

Petlja igre, vreme i kretanje

Kako da se stvari u igri pomeraju jednako brzo na svakom računaru

Danas pravimo: sneg koji pada u igri „Zimska šetnja“



60 pahuljica, svaka svojom brzinom, lagano se
njišu dok padaju

Tri pitanja za zagrevanje

1 • prošla tema: Rect

```
import pygame as pg

r = pg.Rect(0, 0, 60, 20)
r.center = (50, 50)
print(r.left, r.top)
```

20 40

2 • atribut klase

```
class Lik:
    brzina = 2

a = Lik()
b = Lik()
a.brzina = 5
print(a.brzina, b.brzina)
```

5 2

3 • mešano: lista listi

```
sneg = [[10, 0], [50, 5]]
for p in снег:
    p[1] += 3
print(sneg)
```

[[10, 3], [50, 8]]

Šest novih reči za danas

frejm (kadar)

jedna slika igre; igra je niz frejmova, kao crtani film

FPS

broj frejmova u sekundi (frames per second)

petlja igre

petlja koja u svakom prolazu pravi jedan frejm

sat

objekat `pg.time.Clock()` koji meri vreme i usporava petlju

dt

vreme od prošlog frejma, u sekundama (delta t)

vektor

par brojeva (x, y): položaj ili brzina kao strelica

Kretanje u igri je niz mirnih slika



frejm 1



frejm 2



frejm 3



frejm 4

- Svaki frejm je mirna slika; lopta je u svakom malo pomerenjena.
- Kad se smenjuje 30 do 60 slika u sekundi, oko vidi glatko kretanje.
- Program zato u petlji ponavlja isto: pomeri, nacrtaj, prikaži.

Petlja može da čeka događaj ili da stalno crta

Vođena događajima

```
radi = True
while radi:
    dogadjaj = pg.event.wait() # čeka
    if dogadjaj.type == pg.QUIT:
        radi = False
    # crtamo samo kad se nešto desi
```

Za crtež ili igru na poteze: ništa se ne pomera samo od sebe.

Vođena frejmovima

```
radi = True
while radi:
    for dogadjaj in pg.event.get():
        if dogadjaj.type == pg.QUIT:
            radi = False
    # pomeramo i crtamo u svakom frejmu
    sat.tick(60)
```

Za sve što se kreće: frejm za frejmom, čak i kad igrač ništa ne radi.

Petlja igre

Četiri koraka koja se ponavljaju šezdeset puta u sekundi

Svaki frejm ima četiri koraka, uvek istim redom



Prvo menjamo stanje, pa tek onda crtamo. Crtanje ne sme ništa da pomera.

Kostur svake igre ima petnaest redova

```
import pygame as pg

pg.init()
ekran = pg.display.set_mode((640, 360))
sat = pg.time.Clock()
radi = True
while radi:
    for dogadjaj in pg.event.get(): # 1. događaji
        if dogadjaj.type == pg.QUIT:
            radi = False
    # 2. ažuriranje: ovde menjamo položaje
    ekran.fill("midnightblue") # 3. crtanje
    pg.display.flip() # 4. prikaz
    sat.tick(60) # najviše 60 FPS
pg.quit()
```

`sat = pg.time.Clock()`

pravimo sat jednom, pre petlje

`pg.event.get()`

uzima sve događaje od prošlog frejma i ne čeka

`pg.display.flip()`

prikazuje ceo nacrtani frejm odjednom

`sat.tick(60)`

najviše 60 frejmova u sekundi

wait uvek čeka isto, a sat čeka samo koliko treba

Radi, ali čeka uvek isto: `wait(50)`

```
while radi:
    obradi_dogadjaje()
    azuriraj()
    crtaj()
    pg.display.flip()
    pg.time.wait(50) # uvek 50 ms
```

Bolje u petlji igre: `sat.tick(20)`

```
sat = pg.time.Clock()
while radi:
    obradi_dogadjaje()
    azuriraj()
    crtaj()
    pg.display.flip()
    sat.tick(20) # koliko treba
```

Ako crtanje traje 30 ms: wait čeka još 50 ms, pa frejm traje 80 ms (12 FPS). Sat računa: do 50 ms fali još 20, pa frejm traje tačno 50 ms (20 FPS).

Čikica iz starih beleški postaje klasa

```
class Cikica:
    def __init__(self, x, y):
        self.x = x
        self.y = y
        self.dx = 2                # piksela po frejmu
```

```
    def azuriraj(self):
        self.x += self.dx
        if self.x < 0 or self.x + SIRINA_SLIKE > SIRINA:
            self.dx = -self.dx    # stigao je do ivice
```

```
    def crtaj(self, ekran):
        slika = desno if self.dx > 0 else levo
        ekran.blit(slika, (self.x, self.y))
```



na ivici: $dx = -dx$

- azuriraj menja samo brojeve.
- crtaj samo crta; ništa ne pomera.
- Znak broja dx je smer: + desno, – levo.

Koliko piksela u sekundi pređe ovaj autić?

```
sat = pg.time.Clock()
x = 0
while radi:
    ...
    x += 5      # piksela po frejmu
    ...
    sat.tick(20)
```

Pretpostavite da računar stigne svih 20 frejmova u sekundi.

A
5

B
20

C
100

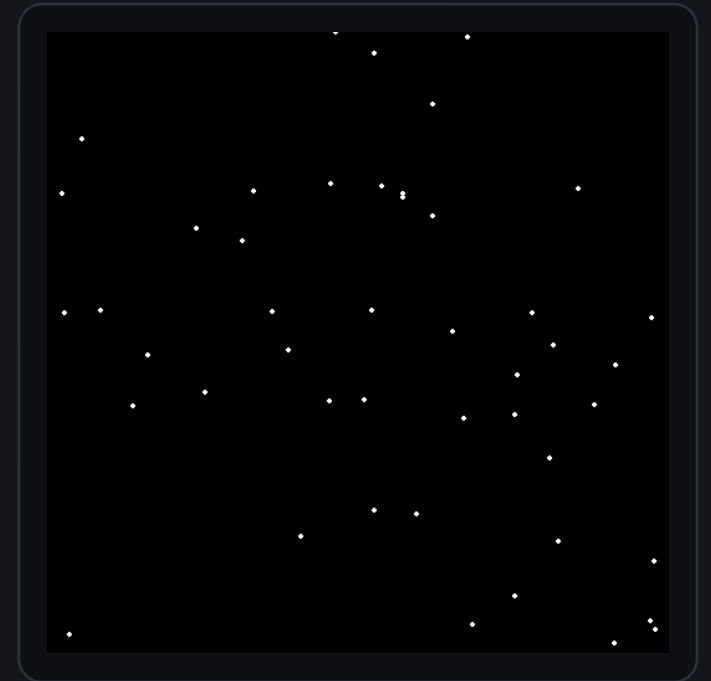
tačno

D
25

Pahuljice padaju: svaka je lista [x, y]

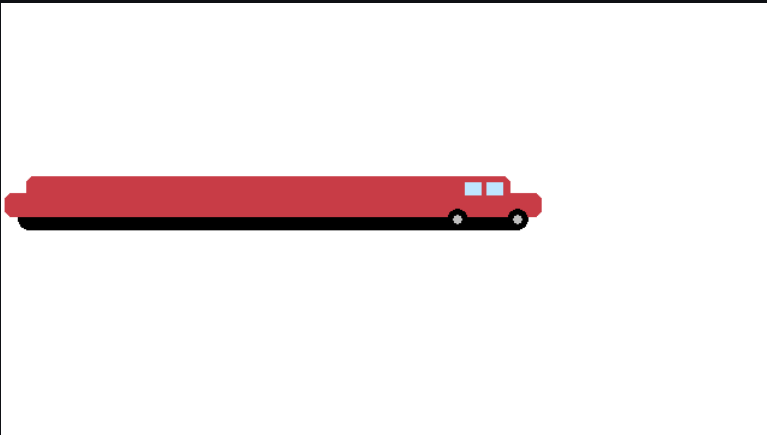
```
for p in pahuljice:      # p je [x, y]
    p[1] += 1            # piksel niže
    if p[1] > VISINA:    # ispala je dole
        p[0] = random.randrange(0, SIRINA)
        p[1] = random.randrange(-40, -10)

ekran.fill("black")
for p in pahuljice:
    pg.draw.circle(ekran, "white", p, 2)
```



Zadatak iz starih beleški: 50
tačkica, 20 frejmova u sekundi.

Čudovište: trag



autić bez ekran.fill(...)

Kako izgleda

Predmet koji se kreće ostavlja za sobom traku, kao četkica po platnu.

Zašto se pojavljuje

Ekran se ne briše na početku crtanja, pa stari frejmovi ostaju ispod novog.

Kako se pobeđuje

Na početku crtanja obojite ceo ekran (fill) ili nacrtajte pozadinu preko svega.

Vreme i brzina

Ista igra na sporom i na brzom računaru

Brzina $\times$ dt ne zavisi od računara

```
for fps in (20, 60):  
    x_frejm = 0  
    x_dt = 0  
    for frejm in range(fps):      # 1 sekunda  
        x_frejm += 5  
        x_dt += 100 * (1 / fps)  
    print(fps, "FPS:", x_frejm, round(x_dt))
```

ISPIS

```
20 FPS: 100 100  
60 FPS: 300 100
```

Na brzom računaru (60 FPS) autić ide tri puta brže

ako ga pomeramo za 5 piksela po frejmu. Kada brzinu zadamo u pikselima u sekundi i pomnožimo je vremenom frejma, za jednu sekundu uvek pređe 100 piksela.

tick vraća milisekunde od prošlog frejma

```
import pygame as pg

sat = pg.time.Clock()
sat.tick()
pg.time.wait(30)    # frejm od 30 ms
print(sat.tick())
```

ISPIS

30

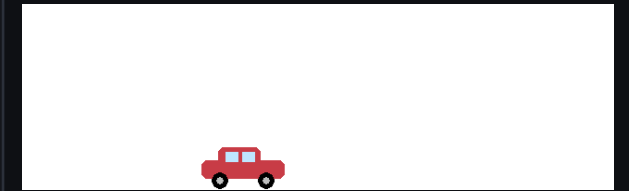
Zato u igri pišemo:

```
brzina = 100                # piksela u sekundi
dt = sat.tick(60) / 1000    # u sekundama
x += brzina * dt
```

1000 ms = 1 s

Autić sa dt: 100 piksela u sekundi svuda

```
brzina = 100                                # piksela u sekundi
x = 0
radi = True
while radi:
    for dogadjaj in pg.event.get():
        if dogadjaj.type == pg.QUIT:
            radi = False
    dt = sat.tick(60) / 1000
    x += brzina * dt
    if x > SIRINA:                            # izašao je desno
        x = -autic.get_width()                # vraćamo ga levo
    ekran.fill("white")
    ekran.blit(autic, (x, VISINA - autic.get_height()))
    pg.display.flip()
```



- dt je mali broj: oko 0,017 pri 60 FPS.
- x je sada realan broj; blit ga zaokruži.
- Ceo prozor od 640 piksela: oko 6,4 sekunde.

Šta ispisuje ovaj program?

```
x = 0  
brzina = 120      # px u sekundi  
for dt in [0.5, 0.5, 0.5]:  
    x += brzina * dt  
print(x)
```

Tri frejma od po pola sekunde: ovo je vrlo spor računar.

A

180

B

tačno

180.0

C

360.0

D

1.5

Vektori

Položaj i brzina kao strelice

Vector2 čuva x i y zajedno i računa sa njima

```
import pygame as pg

polozaj = pg.Vector2(100, 50)
brzina = pg.Vector2(30, 40)
print(polozaj + brzina)
print(brzina * 0.5)
print(brzina.length())
```

ISPIS

```
[130, 90]
[15, 20]
50.0
```

polozaj (100, 50)



brzina (30, 40)

dužina strelice: 50

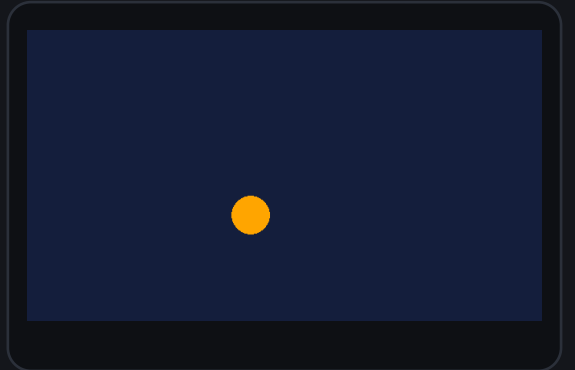
zbir (130, 90)



Lopta se odbija: brzina menja znak

```
def azuriraj(self, dt):  
    self.polozaj += self.brzina * dt  
    p = self.polozaj          # kraće ime za isti vektor  
    r = self.poluprecnik  
    if p.x < r or p.x > SIRINA - r:  
        self.brzina.x = -self.brzina.x  
        p.x = pg.math.clamp(p.x, r, SIRINA - r)  
    if p.y < r or p.y > VISINA - r:  
        self.brzina.y = -self.brzina.y  
        p.y = pg.math.clamp(p.y, r, VISINA - r)
```

`pg.math.clamp(broj, od, do)` vraća broj, ali ne manji od „od“ i ne veći od „do“.



```
clamp(-7, 20, 620) → 20  
clamp(300, 20, 620) → 300  
clamp(655, 20, 620) → 620
```

Šta ispisuje ovaj program?

```
x = 90
dx = 10
for frejm in range(3):
    x += dx
    if x >= 100:
        dx = -dx
    print(x, end=" ")
```

Pažljivo: x se prvo poveća, pa se tek onda proverava.

A

100 110 120

B

tačno

100 90 80

C

100 100 100

D

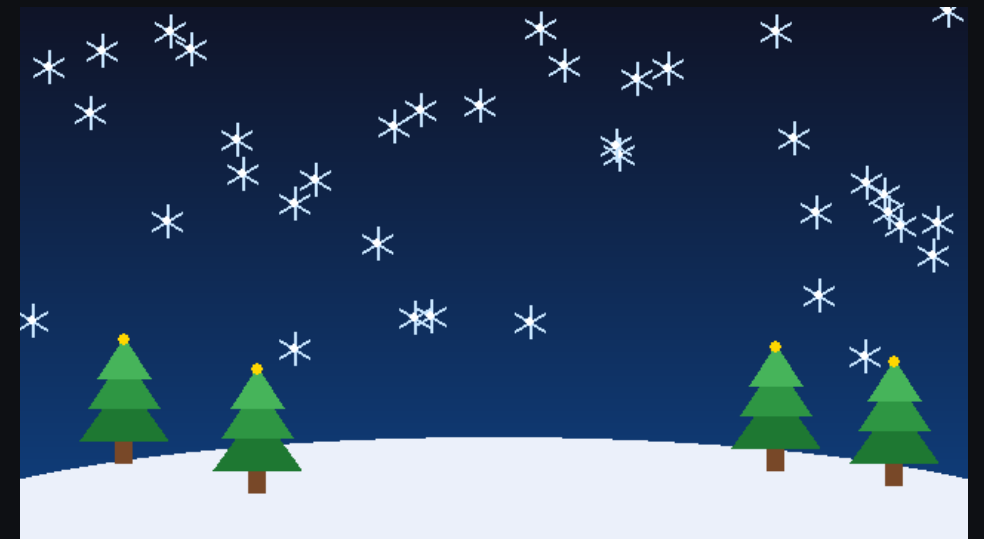
90 80 70

Zimska šetnja, korak 2: pada sneg

Cilj: za 20 minuta po zimskoj sceni pada sneg, a svaka pahuljica ima svoju brzinu.

- 1 Klasa Pahuljica: x, y, brzina u px/s
- 2 azuriraj(dt): pada brzina $\times$ dt
- 3 Ispala je dole? Vрати je iznad prozora
- 4 Njihanje levo-desno: sinus od vremena
- 5 U petlji: prvo sve ažuriramo, pa sve crtamo

KAKO IZGLEDA NA KRAJU



pozadina.png i pahuljica.png pravi program 07_01

Sinus pravi gladko njihanje od -1 do 1

```
import math

for t in [0, 0.25, 0.5, 0.75, 1]:
    pomeraj = 15 * math.sin(2 * math.pi * t)
    print(t, round(pomeraj))
```

ISPIS

```
0 0
0.25 15
0.5 0
0.75 -15
1 0
```

- Za t od 0 do 1 pahuljica ode 15 piksela desno, vrati se, ode 15 levo i opet se vrati.
- U igri umesto t stavljamo vreme od početka igre, pa njihanje traje dok igra traje.

Svaka pahuljica pamti svoju brzinu

```
class Pahuljica:
    def __init__(self):
        self.x0 = random.uniform(0, SIRINA)    # sredina njihanja
        self.x = self.x0
        self.y = random.uniform(-VISINA, 0)    # iznad prozora
        self.brzina = random.uniform(40, 120)  # px u sekundi
        self.faza = random.uniform(0, 6.28)

    def azuriraj(self, dt, vreme):
        self.y += self.brzina * dt
        if self.y > VISINA:
            self.y = -20
        self.x = self.x0 + 15 * math.sin(2 * vreme + self.faza)

    def crtaj(self, ekran):
        ekran.blit(SLIKA, (self.x, self.y))
```

`random.uniform(a, b)`

realan slučajan broj
između a i b

`self.x0`

sredina oko koje se
pahuljica njiše

`self.faza`

da se ne njišu sve u istom
ritmu

Prvo ažuriramo sve, pa crtamo sve

```
vreme = 0
radi = True
while radi:
    for dogadjaj in pg.event.get():
        if dogadjaj.type == pg.QUIT:
            radi = False
    dt = sat.tick(60) / 1000
    vreme += dt
    for p in pahuljice:
        p.azuriraj(dt, vreme)
    ekran.blit(POZADINA, (0, 0))
    for p in pahuljice:
        p.crtaj(ekran)
    pg.display.flip()
```

dt

jedan put po frejmu, iz sata

vreme += dt

sekunde od početka igre, za sinus

POZADINA

crtta se prva i briše stari frejm

Pronađi grešku: autić nestane iz prozora

```
brzina = 100      # px u sekundi
x = 0
while radi:
    ...
    dt = sat.tick(60)
    x += brzina * dt
    ekran.blit(autic, (x, 140))
```

Program se ne ruši i ne javlja grešku. Zašto autić odmah ode?

ISPIS: `print(round(x))`

```
1600
3200
4800
6400
```

sat.tick(60) vraća milisekunde, npr. 16

100 * 16 1600 piksela u jednom frejmu

/ 1000 `dt = sat.tick(60) / 1000`

Izaberite nivo i počnite

LEVEL 1

Autić za 4 sekunde

Autić prelazi prozor širine 640 za tačno četiri sekunde, na svakom računaru.

Pomoć: koliko piksela u sekundi?

LEVEL 2

Čikica sa dt

Prepravite Čikicu da šeta brzinom 150 piksela u sekundi.

Pomoć: azuriraj(self, dt), a znak brzine je smer.

LEVEL 3

Sneg u tri sloja

Daleke pahuljice su male i spore, bliske velike i brze.

Pomoć: klasa sa parametrom sloj (1, 2 ili 3).

Sledeći put lik sluša tastaturu

```
tasteri = pg.key.get_pressed()
if tasteri[pg.K_RIGHT]:
    igrac.x += igrac.brzina * dt
if tasteri[pg.K_LEFT]:
    igrac.x -= igrac.brzina * dt
```

U temi 08 lik iz Zimske šetnje hoda levo i desno dok držimo strelicu, a tajmer svakih nekoliko desetinki pravi novu pahuljicu. Brzina lika je opet u pikselima u sekundi, pomnožena sa dt.

Rezime

petlja igre

dogadjaji → ažuriranje
→ crtanje → flip()

Isti redosled u svakom frejmu.

sat

```
sat = pg.time.Clock()  
sat.tick(60)
```

Najviše 60 frejmova u sekundi.

dt

```
dt = sat.tick(60) / 1000  
x += brzina * dt
```

Brzina je u pikselima u sekundi.

Vector2

```
polozaj += brzina * dt
```

x i y zajedno, u jednom redu.

Pitanje pre odlaska

```
brzina = 90          # px u sekundi  
dt = sat.tick(30) / 1000  
x += brzina * dt
```

Za koliko piksela se predmet pomeri u jednom frejmu, ako računar stiže svih 30 frejmova u sekundi?

Pošaljite jedan broj pre nego što izađete.



Literatura

- Dokumentacija za pygame-ce, moduli time i math: pyga.me/docs
- Kafle, S. Learning Python by Building Games, Packt, 2019 (poglavlja o PyGame-u)
- Sweigart, A. Invent Your Own Computer Games with Python, 4. izdanje, No Starch Press (besplatno na inventwithpython.com)
- Nystrom, R. Game Programming Patterns, poglavlje „Game Loop“ (besplatno na gameprogrammingpatterns.com)

Dodatna literatura i zanimljivi sajtovi

pygame-ce: modul time

pyga.me/docs/ref/time.html

Sat (Clock) i tick: koliko je milisekundi prošlo od prethodnog frejma.

pygame-ce: Vector2

pyga.me/docs/ref/math.html

Sve što vektor ume: sabiranje, dužina, normalizacija, rotacija.

Game Programming Patterns: Game Loop

gameprogrammingpatterns.com/game-loop.html

Poglavlje besplatne knjige Roberta Nistroma o petlji igre, sa crtežima.

Fix Your Timestep!

[gafferongames.com · Fix Your Timestep!](https://gafferongames.com/fix-your-timestep/)

Čuveni tekst Glena Fidlera o tome kako igre računaju vreme.

Hvala!

Pitanja su uvek dobrodošla: na času, u četvrtu ili međim.

Slajdovi, primeri i kartica za podsećanje su na sajtu predmeta.

PREDAVANJA

dr Staša Vujičić Stanković

stasa.vujicic.stankovic@math.rs

www.matf.bg.ac.rs/~stasa.vujicic.stankovic

VEŽBE

Milan Kocić

milan.kocic@matf.bg.ac.rs

poincare.matf.bg.ac.rs/~milan.kocic