Kurs: Pajton - programiranje igara sa Pygame bibliotekom
013_014 čas: dogadjaji_tastaturom
Teme: pygame biblioteka, pygame dogadjaji, petlja dogadjaja u realnom vremenu, dogadjaji tastaturom, prozivanje tastature, program Brzi_tasteri.py, program Resetovanje.py

001 Pygame biblioteka

Pygame je zapravo zasnovan na totalno drugačijoj biblioteci: SDL.
Simple DirectMedia Layer je open source biblioteka koja radi sa 2D grafikom i korisničkim ulazom i veoma lako se može podržati na različitim platformama.
Pošto je Pygame baziran na SDL, najveći broj SDL osobenosti je podržano u Pygame.

002 Pygame događaji

Pygame događaji se bave različitim stvarima u Pygame programima.
Ovo je kompletna lista svih događaja:
QUIT, ACTIVEEVENT, KEYDOWN, KEYUP, MOUSEMOTION, MOUSEBUTTONUP, MOUSEBUTTONDOWN, JOYAXISMOTION, JOYBALLMOTION, JOYHATMOTION, JOYBUTTONUP, JOYBUTTONDOWN, VIDEORESIZE, VIDEOEXPOSE, USEREVENT
Iz spiska se vidi da Pygame podržava upotrebu džojstika kao ulaza.
Džojstik mora biti uključen i konfigurisan sa operativnim sistemom da bi radio sa Pygame.

003 Petlja događaja u realnom vremenu

Upravljanje događajima u Pygame se ostvaruje kroz petlju u realnom vremenu.
Petlja se kreira sa while iskazom i while blokom, u kojem se ceo kod izvršava ponavljanjem sve dok while uslov je tačan.
U dosadašnjim primerima se koristio: while True: kao uslovni kvalifikator.
Najčešće kod kreira beskonačnu petlju iz koje se izlazi preko sys.exit() funkcije.
Da bi se odgovorilo na Pygame događaje, treba ispitati aktivnost događaja i pregledati stanje svakoga.
Pošto može biti mnogo događaja istovremeno, obično u jednostavnim demo programima se koristi samo jedan tip događaja.
Zato treba parsovati događaje kako se pojavljuju. Ovo se izvodi sa pygame.event.get()
Ovaj kod će kreirati listu događaja koji trenutno čekaju da se obrade.
Kroz listu se ide sa for petljom:
for dogadjaj in pygame.event.get()
što daje događaje prema redosledu njihove pojave u igri.
Obično su to pritisak na dugme, otpuštanje dugmeta i pokreti mišem.
Najčešći događaji na koje se mora odgovoriti su QUIT, kada je prozor zatvoren od strane korisnika.

```
while True:
        for dogadjaj in pygame.event.get():
                if dogadjaj.type == QUIT:
                        sys.exit()
```

004 Događaji tastaturom

Događaji tastaturom uključuju KEYUP i KEYDOWN.
Kada treba odgovoriti na pritisnuto dugme, treba potražiti KEYDOWN događaj, a za pušteno dugme, KEYUP.
Često, najbolji način za odgovor na događaj dugmeta su fleg promenjive.
Npr, kada se pritisne SPACE dugme, fleg poput space_dugme = True se setuje.

A kada se dugme pusti, space_dugme = False se setuje.
Na ovaj način, ne treba odgovoriti na događaj odmah čim se desi, ali se može odgovoriti na fleg promenjivu.
Obično, koristi se ESCAPE kao difolt dugme za napuštanje programa(quit key), kao standardan način. Može se kodovati odgovor na pritisak ESCAPE dugmeta:

```
while True:
        for dogadjaj in pygame.event.get():
                if dogadjaj.type == QUIT:
                        sys.exit()
                elif dogadjaj.type == KEYDOWN:
                        if dogadjaj.key == pygame.K_ESCAPE:
                                sys.exit()
```

U ovom primeru iskaz elif se mora koristiti pri evaluaciji tipova događaja.
Ovo je jednostavno kada se radi sa nekoliko događaja ali problem nastupa kada se mora proveravati veliki broj događaja.
U tom slučaju, jedan način je da se koristi key.name osobenost, koja će vratiti string sa imenom ključa.
Drugi način je vezan za tastaturu.
Po difoltu Pygame ne odgovara uzastopce na dugme koje je pritisnuto; samo šalje događaj kada se dugme po prvi put pritisne, a drugi put kada se pritisnuto dugme pusti.
Da bi Pygame generisao uzastopne događaje kao što je držanje pritisnutog dugmeta, potrebno je uključiti ponavljanje ključa:

```
pygame.key.set_repeat(10)
```

Parametar je broj ponavljanja u milisekundama.
Pozivanje ovog metoda bez parametra onemogućava korišćenje ponavljanja ključa.

005 Prozivanje tastature

Sistem događaja u Pygame nije jedini način za detektovanje korisničkog inputa.
Takođe se može prozvati (poll) ulazni uređaj da bi se videlo da li korisnik interaguje sa programom.
Interfejs za prozivanje tastature u Pygame je sa pygame.key.get_pressed().
Ovaj metod vraća listu bulovih vrednosti, to je dugačka lista flegova, po jedan za svako dugme.
Iste konstante se koriste za indeksiranje rezultujućeg niza bulovih vrednosti (poput pygame.K_ESCAPE).
Dobra strana prozivanja svih dugmadi odjednom je sposobnost detektovanja istovremenog pritiskanja većeg broja dugmadi bez potrebe za prolaskom kroz sistem događaja.
Može se zameniti stari kod upravljanja događajima za detektovanje ESCAPE dugmeta sa sledećim:

```
dugmici = pygame.key.get_pressed()
if dugmici[K_ESCAPE]:
        sys.exit()
```

Sve konstante dugmadi u pygame, poput K_RETURN, odgovaraju njihovim ASCII ekvivalentima.
Može se koristiti Pajton funkcija chr() za vraćanje string reprezentacije ASCII kod broja.
Npr, malo slovo a ima ASCII kod od 97.

006 Primena prozivanja tastature

Ovo je mala igra koja koristi tastaturu u petlji realnom vremenu za testiranje brzine kucanja. Nije baš najpreciznija bez celih reči, samo se testira brzo kucanje jedno po jedno dugme, ali jeste dobar način za rukovanje prozivanjem tastature i podržanih funkcija.
U programu se koristi funkcija random.randint() koja generiše slučajan broj unutar opsega brojeva (opseg se predstavlja kao dva pramatra u zagradi).

007 Upotreba modula time

Drugi koristan modul je modul time.
Njegova funkcija time.clock() vraća trenutan broj sekundi (milisekundi su uključeni kao decimalne vrednosti) koje su prošle od trenutka startovanja programa.
Na početku programa `start_casovnika = 0` pa sa kodom,
`trenutno = time.clock() - start_casovnika` se dobija koliko je vremena u milisekundama proteklo od startovanja programa (to nije vreme koje se meri kao deo igre).
Kada je ispunnjen uslov: `kraj_igre = False` započeta je nova runda igre i sa `trenutno = time.clock() - start_casovnika` funkcija se koristi da bi se omogućilo odbrojavanje od 10 do 1.
Promenjiva sekunde započinje od 11 a time.clock() se oduzima od sekunde da bi se dobila vrednost za odbojavanje.
Kada se pritisne ENTER po prvi put za statovanje igre, startuje se deo koda koji reaguje na ovaj događaj:

```
if kljucevi[K_RETURN]:
     if kraj_igre:
          kraj_igre = False
          rezultat = 0
          sekunde = 11
          start_casovnika = time.clock()
```

Uslov kraj_igre == True, je ispunjen i pre početka igre (pre prvog pritiskanja ENTER) i posle svake odigrane partije igre od 10 sekundi.
Tako se resetuju vrednosti u bloku: kraj_igre, rezultat, sekunde i start_casovnika.
Formula kojom se dobija prosečna brzina kucanja (`brzina = rezultat * 6`) je veoma uslovna i koristi se samo za potrebe igre.
Pošto je promenjiva ključevi na početku glavne petlje igre dobila vrednost:
`kljucevi = pygame.key.get_pressed()` to znači da upotrebom promenjive ključevi kao: `kljucevi[tacan_odgovor]` promenjivu tacan_odgovor tretiram kao vrednost dobijenu pritiskom dugmeta tastature.
Ta vrednost se može vizuelno prikazati kao simbol na tastaturi korišćenjem `chr(tacan_odgovor - 32)`.
Promenjiva ključ_zastavica je fleg koji se setuje kada je po prvi put poritisnuto neko dugme na tastaturi tokom igre.

Brzi_tasteri.py

```
import sys, random, time, pygame
from pygame.locals import *

def stampanje_teksta(font, x, y, tekst, boja = (255,255,255)):
    slika_tekst = font.render(tekst, True, boja)
    ekran.blit(slika_tekst, (x,y))

pygame.init()
ekran = pygame.display.set_mode((600,500))
pygame.display.set_caption("Brzi tasteri v1.0")
font1 = pygame.font.Font(None, 24)
font2 = pygame.font.Font(None, 200)
bela = 255,255,255
```

```
zuta = 255,255,0
kljuc_zastavica = False
tacan_odgovor = 97
sekunde = 11
rezultat = 0
start_casovnika = 0
kraj_igre = True

while True:
    for dogadjaj in pygame.event.get():
        if dogadjaj.type == QUIT:
            sys.exit()
        elif dogadjaj.type == KEYDOWN:
            kljuc_zastavica = True
        elif dogadjaj.type == KEYUP:
            kljuc_zastavica = False

    kljucevi = pygame.key.get_pressed()
    if kljucevi[K_ESCAPE]:
        sys.exit()

    if kljucevi[K_RETURN]:
        if kraj_igre:
            kraj_igre = False
            rezultat = 0
            sekunde = 11
            start_casovnika = time.clock()

    trenutno = time.clock() - start_casovnika
    brzina = rezultat * 6
    if sekunde - trenutno < 0:
        kraj_igre = True
    elif trenutno <= 10:
        if kljucevi[tacan_odgovor]:
            tacan_odgovor = random.randint(97,122)
            rezultat += 1

    ekran.fill((0,100,0))
    stampanje_teksta(font1, 0, 0, "Ajde da vidimo kako brzo kucas!")
    stampanje_teksta(font1, 0, 20, "Kucaj najmanje 10 sekundi...")
    if kljuc_zastavica:
        stampanje_teksta(font1, 500, 0, "<dugme>")

    if not kraj_igre:
        stampanje_teksta(font1, 0, 80, "Vreme: " + str(int(sekunde - trenutno)))

    stampanje_teksta(font1, 0, 100, "Brzina: " + str(brzina) + " slova u minuti")
    if kraj_igre:
        stampanje_teksta(font1, 0, 160, "Pritisni ENTER za pocetak...")

    stampanje_teksta(font2, 0, 240, chr(tacan_odgovor - 32), zuta)
    pygame.display.update()
```

008 Funkcije modula time

U Pygame, vreme se predstavlja u milisekundama (hiljaditi deo sekunde).

Sa pygame.time.get_ticks() se dobija proteklo vreme u milisekundama od trenutka poziva funkcije pygame.init().

Sa pygame.time.wait(milisekunde) se pauzira igra za zadato vreme, a u stvarnosti, dopušta se deljenje procesorskog vremena sa drugim programima pa ova funkcija ima manju preciznost.

Sa pygame.time.delay(milisekunde) se pauzira igra za zadato vreme ali sada se koristi procesor za merenje proteklog vremena pa je ova funkcija precizna.
Sa pygame.time.set_timer(dogadjajID, milisekunde) se postavlja da se događaj sa oznakom dogadjajID pojavljuje u nizu događaja na dati broj milisekundi.
Sa pygame.time.Clock() se kreira nov Clock objekat koji se koristi za praćenje proteklog vremena.
Sa Clock.tick(broj_frejmova) se poziva metoda koja se realizuje jednom u svakom frejmu.
Broj frejmova u zagradi ograničava da se igra ne izvodi sa više od zadatog broja frejmova po sekundi.

Resetovanje.py

```
import pygame
pygame.init()
FONT = pygame.font.SysFont("Sans", 60)
BOJA_TEKSTA = (0, 0, 0)
BOJA_POZADINE = (255, 255, 255)
petlja = True
vreme_od_starta = None
pygame.display.set_caption("Resetovanje")
ekran = pygame.display.set_mode((1400, 800))
sat = pygame.time.Clock()
vreme_od_ENTER = 0
while petlja:
    for dogadjaj in pygame.event.get():
        if dogadjaj.type == pygame.QUIT:
            pygame.quit()
            quit()
        if dogadjaj.type == pygame.KEYDOWN:
            if dogadjaj.key == pygame.K_RETURN:
                vreme_od_starta = pygame.time.get_ticks()

    ekran.fill(BOJA_POZADINE)
    if vreme_od_starta:
        vreme_od_ENTER = (pygame.time.get_ticks() - vreme_od_starta) / 1000
        poruka1 = 'Proteklo sekundi od pritiska na dugme ENTER: ' +
str(vreme_od_ENTER)
        ekran.blit(FONT.render(poruka1, True, BOJA_TEKSTA), (20, 20))

    pygame.display.flip()
    sat.tick(60)

pygame.quit()
```

Program meri proteklo vreme od prvog klika na dugme ENTER do sledećeg klika na dugme ENTER.
Svaki klik na ENTER resetuje tajmer

Zadaci

012) Modifikovati kod programa Resetovanje.py, i nazvati ga Za_5_sekundi.py, tako da program traži od igrača da u roku od 5 sekundi klikne na dugme ENTER. Ako se to ne desi, pojavljuje se poruka o kašnjenju.