Autorzy:
Karol Pieczka
Kacper Górka

Prowadzący:
mgr inż. Krzysztof Gracki

DOKUMENTACJA PROJEKTOWA

METRO

PROGRAMOWANIE OBIEKTOWE

1. Wstęp

Motywacja

Główną motywacją projektu jest chęć zwiększenia efektywności transportu publicznego w obliczu rosnącej liczby mieszkańców. Obecnie wzrasta potrzeba coraz wydajniejszego źródła transportu. Jednym z nich jest metro, które zapewnia podróż na innej płaszczyźnie i nie koliduje z transportem naziemnym. Stworzenie wirtualnego modelu pozwoli na dogłębną analizę działania tej instytucji.

Cel projektu

Celem projektu jest dostarczenie symulacji ruchu pociągów i przepływu pasażerów przez stacje. Na jego podstawie, definiując różne parametry, można sprawdzić zachowanie organizacji w różnych sytuacjach. Wyniki symulacji mają wartość zarówno edukacyjną jak i badawczą. Projekt jest cennym narzędziem dydaktycznym dla studentów inżynierii transportu jak i pozwoli badaczom nad odpowiednim planowaniem transportu miejskiego.

2. Architektura programu

Przedstawienie klas systemu

Do zaprojektowania instytucji posłużono się klasami reprezentującymi rzeczywiste obiekty.

klasa

Pasażer

Klasa reprezentująca osoby przychodzące na stację.

- Każda instancja przy tworzeniu obiektu jest przypisywana do konkretnej stacji. Ustalany jest cel podróży jak i pomocniczo kierunek jako stacja końcowa. Na podstawie kierunku pasażer wie do którego pociągu ma wsiąść.
- W symulacji tworzy się unikalny wskaźnik do instancji obiektu. Jest to bardzo istotne, ponieważ pozycja wskaźnika na stacji lub w pociągu obrazuje rzeczywiste położenie pasażera, który nie może się znajdować w dwóch miejscach w tym samym czasie.

klasa

Pociąg

Odzwierciedla skład pociągu, który porusza się po torze.

- Zawiera w sobie listę przewożonych pasażerów (unikalnych wskaźników do obiektów)
- Ma w sobie pola pozwalające określić czy znajduje się na jakiejś stacji, jaka jest stacja końcowa na danym torze.
- Pociąg ma ustalony status czy się porusza lub stoi oraz czy znajduje się na stacji.
- Istotnymi funkcjami jest przyjęcie listy pasażerów na pokład lub ich wysadzenie na stacji docelowej w postaci zwrócenia listy.

klasa

Stacja

Reprezentuje rzeczywiste stacje Metra Warszawskiego.

- Generowany losowo pasażer natychmiast przypisywany jest konkretnej stacji poprzez wstawienie go do listy.
- Stacja zawiera również listę pasażerów, którzy wysiedli z pociągu na docelową stację.
- Głównymi metodami w tej klasie są: przyjęcie pasażera przychodzącego na stację, oddanie pasażera do pociągu oraz przyjęcie pasażera z pociągu.

klasa

Tor

Odzwierciedla trasę po, której poruszają się pociągi

- Do toru przypisane są stacje oraz znajdujące się na nim pociągi
- Ruch pojazdów przedstawiono jako przemieszczanie się wskaźnika (do obiektu pociągu) po wektorze reprezentującym tor.
- Tor zawiera w sobie metody, które wprawiają w ruch pociągi oraz inicjujące wymianę pasażerów między stacją i pojazdem
- Do tworzenia toru biegnącego przez te same stacje ale w przeciwnym kierunku wykorzystuje się specjalną metodę.

klasa

Symulacja

Klasa zawierające metody pozwalające zarządzać symulacją

- Jako argument tej funkcji podaje się tor, który będzie podlegał symulacji oraz czas odstępu pomiędzy kolejnymi iteracjami.
- Powyższa klasa odpowiada za generowanie losowej ilości pasażerów przychodzących na stację
- W klasie zawarte są metody wywołujące metody określone w klasie Tor, które powodują ruch i wymianę pasażerów

Działanie programu z perspektywy pasażera

- Generowanie Pasażerów

W każdej iteracji symulacji generowani są nowi pasażerowie. Proces ten odbywa się za pomocą specjalnie zaprojektowanej funkcji, która losowo przydziela każdemu pasażerowi stację początkową i końcową. Na podstawie tych informacji, pasażer określa kierunek swojej podróży, co pozwala mu na podjęcie decyzji o wyborze odpowiedniego pociągu.

- Proces Wsiadania

Gdy pociąg dociera na stację, pasażerowie czekający na danej stacji są przesiewani w zależności od kierunku ich podróży. Ci, których kierunek zgadza się z kierunkiem pociągu, wsiadają do niego. Ten moment wsiadania oznacza przeniesienie pasażerów z listy oczekujących na stacji do listy pasażerów znajdujących się w pociągu.

- Podróż Pasażera

Po wejściu do pociągu, pasażerowie kontynuują podróż aż do osiągnięcia swojej stacji docelowej. W trakcie podróży pasażerowie są częścią listy pasażerów w pociągu, co pozwala na śledzenie ich obecności i ruchu w ramach systemu transportowego.

- Proces Wysiadania

Gdy pociąg dociera na stację końcową pasażera, następuje proces wysiadania. Pasażerowie, dla których jest to stacja docelowa, są przenoszeni z listy pasażerów w pociągu na listę pasażerów opuszczających stację. Jest to kluczowy moment, który oznacza zakończenie ich podróży w ramach symulacji.

Funkcjonalności programu

- Wczytywanie danych z pliku
- Wyświetlanie wyników na konsoli
- Zapis wyników do pliku
- Możliwość budowy własnej linii metra

  Użytkownik ma możliwość tworzenia torów o dowolnej długości, na których może umieścić dowolną liczbę stacji oraz pociągów. Każdy pociąg może posiadać różne, indywidualne parametry. Długość toru jest elastyczna i może być dostosowana do potrzeb użytkownika, co pozwala na symulację zarówno krótkich, miejskich odcinków, jak i długich tras międzymiastowych. Stacje mogą być rozmieszczone w dowolnych miejscach na torze, co umożliwia precyzyjne odwzorowanie rzeczywistych układów kolejowych lub tworzenie zupełnie nowych, niestandardowych konfiguracji.

- Mechanizm obsługi wyjątków

Program obsługuje 6 rodzajów wyjątków. Są one wyrzucane gdy:

  - Identyfikatory dla w zbiorze stacji lub pociągów powtarzają się
  - Wartość szybkości, symulacji lub liczby iteracji nie jest dodatnia
  - Zostanie podjęta próba inicjalizacji pociągu lub stacji na tej samej pozycji

- Pociąg lub stacja zostaną zainicjalizowane zbyt blisko siebie
- Wskazana pozycja jest poza obrębem toru

3. Wykorzystane technologie

Główne narzędzia

Językiem programowania w projekcie był **C++**, w którym wykorzystano jego możliwości obiektowe. Do śledzenia zmian w kodzie źródłowym wykorzystano system kontroli wersji **Git**.

Biblioteki

Do budowy projektu wykorzystano standardową bibliotekę STL. Zastosowano w nim struktury danych takie jak wektory i listy oraz pary. Obiekty tworzono jako inteligentne wskaźniki. Pojawiły się też funkcje do obsługi strumienia wejścia-wyjścia.

4. Obsługa programu

Definiowanie pliku wejściowego

Struktura pliku wejściowego

W każdej linii definiujemy obiekty. Na początku należy zainicjalizować szybkość symulacji, liczbę iteracji oraz długość toru podając liczby po przecinku. Przykładowo:

```
Time,10
SimulationLength,80
TrackLength,42
```

Następnie należy zdefiniować pociągi lub stację, odpowiednio podając Train lub Station. Po przecinku, w przypadku pociągu należy umieścić jego unikalny identyfikator, dalej nazwę, maksymalną pojemność, liczbę drzwi oraz pozycję startową na torze.

```
Train,1,Chrobry,100,10,1
Train,2,Sobieski,100,10,10
```

Gdy definiujemy stację, to podać należy jedynie unikalny identyfikator, nazwę oraz pozycję na torze.

```
Station,1,Kabaty,3
Station,5,Ursynów,7
```

Oczekiwany wyniki

Spodziewanym wynikiem będzie plansza toru głównego i przeciwnego. Podany jest aktualny czas oraz lista stacji i pociągów. Liczby porządkowe reprezentują pozycje na torze – jest to numer komórki wektora toru. W

nawiasach kwadratowych. Liczby ujemne odnoszą się do pasażerów, którzy wysiedli z pociągu na stację końcową i ją opuścili.

Informacje wyświetlane na konsoli są również zapisywane do pliku `simulation_log.txt` w tej samej lokalizacji co plik wykonywalny.

Definiowanie pliku źródłowego (opcjonalnie)

Chcąc zaprojektować własne metro nie korzystając z pliku wejściowego lecz wpisując kod do main.cpp należy początkowo ustalić początkowe parametry symulacji takie jak:

`int time` – prędkość symulacji, wartość musi być większa od zera, im większa, tym wolniej symulacja przebiega.

`int simulationLength` – liczba iteracji, określa jak długo symulacja będzie wykonywana. Jedna iteracja to w rzczywistym świecie 15s.

Poniżej przedstawiono definicję podstawowych obiektów, których nie należy tworzyć bezpośrednio, lecz poprzez wskaźniki – koniecznie `shared_ptr`.

Tworzenie wskaźnika i obiektu pociągu. Pierwszy argument to unikalne ID (unsigned), drugi nazwa pociągu (`std::string`), kolejne dwa to maksymalna pojemność pociągu oraz liczba drzwi.

```
std::shared_ptr<Train> chrobry = std::make_shared<Train>(1, "Chrobry", 100, 10);
```

Utworzenie stacji wraz ze wskaźnikiem przedstawiono poniżej. Wystarczy jedynie podać w pierszym argumencie identyfikator (unsigned) oraz nazwę stacji.

```
std::shared_ptr<Station> kabaty = std::make_shared<Station>(1, "Kabaty");
```

Przy definicji toru podaje się unikalny identyfikator oraz długość toru. Należy mieć na uwadze że przejechanie pociągu z jednej komórki na drugą zajmuje pociągowi 15s.

```
std::shared_ptr<Track> track1 = std::make_shared<Track>(1, 42);
```

W kolejnej części należy przypisać stacje do toru, gdzie pierwszym argumentem jest stacja, którą przypisujemy, a kolejnym jest pozycja na torze.

```
track1->set_station(kabaty, 3);
```

Po przypisani stacji należy stworzyć tor o przeciwnym kierunku za pomocą metody `make_reverse_track()`. Jako argument metody należy podać identyfikator przeciwnego toru.

```
std::shared_ptr<Track> track2 = track1->make_reverse_track(2);
track1->set_reverse_track(track2);
track2->set_reverse_track(track1);
```

Następnie do torów przypisujemy pociągi. Jako argumenty należy podać wskaźniki do pociągów oraz ich pozycję startową na torze. Można je przypisać do oryginalnego toru jak i do przeciwnego. Skład pociągu automatycznie będzie wiedział w którą stronę jechać. Po dojechaniu do końca toru, pociąg automatycznie zostanie przypięty do toru przeciwnego. Swoją drogą zatoczy krzywą zamkniętą.

```
track1->set_train(chrobry, 1);
```

Poniżej tworzona jest instancja symulacji. Jako argumenty należy podać toru, który ma być symulowany oraz szybkość symulacji.

```
Simulation simulation(track1, time);
```

Przy wywołaniu startu symulacji określić ilość iteracji.

```
simulation.start_simulation(simulationLength);
```

5. Plan rozwoju

Projekt symulacji metra posiada duży potencjał rozwojowy, który można wykorzystać w celu zwiększenia jego realistyczności i funkcjonalności. W planach rozwoju projektu priorytetem jest wykorzystanie aktualnie niedocenianych parametrów, takich jak ilość drzwi w pociągu. Ten aspekt można rozwijać poprzez wprowadzenie modelu szybkości wymiany pasażerów, która będzie zależna od liczby dostępnych drzwi.

Kolejnym krokiem jest bardziej szczegółowe uwzględnienie maksymalnej pojemności pociągów w symulacji. Pozwoli to na realistyczne odwzorowanie wpływu przepustowości pociągów na efektywność całego systemu metra, w tym na częstotliwość i opóźnienia kursów.

Dalsza rozbudowa programu może również obejmować wprowadzenie możliwości wczytywania kilku torów z pliku, co umożliwi tworzenie bardziej złożonych sieci metra z torami o wspólnych stacjach. Taka funkcjonalność pozwoli na symulację złożonych systemów metra w dużych miastach, gdzie różne linie często krzyżują się lub łączą. Dzięki temu użytkownik będzie mógł projektować i testować indywidualne konfiguracje sieci, co jest szczególnie istotne dla planowania rozwoju infrastruktury miejskiej i analizy różnych scenariuszy ruchu.

6. Wnioski

W projekcie wykorzystano język programowania C++, który znany jest z wysokiej wydajności. Jest to kluczowa cecha w przypadku modelowania złożonych systemów transportowych.

Projekt ma zarówno wartość edukacyjną, jak i praktyczną, ponieważ umożliwia nie tylko naukę i doskonalenie umiejętności programistycznych, ale także realistyczne modelowanie i analizę systemów transportu miejskiego. Może być używany jako narzędzie dydaktyczne w edukacji inżynierskiej oraz jako praktyczne narzędzie do planowania i optymalizacji transportu.