WYDZIAŁ ELEKTRONIKI
I TECHNIK INFORMACYJNYCH

Politechnika Warszawska
Wydział Elektroniki i Technik Informacyjnych

Dokumentacja Projektu "Pacman" (Python 3.8.10)

Autorzy:

Wojciech Sekuła

Warszawa,
Styczeń 2022.

Spis treści

1 O Projekcie

1.1 Temat

Tematem Projektu było zaimplementowanie klona gry Pacman w języku Python. Implementacja miała posiadać następujące funkcjonalności:

1. Podstawowe funkcje gry Pacman
2. Możliwość przeprowadzenia wielu rozgrywek pod rząd
3. Menu gry
4. Wczytanie oraz zapis rozgrywki
5. Wyświetlenie tablicy High Scores

1.2 Założenia Gry

Moją implementację napisałem w Pythonie z użyciem biblioteki Pygame. Gra ma kilka prostych założeń które starałem się aby były jak najbardziej zbliżone do oryginału.

1. Ruch Pacmana odbywa się w labiryncie, Pacman może poruszać się w lewo,prawo do góry lub do dołu za pomocą strzałek
2. Pacman może zjadać kulki rozłożone po labiryncie które dają mu punkty
3. Pacman może zjeść specjalne kulki które na 6 sekund dają mu możliwość zjadania duszków. Gdy duszek zostanie zjedzony odrodzi się dopiero po ustaniu czasu trwania efektów specjalnej kulki czyli po 6 sekundach chyba że pacman zje kolejną specjalną kulkę

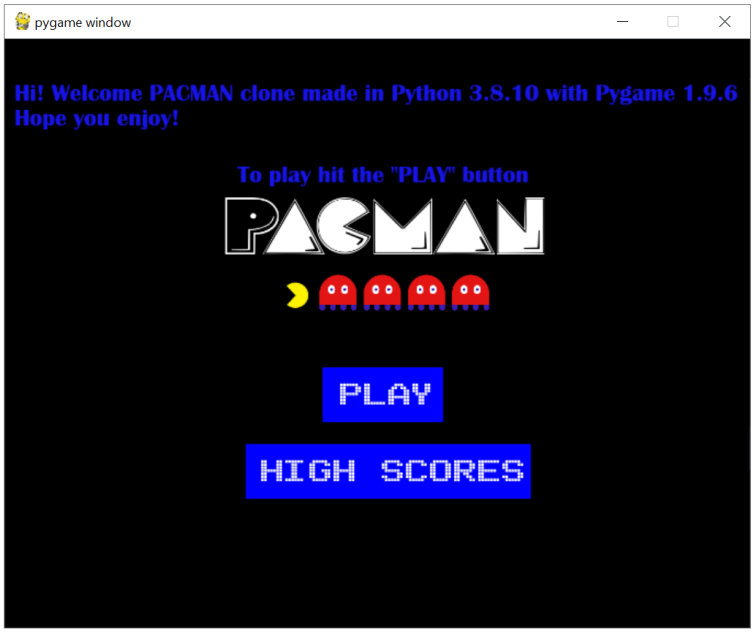
4. Gdy Pacman nie będąc pod wpływem specjalnej kulki trafi w duszka umiera i gra zostaje przerwana

5. Pacman Wygrywa gdy zje wszystkie monety na planszy

6. Ruch duszków jest losowy jednak zmieniają one kierunki w sposób przewidywalny to znaczy że nie ma takiej opcji że duszek natrafiając na ścianę lub skrzyżowanie

1.3 Gra

Główny ekran Gry po jej uruchomieniu:

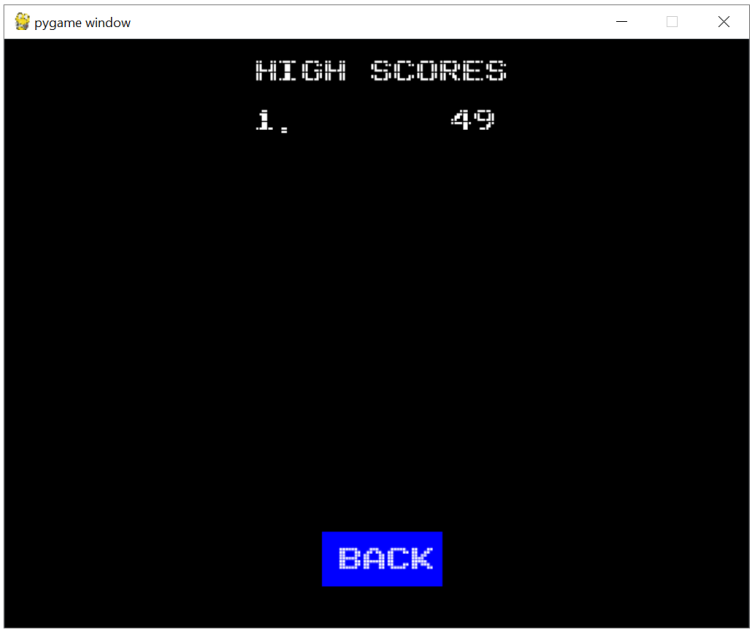


1. Aby rozpocząć grę należy kliknąć w niebieski przycisk `PLAY`

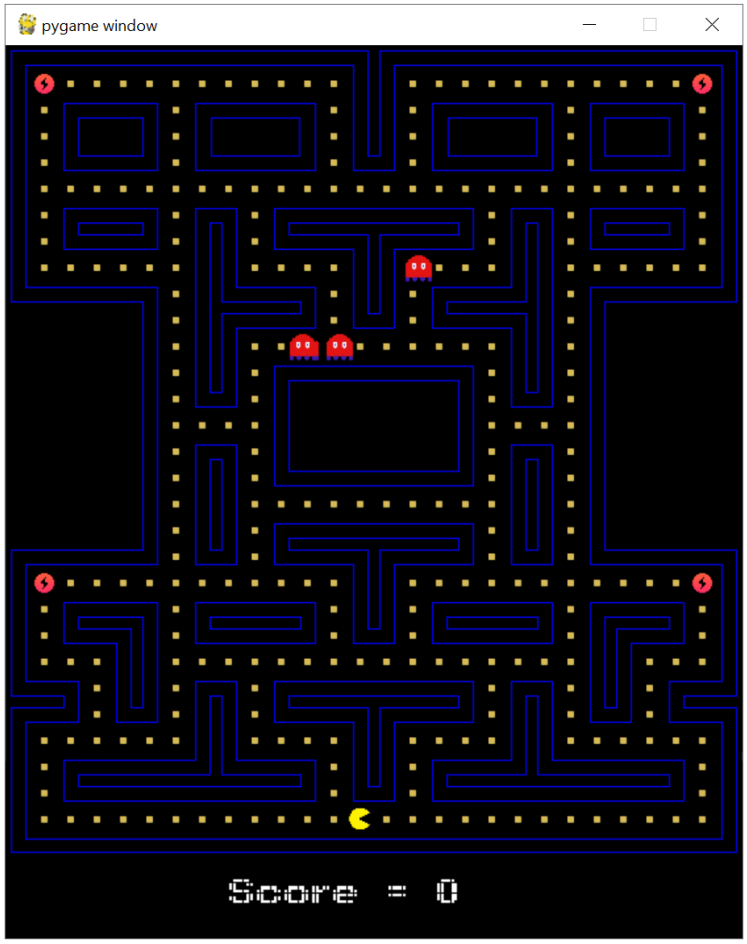
2. Aby wyjść z gry należy zamknąć okno przyciskiem zamykania

3. Aby wejść w tablicę High Scores należy kliknąć w niebieski przycisk `HIGH SCORES`. Aby z niej wyjść z powrotem do Menu gry należy kliknąć przycisk `BACK`

Tablica High Scores:



Rozgrywka:



2 Dokumentacja Programu

2.1 Pliki Programu

1. `main.py` - wywoływanie głównych funkcji
2. `game.py` - klasa Game a w niej metody sterujące rozgrywką oraz wyświetlaniem Gry, Menu, Tablicy Highscores
3. `player.py` - klasa Player dziedzicząca po klasie pygame.Sprite a w niej metody sterujące Pacmanem
4. `menu.py` - klasa Button używana w Menu oraz Tablicy High Scores
5. `enemy.py` - klasa Enemy dziedzicząca po klasie pygame.Sprite w niej metody sterujące duszkami w tym jedna z najważniejszych metod - metoda losowego ruchu duszków
6. `maze.py`: Nieruchome elementy Labiryntu (wszystkie dziedziczą po klasie pygame.Sprite) :
   (a) Klasa Block - niewidoczna przeszkoda blokująca ruchy Pacmana i Duszków
   (b) Klasa Coin - reprezentuje monety które może zjadać Pacman
   (c) Klasa Energizer - reprezentuje Energizery które może zjadać Pacman
7. `settings.py` - Zawiera wszystkie 'ustawienia' gry - zmienne globalne np. kolory w rgb, mapę labiryntu do zczytania przez funkcję create_map(), współrzędne pixelowe skrzyżowań, ścieżki do obrazów itd...
8. `files.py` - Zawiera klasę ScoreTable która zajmuje się wynikami (high scores) oraz funkcje zapisu i odczytu plików .csv
9. `highscores.csv` - plik tekstowy do zapisywania wyników high scores
10. `fonts` - wszystkie czcionki
11. `images` - wszystkie obrazy
12. `test_files.py` - testy pliku files.py
13. `files_for_test` - tylko na potrzeby testów

2.2 Działanie Programu

Główna pętła programu znajduje się w pliku main.py gdzie najpierw zostaje zaiinicjowana gra oraz pierwsze wywołanie funckji intro screen. Następnie program wchodzi do pętli sterującej która sterowana jest zmienną 'running'. Jest to główna zmienna sterująca która zostaje ustawiona na False gdy użytkownik zamknie okno Gry lub Menu lub High Scores (wtedy wyłącza cały program).
Gdy użytkownik wyjdzie z Menu przez kliknięcie 'PLAY' tworzy się gra metodą 'new()' i rozpoczyna się pętla main() w której zostają wywołane kolejne metody sterowania grą.

1. events() - sprawdza czy użytkownik nie zamknął programu

2. update() - update'uje wszystkie sprite'y (poza blokami labiryntu) w grze czyli Pacmana, Duszki, Monety i Energizery

3. draw() - rysuje obraz labiryntu, punkty Pacmana i wszystkie sprite'y (poza blokami labiryntu) Menu jest zarazem ekranem początkowym jaki i końcowym, w przypadku śmierci Pacmana metoda introscreen() jest wywoływana z argumentem True aby po kliknięciu w 'PLAY' utworzyć nową grę

2.2.1 Ważne Decyzje Architektoniczne

1. Menu i gra opierają się na 2 ekranach - IntroScreen oraz Screen (ekran gry)

2. Labirynt zostaje utworzony przez metodę create_map() która zczytuje z mapy (settings.MAP) współrzędne konkretnych obiektów (spritów) i kolejno je tworzy

   (a) `.` = Block

   (b) `P` = Player

   (c) `#` = Coin

   (d) `O` = Energizer

   (e) `E` = Enemy

3. Na potrzeby głównie kolizji ale też wyświetlania (dzięki funkcji płaszczyzn) stworzyłem 6 grup spitów (player, enemy, blocks, coins, energizers oraz all_sprites) Najważniejsza z nich jest grupa all_sprites do której przypisane są wszystkei sprity wymagające ciągłego update'u

4. High Scores są obługiwane przes plik files.py i przechowywane w pliku tekstowym csv z którego są pobierane i zapisywane na początku i końcu każdej rozgrywki

5. Ruch duszków jest losowy jednak oparty o prostą funkcje. Duszek porusza się w jednym kierunku, jeżeli natrafi na przeszkodę to zmienia ten kierunek a jeżeli natrafi na jeden z 9 rodzajów skrzyżowań to zgodnie z rodzajem wybiera następny kierunek. Na przykład jeżeli rafi na skrzyżowanie w kształcie 'x' to wybierze dowolny kierunek z wyłączeniem poprzedniego kierunku czyli nie odbije się od skrzyżowania, jeżeli natrafi na skrzyżowanie w kształcie L to wybierze jedyny możliwy kierunek czyli w górę lub w prawo (znów nie odbije się od skrzyżowania) odpowiednio dla innych rodzajów skrzyżowań

6. Ruch Pacmana różni się od oryginału, postanowiłem że pacman będzie mógł zatrzymywać się gdy użytkownik nie będzie wciskał żadnego przycisku sterującego ruchem. Ponadto na potrzeby ułatwienia rozgrywki (to jest poruszania się po labiryncie) zmniejszyłem rozmiar Pacmana do 18 pikseli. Domyślny rozmiar każdego korytarza to 20 pikseli więc pacman musiałby się idealnie mieścić na każdym zakręcie. Powoduje to efekt trochę uboczny tzn pacman może się delikatnie ruszać góra/dól/lewo/prawo będąc w korytarzu labiryntu przez co zmienia kierunek w który się patrzy. Uznałem to jednak za minimalny kosmetyczny urąbek i postanowiłem zostawić, daje to o wiele większą swobodę ruchu a także możliwość np kręcenia się w kółko stojąc w miejscu Pacmanem

7. Gra kończy się gdy pacman zje wszystkie Monety lub sam zostanie zjedzony przez Duszka. W obydwu przypadkach gra przerywa się dzięki zmiennej sterującej (game.playing) i gdy pacman wygrał wyświetla się odpowiedni komunikat na ekranie menu a jeżeli przegrał również odpowiedni komunikat razem z punktami z rozgrywki

8. Labirytnt tworzony jest z dwóch obiektów (w zasadzie jednego - rodzaju) są to obiekty klasy Block zdefiniowanej w `maze.py` z tym że są to tylko powierzchnie które są tworzone tylko raz nie są updateowane ani rysowane. służą wyłącznie do określenia czy pacman lub duszek się z nimi zderzyły i są tak wielkościowo dopasowane aby odpowiadały 'nałożonemu' na nie obrazowi labiryntu

2.3 Uruchamianie Programu

Aby uruchomić grę należy wejść i uruchomić plik main.py znajdując się w katalogu /pacman (w przeciwnym wypadku dostaniemy błędy związane z nieistniejącymi plikami /images/* lub /fonts/*

2.3.1 Dane Wejściowe/Wyjściowe

Program nie wymaga żadnych danych wejściowych czy wyjściowych podczas uruchamiania, wszystko łądowane jest automatycznie z plików folderu głównego. Jest to spowodowane tym że nie udało mi się zaimplementować funkcji zapsiu/odczytu gry

3 Problemy

Podczas realizacji projektu natrafiłem na wiele róznej natury problemów. Pierwszym był oczywiśćie brak znajomości biblioteki Pygame. Więc zanim usiadłęm do kodu Pacmana napisałem dwie proste gry w pygame (snake i SpaceInvader) posiłkując się różnego rodzaju poradnikami. Potem przeszedłem do obłsługi gita oraz planowania Projektu.
Kolejnym dużym problemem jaki napotkałem było wymyślenie sposobu na narysowanie labiryntu razem z przeszkodami jednak jakoś sobie z nim poradziłem Sporym wyzywaniem było również wymyślenie sposobu sterowanie duszkami. Wybrałem losowy sposób co okazało się całkiem trudne ale po jakiejś 3 próbie implementacji również sobie poradziłem Problemem nie do przeskoczenia pewnie z powodu czasu okazało się wczytanie i zapis Gry. Nie udało mi się zaimplementować tych dwóch funkcjonalności

4 Z czego jestem zadowolony

Po pierwsze to z całego wyglądu Gry. Oczywiście Menu, napisy, Tablicę wyników można by upiększyć jednak jestem bardzo zadowolony z efektów które uzyskałem.
Drugą rzeczą jest ruch duszków, zwyczajnie przyjemnie się na niego patrzy.
Jestem również całkiem zadowolony z plików projektu tzn z jego architektury jako że jest to mój pierwszy taki duży program

5 Z czego NIE jestem zadowolony

Na pewno z niezaimplementowanych funkcjonalności wymaganych w projekcie. Z powodu braku czasu nie udało mi się zaimplementować ulepszenia ruchu duszków aby zaczęły gonić

Pacmana gdy znajdują się w określonej odległości od niego. Zdecydowanie polepszyło by to rozgrywkę. Również nie jestem zadowolony z mojej implementacji klasy Enemy i Player. Powinna być jedna klasa Player i dziedziczące po niej klasy Ghost i Pacman to by oszczędziło kawałek kodu jednak po kilku próbach poddałem się z powodu braku czasu i licznych błędów/problemów jakie napotykałem.