
Uputstvo za izradu igrice PONG

Originalni tekst možete naći na linkovima:

<https://ryanstutorials.net/pygame-tutorial/pygame-basics.php>

<https://ryanstutorials.net/pygame-tutorial/pygame-pong-simple.php>

Ispisan kod igrice za upoređivanje možete naći na:

<https://github.com/Dinolsanovic/Python-prvi-koraci/tree/main/pong>

Kada radite sa Pygame-om, ili bilo kojom drugom bibliotekom igara, postoje različita mjesta u vašem kodu koja je dobra praksa da budete dobro organizovana. Postoje specifični odjeljci za inicijalizaciju varijabli, rukovanje događajima (klikovi mišem i pritisci dugmeta), ažuriranje stanja igre (matematika i logika za premještanje objekata, na primjer) i crtanje. Lako je pomiješati ove koncepte jedan u drugi, ali pokušajte ih držati odvojenima najbolje što možete. Tako da svoj kod komentarišete i specifične dijelove koda odvajajte praznim linijama u editoru.

U programskom kodu ispod definisat će mo polaznu tačke to jest elemente vezane za našu igricu. Prije pisanja programskog dijela koji opisuje konkretne događaje u igrici mi će mo definisati:

Veličinu prozora za igranje

Boje koje dodjeljujemo varijablama kako bi ih lakše pozivali za svaki element posebno

Postavljamo sat koji određuje brzinu izvođenja igre.

Definišemo izlaz iz igre

Ispitujemo unos preko tastature


Određujemo brzinu obnavljanja ekrana FPS

Takođe ostavljamo posebne dijelove koda koji je specifičan za svaku igru zasebno.

Kada budete prepisivali ovaj okvirni kod igre nemojte prepisivati redne brojeve. Oni su tu samo da bi kasnije opisali događaje koji se dešavaju u određenoj liniji koda.

Detaljnije upute za svaku funkciju sa kojom raspolaže Pygame biblioteka možete naći ovdje:

<https://www.pygame.org/docs/>



KADA BUDETE ČITALI OVAJ TEKST, BIT ĆE VAM
JASNO ZBOG ČEGA VAM TREBA JOŠ JEDAN
MONITOR ZA UČENJE PROGRAMIRANJA.

Okvirni program za početak pisanja igrice.

```
1. import pygame, sys, random
2. from pygame.locals import *
3. pygame.init()
4.
5. # Colours
6. BACKGROUND = (255, 255, 255)
7.
8. # Game Setup
9. FPS = 60
10. fpsClock = pygame.time.Clock()
11. WINDOW_WIDTH = 400
12. WINDOW_HEIGHT = 300
13.
14. WINDOW = pygame.display.set_mode((WINDOW_WIDTH, WINDOW_HEIGHT))
15. pygame.display.set_caption('My Game!')
16.
17. # The main function that controls the game
18. def main():
19.     looping = True
20.
21.     # The main game loop
22.     while looping:
23.         # Get inputs
24.         for event in pygame.event.get():
25.             if event.type == QUIT:
26.                 pygame.quit()
27.                 sys.exit()
28.
29.         # Processing
30.         # This section will be built out later
31.
32.         # Render elements of the game
33.         WINDOW.fill(BACKGROUND)
34.         pygame.display.update()
35.         fpsClock.tick(FPS)
36.
37. main()
38.
39.
40.
```

- **Linije 1 i 2 -**

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

Uvoz biblioteka koje ćemo koristiti za kreiranje igre

- **pygame**: Glavna biblioteka za pravljenje igara.
- **sys**: Koristi se za izlazak iz programa.
- **random**: koristimo za generisanje slučajnih brojeva (trenutno nije korišteno, ali može biti korisno za buduće funkcionalnosti).
- ****from pygame.locals import *****: Omogućava direktan pristup često korištenim Pygame konstantama, kao što je QUIT.

- **Linija 3 -**

```
pygame.init()
```

Ovdje se pripremaju biblioteke i ostali resursi igre za daljnju upotrebu, poput traženja OS-a za resurse itd., Ali većinom se ne trebamo brinuti o onome što se događa.

- **Linija 6 –**

```
BACKGROUND = (255, 255, 255)
```

varijabla BACKGROUND dobija RGB vrijednosti za bijelu boju. Svaka boja u RGB formatu ima 3 vrijednosti za crvenu, žutu i plavu boju. Tako da svaka boja ima svoj kod od 3 broja koji se unosi kao tulpice.

- **Linija 9 –**

```
FPS = 60
```

Određujemo brzinu obnavljanja ekrana to jest koliko puta se ekran ponovo iscrta u jednoj sekundi FPS – Frames per second

- **Linija 10 –**

```
fpsClock = pygame.time.Clock()
```

stvaramo objekat **sat (clock)** koji kasnije možemo koristiti za kontrolu brzine naše igre.

- **Linije 11 i 12** - postavite varijable koje će se kasnije koristiti za stvaranje prozora ovih dimenzija za igru. Izmjenom brojčanih parametara možemo uvećavati ili smanjivati veličinu prozora igrice.

```
WINDOW_WIDTH = 400
WINDOW_HEIGHT = 300
```

- **Linija 14 –**

```
WINDOW = pygame.display.set_mode((WINDOW_WIDTH, WINDOW_HEIGHT))
```

Sa ovom komandom kreiramo prozor sa određenim dimenzijama iz lija 11 i 12. WINDOW je glavni objekat tipa površine za crtanje. Tu će biti ispisano, nacrtano sve što vidimo u igri.

- **Linija 15**

```
pygame.display.set_caption('My Game!')
```

postavlja tekst „My game!“ koji će se prikazati na vrhu prozora. Ovu funkciju zapravo možete nazvati u bilo kojem trenutku tokom igre kako biste promijenili tekst koji je tamo prikazan.

- **Linija 18 –**

```
def main():
```

def: Ključna riječ u Pythonu za definisanje funkcija.

main: Naziv funkcije. U ovom slučaju, to je glavna funkcija koja pokreće igru. Unutar ove funkcije će se odvijati kompletna igra do izlaza iz nje.

- **Linija 19-**

```
looping = True
```

looping: Ova promenljiva kontroliše izvršavanje glavne petlje igre. Kada je postavljena na True, petlja igre će nastaviti da se izvršava. Postavljanjem na False možete prekinuti petlju i završiti igru. Postavljanje vrijednosti ove petlje možemo učiniti u bilo kojem dijelu main funkcije.

- **Linija 22-**

```
while looping:
```

Da bi omogućili stalni prikaz prozora u kojem se odvija igra, sve funkcije vezane za izvršavanje igre su smještene u beskonačnu petlju **while True**. U našem slučaju varijabla looping ima vrijednost True. U stvari tek sada pišemo kod igrice u kojem razlikujemo neke elemente:

- Kontrola unos korisnika (event handling).
- Obrada logike igre.
- Crtanje (render) elemenata igre.

- **Linija 24 –**

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

ova linija hvata sve trenutne događaje od posljednjeg puta kada smo provjerili događaje. Događaji su među akcije poput pritisaka na tastaturi i pokreta miša. Vrsta QUIT događaja šalje

OS ako, na primjer, kliknemo dugme za zatvaranje na prozoru. Zbog toga igra ne odustaje, ali daje programu naznaku da bi trebala. Tada to možemo učiniti po vlastitim uvjetima, npr. osiguravajući da se datoteke s visokim rezultatima itd. Spremi prije nego što odustanu.

- **Linija 25 –**

```
if event.type == QUIT:
```

event.type:

Svojstvo objekta event koje predstavlja vrstu događaja.

Na primjer:

QUIT: Kada korisnik pokuša da zatvori prozor igre.

KEYDOWN: Kada korisnik pritisne taster na tastaturi.

MOUSEBUTTONDOWN: Kada korisnik klikne mišem.

QUIT: Konstantna vrijednost u Pygame-u koja označava događaj zatvaranja prozora.

Ovaj događaj se generiše kada korisnik klikne na dugme za zatvaranje ("X") u uglu prozora.

- **Linija 26 i 27**

```
pygame.quit()  
sys.exit()
```

naređujemo PyGameu da se pravilno zatvori, a zatim pošalje signal OS-u da može zatvoriti program.

- **Linija 30** - još ništa ne radimo ovdje, ali to ćemo raditi u narednih nekoliko stranica tutorijala.
- **Linija 33 –**

```
WINDOW.fill(BACKGROUND)
```

šalje naredbu WINDOW-u kako bi cijeli ekran slikao zadanu boju. Ako to ne učinite, tada će sve što je trenutno na ekranu ostati tamo i novi predmeti biti postavljeni iznad vrha što većinu vremena nije ono što želimo. (Kako se vaš lik kreće, na primjer, ostavili bi trag.)

- **Linija 34 –**

```
pygame.display.update()
```

Očitava sve naredbe koje su poslone na WINDOW i primjenjuje ih na stvarni ekran u jednom potezu. Na ovaj način dobijate konstantan ispis svih grafičkih elemenata na ekranu.

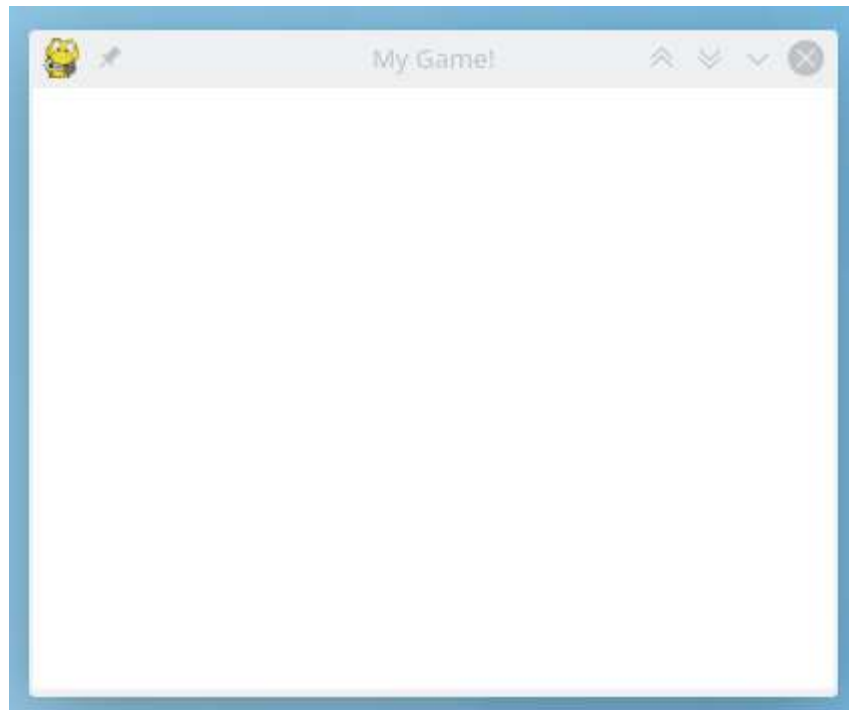
- **Linija 35 –**

```
fpsClock.tick(FPS)
```

ova naredba traži od sistema da nakratko zastane, dok takva da će se igra pokrenuti na datom Frames Per Second. Da to nismo učinili, petlja bi se odvijala u mnogo bržem ciklusu i potencijalno bi se svaki put razlikovala kroz petlju, ovisno o tome što se obrada događa. Krajnji rezultat bila bi pogrešna brzina vaše igre koja je varirala na svakom uređaju ovisno o performansama hardvera uređaja.

- **Linija 37** mogli smo samo staviti linije 19 - 35 ne u funkciju i to bi djelovalo efektivno u redu, ali stavljanjem u funkciju poput ove možemo stvoriti nekoliko sličnih funkcija za predstavljanje različitih ekrana u našoj igri (kućni ekran, postavke, visoki rezultati, stvarna igra, nivoi igara itd.). Također razdvaja stvari koje imaju nekoliko prednosti koje će postati očiglednije jer počinjemo graditi svoje igre i kôd postaje duži.

U ovom dijelu teksta je napravljena baza za vaše buduće projekte pravljenja jednostavnih igara. U nastavku teksta dodajemo stvarni kod igre to jest komande i operacije koje su specifične za ovu igru. Gore napisani kod kada pokrenete dobit će te samo definisani prozor za igranje.



NAPOMENA TEKST NAPISAN SIVOM BOJOM VEĆ POSTOJI U PREDLOŠKU. TEKST NAPISAN CRVENOM BOJOM SE DODAJE U PREDLOŽAK. OBRATITI PAŽNJU NA POZICIJU TEKSTA. JER NEKE KOMANDE MORAJU BITI UVUČENE NA ODREŽENE POZICIJE, INAČE ĆE SE POKAZATI GREŠKA PRILIKOM IZVOĐENJA PROGRAMA

Počinjemo sa prilagođavanjem izgleda prozora igre.

U liniji 15 mijenjamo tekst **My Game!** sa **Pong!**

Također želimo da prozor postane malo veći, tako da imamo više prostora za igranje igre. Promijenite **WINDOW_WIDTH 800** i **WINDOW_HEIGHT 600** na linijama 11 i 12 (ili bilo kojoj drugoj veličini koju želite).

Strategija razvoja igre

Pravilan pristup izradi igre je od velike pomoći. Bitan je metodičan pristup koji nam omogućava da postupno izgradimo igru, testirajući stalno dok idemo. Na taj način, ako pogriješite i prekršite stvari, saznat ćete ranije i to će biti brže i lakše popraviti. Takođe možete eksperimentisati sa svojim kreacijama i idejama u pojedinim odjeljcima koda igre.

Ono što želite raditi je stalno pitati, "što je sljedeći najlakši korak koji bih mogao poduzeti?" a zatim to implementirajte, testirajte i ponovite. Ako preduzmete ovaj pristup, onda izgradnja igre postaje skup jednostavnih koraka, a ne jedan veliki zastrašujući problem. Ne brinite ako još ne znate koji će biti svi koraci. To je poput rješavanja slagalice, što više komada dođete na mjesto, to je lakše što kasnije postaju.

Prilikom izrade grafičkog programa pomoću biblioteke poput PyGame-a, ponekad pogreške koje dobijete mogu biti malo kriptične. Ako se to dogodi, Googling greška često vam može dati nagovještaje o tome što bi moglo biti pogrešno.

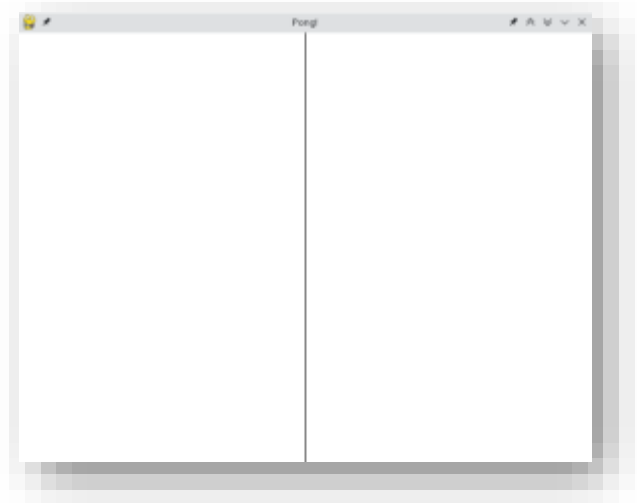
Crtanje elemenata igre

U ovom dijelu dodajemo grafičke elemente koji se nalaze u igri. Dobro mjesto za početak je crtanje elemenata igre na ekranu kada imate samo nekoliko jednostavnih elemenata poput Pong-a. Elementi će biti statični i neće ništa učiniti, ali tada ih možemo progresivno učiniti aktivnim. Najlakši je središnja linija, tako da ćemo početi s tim.

- # Colours
- BACKGROUND = (255, 255, 255)
- ELEMENTCOLOUR = (100, 100, 100)

- `WINDOW.fill(BACKGROUND)`
- `pygame.draw.line(WINDOW, ELEMENTCOLOUR, (WINDOW_WIDTH // 2, 0), (WINDOW_WIDTH // 2, WINDOW_HEIGHT), 2)`
- `pygame.display.update()`

Ako spremite i testirate program, sada biste trebali imati jednu liniju koja ide dolje na sredini zaslona. Također ćete primijetiti da smo koristili dvostruki znak (`//`) umjesto normalnog pojedinačnog znaka (`/`) za podjelu. Dvostruki operator radi isto, ali vraća rezultat kao cijeli broj (cijeli broj). To je važno ovdje (i u mnogim drugim izračunima koje ćemo učiniti) npr. kada postavljamo koordinate oni trebaju biti cijeli brojevi, a ne decimalni.



U template dodajemo loptu i pločice.

Neposredno nakon `"pygame.display.set_caption('Pong!')"` i neposredno prije `def main ()`: dodajte sljedeće linije.

`PADDLEINSET` će biti koliko daleko od prozora su odmaknute pločice :

`PADDLEWIDTH` određuje širinu pločice

`PADDLEHEIGHT` određuje visinu pločice

`BALLSIZE` određuje veličinu loptice

- `pygame.display.set_caption('Pong!')`
- `# Game Element Variables`
- `PADDLEINSET = 20`
- `PADDLEWIDTH = 10`
- `PADDLEHEIGHT = 60`
- `BALLSIZE = 10`

Budući da će se varijable koje ukazuju na to gdje su pločice na zaslonu promijeniti, postaviti ćemo ove unutar glavne funkcije (a oni neće biti imenovani u svim gornjim tokovima).

- `looping = True`
-
- `leftPaddleY = 50`
- `rightPaddleY = 50`
- `ballX = WINDOW_WIDTH // 2`
- `ballY = WINDOW_HEIGHT // 2`

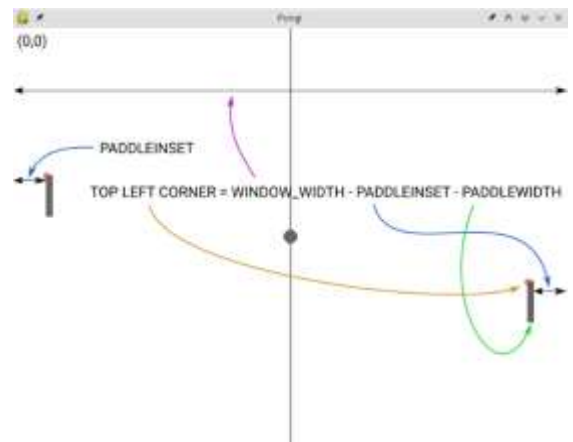
Zatim ćemo stvoriti **Rect** objekte za pločice:

- `# Processing`
-
- `leftPaddleRect = pygame.Rect(PADDLEINSET, leftPaddleY, PADDLEWIDTH, PADDLEHEIGHT)`
- `rightPaddleRect = pygame.Rect(WINDOW_WIDTH - PADDLEINSET - PADDLEWIDTH, rightPaddleY, PADDLEWIDTH, PADDLEHEIGHT)`

Matematika ovdje može izgledati malo zbunjena, ali zapamtite da kada definiramo pravougaonik u Pygameu, navodimo koordinate gornjeg lijevog ugla.

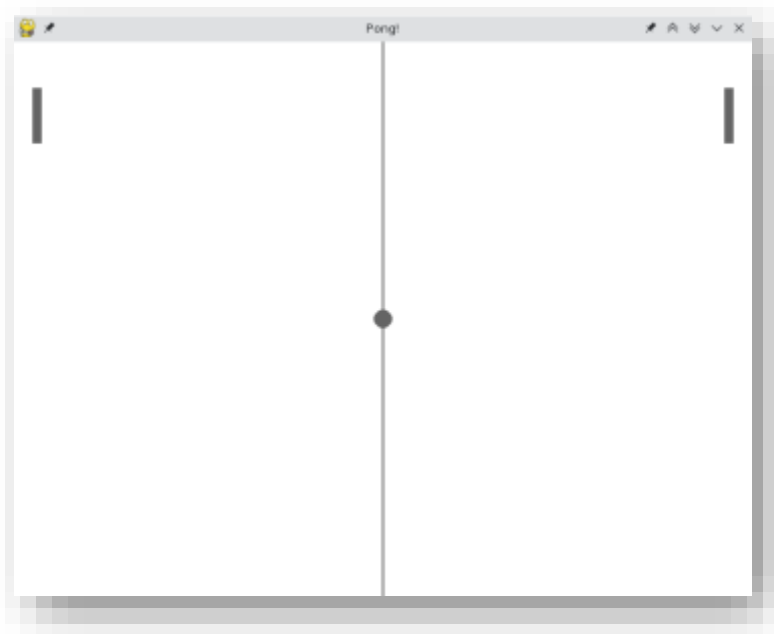
(U dijagramu desno, crveni krugovi označavaju tačke koje ćemo izračunati koordinate.)

I na kraju, nacrtati elemente:



- `WINDOW.fill(BACKGROUND)`
- `pygame.draw.line(WINDOW, ELEMENTCOLOUR, ((WINDOW_WIDTH // 2) - 1, 0), ((WINDOW_WIDTH // 2) - 1, WINDOW_HEIGHT), 2)`
- `pygame.draw.rect(WINDOW, ELEMENTCOLOUR, leftPaddleRect)`
- `pygame.draw.rect(WINDOW, ELEMENTCOLOUR, rightPaddleRect)`
- `pygame.draw.circle(WINDOW, ELEMENTCOLOUR, (int(ballX), int(ballY)), BALLSIZE)`
- `pygame.display.update()`

Kada pokrenete napisani kod trebali bi dobiti sljedeću sliku



Sljedeći najlakši korak je da pokrećemo pločice po ekranu. Budući da se kreću samo gore i dole, potrebno je prilagođavati samo Y vrijednost koordinata pločica.

```
• pygame.quit()
• sys.exit()
•
• pressed = pygame.key.get_pressed()
• if pressed[K_w] :
•     leftPaddleY -= 2
• elif pressed[K_s] :
•     leftPaddleY += 2
• if pressed[K_UP] :
•     rightPaddleY -= 2
• elif pressed[K_DOWN] :
•     rightPaddleY += 2
```

Kada pokrenete ovaj kod trebali bi da možete pokretati pločice po ekranu koristeći strelice za gore i dole (K_UP i K_DOWN) igrač broj 2 koristi tastere S i W (K_s , K_w). Možete odrediti i svoje tsatere na tastaturi.

Primijetit ćete da u ovom trenutku pločice nisu ograničene. To jest, mogu se kretati gore i dole na neodređeno i otići s ekrana. To ćemo ograničiti uz pomoć if pitalica gdje ćemo provjeravati tokom cijele igre da li je pločica došla do vrha ili dna ekrana.

```

• # Processing
•
•     if leftPaddleY < 0 :
•         leftPaddleY = 0
•     if leftPaddleY > WINDOW_HEIGHT - PADDLEHEIGHT :
•         leftPaddleY = WINDOW_HEIGHT - PADDLEHEIGHT
•     if rightPaddleY < 0 :
•         rightPaddleY = 0
•     if rightPaddleY > WINDOW_HEIGHT - PADDLEHEIGHT :
•         rightPaddleY = WINDOW_HEIGHT - PADDLEHEIGHT
•
•     leftPaddleRect = pygame.Rect(PADDLEINSET, leftPaddleY, PADDLEWIDTH,
PADDLEHEIGHT)
•

```

Ono što radimo ovdje je otkrivanje ako koordinate pločica idu dalje od vrha ili dna prozora i čineći ih jednakima vrhu ili dna prozora ako je to slučaj. Mogli smo se testirati ako su koordinate jednake vrhu ili dnu prozora, ali to nije tako sigurno. Ako promijenite brzinu kretanja pločica, koordinate nikada neće tačno izjednačiti rub zaslona (jedan korak bit će neposredno prije ivice i sljedeći korak neposredno nakon ivice).

Posljednji dio kontrola kretanja loptice .

Prvo ćemo dodati dvije varijable za praćenje kretanja lopte na X i Y osi. Dodajte ove varijable unutar glavne funkcije :

```

• looping = True
•
•     leftPaddleY = 50
•     rightPaddleY = 50
•     ballX = WINDOW_WIDTH // 2
•     ballY = WINDOW_HEIGHT // 2
•     ballXMomentum = 1
•     ballYMomentum = 1

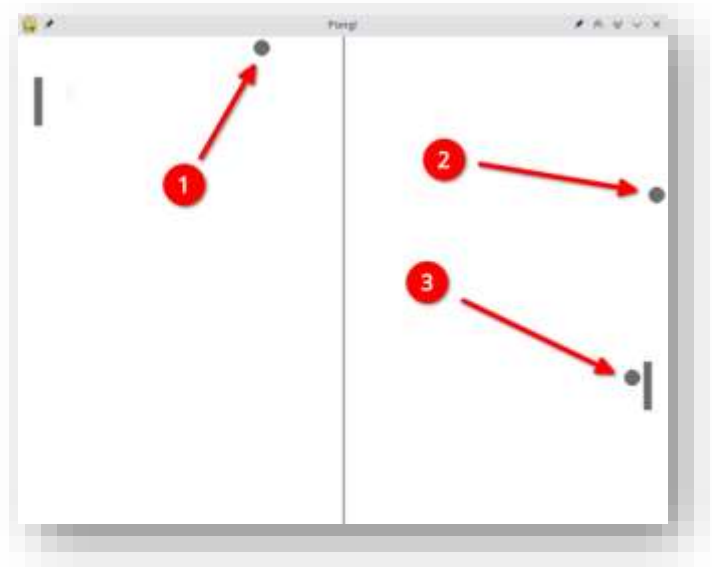
```

Zatim moramo razmisliti o različitim situacijama u koje lopta može naći i kako bi trebala reagirati na njih. Postoje tri događaja koja zahtijevaju da se nešto dogodi :

1. Lopta je pogodila vrh ili dno terena - u ovom slučaju lopati se odbija.
2. Lopta je pogodila bilo koji strani rub terena - igrač nije udario loptu sa svojim veslonom i tako su izgubili poen.
3. Lopta je pogodila pločicu - lopta se odbija u drugom smjeru.

Sada možemo progresivno dodati kod za rješavanje svakog od tih scenarija. Postoji malo matematike i logike koji su uključeni u otkrivanje ovih. Mogli smo samo teško kodirati vrijednosti, ali korištenje proračuna omogućuje nam da kasnije lako podešavamo vrijednosti kao što su pločice, prozor, lopte itd. bez potrebe za vraćanjem i ponovnim izračunavanjem svih tih vrijednosti. To je malo više rada, ali to će vam uštedjeti vrijeme i trud kasnije.

Nastavit ćemo s našim metodičnim pristupom, izgradnji i testiranjem svakog scenarija prije nego što prije nego što prije krenemo na sljedeći. Također ćemo ih kodirati kako bismo ih najlakše implementirali do najteži za pisanje koda.



Najlakši događaji za otkrivanje su vrh ili dno terena. Sve što trebamo učiniti je provjeriti je li promjenjiva loptica manja od radijusa lopte (varijabla BALLSIZE) ili je veća od WINDOW_HEIGHT (predstavljajući dno terena) - radijus lopte (BALLSIZE).

```

• if rightPaddleY > WINDOW_HEIGHT - PADDLEHEIGHT :
•     rightPaddleY = WINDOW_HEIGHT - PADDLEHEIGHT
•
• if ballY < BALLSIZE : # ball has hit the top
•     ballYMomentum = 1
• if ballY > WINDOW_HEIGHT - BALLSIZE : # ball has hit the bottom
•     ballYMomentum = -1
•
• leftPaddleRect = pygame.Rect(PADDLEINSET, leftPaddleY, PADDLEWIDTH,
PADDLEHEIGHT)

```

Loptica se trenutno ne pomjera. Popravimo to ažuriranjem koordinata lopte X i Y dodavanjem odgovarajućeg zamaha.

```

• leftPaddleRect = pygame.Rect(PADDLEINSET, leftPaddleY, PADDLEWIDTH,
PADDLEHEIGHT)
• rightPaddleRect = pygame.Rect(WINDOW_WIDTH - PADDLEINSET -
PADDLEWIDTH, rightPaddleY, PADDLEWIDTH, PADDLEHEIGHT)
•
• ballX = ballX + ballXMomentum

```

- `ballY = ballY + ballYMomentum`
-
- `WINDOW.fill(BACKGROUND)`
- `pygame.draw.line(WINDOW, ELEMENTCOLOUR, ((WINDOW_WIDTH // 2) - 1, 0), ((WINDOW_WIDTH // 2) - 1, WINDOW_HEIGHT), 2)`

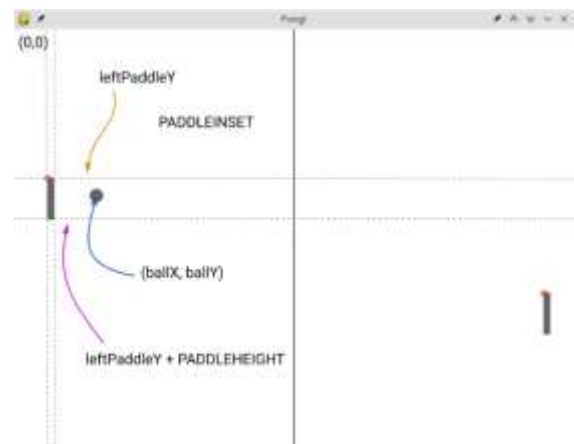
Pokrenite program i vidite šta se dešava.

Sljedeće je najlakše je ako je lopta prošla pored bilo koje od pločica i pogodila bočnu ivicu. To je sličan proces, međutim, radimo s `ballX` umjesto `ballY`. Također ćemo ponovo postaviti loptu u središte terena ako se dogodi bilo koji od ovih događaja.

- `if ballY > WINDOW_HEIGHT - BALLSIZE : # ball has hit the bottom`
- `ballYMomentum = -1`
-
- `if ballX <= BALLSIZE : # Left player loses`
- `ballX = WINDOW_WIDTH // 2`
- `ballY = WINDOW_HEIGHT // 2`
- `ballYMomentum = 1`
- `ballXMomentum = 1`
- `if ballX >= WINDOW_WIDTH - BALLSIZE : # Right player loses`
- `ballX = WINDOW_WIDTH // 2`
- `ballY = WINDOW_HEIGHT // 2`
- `ballYMomentum = 1`
- `ballXMomentum = -1`

- `leftPaddleRect = pygame.Rect(PADDLEINSET, leftPaddleY, PADDLEWIDTH, PADDLEHEIGHT)`
-

Ako sada pokrenete igru, trebali biste primijetiti da će lopta putovati do kraja prozora, a zatim se vratiti na sredinu terena i krenut će u suprotnom smjeru, iznova i iznova. Sada je sve što je ostalo otkriti da li je pločica pogodila loptu i odbila loptu u skladu s tim.



```

• ballYMomentum = 1
•   ballXMomentum = -1
•
•   if ballX <= PADDLEINSET + PADDLEWIDTH and ballX > PADDLEINSET : # work
out if left paddle has hit the ball
•       if leftPaddleY < ballY and leftPaddleY + PADDLEHEIGHT > ballY :
•           ballXMomentum = 1
•       if ballX >= WINDOW_WIDTH - PADDLEINSET - PADDLEWIDTH and ballX <
WINDOW_WIDTH - PADDLEINSET : # work out if right paddle has hit the ball
•           if rightPaddleY < ballY and rightPaddleY + PADDLEHEIGHT > ballY :
•               ballXMomentum = -1
•
•   leftPaddleRect = pygame.Rect(PADDLEINSET, leftPaddleY, PADDLEWIDTH,
PADDLEHEIGHT)

```

Gornji kod može se činiti pomalo zastrašujućim, ali ono što u suštini radimo je prvo provjeravanje ima li loptica X koordinat između prednjeg i stražnjeg dijela pločice, a ako se to dogodilo, onda provjeravamo je li loptica između vrha i dna pločice. Ako je to oboje, onda se mora odbiti od pločice.

Pokrenite igru i Uživajte 😊

Što možete pokušati izmijeniti u igri :

1. Promijeniti boje terena
2. Promijeniti brzinu kretanja i odbijanja loptice
3. Dodati zvučne efekte