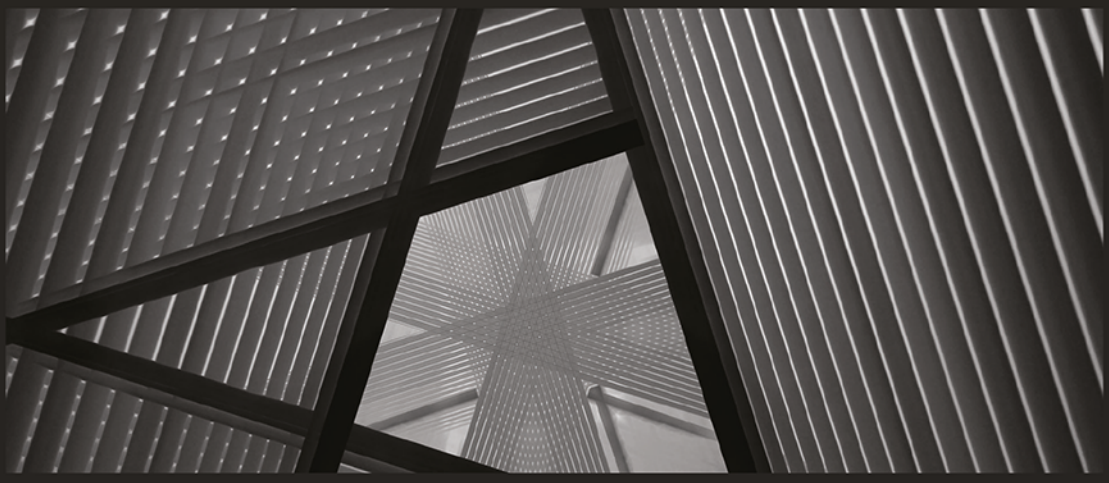
Helion

packt



Matematyka w programowaniu gier i grafice komputerowej

Tworzenie i renderowanie wirtualnych środowisk 3D oraz praca z nimi

PENNY DE BYL

Tytuł oryginału: Mathematics for Game Programming and Computer Graphics:
Explore the essential mathematics for creating, rendering,
and manipulating 3D virtual environments

Tłumaczenie: Anna Mizerska

ISBN: 978-83-289-0879-6

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
https://helion.pl/user/opinie/matopr
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:
https://ftp.helion.pl/przyklady/matopr.zip

Helion S.A.
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *https://helion.pl* (księgarnia internetowa, katalog książek)

Printed in Poland.

Spis treści

CZĘŚĆ 1. Podstawowe narzędzia

CZĘŚĆ 2. Podstawy trygonometrii

Rozdział

2 Zacznijmy rysować

Emocje towarzyszące uruchomieniu swojej pierwszej aplikacji graficznej są wyjątkowe. Można poczuć, że się coś osiągnęło, gdy własne pomysły na ułożenie pikseli urzeczywistniają się na ekranie. Mój pierwszy język programowania, jakiego się nauczyłam, to BASIC uruchamiany na komputerze Amstrad CPC664 i pomimo że pisanie kodu było dość czasochłonne, nic mnie tak nie cieszyło jak rysowanie kształtów i zmienianie kolorów na ekranie.

Obrazy rysowane na komputerze to koniec końców pojedyncze piksele (takie jak te, które rysowaliśmy w poprzednim rozdziale) o różnych kolorach wyświetlane na ekranie. Rysowanie wszystkich obiektów piksel po pikselu byłoby długim i żmudnym procesem. Tak były renderowane obrazy na ekranie w latach 50. ubiegłego wieku, ale teraz, ze znacznie bardziej zaawansowaną technologią, układem kartezjańskim i matematyką, możemy określić proste kształty i za ich pomocą tworzyć obrazy.

W tym rozdziale poznasz podstawowe koncepcje i kształty, na których opierają się wszystkie grafiki, i odkryjesz, jak pracować z nimi w języku Python i bibliotece Pygame. Chciałabym, abyś po tym rozdziale rozumiał te koncepcje i był w stanie je zastosować w praktyce, co będzie podstawą do dalszej nauki i kolejnych projektów.

W tym rozdziale poznasz podstawowe koncepcje przy okazji odkrywania następujących zagadnień:

- kolor,
- linie,
- tekst,
- wieloboki,
- obrazy rastrowe.

Wymagania techniczne

Ćwiczenia w tym rozdziale też używają edytora PyCharm i języka Python, tak samo skonfigurowanego jak w rozdziale 1., „Witaj, graficzne okno, czyli początek podróży”.

Pamiętaj, by kod z każdego rozdziału zapisywać osobno. Zanim przejdziesz do dalszej części tego rozdziału, upewnij się, że utworzyłeś dla niego nowy folder.

Rozwiązania ćwiczeń z tego rozdziału znajdziesz pod adresem *https://ftp.helion.pl/przyklady/matopr.zip*, w folderze *r02*.

Kolor

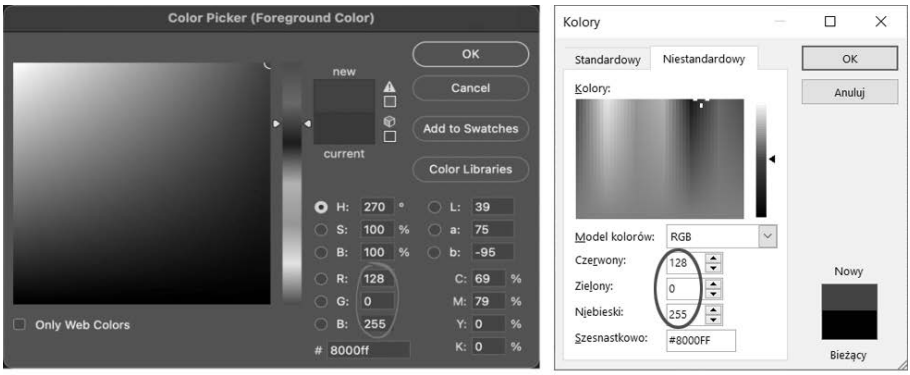
Grafika komputerowa z definicji jest wizualnym środkiem przekazu i jako taki opiera się na kolorze. **Przestrzeń barw** to zakres kolorów, które mogą być wypadkową połączenia podstawowych kolorów urządzenia. Na przykład wyobraź sobie, że malujesz obraz farbami olejnymi. Powiedzmy, że masz tubkę farby czerwonej, tubkę żółtej oraz tubkę niebieskiej i mieszasz je w każdych możliwych proporcjach. W ten sposób uzyskasz kolory, które będą przestrzenią barw dla tego zestawu trzech tubek. Jeśli Twój kolega zrobiłby to samo, ale miał nieco inne odcienie czerwonego, żółtego i niebieskiego, uzyskałby inny zestaw kolorów, unikatowy dla tego zestawu. Natomiast jeżeli obaj (lub oboje) chcielibyście uzyskać kolor zielony przez zmieszanie w takich samych proporcjach swoich odcieni niebieskiego i zielonego, uzyskalibyście kolor zielony złożony z 50% koloru niebieskiego i 50% koloru żółtego, ale ponieważ zaczęliście od nieco innych odcieni niebieskiego i żółtego, Wasze kolory zielone również miałyby różne odcienie.

To samo dotyczy ekranów komputerowych, drukarek, aparatów i wszystkiego z kolorowym wyświetlaczem lub drukowanego. Kolor, który na jednym monitorze wydaje się zielony, może wyglądać zupełnie inaczej na innym, a jeszcze inaczej po wydrukowaniu na papierze. Zakres kolorów, który można stworzyć przez mieszanie barw, nazywamy **skalą barw**. Wracając do przykładu z tubkami farb, możemy powiedzieć, że Ty i Twój kolega pracowaliście z różnymi skalami barw.

Aby okiełznać ten problem, stworzono wiele standardowych przestrzeni kolorów, które mówią, jak dany kolor powinien być pokazany. Najpopularniejszą przestrzenią barw jest **RGB** (skrót od angielskich nazw kolorów czerwonego, zielonego, niebieskiego — *red*, *green*, *blue*). Wszystkie kolory wyświetlane na ekranie komputera mogą być określone przez zmieszanie różnej ilości czerwonego, zielonego i niebieskiego.

Wartości każdego koloru są określone za pomocą wartości z jednego z dwóch zakresów, albo od 0 do 1, albo od 0 do 255. Dla zakresu od 0 do 1 wartość 0 oznacza brak koloru, a 1 maksymalne natężenie koloru. Biorąc pod uwagę, że kolor RGB jest przedstawiany za pomocą krotki, gdzie wartości odpowiadają ilości, odpowiednio, czerwonego, zielonego i niebieskiego, to (0,5, 0, 1) będzie oznaczać kolor z połową maksymalnego natężenia czerwonego, bez zielonego i z pełnym natężeniem niebieskiego. W podobny sposób określamy kolor w zakresie od 0 do 255, i tak ten sam kolor będzie przedstawiony za pomocą (128, 0, 255), gdzie połowa maksymalnej ilości czerwonego jest zmieszana z pełną ilością niebieskiego.

Prawdopodobnie już wcześniej mieszałeś kolory RGB, gdyż są dostępne w wielu programach, od edytorów tekstowych po graficzne. Na rysunku 2.1 pokazano narzędzia do wybierania kolorów w Adobe Photoshop i w Microsoft Word i zaznaczono miejsce, gdzie ustawia się wartości RGB.



Rysunek 2.1. Narzędzia do wyboru koloru z popularnych programów: (a) Adobe Photoshop i (b) Microsoft Word

Wskazówka

Gdy będziesz ustawiał kolor w kodzie, a nie będziesz pewien, jakie wartości RGB dobrać, możesz skorzystać z narzędzia do wybierania kolorów w dowolnym programie, który masz pod ręką. Jeżeli język programowania, w którym piszesz program, przyjmuje te wartości w zakresie od 0 do 1 zamiast od 0 do 255, po prostu podziel wartości w zapisie od 0 do 255 przez 255. Na przykład, aby przełożyć kolor określony jako (100, 50, 90) na zakres od 0 do 1, musisz wykonać następujące dzielenie (100:255, 50:255, 90:255) = (0,39, 0,20, 0,35).

Do dzieła!

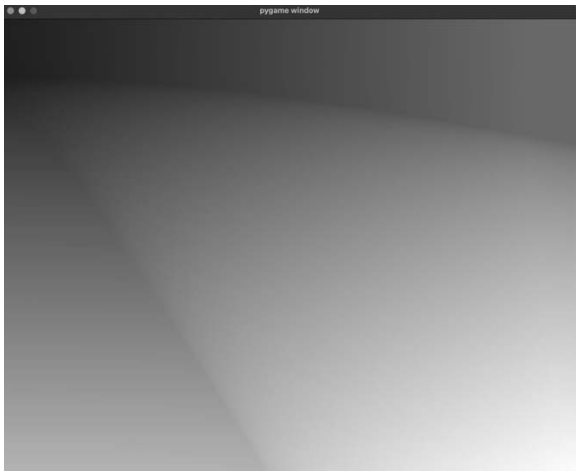
W tym ćwiczeniu zobaczysz, jak użyć kolorów w celu wyświetlenia tęczy z pikseli biegnącej wzdłuż ekranu.

1. Utwórz nowy plik Pythona o nazwie *RBGSpace.py* i dodaj podstawowy kod początkowy utworzony w rozdziale 1., „Witaj, graficzne okno, czyli początek podróży”, by otworzyć okno.
2. Pygame przyjmuje wartości kolorów z zakresu od 0 do 255. Choć możesz bardzo łatwo wyświetlić mieszankę trzech kolorów w dwuwymiarowym oknie, wciąż możesz odkrywać różne kolory przez dobór wartości z zakresu za pomocą programu takiego jak ten poniżej, w którym do bazowego kodu dodano pogrubione linie kodu.

```
import pygame

pygame.init()
screen_width = 1000
screen_height = 800
screen = pygame.display.set_mode((screen_width, screen_height))

done = False

while not done:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done = True
    for y in range(800):
        for x in range(1000):
            screen.set_at((x, y), pygame.Color(0, int(x/creen_width * 255),
                                               int(y/screen_height * 255)))

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

Metoda `color()` pobiera jako argumenty kanały, odpowiednio, czerwony, zielony i niebieski. Zauważysz, że wartości kanału czerwonego i zielonego będą różne, podczas gdy kanał niebieski będzie zawsze przybierał maksymalną wartość równą 255.

Rezultat pokazany na rysunku 2.2 przedstawia zakres kolorów uzyskanych przez zmieszanie wszystkich czerwieni i wszystkich zieleni przy maksymalnej ilości niebieskiego.



Rysunek 2.2. Zakres kolorów uzyskany przez zmieszanie wszystkich możliwych wartości kanałów czerwonego i zielonego przy maksymalnej wartości kanału niebieskiego

Twoja kolej!

Ćwiczenie A

Wyświetl zakres kolorów uzyskanych ze zmieszania wszystkich kanałów zielonych i niebieskich, bez czerwonego.

Jak już wspomniano wcześniej, grafika byłaby niczym bez koloru, W poprzednim ćwiczeniu kolorowaliśmy piksele. Jednak jest wiele innych graficznych obiektów, których podstawą jest piksel, dlatego przejdziemy teraz do kolejnego podstawowego obiektu graficznego — linii.

Linie

Rysowanie linii było niezwykle bliskie mojemu sercu, gdy uczyłam się grafiki komputerowej. Kiedy w szkole uczyłeś się o **układzie kartezjańskim**, prawdopodobnie nauczyłeś się również rysować proste linie i tego, że można je zapisać za pomocą następującego równania:

$$y = ax + b$$

gdzie a to **współczynnik kierunkowy** (lub **nachylenie**), a b to **miejsce przecięcia osi *y***. Współczynnik kierunkowy mówi, jak biegnie pionowa lub pozioma linia. Dla idealnej linii poziomej nachylenie wynosi 0, a idealna linia pionowa ma nieskończony współczynnik kierunkowy. Miejsce przecięcia osi y to wartość na osi y (gdzie $x = 0$), przez którą przechodzi dana linia.

Uwaga

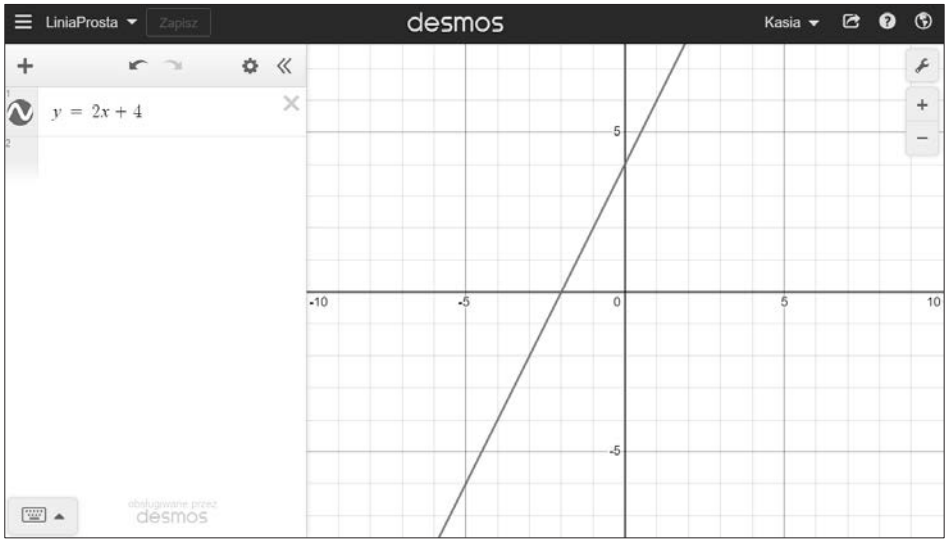
Jeśli chcesz przypomnieć sobie, jak przedstawiać linię matematycznie, przeczytaj artykuł na stronie *https://pl.wikipedia.org/wiki/Równanie_liniowe*.

Badanie równań liniowych ma więcej sensu, gdy możesz je zobaczyć na ekranie i sprawdzać rezultat po zmianie jego zmiennych. Na szczęście w internecie jest dostępne darmowe oprogramowanie, które może nam w tym pomóc.

Do dzieła!

W programie DESMOS możemy zwizualizować związek zmiennych w równaniu liniowym. W tym celu wykonaj następujące kroki:

1. Wejdź na stronę *https://www.desmos.com/calculator?lang=pl*.
2. W polu równania widocznym na rysunku 2.3 wpisz `y = 2x + 4`, by zobaczyć wykres tej linii.



Rysunek 2.3. Kreślenie prostej linii na podstawie równania w programie DESMOS

W równaniu liczba 2 to nachylenie (oznacza, jak stroma jest linia). Im mniejsza wartość, tym linia jest bardziej płaska (pozioma).

3. Aby linia była bardziej pozioma, zmień wartość `2` na `1`. Wartość 1 przełoży się na linię nachyloną pod kątem 45 stopni. Możesz zejść jeszcze niżej, jeśli chcesz, wystarczy, że wpiszesz wartość `0.5` lub `0`, co da linię poziomą.
4. Wartość równa 4 jest miejscem przecięcia się linii z osią *y*, co widać na rysunku 2.3. Możesz podać inną wartość i przekonać się na własne oczy, jak zmieni się linia.

Po krótkim wprowadzeniu do programu DESMOS potrafisz już go używać do sprawdzania innych równań liniowych. To nam się przyda w dalszej części książki do sprawdzania, czy to, co narysowaliśmy na ekranie, jest zgodne z prawdą. Na przykład, jeśli jesteś przekonany, że Twój kod powinien rysować linię poziomą, ale rysuje linię nachyloną o 45 stopni, wtedy możesz użyć takiego programu jak DESMOS, by zobaczyć, co z Twoim kodem może być nie tak.

Twoja kolej!

Ćwiczenie B

Za pomocą DESMOS wykreśl linię o nachyleniu równym 8 i miejscu przecięcia równym 0,5. Teraz, gdy wiesz, jak jest zbudowana linia, zobaczmy, jak ją wykreślić z użyciem biblioteki Pygame. W przeciwieństwie do DESMOS, który pobiera równanie liniowe, Pygame wymaga tylko dwóch punktów, początkowego i końcowego. Jak się przekonasz w punkcie „Rysowanie linii piksel po pikselu”, Pygame wykonuje za nas dużo żmudnej pracy.

Do dzieła!

Zaraz odkryjesz, jak łatwo jest narysować linię z użyciem biblioteki Pygame. W tym ćwiczeniu uwzględnimy również kliknięcia myszką, by rysowanie linii było dynamiczne, i umożliwić użytkownikowi wybór punktu początkowego i końcowego.

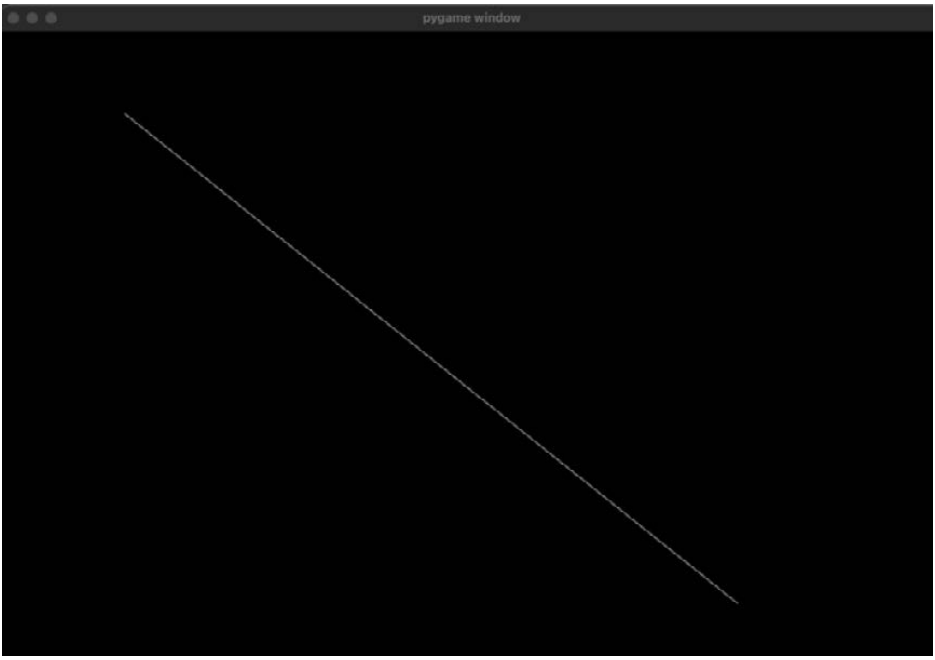
1. Utwórz nowy plik Pythona o nazwie *PygameLine.py*.
2. Dodaj kod pokazany poniżej, który używa wbudowanej metody do rysowania linii.

```
import pygame
from pygame.locals import * # Obsługa przycisków myszki

pygame.init()

screen_width = 1000
screen_height = 800
screen = pygame.display.set_mode((screen_width, screen_ height))

done = False
white = pygame.Color(255, 255, 255)
times_clicked = 0

while not done:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done = True
        elif event.type == MOUSEBUTTONDOWN:
            if times_clicked == 0:
                point1 = pygame.mouse.get_pos()
            else:
                point2 = pygame.mouse.get_pos()
            times_clicked += 1
            if times_clicked > 1:
                pygame.draw.line(screen, white, point1, point2, 1)
                times_clicked = 0
    pygame.display.update()
pygame.quit()
```

Wskazówka techniczna

W Pythonie linię komentarza rozpoczyna znak #.

Ten kod do zapisywania liczby kliknięć myszką używa zmiennej o nazwie `times_clicked`. Te kliknięcia wywołują zdarzenie `MOUSEBUTTONDOWN`. Za każdym razem, gdy przycisk myszy zostanie zwolniony, zapisywany jest punkt. Po dwóch kliknięciach uzyskamy dwa punkty, które będą stanowić dwa końce linii, w tym przykładzie linii koloru białego o grubości 1 piksela, tak jak pokazano na rysunku 2.4.



Rysunek 2.4. Prosta linia narysowana za pomocą Pygame

Uwaga

Więcej informacji o metodzie `pygame.draw` znajdziesz w dokumentacji biblioteki Pygame na stronie *https://www.pygame.org/docs/ref/draw.html* (strona w języku angielskim).

Jak już wiesz, Pygame bardzo ułatwia rysowanie linii w określonym kolorze między dwoma punktami.

Pygame ułatwia również tworzenie graficznych efektów tekstowych, o czym zaraz się przekonasz.

Tekst

Bardzo rzadko spotyka się graficzny interfejs użytkownika bez tekstu. W przeszłości wyświetlacze działały w dwóch trybach, trybie tekstowym i graficznym. Jeśli miałeś okazję używać oryginalnych wersji systemu DOS lub Unix, to pracowałeś w trybie tekstowym. To znaczy, że nie było możliwości rysowania na ekranie (oprócz pisania tekstu), a co za tym idzie, nie można było umieszczać tekstu na przyciskach, tekst nie mógł być wyświetlany do góry nogami ani bokiem, a sam wygląd liter był ograniczony.

Najbardziej wymyślne obrazki wyświetlane na ekranie były rysowane za pomocą tekstu, tak jak ta ryba:

```
|\ \\\\__ o
| \_/ o \ o
> _ (( <_ oo
| / \__+___/
|/ |/
```

Tego typu obrazy nazywamy sztuką znaków ASCII (ang. *ASCII art*), a więcej przykładów znajdziesz na stronie *https://www.asciiart.eu*. Swój początek miały w latach 70. i 80. (ubiegłego wieku) w usłudze komputerowej BBS (ang. *bulletin board system*), ale popularność zyskały, gdy monitory komputerowe i karty graficzne stały się bardziej zaawansowane i pojawiły się interfejsy użytkowników. Ludzie wciąż używają rysunków utworzonych ze znaków ASCII na przykład w czacie gier online, w mailach i na forach internetowych.

Obecnie to tekst jest rysowany na ekranach, czyli jest oparty na grafice, i dlatego możemy zmieniać jego różne atrybuty, między innymi krój czcionki, kolor, rozmiar, odstęp między wyrazami i kierunek.

Krój pisma, częściej nazywany po prostu **czcionką**, to szczególny zestaw znaków w takim samym stylu. Najbardziej znane czcionki to Arial, Verdana oraz Times New Roman. Są dwie kategorie czcionek: **szeryfowa** (ang. *serif*) i **bezszeryfowa** (ang. *sans serif*). Szeryf to ozdobna linia upiększająca czcionkę. Innymi słowy, to małe linie na końcach linii znaku, tak jak pokazano na rysunku 2.5.

Rysunek 2.5. Czcionka szeryfowa i bezszeryfowa

Choć na wyświetlaczach graficznych można używać czcionek szeryfowych, czcionki bezszeryfowe są łatwiejsze do odczytania.

Ten kod pokazuje, jak rysować tekst z użyciem biblioteki Pygame (a efekt jego działania jest pokazany na rysunku 2.6):

```
import pygame

pygame.init()
screen_width = 800
```

```
screen_height = 200
screen = pygame.display.set_mode((screen_width, screen_height))

done = False
white = pygame.Color(255, 255, 255)

pygame.font.init()
font = pygame.font.SysFont('Comic Sans MS', 120, False, True)

text = font.render('Penny de Byl', False, white)
while not done:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done = True
    screen.blit(text, (10, 10))
    pygame.display.update()
pygame.quit()
```

Rysunek 2.6. Biblioteka Pygame użyta do narysowania tekstu

Najpierw metoda `pygame.font.init()` inicjalizuje czcionki. Tę metodę trzeba wywołać tylko raz w całym programie. Potem ustawiany jest krój czcionki, w tym przypadku Comic Sans MS.

Aby w konsoli wylistować wszystkie czcionki systemowe, możesz użyć tego kodu:

```
print(pygame.font.get_fonts())
```

Następnie metoda `font.render` przygotowuje tekst w określonej czcionce. Parametry po łańcuchu znaków to wartości logiczne (prawda, fałsz) dla wygładzania krawędzi i koloru. Na tym etapie tekst nie jest wyświetlany, ale tylko konfigurowany.

W środku pętli `while` znajdziesz wywołanie metody `screen.blit`. Ta metoda umieszcza tekst na ekranie w podanym miejscu (x, y).

Być może się zastanawiasz, dlaczego tekst wymaga metody `screen.blit`, a rysowanie linii nie. Ta metoda służy do utworzenia powierzchni rysowania nad już istniejącą. W tym przypadku możesz postrzegać powierzchnię do rysowania jako obraz z przeźroczystością. Tekst jest rysowany na obrazie z użyciem metody `font.render` i zapisany w zmiennej `text`. Zatem jest przechowywany w pamięci. Metoda `screen.blit` kopiuje piksel po pikselu tekst do okna wyświetlania, łącząc kolory w tekście z tym, co już jest na ekranie. Zobaczysz więcej zastosowań metody `screen.blit`, gdy zaczniemy wgrywać zewnętrzne obrazy.

Aby móc użyć niestandardowej czcionki z biblioteką Pygame, potrzebujesz pliku z definicją czcionki. Rozszerzenie takich plików to *.ttf* (ang. *TrueType font*). Możesz stworzyć własną czcionkę za pomocą odpowiedniego programu, jednak na potrzeby następnego ćwiczenia możesz pobrać darmową czcionkę z internetu.

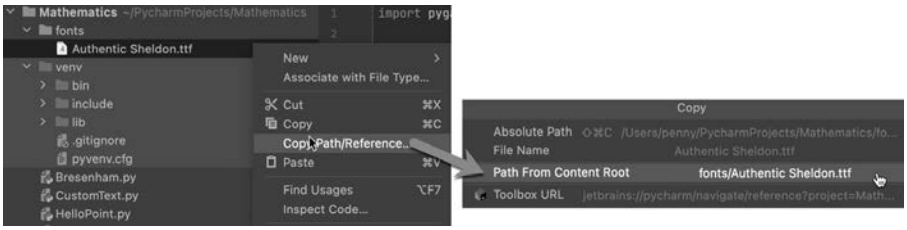
Do dzieła!

W tym ćwiczeniu nauczysz się, jak użyć pobranej czcionki z biblioteką Pygame, by wyświetlić tekst w oknie graficznym.

1. Wejdź na stronę *https://www.dafont.com/* i pobierz plik *.ttf*.
2. Przeciągnij plik z definicją czcionki (trzymając jednocześnie klawisz *Ctrl*) do okna z projektem w PyCharm. Edytor może Cię zapytać, czy chcesz zrefaktoryzować kod — wyraź na to zgodę.
3. Utwórz nowy plik Pythona o nazwie *CustomText.py* i umieść w nim kod z poprzedniego ćwiczenia.
4. Zastąp linie zaczynającą się od `font =` tą linią:

   ```
   font = pygame.font.Font('fonts/Authentic Sheldon.ttf',120)
   ```

   W tej linii łańcuch znaków to nazwa pliku z czcionką, którą pobrałeś. Pamiętaj, że w oknie projektu możesz utworzyć nowy folder i tam umieszczać pliki z czcionkami, by zachować porządek. Aby uzyskać ścieżkę do pliku z czcionką, gdziekolwiek będzie się znajdowała, kliknij prawym przyciskiem myszy nazwę pliku i skopiuj jej ścieżkę względną (przez wybranie *Path from Content Root*), tak jak widać na rysunku 2.7.



Rysunek 2.7. Sposób uzyskania ścieżki i nazwy pliku w edytorze PyCharm

5. Teraz wystarczy nacisnąć przycisk *Play*, by zobaczyć rezultat, widoczny również na rysunku 2.8.

Teraz już potrafisz wyświetlać tekst w oknie z użyciem dowolnej czcionki. W aplikacjach graficznych tekst jest powszechnie używany w interfejsach użytkownika. Następne wyzwanie ma na celu pomóc Ci jeszcze lepiej zrozumieć zastosowanie czcionki.

Rysunek 2.8. Zmiana kroju czcionki z użyciem biblioteki Pygame i pliku .ttf

Twoja kolej!

Ćwiczenie C

Znajdź inny plik z definicją czcionki (w formacie *.ttf*). Użyj go, by wyświetlić drugą linię tekstu, pod tą już zakodowaną.

Uwaga

Jeśli jesteś ciekawy, jak od podstaw stworzyć własną czcionkę, sprawdź edytor czcionek online na stronie *https://fontstruct.com/*.

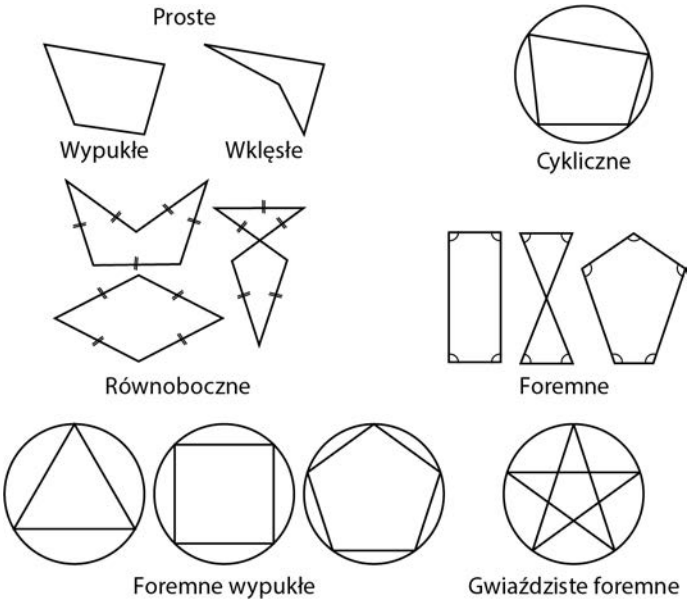
Jak możesz sobie wyobrazić, matematyka związana z rysowaniem czcionek jest dość skomplikowana. Litery mogą być rysowane za pomocą szeregu krzywych, a dokładniej mówiąc — **krzywych Béziera**, które omówimy w części III, „Podstawy geometrii". Ale na razie ograniczamy naszą dyskusję do elementów graficznych złożonych z linii prostych oraz wieloboków, którym przyjrzymy się w następnym podrozdziale.

Wieloboki

Wielobok jest płaskim kształtem określonym za pomocą kilku prostych linii połączonych ze sobą tak, że tworzą pole. Wielobok o najmniejszej liczbie boków to trójkąt. Poszczególne wieloboki (inaczej zwane wielokątami) biorą swoją nazwę od liczby boków. I tak wielokąt o trzech bokach to trójkąt, a o czterech — czworokąt (na przykład prostokąt lub kwadrat). Linie proste ograniczające wieloboki to krawędzie. Punkty, w których krawędzie się łączą, nazywamy wierzchołkami.

Wieloboki są klasyfikowane według ich kształtów, tak jak pokazano na rysunku 2.9.

Zarówno w grafice dwuwymiarowej, jak i trójwymiarowej większość obiektów składa się z wieloboków, od kwadratu przedstawiającego przycisk na ekranie do tysięcy trójkątów, z których złożony jest model 3D. Dlatego tak ważne jest, abyś dobrze rozumiał, czym są, jak je tworzyć i jak je zmieniać.



Rysunek 2.9. Rodzaje wieloboków
(źródło: *https://en.wikipedia.org/wiki/Polygon#/media/File:Polygon_types.svg*)

Oto kilka ciekawych i przydatnych faktów o wielobokach:

- Suma kątów wieloboku wypukłego wynosi 180 stopni lub więcej, na przykład każdy kąt w trójkącie równobocznym ma 60 stopni.
- Wypukły wielobok foremny ma wszystkie krawędzie tej samej długości i wpisuje się w okrąg.
- Suma kątów wewnętrznych prostego wieloboku o n krawędziach wynosi $(n - 2) \cdot 180$ stopni.
- Suma kątów zewnętrznych prostego wieloboku o n krawędziach zawsze wynosi 360 stopni.

Więcej przydatnych faktów znajdziesz w Wikipedii (*https://pl.wikipedia.org/wiki/Wielokąt*). W tej książce będziemy analizować wielokąty i ich geometrię, gdy zaczniemy tworzyć grę i grafikę, jednak zobaczymy krótko, jak z nimi pracować w bibliotece Pygame.

Do dzieła!

Jedyna różnica między rysowaniem linii a rysowaniem wieloboku z użyciem Pygame polega na dodaniu więcej niż dwóch punktów określających wierzchołki kształtu. Spróbuj sam.

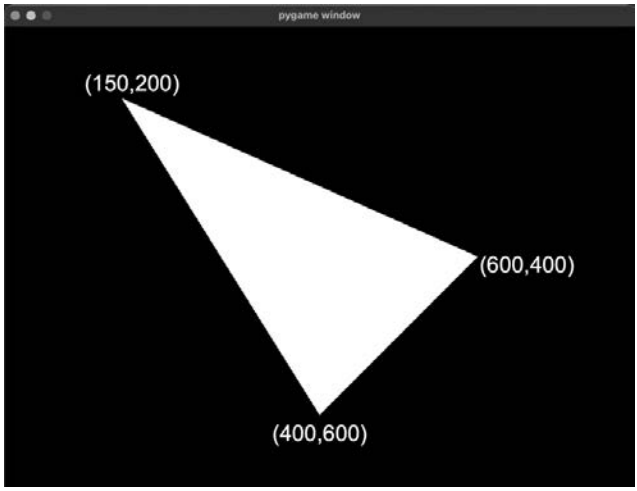
1. Utwórz nowy plik Pythona o nazwie *Polygons.py* i dodaj kod początkowy tworzący okno graficzne.
2. Zdefiniuj zmienną dla koloru białego przed pętlą `while`.

   ```
   white = pygame.Color(255, 255, 255)
   ```

3. W środku pętli `while`, zaraz nad definicją metody `update()`, zacznij linię kodu od `pygame.draw`. Pomoc w edytorze PyCharm pokaże Ci listę kształtów, które możesz narysować, a wśród nich będzie wielobok (*polygon*).
4. Teraz tak uzupełnij tę oto linię:

   ```
   pygame.draw.polygon(screen, white, ((150,200), (600,400),(400,600)))
   ```

   Zwróć szczególną uwagę na to, gdzie znajdują się nawiasy. Każda para współrzędnych *x* i *y* jest umieszczona w nawiasach, także wszystkie trzy pary znajdują się w nawiasach. Należy przekazać trzy pary współrzędnych do metody `polygon()`.
5. Uruchom program. Zobaczysz trójkąt taki sam jak na rysunku 2.10.



Rysunek 2.10. Rysunek wieloboku z oznaczonymi współrzędnymi wierzchołków

Parametry wymagane w celu narysowania kształtu to powierzchnia, na której ma być narysowany wielokąt, kolor wielokąta i w końcu lista wierzchołków.

Twoja kolej!

Ćwiczenie D

Zamiast kodować na sztywno wierzchołki wieloboku, utwórz program, który umożliwi wskazanie trzech wierzchołków trójkąta przez kliknięcie myszką (podobnie jak to miało miejsce w przypadku linii).

Bez względu na to, czy są to linie czy wieloboki, na ekranie to po prostu piksele, W przypadku tych elementów, istnieje matematyczna struktura określająca, w jaki sposób piksele są kolorowane. Następny element graficzny, który będziemy omawiać, już jest przedstawiony za pomocą pikseli. Obrazy rastrowe są szczególnie przydatne, gdy matematyka nie wystarczy. Na przykład gdy chcesz użyć prawdziwych zdjęć, umieszczać teksturę na obiektach 3D i przedstawiać obrazy tła.

Obrazy rastrowe

Ostatnim elementem graficznym omawianym w tym rozdziale są obrazy rastrowe. To standardowe rodzaje obrazów używane w komputerach, zapisywane w plikach *.png* lub *.jpg*. Są to siatki pikseli, gdzie po przybliżeniu obrazu możesz zobaczyć każdy piksel, co pokazano na rysunku 2.11.

Rysunek 2.11. Obraz rastrowy psa i zbliżenie na część oka, gdzie widać pojedyncze piksele

Gdyby obraz miał wysoką rozdzielczość, nie byłbyś w stanie dostrzec każdego piksela.

Jak już wspomniałam przy okazji tworzenia tekstu, w bibliotece Pygame obrazy mogą być rysowane na powierzchniach, które nie są widoczne na głównym ekranie, dopóki to nie będzie konieczne. To dotyczy również obrazów rastrowych.

Obrazy rastrowe najpierw są wczytywane do pamięci, a następnie za pomocą metody `screen.blit` możesz je umieszczać na wyświetlaczu bez końca, w różnych miejscach. Wybrany przez Ciebie obraz może zawierać przeźroczystość. To bardzo przydatne, gdyż dzięki temu możesz narysować sprite'a na istniejącym już tle.

Blitting jest techniką powszechnie stosowaną w grafice i w połączeniu z działaniami logicznymi służy do umieszczania jednego obrazu na drugim przez porównywanie pikseli. Dokładniejsze wyjaśnienie tej techniki znajdziesz na stronie *https://en.wikipedia.org/wiki/Bit_blit* (strona w języku angielskim). W następnym ćwiczeniu wykorzystamy tę technikę, by umieścić obraz tła, a na nim sprite'a.

Do dzieła!

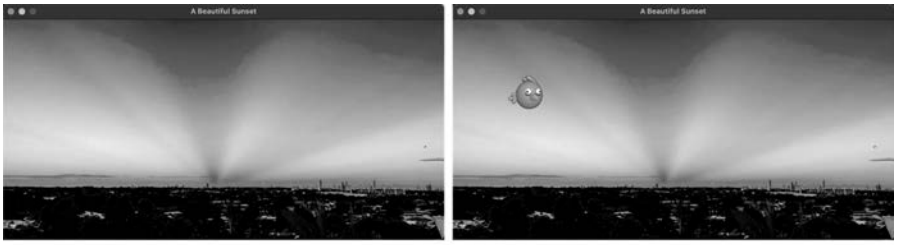
W tym ćwiczeniu będziemy pracować z obrazem tła, który wypełni całe okno, oraz z obrazem dla sprite'a. W tym celu wykonaj te kroki:

1. Utwórz nowy plik Pythona o nazwie *ShowRaster.py* i dodaj nasz standardowy początkowy kod.
2. Poszukaj w internecie obrazu, który mógłby stanowić tło i wypełnić całe okno. Nieważne, jaki rozmiar wybierzesz, możesz dostosować do niego rozmiar okna, żeby obraz dokładnie się wpasował.
3. Przeciągnij plik z obrazem do edytora PyCharm. Zrób to w ten sam sposób, w jaki zrobiłeś to w przypadku pliku z czcionką (to znaczy przytrzymaj klawisz *Ctrl*, by upewnić się, że plik się skopiował). Jeśli chcesz, możesz także utworzyć osobny folder o nazwie *images*, gdzie będziesz umieszczać obrazy.
4. Teraz do kodu dodaj poniżej pokazany kod wczytujący i wyświetlający obraz.

```
import pygame

pygame.init()
screen_width = 800
screen_height = 400
screen = pygame.display.set_mode((screen_width, screen_ height))

# Dodaj tytuł do okna.
pygame.display.set_caption("Piękny zachód słońca")
done = False

# Wczytaj obraz tła.
background = pygame.image.load("images/background.png")
while not done:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done = True
    screen.blit(background, (0, 0))
    pygame.display.update()
pygame.quit()
```

Na początku ustawiłam rozmiar okna na taki sam jak rozmiar obrazu, który wybrałam, czyli 800 na 400. Potem dodałam metodę `set_caption`, by Ci pokazać, jak zmienić tytuł okna. Sam obraz, o nazwie *background*, jest wczytywany za pomocą metody `pygame.image.load`. Zwróć uwagę, że ta metoda pobiera łańcuch znaków będący ścieżką do obrazu.

Ścieżkę możesz uzyskać w ten sam sposób, w jaki kopiowaliśmy ścieżkę dla pliku *.ttf*. Wczytywanie obrazu odbywa się poza pętlą `while`, gdyż jest to jednorazowa akcja. Obraz tła jest następnie nakładany na ekran techniką *blitting*, a jego lewy górny róg znajduje się w punkcie (0,0). Rysunek 2.12 (a) przedstawia obraz użyty jako tło okna.



Rysunek 2.12. Obraz tła (a) ze sprite'em (b)

5. Teraz na tło nałożymy inny obraz, w części przeźroczysty. Do tego celu potrzebujesz obrazu w formacie *.png*, taki który mógłby być obrazem dla postaci z gry 2D. Tego typu obrazy znajdziesz na stronie *https://www.iconarchive.com/*. Pobierz ten, który Ci się podoba i ma rozdzielczość 64×64, i zaimportuj go do edytora PyCharm.
6. Proces dodawania pliku *.png* do okna graficznego jest taki sam jak w przypadku dodawania obrazu tła. Najpierw obraz jest umieszczany w zmiennej poza pętlą `while`, a następnie nakładany na powierzchnię. Umieszczenie kodu dla nowego obrazu za kodem dla obrazu tła da nam pewność, że drugi obraz będzie znajdował się na wierzchu, co widać na rysunku 2.12 (b). We fragmencie tego kodu nowe linie zostały pogrubione.

```
background = pygame.image.load('images/background.png')
sprite = pygame.image.load('images/Bird-blue-icon.png')
while not done:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done = True
    screen.blit(background, (0, 0))
    screen.blit(sprite, (100, 100))
    pygame.display.update()
pygame.quit()
```

Metoda `load()` wczytuje sprite'a do pamięci, gdzie nazwa sprite'a to faktyczny obraz. Teraz musi być tylko narysowany w oknie. To osiągniemy za pomocą metody `blit()`.

Pomimo całej sprytnej matematyki, którą przez lata opracowywali informatycy, często jakość obrazu jest lepsza, a jego złożoność jest lepiej przedstawiana, gdy ma on postać obrazu rastrowego. Jest wiele algorytmów — kilka z nich odkryjesz w tej książce — służących do generowania dość przekonywających obrazów. Ale czasami te algorytmy nie stanowią dobrej alternatywy dla rzeczywistych obrazów.

Podsumowanie

Omówiliśmy podstawowe aspekty grafiki komputerowej. Dzięki temu na ekranie można narysować wszystko. Choć przykłady w tym rozdziale dotyczyły wyłącznie dwóch wymiarów, te same zasady dotyczą grafiki 3D, o czym przekonasz się w dalszej części niniejszej książki. Teraz masz wystarczające umiejętności, by tworzyć własne proste projekty graficzne 2D, które łączą w sobie obrazy i czcionki, a ponadto reagują na kliknięcia myszką.

W rozdziale 3., „Kreślenie linii piksel po pikselu", zajrzymy pod maskę wysokopoziomowych metod rysowania, które poznałeś w tym rozdziale. Przeanalizujemy krok po kroku algorytm, który jest używany od zarania grafiki komputerowej, by zobaczyć, jak pojedyncze piksele powinny być kolorowane podczas rysowania linii.

Odpowiedzi

Ćwiczenie A

```
screen.set_at((x, y), pygame.Color(0, int(x/1000 * 255), int(y/800 * 255)))
```

Ćwiczenie B

```
y = 8x + 0.5
```

Ćwiczenie C

Dla kolejnej linii tekstu wyświetlanego inną czcionką musisz zadeklarować dwie czcionki i dwa teksty:

```
font1 = pygame.font.Font('fonts/Authentic Sheldon.ttf', 120)
font2 = pygame.font.Font('fonts/PinkChicken-Regular.ttf', 120)
text1 = font1.render('Penny de Byl', False, white)
text2 = font2.render('Kolejna linia tekstu', False, white)
```

a następnie w pętli `while` nałożyć na okno oba teksty.

```
screen.blit(text1, (10, 10))
screen.blit(text2, (10, 100))
```

Ćwiczenie D

```
import pygame
from pygame.locals import *

pygame.init()
screen_width = 800
screen_height = 800
screen = pygame.display.set_mode((screen_width, screen_height))
```

```
pygame.display.set_caption('Wyklikany trójkąt')
done = False
white = pygame.Color(255, 255, 255)
timesClicked = 0
pygame.font.init()

while not done:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done = True
        elif event.type == MOUSEBUTTONDOWN:
            if timesClicked == 0:
                point1 = pygame.mouse.get_pos()
            elif timesClicked == 1:
                point2 = pygame.mouse.get_pos()
            else:
                point3 = pygame.mouse.get_pos()
            timesClicked += 1
            if timesClicked > 2:
                pygame.draw.polygon(screen, white, (point1, point2, point3))
                timesClicked = 0

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

Skorowidz

Matematyka: najlepszy sprzymierzeniec programisty i grafika!

Matematyka jest niezbędna do zrozumienia reguł rządzących tworzeniem grafiki komputerowej w czasie rzeczywistym, a także zasad manipulowania obiektami i środowiskami 3D. Idealnym narzędziem ułatwiającym uchwycenie tych zależności jest język Python wraz z bibliotekami Pygame i PyOpenGL. Dzięki nim łatwo zrozumiesz, w jaki sposób komputery tworzą i wprowadzają zmiany w środowiskach trójwymiarowych.

Ta książka wyjaśni Ci rolę matematyki w tworzeniu, renderowaniu i zmienianiu wirtualnych środowisk 3D, a ponadto pozwoli odkryć tajemnice najpopularniejszych dzisiaj silników gier. Za sprawą licznych praktycznych ćwiczeń zorientujesz się, co się kryje za rysowaniem linii i kształtów graficznych, stosowaniem wektorów i wierzchołków, budowaniem i renderowaniem siatek, jak również przekształcaniem wierzchołków. Nauczysz się używać kodu Pythona, a także bibliotek Pygame i PyOpenGL do budowy własnych silników. Dowiesz się też, jak tworzyć przydatne API i korzystać z nich podczas pisania własnych aplikacji.

W książce między innymi:

- praca w Pythonie z edytorem PyCharm, bibliotekami Pygame i PyOpenGL
- polecenia rysowania z różnych graficznych API
- najważniejsze zagadnienia trygonometrii w odniesieniu do środowisk 3D
- wektory i matryce w przenoszeniu, ustawianiu kierunku i skalowaniu obiektów 3D
- renderowanie obiektów 3D z teksturami, kolorami, cieniami i oświetleniem
- przekształcanie wierzchołków w celu przyspieszenia renderowania opartego na GPU

Dr Penny de Byl jest programistką full stack. Od ponad 25 lat wykłada grafikę i programowanie gier na uniwersytetach w Australii i Europie. Jest autorką książki *Holistic Game Development with Unity* i laureatką licznych nagród, w tym Australian Government Excellence in Teaching Award i Unity Mobile Game Curriculum Competition.

Helion
helion.pl
HELION SA
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl
KOD KORZYŚCI
Sięgnij po więcej!
ISBN 978-83-289-0879-6
9 788328 908796
Cena: 89,00 zł
packt