3. Prosta gra w Pythonie

Leszek Klich

3.1 Wstęp

Jedną z najważniejszych zalet języka Python jest z pewnością jego uniwersalność. Przy pomocy tego języka można zaprogramować stronę internetową, program na komputery biurkowe, skrypt administracyjny lub... grę. I właśnie tym tematem zajmiemy się w niniejszym rozdziale. Choć tworzenie gier jest procesem skomplikowanym, to przy wykorzystaniu odpowiednich narzędzi programowanie prostych gier planszowych, logicznych czy zręcznościowych wcale nie musi być trudne. Wystarczy sięgnąć po język Python i wybrać jedną z wielu darmowych bibliotek przeznaczonych do tworzenia gier. Dzięki temu, w bardzo krótkim czasie, można poznać podstawy pisania gier.

W tym rozdziale zapoznamy się z popularną biblioteką PyGame, która ułatwia i przyspiesza tworzenie gier. Dzięki wielu funkcjonalnościom oraz prostocie biblioteka wymaga jedynie podstawowych umiejętności z zakresu programowania w języku Python. W ramach rozdziału zaprogramujemy prostą grę typu „pong", w której będziemy odbijać piłeczkę przy pomocy paletki. Aby uatrakcyjnić grę zaprogramujemy także prostą punktację.

Norway grants
Projekt pt.: „MODELOWE ROZWIĄZANIA NA TRUDNE WYZWANIA – Plan Rozwoju Lokalnego i Instytucjonalnego Stalowej Woli" współfinansowany jest ze środków Norweskiego Mechanizmu Finansowego 2014-2021 (85%) oraz Budżetu Państwa (15%).
Wspólnie działamy na rzecz Europy zielonej, konkurencyjnej i sprzyjającej integracji społecznej.

3.2 Biblioteka PyGame

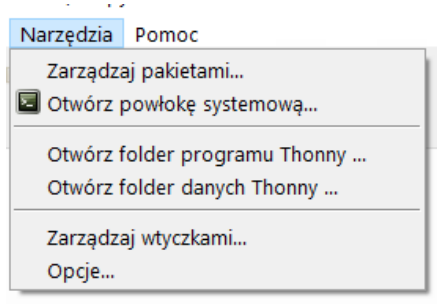
PyGame to popularna biblioteka dla języka Python, która oferuje zestaw funkcjonalności ułatwiających tworzenie gier. Jakie to ułatwienia? Na przykład interfejs graficzny, gotowe funkcje rysowania, matematyka wektorowa, wykrywanie kolizji pomiędzy obiektami, manipulacja pikselami, transformacje, filtry, obsługa czcionek czy klawiatury oraz myszki. Biblioteka ukrywa przed programistą mechanikę niskiego poziomu, oferując w zamian prostotę i szybkość programowania. Istotną cechą jest jej multiplatformowość, co oznacza, że przy jej pomocy można tworzyć gry dla systemów Windows, Linux czy MacOS. Co więcej, gry napisane w PyGame mogą działać także na telefonach i tabletach z systemem Android. Oto kilka podstawowych zalet biblioteki:

- Jest darmowa, a dzięki temu można przy jej pomocy tworzyć zarówno darmowe, jak i płatne gry.
- Używa języka C, a to oznacza, że działa bardzo szybko. Może także wykorzystać wielordzeniowe mikroprocesory.
- Prosta i łatwa w użyciu, a dzięki temu programowaniem gier mogą zająć się nawet dzieci.
- Bardzo popularna i dobrze udokumentowana.

Instalacja biblioteki

Aby zainstalować język Python w systemie Windows skorzystaj z Instrukcji instalacji oprogramowania Python, która została szczegółowo opisana w załączniku A.

Zanim rozpoczniemy pisanie gry musimy zainstalować bibliotekę PyGame. Aby to zrobić należy uruchomić środowisko Thonny i z menu **Narzędzia** wybrać polecenie **Otwórz powłokę systemową**, jak zostało to przedstawione na rysunku 3.1.



Rysunek 3.1: Uruchamianie powłoki systemowej w środowisku Thonny

Norway grants
Projekt pt.: „MODELOWE ROZWIĄZANIA NA TRUDNE WYZWANIA – Plan Rozwoju Lokalnego i Instytucjonalnego Stalowej Woli" współfinansowany jest ze środków Norweskiego Mechanizmu Finansowego 2014-2021 (85%) oraz Budżetu Państwa (15%).
Wspólnie działamy na rzecz Europy zielonej, konkurencyjnej i sprzyjającej integracji społecznej.

Bibliotekę zainstalujemy z poziomu powłoki systemowej, wpisując w oknie polecenie: **pip install pygame** <Enter>. Polecenie **pip** pobierze i zainstaluje bibliotekę (może to troszkę potrwać w zależności od szybkości łącza internetowego). Po zainstalowaniu biblioteki, proces przygotowania środowiska pracy jest zakończony i możemy rozpocząć naukę programowania.

3.3 Podstawy programowania gier

Jeśli została zainstalowana biblioteka PyGame, można przystąpić do napisania pierwszego programu. Na początek będzie to coś bardzo prostego. Otwórz środowisko programistyczne Thonny i wpisz w nim następujący kod:

```
import pygame

pygame.init()
```

Listing 3.1: Pierwszy program

Gdy upewnisz się, że nie popełniłeś błędu, musisz teraz uruchomić program. Istnieje wiele możliwości uruchomienia programu. Poniżej prezentuję kilka przykładów. Przetestuj je wszystkie i wybierz najwygodniejszy sposób.

Definicja 3.1 — Uruchamianie programu. Uruchomienie programu w środowisku Thonny może odbywać się na kilka sposobów. Przetestuj wszystkie poniższe sposoby uruchamiania programu i wybierz swój ulubiony:

- wciskając przycisk **F5**;
- przy pomocy menu **Uruchom** –> **Uruchom aktualny skrypt**;
- przy pomocy paska narzędzi, klikając myszką na przycisk strzałki.

Uwaga - jeśli po raz pierwszy uruchamiasz program, edytor Thonny zapyta o nazwę pliku i zapisze skrypt w wybranym folderze. Podaj zatem nazwę pliku (unikaj spacji oraz polskich znaków w nazwie pliku), wybierz folder, w którym powinien być zapisany program i zapisz program.

Na rysunku 3.2 widać wynik działania programu. Ekran podzielony jest na dwie sekcje. Górna sekcja to edytor kodu źródłowego, do którego wpisaliśmy kod naszego pierwszego programu. Druga (dolna) sekcja to powłoka, w której widoczny jest efekt działania programu, a raczej komunikaty, które dotyczą jego działania. Teraz omówimy wpisany kod. Pierwsza linia **import pygame** oznacza zaimportowanie

Norway grants
Projekt pt.: „MODELOWE ROZWIĄZANIA NA TRUDNE WYZWANIA – Plan Rozwoju Lokalnego i Instytucjonalnego Stalowej Woli" współfinansowany jest ze środków Norweskiego Mechanizmu Finansowego 2014-2021 (85%) oraz Budżetu Państwa (15%).
Wspólnie działamy na rzecz Europy zielonej, konkurencyjnej i sprzyjającej integracji społecznej.

Rysunek 3.2: Środowisko Thonny z uruchomionym programem w Thonny IDE

biblioteki PyGame. W ten sposób poinformowaliśmy Pythona, że w naszym programie chcemy wykorzystać bibliotekę. Kolejna linia to inicjalizacaja biblioteki, czyli wykonanie niezbędnych czynności, które sprawią, że biblioteka będzie poprawnie działać. Aby zakończyć działanie programu, kliknij myszką na przycisk **Stop** lub użyj kombinacji klawiszy **Ctrl**+**F2**.

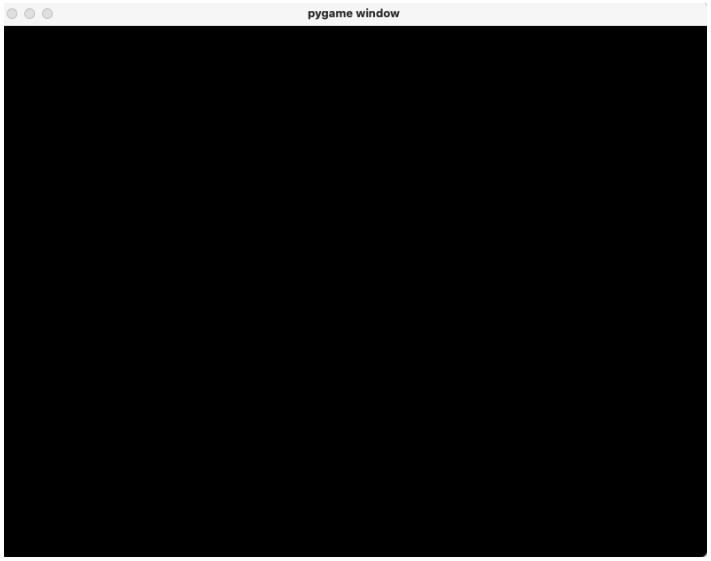
Jak na razie nasz program nie robi zbyt wiele, ale zaraz to zmienimy. Przede wszystkim spróbujemy wyświetlić okno naszej gry. W tym celu utworzymy sobie zmienną o nazwie **ekran** i przypiszemy do niej nowy obiekt okna: *ekran = pygame.display.set_mode((rozmiarX, rozmiarY))*. Jak widać metoda set_mode przyjmuje dwa argumenty, gdzie **rozmiarX** oznacza rozmiar naszego okna (gry) w poziomie, zaś zmienna **rozmiarY** jego wysokość. Zmodyfikujmy zatem program do następującej postaci:

```
import pygame

pygame.init()
ekran = pygame.display.set_mode((800, 600))
```

Listing 3.2: Tworzymy okno gry

Ponownie uruchom program. Jeśli powyższy kod został wpisany bezbłędnie, powinno wyświetlić się okno naszej gry, jak widać na rysunku 3.3. Aby zakończyć działanie programu, kliknij myszką na przycisk **Stop** lub użyj kombinacji klawiszy **Ctrl**+**F2**. Możesz spróbować zmodyfikować rozmiar okna i dostosować jego wielkość do własnego

Norway grants
Projekt pt.: „MODELOWE ROZWIĄZANIA NA TRUDNE WYZWANIA – Plan Rozwoju Lokalnego i Instytucjonalnego Stalowej Woli" współfinansowany jest ze środków Norweskiego Mechanizmu Finansowego 2014-2021 (85%) oraz Budżetu Państwa (15%).
Wspólnie działamy na rzecz Europy zielonej, konkurencyjnej i sprzyjającej integracji społecznej.



Rysunek 3.3: Okno gry napisane przy pomocy biblioteki PyGame

ekranu. Pamiętaj jednak, że istnieje wiele różnych monitorów o różnych rozdzielczościach i ustawienie zbyt dużego rozmiaru może spowodować, że gra się nie wyświetli. Właśnie dlatego użyliśmy dość niskich parametrów rozmiaru. W dalszej części zmodyfikujemy naszą grę, aby uruchamiała się na pełnym ekranie, niezależnie od parametrów monitorów!

Spójrzmy jeszcze raz na nowo utworzone okno gry z rysunku 3.3. Na belce tytułowej widnieje domyślnie wygenerowany tytuł **pygame window**. Aby go zmienić i dodać własny tytuł, np. z nazwą gry, wystarczy dodać nową linię programu i ustawić właściwość okna przy pomocy polecenia *pygame.display.set_ caption("Prosta gra PONG")*. Zatem teraz, kod naszej gry będzie wyglądał następująco:

```
import pygame

pygame.init()

ekran = pygame.display.set_mode((800, 600))
pygame.display.set_caption("Prosta gra PONG")
```

Listing 3.3: Dodajemy tytuł do okna gry

Ponownie uruchom program i zwróć uwagę, czy tytuł okna zawiera prawidłowy tekst. W ten sposób przygotowaliśmy okno gry i poprawnie je nazwaliśmy. Teraz zajmiemy się bardzo istotnym tematem: **obsługą zdarzeń**.

Norway grants
Projekt pt.: „MODELOWE ROZWIĄZANIA NA TRUDNE WYZWANIA – Plan Rozwoju Lokalnego i Instytucjonalnego Stalowej Woli" współfinansowany jest ze środków Norweskiego Mechanizmu Finansowego 2014-2021 (85%) oraz Budżetu Państwa (15%).
Wspólnie działamy na rzecz Europy zielonej, konkurencyjnej i sprzyjającej integracji społecznej.

Jak można zauważyć, uruchamiając naszą grę, otworzone okno nie odpowiadało na zamknięcie i nie zamykało się, gdy próbowaliśmy to zrobić. Dopiero zatrzymanie programu powodowało, że okno zostało zamknięte. Przyczyną tego jest właśnie brak zaprogramowanej reakcji na zdarzenia. Wyjaśnijmy najpierw, co to są zdarzenia.

Definicja 3.2 — Zdarzenia (Events). Termin związany z programowaniem, opartym na zdarzeniach. Wszystko, co dzieje się podczas działania dowolnego programu, tak naprawdę opiera się na generowaniu zdarzeń. Dla przykładu - podczas poruszania myszką, system operacyjny generuje zdarzenie, które zawiera sam fakt poruszenia myszką, ale także zwraca aktualną pozycję wskaźnika myszy. W przypadku, gdy naciśniesz klawisz na klawiaturze, generowane jest zdarzenie, informujące, że właśnie naciśnięto klawisz. Albo gdy klikniesz myszką przycisk w dowolnym programie, generowane jest zdarzenie o tym fakcie.

Ze zdarzeniami wiąże się temat obsługi zdarzeń. Oznacza to, że aby przechwycić występujące zdarzenie, należy go „wyłapać" i obsłużyć, czyli zaimplementować obsługę zdarzenia. Nasza gra również będzie odczytywać zdarzenia związane z oknem i stanem gry i podejmować odpowiednie działania. Dla przykładu - jeśli użytkownik wciśnie przycisk strzałki w lewo, gra przechwyci tę informację (zdarzenie) i odpowiednio przekieruje obiekt paletki w lewo. W przypadku, gdy użytkownik chce zakończyć grę, kliknie przycisk zamknięcia okna, wówczas gra powinna zakończyć działanie.

gra.py *

```
1  import pygame
2
3  pygame.init()
4
5
6  ekran = pygame.display.set_mode((800, 600))
7  pygame.display.set_caption("Prosta gra PONG")
8
9  while True:
10     for event in pygame.event.get():
11         if event.type == pygame.QUIT:
12             exit(0)
```

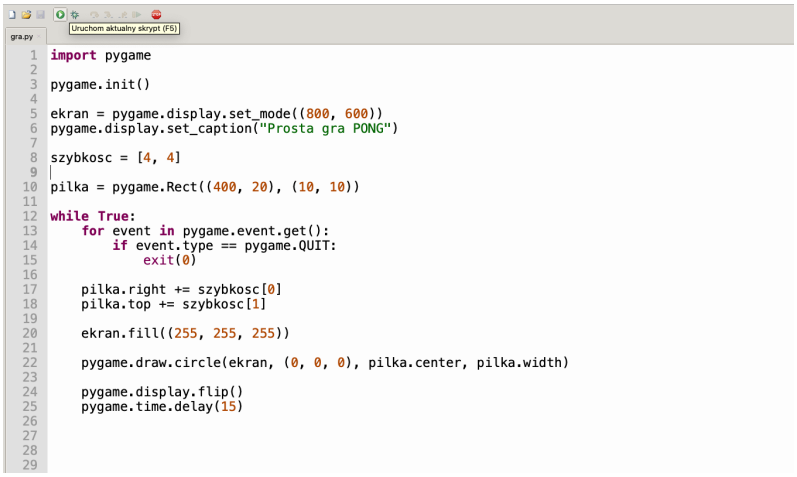
Rysunek 3.4: Obsługa zdarzenia zamknięcia okna gry

Zaprogramujmy zatem pętlę, która będzie zbierać wszystkie zdarzenia i reagować wyłącznie na zdarzenie zakończenia gry (czyli okna). Wprowadź dodatkowy kod do naszej gry (rys. 3.4). Jak widać, wpro-

Norway grants
Projekt pt.: „MODELOWE ROZWIĄZANIA NA TRUDNE WYZWANIA – Plan Rozwoju Lokalnego i Instytucjonalnego Stalowej Woli" współfinansowany jest ze środków Norweskiego Mechanizmu Finansowego 2014-2021 (85%) oraz Budżetu Państwa (15%).
Wspólnie działamy na rzecz Europy zielonej, konkurencyjnej i sprzyjającej integracji społecznej.

wadziliśmy niekończącą się pętlę, w której pobieramy listę zdarzeń gry (wiersz 10). Teraz sprawdzamy wszystkie zdarzenia w kolejce i przeszukujemy je pod kątem wystąpienia zdarzenia **QUIT**, czyli zakończenia gry (wiersz 11). Jeśli takie zdarzenie nastąpi, kończymy program (wiersz 12). Spróbuj uruchomić program ponownie i zamknąć go, klikając krzyżyk zakończenia w oknie. Tym razem program zakończył się poprawnie, co oznacza, że przechwytywanie zdarzeń zadziałało! Na razie zajmiemy się budową pola gry, zaś do zdarzeń wrócimy niebawem.

Teraz zaprogramujemy piłkę i wprawimy ją w ruch. Spójrz na obrazek 3.5 i wprowadź do gry dodatkowe wiersze z obrazka.



Rysunek 3.5: Kod tworzący piłkę oraz jej podstawowy ruch

Jak widać, pojawiły się nowe wiersze. W wierszu 10 utworzyliśmy obiekt **Rect**, który jest charakterystyczny dla biblioteki PyGame. Przechowuje on współrzędne prostokątne. Dzięki temu możemy tworzyć obiekty typu Rect, oznaczające ich granice. W tym przypadku utworzyliśmy obiekt Rect, zaczynający się od X = 400 (oznacza środek ekranu - ponieważ cały ekran w poziomie = 800, to 800/2 = 400 oraz Y = 20, zaś obiekt będzie rozpoczynał się o 20 punktów od górnej granicy ekranu). Dodatkowo określiliśmy wielkość przestrzeni jako 10 punktów w poziomie oraz 10 punktów w pionie. Problem w tym, że samo zadeklarowanie nie zadziała, bowiem w wierszu 10 tylko zadeklarowaliśmy obiekt **pilka**. Należy go jeszcze narysować. I tym zajmuje się wiersz 22, umieszczony w pętli głównej, co oznacza, że będzie się ciągle uruchamiał. Prześledźmy teraz wiersze, które zostały dodane, lecz nie zostały opisane.

Norway grants
Projekt pt.: „MODELOWE ROZWIĄZANIA NA TRUDNE WYZWANIA – Plan Rozwoju Lokalnego i Instytucjonalnego Stalowej Woli" współfinansowany jest ze środków Norweskiego Mechanizmu Finansowego 2014-2021 (85%) oraz Budżetu Państwa (15%).
Wspólnie działamy na rzecz Europy zielonej, konkurencyjnej i sprzyjającej integracji społecznej.

- W wierszu 8 zadeklarowaliśmy listę o nazwie szybkość, zawierającą dwa parametry. Będziemy ich używać do kontrolowania szybkości przemieszczania obiektu **pilka**.
- Wiersz 17 i 18 odpowiada za przesunięcie obiektu **pilka** w prawy dolny róg oraz od góry w dół, z zadeklarowaną szybkością.
- W wierszu 20 czyścimy ekran, wypełniając go białym kolorem.
- Wiersz 22 odpowiada za narysowanie okręgu (piłeczki) na ekranie w czarnym kolorze (0, 0, 0). Center to punkt środkowy okręgu.

Dokładniejszego wyjaśnienia wymaga wiersz 24 oraz 25. Do wyświetlania ekranu używamy metody flip(), która daje nam możliwość, tak zwanego, podwójnego buforowania. Aby zrozumieć działanie tego popularnego mechanizmu w świecie gier, wystarczy wyobrazić sobie kartkę papieru. Ma ona dwie strony, zaś pokazujemy widzowi jedynie jedną ze stron. Drugą zaś w tym czasie rysujemy i gdy skończymy rysować, wykonujemy flip(), czyli odwracamy kartkę. Dzięki temu można wyeliminować lub zminimalizować wszelkie artefakty rysowania i obraz jest lepszy. Jeśli chodzi o wiersz 25, w którym użyliśmy opóźnienia o wartości 15 milisekund. Opóźnienie służy do spowolnienia naszej gry. Oczywiście możesz usunąć tę linię i spróbować manipulować prędkością piłeczki i płynnością gry zmieniając wartości zmiennej **szybkosc**. Praktyka jednak wykazuje, że czasami dodanie opóźnienia w postaci delay(), daje nam lepszą elastyczność wpływania na płynność działania gry. Czas zatem uruchomić naszą grę. Po uruchomieniu zobaczysz, że piłeczka upada z góry w dół kierując się do prawego rogu. Następnie piłka znika z ekranu i... nie dzieje się nic. Jest to zachowanie najzupełniej prawidłowe, ponieważ nie określiliśmy ograniczeń i warunków, które zadziałają, gdy piłeczka "wyjedzie" poza ekran. Spróbujemy to teraz naprawić.

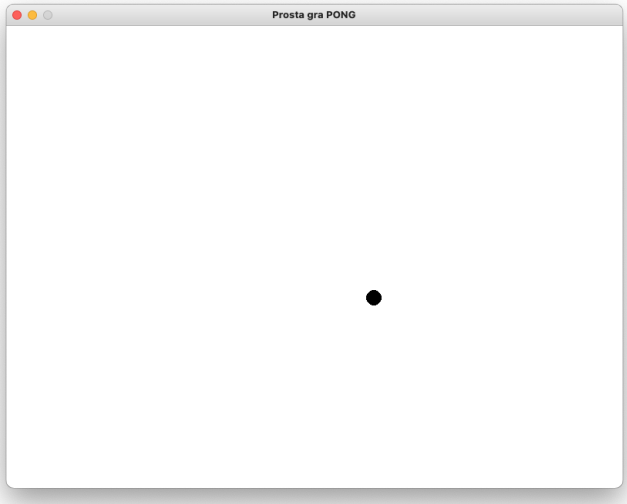
W poniższym listingu wprowadziliśmy ograniczenia dla przemieszczającej się piłki. Jeśli górna lub dolna część piłki znajduje się u góry (0) lub na dole (600) ekranu, to zmieni się jej kierunek ruchu w pionie.

```
import pygame

pygame.init()

ekran = pygame.display.set_mode((800, 600))
pygame.display.set_caption("Prosta gra PONG")

szybkosc = [4, 4]

pilka = pygame.Rect((400, 20, (10, 10))

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

Norway grants
Projekt pt.: „MODELOWE ROZWIĄZANIA NA TRUDNE WYZWANIA – Plan Rozwoju Lokalnego i Instytucjonalnego Stalowej Woli" współfinansowany jest ze środków Norweskiego Mechanizmu Finansowego 2014-2021 (85%) oraz Budżetu Państwa (15%).
Wspólnie działamy na rzecz Europy zielonej, konkurencyjnej i sprzyjającej integracji społecznej.

```
            exit(0)

    pilka.right += szybkosc[0]
    pilka.top += -szybkosc[1]

    if pilka.top <= 0 or pilka.bottom >= 600:
        szybkosc[1] = -szybkosc[0]

    if pilka.right >= 800:
        szybkosc[0] = -szybkosc[0]

    if pilka.left <= 0:
        szybkosc[0] = -szybkosc[0]

    ekran.fill((255, 255, 255))

    pygame.draw.circle(ekran, (0, 0, 0), pilka.center, pilka.
    width)

    pygame.display.flip()
    pygame.time.delay(15)
```

Listing 3.4: Implementacja odbijania się piłki od ścian okna



Rysunek 3.6: Niekończące się odbijanie piłki w oknie

Analogicznie w przypadku, gdy piłeczka zbliży się do prawej lub lewej części ekranu, zmieni kierunek na przeciwny. Dzięki temu otrzymujemy niekończącą się pętlę, odbijającej się piłeczki od ścian ekranu gry. Uruchom grę i sprawdź jak piłeczka wędruje po ekranie i odbija się od ścian.

Norway grants
Projekt pt.: „MODELOWE ROZWIĄZANIA NA TRUDNE WYZWANIA – Plan Rozwoju Lokalnego i Instytucjonalnego Stalowej Woli" współfinansowany jest ze środków Norweskiego Mechanizmu Finansowego 2014-2021 (85%) oraz Budżetu Państwa (15%).
Wspólnie działamy na rzecz Europy zielonej, konkurencyjnej i sprzyjającej integracji społecznej.

```
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            exit(0)

        # kontrola paletki przy pomocy myszki
        elif event.type == pygame.MOUSEMOTION:
            paletka.centerx = event.pos[0]
            # korekta paletki gdy wyjdzie poza okno gry
            if paletka.left < 0:
                paletka.left = 0
            elif paletka.right >= 800:
                paletka.right = 800

    # kontrola paletki przy pomocy strzałek lewo/prawo
    if event.type == pygame.KEYDOWN:
        if event.key == pygame.K_LEFT and paletka.left > 0:
            paletka.move_ip(-5, 0)
        elif event.key == pygame.K_RIGHT and paletka.right <
    800:
            paletka.move_ip(5, 0)
```

Listing 3.5: Obsługa zdarzeń myszki i klawiatury - sterowanie paletką

Uzyskany niewielkim nakładem pracy efekt, wydaje się być miły dla oka, lecz szybko się znudzi. Narysujmy więc paletkę, którą będzie można przesuwać przy pomocy myszki lub klawiszy <– oraz –>. Spójrz na powyższy listing, który zawiera zmodyfikowany kod naszej gry. Obejmuje on obsługę zdarzeń myszki, czyli przesuwa paletkę w prawo lub w lewo, lecz tylko w przypadku, gdy paletka nie znajduje się w skrajnie lewym lub skrajnie prawym obszarze okna gry. Przechwytujemy także zdarzenia wciśnięcia klawiszy strzałek - (K_LEFT oraz k_RIGHT). Do przesuwania obiektu paletki, po wciśnięciu odpowiedniego klawisza, użyliśmy metody **move_ip(x, y)**, która przyjmuje dwa argumenty (x oznacza przesunięcie w poziomie), zaś y oznacza przesunięcie w pionie. Jednocześnie przed przesunięciem kontrolujemy, czy paletka nie przekracza wartości minimalnej (0) i maksymalnej (800) okna gry w poziomie.

Po uruchomieniu gry piłeczka odbija się od ścian, zaś paletka reaguje na ruch myszką oraz klawisze strzałek. Jednak gdy piłka zderzy się z paletką, nie powoduje to jej odbicia. Zamiast tego, piłeczka „przelatuje" przez paletkę, ignorując zasady fizyki. Musimy zatem dopisać jeszcze obsługę kolizji piłeczki z paletką i zdecydować, co się wówczas powinno wydarzyć. Najpierw musimy zaprogramować, aby piłeczka odbiła się od paletki. W tym celu wykorzystamy metodę **colliderect()**, którą wykonamy na obiekcie **pilka**. Przyjmuje ona argument obiektu, z którym następuje kolizja.

Norway grants
Projekt pt.: „MODELOWE ROZWIĄZANIA NA TRUDNE WYZWANIA – Plan Rozwoju Lokalnego i Instytucjonalnego Stalowej Woli" współfinansowany jest ze środków Norweskiego Mechanizmu Finansowego 2014-2021 (85%) oraz Budżetu Państwa (15%).
Wspólnie działamy na rzecz Europy zielonej, konkurencyjnej i sprzyjającej integracji społecznej.

```
# obsłużymy zderzenie piłki z paletką
# na razie tylko odbijemy piłkę
if paletka.colliderect(pilka):
    szybkosc[1] = -szybkosc[1]
```

Rysunek 3.7: Obsługa kolizji piłeczki z paletką

Wprowadzenie jednego warunku, który został umieszczony na rysunku 3.7, spowoduje, że piłka będzie za każdym razem odbijać się od paletki w przypadku, gdy przy pomocy klawiszy lub myszki „trafimy" paletką w piłkę. Na obecnym etapie nasza gra jest już prawie gotowa. Jest jednak troszkę nudna, ponieważ gra jest niewrażliwa na błędy gracza. Z tego powodu, nawet wówczas, gdy piłeczka upadnie, nie ma to wpływu na dalszą grę. Dodajmy zatem efekt rywalizacji, czyli punkty. Ustalmy też proste zasady gry: za każdym poprawnym odbiciem piłeczki, gracz otrzyma jeden punkt. W przypadku, gdy piłeczka upadnie, punkt zostanie odjęty. Koniec gry następuje wówczas, gdy gracz posiada zero punktów i piłka ponownie upadnie. W tym przypadku gra zostanie zakończona.

```
# Piłka upadła - odejmij punkt lub zakończ grę
# gdy gracz nie posiada już żadnych punktów
elif pilka.bottom >= 600:
    if punkty > 0:
        punkty -= 1
    else:
        exit(0)

# obsłużymy zderzenie piłki z paletką
# jeśli paletka trafi w piłkę, dodajemy punkt
if paletka.colliderect(pilka):
    szybkosc[1] = -szybkosc[1]
    punkty += 1
```

Rysunek 3.8: Obsługa kolizji piłeczki z paletką

Na rysunku 3.8 znajduje się kod, który implementuje obsługę punktów zgodną z zasadami gry. Można już przetestować grę, jednak brakuje jeszcze jednego elementu - wyświetlania punktów. Oto gotowy kod naszej gry, który uwzględnia wyświetlanie zdobytych punktów:

```
import pygame

pygame.init()

ekran = pygame.display.set_mode((800, 600))
pygame.display.set_caption("Prosta gra PONG")

szybkosc = [4, 4]
punkty = 0

pilka = pygame.Rect((400, 20), (10, 10))
```

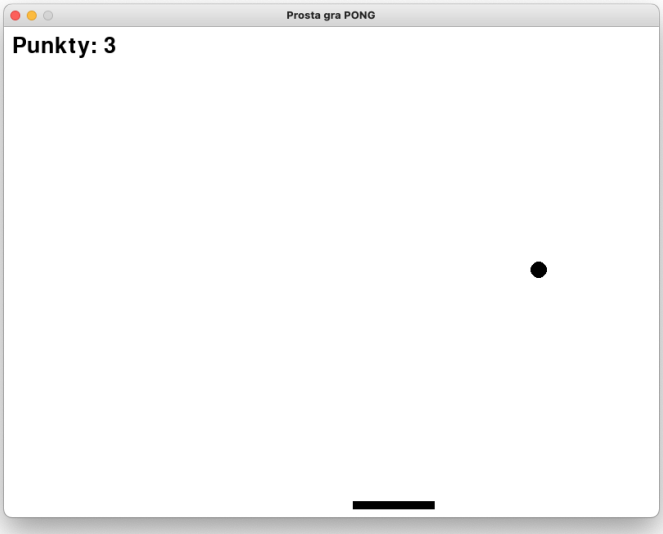
Norway grants
Projekt pt.: „MODELOWE ROZWIĄZANIA NA TRUDNE WYZWANIA – Plan Rozwoju Lokalnego i Instytucjonalnego Stalowej Woli" współfinansowany jest ze środków Norweskiego Mechanizmu Finansowego 2014-2021 (85%) oraz Budżetu Państwa (15%).
Wspólnie działamy na rzecz Europy zielonej, konkurencyjnej i sprzyjającej integracji społecznej.

```
paletka = pygame.Rect(400, 580, 100, 10)

font = pygame.font.Font(None, 40)

while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            exit(0)

        # kontrola paletki przy pomocy myszki
        elif event.type == pygame.MOUSEMOTION:
            paletka.centerx = event.pos[0]
            # korekta paletki gdy wyjdzie poza okno gry
            if paletka.left < 0:
                paletka.left = 0
            elif paletka.right >= 800:
                paletka.right = 800

    # kontrola paletki przy pomocy strzałek lewo/prawo
    if event.type == pygame.KEYDOWN:
        if event.key == pygame.K_LEFT and paletka.left > 0:
            paletka.move_ip(-5, 0)
        elif event.key == pygame.K_RIGHT and paletka.right <
    800:
            paletka.move_ip(5, 0)

    pilka.right += szybkosc[0]
    pilka.top += szybkosc[1]

    if pilka.top <= 0 or pilka.bottom >= 600:
        szybkosc[1] = -szybkosc[1]

    if pilka.right >= 800:
        szybkosc[0] = -szybkosc[0]

    if pilka.left <= 0:
        szybkosc[0] = -szybkosc[0]

    # piłka upadła - odejmij punkt lub zakończ grę
    # gdy gracz nie posiada już żadnych punktów
    elif pilka.bottom >= 600:
        if punkty > 0:
            punkty -= 1
        else:
            exit(0)

    # obsłużymy zderzenie piłki z paletką
    # jeśli paletka trafi w piłkę, dodajemy punkt
    if paletka.colliderect(pilka):
        szybkosc[1] = -szybkosc[1]
        punkty += 1

    ekran.fill((255, 255, 255))

    pygame.draw.circle(ekran, (0, 0, 0), pilka.center, pilka.
    width)
    pygame.draw.rect(ekran, (0, 0, 0), paletka)
    wynik = font.render("Punkty: " + str(punkty), True, (0,
    0, 0))
    ekran.blit(wynik, (10, 10))
```

Norway grants
Projekt pt.: „MODELOWE ROZWIĄZANIA NA TRUDNE WYZWANIA – Plan Rozwoju Lokalnego i Instytucjonalnego Stalowej Woli" współfinansowany jest ze środków Norweskiego Mechanizmu Finansowego 2014-2021 (85%) oraz Budżetu Państwa (15%).
Wspólnie działamy na rzecz Europy zielonej, konkurencyjnej i sprzyjającej integracji społecznej.

```
    pygame.display.flip()
    pygame.time.delay(15)
```

Omówienia wymaga metoda blit(), wykorzystana do nakładania obiektu punktów na powierzchnię gry. Służy ona do rysowania obiektów w oknie gry i ma postać *screen.blit(wynik,(x,y))*, gdzie **wynik** to obiekt, który ma być umieszczony na ekranie gry, zaś (x, y) to pozycja w oknie, od której zostanie on umieszczony w oknie gry.



Rysunek 3.9: Gotowa gra

3.4 Podsumowanie

W rozdziale zostały opisane podstawy wykorzystania biblioteki PyGame. W ramach rozdziału zostały zaprezentowane absolutne podstawy wykorzystania biblioteki. W praktyce, biblioteka jest na tyle rozbudowana, że pozwala na programowani o wiele bardziej skomplikowanych projektów. Celem tego rozdziału było jednak zaznajomienie czytelnika z absolutnymi podstawami. W ramach praktyki stworzyliśmy także prostą grę typu „pong", którą można rozbudować o muzykę, grafikę, tryb dwuosobowy, czy lepsze algorytmy toru piłeczki. Mam nadzieję, że zaprezentowane materiały będą zachętą do zgłębienia tematu programowania gier przy pomocy języka Python.

Norway grants
Projekt pt.: „MODELOWE ROZWIĄZANIA NA TRUDNE WYZWANIA – Plan Rozwoju Lokalnego i Instytucjonalnego Stalowej Woli" współfinansowany jest ze środków Norweskiego Mechanizmu Finansowego 2014-2021 (85%) oraz Budżetu Państwa (15%).
Wspólnie działamy na rzecz Europy zielonej, konkurencyjnej i sprzyjającej integracji społecznej.