Programowanie
po 9 wykładzie

Andrzej Giniewicz

10.05.2024

Dziś zrobimy Ponga! Stwórz wirtualne środowisko i zainstaluj bibliotekę `pygame`. *Na początku każdej sekcji jest link do kodu w formie, w której łatwiej go skopiować w całości, jeśli zgubisz się w modyfikacjach pomiędzy wersjami lub jeśli nie chce Ci się przepisywać.* Zaczynamy!

Przez pierwsze 5 sekcji postaraj się podążać za kodem i zrozumieć to, co jest w nim zawarte.

1 Wersja 1: okienko z możliwością zamknięcia

`http://prac.im.pwr.edu.pl/~giniew/doku.php?id=rok2324:letni:prog:v1`

Zaczynamy od zaimportowania biblioteki PyGame. Posłuży nam ona do zrobienia gry. Pamiętaj, że pracujemy z kodem, który ma swoje okienko, więc należy go uruchamiać jako skrypt.

```
import pygame
```

Następnie ustawiamy kilka stałych globalnych — szerokość okna, wysokość okna, docelową liczbę klatek na sekundę, którą chcemy wyświetlać oraz kolor tła okna. Kolor tła podajemy jako krotkę trzech wartości od 0 do 255 reprezentującą kolor w palecie RGB.

```
OKNO_SZER = 800
OKNO_WYS = 600
FPS = 60

TŁO = (0, 0, 0)
```

Kolejne cztery linie inicjalizują bibliotekę oraz tworzą okno o zadanych rozmiarach i tytułem „Przykład Ponga”. Tworzymy też zegarek, który posłuży do sterowania prędkością wyświetlania.

```
pygame.init()
```

```
okienko = pygame.display.set_mode((OKNO_SZER, OKNO_WYS), 0, 32)
pygame.display.set_caption("Przykład Ponga")
zegarek = pygame.time.Clock()
```

Po stworzeniu okna mamy pętlę, która działa tak długo, jak długo włączona będzie gra. Gdy ustawimy zmienną `graj` na fałsz, Python wyjdzie z pętli. A co robimy wewnątrz pętli? Mamy trzy etapy — najpierw sprawdzamy wszystkie zdarzenia. Zdarzenia mogą być różne — wciśnięcia klawiszy, kliknięcia myszki, ale również zamknięcie okna. To jedyny typ zdarzenia, jaki na razie obsłużymy. Po obsłużeniu wszystkich zdarzeń rysujemy elementy gry — na razie tylko tło. Po narysowaniu elementów odświeżamy okienko, aby wyświetliło nową zawartość. Następnie czekamy tyle milisekund, aby uzyskać stałą prędkość gry w klatkach na sekundę. Zadba o to metoda `tick` klasy `Clock`.

```
graj = True
while graj:
    for zdarzenie in pygame.event.get():
        if zdarzenie.type == pygame.QUIT:
            graj = False

    okienko.fill(TŁO)

    pygame.display.update()
    zegarek.tick(FPS)
```

Gdy Python zakończy pętlę, uruchamiamy funkcję `quit` z modułu `pygame`, aby zwolnić wszystkie zasoby zadeklarowane przez `init` i zamknąć okienko.

```
pygame.quit()
```

To wszystko! Mamy okienko z czarnym tłem, nazwą i obsługą krzyżyka do zamykania. Niby niewiele, ale to już pierwszy krok.

2 Wersja 2: rysujemy paletki i piłkę

`http://prac.im.pwr.edu.pl/~giniew/doku.php?id=rok2324:letni:prog:v2`

Dodamy teraz dwie paletki oraz piłkę. Paletki będą reprezentowane przez obiekt `Rect`, któremu musimy ustalić położenie lewego górnego rogu oraz szerokość i wysokość. Podamy je za pomocą stałych. Podobnie potraktujemy piłkę, której promień musimy znać podczas rysowania. Piłka na początku będzie na środku ekranu a paletki na środku jego krawędzi bocznych.

Dodajmy potrzebne stałe obok pozostałych stałych.

```
PALETKA_WYS = 80
PALETKA_SZER = 20
PIŁKA_R = 10

PALETKA = (255, 255, 255)
PIŁKA = (255, 255, 255)
```

Są to odpowiednio wysokość i szerokość paletki, promień piłki oraz kolor paletek i piłki. Stwórzmy teraz dwa obiekty, które posłużą do reprezentacji graczy. Będą to obiekty `Rect` o odpowiednim położeniu i rozmiarach. Piłkę będziemy reprezentować przez współrzędne. Po inicjalizacji, ale przed pętlą, dodajmy kod.

```
gracz1 = pygame.Rect(0, OKNO_WYS//2-PALETKA_WYS//2, PALETKA_SZER, PALETKA_WYS)
gracz2 = pygame.Rect(OKNO_SZER-PALETKA_SZER, OKNO_WYS//2-PALETKA_WYS//2,
                     PALETKA_SZER, PALETKA_WYS)
piłka_x = OKNO_SZER//2
piłka_y = OKNO_WYS//2
```

Spróbuj przeanalizować wzory wyznaczające lewy górny róg paletek. Czy dostrzegasz, skąd się biorą poszczególne operacje arytmetyczne? Spróbuj narysować okno i paletki na kartce i określić ich pozycje. Po ustaleniu pozycji obiektów pozostaje je narysować. Robimy to po wypełnieniu okienka tłem, ale przed odświeżeniem ekranu.

```
pygame.draw.rect(okienko, PALETKA, gracz1)
pygame.draw.rect(okienko, PALETKA, gracz2)
pygame.draw.circle(okienko, PIŁKA, (piłka_x, piłka_y), PIŁKA_R)
```

Jako ćwiczenie sprawdź dokumentację funkcji `pygame.draw.circle`[1] i spróbuj zidentyfikować wszystkie użyte parametry.

Uruchom kod jako skrypt i spróbuj zobaczyć, czy wszystkie elementy narysowały się we właściwych miejscach.

3 Wersja 3: sterowanie paletkami

`http://prac.im.pwr.edu.pl/~giniew/doku.php?id=rok2324:letni:prog:v3`

Pora wprawić paletki w ruch. Stworzymy nowe zmienne, dla każdego gracza, które będą zawierały wektor prędkości względem osi pionowej. Jeśli gracz wciśnie przycisk oznaczający ruch w górę, zmienna ta ustawi się na minus pewną wartość, aby paletka zmniejszała swoje położenie i poruszała się w górę. Gdy zwolni ten przycisk, wektor zwiększymy, aby przestała poruszać się w górę. Analogicznie zrobimy dla klawisza oznaczającego „w dół". Stwórzmy odpowiednie zmienne.

[1] `https://www.pygame.org/docs/ref/draw.html#pygame.draw.circle`

```
SPEED = 5
gracz1_speed = 0
gracz2_speed = 0
```

Stała `SPEED` oznacza prędkość paletek w pikselach na klatkę. Początkowe prędkości paletek to zero, zatem stoją w bezruchu. Nasze paletki poruszać będziemy klawiszami „W” oraz „S” dla gracza 1 oraz klawiszami strzałek w górę i dół dla gracza 2. Dodajmy obsługę klawiszy do zdarzeń.

```
elif zdarzenie.type == pygame.KEYDOWN:
    if zdarzenie.key == pygame.K_w:
        gracz1_speed -= SPEED
    elif zdarzenie.key == pygame.K_s:
        gracz1_speed += SPEED
    elif zdarzenie.key == pygame.K_UP:
        gracz2_speed -= SPEED
    elif zdarzenie.key == pygame.K_DOWN:
        gracz2_speed += SPEED
elif zdarzenie.type == pygame.KEYUP:
    if zdarzenie.key == pygame.K_w:
        gracz1_speed += SPEED
    elif zdarzenie.key == pygame.K_s:
        gracz1_speed -= SPEED
    elif zdarzenie.key == pygame.K_UP:
        gracz2_speed += SPEED
    elif zdarzenie.key == pygame.K_DOWN:
        gracz2_speed -= SPEED
```

Po obsłudze zdarzeń, ale przed rysowaniem, musimy jeszcze przesunąć paletkę. Robimy to raz na klatkę, nie raz na zdarzenie.

```
gracz1.y += gracz1_speed
if gracz1.y < 0:
    gracz1.y = 0
elif gracz1.y > OKNO_WYS-PALETKA_WYS:
    gracz1.y = OKNO_WYS-PALETKA_WYS
gracz2.y += gracz2_speed
if gracz2.y < 0:
    gracz2.y = 0
elif gracz2.y > OKNO_WYS-PALETKA_WYS:
    gracz2.y = OKNO_WYS-PALETKA_WYS
```

Zauważmy, że dbamy tutaj, aby paletki nie wypadły poza ekran. Te zmiany wystarczają, aby poruszać paletkami. W kolejnym kroku zaczniemy poruszać piłką.

4 Wersja 4: fizyka piłki — ruch i kolizje

http://prac.im.pwr.edu.pl/~giniew/doku.php?id=rok2324:letni:prog:v4

Wykorzystamy tym razem kolejne zmienne globalne oraz funkcję `nowa_piłka`, która ustawi piłkę na środku i zagra ją w losowym kierunku oraz pod losowym kątem. Będziemy potrzebowali do tego kilku funkcji z biblioteki standardowej Pythona.

```
from math import sin, cos, pi
from random import choice, uniform
```

Zanim przejdziesz dalej, sprawdź w dokumentacji Pythona, co robią funkcje, które widzisz pierwszy raz.

Przygotujmy teraz dwie zmienne na wektor przesunięcia piłki oraz funkcję do losowego serwowania.

```
piłka_dx = 0
piłka_dy = 0

def nowa_piłka():
    global piłka_x, piłka_y, piłka_dx, piłka_dy
    phi = uniform(-pi/3, pi/3)
    piłka_x = OKNO_SZER//2
    piłka_y = OKNO_WYS//2
    piłka_dx = SPEED * choice([-1, 1]) * cos(phi)
    piłka_dy = SPEED * sin(phi)

nowa_piłka()
```

Zwróć uwagę na słowo kluczowe `global`, dzięki któremu funkcja zmienia wartości zdefiniowane globalnie. W większości sytuacji nie jest to dobrą praktyką, ale w przypadku gier, które same w sobie bazują na modyfikacji stanu przez na przykład poruszanie się elementów po ekranie, jest to często stosowany zabieg. Musimy go jednak stosować ostrożnie i tylko po to, aby nie duplikować kodu w wielu miejscach. Tutaj pozwala nam to uniknąć tego samego kodu do losowania w kilku miejscach. Dzięki temu, jeśli zmienimy go raz, ta sama zmiana będzie działała i w innych miejscach. Jak działa `nowa_piłka`? Ustawia piłkę na środku i losuje kąt z zakresu od $-\frac{\pi}{3}$ do $\frac{\pi}{3}$ co odpowiada nachyleniu $\pm 60^\circ$ do osi OX. Dodatkowo dorzucamy losowy kierunek mnożąc wektor w kierunku osi OX razy losową wartość ze zbioru $\{-1, 1\}$. Funkcja zaraz po zdefiniowaniu jest użyta. Tuż przed rysowaniem pozostaje dodać przesunięcie piłki.

```
piłka_x += piłka_dx
piłka_y += piłka_dy
```

Kolejnym krokiem będzie obsługa kolizji. Piłka może się odbyć od poziomych ścian, odbić od paletki lub wypaść poza ekran, jeśli jej nie złapiemy. Zwróćmy uwagę, że piłka porusza się po kilka pikseli naraz, nie wystarczy więc zmienić kierunku, gdy dotknie bandy. Musimy również obliczyć, ile wystaje poza ramkę ekranu i skorygować jej pozycję tak, aby powstało złudzenie ruchu jednostajnego. Obsłużmy najpierw ściany poziome.

```
if piłka_y < PIŁKA_R:
    wystaje = PIŁKA_R-piłka_y
    piłka_y = PIŁKA_R+wystaje
    piłka_dy *= -1
elif piłka_y > OKNO_WYS-PIŁKA_R:
    wystaje = piłka_y - (OKNO_WYS-PIŁKA_R)
    piłka_y = OKNO_WYS-PIŁKA_R-wystaje
    piłka_dy *= -1
```

Zauważmy, że piłka powinna była się odbić, jeśli współrzędna Y jej środka jest mniejsza od promienia lub większa od wysokości okna pomniejszonej o promień. Jeśli nie widzisz, skąd się biorą odpowiednie przekształcenia, polecam narysować piłkę i okno na kartce i zaznaczyć współrzędne. W podobny sposób możemy ustalić, czy udało się odbić piłkę.

```
if piłka_x < PIŁKA_R+PALETKA_SZER:
    if gracz1.y <= piłka_y <= gracz1.y+PALETKA_WYS:
        wystaje = PIŁKA_R+PALETKA_SZER - piłka_x
        piłka_x = PIŁKA_R+PALETKA_SZER+wystaje
        piłka_dx *= -1
    else:
        nowa_piłka()
elif piłka_x > OKNO_SZER-PALETKA_SZER-PIŁKA_R:
    if gracz2.y <= piłka_y <= gracz2.y+PALETKA_WYS:
        wystaje = piłka_x - (OKNO_SZER-PALETKA_SZER-PIŁKA_R)
        piłka_x = OKNO_SZER-PALETKA_SZER-PIŁKA_R - wystaje
        piłka_dx *= -1
    else:
        nowa_piłka()
```

Tutaj oprócz wypadnięcia poza zasięg paletki, sprawdzamy jeszcze, czy współrzędna Y jest w zasięgu paletki. Jeśli nie, serwujemy nową piłkę. Nasza gra już prawie działa, choć gracze muszą samodzielnie liczyć punkty. Ostatnim krokiem, który zrobimy, będzie dodanie zliczania punktów. Ruszamy!

5 Wersja 5: punkty i zwycięzca

http://prac.im.pwr.edu.pl/~giniew/doku.php?id=rok2324:letni:prog:v5

Czekają nas dwa wyzwania — zliczanie punktów i wyświetlanie punktów. Ustalimy maksymalną liczbę punktów, po których kończy się gra. Zwycięzcę zasygnalizujemy zmianą koloru. Do wyświetlania tekstu posłużymy się fontem systemowym, który ma większość osób — popularnym krojem pisma Arial. Najpierw zacznijmy od stałych: odpowiedzialną za liczbę punktów oraz kolory. Tekst będzie szary, a gdy gra się zakończy, zwycięzca będzie pokolorowany na zielono a przegrany na czerwono.

```
STOP_PLAY = 5


TEKST = (127, 127, 127)
WINNER = (127, 255, 127)
LOOSER = (255, 127, 127)
```

Zaraz po inicjalizacji gry, powinniśmy załadować font.

```
font = pygame.font.SysFont('Arial', 64)
```

Tuż przed pętlą ustalamy też obecną liczbę punktów i tworzymy obrazek cyfry „o" dla każdego z graczy. Rysowanie tekstu jest dwuetapowe. Najpierw przelewamy tekst na obrazek, który potem już narysowany umieszczamy na ekranie. Operacja tworzenia obrazka z tekstem jest pracochłonna, dlatego nie robimy tego co klatkę, tylko wtedy, gdy tekst się zmienia.

```
gracz1_wynik = 0
gracz2_wynik = 0
gracz1_text = font.render("0", True, TEKST)
gracz2_text = font.render("0", True, TEKST)
```

Dwa kolejne fragmenty kodu dodajemy przed „nową piłką", gdy wypadnie obok paletki. Dla sytuacji, gdy wypada po lewej stronie ekranu dodajemy

```
gracz2_wynik += 1
if gracz2_wynik >= STOP_PLAY:
    gracz1_text = font.render(str(gracz1_wynik), True, LOOSER)
    gracz2_text = font.render(str(gracz2_wynik), True, WINNER)
    SPEED = 0
    gracz1_speed = 0
    gracz2_speed = 0
else:
    gracz2_text = font.render(str(gracz2_wynik), True, TEKST)
```

a gdy po prawej

```
gracz1_wynik += 1
if gracz1_wynik >= STOP_PLAY:
    gracz1_text = font.render(str(gracz1_wynik), True, WINNER)
    gracz2_text = font.render(str(gracz2_wynik), True, LOOSER)
    SPEED = 0
```

```
        gracz1_speed = 0
        gracz2_speed = 0
    else:
        gracz1_text = font.render(str(gracz1_wynik), True, TEKST)
```

W kodzie powyżej zwiększamy o 1 liczbę punktów i sprawdzamy, czy gracz wygrał dzięki temu punktowi. Jeśli tak, odświeżamy tekst odpowiedzialny za obu graczy, by nadać im odpowiednie kolory. Dodatkowo zerujemy wszystkie prędkości, aby gra się zatrzymała. Jeśli nikt nie wygrał, odświeżamy jedynie obrazek przedstawiający liczbę punktów gracza, który właśnie zdobył punkt. Ostatnią rzeczą jest narysowanie punktów na ekranie. Najlepiej zrobić to zaraz po narysowaniu tła, ale przed paletką, aby cyfra nie zasłaniała piłki.

```
okienko.blit(gracz1_text, (40, 40))
okienko.blit(gracz2_text, (OKNO_SZER-40-gracz2_text.get_rect().width, 40))
```

Dlaczego w kodzie we współrzędnej X dla drugiego gracza dopisane jest tajemnicze wyrażenie `gracz2_text.get_rect().width`? Jest to zmierzona szerokość rysunku z tekstem. Dzięki temu tekst umieszczony jest o 40 pikseli od prawej krawędzi okna — aby wyznaczyć współrzędną lewego górnego rogu ekranu, musimy odjąć szerokość rysunku.