

# PyGame - Gra w Ponga

**Opis implementacji:** Używając biblioteki PyGame oraz języka Python, stworzymy prostą grę Pong.

**Autorzy:** Łukasz Zarzecki, Robert Bednarz

**Czas realizacji:** 90 min

**Poziom trudności:** Poziom 3

Biblioteka PyGame ułatwia tworzenie aplikacji multimedialnych, w tym gier. Poniższy scenariusz prezentuje implementację prostej gry Pong.

## Spis treści

PyGame – Gra w Ponga.....	1
I. Zmienne i plansza gry.....	2
Kod I.1.....	2
II. Obiekty graficzne.....	2
Kod II.1.....	2
III. Wyświetlanie tekstu.....	3
Kod III.1.....	3
IV. Główna pętla programu.....	4
Kod IV.1.....	4
Kod IV.2.....	4
POĆWICZ SAM.....	7

## I. Zmienne i plansza gry

Tworzymy plik **pong.py** w terminalu lub w wybranym edytorze i zaczynamy od zdefiniowania zmiennych określających właściwości obiektów w naszej grze.

Kod I.1

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import pygame, sys
from pygame.locals import *

# Przygotowanie zmiennych opisujących okno gry oraz obiekty gry i ich właściwości (paletki, piłeczka)
# Inicjacja modułu i obiektów Pygame'a

# inicjacja modułu pygame
pygame.init()

# liczba klatek na sekundę
FPS = 30
# obiekt zegara, który pozwala śledzić czas i odświeżać
fpsClock = pygame.time.Clock()

# szerokość i wysokość okna gry
OKNOGRY_SZER = 800
OKNOGRY_WYS = 400

# przygotowanie powierzchni do rysowania, czyli inicjacja okna gry
OKNOGRY = pygame.display.set_mode((OKNOGRY_SZER, OKNOGRY_WYS), 0, 32)
# tytuł okna gry
pygame.display.set_caption('Prosty Pong')

# kolory wykorzystywane w grze, których składowe RGB zapisane są w tuplach
LT_BLUE = (230, 255, 255)
WHITE = (255, 255, 255)
RED = (255, 0, 0)
GREEN = (0, 255, 0)
BLUE = (0, 0, 255)
```

W instrukcji `pygame.display.set_mode()` inicjalizujemy okno gry o rozmiarach 800x400 pikseli i 32 bitowej głębi kolorów. Tworzymy w ten sposób powierzchnię główną do rysowania zapisaną w zmiennej `OKNOGRY`. Definiujemy również kolory w formacie RGB (Red, Green, Blue) podając składowe poszczególnych kanałów w tuplach, np. `(0, 0, 255)`.

## II. Obiekty graficzne

W dalszej kolejności zajmiemy się określeniem właściwości i inicjalizacją paetek i piłki.

Kod II.1

```
# szerokość, wysokość i pozycja paetek
PALETKA_SZER = 100
PALETKA_WYS = 20

# Inicjacja PALETEK:
# utworzenie powierzchni dla obrazka, wypełnienie jej kolorem,
# pobranie prostokątnego obszaru obrazka i ustawienie go na wstępnej pozycji

PALETKA_1_POZ = (350, 360) # początkowa pozycja paletki gracza
paletka1_obr = pygame.Surface([PALETKA_SZER, PALETKA_WYS])
paletka1_obr.fill(BLUE)
paletka1_prost = paletka1_obr.get_rect()
```

```
paletka1_prost.x = PALETKA_1_POZ[0]
paletka1_prost.y = PALETKA_1_POZ[1]

PALETKA_2_POZ = (350, 20) # początkowa pozycja paletki komputera
paletka2_obr = pygame.Surface([PALETKA_SZER, PALETKA_WYS])
paletka2_obr.fill(RED)
paletka2_prost = paletka2_obr.get_rect()
paletka2_prost.x = PALETKA_2_POZ[0]
paletka2_prost.y = PALETKA_2_POZ[1]

# szybkość paletki 1 (AI – ang. artificial intelligence, sztuczna inteligencja), czyli komputera
AI_PREDKOSC = 3

# Inicjacja PIŁKI
# szerokość, wysokość, prędkość pozioma (x) i pionowa (y) PIŁKI
# utworzenie powierzchni dla piłki, narysowanie na niej koła, ustawienie pozycji początkowej
PILKA_SZER = 20
PILKA_WYS = 20
PILKA_PREDKOSC_X = 6
PILKA_PREDKOSC_Y = 6
pilka_obr = pygame.Surface([PILKA_SZER, PILKA_WYS], pygame.SRCALPHA, 32).convert_alpha()
pygame.draw.ellipse(pilka_obr, GREEN, [0, 0, PILKA_SZER, PILKA_WYS])
pilka_prost = pilka_obr.get_rect()
pilka_prost.x = OKNOGRY_SZER/2
pilka_prost.y = OKNOGRY_WYS/2
```

Schemat dodawania obiektów graficznych jest prosty. Po określeniu wymiarów obiektu (szerokości i wysokości), tworzymy powierzchnię (np. `pygame.Surface([PALETKA_SZER, PALETKA_WYS])`), którą wypełniamy odpowiednim kolorem (`.fill()`). W przypadku piłki do metody `Surface()` przekazujemy dodatkowe argumenty (`pygame.SRCALPHA`) umożliwiające uzyskanie powierzchni z przezroczystymi pikselami (z kanałem alpha), na której rysujemy koło (`pygame.draw.ellipse()`) o podanym kolorze, środku i rozmiarach. W kolejnym kroku pobieramy prostokąt (np. `paletka1_prost = paletka1_obr.get_rect()`) zajmowany przez obiekt, za pomocą którego łatwiej ustawić wstępne położenie obiektu, a później nim manipulować (właściwości `.x` i `.y` obiektu **Rect** zwróconego przez metodę `.get_rect()`).

### III. Wyświetlanie tekstu

W grze chcemy wyświetlać punkty zdobywane przez graczy. Dopisujemy więc poniższy kod:

#### Kod III.1

```
# Rysowanie komunikatów tekstowych
# ustawienie początkowych wartości liczników punktów
# utworzenie obiektu czcionki z podanego pliku o podanym rozmiarze
GRACZ_1_PKT = '0'
GRACZ_2_PKT = '0'
fontObj = pygame.font.Font('freesansbold.ttf', 64)

# funkcje wyświetlające punkty gracza
# tworzą nowy obrazek z tekstem, pobierają prostokątny obszar obrazka,
# pozycjonują go i rysują w oknie gry

def drukuj_punkty_p1():
    tekst_obr1 = fontObj.render(GRACZ_1_PKT, True, (0,0,0))
    tekst_prost1 = tekst_obr1.get_rect()
    tekst_prost1.center = (OKNOGRY_SZER/2, OKNOGRY_WYS*0.75)
    OKNOGRY.blit(tekst_obr1, tekst_prost1)

def drukuj_punkty_p2():
```

```
tekst_obr2 = fontObj.render(GRACZ_2_PKT, True, (0,0,0))
tekst_prost2 = tekst_obr2.get_rect()
tekst_prost2.center = (OKNOGRY_SZER/2, OKNOGRY_WYS/4)
OKNOGRY.blit(tekst_obr2, tekst_prost2)
```

Po zdefiniowaniu zmiennych przechowujących punkty graczy, tworzymy obiekt czcionki z podanego pliku (`pygame.font.Font()`)<sup>1</sup>. Następnie definiujemy funkcje, których zadaniem jest rysowanie punktacji graczy. Na początku tworzą one nową powierzchnię z punktacją gracza (`.render()`), pobierają jej prostokąt (`.get_rect()`), pozycjonują go (`.center()`) i rysują na głównej powierzchni gry (`.blit()`).

## IV. Główna pętla programu

Programy interaktywne, w tym gry, reagujące na działania użytkownika, takie jak ruchy czy kliknięcia myszą, działają w pętli, której zadaniem jest:

1. przechwycenie i obsługa działań użytkownika, czyli tzw. zdarzeń (ruchy, kliknięcia myszą, naciśnięcie klawiszy),
2. aktualizacja stanu gry (przesunięcia elementów, aktualizacja planszy),
3. aktualizacja wyświetlanego okna (narysowanie nowego stanu gry).

Dopisujemy więc do kodu główną pętlę wraz z obsługą zdarzeń:

**Kod IV.1**

```
# pętla główna programu
while True:
    # obsługa zdarzeń generowanych przez gracza
    for event in pygame.event.get():
        # przechwycić zamknięcie okna
        if event.type == QUIT:
            pygame.quit()
            sys.exit()
        # przechwycić ruch myszy
        if event.type == MOUSEMOTION:
            # pobierz współrzędne x, y kursora myszy
            myszaX, myszaY = event.pos

            # przesunięcie paletki gracza
            przesuniecie = myszaX - (PALETKA_SZER/2)

            # jeżeli wykraczamy poza okno gry w prawo
            if przesuniecie > OKNOGRY_SZER - PALETKA_SZER:
                przesuniecie = OKNOGRY_SZER - PALETKA_SZER
            # jeżeli wykraczamy poza okno gry w lewo
            if przesuniecie < 0:
                przesuniecie = 0
            paletka1_prost.x = przesuniecie
```

W obrębie głównej pętli programu pętla `for` odczytuje kolejne zdarzenia zwracane przez metodę `pygame.event.get()`. Jak widać, za pomocą instrukcji warunkowych (`if event.type ==`) obsługujemy wydarzenia typu QUIT, czyli zakończenie aplikacji, oraz MOUSEMOTION, a więc ruch myszy. W tym drugim przypadku pobieramy współrzędne kursora (`event.pos`) i obliczamy przesunięcie myszy w poziomie. Kolejne instrukcje uniemożliwiają wyjście paletki gracza poza okno gry.

---

<sup>1</sup> Plik wykorzystywany do wyświetlania tekstu (**freesansbold.ttf**) musi znaleźć się w katalogu ze skryptem.

Do pętli głównej musimy dopisać jeszcze kod kontrolujący paletkę komputera, piłkę i jej interakcje ze ścianami okna gry oraz paletkami, a także rysujący poszczególne obiekty:

#### Kod IV.2

```
# AI (jak gra komputer)
# jeżeli środek piłki jest większy niż środek paletki AI
# przesun w prawo paletkę z ustawioną prędkością
if pilka_prost.centerx > paletka2_prost.centerx:
    paletka2_prost.x += AI_PREDKOSC
# w przeciwnym wypadku przesun w lewo
elif pilka_prost.centerx < paletka2_prost.centerx:
    paletka2_prost.x -= AI_PREDKOSC

# przesun piłkę po zdarzeniu
pilka_prost.x += PILKA_PREDKOSC_X
pilka_prost.y += PILKA_PREDKOSC_Y

# Sprawdzamy kolizje piłki z obiektami
# Ściany: jeżeli piłka wykracza poza okno z lewej lub prawej strony odbij ją od ściany
if pilka_prost.right >= OKNOGRY_SZER:
    PILKA_PREDKOSC_X *= -1
if pilka_prost.left <= 0:
    PILKA_PREDKOSC_X *= -1

# Piłka i paletki

# Jeżeli piłka dotknie paletki, skieruj ją w przeciwną stronę
if pilka_prost.colliderect(paletka1_prost):
    PILKA_PREDKOSC_Y *= -1
    # uwzględnij nachodzenie paletki na piłkę (przysłonięcie)
    pilka_prost.bottom = paletka1_prost.top

if pilka_prost.colliderect(paletka2_prost):
    PILKA_PREDKOSC_Y *= -1
    # uwzględnij nachodzenie paletki na piłkę (przysłonięcie)
    pilka_prost.top = paletka2_prost.bottom

# jeżeli piłka wyszła poza pole gry u góry lub z dołu ustaw domyślną pozycję piłki
# i przypisz punkt odpowiedniemu graczowi
if pilka_prost.top <= 0:
    pilka_prost.x = OKNOGRY_SZER/2
    pilka_prost.y = OKNOGRY_WYS/2
    GRACZ_1_PKT = str(int(GRACZ_1_PKT)+1)

if pilka_prost.bottom >= OKNOGRY_WYS:
    pilka_prost.x = OKNOGRY_SZER/2
    pilka_prost.y = OKNOGRY_WYS/2
    GRACZ_2_PKT = str(int(GRACZ_2_PKT)+1)

# Rysowanie obiektów
OKNOGRY.fill(LT_BLUE) # kolor okna gry

drukuj_punkty_p1() # wyświetl punkty komputera
drukuj_punkty_p2() # wyświetl punkty gracza

# narysuj w oknie gry paletki
OKNOGRY.blit(paletka1_obr, paletka1_prost)
OKNOGRY.blit(paletka2_obr, paletka2_prost)

# narysuj w oknie piłkę
OKNOGRY.blit(pilka_obr, pilka_prost)

# zaktualizuj okno i wyświetl
pygame.display.update()
```

```
# zaktualizuj zegar po narysowaniu obiektów  
fpsClock.tick(FPS)
```

#KONIEC

Komentarze w kodzie wyjaśniają kolejne czynności. Warto zwrócić uwagę na sposób odczytywania pozycji obiektów klasy `Rect` (prostokątów), czyli właściwości `.x`, `.y`, `.centerx`, `.right`, `.left`, `.top`, `.bottom`; oraz na sprawdzanie kolizji piłki z paletkami, czyli metodę `.collidirect()`. Ostatnie linie kodu rysują okno gry i obiekty (tekst z wynikami graczy, paletki i piłkę) ze zmienionymi właściwościami (liczba punktów, położenie). Funkcja `pygame.display.update()`, która musi być wykonywana na końcu rysowania, aktualizuje obraz gry na ekranie. Ostatnia linia natomiast (`fpsClock.tick()`) blokuje grę na 30 klatek na sekundę, aby nie działała tak szybko jak pozwala sprzęt, lecz ze stałą prędkością.

Grę możemy uruchomić poleceniem wpisanym w terminalu: `python pong.py`.

## Słownik

- **Klatki na sekundę (FPS)** – liczba klatek wyświetlanych w ciągu sekundy, czyli częstotliwość, z jaką statyczne obrazy pojawiają się na ekranie. Jest ona miarą płynności wyświetlania ruchomych obrazów.
- **Kanał alfa (ang. alpha channel)** – w grafice komputerowej jest kanałem, który definiuje przezroczyste obszary grafiki. Jest on zapisywany dodatkowo wewnątrz grafiki razem z trzema wartościami barw składowych RGB.
- **Inicjalizacja** – proces wstępnego przypisania wartości zmiennym i obiektom. Każdy obiekt jest inicjalizowany różnymi sposobami zależnie od swojego typu.
- **Iteracja** – czynność powtarzania (najczęściej wielokrotnego) tej samej instrukcji (albo wielu instrukcji) w pętli. Mianem iteracji określa się także operacje wykonywane wewnątrz takiej pętli.
- **Zdarzenie (ang. event)** – zapis zajścia w systemie komputerowym określonej sytuacji, np. poruszenie myszką, kliknięcie, naciśnięcie klawisza.
- **pygame.time.Clock()** – tworzy obiekt do śledzenia czasu; **.tick()** – kontroluje ile milisekund upłynęło od poprzedniego wywołania.
- **pygame.display.set\_mode()** – inicjuje okno lub ekran do wyświetlania, parametry: rozdzielczość w pikselach = (x,y), flagi, głębokość koloru.
- **pygame.display.set\_caption()** – ustawia tytuł okna, parametr: tekst tytułu.
- **pygame.Surface()** – obiekt reprezentujący dowolny obrazek (grafikę), który ma określoną rozdzielczość (szerokość i wysokość) oraz format pikseli (głębokość, przezroczystość); **SRCALPHA** – oznacza, że format pikseli będzie zawierał ustawienie alfa (przezroczystości); **.fill()** – wypełnia obrazek kolorem; **.get\_rect()** – zwraca prostokąt zawierający obrazek, czyli obiekt **Rect**; **.convert\_alpha()** – zmienia format pikseli, w tym przezroczystość; **.blit()** – rysuje jeden obrazek na drugim, parametry: źródło, cel.
- **pygame.draw.ellipse()** – rysuje okrągły kształt wewnątrz prostokąta, parametry: przestrzeń, kolor, prostokąt.
- **pygame.font.Font()** – tworzy obiekt czcionki z podanego pliku; **.render()** – tworzy nową powierzchnię z podanym tekstem, parametry: tekst, antyalias, kolor, tło.

- **pygame.event.get()** – pobiera zdarzenia z kolejki zdarzeń; **event.type()** – zwraca identyfikator SDL typu zdarzenia, np. KEYDOWN, KEYUP, MOUSEMOTION, MOUSEBUTTONDOWN, QUIT.
- **SDL** (Simple DirectMedia Layer) – międzyplatformowa biblioteka ułatwiająca tworzenie gier i programów multimedialnych.
- **pygame.Rect** – obiekt pygame przechowujący współrzędne prostokąta; **.centerx**, **.x**, **.y**, **.top**, **.bottom**, **.left**, **.right** – wirtualne własności obiektu prostokąta określające jego położenie; **.colliderect()** – metoda sprawdza czy dwa prostokąty nachodzą na siebie.

## **POĆWICZ SAM**

---

Zmodyfikuj właściwości obiektów (pałek, piłki) takie jak rozmiar, kolor, początkowa pozycja.

Zmień położenie pałek tak aby znalazły przy lewej i prawej krawędzi okna, wprowadź potrzebne zmiany w kodzie, aby umożliwić rozgrywkę.

Dodaj trzecią pałkę, która co jakiś czas będzie "przelatywać" przez środek planszy i zmieniać w przypadku kolizji tor i kolor piłki.