

Za radoznale: PyGame

Ovaj fajl je za one koji žele više. Ništa odavde nije potrebno za praćenje časova. Svaki deo počinje primerom koji smo radili na času, a zatim pokazuje kako se ista stvar može uraditi naprednije. Svi primeri su i u folderu primeri/za_radoznale, pa ih možete pokrenuti, menjati i gledati šta se dešava. Programi koji otvaraju prozor rade dok ne kliknete na X; slike ispod njih su snimljene posle nekoliko kadrova.

1. clock.tick i dt: brzina u pikselima po sekundi

Na kraju časa smo u najavi videli prvu animaciju. Krug se u svakom kadru pomera za 3 piksela, a sat ograničava petlju na najviše 60 kadrova u sekundi:

```
import pygame as pg

pg.init()
(sirina, visina) = (400, 300)
ekran = pg.display.set_mode((sirina, visina))
pg.display.set_caption("Prva animacija")

sat = pg.time.Clock()
x = 0
kraj = False
while not kraj:
    for dogadjaj in pg.event.get():
        if dogadjaj.type == pg.QUIT:
            kraj = True
    ekran.fill("black")
    pg.draw.circle(ekran, "yellow", (x, 150), 20)
    pg.display.update()
    x = (x + 3) % sirina
    sat.tick(60) # najviše 60 kadrova u sekundi
pg.quit()
```

Ispis:

(u konzoli se ništa ne ispisuje; žuti krug putuje sleva nadesno)

Ako je računar spor i stigne samo 30 kadrova u sekundi, ovaj krug ide dvaput sporije. Igre to rešavaju ovako: sat.tick vraća koliko je milisekundi prošlo od prošlog kadra. Kad to podelimo sa 1000, dobijamo dt, vreme u sekundama. Brzinu onda zadajemo u pikselima po sekundi, a pomeraj u kadru je brzina * dt. Na brzom i na sporom računaru krug za jednu sekundu pređe isti put.

```
import pygame as pg

pg.init()
(sirina, visina) = (400, 300)
ekran = pg.display.set_mode((sirina, visina))
pg.display.set_caption("Kretanje sa dt")

sat = pg.time.Clock()
x = 0.0
brzina = 180 # piksela u sekundi
kraj = False
while not kraj:
    for dogadjaj in pg.event.get():
        if dogadjaj.type == pg.QUIT:
            kraj = True
    dt = sat.tick(60) / 1000 # sekunde od prošlog kadra
    x = (x + brzina * dt) % sirina
    ekran.fill("black")
    pg.draw.circle(ekran, "yellow", (round(x), 150), 20)
    pg.display.update()
pg.quit()
```

Ispis:

(u konzoli se ništa ne ispisiuje; žuti krug putuje sleva nadesno brzinom od 180 piksela u sekundi)



Žuti krug na crnoj pozadini

x je sada realan broj, jer brzina * dt retko daje ceo broj. Za crtanje ga zaokružujemo funkcijom round.

Zašto ovo nismo radili na času: Animacija i petlja igre su tema predmeta PZU2. Na ovom času smo crtali mirne slike, pa nam vreme između kadrova nije trebalo.

2. Vector2: položaj i brzina kao jedna vrednost

Na času smo položaj čuvali kao dva broja, x i y, pa smo svaki menjali posebno:

```
(x, y) = (100, 50)
(dx, dy) = (3, 4)
x = x + dx
y = y + dy
print(x, y)
```

Ispis:

103 54

PyGame ima Vector2, vrednost koja u sebi čuva i x i y. Dva vektora se sabiraju jednim znakom plus, a vektor ume i da izmeri svoju dužinu. Metoda normalize pravi vektor dužine 1 u istom smeru, što je zgodno kad lik treba da ide ka nekoj tački.

```

from pygame.math import Vector2

polozaj = Vector2(100, 50)
brzina = Vector2(3, 4)
polozaj += brzina
print(polozaj.x, polozaj.y)
print(brzina.length())

cilj = Vector2(403, 454)
smer = (cilj - polozaj).normalize()
print(round(smer.x, 2), round(smer.y, 2))

```

Ispis:

```

103.0 54.0
5.0
0.6 0.8

```

Brzina (3, 4) ima dužinu 5, jer je $3 \cdot 3 + 4 \cdot 4 = 25$. Smer ka cilju je (0.6, 0.8): vektor pokazuje ka cilju, a dužina mu je 1, pa ga možemo pomnožiti bilo kojom brzinom koju želimo.

Zašto ovo nismo radili na času: Na času smo hteli da x i y vidimo odvojeno, da bi koordinatni sistem bio jasan. Vektori olakšavaju kretanje, a kretanje radimo tek kasnije.

3. Rect: da li se dva oblika dodiruju

Na času smo videli da Rect pamti levu ivicu, vrh, širinu i visinu i da iz njih računa ostale ivice:

```

import pygame as pg

r = pg.Rect(50, 40, 100, 60)
print(r.right, r.bottom)

```

Ispis:

```

150 100

```

U igri nas stalno zanima da li je lik dodirnuo zid ili novčić. Rect to zna sam: colliderect kaže da li se dva pravougaonika preklapaju, a collidepoint da li je tačka, na primer mesto klika mišem, unutar pravougaonika. Metoda move_ip pomera pravougaonik za zadati pomeraj.

```

import pygame as pg

lik = pg.Rect(50, 50, 40, 40)
zid = pg.Rect(80, 60, 100, 20)
novcic = pg.Rect(300, 200, 10, 10)

print(lik.colliderect(zid))
print(lik.colliderect(novcic))
print(lik.collidepoint(60, 60))

lik.move_ip(250, 150)
print(lik.topleft, lik.colliderect(novcic))

```

Ispis:

```
True
False
True
(300, 200) True
```

Zašto ovo nismo radili na času: Bez kretanja i tastature sudari nemaju mnogo smisla. Na času smo Rect koristili samo da sliku postavimo na pravo mesto.

4. Sprite i grupa: mnogo pahuljica odjednom

Na času smo dvadeset pahuljica nacrtali petljom, a za svaku smo sami računali mesto:

```
import pygame as pg
import random

pg.init()
(sirina, visina) = (400, 300)
ekran = pg.display.set_mode((sirina, visina))
ekran.fill((20, 30, 60))

slika = pg.Surface((12, 12), pg.SRCALPHA)
pg.draw.circle(slika, "white", (6, 6), 6)
for i in range(20):
    x = random.randint(0, sirina - slika.get_width())
    y = random.randint(0, visina - slika.get_height())
    ekran.blit(slika, (x, y))

pg.display.update()
while pg.event.wait().type != pg.QUIT:
    pass
pg.quit()
```

Ispis:

(u konzoli se ništa ne ispisuje; dvadeset belih pahuljica na nasumičnim mestima)

Kad pahuljice treba i da padaju, zgodno je da svaka sama pamti svoju sliku i svoj položaj. U PyGame-u za to služi sprite: napravimo svoju vrstu objekta, Pahuljica, koja ima sliku (image), okvir (rect) i metodu update koja je pomera naniže. Sve pahuljice stavimo u grupu, pa grupa.update() pomera sve odjednom, a grupa.draw(ekran) sve crta. Reč class pravi novu vrstu objekta; o tome se više uči u objektno orijentisanom programiranju.

```

import pygame as pg
import random

class Pahuljica(pg.sprite.Sprite):
    def __init__(self, x, y):
        super().__init__()
        self.image = pg.Surface((12, 12), pg.SRCALPHA)
        pg.draw.circle(self.image, "white", (6, 6), 6)
        self.rect = self.image.get_rect(center=(x, y))

    def update(self):
        self.rect.y += 2
        if self.rect.top > 300:  # ispala je ispod dna, vraćamo je na vrh
            self.rect.bottom = 0

pg.init()
ekran = pg.display.set_mode((400, 300))
pg.display.set_caption("Pahuljice kao sprite")
sat = pg.time.Clock()

grupa = pg.sprite.Group()
for i in range(40):
    grupa.add(Pahuljica(random.randint(0, 400), random.randint(0, 300)))
print("Pahuljica u grupi:", len(grupa))

kraj = False
while not kraj:
    for dogadjaj in pg.event.get():
        if dogadjaj.type == pg.QUIT:
            kraj = True
    grupa.update()
    ekran.fill((20, 30, 60))
    grupa.draw(ekran)
    pg.display.update()
    sat.tick(60)
pg.quit()

```

Ispis:

```
Pahuljica u grupi: 40
```



pahuljice koje padaju

Pored ispisa u konzoli otvara se prozor u kom 40 pahuljica pada, a svaka koja ispadne ispod dna vraća se na vrh.

Zašto ovo nismo radili na času: Sprite se pravi pomoću klase, a klase su sasvim nov način pisanja programa. Za mirne slike sa časa petlja i blit su potpuno dovoljni.

5. Providne boje: krugovi koji se mešaju

Na času smo videli da boja ima i četvrti broj, providnost (alfa), ali smo uvek crtali neprovidno:

```

import pygame as pg

print(pg.Color("orange"))

```

Ispis:

(255, 165, 0, 255)

Ako crtamo na posebnu površinu napravljenu sa `pg.SRCALPHA`, četvrti broj boje se poštuje: 255 je potpuno neprovidno, a 0 potpuno providno. Ovde tri kruga sa providnošću 120 crtamo svaki na svoju površinu i lepimo jedan preko drugog, pa se boje mešaju kao akvarel. Na kraju ispisujemo boju piksela u sredini, gde se sva tri kruga preklapaju.

```
import pygame as pg

pg.init()
ekran = pg.display.set_mode((400, 300))
pg.display.set_caption("Providni krugovi")
ekran.fill("white")

krugovi = [((255, 0, 0, 120), (170, 130)),
            ((0, 160, 0, 120), (230, 130)),
            ((0, 0, 255, 120), (200, 180))]
for boja, centar in krugovi:
    sloj = pg.Surface((400, 300), pg.SRCALPHA)
    pg.draw.circle(sloj, boja, centar, 80)
    ekran.blit(sloj, (0, 0))

print(ekran.get_at((200, 145)))
pg.display.update()
while pg.event.wait().type != pg.QUIT:
    pass
pg.quit()
```

Ispis:

(71, 78, 158, 255)



tri providna kruga koja se preklapaju

Pored ispisa u konzoli otvara se prozor sa tri kruga; tamo gde se preklapaju nastaju nove, tamnije boje.

Zašto ovo nismo radili na času: Providnost traži posebnu površinu i lepljenje sloj po sloj, a to je jedan korak više od onoga što nam je trebalo za prve crteže.