Dokumentacja pliku Game.py

**Opis ogólny**

Plik Game.py implementuje główną logikę gry wyścigowej przy użyciu biblioteki pygame. Gra składa się z wyścigu pomiędzy graczem a komputerowymi przeciwnikami (botami), którzy rywalizują na torze stworzonym z kawałków powierzchni. Gra obsługuje kolizje, radia, zegar odliczający oraz liczenie i wyświetlanie wyników.

**Klasy**

**1. Game**

Główna klasa gry. Zawiera logikę dotyczącą tworzenia gry, zarządzania wyścigiem, sterowania samochodami, detekcji kolizji, oraz renderowania wszystkich elementów gry na ekranie.

**Atrybuty:**

* PLAYER\_NICK: Nick gracza.
* CHOSEN\_LEVEL: Wybrany poziom (1, 2 lub 3).
* running: Flaga określająca, czy gra jest aktywna.
* WHITE, BLACK, YELLOW: Kolory wykorzystywane do renderowania.
* window: Okno gry.
* WINDOW\_WIDTH, WINDOW\_HEIGHT: Rozmiar okna.
* clock: Zegar sterujący FPS.
* pressed\_keys: Zbiór wciśniętych klawiszy.
* font, bigger\_font: Czcionki do wyświetlania tekstów.
* radio: Obiekt radia.
* buffered: Powierzchnia buforująca tor w celu poprawy wydajności.
* track: Lista kawałków toru.
* player: Obiekt reprezentujący samochód gracza.
* enemies: Krotka przeciwników (botów).
* allCars: Wszystkie samochody na torze (gracz + boty).
* max\_laps: Maksymalna liczba okrążeń.
* scores: Lista wyników czasowych samochodów.
* winners: Lista samochodów, które ukończyły wyścig.
* begginingTime: Czas rozpoczęcia wyścigu.
* oil\_img: Ikonka oleju.
* sorted\_cars: Posortowana lista samochodów według wyników.
* car\_names: Słownik przypisujący nazwy do samochodów (gracz i boty).
* is\_countdown\_finished: Flaga wskazująca, czy odliczanie się zakończyło.

**Metody:**

* \_\_init\_\_(self, window, player\_nick, chosen\_level): Inicjalizuje wszystkie atrybuty gry, tworzy tor, ustawia samochody, ładuje obrazki, itp.
* handle\_events(self): Obsługuje zdarzenia, takie jak naciśnięcie klawiszy i kliknięcie myszką.
* countdown(self): Funkcja odliczająca czas przed rozpoczęciem wyścigu.
* draw(self): Rysuje wszystkie elementy gry na ekranie, takie jak samochody, tło, napisy z wynikami, itp.
* check\_winner(self, oCar): Sprawdza, czy samochód ukończył wyścig, i aktualizuje wyniki.
* aproximate\_scores(self): Szacuje wyniki przeciwników, którzy nie ukończyli jeszcze wyścigu.
* check\_collisions(self, oCar: Car): Sprawdza kolizje między samochodami.
* sort\_cars(self): Sortuje samochody według wyników wyścigu.
* game\_loop(self): Pętla główna gry, odpowiedzialna za aktualizację gry, rysowanie elementów, obsługę zdarzeń, itp.

**Zależności**

* pygame: Biblioteka do tworzenia gier 2D.
* ai: Moduł odpowiedzialny za logikę sterowania botami.
* Track: Moduł zawierający definicje toru.
* Radio: Moduł odpowiedzialny za zarządzanie radiem w grze.
* time: Biblioteka umożliwiająca obsługę czasu, np. do mierzenia czasów okrążeń.
* random: Moduł do generowania losowych liczb (np. przy losowaniu piosenek w radiu).

**Szczegóły implementacji**

1. **Inicjalizacja gry**
   * Gra jest tworzona przez wywołanie metody \_\_init\_\_, która ustawia wszystkie elementy, takie jak samochody, tor, tło, licznik okrążeń, itp.
2. **Zdarzenia**
   * Obsługa zdarzeń jest realizowana w metodzie handle\_events(), która reaguje na naciśnięcia klawiszy i kliknięcia myszy.
   * Gracz może używać klawiszy do sterowania, np. R do włączenia/wyłączenia radia, ESC do przerwania gry.
3. **Liczenie okrążeń i wyników**
   * Każde auto (gracz oraz boty) przechowuje informacje o swoim czasie okrążenia i postępie w wyścigu.
   * Po zakończeniu wyścigu sprawdzane są wyniki, a następnie są one wyświetlane na ekranie.
4. **Odliczanie przed wyścigiem**
   * Metoda countdown() odpowiada za wyświetlanie odliczania przed rozpoczęciem wyścigu (3, 2, 1).
5. **Rysowanie**
   * Funkcja draw() odpowiedzialna jest za renderowanie wszystkich elementów gry na ekranie, w tym samochodów, toru, tekstów wyników oraz innych interfejsów graficznych.
6. **Detekcja kolizji**
   * Funkcja check\_collisions() sprawdza kolizje między samochodami, a także modyfikuje prędkość i pozycję pojazdów, aby symulować efekt odbicia.
7. **Radia**
   * Gra obsługuje radio, które odtwarza losowo wybrane utwory. Funkcje toggle\_radio() oraz draw() są odpowiedzialne za włączenie/wyłączenie radia oraz jego renderowanie na ekranie.
8. **Sortowanie samochodów**
   * Na końcu wyścigu samochody są sortowane na podstawie wyników (czasów) przy użyciu metody sort\_cars().

**Użycie**

Plik Game.py jest uruchamiany w kontekście całej gry, gdzie inicjowanie gry odbywa się w głównym pliku (np. Main.py). Gracz wybiera poziom, wprowadza swój nick, a następnie rozpoczyna wyścig, który odbywa się na jednym z trzech dostępnych torów.

**Podsumowanie**

Plik Game.py stanowi rdzeń gry wyścigowej, implementując logikę sterowania, wyścigu, kolizji, wyników, a także elementy rozrywki jak radio. Obsługiwane są także interakcje z graczem poprzez klawiaturę i myszkę.