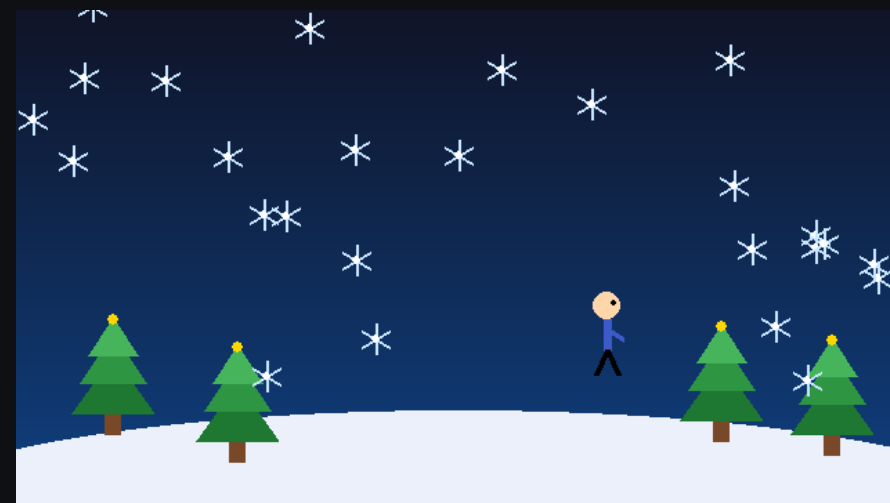


Događaji i unos

Kako igra sazna šta igrač radi: tastatura, miš i tajmeri

Danas pravimo: Čikica hoda strelicama i skače, a tajmer pravi nove pahuljice



Strelice: hod levo i desno · razmak: skok · ESC:
kraj

Tri pitanja za zagrevanje

1. prošla tema: dt

```
brzina = 150
x = 0
for dt in [0.1, 0.2, 0.1]:
    x += brzina * dt
print(round(x))
```

60

2. tema 05: __str__

```
class Lik:
    def __str__(self):
        return "Čikica"

c = Lik()
print(c, "hoda")
```

Čikica hoda

3. mešano: rečnik

```
smer = {"L": -1, "D": 1}
x = 100
for k in "DDL":
    x += 10 * smer[k]
print(x)
```

110

Šest novih reči za danas

dogadjaj

nešto što se desilo: taster, klik, pomeranje miša, X na prozoru

red dogadjaja

lista dogadjaja koji čekaju da ih program obradi

KEYDOWN / KEYUP

taster je upravo pritisnut / upravo pušten

get_pressed()

koji tasteri su dole baš u ovom frejmu

tajmer

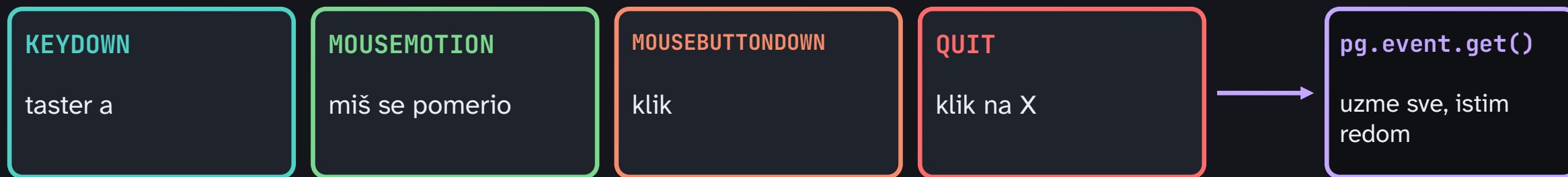
šalje dogadjaj u red na svakih nekoliko milisekundi

sopstveni dogadjaj

nova vrsta dogadjaja koju izmislimo mi

Događaji čekaju u redu, kao ljudi na šalteru

korisnik i tajmeri



- Biblioteka stavlja u red sve što se desilo od prošlog frejma.
- Petlja for u svakom frejmu uzme ceo red i obradi događaj po događaj.
- Red se posle `get()` isprazni: svaki događaj dobijamo samo jednom.

Možemo i sami da stavimo događaj u red

```
import pygame as pg

pg.init()
pg.display.set_mode((200, 200))
pg.event.clear()          # brišemo događaje samog prozora
taster = pg.event.Event(pg.KEYDOWN, key=pg.K_a, unicode="a")
klik = pg.event.Event(pg.MOUSEBUTTONDOWN, pos=(50, 80), button=1)
for d in [taster, klik, pg.event.Event(pg.QUIT)]:
    pg.event.post(d)       # stavljamo događaj u red
for dogadjaj in pg.event.get():
    print(pg.event.event_name(dogadjaj.type), dogadjaj.dict)
```

ISPIS

```
KeyDown {'key': 97, 'unicode': 'a'}
MouseButtonDown {'pos': (50, 80), 'button': 1}
Quit {}
```

`pg.event.Event(vrsta, ...)`
pravi događaj; posle vrste idu njegovi podaci

`pg.event.post(d)`
stavlja događaj na kraj reda

`dogadjaj.dict`
svi podaci događaja, kao rečnik

Svaka vrsta događaja nosi svoje podatke

vrsta (dogadjaj.type)	podaci	kada stiže
<code>pg.QUIT</code>	nema podataka	klik na X u uglu prozora
<code>pg.KEYDOWN</code>	<code>key</code> , <code>unicode</code>	taster je upravo pritisnut
<code>pg.KEYUP</code>	<code>key</code>	taster je upravo pušten
<code>pg.MOUSEBUTTONDOWN</code>	<code>pos</code> , <code>button</code>	klik: gde i kojim tasterom
<code>pg.MOUSEBUTTONUP</code>	<code>pos</code> , <code>button</code>	taster miša je pušten
<code>pg.MOUSEMOTION</code>	<code>pos</code> , <code>rel</code>	miš se pomerio (rel: za koliko)

Podatke čitamo tačkom: `dogadjaj.key`, `dogadjaj.pos`. Tajmeri šalju sopstvene vrste događaja.

Tastatura

Jedan pritisak ili držanje tastera

KEYDOWN stiže jednom, u trenutku pritiska

```
for dogadjaj in pg.event.get():
    if dogadjaj.type == pg.QUIT:
        radi = False
    elif dogadjaj.type == pg.KEYDOWN:
        if dogadjaj.key == pg.K_ESCAPE:
            radi = False
        elif dogadjaj.key == pg.K_SPACE:
            print("skok!")
        else:
            print("slovo:", dogadjaj.unicode)
```

```
import pygame as pg

print(pg.KEYDOWN, pg.KEYUP)
print(pg.K_a, pg.K_SPACE)
print(pg.key.name(pg.K_LEFT))
```

ISPIS

```
768 769
97 32
left
```

Vrsta događaja i taster su obični celi brojevi sa imenom. Nikad ih ne poredimo međusobno.

key je taster, a unicode je znak koji piše

pritisnemo

dogadjaj.key

dogadjaj.unicode

taster a

pg.K_a (97)

"a"

Shift + a

pg.K_a (97)

"A"

razmak

pg.K_SPACE (32)

" "

strelica levo

pg.K_LEFT

"" (prazno)

- Za upravljanje (skok, pauza, ESC) gledamo key.
- Za tekst i slova (igra slova, upis imena) gledamo unicode.

Koliko događaja KEYDOWN stigne?

Igra radi 60 frejmova u sekundi.

Igrač pritisne strelicu desno i drži je tačno jednu sekundu, pa je pusti.

Koliko događaja KEYDOWN za tu strelicu dobije program?

A

1

tačno

B

60

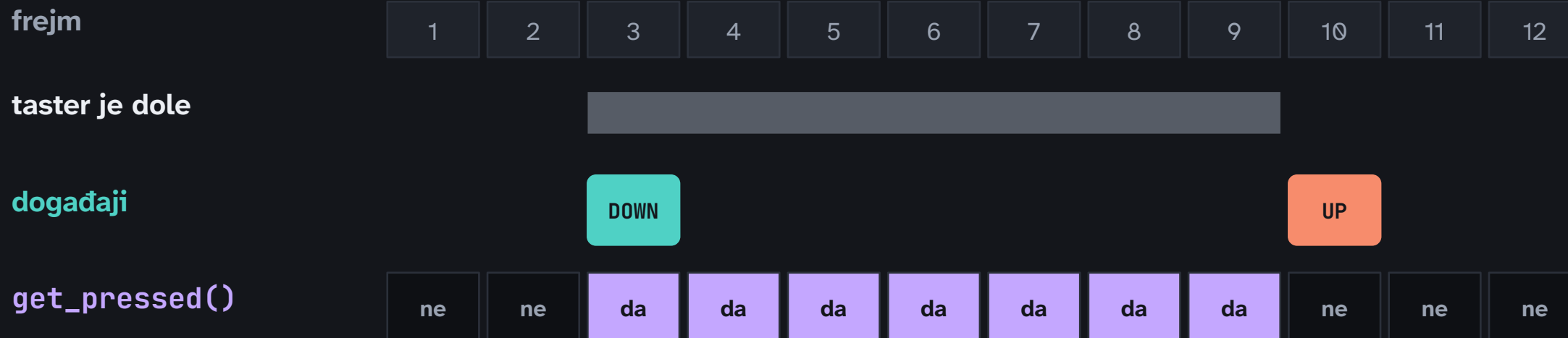
C

220

D

0

Jedan pritisak ili držanje: dva pitanja



Jednom, u trenutku pritiska: **KEYDOWN**

skok, pucanj, pauza, izbor u meniju, slovo u igri slova

U svakom frejmu dok držimo: **get_pressed()**

hodanje, trčanje, nišanje, gas u autu

Čudovište: lik se pomeri samo jednom

```
for dogadjaj in pg.event.get():  
    ...  
    if dogadjaj.type == pg.KEYDOWN:  
        if dogadjaj.key == pg.K_RIGHT:  
            x += brzina * dt    # samo u frejmu pritiska
```

Kako izgleda

Lik mrdne za par piksela pa stane, iako držimo strelicu.

Zašto se pojavljuje

Kretanje je u KEYDOWN, a taj događaj stiže samo jednom.

Kako se pobeđuje

Hodanje ide u get_pressed(), posle petlje for, u svakom frejmu.

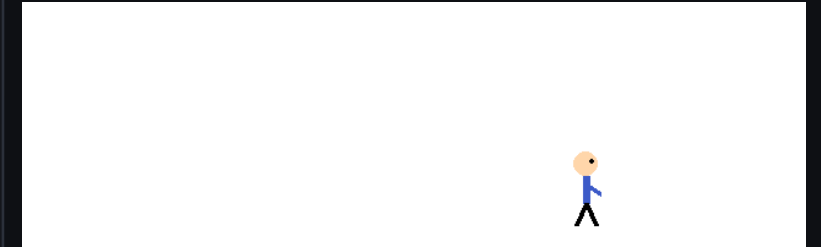


jedan pritisak strelice: x = 4

get_pressed: lik ide dok držimo strelicu

```
tasteri = pg.key.get_pressed() # posle petlje for
if tasteri[pg.K_LEFT]:
    x -= brzina * dt
    smer = -1
if tasteri[pg.K_RIGHT]:
    x += brzina * dt
    smer = 1
x = pg.math.clamp(x, 0, SIRINA - SIRINA_SLIKE)
```

Isto kao u temi 07: brzina je u pikselima u sekundi, pomnožena sa dt.



- `tasteri[pg.K_LEFT]` je True dok je taster dole.
- Dva if, a ne if i elif: obe strelice se proveravaju.
- `clamp` drži Čikicu u prozoru.

Šta ispisuje ovaj program?

```
brzina = 200
dt = 0.5
x = 100
levo = True           # obe strelice su
desno = True          # pritisnute
if levo:
    x -= brzina * dt
elif desno:
    x += brzina * dt
print(x)
```

Obe strelice su pritisnute u istom frejmu.

A

100.0

B

tačno

0.0

C

200.0

D

0

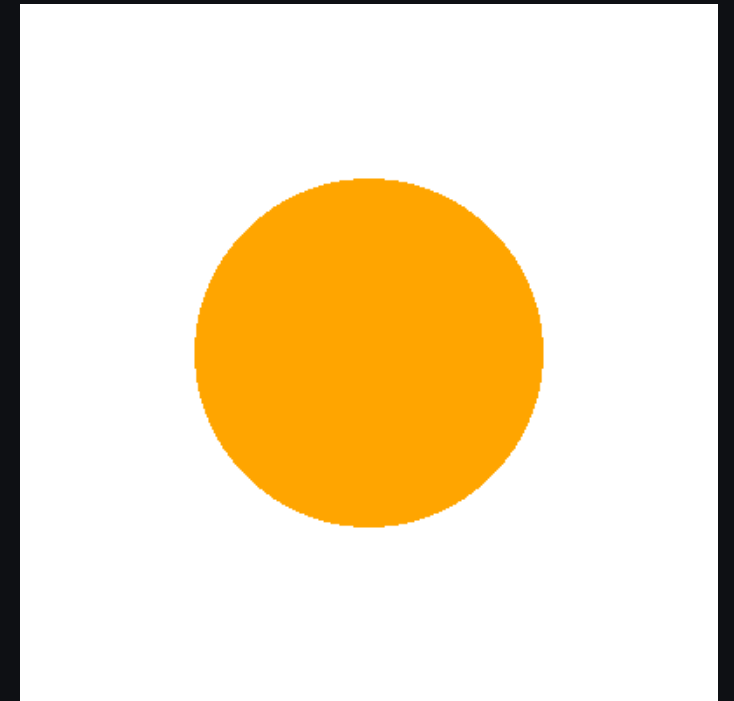
Miš

Klik, mesto klika i taster koji drži

Krug je pun dok držimo taster miša

```
dogadjaj = pg.event.wait()          # čekamo događaj
if dogadjaj.type == pg.QUIT:
    radi = False
elif dogadjaj.type == pg.MOUSEBUTTONDOWN:
    obojen = True
    treba_crtati = True
elif dogadjaj.type == pg.MOUSEBUTTONUP:
    obojen = False
    treba_crtati = True
```

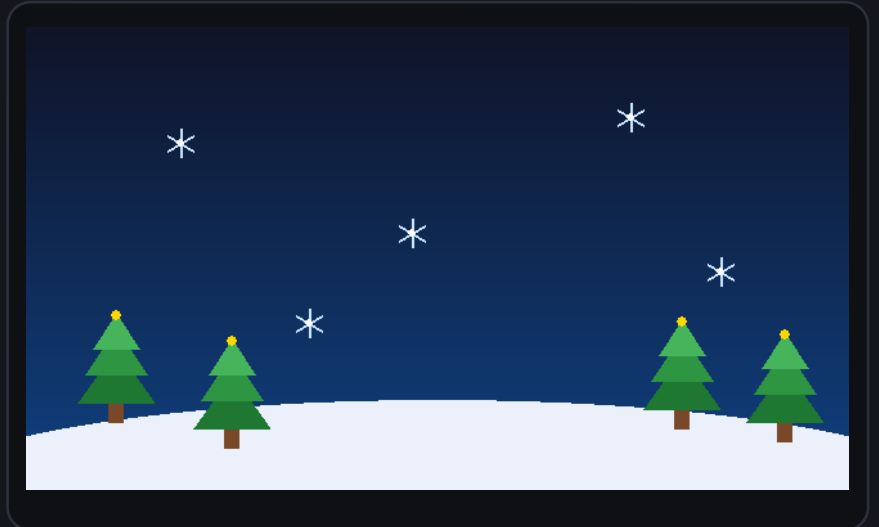
```
debljina = 0 if obojen else 2        # 0 je pun krug
pg.draw.circle(ekran, "orange", (200, 200), 100, debljina)
```



Zadatak iz starih beleški. Ništa se ne pomera samo od sebe, pa petlja čeka događaj.

event.pos je mesto gde smo kliknuli

```
for dogadjaj in pg.event.get():  
    if dogadjaj.type == pg.QUIT:  
        radi = False  
    elif dogadjaj.type == pg.MOUSEBUTTONDOWN:  
        if dogadjaj.button == 1:          # levi taster  
            mesta.append(dogadjaj.pos)    # mesto klika  
ekran.blit(POZADINA, (0, 0))  
for mesto in mesta:  
    ekran.blit(SLIKA, SLIKA.get_rect(center=mesto))
```



- button: 1 levi, 2 srednji, 3 desni taster.
- I točkić šalje MOUSEBUTTONDOWN (4 i 5), zato proveravamo button.
- get_rect(center=mesto) centrirá pahuljicu na klik.

Miš ima i stanje, kao tastatura

```
import pygame as pg

pg.init()
pg.display.set_mode((200, 200))
print(pg.mouse.get_pressed())
```

ISPIS

```
(False, False, False)
```

`pg.mouse.get_pressed()`

tri tastera miša u ovom frejmu: levi, srednji, desni

`pg.mouse.get_pos()`

gde je miš sada, kao par (x, y)

Pazite

posle klika koristite event.pos: `get_pos()` je mesto miša sada, a ne u trenutku klika

Tajmeri

Događaji koje izmislimo sami

Ne čekamo sa wait: tajmer šalje događaj

Ovako ne: igra se zamrzne na 0,7 s

```
pg.time.wait(700)          # prozor se „zamrzne“  
pahuljice.append(Pahuljica())
```

Ovako da: petlja radi, a događaj stiže sam

```
NOVA_PAHULJICA = pg.event.custom_type()    # pre petlje  
pg.time.set_timer(NOVA_PAHULJICA, 700)     # svakih 0,7 s  
...  
for dogadjaj in pg.event.get():  
    if dogadjaj.type == NOVA_PAHULJICA:  
        pahuljice.append(Pahuljica())
```

`custom_type()`

rezerviše novu vrstu događaja i vraća njen broj

`set_timer(vrsta, ms)`

šalje taj događaj na svakih ms milisekundi

`set_timer(vrsta, 0)`

isključuje tajmer

Tajmer na 0,1 s: za pola sekunde pet događaja

```
import pygame as pg

pg.init()
pg.display.set_mode((200, 200))
NOVA_PAHULJICA = pg.event.custom_type()
pg.time.set_timer(NOVA_PAHULJICA, 100) # svakih 0,1 s
sat = pg.time.Clock()
vreme = 0
broj = 0
while vreme < 0.55:
    vreme += sat.tick(60) / 1000
    for dogadjaj in pg.event.get():
        if dogadjaj.type == NOVA_PAHULJICA:
            broj += 1
print("pahuljica:", broj)
```

ISPIS

pahuljica: 5

- Petlja ne zna kad će događaj stići: samo proverava red u svakom frejmu.
- Na sporom računaru broj može da bude 4 ili 6.

Koliko novih pahuljica za 5 sekundi?

```
NOVA = pg.event.custom_type()
pg.time.set_timer(NOVA, 700)
# ... igra radi tačno 5 sekundi;
# za svaki događaj NOVA
# pravimo jednu pahuljicu
```

Prva pahuljica stiže posle 0,7 s, a ne odmah na početku.

A

5

B

tačno

7

C

3.5

D

700

I biblioteka šalje događaje: kraj pesme

```
KRAJ_PESME = pg.event.custom_type()
pg.mixer.music.load("ton.wav")
pg.mixer.music.set_endevent(KRAJ_PESME)
pg.mixer.music.play()
...
if dogadjaj.type == KRAJ_PESME:
    print("pesma je završena")
```

Zadatak iz starih beleški. O zvuku i muzici više u temi 11.

set_endevent(vrsta)

kad se pesma završi, u red stiže događaj te vrste

isti obrazac

kao tajmer: proverimo vrstu u petlji for

ton.wav

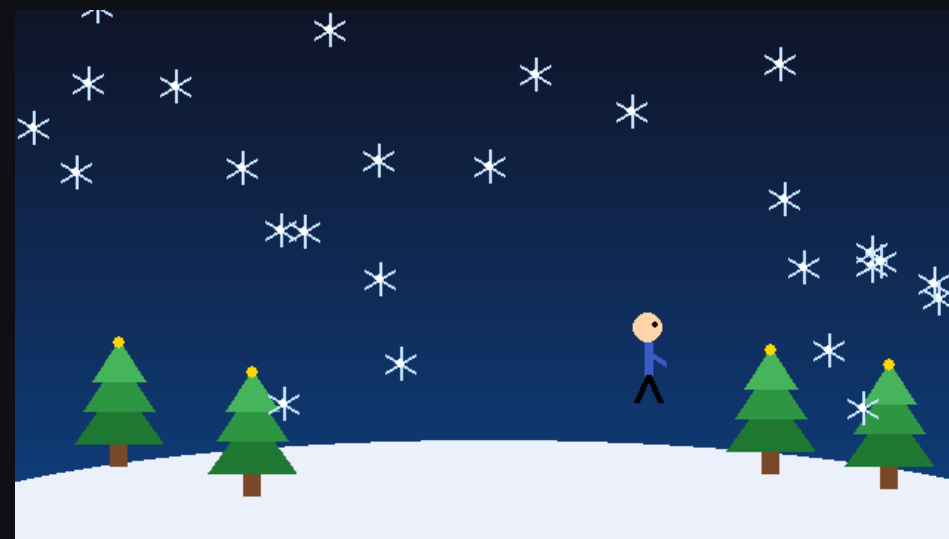
pravi ga sam primer, modulom wave

Zimska šetnja, korak 3: Čikica hoda

Cilj: Čikica hoda strelicama, skače na razmak, a sneg dobija nove pahuljice.

- 1 Klasa Igrac: x, y, brzina = 220, smer
- 2 Hod: get_pressed + dt, pa clamp
- 3 Razmak (KEYDOWN): skoci()
- 4 ESC (KEYDOWN): kraj igre
- 5 Tajmer na 0,7 s: nova pahuljica

KAKO IZGLEDA NA KRAJU



Sličice pravi program 08_01, isti kao u temi 07.

Igrač pamti položaj, brzinu i smer

```
class Igrac:
    def __init__(self, x):
        self.x = x
        self.y = TL0                # gornja ivica slike
        self.brzina = 220            # px u sekundi
        self.smer = 1                # 1 desno, -1 levo
        self.brzina_y = 0            # za skok, px u sekundi

    def skoci(self):
        if self.y == TL0:            # samo sa zemlje
            self.brzina_y = -450     # minus je nagore
```

TL0

y kad Čikica stoji na snegu

self.brzina_y

brzina gore-dole; za hod je ne koristimo

skoci()

skok samo sa zemlje: nema skoka iz vazduha

azuriraj dobija dt i tastere

```
def azuriraj(self, dt, tasteri):  
    if tasteri[pg.K_LEFT]:  
        self.x -= self.brzina * dt  
        self.smer = -1  
    if tasteri[pg.K_RIGHT]:  
        self.x += self.brzina * dt  
        self.smer = 1  
    self.x = pg.math.clamp(self.x, 0, SIRINA - SIRINA_SLIKE)  
    self.brzina_y += GRAVITACIJA * dt      # skok  
    self.y += self.brzina_y * dt  
    if self.y >= TL0:                      # dočekao se  
        self.y = TL0  
        self.brzina_y = 0
```

tasteri

ono što vrati get_pressed() u ovom frejmu

GRAVITACIJA

svake sekunde brzina nadole raste za 1500 px/s

self.y >= TL0

dočekao se na sneg: stoji

Skok: gravitacija polako koči lik

```
GRAVITACIJA = 1500      # px u sekundi, svake sekunde  
brzina_y = -450  
y = 0  
for frejm in range(4):  
    brzina_y += GRAVITACIJA * 0.1  
    y += brzina_y * 0.1  
    print(round(brzina_y), round(y))
```

ISPIS: brzina_y, y

```
-300 -30  
-150 -45  
0 -45  
150 -30
```

- y ide nagore (−30, −45), stane, pa krene nadole.
- Minus je gore: u prozoru y raste nadole.
- Ovo je ceo skok: dva reda u azuriraj.

U petlji: jednom za pritisak, stalno za hod

```
while radi:
    for dogadjaj in pg.event.get():
        if dogadjaj.type == pg.QUIT:
            radi = False
        elif dogadjaj.type == pg.KEYDOWN:
            if dogadjaj.key == pg.K_ESCAPE:
                radi = False
            elif dogadjaj.key == pg.K_SPACE:
                igrac.skoci()
            elif dogadjaj.type == NOVA_PAHULJICA:
                if len(pahuljice) < 80:
                    pahuljice.append(Pahuljica())
    dt = sat.tick(60) / 1000
    vreme += dt
    igrac.azuriraj(dt, pg.key.get_pressed())
```

KEYDOWN

ESC i razmak: jednom, u trenutku pritiska

NOVA_PAHULJICA

naš događaj iz tajmera; najviše 80 pahuljica

get_pressed()

hod: posle petlje for, u svakom frejmu

Pronađi grešku: Čikica nikad ne skoči

```
for dogadjaj in pg.event.get():  
    if dogadjaj.type == pg.QUIT:  
        radi = False  
    elif dogadjaj.type == pg.K_SPACE:  
        igrac.skoci()
```

Program se ne ruši i ne javlja grešku. Razmak ne radi ništa.

```
ISPIS: print(pg.KEYDOWN, pg.K_SPACE)
```

```
768 32
```

type

vrsta: 768 je KEYDOWN

K_SPACE

taster: 32, nikad jednak vrsti

ispravka

type == KEYDOWN i key == K_SPACE

Izaberite nivo i počnite

LEVEL 1

Četiri smeri

Čikica ide i gore-dole; rade i strelice i W, A, S, D. Ne izlazi iz prozora.

Pomoć: dva nova if i clamp za y.

LEVEL 2

Sneg na pauzi

Taster P zaustavlja i pokreće sneg, a klik pravi pahuljicu na mestu klika.

Pomoć: pauza = not pauza u KEYDOWN.

LEVEL 3

Igra slova

Slova reči su rasuta po prozoru; pritisak slova ga uklanja. Na kraju „Bravo!“ i kraj posle 2 s.

Pomoć: event.unicode i tajmer.

Sledeći put Čikica postaje sprajt

```
class Igrac(pg.sprite.Sprite):  
    def __init__(self, x):  
        super().__init__()  
        self.image = desno[0]  
        self.rect = self.image.get_rect()
```

U temi 09 Igrac i Pahuljica nasleđuju klasu pg.sprite.Sprite, pahuljice idu u grupu, a Čikica dok hoda mrda nogama: slike se smenjuju na svakih 0,12 sekundi.

Rezime

red događaja

```
for dogadjaj in pg.event.get():
```

Sve što se desilo, istim redom.

jedan pritisak

```
dogadjaj.type == pg.KEYDOWN  
dogadjaj.key == pg.K_SPACE
```

Skok, pauza, ESC, slova (unicode).

držanje tastera

```
tasteri = pg.key.get_pressed()  
tasteri[pg.K_LEFT]
```

Hodanje u svakom frejmu, sa dt.

tajmer

```
N = pg.event.custom_type()  
pg.time.set_timer(N, 700)
```

Događaj na svakih 0,7 s, bez čekanja.

Pitanje pre odlaska

```
if dogadjaj.type == pg.KEYDOWN:  
    ...  
    tasteri = pg.key.get_pressed()
```

Pucanj: jedan metak za jedan pritisak razmaka, čak i kad igrač drži taster. Koji od ova dva načina koristite?

Pošaljite jednu reč pre nego što izađete.



Literatura

- Dokumentacija za pygame-ce, moduli event, key, mouse i time: 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 „Event Queue“ (besplatno na gameprogrammingpatterns.com)

Dodatna literatura i zanimljivi sajtovi

pygame-ce: modul event

pyga.me/docs/ref/event.html

Red događaja, get, post i custom_type za sopstvene događaje.

pygame-ce: modul key

pyga.me/docs/ref/key.html

get_pressed, key.name i tabela svih konstanti tastera.

pygame-ce: modul mouse

pyga.me/docs/ref/mouse.html

Položaj i tasteri miša: get_pos i get_pressed.

pygame-ce: modul time

pyga.me/docs/ref/time.html

set_timer šalje događaj na svakih nekoliko milisekundi; loops kaže koliko puta.

Game Programming Patterns: Event Queue

gameprogrammingpatterns.com/event-queue.html

Poglavlje besplatne knjige Roberta Nistroma o redu događaja.

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