УДК 681.3.06

А.М. Ковальчук, аспір.
Житомирський інженерно-технологічний інститут

ОСОБЛИВОСТІ РЕАЛІЗАЦІЇ БАГАТОПОТОЧНОЇ ВІЗУАЛІЗАЦІЇ ДАНИХ В ПРОГРАМНІЙ ПІДСИСТЕМІ *"Graph Server"*

(Представлено к.т.н., доц. М.М. Колодницьким)

Розглянуто особливості реалізації багатопоточної візуалізації даних в програмній підсистемі "Graph Server", яка є частиною програмного комплексу "DSR Open Lab 1.0". Представлено схеми взаємодії потоків та схему обміну даними між ними. Визначено вимоги для забезпечення цілісності даних при сумісному доступі до них з різних потоків. Представлено результати роботи розглянутої підсистеми.

В статті представлено реалізацію механізму багатопоточної візуалізації даних, який використовується у прикладній програмній підсистемі "*Graph Server*". Вона створена в рамках проекту "*DSR Open Lab 1.0*" [1, 2] (Dynamical System Research Laboratory 1.0), призначеного для вирішення задач математичного моделювання та орієнтованого на використання в наукових дослідженнях та в навчальному процесі. Особливого значення задача представлення результатів дослідження набуває при використанні програмного комплексу в навчальному процесі, тобто якщо користувач – це учень або студент. Отже, програмна підсистема "*Graph Server*" як інструмент, що використовується для візуалізації даних, має забезпечити користувача максимально повним набором дій над зображенням. Однак існують дії, реалізувати які без використання додаткових потоків неможливо. До них можна віднести, наприклад: збільшення або зменшення масштабу зображення в автоматичному режимі, обертання зображення навколо координатних осей у автоматичному режимі, режим осцилографа, анімація (режим відображення з затримкою у часі), паралельний процес візуалізації декількох наборів даних. Тому використання додаткових потоків для візуалізації даних є необхідним.

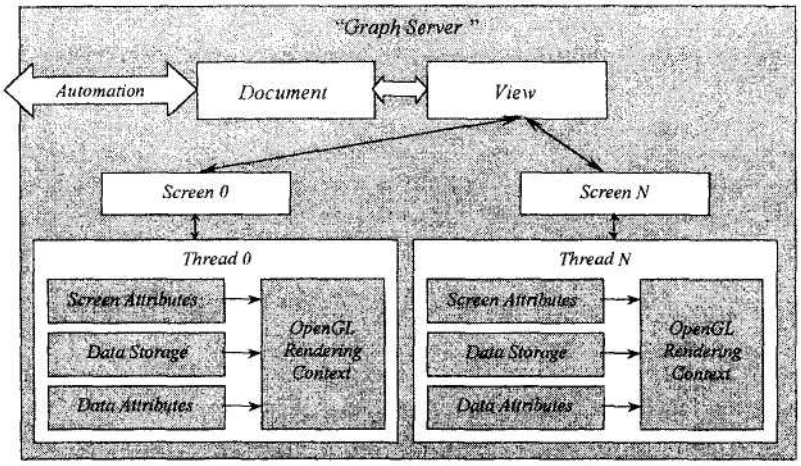

Рис. 1. Структурна схема програмної підсистеми "Graph Server"

Підсистема побудована як *Active Document Server* [3] з використанням *Document/View* архітектури, тому набори даних (*data sets*) передаються до документу (*Document*) за допомогою методів автоматизації (*OLE Automation*) і далі потрапляють до вікна представлення (*View*), яке, в свою чергу, передає їх до вікна екрану (*Screen*). Вікно-екран за допомогою спеціальної системи повідомлень передає дані до відповідного потоку (*Thread*), який, власне, і відповідає за їх візуалізацію. Як видно з рис. 1, кожне вікно представлення може мати кілька вікон екранів, тобто в одному вікні представлення можуть одночасно відображатися декілька принципово різних наборів даних. Кожен екран (*Screen*) є контейнером для додаткового потоку, він відповідає за його створення, знищення та взаємодію з ним.

Вислів "багатопоточна візуалізація даних" у програмній підсистемі "*Graph Server*" означає, що процес візуалізації різнотипних наборів даних (детальніше див. [4]) відбувається одночасно. Такого результату було досягнуто завдяки використанню додаткових потоків, які

працюють паралельно з головним потоком підсистеми. Для взаємодії між об'єктами, що знаходяться в головному потоці підсистеми та об'єктами з додаткових потоків було розроблено систему обробки міжпотокових повідомлень, яка має дві схеми роботи з повідомленнями: *PostMessage* та *SendMessage* (рис. 2).

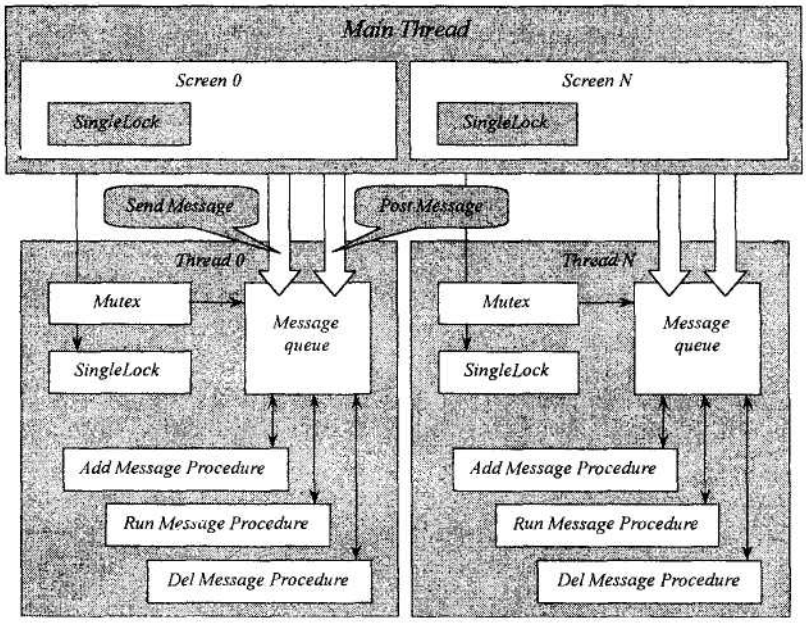

Рис. 2. Структурна схема взаємодії між потоками

Розглянемо механізм надсилання та обробки повідомлень за схемою *PostMessage*. Процес надсилання повідомлень відбувається у такій послідовності:

Доступ до додаткового потоку блокується, для цього використовуються об'єкти синхронізації *SingleLock*, *Mutex*.

Аналізується вміст черги повідомлень (*Message Queue*) і нове повідомлення заноситься в кінець цієї черги, для кожного типу повідомлень використовується індивідуальний підхід.

Доступ до додаткового потоку розблоковується.

Відновлюється робота додаткового потоку.

Цикл обробки повідомлень, надісланих за схемою *PostMessage*, відбувається у такій послідовності:

Доступ до додаткового потоку блокується.

Повідомлення виймається з черги повідомлень.

Доступ до додаткового потоку розблоковується.

Визивається процедура обробки повідомлення.

Визивається процедура знищення повідомлення.

Перевіряється, чи залишились повідомлення в черзі повідомлень, якщо залишились, то цикл обробки повторюється знову, якщо ні, то робота додаткового потоку зупиняється.

Розглянемо механізм надсилання та обробки повідомлень за схемою *SendMessage*. Процес надсилання відбувається у такій послідовності:

Доступ до додаткового потоку блокується.

Нове повідомлення заноситься на початок черги повідомлень.

Доступ до додаткового потоку розблоковується.

Відновлюється робота додаткового потоку.

Головний потік зупиняється на час, необхідний для обробки повідомлення, яке надсилається.

Процес обробки повідомлень, надісланих за схемою *SendMessage*, відбувається у такій послідовності:

Доступ до додаткового потоку блокується.

Повідомлення виймається з черги повідомлень.

Визивається процедура обробки повідомлення.

Доступ до додаткового потоку розблоковується.

Перевіряється, чи залишились повідомлення в черзі повідомлень, якщо залишились, то цикл обробки для повідомлень, надісланих за схемою *PostMessage*, повторюється знову, якщо ні, то робота додаткового потоку зупиняється.

Для доступу до даних, що зберігаються в допоміжному потоці, надсилаємо повідомлення за схемою *SendMessage* та під час обробки таких повідомлень робимо копію потрібних даних. Така схема дає можливість забезпечити цілісність даних при сумісному доступі до них з різних потоків, а саме – з головного потоку програмної підсистеми та допоміжного.

Також для збереження часової послідовності в процесі візуалізації було введено таке правило: надсилати повідомлення, які відповідають за модифікацію даних, за схемою *PostMessage*. Це, звичайно, не виключає можливості використання схеми *SendMessage*, але у такому випадку можна не досягнути бажаних результатів.

На рис. 3 представлено результати візуалізації 3D наборів даних за допомогою програмної підсистеми "*Graph Server*".

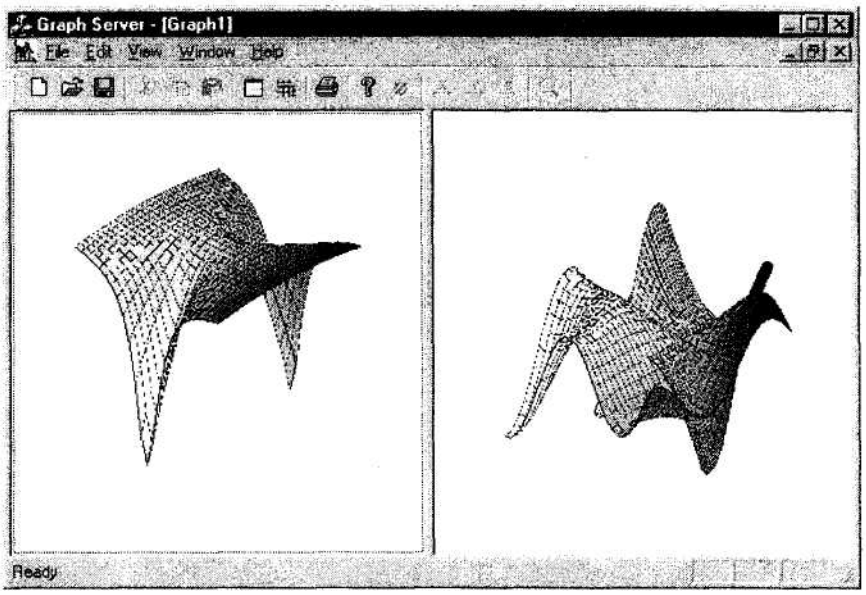

Рис. 3. Результати візуалізації 3D наборів даних

ЛІТЕРАТУРА:

1. *Колодницький М.М.* Тривимірна компонентна архітектура прикладної програмної системи "Dsr Open Lab 1.0" як втілення концепцій реінженерії // Теорія програмування. – 1998. – № 4. – С. 37–45.
2. *Kolodnytsky M., Ivanitsky I., Kovalchuk A., Kuryata S., Levitsky V.* "DSR Open Lab 1.0" – software system for simulation // Proceedings of 21st International Conference on Information Technology Interfaces, Pula, 1999. – P. 31.
3. *Грегори К.* Использование Visual C++ 5. – К.: Диалектика, 1997. – 816 с.
4. *Колодницький М.М.* Типологія математичних моделей технічних систем. Частина 2 // Вісник ЖІТІ. – 1998. – № 7. – С. 208–218.

КОВАЛЬЧУК Андрій Михайлович – аспірант Житомирського інженерно-технологічного інституту.

Наукові інтереси:
- комп'ютерні інформаційні технології;
- моделювання та розв'язок задач за допомогою обчислювальної техніки;
- використання обчислювальної техніки в навчальному процесі;
- комп'ютерна графіка.

Подано 11.09.2000