AKADEMIA GÓRNICZO-HUTNICZA

Dokumentacja do projektu

Symulator Sygnalizacji Świetlnej

z przedmiotu

Języki programowania obiektowego

Elektronika i Telekomunikacja, 3 rok

Sławomir Tenerowicz

Grupa: poniedziałek 12:50

prowadzący: Rafał Frączek

11.02.2021

1. Opis projektu

Projekt umożliwia symulację skrzyżowania dwóch dróg (głównej i podporządkowanej) wraz z sygnalizatorami przejść dla pieszych. Sygnalizatory świetlne zostały przydzielone do dwóch grup logicznych. W każdej grupie logicznej znajdują się dwa sygnalizatory dla ruchu drogowego i cztery sygnalizatory dla ruchu pieszych. Wszystkie sygnalizatory wewnątrz jednej grupy mogą jednocześnie zapalić światło zielone, nie powodując błędów ruchu. W danym momencie tylko jedna z grup może aktywować światła zielone swoich sygnalizatorów.

2. Project description

The project enables to conduct crossing simulation of two roads (main road and cross road) including traffic lights for pedestrians. All Traffic Lights are assigned to two logic groups. In each logic group there are two Traffic Lights that control Cars Traffic and four Traffic Lights that control Pedestrian Traffic flow. All Traffic Lights that belong to the same logic group can simultaneously set Green Light without causing any traffic problems. At the time only one group can activate Green Lights of its Traffic Lights.

3. Instrukcja użytkownika

Program rozpoczyna się informacją powitalną oraz zestawem komend w języku angielskim. Komenda *commands* pozwala na wyświetlenie zestawu komend. Komenda *exit* kończy program, *run* uruchamia symulator, *running-config* wyświetla aktualną konfigurację symulacji natomiast *change-config* pozwala na zmianę parametrów – czasu trwania światła zielonego na drodze głównej oraz podporządkowanej, światła zielonego dla pieszych na przejściach równoległych do dróg głównej i podporządkowanej jak również pozwala na ustalenie ilości cykli, które program zasymuluje. W trakcie działania symulacji komenda *exit* nie jest wykonalna. Domyślnie program zapala światło zielone na drodze głównej.

Zmiany zachodzące w czasie wykonywania programu prezentowane są w uproszczonej formie graficznej w konsoli.

4. Kompilacja

Program podlega standardowej kompilacji.

5. Pliki źródłowe

Projekt składa się z następujących plików źródłowych:

- *Light.h*, *Light.cpp* – najmniejszy element w hierarchii programu, klasa wirtualna, rodzic dla świateł zielonego, żółtego i czerwonego, ustawia stan światła (on/off) oraz jego atrybuty graficzne (pozycja w konsoli i kolor w konsoli),
- *GreenLight.h*, *GreenLight.cpp*, *RedLight.h*, *RedLight.cpp*, *YellowLight.h*, *YellowLight.cpp* – pochodne klasy Light, zawierają na stałe przypisany kolor światła
- *TrafficLight.h*, *TrafficLight.cpp* – klasa imitująca sygnalizator świetlny dla ruchu samochodowego, zawiera obiekty typu GreenLight, YellowLight oraz RedLight,
- *PedestrianTrafficLight.h*, *PedestrianTrafficLight.cpp* – klasa imitująca sygnalizator świetlny dla ruchu pieszych, zawiera obiekty typu GreenLight oraz RedLight,
- *TrafficLogicNode.h, TrafficLogicNode.cpp* – klasa zestawiająca obiekty typów TrafficLight oraz PedestrianTrafficLight w jedną logiczną grupę, przechowuje czasy trwania poszczególnych cykli oraz definiuje sekwencje zmian świateł w obrębie grupy logicznej
- *TrafficManagement.h, TrafficManagement.cpp* – klasa nadrzędna, składa się z dwóch obiektów typu TrafficLogicNode (jedna grupa logiczna dla drogi głównej a jedna dla drogi podporządkowanej),

zawiera menu kontekstowe, dostarcza uproszczony interfejs graficzny, wykonuje sekwencje zmian świateł zgodnie z danymi dotyczącymi danej grupy logicznej.

6. Zależności

- brak

7. Opis klas

W tym punkcie należy umieścić opis wszystkich stworzonych w projekcie klas. Należy podać do czego służy dana klasa oraz informację o jej publicznych metodach. Opcjonalnie można załączyć fragmenty kodu źródłowego. Na przykład:

W projekcie utworzono następujące klasy:

- `Light` – klasa wirtualna definiująca światło drogowe dowolnego typu.
  - `void set_state(bool self_state)`– ustawia stan światła 0 – OFF, 1 - ON oraz jego kolor w konsoli (korzysta z funkcji set_color oraz clear_color)
  - `bool get_state(void) const`– zwraca aktualny stan światła (wł/wył),
  - `void set_coords(int self_x, int self_y) - ustawia pozycję (x,y) światła w konsoli`
  - `int get_coords_x(void) const - zwraca pozycję (x) światła w konsoli`
  - `int get_coords_y(void) const - zwraca pozycję (y) światła w konsoli`
  - `void gotoxy(const int x, const int y) - funkcja umożliwiająca manipulację poszczególnymi pikselami na pozycjach (x,y) w konsoli,`
  - `void set_color(void) - funkcja ustawia kolor (w konsoli) piksela który odpowiada pozycji światła (światło włączone)`
  - `void clear_color(void) - funkcja ustawia kolor (w konsoli) piksela który odpowiada pozycji światła (światło wyłączone)`
  - `virtual string get_color(void) const = 0 - metoda czysto wirtualna, zapewnia wirtualność klasy`
- `RedLight` – klasa definiująca światło czerwone (utworzenie obiektu automatycznie ustawia atrybut color na czerwony, który jest potem brany pod uwagę przy ustawianiu kolorów świateł w konsoli)
  - `string get_color(void) const)` – zwraca informacje o kolorze światła,
- `YellowLight` – klasa definiująca światło żółte (utworzenie obiektu automatycznie ustawia atrybut color na żółty, który jest potem brany pod uwagę przy ustawianiu kolorów świateł w konsoli)
  - `string get_color(void) const)` – zwraca informacje o kolorze światła,
- `GreenLight` – klasa definiująca światło zielone (utworzenie obiektu automatycznie ustawia atrybut color na zielony, który jest potem brany pod uwagę przy ustawianiu kolorów świateł w konsoli)
  - `string get_color(void) const)` – zwraca informacje o kolorze światła,
- `TrafficLight` – klasa definiująca sygnalizator świetlny dla ruchu samochodowego, zawiera obiekty klas światła Czerwonego, Żółtego i Zielonego

- `void set_coords_auto(int x_start, int y_start)`– automatycznie ustawia pozycję wszystkich świateł jednego sygnalizatora w konsoli, wymaga podania jedynie koordynatów światła czerwonego

- `PedestrianTrafficLight` – klasa definiująca sygnalizator świetlny dla ruchu pieszych, zawiera obiekty klas światła Czerwonego, i Zielonego
  - `void set_coords_auto(int x_start, int y_start)`– automatycznie ustawia pozycję wszystkich świateł jednego sygnalizatora w konsoli, wymaga podania jedynie koordynatów światła czerwonego

- `TrafficLogicNode` – klasa zestawiająca sygnalizatory dla ruchu pieszych i samochodów w jedną logiczną grupę
  - `void set_traffic_green_timeout(int self_tgt)` – ustawia czas trwania światła zielonego dla ruchu samochodów w danej grupie logicznej
  - `int get_traffic_green_timeout(void) const` – pobiera informacje o czasie trwania światła zielonego dla ruchu samochodów w danej grupie logicznej
  - `void set_pedestrian_green_timeout(int self_tgt)` – ustawia czas trwania światła zielonego dla ruchu pieszych w danej grupie logicznej (przejścia równoległe do ruchu samochodów w grupie logicznej)
  - `int get_pedestrian_green_timeout(void) const` – pobiera informacje o czasie trwania światła zielonego dla ruchu pieszych w jednej grupie logicznej
  - `void ptl_set_state(int state)` – ustawia stan (0 – OFF, 1 – ON) wszystkich sygnalizatorów ruchu pieszych w jednej grupie logicznej, składa się z funkcji `void ptl_Green_set(int state)` oraz `void ptl_Red_set(int state)`
  - `void ptl_Green_set(int state)` – ustawia wszystkie światła sygnalizacji ruchu pieszych na kolor zielony w jednej grupie logicznej
  - `void ptl_Red_set(int state)` – ustawia wszystkie światła sygnalizacji ruchu pieszych na kolor czerwony w jednej grupie logicznej
  - `void ptl_Sequence_on(void)` – zawiera definicję sekwencji włączania świateł ruchu pieszych wraz z opóźnieniem (ustalone na stałe)
  - `void ptl_Sequence_off(void)` – zawiera definicję sekwencji wyłączania świateł ruchu pieszych (miganie światła zielonego przed zmianą na czerwone) wraz z opóźnieniem (ustalone na stałe)
  - `void tl_set_state(int state)` – ustawia stan (0 –ON, 1 – OFF) wszystkich świateł ruchu samochodów w jednej grupie logicznej
  - `int tl_Sequence_on(int change_time)` – definicja sekwencji włączania świateł ruchu samochodowego wraz z opóźnieniem (`int change_time`)
  - `int tl_Sequence_off(int change_time)` – definicja sekwencji wyłączania świateł ruchu samochodowego wraz z opóźnieniem (`int change_time`)
  - `void starting_state(int state)` – ustawia stan początkowy świateł po uruchomieniu programu

- `TrafficManagement`– klasa implementująca dane o sterowaniu ruchem i wyświetlająca je na ekranie konsoli w formie graficznej

- `void context_menu() - wywołanie tej metody na obiekcie klasy TrafficManagement rozpoczyna działanie programu - wyświetlenie menu z komendami w konsoli`

8. Zasoby

- brak

9. Dalszy rozwój i ulepszenia

Projekt można by udoskonalić zarówno graficznie jak i funkcjonalnie.

Poprawa graficzna:

- wyrysowanie pasów na osi jezdni, wyrysowanie przejść dla pieszych,
- wizualizacja ruchu pieszych i aut (po dodaniu tej funkcjonalności).

Poprawki funkcjonalne:

- symulacja ruchu samochodowego i pieszych (generator średniego ruchu aut i pieszych na każdą z dróg – symulacja korków i ich optymalizacja, automatyczna, gdzie program sam dobiera czasy poszczególnych świateł tak, by zakorkowanie było jak najmniejsze),
- możliwość ustawiania pozycji poszczególnych świateł i dróg z poziomu konsoli,
- zwiększenie ilości pasów i zarządzanie ruchem na nich (pas do zawracania, pas do bezkolizyjnej jazdy w lewo, strzałka warunkowego skrętu w lewo)
- stan wyłączenia sygnalizacji (mrugające światła żółte , wyłączone sygnalizatory dla pieszych).

10. Inne

Początkowo klasy RedLight, GreenLight oraz YellowLight miały bardziej rozbudowane funkcjonalności (zawierały czasy trwania oraz specjalne metody, jak mruganie w przypadku świateł żółtego i zielonego). W miarę postępu projektu te metody i atrybuty zostały przeniesione do klas nadrzędnych, głównie ze względu na problem z synchronizacją timerów w każdym z poszczególnych sygnalizatorów. Mimo to uznano, że warto wyodrębnić przez dziedziczenie z klasy Light klasy RedLight, GreenLight oraz YellowLight. W dalszym rozwoju projektu jest możliwość dziedziczenia świateł bezkolizyjnych i strzałek warunkowych po tych właśnie klasach. Takie rozwiązanie pozwoliło również na wykorzystanie dziedziczenia w projekcie jak i utworzenie klasy wirtualnej. W przeważającej części projektu w klasach skorzystano z agregacji klas.