Python i Pygame

Programiranje arkadnih igrica u Pythonu koristeći Pygame

Uvod u animaciju

The bouncing rectangle

- Za prvu animaciju ćemo izraditi program koji prikazuje bijeli četverokut koji se odbija od rubova ekrana
- Započet ćemo sa kosturom programa kojeg smo definirali u 4. dijelu, koji iscrtava prazni ekran
- Prvo, postavljamo boju ekrana na crno
  `screen.fill(BLACK)`
- Zatim crtamo četverokut kojeg ćemo animirati
  `pygame.draw.rect(screen, WHITE, [50, 50, 50, 50])`

  Gornji lijevi vrh je na koordinati (50,50)
  Četverokut je širine 50 i visine 50 pixela - kvadrat

Glavna petlja programa

- Pravokutnik se pomiče po ekranu, pa koordinate gornjeg lijevog vrha moramo izraziti pomoću varijabli
- Glavna petlja sada izgleda ovako:

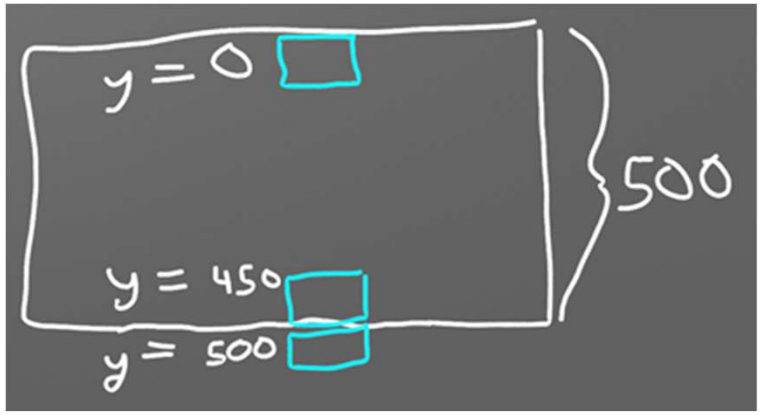
```
# Pocetna pozicija cetverokuta
rect_x = 50
rect_y = 50

# Brzina i smjer cetverokuta
rect_change_x = 5
rect_change_y = 5

# -------- Glavna petlja -----------
while done == False:
    for event in pygame.event.get(): # Uhvati akciju
        if event.type == pygame.QUIT: # Ako je kliknuo na close
            done = True # zastavica za izlazak iz petlje
    # Postavi pozadinu
    screen.fill(BLACK)
    # nacrtaj četverokut
    pygame.draw.rect(screen, WHITE, [rect_x, rect_y, 50, 50])
    # Promijeni koordinate gornjeg lijevog vrha
    rect_x += rect_change_x
    rect_y += rect_change_y
```

Odbijanje od ruba ekrana

- Dimenzije ekrana su postavljene na 500x500 pixela
- Kada x koordinata pravokutnika dođe do 450 potrebno je promijeniti smjer kretanja
- Također, kada y koordinata pravokutnika dođe do 450 potrebno je promijeniti smjer kretanja




```
# Bounce the rectangle if needed
if rect_y > 450 or rect_y < 0:
    rect_change_y = rect_change_y * -1
if rect_x > 450 or rect_x < 0:
    rect_change_x = rect_change_x * -1
```

Cijeli program

```
# Importiraj biblioteku funkcija Pygame
import pygame

# Inicijaliziraj game engine
pygame.init()


# Definiraj boje
BLACK = (0, 0, 0)
WHITE = (255, 255, 255)
GREEN = (0, 255, 0)
RED = ( 255, 0, 0)
BLUE = (0, 0, 255)



# Postavi veličinu prozora
size = (500, 500)
screen = pygame.display.set_mode(size)



# Varijabla koja označava kad završavamo program
done = False
# varijabla koja prati brzinu izvršavanja programa
clock = pygame.time.Clock()



# Pocetna pozicija cetverokuta
rect_x = 50
rect_y = 50

# Brzina i smjer cetverokuta
rect_change_x = 5
rect_change_y = 5
```

Cijeli program, nastavak

```
# Glavna petlja
while not done:
    # Petlja glavnog eventa
    for event in pygame.event.get():         # Hvatanje akcije igrača
        if event.type == pygame.QUIT: # Igrač je pritisnuo close window
            done = True # postavljamo varijablu za kraj programa
    # Postavi boju ekrana na bijelo
    screen.fill(BLACK)



    # Nacrtaj pravokutnik od (rect_x, rect_y) sirine 50 visine 50
    pygame.draw.rect(screen, WHITE, [rect_x, rect_y, 50, 50])



    # Promijeni koordinate gornjeg lijevog vrha
    rect_x += rect_change_x
    rect_y += rect_change_y



    # Odbijanje pravokutnika ako je potrebno
    if rect_y > 450 or rect_y < 0:
        rect_change_y = rect_change_y * -1
    if rect_x > 450 or rect_x < 0:
        rect_change_x = rect_change_x * -1

    # Ažuriranje ekrana
    pygame.display.flip()

    # Postavi limit na 60 FPS
    clock.tick(60)



# Po izlasku iz petlje zatvori ekran
pygame.quit()
```

Animacija snijega

- Ponovo koristimo kostur programa iz 4. dijela
- Za generiranje x,y koordinata koristit ćemo slučajne brojeve.
  Ova petlja crta 50 krugova na slučajnim lokacijama

```
for i in range(50):
    x = random.randrange(0, 499)
    y = random.randrange(0, 499)
    pygame.draw.circle(screen, WHITE, [x, y], 2)
```

- Jednom generirane pahuljice želimo zadržati na istom mjestu, zato generirane lokacije prije petlje spremimo u niz

```
for i in range(50):
    x = random.randrange(0, 499)
    y = random.randrange(0, 499)
    snow_list.append([x,y])
```

- Ovime kreiramo niz koordinata x,y – niz u nizu, koji izgleda kao npr.:

```
snow_list = [[34,10], [10,50], [20,10]]
```

Animacija snijega, nastavak

- Ako želimo odvojeno pristupiti x i y koordinatama svake pahuljice, to radimo na slijedeći način:
  x koordinata pahuljice_x -> [pahuljica_x][0]
  y koordinata pahuljice_x -> [pahuljica_x][1]
  ili u našem primjeru
  >>> print(snow_list[0][0])
  34
  >>> print(snow_list[0][1])
  10
- U glavnoj petlji ćemo koristiti petlju for za ispis svih pahuljica iz liste, i pomak pahuljica za jedan pixel dolje

```
for i in range(len(snow_list)):
    pygame.draw.circle(screen, WHITE, snow_list[i], 2)
    snow_list[i][1]+=1
```

Animacija snijega, nastavak

- Ako želimo detektirati kada pahuljica dođe do dna ekrana, to ćemo učiniti slijedećom if petljom

```
# Ako je pahuljica prosla dno ekrana
if snow_list[i][1] > 499:
    # Postavi y koordinatu pahuljice ispod ekrana
    y = random.randrange(-50, -10)
    snow_list[i][1] = y
    # Postavi x koordinatu pahuljice
    x = random.randrange(0, 499)
    snow_list[i][0] = x
```

Cijeli program

```
# Importiraj biblioteke funkcija Pygame i random
import pygame
import random

# Inicijaliziraj game engine
pygame.init()


# Definiraj boje
BLACK = (0, 0, 0)
WHITE = (255, 255, 255)

# Postavi veličinu prozora
size = (500, 500)
screen = pygame.display.set_mode(size)



# Varijabla koja označava kad završavamo program
done = False
# varijabla koja prati brzinu izvršavanja programa
clock = pygame.time.Clock()



# Kreiraj praznu listu pahuljica
snow_list = []


# Generiraj 50 pahuljica sa random x,y koordinatama
for i in range(50):
    x = random.randrange(0, 499)
    y = random.randrange(0, 499)
    snow_list.append([x, y])
```

Cijeli program, nastavak

```
# Glavna petlja
while not done:
    # Petlja glavnog eventa
    for event in pygame.event.get():         # Hvatanje akcije igrača
        if event.type == pygame.QUIT: # Igrač je pritisnuo close window
            done = True # postavljamo varijablu za kraj programa
    # Postavi boju ekrana na bijelo
    screen.fill(BLACK)



    # Procesiraj svaku pahuljicu iz liste
    for i in range(len(snow_list)):
        # Nacrtaj pahuljicu
        pygame.draw.circle(screen, WHITE, snow_list[i], 2)

        # Pomakni pahuljicu za 1 pixel dolje
        snow_list[i][1] += 1



        # Ako je pahuljica prosla dno ekrana
        if snow_list[i][1] > 499:
            # Postavi y koordinatu pahuljice ispod ekrana
            y = random.randrange(-50, -10)
            snow_list[i][1] = y
            # Postavi x koordinatu pahuljice
            x = random.randrange(0, 499)
            snow_list[i][0] = x

    # Ažuriranje ekrana
    pygame.display.flip()

    # Postavi limit na 20 FPS
    clock.tick(20)

# Po izlasku iz petlje zatvori ekran
pygame.quit()
```

Kviz

- Slijedi link ispod:

  http://programarcadegames.com/quiz/quiz.php?file=animation&lang=en

Kviz odgovori

- P1: In the bouncing rectangle program, if rect_change_x is positive and rect_change_y is negative, which way will the rectangle travel?
  - Up
  - Up and left
  - Down and left
  - Down and right
  - Up and right
- P2: In the bouncing rectangle program, if rect_change_x is zero and rect_change_y is positive, which way will the rectangle travel?
  - Right
  - Up
  - Down and left
  - Up and left
  - Down