PyGameZero

Tworzenie gier wymaga zapanowania nad m.in. wyświetlaniem obiektów na ekranie, wykrywaniem kolizji, obsługą urządzeń wejścia (np. gamepadów) itd. Stąd też powszechne jest korzystanie z gotowych "silników" i bibliotek zapewniających te rzeczy - tak, żeby programiści mogli się skoncentrować na samej logice gry.

PyGame Zero buduje na innej bibliotece - PyGame. Upraszcza ją jednak trochę np. dba o uruchamianie właściwej pętli z programem.

https://pygame-zero.readthedocs.io/

Chcąc skorzystać z bazowej biblioteki PyGame, bez ułatwień oferowanych przez PGZero, warto sięgnąć po PyGame CE - to nowsza odsłona tego projektu.

Instalacja

```
pip3 install pgzero
```

Szablon aplikacji

```
import pgzrun
from pygame import display
from pgzero import screen as pgzero_screen
from pgzero.builtins import keyboard

# rozmiar okna - pozniej wrzucimy to do osobnego pliku
WIDTH = 800
HEIGHT = 600

screen = pgzero_screen.Screen(display.set_mode((WIDTH, HEIGHT), 0))

# pygame uruchamia pętlę gry automatycznie
# ... ale potrzebuje do tego odpowiednio nazwanych funkcji.
```

```
# Korzystamy z funkcji update i draw.


def update():
    # tu aktualizujemy stan obiektow
    pass


def draw():
    # tu je rysujemy
    pass


# wywołanie pętli gry
pgzrun.go()
```

W powyższym szablonie została nieco rozbudowana sekcja importów - tak, aby edytory kodu nie "marudziły" podczas tworzenia aplikacji (biblioteka niestety jest napisana w sposób, który niezbyt dobrze współgra z tego typu narzędziami).

Folder images i sounds

W folderze z projektem dobrze jest stworzyć podfoldery `sounds` i `images`. To miejsca w których domyślnie PGZero będzie szukać dźwięków i obrazków.

Niezbędne funkcje i klasy

Funkcja update()

Jeśli będziemy mieli jakieś obiekty, których stan się będzie zmieniał niezbędna będzie nam funkcja update(). W nią wrzucimy takie rzeczy jak np. sterowanie, wykrywanie kolizji etc. - czyli wszystko co zmienia stan obiektu (np. dodaje / odejmuje jakąś wartość do atrybutu odopowiadającego za jego położenie na ekranie).

Funkcja draw()

W kodzie na pewno będziemy potrzebowali stworzyć funkcję draw. Odpowiada ona rysowanie obiektów.

Klasa Actor

Wszędzie tam gdzie będą obiekty wchodzące z sobą w interakcje przyda nam się klasa Actor. Zapewni nam obsługę kolizji, zadba, że odpowiedni obrazek znajdzie się we właściwym miejscu itd.

Metoda .draw.text z screen

Służy do obsługi tekstu w grze.

Wersja #9
Utworzono 2025-08-11 13:31:32 UTC przez Przemek Jeske
Zaktualizowano 2025-08-12 11:16:39 UTC przez Przemek Jeske