

Za radoznale: Zvuk, transformacije slike i čestice

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. Melodija od nota: koja frekvencija je koji ton

Funkcija ton pravi zvuk zadate visine, u hercima (talasa u sekundi). Muzičke note imaju tačno određene frekvencije: ton A iznad srednjeg C je 440 Hz, a svaki sledeći polustepen je 2 na 1/12 puta viši. Posle 12 polustepeni frekvencija je tačno dvostruka: to je oktava. Tako od brojeva možemo da napravimo lestvicu ili melodiju.

```
import math
import struct
import wave

FREKVENCIJA = 22050
# polustepeni od tona A (440 Hz): C, D, E, F, G, A, H, C
POLUSTEPENI = [-9, -7, -5, -4, -2, 0, 2, 3]
uzorci = []
for n in POLUSTEPENI:
    visina = 440 * 2 ** (n / 12)
    print(round(visina, 1), end=" ")
    for i in range(int(FREKVENCIJA * 0.3)):
        t = i / FREKVENCIJA
        uzorci.append(0.4 * math.sin(2 * math.pi * visina * t) * math.exp(-4 * t))
with wave.open("lestvica.wav", "wb") as f:
    f.setnchannels(1)
    f.setsampwidth(2)
    f.setframerate(FREKVENCIJA)
    f.writeframes(b"".join(struct.pack("<h", int(u * 32000)) for u in uzorci))
print()
print("snimljeno uzoraka:", len(uzorci))
```

Ispis:

```
261.6 293.7 329.6 349.2 392.0 440.0 493.9 523.3
snimljeno uzoraka: 52920
```

Ispis su frekvencije lestvice C-dur, od srednjeg C (oko 261,6 Hz) do C oktavu više (oko 523,3 Hz, tačno dva puta više). Fajl lestvica.wav možete da pustite dvoklikom ili sa pg.mixer.Sound. Spajanje bajtova preko join je brže od sabiranja u petlji, kada je uzoraka mnogo.

Zašto ovo nismo radili na času: za igru su nam bila dovoljna tri zvuka, a muzička teorija nije deo ovog predmeta.

Fajl: `primeri/za_radoznale/11_r1_melodija_od_nota.py`

2. Čestice koje svetle: sabiranje boja (BLEND_ADD)

Obično blit crta sliku preko onoga što je već na ekranu. Sa `special_flags=pg.BLEND_ADD` boje se sabiraju: svaka čestica posvetli ono ispod sebe, a tamo gde se više čestica preklapa svetlo je najjače. Tako se prave vatra, iskre i magija koja „sija“.

```
import pygame as pg

ekran = pg.Surface((10, 10))
ekran.fill((40, 40, 80))           # tamnoplavo nebo
iskra = pg.Surface((10, 10))
iskra.fill((100, 100, 100))
ekran.blit(iskra, (0, 0), special_flags=pg.BLEND_ADD)
print(ekran.get_at((0, 0)))
ekran.blit(iskra, (0, 0), special_flags=pg.BLEND_ADD)
print(ekran.get_at((0, 0)))
```

Ispis:

```
Color(140, 140, 180, 255)
Color(240, 240, 255, 255)
```

Prva iskra je svakoj komponenti boje dodala 100. Druga je dodala još 100, ali komponenta ne može da pređe 255, pa je plava stala na 255. Zato se mnogo preklopljenih iskri približava beloj boji, kao pravo jako svetlo.

Zašto ovo nismo radili na času: na času smo čestice gasili providnošću (alfa); sabiranje boja je drugi način mešanja, koji traži da se razume kako se boje kombinuju.

Fajl: `primeri/za_radoznale/11_r2_blend_add.py`

3. Odras u vodi: flip i providnost zajedno

Transformacije se lako kombinuju. Odras lika u zaleđenom jezeru je ista slika okrenuta naopako (flip po y), malo providna i nacrtana tačno ispod lika.

```
import pygame as pg

lik = pg.Surface((40, 64), pg.SRCALPHA)
pg.draw.rect(lik, (60, 90, 200), (10, 0, 20, 40)) # telo gore
odraz = pg.transform.flip(lik, False, True)         # naopako
odraz.set_alpha(90)                                # jedva se vidi
print(odraz.get_size(), odraz.get_alpha())
print(lik.get_at((20, 5))[3], odraz.get_at((20, 5))[3])
print(lik.get_at((20, 60))[3], odraz.get_at((20, 60))[3])
```

Ispis:

```
(40, 64) 90
255 0
0 255
```

Odras je iste veličine, a njegova providnost je 90 od 255. Četvrti broj u rezultatu `get_at` (indeks 3) je alfa jednog piksela: telo lika je gore, pa je gornji piksel lika neprovidan (255), a kod odraza je telo dole. Odras se crta sa `ekran.blit(odraz, igrac.rect.bottomleft)`.

Zašto ovo nismo radili na času: naša scena nema jezero; ovo je ideja za seminarski rad.

Fajl: `primeri/za_radoznale/11_r3_odraz_u_vodi.py`