Filip Browarny, 318637

Cel i opis projektu

Celem projektu było stworzenie prostej wersji znanej każdemu gry Frogger.
W grze wcielamy się w żabę, która musi pokonać trasę na drugą stronę mapy. Plansza dzieli się na kilka części. Drogę, po której jeżdżą pojazdy, kontakt z pojazdem oznacza utratę życia. Rzekę, po której pływają różnej wielkości obiekty. Gracz musi poruszać się po mapie tak aby zawsze mieć kontakt z jednym z pływających obiektów. Na planszy znajdują się także trzy żółte bezpieczne strefy, na początku, pomiędzy drogą, a rzeką i na końcu. Dojście do ostatniej żółtej strefy oznacza przejście mapy oraz załadowanie nowego trudniejszego poziomu. Na ekranie, nad strefą, w której porusza się gracz, znajdują się napisy informujące o pozostałych, życiach, obecnym wyniku, najwyższym wyniku oraz czasie. Koniec czasu oznacza utratę życia oraz powrót na początek mapy. W grze poruszamy się strzałkami.

Opis plików

Program składa się z 5 plików pythonowych oraz pliku ttf. Opis argumentów, które przyjmują poszczególne klasy jest zamieszczony w dokumentacji wewnątrz kodu.

- Life_layout.py
  - 5 funkcji służących do prostej konfiguracji kilku parametrów gry.
  - Klasa HeadLine
    - Służy do generowania nagłówka zawierającego podstawowe informacje.
  - Klasa Frame
    - Służy do generowania ramki wokół mapy.
  - Klasa Safe_Zone_Water
    - Służy do generowania bezpiecznych stref oraz wody.
- Enemies_friends_frog.py
  - Klasa Car
    - Obiekt podstawowego pojazdu, zawiera metody służące do poruszania się oraz „przesuwania” auta na początek drogi, jeśli wyszedł poza mapę.
  - Klasy Buldozer, Race_car
    - Dziedziczy po Car
    - Delikatnie zmodyfikowane podstawowe auta. Odpowiednio większe i szybsze.
  - Klasa Friend
    - Dziedziczy po Car
    - „Pomocnicy” w przechodzeniu rzeki
  - Klasa Frog
    - Dziedziczy po Car
    - Jest to postać, którą porusza gracz.
- Logic.py
  - Klasa Logic
    - Odpowiada za podstawy logiki gry, generowanie, poruszanie obiektów, detekcja zdarzeń, obsługa żyć i zegara.

        - Metody car_init oraz friends_init służą do generowanie odpowiednio aut oraz obiektów pływających po rzece. Obiekty są generowane na losowej pozycji, mają zmienne wielkości oraz prędkość.
        - Metody clock_time oraz car_tick służą do liczenia czasu.
- Run.py
    - Klasa Game
        - Odpowiada za obsługę zdarzeń, generowanie mapy i inne rzeczy związane z interfejsem.
        - Metoda events służy do przechwytywania zdarzeń i obsługi ich.
        - Metoda draw służy do rysowania ekranu.
- Test_frogger.py
    - Testy jednostkowe
- ARCADECLASSIC.TTF
    - Plik zawierający czcionkę wykorzystaną w nagłówku.

Obsługa programu

Do uruchomienia gry potrzebne są biblioteki PyGame oraz random. Po pobraniu bibliotek wystarczy odpalić plik run.py w interpreterze.

Pomysł

Projekt można było zrealizować w konsoli albo w GUI. Zdecydowałem się na interfejs graficzny. Dzięki czemu nawet prosta gra będzie dużo przyjemniejsza dla użytkownika. Do projektu wybrałem bibliotekę PyGame. Używanie gotowej biblioteki służącej do tworzenia gier, daje nam dużo gotowych rozwiązań, co ułatwi pracę. PyGame wydał się także stosunkowo prosty do nauczenia. Pierwotny pomysł na rozwiązanie był bardzo podobny do wersji ostatecznej.

Problemy

Duży kłopot sprawiła mi refaktoryzacja kodu. Pierwsza wersja kodu nie miała klasy Logic. Wszystkie metody były w klasie Game. Do tego wiele linii kodu powtarzało się. Ostateczna wersja jest według mnie lepsza, chociaż prawdopodobnie kod można by napisać dużo lepiej. Problematyczne okazało się także testowanie. Być może jest to efekt źle napisanego kodu, ale do wielu funkcji nie potrafiłem zrobić dobrych testów, stąd też jest ich tak mało.

Estetyka programu

Jednym z elementów, który na pewno bym zmienił, gdybym miał dalej rozwijać program jest dodanie modeli postaci, zamiast kwadratów i prostokątów. Myślę, że taka zmiana mocno zmieniłaby odczucia estetyczne. Nie zrobiłem tego z kilku powodów. Po pierwsze nie mam zdolności graficznych. Po drugie tutorial, który oglądałem dotyczący dodawania grafik wektorowych do pygame'a wydawał się dość skomplikowany. Po trzecie wolałem się skupić na grywalności i usunięciu wszystkich błędów, bardziej niż na estetyce.

Wady

Jednym z elementów, który można by poprawić jest balans. Przede wszystkim rzeka. Od samego początku chciałem, żeby wszystkie elementy były generowane losowo. Tak też zaimplementowałem pojawianie się „pomocników" w przechodzeniu rzeki. Plusem takiego rozwiązania jest większa

różnorodność. Minusem to, że na pierwszych poziomach, przy niekorzystnym rozkładzie, trzeba długo czekać na możliwość przejścia. Dlatego pierwsze kilka leveli może wydawać się żmudne. Ostatnią wadą jest zła budowa plików, o której wspomniałem wcześniej.

Zalety

Subiektywnie uważam, że odczucia z gry są przyjemne. Gra jest dynamiczna, po porażce od razu przechodzi się do kolejnej rozgrywki. Nie ma żadnych ekranów czy wyborów, które mimo, że trwają kilka sekund są zawsze irytujące dla gracza, który po porażce ma ochotę od razu zacząć kolejną partię.