УДК 004.4'24 https://doi.org/10.15407/pp2019.02.003

А.Ю. Дорошенко, Р.В. Кушніренко, О.А. Яценко

ПРОЕКТУВАННЯ ПРОГРАМИ ВІЗУАЛІЗАЦІЇ ЗЕМНОЇ ПОВЕРХНІ З ВИКОРИСТАННЯМ АЛГЕБРО-АЛГОРИТМІЧНОГО ІНСТРУМЕНТАРІЮ

Однією з важливих задач в рамках метеорологічного прогнозування є комп'ютерна візуалізація отриманих результатів прогнозу, зокрема, тривимірне моделювання рельєфу земної поверхні. Одним із найпопулярніших програмних засобів розробки застосунків в області візуалізації даних є OpenGL – специфікація, що визначає незалежний від мови програмування кросплатформний програмний інтерфейс для написання застосунків, що використовують двовимірну й тривимірну графіку. Візуалізація даних є досить складною задачею, тому актуальним є питання створення спеціальних засобів автоматизації, що дозволяли б генерувати програмний код для задач з даної предметної області. У статті виконане налаштування раніше створеного алгебро-алгоритмічного інструментарію на проектування та синтез OpenGL програм. Автоматизоване конструювання програм здійснюється з використанням високорівневої мови, що ґрунтується на системах алгоритмічних алгебр Глушкова. Підхід продемонстровано на проектуванні програми візуалізації рельєфу підстильної поверхні для задачі прогнозування погоди.

Ключові слова: автоматизоване проектування програм, алгебра алгоритмів, візуалізація даних, метеорологічне прогнозування, рельєф земної поверхні, синтез програм, тривимірне моделювання, OpenGL.

Вступ

Системи прогнозування стають все більш актуальними у зв'язку зі впровадженням різноманітних автоматизованих систем прийняття рішень, керування та оцінки ризиків у різних сферах людської діяльності. Зокрема, метеорологічне прогнозування [1], окрім задоволення потреб звичайних користувачів, використовується для розв'язання різноманітних задач у транспортній галузі (наприклад, авіації), сільському господарстві, наукових дослідженнях тощо. Суттєве значення має актуальність отриманого прогнозу, його достовірність та оперативність його отримання. Не менш важливим є наочність відображення отриманих даних.

В роботі [2] спроектовано та реалізовано сервісно-орієнтовану портальну систему для надання послуг метеорологічного прогнозування на мультипроцесорній платформі. В рамках системи виконана двовимірна візуалізація результатів прогнозу (ізоліній) за допомогою OpenGL на графічному прискорювачі [3]. OpenGL [4] є одним із найпопулярніших програмних засобів розробки застосунків у області візуалізації даних. Він визначає незалежний від мови програмування кросплатформний програмний інтерфейс для написання програм, що використовують двовимірну й тривимірну графіку. Реалізація OpenGL підтримується більшістю виробників апаратних і програмних платформ і включає декілька бібліотек. Відмітимо, що візуалізація даних є досить складною задачею, тому актуальним є питання розробки спеціальних засобів автоматизації, що дозволяли б генерувати програмний код для таких задач.

В роботах [5–7] авторами запропоновані теорія, методологія та інструментарій для автоматизованого проектування, синтезу та перетворення програм, що ґрунтуються на засобах алгебр алгоритмів та переписувальних правил. Предметною областю застосування інструментальних засобів, зокрема, є метеорологічне прогнозування.

У даній статті виконане налаштування алгебро-алгоритмічного інструментарію на проектування та генерацію OpenGL програм. Застосування інструментарію продемонстроване на задачі тривимірної візуалізації рельєфу підстильної поверхні. Підстильна поверхня є сукупністю природних і перетворених (змінених та створених людиною) ландшафтів на земній поверхні, що знаходяться у взаємодії з атмосферою в процесі обміну теплом і вологою [1].

1. Інструментарій алгебр алгоритмів та його налаштування на конструювання OpenGL програм

Підхід до проектування програм, що розглядається в даній роботі, ґрунтується на засобах систем алгоритмічних алгебр (САА) В.М. Глушкова [5, 6]. САА призначені для формалізованого подання схем алгоритмів і програм та їх трансформації, зокрема, з метою оптимізації за певними критеріями. САА покладено в основу інструментарію автоматизованої розробки програмного забезпечення [5–7].

1.1. Основні конструкції та інструментарій алгоритмічних алгебр. САА Глушкова (або алгебра Глушкова) є двоосновною алгеброю

$$GA = < \{Pr, Op\};\ \Omega_{GA} >,$$

де Pr й Op – множини логічних умов й операторів, визначених на інформаційній множині IS; Ω_{GA} – сигнатура операцій. Оператори є відображеннями (можливо, частковими) інформаційної множини в себе, логічні умови – предикати на множині IS, які приймають значення тризначної логіки $E_3 = \{0, 1, \mu\}$, де 0 – "хибність", 1 – "істина", μ – "невизначеність". Сигнатура $\Omega_{GA} = \Omega_1 \cup \Omega_2$ складається з системи Ω_1 логічних операцій (диз'юнкції, кон'юнкції, заперечення, лівого множення оператора на умову), що приймають значення в множині Pr, і системи Ω_2 операторних операцій, що приймають значення в множині Op (композиції, альтернативи, циклу та ін.). Операторні конструкції будуть розглянуті далі.

Предикати й оператори поділяються на базисні та складені. Базисні поняття вважаються атомарними, неподільними абстракціями та пов'язані з предметною областю алгоритму, що проектується. Складені елементи будуються з базисних за допомогою операцій послідовного й паралельного виконання операторів та/або логічних операцій.

На САА ґрунтується алгоритмічна мова САА/1 [5, 6], призначена для багаторівневого структурного проектування й документування послідовних та паралельних алгоритмів і програм. Перевагою її використання є можливість опису алгоритмів у природно-лінгвістичній формі. Подання операторів мовою САА/1 називаються САА-схемами.

Основними операторними конструкціями мови САА/1 є такі:

- композиція (послідовне виконання операторів):

"*operator* 1";
"*operator* 2";

- альтернатива (розгалуження):

IF '*condition*' THEN
  "*operator* 1"
ELSE
  "*operator* 2"
END IF;

- цикл:

WHILE '*condition*'
LOOP "*operator*"
END OF LOOP.

В роботах [5–7] розглядаються додаткові операції, призначені для формалізованого проектування паралельних алгоритмів.

Інструментарій проектування та синтезу програм (ІПС) ґрунтується на використанні САА й інтегрує три форми подання алгоритмів: аналітичну, природно-лінгвістичну й графову.

Основними компонентами системи ІПС є такі:

- діалоговий конструктор синтаксично правильних програм (ДСП-конструктор), призначений для проектування схем алгоритмів та синтезу програм цільовими мовами програмування Java, C, C++;
- редактор граф-схем;
- база даних, у якій зберігається опис конструкцій САА і базисних понять, а також шаблони їх програмних реалізацій;

- генератор САА-схем на основі схем більш високого рівня (гіперсхем).

За допомогою ДСП-конструктора виконується порівневе проектування схем алгоритмів зверху вниз за допомогою деталізації мовних конструкцій САА. На кожному кроці конструювання система надає користувачу на вибір лише ті конструкції, підстановка яких у текст схеми, що формується, не порушує її синтаксичну правильність. ДСП-конструктор використовує список конструкцій САА і дерево конструювання алгоритму. У процесі проектування обрані користувачем операції відображаються в дереві з подальшою деталізацією їхніх змінних. Залежно від типу обраної змінної (логічної або операторної), система пропонує відповідний список операцій САА. На основі побудованого дерева виконується генерація схеми алгоритму та тексту програми цільовою мовою програмування.

З метою налаштування системи ІПС на проектування й синтез OpenGL програм, в базу даних системи включені нові конструкції, що розглядаються у наступному підрозділі.

1.2. Розширення САА конструкціями, орієнтованими на проектування OpenGL програм. Основним принципом роботи OpenGL [4] є отримання наборів векторних графічних примітивів у вигляді точок, ліній та багатокутників з подальшою математичною обробкою отриманих даних та побудовою растрової картинки на екрані та/або в пам'яті. Векторні трансформації та растеризація виконуються графічним конвеєром, який можна розглядати як дискретний автомат. Команди OpenGL потрапляють в одну з двох груп: вони або додають графічні примітиви на вхід конвеєра, або конфігурують конвеєр на різне виконання трансформацій.

Основна бібліотека OpenGL обробляє і відображує у буфері кадру графічні примітиви з урахуванням певного числа вибраних режимів. Кожний примітив – це точка, відрізок, багатокутник і т. д. Кожний режим може бути змінений незалежно від інших. Визначення примітивів, вибір режимів та інші операції описуються за допомогою команд у формі викликів функцій прикладної бібліотеки. Примітиви визначаються набором з однієї або більше вершин. Вершина визначає точку, кінець відрізку або кут багатокутника. З кожною вершиною асоціюються певні дані (координати, колір, нормаль, текстурні координати і т. п.), що називаються атрибутами. У переважній більшості випадків кожна вершина обробляється незалежно від інших.

В рамках OpenGL використовується VBO (Vertex Buffer Objects) – технологія, що дозволяє зберігати координати вершин спільно з їх атрибутами у відеопам'яті [4]. Робота з VBO складається з таких етапів:

1) створення буферів, передача в буфери координат і атрибутів вершин, передача даних у відеопам'ять;

2) активація потрібного буфера, візуалізація (рендерінг) буфера та його деактивація;

3) видалення даних у відеопам'яті.

Існує низка бібліотек, створених над або на додаток до OpenGL. Зокрема, в програмі візуалізації поверхні, що розглядається в даній роботі, використовується GLFW (Graphics Library Framework) [8] – кросплатформна бібліотека з відкритим кодом для створення вікон, контексту OpenGL і керування введенням, а також бібліотека GLM (OpenGL Mathematics) – математика для OpenGL [9].

Далі наведено список основних базисних операторів, призначених для проектування OpenGL програм і доданих в алгебру алгоритмів.

1. Створення контролера вікна:

"Create and initialize window controller (*controller*)",

що включає ініціалізацію бібліотеки GLFW та створення вікна застосунку.

2. Генерація імені масиву атрибутів вершин зі збереженням його у змінній *var*:

"Generate vertex attribute array name (*var*)".

3. Генерація імені буфера:

"Generate buffer name (*var*)".

4. Зв'язування масиву атрибутів вершин з іменем:

"Bind the vertex attribute array to name (*var*)".

5. Зв'язування буферу з атрибутами вершин:

"Bind the buffer (*buffer*) to vertex attributes".

6. Копіювання масиву даних користувача в буфер атрибутів вершин:

"Copy the user data (*data*) to the vertex attributes buffer".

7. Зв'язування буферу з індексами масиву вершин:

"Bind the buffer (*buffer*) to vertex array indices".

8. Копіювання масиву даних користувача в буфер індексів вершин:

"Copy the user data (*data*) to the vertex indices buffer".

9. Зберігання положення атрибута *atr_name* в шейдерній програмі *program* у змінну *var*:

"Save the location of an attribute (*atr_name*) in the shader program (*program*) to variable (*var*)".

10. Активація масиву атрибутів вершин:

"Enable a generic vertex attribute array (*arr*)".

11. Видалення зв'язку масиву атрибутів вершин з іменем:

"Unbind the vertex attribute array to a name".

12. Створення та використання шейдерної програми (коду, що виконується на графічному прискорювачі) включає в себе такі оператори:

"Create shader program (*prog_id*)";
"Create vertex/fragment shader (*var*) based on the source file (*file*)";
"Attach shader (*shader_id*) to program (*prog_id*)";
"Link shader program (*prog_id*)";
"Check shader program (*prog_id*)";
"Use shader program (*prog_id*)",

де *prog_id* – цілочислена змінна, у якій заданий ідентифікатор програми; *file* – файл з вихідним програмним кодом вершинного або фрагментного шейдера. Шейдери під'єднуються до програми *prog_id*, після чого виконується її збирання, перевірка та використання.

13. Установка матриці моделі виду проекції (Model View Projection, MVP) для контролера і шейдерної програми:

"Set MVP matrix for controller (*controller*) and shader program (*program*)".

14. Візуалізація трикутників на основі масиву даних користувача:

"Render triangles from the user array data (*data*)".

15. Видалення масиву атрибутів вершин:

"Delete vertex attribute array (*arr*)".

16. Видалення буферу:

"Delete buffer (*buffer*)".

Приклад використання вищерозглянутих конструкцій наведено у розділі 2.

2. Застосування системи ІПС для проектування програми візуалізації рельєфу

У даному розділі розглядається використання інтегрованого інструментарію для автоматизованої розробки OpenGL

програми візуалізації рельєфу підстильної поверхні.

Далі наведено САА-схему основного алгоритму візуалізації, побудовану за допомогою системи ІПС. Схема містить один складений оператор – main, у якому використовуються операції САА та базисні оператори, розглянуті у попередньому розділі. У схемі, зокрема, вказано такі змінні: *terrain* – масив даних користувача (точки поверхні, задані координатами *x, y, z*), що зчитуються з бінарного файлу *input_bin*; *VAO* – масив атрибутів вершин; *VBO* – буфер атрибутів вершин; *EBO* – буфер індексів вершин.

```
SCHEME TERRAIN SURFACE
VISUALIZATION ====
    "Terrain surface visualization using
     OpenGL"
  END OF COMMENTS

"main"
==== "Create and initialize window
       controller (controller)";
     "Create mesh (terrain) based on the
       binary data from file (input_bin)";
     "Create shader program (program)";
     "Declare the list of variables (VAO,
      VBO, EBO) of type (unsigned int)";
     "Generate vertex attribute array name
      (VAO)";
     "Generate buffer name (VBO)";
     "Generate buffer name (EBO)";
     "Bind the vertex attribute array to name
       (VAO)";
     "Bind the buffer (VBO) to vertex
       attributes";
     "Copy the user data (terrain) to the
       vertex attributes buffer";
     "Bind the buffer (EBO) to vertex
       array indices";
     "Copy the user data (terrain) to the
       vertex indices buffer";
     "Save the location of an attribute
       (aPos) in the shader program
       (program) to variable (vertloc)";
     "Define an array of generic vertex
       attribute data (vertloc) of type
       (GL_FLOAT) and size
       (3 * sizeof(float))";
     "Enable a generic vertex attribute array
      (vertloc)";
     "Unbind the vertex attribute array
       to a name";
     WHILE NOT 'Window controlled by
                (controller) should
                close'
     LOOP
       "Specify clear values for the color
        buffers (0.0f, 0.0f, 0.5f, 1.0f)";
       "Clear colour and depth buffers";
       "Enable depth test";
       "Disable polygon culling";
       "Use shader program (program)";
       "Set MVP matrix for controller
        (controller) and shader program
        (program)";
       "Bind the vertex attribute array to
        name (VAO)";
       "Render triangles from the user array
        data (terrain)";
       "Unbind the vertex attribute array
        to a name"
     END OF LOOP;
     "Delete vertex attribute array (VAO)";
     "Delete buffer (VBO)";
     "Delete buffer (EBO)";
     "Return value (0)";

END OF SCHEME TERRAIN SURFACE
VISUALIZATION
```

На рис. 1 показано копію екрану системи ІПС з побудованою схемою.

Далі наведено САА-схему DATAREADER, що містить визначення основних структур даних задачі (вершина *Vertex*, вектори вершин *Points* та індексів *Indices*, сітка *Mesh*), а також загальне визначення класу *DataReader*, призначеного для зчитування точок поверхні з файлу.

```
SCHEME DATAREADER ====
    "Data reader for the terrain surface
     visualization task (header file)"
  END OF COMMENTS
```

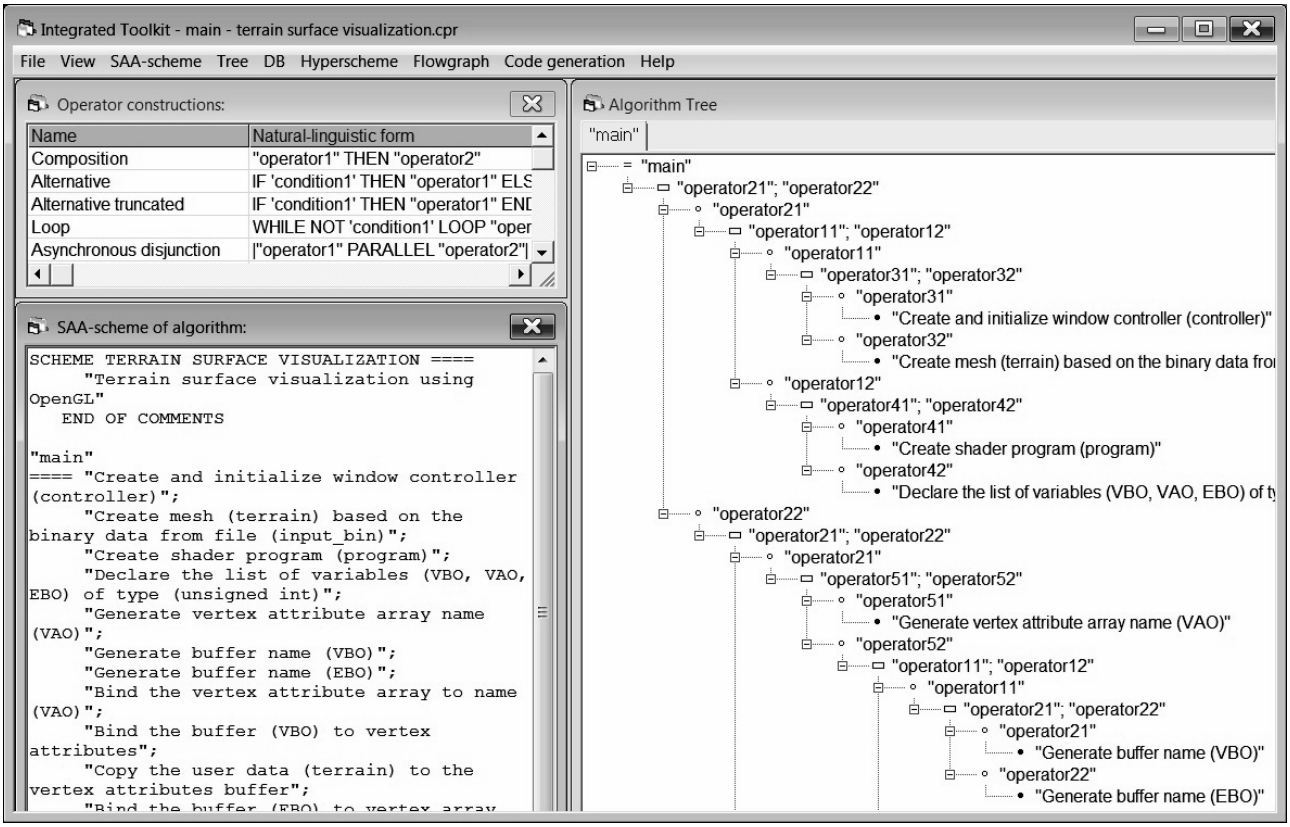


Рис. 1. Копія екрану системи ІПС з побудованою САА-схемою алгоритму візуалізації поверхні

STRUCTURE *Vertex* (
  “Declare a variable (*x*) of type (*float*)”;
  “Declare a variable (*y*) of type (*float*)”;
  “Declare a variable (*z*) of type (*float*)”);

“Define vector type (*Points*) of type (*Vertex*)”;
“Define vector type (*Indices*) of type (*unsigned*)”;

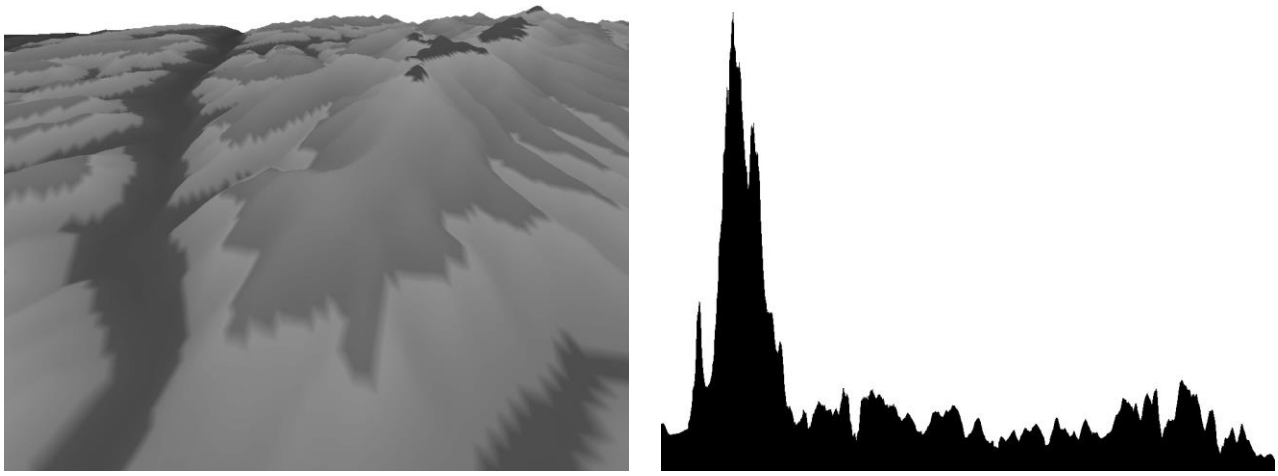
STRUCTURE *Mesh* (
  CONSTRUCTOR *Mesh*(“Declare a constant (*points*) of type (*Points&*)”);
  “Declare a constant (*points*) of type (*Points*)”;
  “Declare a constant (*indices*) of type (*Indices*)”);

CLASS *DataReader*
  PUBLIC:
    CONSTRUCTOR *DataReader*(“Declare a string constant (*filename*)”);
    METHOD *getMesh*(): *Mesh*;
  PRIVATE:
    “Declare a variable (*mesh*) of type (*Mesh*)”
END OF CLASS *DataReader*;

END OF SCHEME DATAREADER

На основі побудованих САА-схем в системі ІПС виконана генерація програмного коду C++, що використовує OpenGL. Окрім реалізації основної функції (main), програма містить чотири основні класи: Controller (контролер вікна застосунку), DataReader (зчитування вхідних даних), Shader та ShaderProgram (шейдерна програма).

Проведено експеримент з виконання розробленої OpenGL програми на графічному прискорювачі. На рис. 2, *а* наведено тривимірне зображення рельєфу підстильної поверхні у межах області моделювання. На рис. 2, *б* показано вертикальний переріз поверхні уздовж 48° пн. ш.



а *б*

Рис. 2. Результати роботи програми візуалізації рельєфу підстильної поверхні:
а – тривимірне зображення рельєфу у межах області моделювання,
б – вертикальний переріз поверхні уздовж 48° пн. ш.

Висновки

Виконане налаштування алгебро-алгоритмічного інструментарію на проектування та синтез OpenGL програм. Автоматизоване конструювання програм виконується з використанням високорівневої мови, що ґрунтується на системах алгоритмічних алгебр Глушкова та орієнтована на природно-лінгвістичну форму подання алгоритмів. В основу інструментарію покладене діалогове проектування схем алгоритмів зверху вниз на основі суперпозиції операцій алгебри алгоритмів. На основі схем виконується генерація коду цільовою мовою програмування. Підхід продемонстровано на проектуванні програми візуалізації рельєфу підстильної поверхні для задачі прогнозування погоди.

Література

1. Прусов В.А., Дорошенко А.Ю. Моделювання природних і техногенних процесів в атмосфері. Київ: Наукова думка, 2006. 542 с.
2. Дорошенко А.Ю., Іваненко П.А., Овдій О.М., Павлючин Т.О., Вітряк Є.А. До створення Інтернет-порталу надання послуг метеорологічного прогнозування на мультипроцесорній платформі. *Проблеми програмування*. 2015. № 3. С. 24–32.
3. Бекетов О.Г., Вітряк Є.А., Мироненко І.О., Овдій О.М. Розвиток Інтернет-порталу метеорологічного прогнозування на мультипроцесорній платформі. *Проблеми програмування*. 2016. № 2–3. С. 246–253.
4. OpenGL – The Industry Standard for High Performance Graphics [Електронний ресурс]. Режим доступу до ресурсу: https://www.opengl.org (дата звернення 19.04.2019 р.).
5. Андон Ф.И., Дорошенко А.Е., Жереб К.А., Шевченко Р.С., Яценко Е.А. Методы алгебраического программирования. Формальные методы разработки параллельных программ. Киев: Наукова думка, 2017. 440 с.
6. Андон Ф.И., Дорошенко А.Е., Цейтлин Г.Е., Яценко Е.А. Алгеброалгоритмические модели и методы параллельного программирования. Киев: Академпериодика, 2007. 631 с.
7. Doroshenko A., Zhereb K., Yatsenko O. Developing and optimizing parallel programs with algebra-algorithmic and term rewriting tools. *Communications in Computer and Information Science. Information and Communication Technologies in Education, Research, and Industrial Applications*. Berlin: Springer, 2013. Vol. 412. P. 70–92.
8. GLFW [Електронний ресурс]. Режим доступу до ресурсу: https://www.glfw.org (дата звернення 19.04.2019 р.).
9. GLM. OpenGL Mathematics [Електронний ресурс]. Режим доступу до ресурсу: https://glm.g-truc.net/0.9.9/index.html (дата звернення 19.04.2019 р.).

References

1. Prusov V.A. & Doroshenko A.Yu. (2006) Simulation of natural and anthropogenic processes in the atmosphere. Kyiv: Naukova dumka. (in Ukrainian)
2. Doroshenko A.Yu., Ivanenko P.A., Ovdii O.M., Pavliuchyn T.O. & Vitriak I.A. (2015) Creation of an Internet portal providing meteorological forecasting services on multiprocessor platform. Problems in programming. (3). P. 24–32. (in Ukrainian)
3. Beketov O.G., Vitriak E.A., Myronenko I.O., Ovdii O.M. (2016) Development of meteorological forecasting web portal on multiprocessor platform. Problems in programming. (2–3). P. 246–253. (in Ukrainian)
4. OpenGL – The Industry Standard for High Performance Graphics. [Online] Available from: https://www.opengl.org [Accessed: 19 April 2019]
5. Andon P.I. et al. (2017) Methods of algebraic programming. Formal methods of parallel program development. Kyiv: Naukova dumka. (in Russian)
6. Andon P.I. et al. (2007) Algebra-algorithmic models and methods of parallel programming. Kyiv: Akademperiodyka. (in Russian)
7. Doroshenko A., Zhereb K. & Yatsenko O. (2013) Developing and optimizing parallel programs with algebra-algorithmic and term rewriting tools. In Proc. 9th International Conference "ICT in Education, Research, and Industrial Applications" (ICTERI 2013), Revised Selected Papers. Kherson, Ukraine, 19-22 June 2013. Berlin: Springer. 412. P. 70-92.
8. GLFW. [Online] Available from: https://www.glfw.org [Accessed: 19 April 2019]
9. GLM. OpenGL Mathematics. [Online] Available from: https://glm.g-truc.net/0.9.9/index.html [Accessed: 19 April 2019]

Одержано 25.04.2019

***Про авторів*:**

Дорошенко Анатолій Юхимович,
доктор фізико-математичних наук,
професор, завідувач відділу теорії
комп'ютерних обчислень,
професор кафедри автоматики та
управління в технічних системах
НТУУ "КПІ імені Ігоря Сікорського".
Кількість наукових публікацій в
українських виданнях – понад 150.
Кількість наукових публікацій в
зарубіжних виданнях – понад 50.
Індекс Хірша – 5.
http://orcid.org/0000-0002-8435-1451,

Кушніренко Роман Владиславович,
магістр кафедри Теорії та технології
програмування факультету комп'ютерних
наук та кібернетики.
https://orcid.org/0000-0002-1990-8727,

Яценко Олена Анатоліївна,
кандидат фізико-математичних наук,
старший науковий співробітник.
Кількість наукових публікацій в
українських виданнях – 44.
Кількість наукових публікацій в
зарубіжних виданнях – 14.
Індекс Хірша – 2.
http://orcid.org/0000-0002-4700-6704.

Місце роботи авторів:

Інститут програмних систем
НАН України,
03187, м. Київ,
проспект Академіка Глушкова, 40.
Тел.: (044) 526 3559.

Київський національний університет
імені Тараса Шевченка,
01601, Київ, вул. Володимирська, 60.

E-mail: doroshenkoanatoliy2@gmail.com,
romashka1996@gmail.com,
oayat@ukr.net