseX, mouseY, balonX[i], balonY[i], balonRX[i], balonRY[i])) {
5     vurus = i;
6     break;
7   }
8 }
```

Balon Patlat (2/3): yardımcılarını yapay zekâdan isteyin

A • Nokta elipsin içinde mi?

`noktaElipsIcindeMi(px, py, cx, cy, rx, ry)` → boolean.
Elips denklemini kullansın.

B • Çakışmayan yeni balon

`yeniBalonOlustur(...)`:
tuvalin içinde, diğer balonlara değmeyen rastgele bir balon.
Yer bulamazsa `null`. Dönüş:
`[x, y, rx, ry]`.

C • Uyumlu renk

`uyumluBalonRengi()` → `[r, g, b]`. Pastel, birbirine yakışan renkler.

Sonra denetleyin

`noktaElipsIcindeMi` gerçekten elips mi kullanıyor, yoksa `dist` ile daire mi? `yeniBalonOlustur` kenarları ve diğer balonları hesaba katıyor mu? Siz `splice` / `push` 'u bütün dizilerde birlikte yaptınız mı?

Balon Patlat (3/3): makyaj

1. Mekanik çalışıyorsa sketch'in tamamını yapıştırıp görsel ve ses cilasını isteyin: patlayınca küçülerek kaybolma, yeni balonda büyüyerek belirme, kısa ses efektleri. "Mekaniği bozma, class kullanma."
2. "Neleri değiştirdin, bu ders düzeyinde madde madde açıkla" deyin. Anlamadığınız satır için devam sorusu sorun.
3. Önce yalnızca fikir isteyin: "Bu oyunu daha eğlenceli yapacak 6–8 makyaj fikri öner; kod yazma." Beğendiğinizi sonra uygulatın.

Hatırlatma

Animasyon için çoğu zaman ek diziler gelir: `balon01cek`, `balonAlfa`. Onların da birlikte `push` / `splice` edildiğini kontrol edin. Düzeyinizin üstünde bir parça gelirse "daha basit anlat ya da sadeleştir" demek doğaldır.

Yapay zekâyla: paralel diziden nesneye

 15-20 dk

1. $x[i]$, $y[i]$, $r[i]$ için bir nesne biçimi tasarlayın: $\{x, y, r\}$.
2. Yapay zekâdan paralel dizi $\leftrightarrow$ nesne dizisi dönüşümünün iskeletini isteyin. Henüz sınıf kullanmayın.
3. Bu aşamada hangi modelin daha okunur olduğunu bir cümleyle not edin.

MAKS = 3. Bu çağrılardan sonra tepe kaç, yığın nasıl?

sketch.js

Kopyala

Büyüt

```
yiginEkle(4); yiginEkle(9); yiginEkle(2); yiginEkle(7);  
yiginCikar();
```

A tepe 3 · [4, 9, 7]

☒ **✓** tepe 2 · [4, 9, -1]

C tepe 3 · [9, 2, 7]

D tepe 4 · [4, 9, 2, 7]

Dördüncü ekleme reddedilir: yer yok. Sonra en üstteki 2 çıkar. Dizi asla 3 yuvayı aşmaz.

`d = [5, 8, 1, 6]` iken `d.splice(1, 1)` sonrası `d[2]` kaç?

A 1

✓ 6

C 8

D undefined

8 çıkar; dizi `[5, 1, 6]` olur. Sonraki elemanlar bir yuva sola kaydığı için indeksleri değişir.

Can sayısı 0 olabilen bir oyunda, “bu yuvada oyuncu yok” işareti olarak hangisi güvenli?

A

0

✓

-1

C

3

D

Hepsi olur

0 geçerli bir can değeri; “boş” ile karışır. İşaret, oyunda asla oluşmayacak bir değer olmalı.

Merak edenler için kaynaklar

THE CODING TRAIN

7.5 Removing Objects from Arrays

`splice` ile diziden eleman çıkarmak; balon patlatmanın tam karşılığı.

[thecodingtrain.com > 7-arrays > 5-removing-elements](https://thecodingtrain.com/7-arrays/5-removing-elements)

THE CODING TRAIN

7.4 Mouse Interaction with Objects

Tıklanan nesneyi bulmak.

[thecodingtrain.com > 7-arrays > 4-mouse-interaction](https://thecodingtrain.com/7-arrays/4-mouse-interaction)

PATT VİRA

Array of Objects

Paralel dizilerden sonraki adım: her balonu tek bir nesnede tutmak.

pattvira.com/intro-to-creative-coding/array-of-objects

PATT VİRA

Changing Size of Array

Diziyi çalışırken büyütüp küçültmek.

pattvira.com/intro-to-creative-coding/changing-size-of-array

Bugün ne öğrendik?

- ✓ Sabit kapasite: dizi baştan `MAKS` uzunlukta kurulur, boş yuvalar `B0S` ile işaretlenir.
- ✓ `tepe` hem dolu yuva sayısını hem sıradaki boş yuvayı gösterir.
- ✓ Ekleme: `yigin[tepe] = deger; tepe++`. Çıkarma: `tepe--` ve yuvaya `B0S`.
- ✓ `B0S` geçerli bir oyun değeriyle çakışmamalı.
- ✓ Paralel dizilerde `push`, `pop`, `splice` her zaman bütün dizilerde birlikte yapılır.
- ✓ `splice(i, 1)` aradan eleman çıkarır; sonrakiler sola kayar.

SIRADAKİ DERS

Kontrol oyuncuya geçiyor

Klavye: basılı tutmak, bir kez basmak ve tuş kodları.

Sonraki sunum →

← Önceki