Gra Frog

Dokumentacja techniczna

Projekt PIPR

Bartłomiej Niewiarowski

Numer indeksu: **318701**

- # Cel projektu

Celem mojego projektu było stworzenie programu, który umożliwia użytkownikowi gry w grę Frog. Program wizualnie miał być intuicyjny dla użytkownika, dlatego zdecydowałem się na interfejs graficzny zamiast konsolowego. Projekt ma być zrealizowany według obiektowego paradygmatu programowania, w taki sposób aby jego przyszły rozwój był możliwy bez ingerencji w dotychczasowy kod.

- # Opis projektu

Na szkielet projektu składają się trzy główne części, które współpracując ze sobą tworzą komplementarny program. Pierwsza z nich to klasa reprezentująca główną postać w grze jaką jest froger. Następnie zaimplmentowałem obiekty znajdujące się oraz poruszające się po planszy. Ostatnim etapem prac było stworzenie interfejsu graficznego oraz poziomów gry które łączą wcześniej opracowane elementy kodu.

- # Opis techniczny

Klasa Frog:

Jest to klasa reprezentująca główną postać w grze, jako użytkownicy poruszamy się nim, w ten sposób prubóbując przejść kolejne poziomy.

Atrybyty klasy:

- lives

Ilość żyć pozostała dla gracza.

- x

Współrzędna x obiektu frog.

- y

Współrzędna y obiektu frog.

- lenght

Długość obiektu frog.

- step

Długość kroku wykonywanego przez froga.

- points

Punkty uzyskane przez gracza.

- image

Obraz jpg reprezentujący obiekt frog.

Metody:

- def image(self)

Getter atrybutu image.

- def lives(self)

Getter atrybutu lives.

- def x(self)

Getter atrybutu x.

- def y(self)

Getter atrybutu y.

- def lenght(self)

Getter atrybutu y.

- def points(self)

Getter atrybutu points.

- def step(self)

Getter atrybutu step.

- def set_x(self, new_x)

Setter atrybutu x.

- def set_y(self, new_y)

Setter atrybutu y.

- def set_lives(self, new_lives)

Setter atrybutu lives.

- def add_points(self, points)

Metoda dodająca punkty po wygranym poziomie.

- def move(self, width, hight)

Metoda odpowiadająca za ruch użytkownika.

- def move_with_object(self, object)

Metoda odpowiadająca za ruch użytkownika podczas gdy znajduje się na innym porszającym się obiekcie.

- def draw_frog(self, surface)

Metoda odpowiadająca za rysowanie froga na ekranie.

- def is_overlaping(self, object_x,

  object_lenght, track_y, hight)

Metoda sprawdzająca czy frog koliduje z innym obiektem.

- def die(self, window_size)

Metoda wywoływana w momencie kolizji, odejmuje życie, resetuje punkty i położenie froga.

- def scale_frog(self, width, hight)

Metoda skalująca obiekt frog do wybranej przez użytkownika wielkości ekranu.

- def load_image(cls)

Metoda ładująca obraz dla obiektu frog.

- def scale_image(cls, width, hight)

Metoda skalująca obraz reprezentujący froga do wybranej wielkości ekranu.

Klasa Moveable:

Klasa moveable reprezentuje obiekty które poruszają się po planszy, w czasie gry, sprawdzana jest kolizja z takimi obiektami, i w zależności od rodzaju obiektu kontakt froga z obiektem klasy moveable, może wywołać inne akcje.

Atrybyty klasy:

- x

Współrzędna x obiektu.

- speed

Szybkość obiektu.

Metody klasy:

- def x(self)

Getter atrybutu x.

- def lenght(self)

Getter atrybutu lenght.

- def set_speed(self, new_speed)

Setter atrybutu speed.

- def move(self, width)

Metoda odpowiedzialna za poruszanie się obiektu.

- def scale_object(self, width)

Metoda odpowiedzialna za skalowanie obiektu.

- def speed(self)

Getter atrybutu speed.

Klasa Car:

Klasa Car dziedziczy po klasie moveable, reprezentuje samochód poruszający się po drodze, w momencie kontaktu frogera z samochodem, froger traci jedno życie i jest cofany do początku aktualnego poziomu.

Atrybyty klasy:

- x

Współrzędna x obiektu.

- speed

Szybkość obiektu.

- lenght

Długość obiektu.

Metody klasy:

- def load_image(cls)

Funkcja ładująca obraz jpg, jako reprezentację obiektów klasy.

- def image(cls):

Getter atrybutu image.

- def scale_image(cls, width, hight):

Metoda skalująca obraz do wielkości okna wybranego przez użytkownika.

Klasa Alligator:

Klasa Alligator dziedziczy po klasie moveable, reprezentuje aligatora pływajacego po rzece, w momencie kontaktu frogera z aligatorem, froger traci jedno życie i jest cofany do początku aktualnego poziomu.

Atrybyty klasy:

- x

Współrzędna x obiektu.

- speed

Szybkość obiektu.

- lenght

Długość obiektu.

Metody klasy:

- def load_image(cls)

Funkcja ładująca obraz jpg, jako reprezentację obiektów klasy.

- def image(cls):

Getter atrybutu image.

- def scale_image(cls, width, hight):

Metoda skalująca obraz do wielkości okna wybranego przez użytkownika.

Klasa Log:

Klasa Log dziedziczy po klasie moveable, reprezentuje kłodę pływajacą po rzece, w momencie kontaktu frogera z kloda, froger zatrzymuje sie na klodzie i znajduje sie w stanie bezpiecznym oraz porusza sie po rzece.

Atrybyty klasy:

- x

Współrzędna x obiektu.

- speed

Szybkość obiektu.

- lenght

Długość obiektu.

Metody klasy:

- def load_image(cls)

Funkcja ładująca obraz jpg, jako reprezentację obiektów klasy.

- def image(cls):

Getter atrybutu image.

- def scale_image(cls, width, hight):

Metoda skalująca obraz do wielkości okna wybranego przez użytkownika.

Klasa Turtle:

Klasa Turtle dziedziczy po klasie moveable, reprezentuje zółwia pływajacego po rzece, w momencie kontaktu frogera z żółwiem, froger zatrzymuje sie na żółwiu i znajduje sie w stanie bezpiecznym oraz porusza sie po rzece.

Atrybyty klasy:

- x

Współrzędna x obiektu.

- speed

Szybkość obiektu.

- lenght

Długość obiektu.

Metody klasy:

- def load_image(cls)

Funkcja ładująca obraz jpg, jako reprezentację obiektów klasy.

- def image(cls):

Getter atrybutu image.

- def scale_image(cls, width, hight):

Metoda skalująca obraz do wielkości okna wybranego przez użytkownika.

Klasa Checkpoint:

Klasa checkpoint reprezentuje miejsce do ktorego musimy dotrzec aby ukonczyc poziom. W kazdym poziomie checkpointy znajduja sie w tych samych miejscach.

Atrybyty klasy:

- x

Wspolrzedna x obiektu checkpoint.

- lenght

Dlugosc obiektu checkpoint.

Metody klasy:

- def image(cls)

Getter atrybutu image.

- def x(self)

Getter atrybutu x.

- def lenght(self)

Getter atrybutu lenght.

- def set_x(self, new_x)

Setter atrybutu x.

Klasa Road:

Klasa road reprezentuje droge znajdujaca sie w poziomie, w każdym poziomie znajduja sie 3 drogi, po ktorych poruszaja sie samochody.

Atrybyty klasy:

- y

Wysokosc na ktorej znajduje sie droga.

- cars

Lista samochodow poruszajcych sie po drodze.

Metody klasy:

- def y(self):

Getter atrybutu y.

- def cars(self):

Getter atrybutu cars.

- def move_cars(self, width):

Metoda odpowiadajca za poruszanie sie wszystkich samochodow po drodze.

- def draw_road(self, surface):

Metoda odpowiadajaca za rysowanie wszytkich elementow na drodze.

- def scale_road(self, width, hight):

Metoda odpowiadajaca za skalowanie wszytkich elmentow drogi.

- def check_safety_on_the_road(self, frog, hight):

Metoda sprawdzajaca czy frog koliduje z jakimkolwiek samochodem na drodze.

Klasa River:

Atrybyty klasy:

- y

Wysokosc na ktorej znajduje sie rzeka.

- alligators

Lista aligatorow ktore plywaja po rzece.

- logs

Lista kłód ktore plywaja po rzece.

- turtles

Lista żółwi ktore plywaja po rzece.

Metody klasy:

- def y(self):

Getter atrybutu y.

- def alligators(self):

Getter atrybutu alligators.

- def logs(self):

Getter atrybutu logs.

- def turtles(self):

Getter atrybutu turtles.

- def move_objects(self, width):

Metoda odpowiadajaca za poruszanie wszystkich obiektow na rzece.

- def draw_river(self, surface):

Metoda odpowiadajaca za rysowanie wszytkich elementow na rzece.

- def scale_river(self, width, hight):

Metoda odpowiadajaca za skalowanie wszytkich elmentow drogi.

- def check_safety_in_river(self, frog, hight):

Metoda sprawdzajaca czy frog koliduje z jakimkolwiek obiektem na drodze.

Klasa Final_line:

Klasa reprezentujaca ostania linie na drodze, znajduja sie na niej obiekty klasy checkpoint do ktorych musimy dotrzec aby ukonczyc poziom.

Atrybyty klasy:

- y

Wysokosc na ktorej znajduje sie rzeka.

- checkpoints

Lista checkpoint'ow poruszajcych sie po drodze.

Metody klasy:

- def y(self):

Getter atrybutu y.

- def checkpoints(self):

Getter atrybutu checkpoints.

Klasa User:

Atrybuty klasy:

- name

Nazwa użytkownika.

- window_width

Szerokość okna wybrana przez użytkownika.

- window_hight

Wysokość okna wybrana przez użytkownika.

- level

Aktualny poziom użytkownika.

- points

Punkty uzyskane przez użytkownika.

- width_scale

Mnożnik skalowania okna w szerokosci.

- hight_scale

Mnożnik skalowania okna w wysokosci.

Metody klasy:

- def name(self):

Getter atrybutu name.

- def window_hight(self):

Getter atrybutu window_hight.

- def window_width(self):

Getter atrybutu window_width.

- def set_window_size(self, width, hight):

Setter atrybutu window_size.

- def level(self):

Getter atrybutu level.

- def points(self):

Getter atrybutu points.

Klasa Level:

Atrybuty klasy:

- frog

Obiekt klasy frog, podczas gry poruszamy nim, staramy sie przejsc do jednego z checkpoints.

- roads

Lista dróg znajdujących się na mapie.

- rivers

Lista rzek znajdujących się na mapie.

- checkpoints

Lista checkpoints znajdujących się na mapie.

- time

Czas na rozegranie poziomu.

Metody klasy:

- def frog(self):

  Getter atrybutu frog.

- def roads(self):

Getter atrybutu roads.

- def rivers(self):

Getter atrybutu rivers.

- def checkpoints(self):

Getter atrybutu checkpoints.

- def time(self):

Getter atrybutu time.

- def move_all_elements(self, window_size):

Metoda poruszajca wszystkie elementy na mapie.

- def scale_level(self, width, hight):

Metoda skalowania całego pzoiomu.

- def check_colisions_with_cars(self, window_size):

Metoda sprawdzajaca czy frog koliduje za jakimkolwiek samochodem.

- def check_safety_in_river(self, window_size):

Metoda sprawdzajaca bezpieczenstwo froga w rzece.

- def check_win(self, window_size, timer):

Metoda sprawdzajaca czy frog uzyskal zwyciestwo.

- def print_subtitles(self, window_size, timer):

Metoda wypisujaca informacje do użytkownika na ekranie.

- def draw_all_elements(self):

Metoda rysujaca wszystkie elementy na drodze.

- def play_level(self, window_size, timer):

Metoda odpowiadajaca za rozegranie calego poziomu.

Klasa levelFactory:

Klasa levelFactory odpowiada za inicjacje oraz przekazanie do funkcji Main obiektów poziomów koniecznych do przeprowadzenia rozgrywki.

Wykorzystane moduły

Do stworzenia programu wykorzystaliśmy biblioteki:

- pygame
- time
- falke8
- py

- pytest
- os.path
- argparse
- json

Testy

Stworzony przeze mnie program przeszedł 41 testów, które dotyczą każdego modułu mojego programu. Ponadto postanowiłem wykonać testy manualne. W ten sposób uzyskałem pewność że program umożliwia grę dla użytkownika, jednak zauważyłem także pewne problemy.

Problemy

Przeprowadzajac testy manualne, zauważyłem że poziom trudności wzrasta skokowo, jest to pewien problem gdyż mimio prób nie udało się naprawić tego problemu, możliwe że aby ilość poziomów mogła być większa do gry powinny być dodane kolejne elementy które zmieniały by się w zależności od poziomu.

Podsumowanie

Cały projekt został przez nas napisany w języku Python. Ze względu na małe doświadczenie w programowaniu, oraz specyfikę języka napotkałem pewne problemy, które jednak ostatecznie udało się rozwiązać. Efektem mojej pracy jest produkt, który mam nadzieje, że odpowiada na temat, który został mi przydzielony. Wierze, że może być on konkurencyjny w porównaniu do rozwiązań proponowanych przez studentów z podobnym do mojego doświadczeniem.

Sterowanie i pruszanie sie po grze

Grę mozemy uruchomic z poziomu VisualStudioCode lub z terminala, w przypadku uruchamiania gry z terminala mozna podac dwa opcjonalne argumenty, --restet przyrwacajacy podstawowe ustawienia i wracajacy do pierwszego poziomu, oraz afrgument --change_window_size zmieniajacy rozmiar okna na ktorym bedziemy grac.

W samej grze poruszamy sie za pomoca przyciskow: w,a,s,d. Naszym zadaniem jest dojscie na sam koniec planszy, w ten sposob przejdziemy dany poziom. Po ukonczeniu ostatniego poziomu, gra wyswietli okno koncowe, informujace nas o tym ze doszlismy do konca.