Kurs2: Programiranje igara sa programskim jezikom Pajton
028 čas: Program Pita.py
Teme: pygame biblioteka, upotreba teksta sa fontovima, crtanje oblika

Biblioteka za igre se naziva Pygame i razvijena je da bi se crtala grafika, koristio korisnički ulaz, animacija i koristio tajmer koji čini da se igra odvija u konstantnom frejm rejtu. Pygame omogućava i rad sa audio plejbekom i korišćenje miša i tastature.

001 Upotreba Pygame

```
import pygame, sys
from pygame.locals import *
pygame.init()
ekran = pygame.display.set_mode((600,500))
```

Prvi korak je import pygame i sys modula u Pajton program a time se omogućava da naš program koristi funkcije koje se nalaze u ovim modulima. Sve Pygame funkcije koje rade sa grafikom, zvukom i drugim mogućnostima se nalaze u pygame modulu. Istovremeno sa importovanjem pygame modula, importuju se svi ostali moduli koji su unutar pygame modula, poput pygame.images i pygame.mixer.music. Zato nema potrebe posebno importovati module unutar ovog modula.

Sledeće je import svih konstanti u Pygame tako da su spremne za korišćenje u kodu. I ova linija predstavlja import naredbu ali koristi drugačiji format: from ime_modula import *. Obično kada se poziva funkcija koja je u ovom modulu, koristi se format: ime_modula.ime_funkcije(), posle importovanja modula. Ali, ako se korisit ovakav drugačiji format import naredbe, može se preskočiti korišćenje ime_modula i samo upotrebljavati ime_funkcije() (kao kada se koriste ugrađene funkcije u Pajtonu). Ovaj format se koristi zato što pygame.locals sadrži nekoliko konstanti promenjivih koje se lako identifikuju kao deo pygame.locals modula pa i nema potrebe kucati taj deo importa. Ipak, za sve ostale module je bolje koristiti klasičan format importovanja.

U liniji `pygame.init()` je poziv funkcije koji se mora obaviti posle importovanja pygame modula ali pre pozivanja bilo koje druge Pygame funkcije. Ovime se inicijalizuju svi importovani pygame moduli.

Sledeće je pristup sistemu prikaza i kreiranje prozora. Funkcija `pygame.display.set_mode()` vraća pygame.Surface objekat za prozor i koristi se za predstavljanje bilo koje slike (image). U zagradi ((600, 500)) je ntorka koja govori funkcije set_mode() koje su dimenzije prozora u pikselima. Pišu se duple zagrade pošto u funkcijsku zagradu se smešta podatak kao jedna ntorka sa dva elementa. U promenjivu ekran se smešta vraćeni pygame.Surface objekat.

002 Štampanje teksta

Pygame podržava izlaz teksta na grafički prozor korišćenjem pygame.font. Da bi se crtao tekst, prvo se mora kreirati font objekat:

```
moj_font = pygame.font.Font(None,60)
```

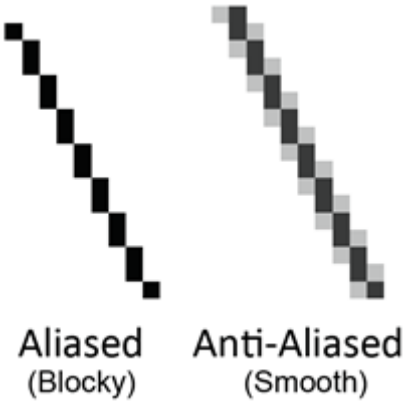
Parametri konstruktora funkcije pygame.font.Font() su string koji predstavlja naziv fonta koji će se koristiti i ceo broj koji predstavlja veličinu fonta za prikaz. Naziv TrueType fonta se može predati u pygame.font.Font() konstruktor, poput „Arial“ ali korišćenjem None izaziva da se koristi difolt Pygame font. Tekst se u Pygame mora prvo renderovati na površinu koja se zatim crta na ekranu. Pošto je ovo proces koji traje neko vreme, lakše je prvo kreirati tekst površinu

(ili sliku) u memoriji a zatim nacrtati tekst kao sliku. Ako se tekst neće menjati, bolje je prerenderovati tekst u sliku.

```
bela = 255,255,255
plava = 0,0,255
tekst_slika = moj_font.render("Zdravo Pygame", True, bela)
```

Sa poslednjom linijom se kreira pygame.Surface objekat tekst_slika sa nacrtanim tekstom na njoj. To se izvodi pozivanjem objektove Font metode render(). Prvi parametar je string teksta koji će se rederovati, drugi parametar je Bulova vrednost vezana za anti-aliasing, treći parametar je boja teksta (RGB vrednost) a poslednji je boja pozadine (ako je nema uzima se providna pozadina po difoltu).

Anti-aliasing je grafička tehnika kojom kreirani tekstova i oblici izgledaju manje kockasti, tako što se na njihove ivice dodaje blago zamućenje. Ovom tehnikom se povećava vreme potrebno za iscrtavanje oblika.



Objekat tekst_slika je zapravo površina na koju se crta sa screen.blit().

Da bi se nacrtao tekst, obično se koristi proces čišćenja ekrana, crtanja a zatim osvežavanje ekrana:

```
ekran.fill(plava)
ekran.blit(tekst_slika, (100,100))
pygame.display.update()
```

Ako bi se u ovom momentu startovao program, video bi se prozor za tren a odmah i nestao pošto nije definisano odlaganje (delay) prikaza.

Metoda fill() nije funkcija već metoda pygame.Surface objekata. Ona će popuniti ceo objekat sa vrednošću parametra koji predstavlja željenu boju.

Funkcija blit() crta sadržaj jednog Sufrace objekta u drugi Surface objekat. U ovom slučaju se kopira objekat tekst_slika u objekat ekran. Pri tome s ekoriste dva parametra, prvi je izvor Surface objekta koji će biti kopiran u drugi objekat; drugi parametar je ntorka sa dva cela broja za x i y vrednosti (u odnosu na gornji levi ugao drugog objekta).

003 Petlja

Postoje dva problema: nestala slika se samo jednom pojavila pa je nestala i nema mogućnosti unosa od strane korisnika.

Da bi se rešio prvi problem, koristiće se u kodu while petlja koja će realizovati kod sve dok je uslov tačan: while True:

Sledeće, kreiraće se menadžer događaja (event handler). Potrebno je da prozor ostane na ekranu sve dok korisnik ga ne zatvori. Događaj zatvaranja prozora će biti klik na X u gornje desnom uglu prozora, ili samo pritisak na bilo koje dugme tastature.

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

```
        if dogadjaj.type in (QUIT, KEYDOWN):
            sys.exit()
```

U većini pygame igara koristi se main game loop (petlja glavnog dela igre). Unutar ove petlje se odigravaju trei dela koda naizmenično: upravljanje (menadžment) događajima (event handling), updejtovanje stanja igre, crtanje stanja igre na ekranu.

Stanje igre predstavlja set vrednosti za sve promenjive u kodu. Bilo kakva promena u vrednostima promenjivih u kodu igre se naziva promena stanja igre (game state). Izlazak iz igre ili pauziranje igre sprečavaju da se promeni stanje igre. Stanje igre se updejtuje kao odgovor na događaje (klik mišem ili pritisak na tastaturi) ili protok vremena, dok petlja igre stalno proverava da li je došlo do bilo kakve promene u stanju igre. Unutar petlje glavnog dela igre postoji kod koji proverava koji su se događaji desili (u Pygame to se izvodi pozivanjem funkcije pygame.event.get()). Ova petlja takođe ima kod koji updejtuje stanje igre u zavisnosti od događaja u igri i to se naziva menadžment događajima.

Svaki put kada korisnik uradi neku od akcija tokom igre, poput pritiska tastature ili pomeranje mišem unutar programskog prozora, kreira se pygame.event.ime_dogadjaja objekat unutar Pygame bibiloteke za pamćenje ovakvog događaja. Ovo je tip objekta koji postoji u event modulu, koji je unutar pygame modula. Može se otkriti koji se to događaj desio pozivanjem funkcije pygame.event.get(), koja vraća listu pygame.event.ime_dogadjaja objekata (ponekad se kratko nazivaju Event objekti). Ova lista Event objekata sadrži svaki događaj koji se desio od poslednjeg pozivanja pygame.event.get() funkcije.

U kodu, for petlja će iterirati preko liste Event objekata koje vraća pygame.event.get(). Na svakoj iteraciji promenjiva dogadjaj će dobiti vrednost sledećeg Event objekta sa liste. Redosled događaja na listi je prema redosledu dešavanja događaja u igri. Ako se nije desio nikakav događaj, vraća se prazna lista.

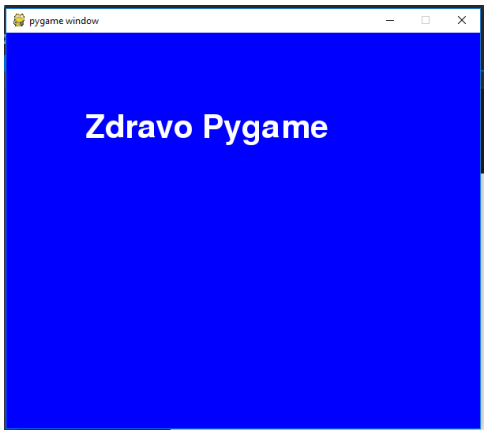
Event objekti imaju člana promenjivu (atributi ili osobenosti, properties) koja se naziva type, koja govori koji tip događaja objekat predstavlja. Pygame ima konstantne promenjive za svaki od mogućih tipova u pygame.locals modulima. Linija `if dogadjaj.type in (QUIT, KEYDOWN):` proverava da li je Event objekat type identičan sa konstantama QUIT ili sa KEYDOWN. Ako jeste, desiće se poziv funkcije sys.exit() za prekid programa. Konstanta QUIT označava klik na X prozora, dok KEYDOWN označava pritisak na bilo koje dugme tastature. U ovom slučaju, ne postoji kod za menadžera događaja, pošto se ne proveravaju dešavanja ostalih mogućih događaja.

Na kraju petlje se dodaje kod za crtanje i osvežavanje ekrana:

```
ekran.fill(plava)
ekran.blit(tekst_slika, (100,100))
pygame.display.update()
```

program Pozdrav.py

```
import pygame, sys
from pygame.locals import *
pygame.init()
ekran = pygame.display.set_mode((600,500))
moj_font = pygame.font.Font(None,60)
bela = 255,255,255
plava = 0,0,255
tekst_slika = moj_font.render("Zdravo Pygame", True, bela)
ekran.fill(plava)
ekran.blit(tekst_slika, (100,100))
pygame.display.update()
while True:
    for dogadjaj in pygame.event.get():
        if dogadjaj.type in (QUIT, KEYDOWN):
            sys.exit()
    ekran.fill(plava)
    ekran.blit(tekst_slika, (100,100))
    pygame.display.update()
```



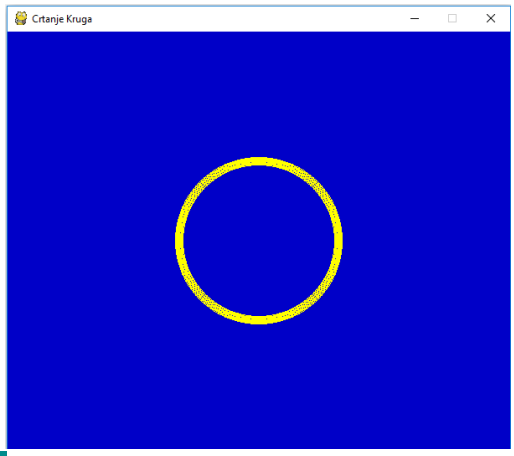
004 Crtanje krugova

Mogu se crtati različiti oblici sa pygame.draw bibliotekom. Za crtanje kruga, koristi se pygame.draw.circle() istovremeno se predaju parametri za uređenje veličine, boje i pozicije kruga. Metoda pygame.display.set_caption() ima kao parametar string koji će biti ispisan u naslovu prozora.

program Crtanje_Kruga.py

```
import pygame, sys
from pygame.locals import *
pygame.init()
ekran = pygame.display.set_mode((600,500))
pygame.display.set_caption("Crtanje Kruga")
while True:
    for dogadjaj in pygame.event.get():
        if dogadjaj.type in (QUIT, KEYDOWN):
            sys.exit()

    ekran.fill((0,0,200))
    #crta krug
    color = 255,255,0
    position = 300,250
    radius = 100
    width = 10
    pygame.draw.circle(ekran, color, position, radius, width)
    pygame.display.update()
```



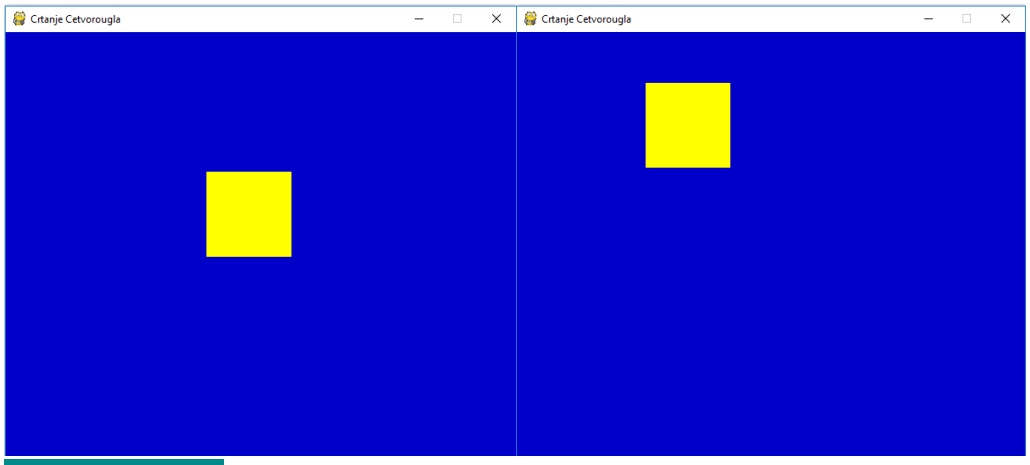
005 Crtanje pravougaonika

Za crtanje pravougaonika, koristi se pygame.draw.rect() funkcije sa nekoliko parametara. U ovom primeru ne samo da se crta pravougaonik već se on i pomera. To se radi tako što se vodi evidencija o poziciji pravougaonika izvan while petlje (x_pozicija, y_pozicija) i kreira se par promenjivih za opis brzine (x-brzina, y_brzina). Unutar petlje može se updejtovati pozicija korišćenjem brzine a preko logike se održava pravougaonik unutar ekrana. Zato kada god pravougaonik dostigne ivicu ekrana, brzina se postavi suprotno.

program Crtanje_Cetvorougla.py

```
import pygame, sys
from pygame.locals import *
pygame.init()
ekran = pygame.display.set_mode((600,500))
pygame.display.set_caption("Crtanje Cetvorougla")
x_pozicija = 300
y_pozicija = 250
x_brzina = 2
y_brzina = 1
while True:
    for dogadjaj in pygame.event.get():
        if dogadjaj.type in (QUIT, KEYDOWN):
            sys.exit()

    ekran.fill((0,0,200))
    #pomera cetvorougao
    x_pozicija += x_brzina
    y_pozicija += y_brzina
    #zadrzava cetvorougao unutar ekrana
    if x_pozicija > 500 or x_pozicija < 0:
        x_brzina = -x_brzina
    if y_pozicija > 400 or y_pozicija < 0:
        y_brzina = -y_brzina
    #crta cetvorougao
    boja = 255,255,0
    debljina_linije = 0 #popunjeno skroz
    pos = x_pozicija, y_pozicija, 100, 100
    pygame.draw.rect(ekran, boja, pos, debljina_linije)
    pygame.display.update()
```



006 Crtanje linija

Prave se mogu crtati korišćenjem pygame.draw.line() funkcije. Crtanje linija je donekle kompleksnije od crtanja drugih oblika, pošto se moraju navesti i početna pozicija i krajnja pozicija linije. Crta se linija u bilo kojoj boji i sa željenom debljinom.

program Crtanje_Linije.py

```
import pygame, sys
from pygame.locals import *
pygame.init()
ekran = pygame.display.set_mode((600,500))
pygame.display.set_caption("Drawing Lines")
while True:
    for dogadjaj in pygame.event.get():
        if dogadjaj.type in (QUIT, KEYDOWN):
            sys.exit()

    ekran.fill((0,80,0))
    #crta liniju
    boja = 100,255,200
    debljina_linije = 8
    pygame.draw.line(ekran, boja, (100,100), (500,400), debljina_linije)
    pygame.display.update()
```

007 Crtanje lukova

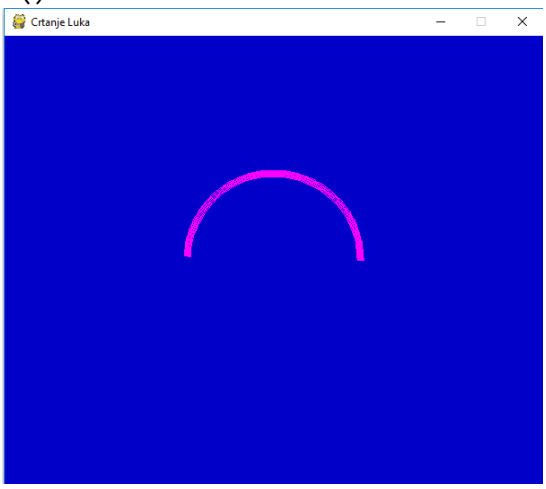
Luk je delimična kružnica koji se može nacrtati pomoću pygame.draw.arc() funkcijom. Ovo je kompleksan oblik koji traži dodatne parametre kao što su granice luka sa početkom u gornjem

levom uglu, širina i visina. Potrebno je odrediti početni ugao i krajnji ugao (uglovi u radijanima). Da bi se konvertovali stepeni u radijane, koristi se funkcija math.radians(), gde je ugao u stepenima parametar (zato se importuje i math modul).

program Crtanje_Luka.py

```
import math, pygame, sys
from pygame.locals import *
pygame.init()
ekran = pygame.display.set_mode((600,500))
pygame.display.set_caption("Crtanje Luka")
while True:
    for dogadjaj in pygame.event.get():
        if dogadjaj.type in (QUIT, KEYDOWN):
            sys.exit()

    ekran.fill((0,0,200))
    #crta luk
    boja = 255,0,255
    pozicija = 200,150,200,200
    ugao_start = math.radians(0)
    ugao_kraj = math.radians(180)
    debljina_linije = 8
    pygame.draw.arc(ekran, boja, pozicija, ugao_start, ugao_kraj, debljina_linije)
    pygame.display.update()
```



Igra Pita

Ova igra pokazuje osnovna načela u kreiranju grafičkih igara. U igri se koristi tastatura sa dugmadima 1, 2, 3 i 4. Pritiskom na neko od njih, odgovarajuće parče pite se crta. Kada se nacrtaju sva četiri parčeta pite, cela pita menja boju kao i svi napisani brojevi na njoj.

Ovde se koristi state-base programming (programmiranje zasnovano na stanju). Četiri dela pite se se crtaju automatski kada igrač pritisne odgovarajuće dugme. Umesto toga, state flag (pokazivač (fleg) stanja) se postavlja (setuje) kada je dugme pritisnuto, i taj fleg se kasnije koristi za crtanje delova pite bazirano na tom flegu. Ovaj koncept ukazuje kako se rukuje događajima i kako igračeva interakcija utiče na njih **indirektno**.

program Pita.py

```
import math, pygame, sys
from pygame.locals import *
pygame.init()
ekran = pygame.display.set_mode((600,500))
pygame.display.set_caption("Igra Pita - pritisni 1, 2, 3 ili 4")
moj_font = pygame.font.Font(None, 60)
boja = 200, 80, 60
```

```
debljina_linije = 4
x = 300
y = 250
poluprecnik = 200
pozicija = x - poluprecnik, y - poluprecnik, poluprecnik * 2, poluprecnik * 2
deo1 = False
deo2 = False
deo3 = False
deo4 = False
while True:
    for dogadjaj in pygame.event.get():
        if dogadjaj.type == QUIT:
            sys.exit()
        elif dogadjaj.type == KEYUP:
            if dogadjaj.key == pygame.K_ESCAPE:
                sys.exit()
            elif dogadjaj.key == pygame.K_1:
                deo1 = True
            elif dogadjaj.key == pygame.K_2:
                deo2 = True
            elif dogadjaj.key == pygame.K_3:
                deo3 = True
            elif dogadjaj.key == pygame.K_4:
                deo4 = True
    #ciscenje ekrana
    ekran.fill((0,0,200))
    #crtanje cetiri broja
    slika1 = moj_font.render("1", True, boja)
    ekran.blit(slika1, (x + poluprecnik / 2 - 20, y - poluprecnik / 2))
    slika2 = moj_font.render("2", True, boja)
    ekran.blit(slika2, (x - poluprecnik / 2, y - poluprecnik / 2))
    slika3 = moj_font.render("3", True, boja)
    ekran.blit(slika3, (x - poluprecnik / 2, y + poluprecnik / 2 - 20))
    slika4 = moj_font.render("4", True, boja)
    ekran.blit(slika4, (x + poluprecnik / 2 - 20, y + poluprecnik / 2 - 20))
    #odlucivanje o crtanju dela pite
    if deo1:
        ugao_start = math.radians(0)
        ugao_kraj = math.radians(90)
        pygame.draw.arc(ekran, boja, pozicija, ugao_start, ugao_kraj, debljina_linije)
        pygame.draw.line(ekran, boja, (x, y), (x, y - poluprecnik), debljina_linije)
        pygame.draw.line(ekran, boja, (x, y), (x + poluprecnik, y), debljina_linije)
    if deo2:
        ugao_start = math.radians(90)
        ugao_kraj = math.radians(180)
        pygame.draw.arc(ekran, boja, pozicija, ugao_start, ugao_kraj, debljina_linije)
        pygame.draw.line(ekran, boja, (x, y), (x, y - poluprecnik), debljina_linije)
        pygame.draw.line(ekran, boja, (x, y), (x - poluprecnik, y), debljina_linije)
    if deo3:
        ugao_start = math.radians(180)
        ugao_kraj = math.radians(270)
        pygame.draw.arc(ekran, boja, pozicija, ugao_start, ugao_kraj, debljina_linije)
        pygame.draw.line(ekran, boja, (x, y), (x - poluprecnik, y), debljina_linije)
        pygame.draw.line(ekran, boja, (x, y), (x, y + poluprecnik), debljina_linije)
    if deo4:
        ugao_start = math.radians(270)
        ugao_kraj = math.radians(360)
        pygame.draw.arc(ekran, boja, pozicija, ugao_start, ugao_kraj, debljina_linije)
        pygame.draw.line(ekran, boja, (x, y), (x, y + poluprecnik), debljina_linije)
        pygame.draw.line(ekran, boja, (x, y), (x + poluprecnik, y), debljina_linije)
    #odlucivanje da li je pita gotova
    if deo1 and deo2 and deo3 and deo4:
```

```
    boja = 0,255,0
pygame.display.update()
```