# Kurs rozszerzony języka Python

## PyGame

Marcin Młotkowski

22 grudnia 2017

# Plan wykładu

# Plan wykładu

# Co to takiego

Biblioteka w Pythonie do programowania aplikacji multimedialnych, w tym gier.

# Elementy składowe

- Obsługa ekranu (obrazki, kursor, fonty);
- urządzenia peryferyjne (myszki, joysticki, klawiatury);
- dźwięk;
- przekształcanie obrazów (w tym OpenGL).

# Jeszcze parę uwag

## PyGame używa SDL

Simple DirectMedia Layer: przenośna biblioteka multimedialna;

# Jeszcze parę uwag

## PyGame używa SDL

Simple DirectMedia Layer: przenośna biblioteka multimedialna;

## Python 3

PyGame nie jest jeszcze w pełni zintegrowane z Pythonem 3.

# Schemat programu

```
import pygame

pygame.init()

# ustawienie parameterów wyświetlania

while True:
    # reakcja na zdarzenia
    pygame.display.update()
```

# Parametry ekranu

# Obiekt klasy **Surface**
screen = pygame.display.set_mode( *rozdzielczość*, *flaga*, *kolory* )

- rozdzielczość: krotka (szerokość, wysokość), np. (640, 480);
- flaga: sposób wyświetlania; można dać 0, FULLSCREEN, NOFRAME, OPENGL etc.
- kolory: liczba bitów na kolor (8, 15, 16, 24, 32)

# Parametry ekranu

# Obiekt klasy **Surface**
screen = pygame.display.set_mode( *rozdzielczość*, *flaga*, *kolory* )

- rozdzielczość: krotka (szerokość, wysokość), np. (640, 480);
- flaga: sposób wyświetlania; można dać 0, FULLSCREEN, NOFRAME, OPENGL etc.
- kolory: liczba bitów na kolor (8, 15, 16, 24, 32)

Nagłówek okna:

```
pygame.display.set_caption('Hello, PyGame!')
```

# Powierzchnie

- *powierzchnie* reprezentują obrazy;
- obrazami mogą być utworzone z pliku graficznego, fonty, można też samodzielnie tworzyć obrazy;
- klasa **pygame.Surface**

# Pliki graficzne

pygame.image.load('plik.jpg')

- tworzy powierzchnię rozmiaru i o tych samych kolorach co oryginalny obrazek;
- zalecane jest używanie zmodyfikowanej *kopii* powierzchni:
  obrazek = pygame.image.load('plik.jpg').convert()
  ze względu na szybkość działania przy wielokrotnym wczytywaniu obrazka;
- można modyfikować obrazki, np. dodając kanał alfa (`convert_alpha()`).

# Tworzenie "pustych" powierzchni

```
pygame.Surface((256,256))
```

Tworzy powierzchnię 256x256 z tą samą liczbą kolorów jak ekran.

# Wyświetlanie powierzchni na powierzchni

.blit(`powierzchnia`)

# Wyświetlanie powierzchni na powierzchni

.blit(`powierzchnia`)

## Umieszczenie obrazka na ekranie

screen = pygame.display.set_mode( *rozdzielczość*, *flaga*, *kolory*)
obrazek = pygame.image.load('plik.jpg').convert()
screen.blit(obrazek, (0,0))

# Wyświetlanie powierzchni na powierzchni

.blit(`powierzchnia`)

## Umieszczenie obrazka na ekranie

screen = pygame.display.set_mode( *rozdzielczość*, *flaga*, *kolory* )
obrazek = pygame.image.load('plik.jpg').convert()
screen.blit(obrazek, (0,0))

pygame.display.update()

# Fonty i napisy

Fonty w PyGame:

- PyGame korzysta z systemowych fontów TTF;
- przed użyciem konieczne jest utworzenie obiektu klasy `Font`, np.:
  ```
  font_podstawowy = pygame.font.SysFont("arial", 16)
  ```
- renderowanie fontu to utworzenie powierzchni:
  ```
  napis = font_podstawowy.render('Python is cool!',
                                 (0,0,0), (255,255,255))
  screen.blit(napis, (10, 10))
  ```

# Zapisywanie powierzchni

```
pygame.image.save(powierzchnia, 'plik.png')
```

# Plan wykładu

# Zdarzenia

Zdarzenia typu naciśnięcie klawisza są trzymane w kolejce.

# Zdarzenia

Zdarzenia typu naciśnięcie klawisza są trzymane w kolejce.

## Przykład obsługi zdarzenia "zamknięcie okna"

```
from pygames.locals import *
for event in pygame.event.get():
    if event.type == QUIT:
        exit()
```

# Naciśnięcie dowolnego klawisza

## pygame.key.get_pressed()

Zwraca listę boolowską, gdzie na odpowiednich pozycjach jest informacja, czy klawisz zsotał naciśnięty.

# Naciśnięcie dowolnego klawisza

## pygame.key.get_pressed()

Zwraca listę boolowską, gdzie na odpowiednich pozycjach jest informacja, czy klawisz zsotał naciśnięty.

## Przykład

```
pressed_keys = pygame.key.get_pressed()
if pressed_keys[K_SPACE]:
    fire()
```

# Schemat obsługi klawiszy kontrolnych

## Zdarzenia

KEYDOWN, KEYUP

```
if event.type == KEYDOWN:
    if event.key == K_LEFT:
        ...
    if event.key == K_RIGHT:
        ...
```

# Zdarzenia myszy

MOUSEMOTION
MOUSEBUTTONDOWN
MOUSEBUTTONUP

# Pozycja myszy

```
x, y = pygame.mouse.get_pos()
```

# Własny kursor

## Przykład

```
myszka = pygame.image.load('logo.gif').convert()
while True:
    x, y = pygame.mouse.get_pos()
    screen.blit(myszka, (x, y))
```

# Własny kursor

## Przykład

```
myszka = pygame.image.load('logo.gif').convert()
while True:
    x, y = pygame.mouse.get_pos()
    screen.blit(myszka, (x, y))
```

## Uwaga

"Kursor" myszki jest rysowany wielokrotnie.

# Podstawowe funkcje

## Inicjowanie modułu

pygame.joystick.init()

## Liczba podłączonych dżojstików

pygame.joystick.get_count()

# Obsługa dżojstików

## Klasa **Joystick**

joy1 = pygame.joystick.Joystick(0)

Dżojstiki są numerowane kolejnymi liczbami naturalnymi.

# Obsługa dżojstików

## Klasa **Joystick**

joy1 = pygame.joystick.Joystick(0)

Dżojstiki są numerowane kolejnymi liczbami naturalnymi.

## Liczba aktywnych przycisków

Joystick.get_numbuttons()

# Zdarzenia związane z dżojstikiem

JOYAXISMOTION, JOYBALLMOTION, JOYHATMOTION, JOYBUTTONUP, JOYBUTTONDOWN

Informacje związane ze zdarzeniem:

- .joy(): numer dżojstika;
- .axis(): kierunek

# Plan wykładu

# Inicjowanie modułu

pygame.mixer.init(*częstotliwość*, *rozmiar*, *stereo*, *bufor*)

częstotliwość: częstotliwość próbkowania
rozmiar: liczba bitów na próbkę
stereo: 1 – mono, 2 – stereo
bufor: rozmiar bufora na próbki

## Przykładowe ustawienia

```
pygame.mixer.init(44100, 16, 2, 4096)
```

# Odtwarzanie plików dźwiękowych

```
bum = pygame.mixer.Sound('bummm.ogg')
```

# Odtwarzanie plików dźwiękowych

```
bum = pygame.mixer.Sound('bummm.ogg')
```

## Odtwarzanie dźwięków

- .play(): rozpoczęcie odtwarzania;
- .stop(): zatrzymanie odtwarzania;
- .fadeout(): zatrzymanie odtwarzania po ściszeniu

## Zainicjowanie strumienia

```
pygame.mixer.music.load('opera.ogg')
```

## Odtwarzanie

- pygame.mixer.music.play(): rozpoczęcie odtwarzania;
- pygame.mixer.music.stop()
- pygame.muxer.music.pause()
- pygame.mixer.music.unpause()

# Moduł midi

pygame.midi

# Plan wykładu

# Małe przypomnienie

- ekranem jest obiekt klasy Surface;
- powierzchniami są wczytane obrazki lub napisy.

# Zabawa z pikselami

```
kartka = pygame.Surface( (1024, 1024), depth=24)
kartka.set_at( (20, 20), (255, 0, 0) )
```

# Zabawa z pikselami

```
kartka = pygame.Surface( (1024, 1024), depth=24)
kartka.set_at( (20, 20), (255, 0, 0) )
```

Zbadanie koloru:

```
    tuple(kartka.get_at((20, 20)))
```

# Figury geometryczne

Funkcje modułu `pygame.draw.`:

- rect(powierzchnia, kolor, Rect( pozycja, rozmiar) )
- circle(powierzchnia, (0,0,255), środek, promień)
- line(powierzchnia, kolor, początek, koniec)
- inne ...

# Samodzielna implementacja

## Reprezentacja obiektów

Obiekty trójwymiarowe są reprezentowane jako trójwymiarowe.

## Rzutowanie

Odwzorowanie obiektu trójwymiarowego na płaszczyźnie.

Typowe rzuty: rzut równoległy bądź perspektywiczny.

# OpenGL: Open Graphics Library

Specyfikacja uniwersalnego API do generowania grafiki.

```
from OpenGL.GL import *
screen = pygame.display.set_mode((640, 480),
    HWSURFACE|OPENGL|DOUBLEBUF)
```

# Plan wykładu