Programowanie obiektowe

Polskie Waymo

Wojciech Legierski

1. Wstęp

1.1. **Cel dokumentu:** Dokumentacja ta ma na celu dostarczenie informacji zarówno dla programistów rozwijających projekt, jak i dla użytkowników korzystających z symulatora. Opisuje ona funkcjonalności programu, architekturę opartą na programowaniu obiektowym, algorytmy, obsługę sterowania oraz plany rozwoju.

1.2. **Zakres projektu:** "Polskie Waymo" to symulator jazdy samochodem, umożliwiający użytkownikowi sterowanie pojazdem w wirtualnym środowisku. Symulacja uwzględnia jazdę po trasie, zjazdy z trasy z efektem spowolnienia, zmianę biegów oraz pomiar czasu przejazdu.

2. Wymagania

2.1. Wymagania funkcjonalne:

2.1.1. Sterowanie pojazdem

- Użytkownik może skręcać.
- Użytkownik może przyspieszać i hamować.
- Użytkownik może zmieniać biegi manualnie.

2.1.2. Symulacja jazdy

- Symulator odwzorowuje fizykę jazdy - przyspieszenie, hamowanie.
- Symulator wyświetla otoczenie.
- Symulator generuje dźwięki silnika.

2.1.3. Interakcja z trasą

- Symulator wyświetla mapę trasy.
- Symulator informuje o zbliżających się skrętach i przeszkodach.
- Symulator rejestruje czas przejazdu.

2.1.4. Interakcja z interfejsem użytkownika (UI)

- Użytkownik może konfigurować ustawienia symulacji - widok kamery.
  .
- Symulator wyświetla informacje o prędkości, obrotach silnika.

2.1.5. Zdarzenia i reakcje

- Wykroczenie poza trasę powoduje spowolnienie pojazdu.

2.2. Wymagania niefunkcjonalne

2.2.1. Wydajność

- Symulator powinien działać płynnie z co najmniej 60 klatkami na sekundę na komputerach o określonej konfiguracji.
- Czas ładowania trasy nie powinien przekraczać 2 sekund.
- Opóźnienie między akcją użytkownika a reakcją symulatora powinno być minimalne.

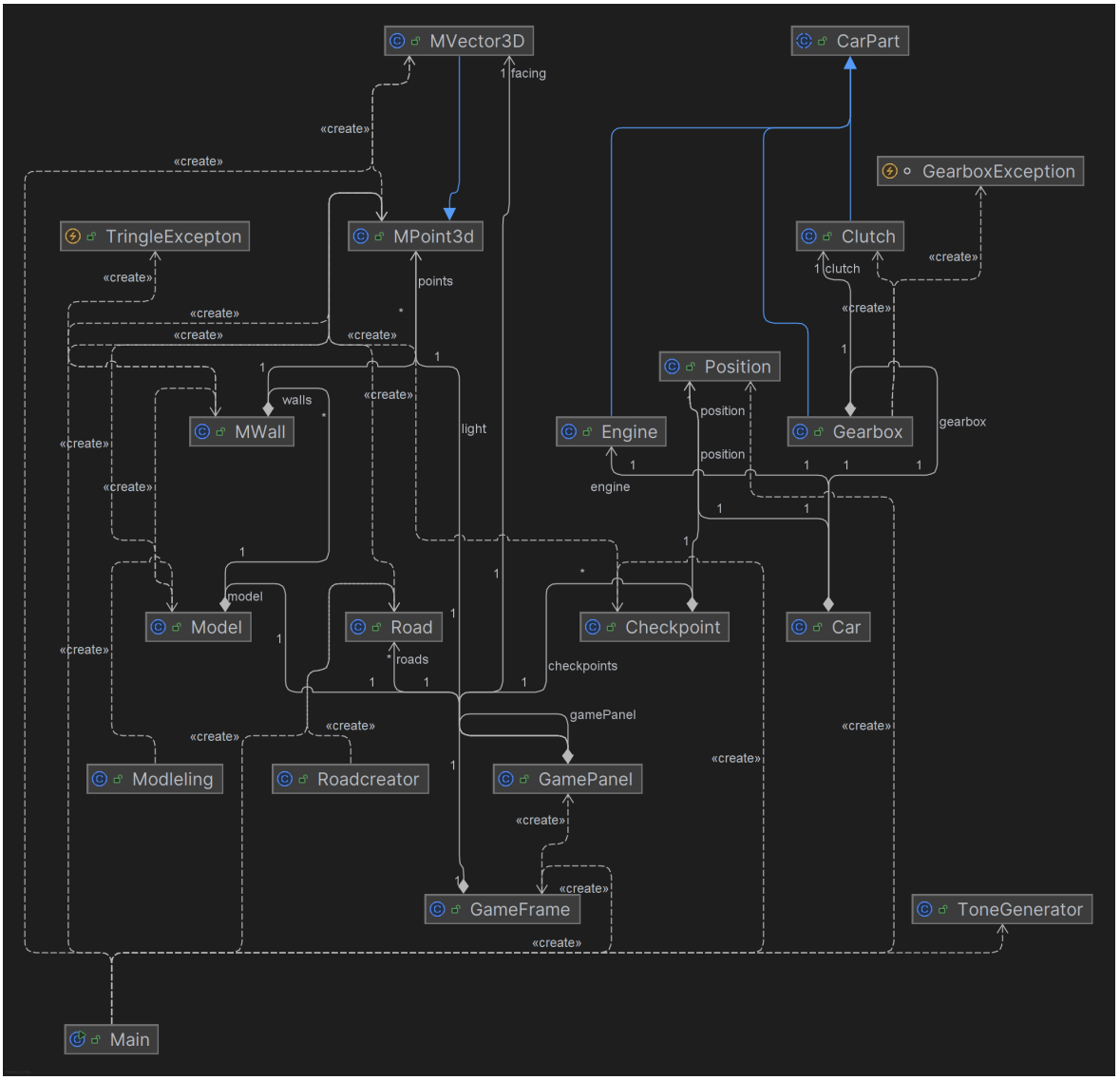
2.2.2. Użyteczność (Usability)

- Interfejs użytkownika powinien być intuicyjny i łatwy w obsłudze.
- Sterowanie pojazdem powinno być realistyczne i responsywne.
- Dokumentacja i instrukcje powinny być jasne i zrozumiałe.

2.2.3. Niezawodność (Reliability)

- Symulator powinien być stabilny i nie powodować awarii.
- System powinien być odporny na błędy użytkownika.

3. Architektura i Struktura Klas



4. Architektura i Wzorce Projektowe

Bulder zastosowany w klasie Car.

Fasade z zastosowany w klasie Game.

5. Algorytmy i logika

5.1. Obroty silnika

5.2. Przyspieszenie

prezlicza w dnaym moencie moment obrotowy na siłe po przez mnozeniei srednice koła oraz przełozenie puzniej dzielone jest to przez mase

5.3. Przemieszczanie

od przyspiesinia odejmywna jesst suma oporó tocznenia oraz aurodynamiczny dzielony dzielone przez mase

5.4. Render

Klasa **MVector3d** implementuje podstawowe operacje na wektorach, w tym:

1. **Iloczyn wektorowy (cross product)** – oblicza wektor prostopadły do dwóch zadanych wektorów.
2. **Iloczyn skalarny (dot product)** – zwraca skalarną miarę "równoległości" wektorów.
3. **Długość wektora**: $\|v\| = \sqrt{(vx^2 + vy^2 + vz^2)}$.
4. **Odwrócenie wektora** – zmiana kierunku wektora na przeciwny.
5. **Ustawienie długości** – normalizacja wektora i pomnożenie przez zadaną wartość.
6. **Kąt między wektorami** – obliczenie sinus i cosinus kąta między dwoma wektorami.
    - Uwzględniany jest pełny zakres 360° (2π), przy założeniu, że podany jest dodatkowy wektor prostopadły do wektora, względem którego liczony jest kąt.
    - Mechanizm umożliwia określenie współrzędnych biegunowych, gdzie płaszczyzna i wektor definiują "górę" układu odniesienia.

Klasa **MVector3d** implementuje podstawowe operacje na wektorach, w tym:

1. **Iloczyn wektorowy (cross product)** – oblicza wektor prostopadły do dwóch zadanych wektorów.
2. **Iloczyn skalarny (dot product)** – zwraca skalarną miarę "równoległości" wektorów.
3. **Długość wektora**: $\|v\| = \sqrt{(vx^2 + vy^2 + vz^2)}$.
4. **Odwrócenie wektora** – zmiana kierunku wektora na przeciwny.
5. **Ustawienie długości** – normalizacja wektora i pomnożenie przez zadaną wartość.

6. **Kąt między wektorami** – obliczenie sinus i cosinus kąta między dwoma wektorami.
    - Uwzględniany jest pełny zakres 360° (2π), przy założeniu, że podany jest dodatkowy wektor prostopadły do wektora, względem którego liczony jest kąt.
    - Mechanizm umożliwia określenie współrzędnych biegunowych, gdzie płaszczyzna i wektor definiują "górę" układu odniesienia

Ogniskowa kamery jest określana na podstawie:

- **Długości ogniskowej (focalLength)** – odległości ogniskowej od płaszczyzny projekcji,
- **Punktu ogniskowego**, obliczanego jako:
  $Pf = P_0 - facing * focalLength$,
  gdzie facing to wektor normalny płaszczyzny.

Punkt góry kamery

Punkt "góry" (Ptop) kamery:

- Jest definiowany względem płaszczyzny,
- Aby upewnić się, że znajduje się na płaszczyźnie, jest rzutowany podobnie jak inne punkty.

Rzutowanie punktów na ekran przebiega następująco:

1. **Wyznaczenie prostej**, przechodzącej przez punkt P oraz punkt ogniskowy Pf.
   Równania parametryczne prostej:
   $x(t) = Px + \Delta x * t$,
   $y(t) = Py + \Delta y * t$,
   $z(t) = Pz + \Delta z * t$,
   gdzie $\Delta x = Pfx - Px$, $\Delta y = Pfy - Py$, $\Delta z = Pfz - Pz$.
2. **Wyznaczenie parametru t**, w którym prosta przecina płaszczyznę projekcji:
   $t = -(A * Px + B * Py + C * Pz + D) / (A * \Delta x + B * \Delta y + C * \Delta z)$.
3. **Obliczenie współrzędnych punktu rzutowanego**:
   $Pr = (xr, yr, zr) = (Px + \Delta x * t, Py + \Delta y * t, Pz + \Delta z * t)$.
4. **Zmiana współrzędnych kartezjańskich na biegunowe**:
   $r = \|v\|$,
   $\alpha = \arccos((v \cdot g) / (\|v\| * \|g\|))$,
   gdzie **v** to wektor od środka płaszczyzny do punktu, a **g** to wektor "góry".
   Współrzędne biegunowe konwertuje się na kartezjańskie:
   $x = r * \sin(\alpha)$,
   $y = r * \cos(\alpha)$.

Ściany (zdefiniowane przez zbiór punktów) są sortowane według odległości od źródła (ogniskowej).

Każda ściana jest rysowana jako wielokąt na ekranie, gdzie:

- Kolor ściany zależy od kąta między wektorem normalnym ściany (“skierowanym w stronę ogniskowej”) a wektorem od środka ściany do źródła światła.
- Używana jest także odległość ściany od źródła światła.

Cienie są obliczane przez rzutowanie ścian na płaszczyznę cieni (np. podłogę), przy użyciu uprocznej metody rzutowania punktów na płaszczyznę projekcji pnieważ zutejmy zawsze na płaczyzne z=0 a zuujemy wgledm punktu światła.

6. Instrukcja Użytkownika

6.1. Wprowadzenie:

Niniejsza instrukcja opisuje zasady sterowania pojazdem w symulatorze jazdy. Celem gry jest pokonanie okrążenia trasy w jak najkrótszym czasie.

6.2. Sterowanie:

6.2.1. Kierowanie:

- **Strzałka w lewo:** Skręt w lewo.
- **Strzałka w prawo:** Skręt w prawo.

6.2.2. Przyspieszenie/Hamowanie:

- **Strzałka w górę:** Przyspieszenie (dodawanie gazu).
- **Strzałka w dół:** Hamowanie.

6.2.3. Zmiana biegów (skrzynia manualna):

- **Shift:** Wciśnięcie sprzęgła. Należy pamiętać o wciśnięciu sprzęgła przed zmianą biegu.
- **E:** Zmiana na wyższy bieg.
- **Q:** Zmiana na niższy bieg.

6.3. Zasady gry:

6.3.1. Trasa:

W grze dostępna jest jedna, zamknięta trasa, którą należy pokonać w jak najkrótszym czasie.

6.3.2. Prędkość:

- **Jazda po drodze:** Podczas jazdy po wyznaczonej trasie, prędkość pojazdu będzie rosła wraz z wciśniętym pedałem gazu (strzałka w górę).
- **Zjazd z trasy:** W przypadku zjechania z trasy (np. na trawę, pobocze), prędkość pojazdu zostanie znacząco zmniejszona. Należy unikać zjeżdżania z trasy, aby osiągnąć jak najlepszy czas.

6.3.3. Pomiar czasu:

Czas przejazdu jest mierzony od momentu startu do momentu ponownego przekroczenia linii startu/mety. Na ekranie wyświetlany jest aktualny czas przejazdu.

6.3.4. Wygrana:

Gracz, który uzyska najkrótszy czas okrążenia, wygrywa.

6.4. Dodatkowe wskazówki:

6.4.1. Płynne sterowanie kierownicą i pedałami pozwoli na utrzymanie optymalnej prędkości i uniknięcie poślizgów.

6.4.2. Umiejętne korzystanie ze zmiany biegów (sprzęgło, E i Q) pozwoli na lepsze przyspieszenie i hamowanie, co jest kluczowe do osiągnięcia dobrego czasu.

6.4.3. Obserwuj trasę i staraj się utrzymać na niej jak najdłużej, aby uniknąć spowolnienia.

6.4.4. Ćwicz i eksperymentuj z różnymi stylami jazdy, aby znaleźć optymalny sposób pokonywania trasy.

6.5. Interfejs użytkownika (UI):

6.5.1. **Ekran gry:** Wyświetla trasę, pojazd gracza oraz informacje o prędkości, aktualnym biegu i czasie przejazdu.

6.6. Przykładowy scenariusz:

6.6.1. Uruchom program.

6.6.2. Rozpocznij grę.

6.6.3. Używaj strzałek do kierowania i przyspieszania/hamowania.

6.6.4. Używaj klawiszy Shift, E i Q do zmiany biegów.

6.6.5. Staraj się utrzymać na trasie i pokonać okrążenie w jak najkrótszym czasie.

6.6.6. Po przekroczeniu linii startu/mety, na ekranie zostanie wyświetlony Twój czas.