УДК 681.51

Д.С. Печенежський, аспір.
В.М. Томашевський, д.т.н., проф.
Національний технічний університет України "КПІ"

КЕРУВАННЯ АВТОМОБІЛЬНО-ДОРОЖНІМ РУХОМ З ВИКОРИСТАННЯМ ІМІТАЦІЙНОЇ МОДЕЛІ

Запропоновано концепцію створення імітаційного проекту для управління рухом автотранспортних засобів на основі візуального моделювання. Описано архітектуру та принципи функціонування основних компонентів і засоби їхньої реалізації. Імітатор моделі руху транспортних засобів використовує розподілену обробку у мережі і технологію Java.

Вступ

Інтеграція України в європейські структури ставить нові завдання щодо покращення дорожнього руху. З цією метою розроблена "Державна програма розвитку дорожнього руху на автомобільних дорогах, вулицях ...". Одне із завдань цієї програми полягає в розробці імітаційних моделей дорожнього руху для аналізу пропускної спроможності та місць концентрації дорожньо-транспортних подій. Пропускна спроможність залежить від множини чинників: конфігурації доріг, кількості смуг, кількості і типів перехресть, переходів тощо. Основною причиною зниження пропускної здатності доріг є, так звані, "вузькі місця", пошук яких і є головною метою досліджень. Застосування аналітичних моделей для вирішення такого завдання важко через його складність.

Раніше створені програми подібного призначення імітували дорожній рух досить умовно. Основне їхнє призначення полягало в знаходженні наближених оцінок пропускної спроможності та коефіцієнтів завантаження ділянок міської мережі при різних варіантах планувальних рішень. Імітатори можуть виконувати більш широке коло завдань, включаючи прогнозування ситуацій на дорогах і перевірку з їхньою допомогою тривалостей переключення сигналів світлофорів для визначеного світла.

Основним недоліком цих моделей була їхня неточність. Ділянки доріг розглядаються як багатоканальні прилади, а перехрестя не моделювалися взагалі, тобто передбачалося, що будь-який автомобіль може вільно проїхати перехрестя в довільний момент часу, незважаючи на ситуацію на ньому. Дане припущення доречно при грубих оцінках і при невеликому коефіцієнті завантаження ділянок доріг, що прилягають до перехрестя. При збільшенні коефіцієнта завантаження доріг рух кожного автомобіля залежать від руху інших. Також посилюється вплив конфігурації перехрестя, режим його регулювання тощо. Аналітично прорахувати вплив конфігурації перехрестя на його пропускну здатність дуже складно, тому що збільшення коефіцієнта завантаження доріг веде до зниження пропускної спроможності перехрестя, що, у свою чергу, призводить до ще більшого збільшення коефіцієнтів завантаження доріг.

Ще одним істотним недоліком вже існуючих моделей є їх нерозвинений інтерактивний інтерфейс, що виконує з ними трудомістку роботу.

Виходячи з вищесказаного, можна зробити висновок, що є необхідність у створенні нових імітаційних систем для моделювання руху автомобільного транспорту. Подібні системи імітації дорожнього руху розроблені в країнах Західної Європи й Америки, такі як MITSIM, FRESIM, PHAROS (США), AUTOBAHN, PLANSIM-T (Німеччина), DRACULA, PADSIM (Великобританія), ANATOLL, SIMDAC (Франція) тощо. Вони використовуються, у першу чергу, для покращення умов руху міського і суспільного транспорту, а також для оцінки його економічної ефективності. Європейським співтовариством розроблений перелік вимог для систем подібного роду (див. **http://www.its.leeds.ac.uk/smartest**). Ці системи працюють у режимі реального часу і мають імітатори потоків руху транспорту. Проте в Україні поки що немає подібних систем. Побудова таких систем пов'язана з розробкою інтелектуальних систем керування дорожнім рухом [1].

Мета і концепція

Метою створення імітаційного проекту автомобільного дорожнього руху є моделювання мережі дорожнього руху, визначення пропускної спроможності ділянок доріг, а також знаходження оптимальних маршрутів для спеціальних транспортних засобів, таких як швидка допомога, міліція, пожежна служба тощо.

Виходячи з принципів побудови візуальних об'єктних систем, необхідно провести статичну декомпозицію об'єктів для моделі дорожнього руху. Основним компонентом верхнього рівня візуального розміщення імітаційної моделі є карта основних автомагістралей України, що, у свою чергу, розділяється на множину інших компонентів, кожен із яких являє собою карту міста або карту заміських доріг. Кожен компонент міського розміщення може бути розчленований на компоненти, які відображають усілякі фрагменти міської мережі доріг (районів). Ті, у свою чергу, поділяються на різноманітні ділянки доріг: перетинання, лінійні ділянки, ділянки з круговим рухом, нахили, підйоми, ділянки з поворотами, ділянки з тунелями, ділянки з обмеженням за висотою (проїзди під мостами), ділянки із зупинками громадського транспорту, ділянки з обмеженнями вантажності (мости, естокади).

Відмінністю розробленої моделі від інших її подібних аналогів, перерахованих вище, є те, що імітаційна модель створюється автоматично для тієї ділянки дорожнього руху, яка відображається на екрані дисплея. Такий підхід не вимагає побудови повної вичерпної імітаційної моделі та її збереження в пам'яті комп'ютера, тобто використовується принцип "що відображаю, те і моделюю". Як найменший відображуваний елемент моделювання був узятий *домен* [2], що являє собою односмугову регульовану або нерегульовану ділянку дороги. Передбачається, що ширина смуги достатня для розміщення автомобіля будь-якої ширини, але не більше одного автомобіля за шириною. Ділянки доріг розділяються на домени так, щоб світлофори, знаки, переходи, розгалуження, консолі доріг були на кінцях доменів. Таким чином, ***домен*** – це елементарна односмугова ділянка дороги, протягом якої автомобілі не зустрічають перешкод крім інших автомобілів. Наприкінці такої ділянки по ходу руху автомобіля можуть знаходитися перешкоди: світлофор, знак, перехід, дороги, що перетинаються з ділянкою, розгалуження доріг. Очевидно, що будь-яка ділянка дороги може бути розбита на домени, причому, на розглянутій ділянці реальної дороги можуть знаходитися одночасно декілька доменів, як показано на рис. 1.

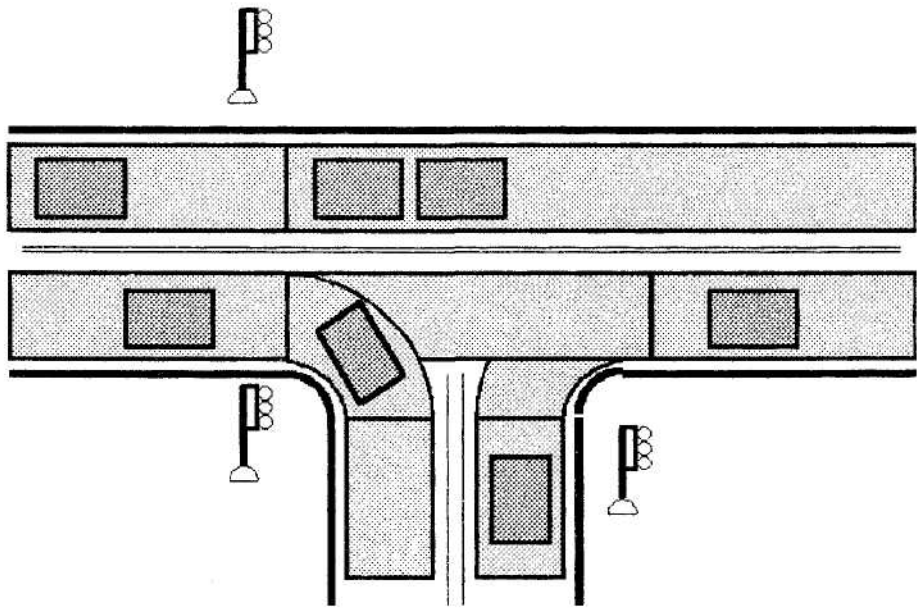

Рис. 1. Регульоване перехрестя з доменами

У залежності від ділянок доріг, на які накладаються домени, динамічний об'єкт доменів характеризується різною довжиною та кількістю автомобілів на ньому.

Домени містять у собі динамічні об'єкти *автомобілів*. Об'єкт ***автомобіль*** використовується для моделювання руху автотранспортного засобу. Він створюється на ділянці дороги для позначення конкретного транспортного засобу з його характеристиками. Транспортний засіб може належати до одного з таких типів: мотоцикл, легковий або вантажний автомобіль, пасажирський транспорт, спеціальний транспорт (міліція, пожежна служба, швидка допомога) тощо.

Динамічний об'єкт *автомобіль* у залежності від транспортного засобу, що він відображає, має різноманітні характеристики, що включають: негативне мінімальне прискорення (гальмування), позитивне максимальне прискорення (розгін), довжину, висоту, ширину і повну масу автомобіля (маса спорядженого транспортного засобу з вантажем і пасажирами), пріоритет, що може встановлюватися для громадського транспорту і спеціального транспорту.

Об'єкт *автомобіль* може входити в об'єкт домен, пересуватися в ньому, зупинятися, ставати у чергу і покидати його. Попадаючи в якийсь визначений домен, "поведінку" автомобіля визначають параметри домену (максимальна дозволена швидкість, знаки тощо).

У моделі повинні відтворюватися особливі ситуації, що можуть бути активізовані в будь-який час. Під особливими ситуаціями припускається затор на трасі, аварія, ефект поперечного звуження тощо. Інцидент може бути генерований стохастичним або детермінованим засобом. Детерміновані інциденти повинні визначатися користувачем. Стохастичні інциденти створюються для кожної ділянки дороги відповідно до заданого імовірнісного розподілу.

Представлення статичних об'єктів

Для імітації дорожнього руху необхідно мережне уявлення ділянок доріг (граф), організація керування потоком автомобілів, відтворення особливих ситуацій, маршрутизація транспортних засобів і їх переміщення.

Деталізація імітаційної моделі повинна змінюватися в залежності від рівнів візуального відображення, визначуваних користувачем, що дає можливість моделювати як окремі транспортні засоби, так і потоки автомобілів.

Для імітатора дороги представляються зв'язками, вузлами мережі та доменами. Дані, що описують модельовану мережу, читаються з мережної бази даних, яка може бути легко створена, використовуючи інтерактивний графічний редактор мережі доріг, що розроблений для імітатора. Мережна база даних (БД) включає опис усіх мережних об'єктів, з'єднання ділянок доріг, пріоритет використання траси, регулювання розворотів і переміщень на перетинаннях (наприклад, заборона лівих поворотів і розгортання) та керуючих пристроїв (світлофорів).

Вузол може бути або перетинанням окремих доріг, або джерелом, або стоком, де потоки руху вводяться або покидають мережу, що імітується. Кожен вузол представляється такими елементами даних:

- тип вузла, що визначає чи є вузол перетинанням, джерелом або стоком;
- код вузла – це унікальний пізнавальний номер для вузла;
- ім'я вузла – може використовуватися довільне ім'я вузла, щоб маркерувати вузол у графічному вікні та звіті моделювання.

Зв'язки – протяжні ділянки доріг без перетинань, що з'єднують вузли. Кожен зв'язок може складатися з однієї або більшої кількості ділянок дороги і характеризується:

- типом зв'язку, що може бути автострадою, ухилом, міським проїздом, тунелем;
- кодом зв'язку; код зв'язку – пізнавальний номер, який призначений для кожного зв'язку;
- стартовими і кінцевими вузлами для з'єднання зв'язків;
- числом доменів на ділянці;
- числом смуг на ділянці дороги (кількість паралельних смуг одного напрямку руху).

Вхідні та вихідні зв'язки вузлів з'єднується, якщо існує, принаймні, одне з'єднання шляху між цими двома зв'язками. Обмеження перебудувань з одного зв'язку в інший, подається в таблиці, яка містить список обмежень на виконання перебудувань для кожної визначеної пари.

Імітатор представляє мережу на рівні доменів, що накладаються на смуги ділянок доріг. Кожен домен описаний двома елементами даних: 1) кодом домену, тобто унікальним пізнавальним кодом; 2) маршрутом, який змінює напрямок і код пріоритету використання траси, що визначається за деякими правилами.

Правило зміни смуги:
- дозволено поворотом ліворуч;
- дозволено поворотом праворуч;
- дозволено поворотом і ліворуч, і праворуч.

Правило пріоритету використання смуги:
- Заборона руху вантажівкам;
- Обмеження проїзду за висотою;
- Обмеження за масою.
- Тільки для обслуговуючих транспортних засобів.
- Тільки для громадського транспорту.

Кодом пріоритету використання смуги для розглянутої ділянки дороги, може бути будь-яка незаперечлива комбінація з п'яти типів, які визначаються вище.

Кожна ділянка дороги може бути з'єднана з наступною і попередньою ділянкою дороги. Для цього використовується список пар кодів ділянок доріг у мережній базі даних, щоб представити з'єднання між попередньою і наступною ділянкою дороги.

Домен має такі атрибути: номер; довжину; логічну функцію умов дозволу покинути автомобілю домен; максимальну дозволену швидкість на домені, яка залежить від кривизни ділянки дороги або обмежується знаками; упорядкований список автомобілів, що знаходяться на домені; час останнього відновлення атрибутів домену.

Домен моделює стан ділянки дороги і має метод оброблювача домену, який обчислює предикат дозволу автомобілю покинути домен, і, у залежності від результату, планує поведінку автомобілів на даному домені. Функція-оброблювач визначає час настання такої події. Під подією розуміється зміна логічної умови, зміна прискорення будь-яким автомобілем (гальмування, зупинка, початок руху, досягнення максимальної або заданої швидкості), прибуття або вибуття автомобіля на (із) домен. Особливо розглядається перший у списку автомобіль. Тільки він може покинути домен під час обробки однієї події. Можливість покинути першим автомобілем домен залежить від умов: чи досяг він кінця ділянки, від його швидкості та прискорення; чи дозволено йому покинути домен; яке положення, швидкість і прискорення останнього автомобіля на такій ділянці.

Рух наступних автомобілів залежить тільки від їхнього положення на домені, швидкості, прискорення і таких же характеристик для попереднього автомобіля.

Наприкінці обробки з усіх можливих подій вибирається найближча і заноситься в таблицю подій, що призначена для керування процесом моделювання рухом автомобілів у імітаційній моделі.

Дані щодо мережної геометрії необов'язкові і необхідні тільки тоді, якщо потрібен графічний інтерфейс для користувача. Ці дані визначають геометрію з двох основних ліній (ліві та праві смуги з обмеженнями) для кожної ділянки дороги.

Перетинання шляхів з круговим рухом функціонує як односпрямована система, яка циркулює навколо центральної ізольованої ділянки, де керування відбувається за маркеруванням і пріоритетом руху на шляху з круговим рухом.

Уявлення динамічних об'єктів

Обробка динамічних об'єктів імітаційної моделі руху складається з циклу звертання до набору модулів у певні інтервали часу або при створенні деяких подій (дискретний подійний підхід).

Моделювання починається за визначеним сценарієм із завантаження параметрів моделювання і статичних компонентів мережі доріг. Потім ініціюється ітераційна процедура із заданим розміром кроку. Завдання, що виконується усередині кожної ітерації, включає:

1. Модифікацію стана сигналів руху, ознак і особливих ситуацій;
2. Модифікацію самих коротких шляхів і таблиць маршрутів;
3. Читання нових місць призначення, початку координат таблиці маршрутів і відповідне місце транспортних засобів у віртуальних чергах;
4. Завантаження транспортних засобів із віртуальних черг у мережу;
5. Модифікація можливостей прискорення транспортних засобів і перевірка, чи повинні вони змінювати смугу або допустимі інтервали для бажаної зміни смуги;

6. Відповідно до нових позицій транспортних засобів модифікуються їхні швидкості. Нові дані (швидкість, розміщення тощо) записуються модулем системи спостереження. Наприкінці ділянки дороги транспортний засіб або віддалиться з мережі (якщо він досягає свого місця призначення), або переміщається на наступну ділянку по ходу дороги;

7. Модифікується відображення на дисплеї, якщо допускається графічний інтерфейс користувача;

8. Обчислюються міри ефективності або передається стан мережі зовнішньому графічному інтерфейсу користувача або модулям;

9. Модифікується час моделювання і повернення до наступної ітерації.

Імітатор використовує підхід моделювання з часовим поділом для обробки переміщень транспортних засобів. У відповідь на зміну смуги й одержання інформації від попередніх автомобілів, що пов'язані із зміною смуги, тобто викликаються функції розрахунку динамічних параметрів для кожного транспортного засобу кожні ω секунд. Швидкості і позиції транспортних засобів модифікуються кожний τ секунд. Розмір кроку повинен бути в розрахунковому модулі більшим, ніж ω або дорівнювати розміру кроку τ. Цей фіксований розмір кроку прийнятий для загального керування потоком даних. Для індивідуальних транспортних засобів, розмір кроку для виклику визначених функцій моделювання може змінюватися, як час реакції водія, що зустрічає специфічні умови, наприклад, різке скорочення відстані між транспортними засобами тощо.

Модель вибору маршруту використовується, щоб фіксувати рішення про вибір маршруту водієм у відповідь на інформацію про графік руху. У залежності від доступу до інформації в імітаторі прийняті два типи водіїв: поінформовані і не поінформовані. Оскільки водій вибирає маршрут, то обчислюється можливість вибору вихідного зв'язку для кожного перетинання доріг. У даний час імітатор використовує модель логічного багаточлена і розглядає тільки затримку в часу руху при виборі маршруту (рис. 2):

$$p(l \mid j,t) = \frac{exp(u_l(t))}{\sum_{m \in L_j} exp(u_m(t))}, \qquad (1)$$

де $p(l \mid j,t)$ – можливість вибору зв'язку l для транспортного засобу, що очікує прибуття у вузол j у момент часу; L_j – множина виходів із зв'язку вузла j; $u_l(t)$ – вигода вибору наступного зв'язку маршруту після зв'язку l, що визначається так:

$$u_l = \alpha \cdot c_l(t) + \beta \cdot C_k(t + c_l(t)), \qquad (2)$$

де $c_l(t)$ – очікуваний час для транспортного засобу, щоб перетнути зв'язок l, що вводить цей зв'язок під час t; $C_k(t + c_l(t))$ – очікуваний час переміщення для самого короткого шляху з вузла k (що знаходиться до вузла зв'язку l) до місця визначення транспортного засобу, який досягає вузла j під час t; ц, ц – модельні параметри.

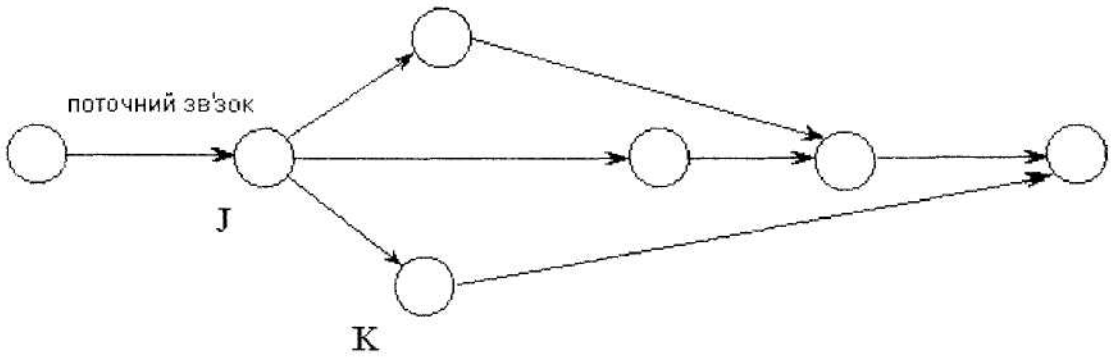

Рис. 2. Граф дорожньої мережі

Множина вибору L_j включає все з'єднання вихідних зв'язків, що знаходяться далі по ходу поточного зв'язку вузла, які можуть вибрати транспортні засоби ближче до його місця призначення. Таким чином, зв'язок повинен задовольняти обмеження $C_k(t + c_l(t)) \le C_j(t)$, щоб

бути включеним до набору вибору маршруту. Це обмеження також запобігає транспортним засобам від вибору такого шляху, який містить цикли.

Можуть також використовуватися більш складні моделі, що включають інші алгоритми вибору маршруту. Модульна конструкція імітатора враховує просту заміну заданих за умовчанням моделей.

Програмна реалізація мікроімітатора

Імітатор дорожнього руху нижнього рівня (далі LLS – Low Level Simulator) складається з таких трьох основних компонентів (рис. 3):

1. Ядро імітатора **(LLS Core)** – основний компонент системи, який робить усі обчислення й управляє самим процесом моделювання. Графічна підсистема **(LLS Graphic Engine)** призначена для візуального відображення процесу моделювання. Вона реалізована у вигляді Java додатку, яка виконується Віртуальною Ява Машиною, що входить до складу сучасних програм-броузерів (Netscape Navigator і Internet Explorer). Використання технології Java наділяє імітаційний проект декількома важливими якостями: розвинуті графічні можливості (прорисовування об'єктів у тривимірному просторі) і незалежність від програмної платформи.
2. Мережна реляційна система керування базами даних, підтримуюча структуровану мову запитів SQL (DBMS with SQL-server), наприклад, Interbase, MySql, PostgreSQL. БД служить для зв'язку ядра і графічної підсистеми імітатора.

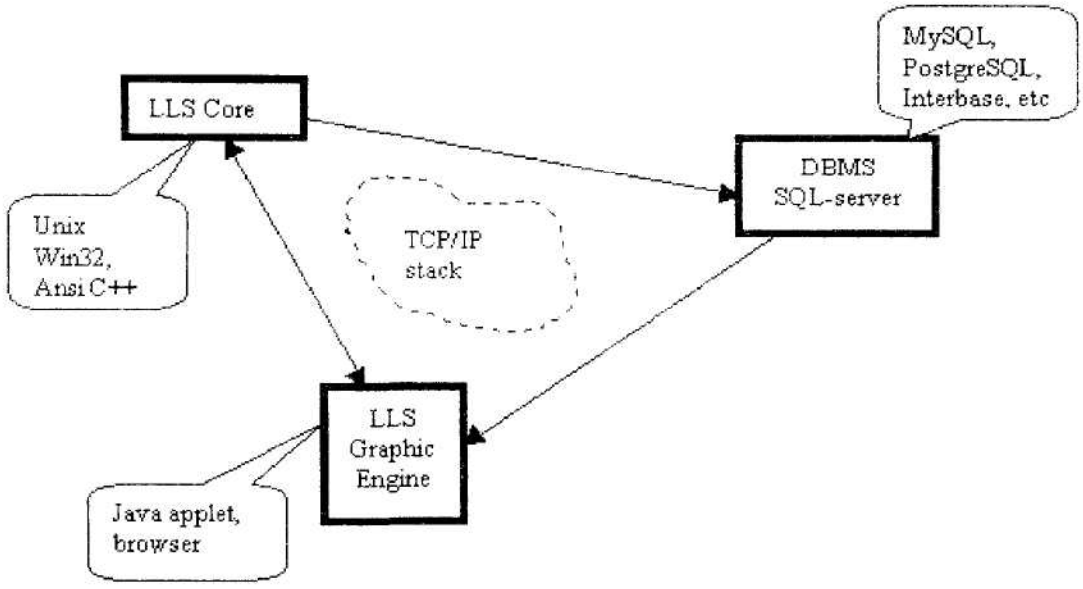

Рис. 3. Структура імітатора нижнього рівня

Компоненти системи взаємодіють між собою через локальну або глобальну комп'ютерну мережу з використанням протоколів сімейства TCP/IP. Завдяки такій структурі компоненти системи можуть розміщуватися як на однім комп'ютері, так і у взаємодії через Internet. У першому випадку, логіка роботи імітатора залишається незмінною, використовуючи як мережний інтерфейс, устрій lo0 (network loopback, фіктивний інтерфейс "петля"). Після кожного циклу (tick) моделювання ядро системи заносить дані про координати об'єктів (автомобілів) у таблицю бази даних, звідки ці дані, у свою чергу, зчитуються графічною підсистемою. Кожен рядок у таблиці відповідає положенню деякого об'єкта в конкретний момент модельного часу. Очевидно, що у випадку, коли в системі знаходиться декілька об'єктів, то з конкретним моментом модельного часу пов'язаний набір записів у таблиці з інформацією про стан об'єктів. Як ідентифікатори таких наборів використовується визначене поле таблиці, у яке заноситься значення модельного часу.

Такий алгоритм зв'язку обраний із міркувань модульності і можливості розподіленої обробки (на декількох комп'ютерах) імітаційної моделі. Модульність системи полягає в тому, що вибір іншого продукту бази даних призведе до потреби змінити код модуля зв'язку із БД і не вплине на всю систему в цілому.

Розподілена обробка полягає в тому, що об'єкти системи є мережними додатками і не пов'язані із якимось фізичним комп'ютером. Наприклад, ядро і БД можуть бути встановлені на сервері, а Ви можете спостерігати за процесом моделювання з будь-якої робочої станції.

Через те що всі переміщення об'єктів заносяться і зберігаються в базі даних, то процес моделювання може бути відтворений у довільний час із будь-якого моменту внутрішнього модельного часу.

Кожному запущеному процесу ядра призначається ідентифікатор сесії (Session ID). Назвемо сесією множину записів, що занесена до БД у процесі роботи одного конкретного примірника ядра. Іншими словами сесія – це результати моделювання конкретної ділянки дороги. Інформація про запущені сесії і сесії, що відпрацювали, заноситься до БД. Також у БД заносяться дані (IP-адреса і номер порту) усіх запущених процесів ядра.

Часова синхронізація між графічною підсистемою і ядром здійснюється також за мережею. Кожен примірник графічної підсистеми після запуску, позв'язується з процесом ядра, повідомлякчи йому свою IP-адресу і номер порту, а потім починає спостерігати за сесією, яку він виконує. Після занесення чергового набору записів з інформацією про об'єкти до БД, ядро посилає мережний пакет клієнтам, що зареєструвалися в нього – примірникам графічної підсистеми. Формат пакета визначається спеціальним протоколом синхронізації SSP (Simulator synchronization protocol). У пакет включається поточне значення модельного часу ядра. Графічна підсистема фіксує в себе значення модельного часу для останнього відображеного набору. Такий механізм дозволить домогтися послідовного відображення інформації про об'єкти навіть у перевантажених мережах, де висока можливість утрати пакетів синхронізації. До того ж такий механізм дозволить черговому клієнту включитися в спостереження в будь-який момент роботи ядра.

Ядро імітатора нижнього рівня (LSS Core) написано мовою C++. Тому що мережні механізми протоколу TCP/IP більшості сучасних операційних систем (FreeBSD, Linux, Windows NT, Windows 98) грунтуються на коді BSD Sockets, розробленому в університеті Berkeley. Це дозволяє компілювати код ядра без особливих зусиль під будь-яку з ОС, у якій використовується механізм Sockets.

Основним компонентом ядра (рис. 4) є клас **диспетчер**. У його функції входить синхронізація роботи всіх компонентів ядра. Диспетчер містить список покажчиків на домени. Клас **таймер** здійснює контроль і облік модельного часу, запускає цикли моделювання.

Об'єкт класу **домен** представляє основні параметри ділянки дороги, такі як тип покриття, довжина домену. Клас містить покажчик на список машин, що знаходяться на цьому домені. Цей клас також клас містить ряд службових функцій:

drive – основна логіка керування автомобілями. На кожному циклі перераховується координати автомобілів, а також змінюється (за деяким алгоритмом) параметри автомобілів (прискорення) у залежності від обстановки.

- car_in – функція, що вводить у домен новий автомобіль.
- domain_exit_blocked – функція, яка визначає можливість покинути домен.

```
class domain
{   uint len;
    int uklon;
    int cover_type; // тип покриття ( асфальт, грунт, ями)
    int code; // номер домену
public:
    int domain_exit_blocked();
    clist *car_list; // список машин
    void car_in( car * ); // новий автомобіль
}
```

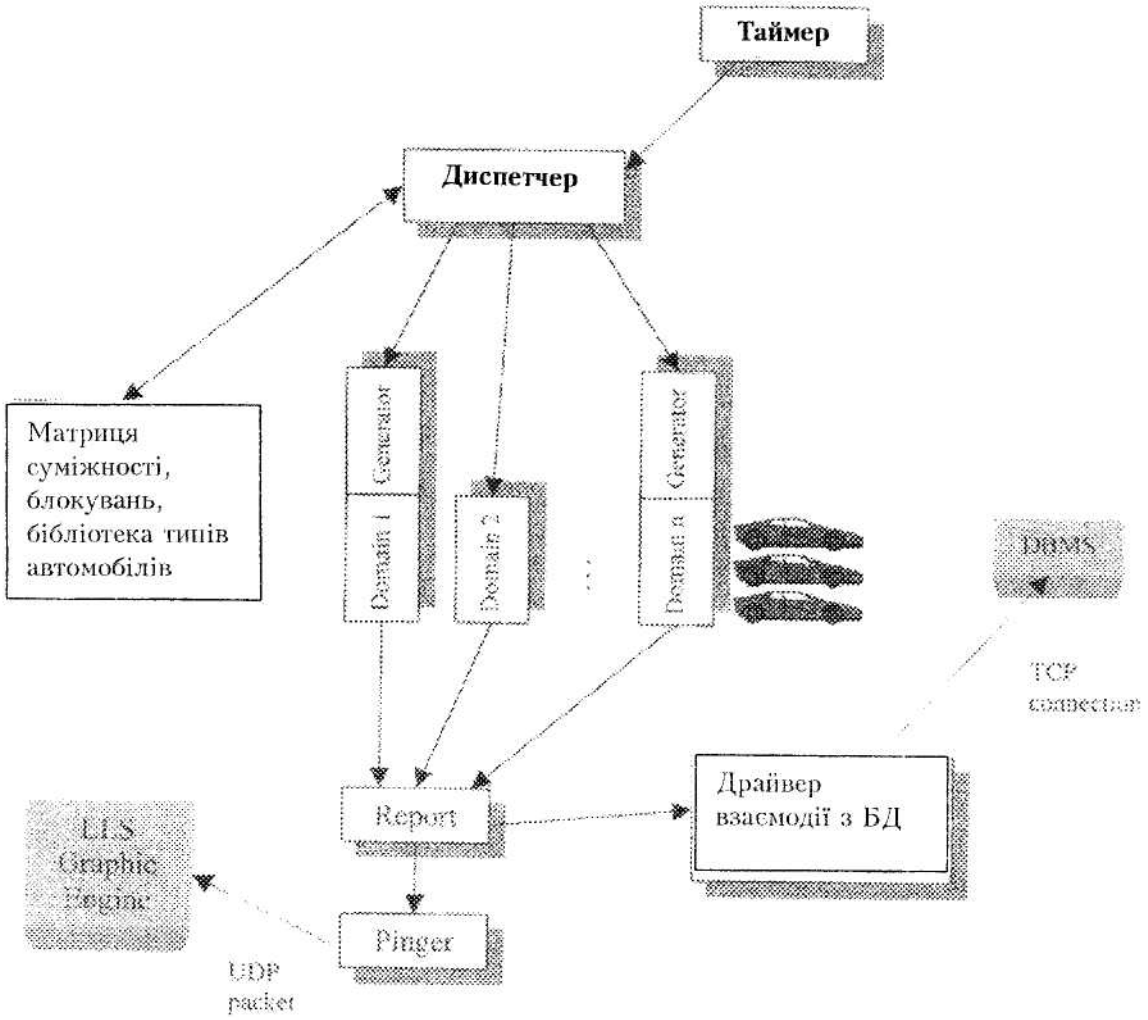

Рис. 4. Структура ядра імітатора нижнього рівня

Після приходу сигналу від таймера функція диспетчер запускає функцію **drive** для кожного домену із списку.

Алгоритм роботи функції drive можна звести до наступного:

1. Перерахунок координат і швидкостей автомобілів на основі їх властивостей;
2. Зміна параметра прискорення на основі дорожньої обстановки.

Для вхідних доменів можуть бути створені примірники класу **Generator**, що здійснюють створення об'єктів-автомобілів за поданою закономірністю, з визначеними характеристиками і здійснює їх включення в домени.

Клас **автомобіль** представляє прообраз реальних автомобілів. Об'єкт класу містить покажчик на домен, до якого він належить, основні поточні характеристики автомобіля (швидкість, прискорення, позиція в домені), покажчик на об'єкт класу car_type, що визначає деякі загальні властивості, що характерні для автомобіля даного типу.

```
class car
{   int type; car_type *characteristics;
    domain *parent;
    int speed, acc; // поточні характеристики
    int x_position; // координати передній частини машини в домені
}
```

Клас **car_type** визначає основні властивості автомобіля. Набір класів типу car_type складає бібліотеку типів автомобілів. Основні властивості, такі як максимальна швидкість, прискорення

розгону і гальмування, швидкість, що є найбільш бажаною, прискорення гальмування, ширина, довжина, маса, колір і тип кузова.

```
class car_type
{   uint width, len;
    int max_speed, max_acc, max_deacc;
    int prf_speed, prf_deacc; // швидкість, що є найбільш бажаною
    char color; // колір
    char body_type;
}
```

Службовий клас **ctype** – це інтерфейс для створення упорядкованих списків об'єктів. Він є аналогом компонента Tlist у системі Delphi.

```
class clist;
 {  car *first;
    int item_number;
public:
    int max_car_num ();
    car *get_by_num ( int index );
 }
```

Висновок

Імітаційний проект – зручний засіб моделювання процесу автомобільного руху на різноманітних ділянках доріг. Він може бути пов'язаний із топографічною картою району, міста або області, тому що імітатор докладно відображає мережі доріг і моделює індивідуальні переміщення транспортних засобів із використанням зміни смуг і логіки сигналів руху. Наявність засобів візуалізації дозволяє будувати саму імітаційну модель ділянок доріг, і відображати процес переміщення транспортних засобів під час моделювання.

ЛІТЕРАТУРА:

1. *Теленик С.Ф., Томашевский В.Н.* Концепция моделирования и управления движением автотранспортных средств // Автомобильный транспорт. Сб. науч. трудов. – Вып. 1. Харьков: ХГАДТУ, 1998. – С. 98–100.
2. *Томашевський В.М., Печенежський Д.С.* Концептуальні основи імітаційного моделювання автомобільного дорожнього руху // Праці Української конференції з автоматичного управління "Автоматика-98". – Ч. III. – Київ: НТТУ "КПІ", 1998. – С. 317–323.

ПЕЧЕНЕЖСЬКИЙ Дмитро Сергійович – аспірант кафедри АСОІУ, Національний технічний університет України "КПІ".

Наукові інтереси:
- імітаційне моделювання розподілених систем;
- візуальне моделювання.

ТОМАШЕВСЬКИЙ Валентин Миколайович – професор кафедри АСОІУ, Національний технічний університет України "КПІ".

Наукові інтереси:
- імітаційне моделювання дискретних систем;
- засоби автоматизації створення імітаційних моделей;
- прискорення процесів моделювання;
- статистична обробка інформації та стохастичне прогнозування;
- Інтернет-технології;
- дистанційне навчання.

E-mail: simtom@svitonline.com

Подано 22.09.2000