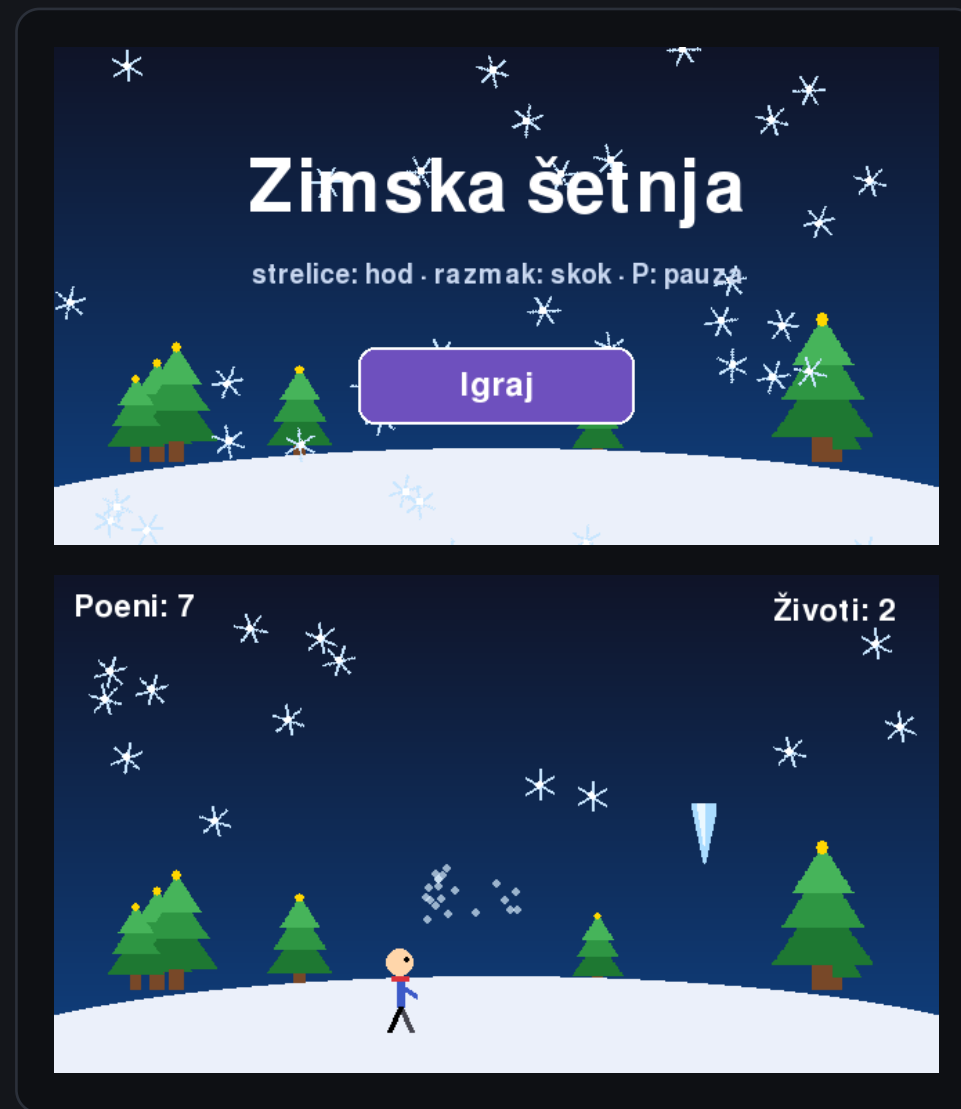


Stanja igre i završni projekat

Kako igra prelazi iz menija u igru, u pauzu i na kraj, a veliki program se deli na fajlove

Danas pravimo: Zimska šetnja, korak 7, cela igra



Tri pitanja za zagrevanje

1. prošla tema: rotate

```
import pygame as pg

s = pg.Surface((10, 10))
a = pg.transform.rotate(s, 90)
b = pg.transform.rotate(s, 45)
print(a.get_size())
print(b.get_size())
```

```
(10, 10)
(14, 14)
```

2. sprajтови i grupe

```
import pygame as pg

grupa = pg.sprite.Group()
a = pg.sprite.Sprite()
b = pg.sprite.Sprite()
grupa.add(a, b)
a.kill()
print(len(grupa))
print(a.alive())
```

```
1
False
```

3. mešano: nasleđivanje

```
class Scena:
    def ime(self):
        return "scena"

class Meni(Scena):
    def ime(self):
        return "meni"

for s in [Scena(), Meni()]:
    print(s.ime())
```

```
scena
meni
```

Šest novih reči za danas

stanje (scena)

deo igre sa svojim ponašanjem: meni, igra, pauza, kraj

prelaz

trenutak kad igra iz jedne scene pređe u drugu

dugme

pravougaonik na ekranu koji reaguje na klik mišem

modul

jedan .py fajl čije klase i funkcije uvozimo sa import

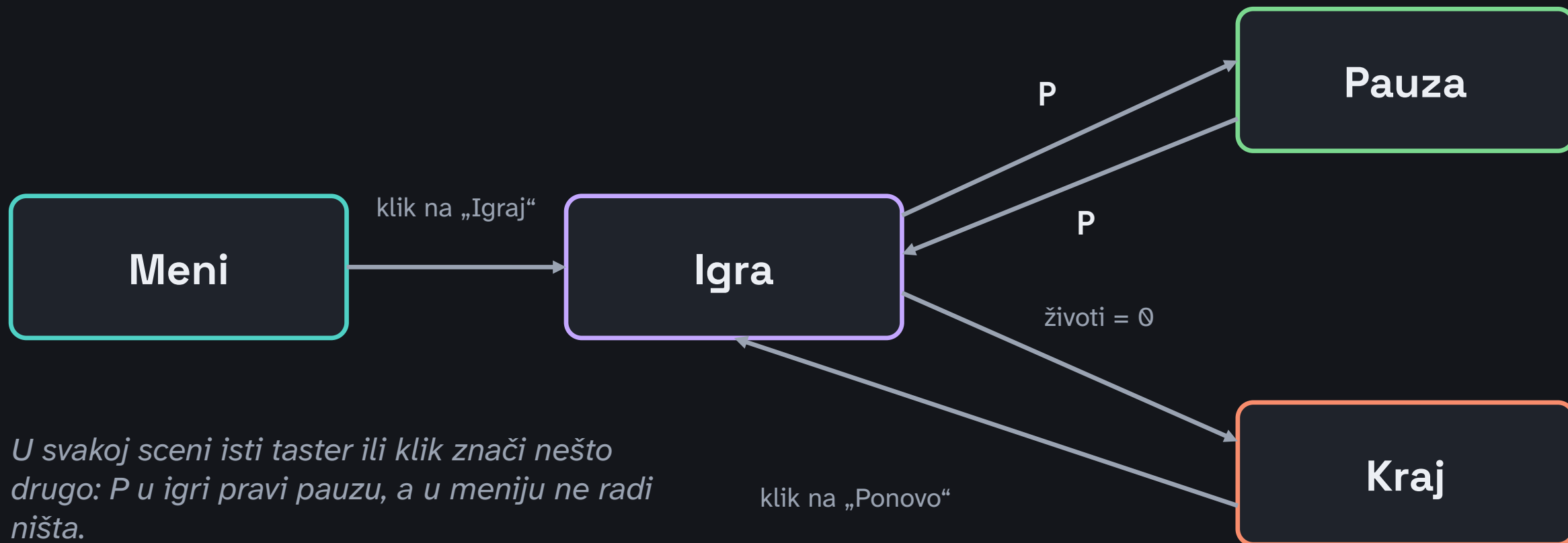
resursi

slike, zvuci i muzika koje igra učitava iz fajlova

seminarski rad

vaša igra: završni projekat predmeta

Igra se sastoji od scena i prelaza među njima



Tri logičke promenljive brzo postanu zbrka

```
na_meniju = True
pauza = False
kraj_igre = False
...
if dogadjaj.type == pg.KEYDOWN:
    if na_meniju and dogadjaj.key == pg.K_RETURN:
        na_meniju = False
    elif not na_meniju and not kraj_igre and dogadjaj.key == pg.K_p:
        pauza = not pauza
...
if not na_meniju and not pauza and not kraj_igre:
    vreme += dt          # igra teče samo ovde
```

Tri promenljive daju $2 \times 2 \times 2 = 8$ kombinacija, a smislene su samo četiri. Svaki uslov mora da proveri sve tri, inače P radi i u meniju.

Jedna promenljiva stanje je bolja, ali ne dovoljno

```
stanje = "meni"          # "meni", "igra", "pauza" ili "kraj"
...
if stanje == "meni" and dogadjaj.key == pg.K_RETURN:
    stanje = "igra"
elif stanje == "igra" and dogadjaj.key == pg.K_p:
    stanje = "pauza"
elif stanje == "pauza" and dogadjaj.key == pg.K_p:
    stanje = "igra"
...
if stanje == "igra":
    vreme += dt
```

- Igra je uvek u tačno jednom stanju.
- Ali if-elif za stanje stoji na tri mesta: događaji, ažuriranje i crtanje.
- Nova scena znači menjanje sva tri mesta.

Scene kao klase

Svaka scena zna šta radi, a petlja samo poziva tri metode

Svaka scena ima iste tri metode

```
class Scena:  
    def __init__(self):  
        self.sledeca = None # None: ostajemo ovde  
  
    def obradi(self, dogadjaj):  
        pass  
  
    def azuriraj(self, dt):  
        pass  
  
    def crtaj(self, ekran):  
        pass
```

obradi(dogadjaj)

šta scena radi na taster ili klik

azuriraj(dt)

šta se pomera u ovom frejmu

crtaj(ekran)

šta se vidi na ekranu

self.sledeca

kuda dalje; None znači ostajemo

Glavna petlja ne zna u kojoj je sceni

```
scena = Meni()
radi = True
while radi:
    for dogadjaj in pg.event.get():
        if dogadjaj.type == pg.QUIT:
            radi = False
        else:
            scena.obradi(dogadjaj)
    dt = sat.tick(60) / 1000
    scena.azuriraj(dt)
    if scena.sledeca is not None:
        nova = scena.sledeca
        scena.sledeca = None
        scena = nova
    scena.crtaj(ekran)
pg.display.flip()
```

`scena.obradi(...)`

Meni, Igra, Pauza ili Kraj: svaka na svoj način

`scena.sledeca`

scena je tražila prelaz, pa menjamo scenu

`scena.sledeca = None`

brišemo zahtev, da se prelaz ne ponovi

Šta ispisuje ovaj program?

```
scena = Meni()
for taster in ["p", "enter", "p", "x", "p"]:
    scena.obradi(taster)
    if scena.sledeca is not None:
        nova = scena.sledeca
        scena.sledeca = None
        scena = nova
    print(type(scena).__name__, end=" ")
```

Meni: enter → Igra

Igra: p → Pauza(self)

Pauza: p → nazad u istu igru

A

Meni Meni
Igra Pauza
Pauza

B

tačno

Meni Igra
Pauza Pauza
Igra

C

Meni Igra
Pauza Igra
Pauza

D

Pauza Igra
Pauza Pauza
Igra

Dugme je pravougaonik koji pita: da li je klik unutra?

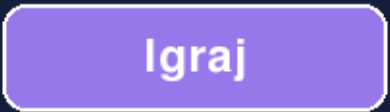
```
class Dugme:
    def __init__(self, tekst, centar):
        self.tekst = tekst
        self.rect = pg.Rect(0, 0, 200, 56)
        self.rect.center = centar

    def je_kliknuto(self, dogadjaj):
        return (dogadjaj.type == pg.MOUSEBUTTONDOWN and dogadjaj.button == 1
                and self.rect.collidepoint(dogadjaj.pos))
```



Igraj

obično



Igraj

miš je iznad

- MOUSEBUTTONDOWN: taster miša je pritisnut
- button == 1: levi taster
- collidepoint(pos): tačka klika je u pravougaoniku

Šta ispisuje ovaj program?

```
import pygame as pg

r = pg.Rect(100, 50, 200, 60)
print(r.right, r.collidepoint(r.right, 80))
```

Dugme počinje na $x = 100$ i široko je 200 piksela.

A

300 True

B

tačno

300 False

C

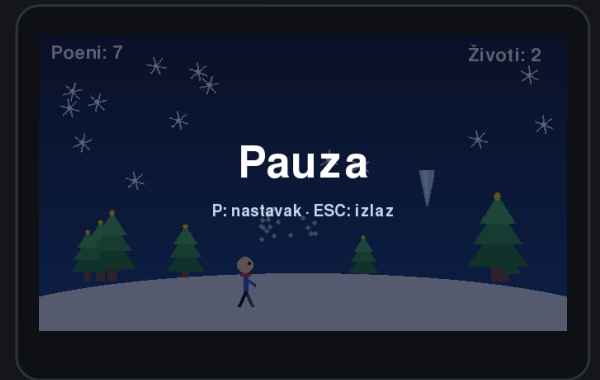
299 True

D

299 False

Pauza pamti igru i ne pomera ništa

```
class Pauza(Scena):  
    def __init__(self, igra):  
        super().__init__()   
        self.igra = igra          # pamtimo igru  
  
    def obradi(self, dogadjaj):  
        if dogadjaj.type == pg.KEYDOWN and dogadjaj.key == pg.K_p:  
            self.sledeca = self.igra  
  
        # azuriraj ne redefinišemo: sve stoji  
  
    def crtaj(self, ekran):  
        self.igra.crtaj(ekran)  
        ekran.blit(self.zatamni, (0, 0))
```



- Ista igra: isti poeni i položaji.
- Nema azuriraj: nasleđena metoda ne radi ništa.
- Crta igru, pa preko nje zatamnjenje.

Natpis na kraju: petlja radi i dok čekamo

Prozor ne reaguje dve sekunde:

```
napisi(ekran, "Bravo!", CENTAR)
pg.display.flip()
pg.time.delay(2000) # 2 s ništa ne radi
```

Za to vreme program ne čita događaje, pa ni klik na X ne radi.

Prozor reaguje sve vreme:

```
preostalo = 2.0 # sekundi
radi = True
while radi:
    for dogadjaj in pg.event.get():
        if dogadjaj.type == pg.QUIT:
            radi = False
    dt = sat.tick(60) / 1000
    preostalo -= dt
    if preostalo <= 0:
        radi = False
    ekran.fill((20, 30, 60))
    napisi(ekran, "Bravo!", CENTAR)
    pg.display.flip()
```

Čudovište: stari poeni u novoj igri

```
class Igra:
    def __init__(self):
        self.poeni = 0

class Kraj:
    def __init__(self, igra):
        self.igra = igra

    def ponovo(self):
        return self.igra # stara igra!

igra = Igra()
igra.poeni = 12
nova = Kraj(igra).ponovo()
print("Nova igra:", nova.poeni, "poena")
```

ISPIS

Nova igra: 12 poena

Kako izgleda

Posle „Ponovo“ igra počinje sa poenima i životima iz prošle partije.

Zašto se pojavljuje

Nova partija koristi stari objekat ili stare promenljive.

Kako se pobeđuje

Za novu partiju nov objekat: `self.sledeca = Igra()`.

Zimska šetnja, korak 7

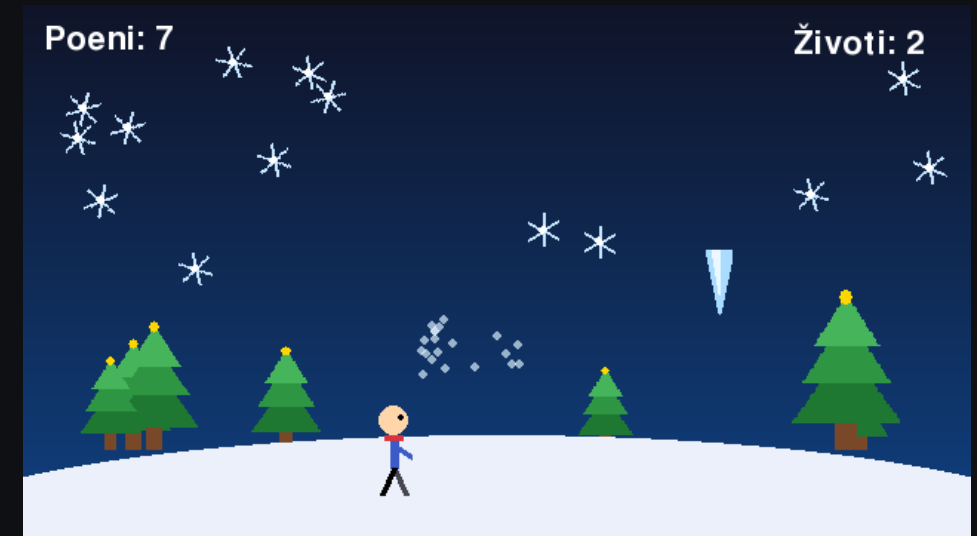
Cela igra: četiri scene i četiri fajla

Zimska šetnja, korak 7: scene i moduli

Cilj: igra ima meni, pauzu i kraj sa rezultatom, a kod je u četiri fajla.

- 1 Klase iz koraka 1–6 selimo u `sprajтови.py`
- 2 Klasa `Scena` i klasa `Dugme` u `scene.py`
- 3 Igra: sve što je bilo u petlji ide u scenu
- 4 Meni sa dugmetom „Igraj“, Pauza na P
- 5 Kraj sa rezultatom i dugmetom „Ponovo“
- 6 `main.py`: prozor, muzika i glavna petlja

KAKO IZGLEDA NA KRAJU



Slike i zvuke pravi `pravimo_resurse.py` pri prvom pokretanju

Igra u četiri fajla: svaki ima jedan posao

```
12_11_zimska_setnja/  
├── main.py  
│   prozor, muzika, petlja  
├── scene.py  
│   Dugme, Meni, Igra, Pauza, Kraj  
├── sprajtovi.py  
│   Igrac, Pahuljica, Prepreka  
├── pravimo_resurse.py  
│   crta slike, pravi zvuke  
└── resursi/  
    pozadina.png, dzing.wav ...
```

Ko koga uvozi:

```
# scene.py  
from sprajtovi import SIRINA, VISINA, SLIKE, ZVUCI  
from sprajtovi import Igrac, Pahuljica  
from sprajtovi import Prepreka, Cestica
```

```
# main.py  
import sprajtovi  
from scene import Meni  
sprajtovi.ucitaj_resurse()
```

Scena Igra: tajmer, skok i pauza

```
def obradi(self, dogadjaj):  
    if dogadjaj.type == NOVA_PAHULJICA:  
        self.pahuljice.add(Pahuljica())  
    elif dogadjaj.type == pg.KEYDOWN:  
        if dogadjaj.key == pg.K_SPACE:  
            self.igrac.skoci()  
        elif dogadjaj.key == pg.K_p:  
            self.sledeca = Pauza(self)
```

Strelice ne obrađujemo ovde: Igrac ih sam čita u update preko get_pressed, jer se drže.

NOVA_PAHULJICA

sopstveni događaj; tajmer ga šalje svakih 0,7 s

K_SPACE

skok: jedan pritisak, zato KEYDOWN

Pauza(self)

pauza dobija ovu igru, da bi se vratila u nju

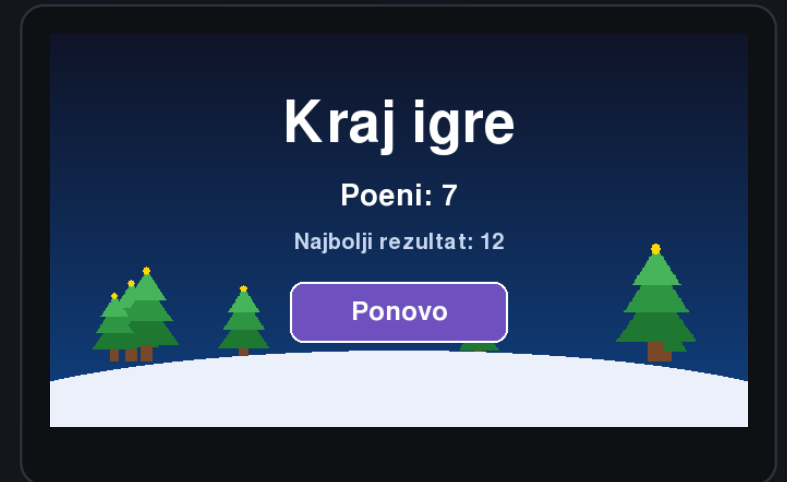
Uhvaćena pahuljica: poen, zvuk i 20 čestica

```
uhvacene = pg.sprite.spritecollide(  
    self.igrac, self.pahuljice, True, pg.sprite.collide_circle)  
for p in uhvacene:  
    self.poeni += 1  
    ZVUCI["dzing"].play()  
    for i in range(20):  
        self.cestice.add(Cestica(p.rect.center))  
  
...  
if self.igrac.zivoti == 0:  
    self.sledeca = Kraj(self.poeni)
```

- dokill=True: uhvaćena pahuljica nestaje iz grupe.
- collide_circle: krug, bez lažnih sudara u uglovima.
- Kraj dobija samo broj poena.

Kraj pravi novu igru, a pamti najbolji rezultat

```
class Kraj(Scena):  
    najbolji = 0 # atribut klase  
  
    def __init__(self, poeni):  
        super().__init__()  
        self.poeni = poeni  
        if poeni > Kraj.najbolji:  
            Kraj.najbolji = poeni  
        self.dugme = Dugme("Ponovo", (SIRINA / 2, 255))  
  
    def obradi(self, dogadjaj):  
        if self.dugme.je_kliknuto(dogadjaj):  
            self.sledeca = Igra() # NOVA igra
```



- najbolji je atribut klase: ostaje i kad nastane novi Kraj.
- Igra() je nov objekat: poeni 0, životi 3.

Igra je rasla kroz sedam tema

tema 06

korak 1: scena, jelkice,
naslov

tema 07

korak 2: sneg pada, dt

tema 08

korak 3: hod
strelicama, skok, tajmer

tema 09

korak 4: sprajtovi,
animacija hoda

tema 10

korak 5: sudari, poeni,
životi

tema 11

korak 6: zvuk,
okretanje, čestice

tema 12

korak 7: scene, dugme,
moduli

Sve što ste naučili od teme 01 do danas je u ovoj igri: klase, nasleđivanje, specijalne metode u Vector2, petlja, događaji.

Pronađi grešku: dugme ne reaguje na klik

```
class Dugme:
    def __init__(self, x, y):
        self.rect = pg.Rect(x, y, 200, 56)

    def je_kliknuto(self, dogadjaj):
        return (dogadjaj.type == pg.MOUSEMOTION
                and self.rect.collidepoint(dogadjaj.pos))

dugme = Dugme(220, 200)
klik = pg.event.Event(pg.MOUSEBUTTONDOWN, pos=(300, 220), button=1)
print(dugme.je_kliknuto(klik))
```

ISPIS

False

Ispravka

MOUSEMOTION je pomeranje miša. Klik je MOUSEBUTTONDOWN.

Program se ne ruši, ali klik tačno na dugme ne radi ništa.

Završni projekat

Vaša igra za seminarski rad

Šta seminarski rad treba da pokaže

PREDLOG; konačna pravila daje predavač

- 1 igra u pygame-ce koja se pokreće bez greške
- 2 bar tri scene, na primer meni, igra i kraj
- 3 sopstvene klase, bar jedna izvedena
- 4 kretanje sa dt: brzina u pikselima u sekundi

- 1 unos tastaturom ili mišem
- 2 sudari i pravilo: poeni, životi ili vreme
- 3 kod u bar dva modula
- 4 resursi koje ste napravili ili smete da koristite

Od ideje do igre u pet koraka

- 1 Mala ideja** jedno pravilo: hvataj, izbegavaj, stigni do cilja
- 2 Skica scena** na papiru: koje scene i šta vodi iz jedne u drugu
- 3 Kostur** prozor, petlja, prazne scene sa prelazima
- 4 Korak po korak** kao Zimska šetnja: posle svake izmene pokrenite
- 5 Ukas na kraju** zvuk, čestice, animacija, kad pravila rade

Šta se predaje i kako se proverava licenca

U folderu rada

- svi .py fajlovi; main.py pokreće igru
- folder resursi/ sa svim slikama i zvucima
- kratak opis: kako se igra i kako se pokreće
- izvori: odakle je svaka tuđa slika ili zvuk

Tuđi resursi

- Kenney: CC0, slobodno i bez navođenja autora
- OpenGameArt: licenca piše uz svaki rad
- Freesound: Creative Commons, uz neke se navodi autor
- Najsigurnije: nacrtajte kodom, kao mi

Izaberite nivo i počnite

LEVEL 1

Dugme Izlaz

U meniju igre „Klikni pahuljicu“ dodajte dugme „Izlaz“ koje zatvara igru.

Pomoć: pošaljite događaj QUIT sa `pg.event.post`.

LEVEL 2

Scena Pomoć

Dugme „Pomoć“ u meniju otvara uputstvo; bilo koji taster vraća u meni.

Pomoć: nova klasa nasleđuje Scena.

LEVEL 3

Odbrojavanje

Pre svake partije na sredini ekrana piše 3, 2, 1, pa igra počinje.

Pomoć: preostalo `-= dt`, kao natpis na kraju.

Sledeći put: vežbanje i priprema za ispit

Tema 13

Turnir pitanja po timovima kroz svih dvanaest tema, mapa pojmova i kako se za dva sata napiše ceo mali program.

Odbrana

U poslednjem bloku branite seminarski rad: pokrenete igru, objasnite kod i odgovorite na pitanja.

Rezime

scena

```
obradi(dogadjaj)
azuriraj(dt)
crtaj(ekran)
```

Svaka scena je klasa sa iste tri metode.

prelaz

```
self.sledeca = Igra()
```

Petlja menja scenu i briše zahtev.

dugme

```
self.rect.collidepoint(
    dogadjaj.pos)
```

Klik je MOUSEBUTTONDOWN.

moduli

```
from scene import Meni
```

main, scene, sprajтови, resursi.

Pitanje pre odlaska

```
scena = Meni()  
for taster in ["enter", "p", "enter", "p"]:  
    ...                # obradi i prelaz
```

Prelazi su isti kao u igri Zimska šetnja (enter umesto klika na „Igraj“). U kojoj sceni smo na kraju?

Pošaljite jednu reč pre nego što izađete.



Literatura

- Dokumentacija za pygame-ce, Rect i event: pyga.me/docs
- Nystrom, R. Game Programming Patterns, poglavlje „State“ (besplatno na gameprogrammingpatterns.com)
- Zvanični Python tutorijal, poglavlje 6 „Modules“: docs.python.org/3/tutorial/modules.html
- Sweigart, A. Invent Your Own Computer Games with Python, 4. izdanje, No Starch Press (besplatno na inventwithpython.com)

Dodatna literatura i zanimljivi sajtovi

Game Programming Patterns: State

gameprogrammingpatterns.com/state.html

Poglavlje o stanjima: junakinja koja stoji, skače i čuči, sa stanjima kao klasama.

pygame-ce: Rect

pyga.me/docs/ref/rect.html

Sve o pravougaoniku; tu piše i da tačka na desnoj ili donjoj ivici nije unutra.

Python tutorijal: moduli

docs.python.org/3/tutorial/modules.html

Šta je modul i kako rade import i from ... import.

Kenney

kenney.nl/assets

Grafike i zvukovi za igre pod licencom CC0, slobodni i za seminarski rad.

OpenGameArt

opengameart.org

Grafike, muzika i zvučni efekti; licenca piše uz svaki rad.

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