Pytania i zadania

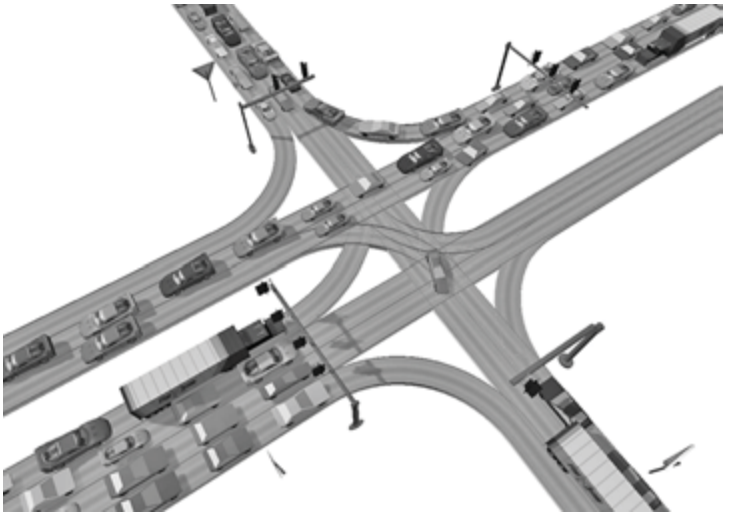
1. W przedstawionym przypadku autobus zatrzymuje się na każdym przystanku po 15 s. Co należy zrobić, aby czas postoju pojazdu był losowo zmieniany w przedziale od 5 do 20 s?
2. Zbuduj i uruchom przedstawiony model na 150 kursów. Sprawdź, czy w etykiecie *Wszyscy* są zliczani pasażerowie wszystkich kursów. Jeżeli nie, to jakie modyfikacje należy wprowadzić?
3. W jaki sposób obliczyć ilość zużytego paliwa?
4. Spróbuj za pomocą obiektu *Text* wyświetlać w oknie głównym programu inne parametry, np. prędkość chwilową lub dystans całkowity.
5. Czy przy zastosowanych rozkładach statystycznych ogólna liczba przewiezionych pasażerów nie wydaje ci się zbyt duża? Zmniejsz suwakiem prędkość symulacji i sprawdź, czy na każdym przystanku liczba pasażerów nie jest naliczana podwójnie. Jeżeli tak, to gdzie jest błąd? Wykorzystaj komunikaty MSG do wprowadzenia poprawek.
6. Zwróć uwagę na parametry *travel empty* i *travel loaded*. Jak je zinterpretujesz?
7. Wykonaj serię eksperymentów uzależniających prędkość pojazdu od liczby pasażerów w autobusie dobieranych losowo. Pokaż, jak zmienia się całkowity czas jednego przejazdu w zależności od obciążenia autobusu.

2.3. Modelowanie ruchu na skrzyżowaniu

Zadanie. Należy zbudować model skrzyżowania umożliwiający odczyt wybranych parametrów ruchu oraz regulację czasu pracy sygnalizatorów świetlnych. Założono losowe generowanie natężenia ruchu pojazdów z wykorzystaniem wybranych rozkładów statystycznych dostępnych w programie. Operator powinien mieć możliwość zmiany ustawień czasu trwania zielonego światła. W wariantach bardziej skomplikowanych można też uwzględnić ruch pieszy. Na rysunku 42 pokazano przykład skrzyżowania wielopasmowego, które można zasymulować w środowisku FlexSim. Jest to dosyć rozbudowany model prezentujący możliwości programu. Dla celów edukacyjnych zostanie przedstawiony prostszy model skrzyżowania (rys. 43), aby użytkownik w przejrzysty sposób poznał zasady programowania obiektów do sterowania ruchem.

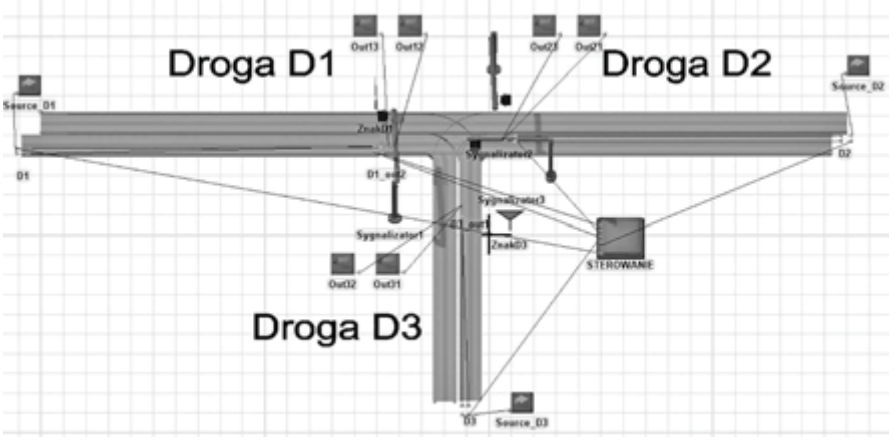
Rozwiązanie. Wykorzystane będą przede wszystkim obiekty typu przenośnik taśmowy *Conveyor* jako elementy, z których można budować dowolne układy drogowe. Oprócz tego będą potrzebne elementy typu *Source, Sink* oraz *Queue* do sterowania sygnalizacją świetlną. Standardowo przenośniki taśmowe mają nogi oraz rolki, po których przemieszczają się elementy przepływu. Rolki można jednak zastąpić dowolną teksturą (zakładka *General*), w tym przypadku użyto tekstury przypominającej asfalt, a nogi wyłączono w opcji *Leg base relative to conveyor* w zakładce *Conveyor* każdego przenośnika[1]. Tekstury graficzne można

[1] W wersjach FlexSim 2016 i późniejszych, aby uzyskać standardowy wygląd przenośników taśmowych, które zostały omówione w tym zadniu, należy włączyć opcję *Display Legacy Conveyors in the Library* w zakadce *File* ⇨ *Global Preferences* ⇨ *Environment.*



Rysunek 42

utworzyć samodzielnie, także pobrać darmowe próbki z Internetu. Koncepcja sterowania ruchem została zaczerpnięta z projektu opublikowanego na forum użytkowników FlexSim (Peterson, 2007). Do celów edukacyjnych zbudowano inny model skrzyżowania z rys. 43. Są na nim drogi z pierwszeństwem przejazdu D1 i D2 oraz droga podporządkowana D3, co jest istotne w przypadku awarii sygnalizacji świetlnej. W normalnych warunkach ruchem sterują trzy sygnalizatory, dające zielone światło w pewnych interwałach dla każdej z dróg.



Rysunek 43

Program obsługujący sygnalizację świetlną umieszczono w obiekcie (*Queue*) STEROWANIE. Pojazdy z dróg D1 i D2 mogą skręcać w lewo lub jechać prosto. Pojazdy z drogi D3 mogą poruszać się w prawo lub w lewo. Kod programu ma za zadanie włączanie kolejno zielonego światła najpierw na drodze D1, a następnie D2 i D3, po czym cały cykl się powtarza. Do rozwiązania problemu sterowania światłami zastosowano mechanizm podobny do omówionego już w zadaniu z podrozdz. 2.1. Rozpoczęcie sekwencji zadań dla sygnalizacji świetlnej jest inicjowane za pomocą znanego już z poprzednich przykładów polecenia:

```
senddelayedmessage(current, 0,current);
```

Wiadomość z opóźnieniem zero obiekt STEROWANIE wysyła sam do siebie, zakładka *Tirggers* ⇨ *OnReset,* aby zainicjować proces działania sygnalizacji. Dalsza praca programu przebiega już zgodnie z sekwencjami *case* 1, 2, 3, 4, 5 uruchamianymi kolejno (zakładka *Tirggers* ⇨ *OnMessage*). Zmiennymi typu *double* zdefiniowano w sekundach czasy trwania zielonego światła dla poszczególnych dróg D1, D2 i D3. Na początku przez 20 s występuje sekwencja oczekiwania na dojazd pojazdów do skrzyżowania (wszystkie światła czerwone, może być to wykorzystane na ruch pieszych, ponieważ wszystkie samochody stoją – *case 1*). Następnie (*case 2*) jest włączane zielone światło na 50 s dla drogi D1, dalej na 60 s dla drogi D2 i na 30 s dla drogi D3. Po czym następuje zamknięcie wszystkich portów na 20 s i cała sekwencja powtarza się od początku. Zastosowano polecenie przełącznika typu *switch*(*value*), gdzie możliwe jest dołączenie dowolnej liczby przypadków. Generalnie przełącznik jest szybszy jako procedura numeryczna od funkcji *if – else* i prostszy programistycznie w tym konkretnym przypadku. Poniżej zaprezentowano kod programu dla obiektu STEROWANIE:

```
/**Custom Code*/
treenode current = ownerobject(c);
treenode sendingobject = parnode(1);
double value = parval(2);
double dojazd = 20;
double czasD1 = 50;
double czasD2 = 60;
double czasD3 = 30;
switch(value){
case 1:  closeoutput(outobject(current, 1));
         closeoutput(outobject(current, 2));
         closeoutput(outobject(current, 3));
         senddelayedmessage(current,dojazd,current,2,0,0);
      break;
case 2:  closeoutput(outobject(current, 2));
         closeoutput(outobject(current, 3));
         openoutput(outobject(current, 1));
         senddelayedmessage(current,czasD1,current,3,0,0);
      break;
case 3:  closeoutput(outobject(current, 1));
         closeoutput(outobject(current, 3));
         openoutput(outobject(current, 2));
         senddelayedmessage(current,czasD2,current,4,0,0);
      break;
case 4:  closeoutput(outobject(current, 1));
         closeoutput(outobject(current, 2));
         openoutput(outobject(current, 3));
         senddelayedmessage(current,czasD3,current,5,0,0);
```