

Za radoznale: Sudari i pravila igre

Ovo nije obavezno i neće biti na ispitu. Za one koji žele da vide kako se ista stvar radi naprednije. Svi ispisi su dobijeni pokretanjem u pygame-ce.

1. Tunelovanje: brz predmet prođe kroz tanak zid

Sudar proveravamo jednom u frejmu. Ako se predmet u jednom frejmu pomeri više nego što je zid debeo, može da bude ispred zida u jednom frejmu, a iza zida u sledećem, pa sudar nikad ne bude otkriven. To se zove tunelovanje. Rešenje je da se dugačak pomeraj podeli na nekoliko manjih koraka i da se sudar proveri posle svakog.

```
import pygame as pg

zid = pg.Rect(120, 0, 10, 200)          # zid debljine 10 piksela
for brzina in (300, 3000):              # piksela u sekundi
    metak = pg.Rect(0, 50, 6, 6)
    pogodak = False
    for frejm in range(40):              # 40 frejmova pri 60 FPS
        metak.x += brzina // 60
        if metak.colliderect(zid):
            pogodak = True
    print(brzina, "px/s, pogodio zid:", pogodak)

# rešenje: isti pomeraj u pet manjih koraka
metak = pg.Rect(0, 50, 6, 6)
pogodak = False
for frejm in range(40):
    for korak in range(5):
        metak.x += 3000 // 60 // 5
        if metak.colliderect(zid):
            pogodak = True
print("u pet koraka, pogodio zid:", pogodak)
```

Ispis:

```
300 px/s, pogodio zid: True
3000 px/s, pogodio zid: False
u pet koraka, pogodio zid: True
```

Pri 3000 piksela u sekundi metak se u svakom frejmu pomeri za 50 piksela: posle drugog frejma je na 100, a posle trećeg na 150, dok je zid od 120 do 130. Nijedan položaj nije u zidu. Kad isti pomeraj podelimo na pet koraka od po 10 piksela, metak bar jednom završi u zidu.

Zašto ovo nismo radili na času: u našoj igri ništa se ne kreće toliko brzo, a pahuljice i grudve su deblje od pomeraja u jednom frejmu.

Fajl: `primeri/za_radoznale/10_r1_tunelovanje.py`

2. Gde i koliko se maske preklapaju

`collide_mask` vraća tačku prvog dodira ili `None`. Maska ume i više: `overlap` vraća prvu tačku u kojoj se dve maske preklapaju, a `overlap_area` broj piksela preklapanja. Drugi argument je pomak druge maske u

odnosu na prvu. Broj piksela preklapanja može da posluži, na primer, da jači udarac oduzme više života.

```
import pygame as pg

krug = pg.Surface((20, 20), pg.SRCALPHA)
pg.draw.circle(krug, "white", (10, 10), 10)
a = pg.mask.from_surface(krug)
b = pg.mask.from_surface(krug)
for pomak in [(15, 0), (25, 0)]:      # b je pomeren udesno
    print(pomak, a.overlap(b, pomak), a.overlap_area(b, pomak))
```

Ispis:

```
(15, 0) (17, 4) 44
(25, 0) None 0
```

Kad je drugi krug pomeren za 15 piksela, maske se preklapaju u uskom sočivu: overlap vraća prvu takvu tačku, a overlap_area broj piksela u njemu. Pri pomaku od 25 krugovi se ne dodiruju, pa je odgovor None i 0.

Zašto ovo nismo radili na času: na času nam je bilo dovoljno da znamo da li se dodiruju; pomak između maski zahteva računanje sa koordinatama dva pravougaonika.

Fajl: `primeri/za_radoznale/10_r2_preklapanje_maski.py`

3. clamp_ip: pravougaonik ostaje u prozoru

U igri smo igrača držali u prozoru preko `pg.math.clamp` za x koordinatu. `Rect` ima metodu `clamp_ip`, koja pomeri pravougaonik unutar drugog pravougaonika, na primer prozora, u jednom redu. Nastavak `_ip` znači „u mestu“ (in place): menja se sam pravougaonik, a ne pravi se nov.

```
import pygame as pg

prozor = pg.Rect(0, 0, 640, 360)
igrac = pg.Rect(620, -10, 40, 64)      # izašao desno i gore
igrac.clamp_ip(prozor)
print(igrac.topleft)
nov = pg.Rect(-30, 400, 40, 64).clamp(prozor)
print(nov.topleft)
```

Ispis:

```
(600, 0)
(0, 296)
```

`clamp_ip` je pomerio igrača tako da mu desna ivica bude na 640, a gornja na 0. Metoda `clamp` bez `_ip` ne menja pravougaonik, nego vraća nov, već pomeren.

Zašto ovo nismo radili na času: naš igrač čuva položaj u `Vector2`, a pravougaonik se pravi iz njega u svakom frejmu, pa granice postavljamo na položaj.

Fajl: `primeri/za_radoznale/10_r3_clamp_ip.py`