УДК 519.6

А. В. Миргородський
Є. С. Воронін

ЗАСТОСУВАННЯ ЛІНІЙНОЇ АЛГЕБРИ У РОБОТІ З КОМП'ЮТЕРНОЮ ГРАФІКОЮ

Вінницький національний технічний університет

Анотація

Наведено напрями використання вищої математики в IT-сфері на прикладі комп'ютерної 2D та 3D графіки, розглянуто застосування лінійної алгебри при роботі із сучасними низькорівневими графічними API.

Ключові слова: математика, лінійна алгебра, IT, комп'ютерна графіка.

Abstract

The directions of using Higher Mathematics in the IT sphere on the example of computer 2D and 3D graphics are presented, the use of linear algebra for work with modern low-level graphic API is considered.

Keywords: mathematics, linear algebra, IT, computer graphics.

Вступ

Відомо, що математика розвиває логічне мислення та дозволяє оптимізувати процеси роботи в будь-якій сфері життя. Особливо це стосується IT. Застосування окремих розділів математики має місце в таких областях як штучний інтелект, нейронні мережі, машинне навчання, комп'ютерний зір, криптографія, 2Д та 3Д графіка. В своїй роботі зосередимо увагу на застосуванні лінійної алгебри в 2Д та 3Д графіці.

Результати дослідження

В комп'ютерній графіці найчастіше використовують такі 3Д редактори як 3Ds Max, 4D Cinema, Blender. Основою цих програми є низькорівневі прикладні програмні інтерфейси (API) для відображення графіки. Найпопулярнішими сучасними графічними API є Direct3D, OpenGL та Vulkan [1-2].

Графічний API дає можливість зберігати в пам'яті відеокарти будь-які дані, обробляти їх за допомогою графічного процесора та виводити їх на екран. Якщо точніше, то будь-який об'єкт можна представити в вигляді набору точок. Для прикладу розглянемо OpenGL. В ньому використовується система NDC або Normalized Device Coordinates, що зображена на рис. 1. Вона представлена у вигляді куба, побудованого на координатних осях з центром в початку координат [1]. Координати X, Y та Z приймають значення з відрізку $[-1, 1]$. Ми завжди отримуємо зображення цих точок, дивлячись на них з однієї фіксованої грані куба. Це базові принципи роботи цього API, які неможливо змінити.

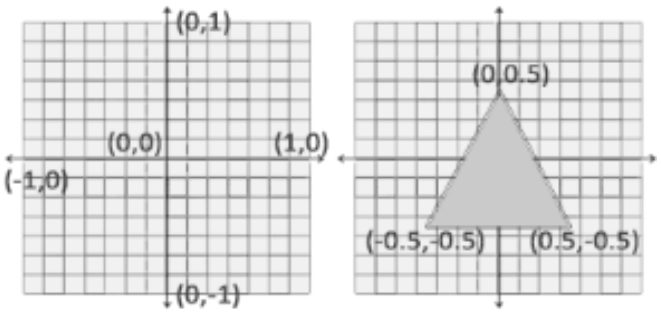

Рис. 1. Схематичне зображення реалізації NDC

Наприклад, якщо необхідно відобразити піраміду на екрані в верхньому правому куті графічного вікна,

то необхідно завантажити в відеопам’ять всі точки цієї фігури та вивести їх на екран. Для цих точок у нас в розпорядженні є лише тривимірний кубічний простір з центром в початку координат. Тобто, середина фігури матиме координати приблизно +0,9 по осі X та +0,9 по осі Y. Працювати з дробовими координатами не дуже зручно, немає можливості дивитись на фігуру з іншої грані.

Виникає потреба у використанні більш зручної системи координат для роботи з віртуальним простором. Після обробки всіх точок, дані трансформуються з NDC в координати екранного простору (screen-space coordinates), що відповідають роздільній здатності пристрою виведення або графічного вікна операційної системи. Було б зручно, якби ми відразу використовували таку розмірність, наприклад, від 0 до 800 замість від -1 до 1 по осі X.

Вирішення цієї проблеми передбачає використання матриці проекцій, як правило, ортогональну та перспективну проекції. Для розрахунку першої користуються формулою (1).

$$\begin{pmatrix} \frac{2}{right-left} & 0 & 0 & -\frac{right+left}{right-left} \\ 0 & \frac{2}{top-bottom} & 0 & -\frac{top+bottom}{top-bottom} \\ 0 & 0 & -\frac{2}{far-near} & -\frac{far+near}{far-near} \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (1)$$

В формулі (1) замість значень left, right, top, bottom, far та near необхідно підставити нові значення границь віртуального простору. В результаті буде отримано матрицю ортогональної проекції, її принцип роботи зображено на рис. 2.

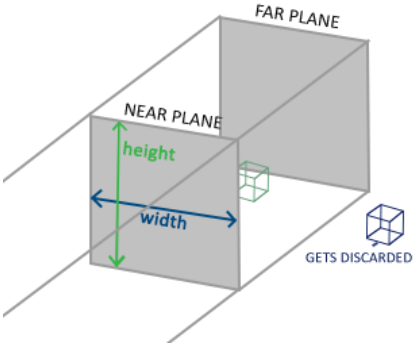

Рис 2. Принцип роботи ортогональної проекції

Після цього змінюють координати всіх точок фігури для відображення. Для відображення фігури необхідно розглянути координати точки як вектор, помножити цей вектор на матрицю. В результаті цієї операції отримаємо координати в форматі NDC, який підтримується графічним API.

Ортогональна матриця підходить лише для відображення 2Д об’єктів. При розміщенні в віртуальному просторі двох об’єктів, за умови що один з них буде далі від камери, при рендері такого зображення, вони будуть мати однаковий розмір. Для створення реалістичних 3D зображень необхідно реалізувати закони перспективи. Так як об’єкти здаються нам меншими при віддалені, то матриця перспективної проекції буде зменшувати відстань між точками, що знаходяться далеко від камери, створюючи ілюзію цієї самої перспективи. Різницю у відображенні об’єктів при різній проекції показано на рис. 3.

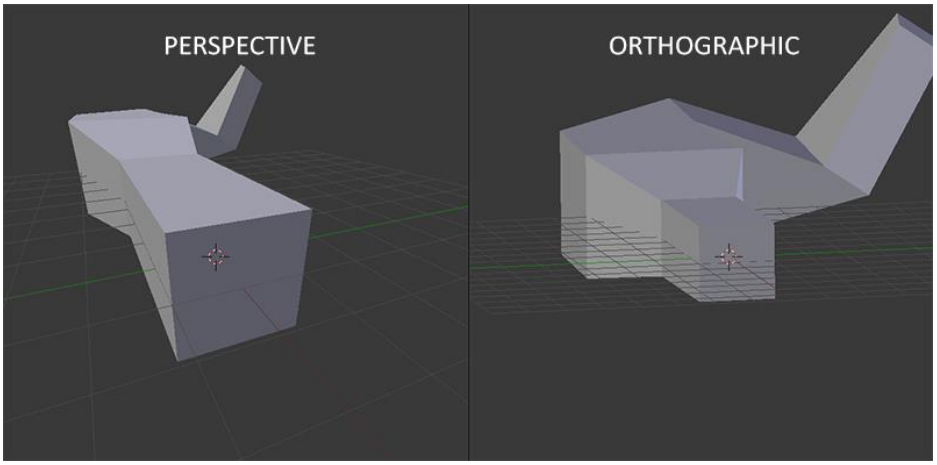

Рис. 3. Відображення двох однакових об’єктів в перспективній (зліва) та ортогональній (справа) проекціях

Розрахувати таку матрицю перспективної проекції можливо за допомогою формули (2).

$$\begin{pmatrix} \frac{1}{aspect*\tan\frac{fov}{2}} & 0 & 0 & 0 \\ 0 & \frac{1}{\tan\frac{fov}{2}} & 0 & 0 \\ 0 & 0 & -\frac{far+near}{far-near} & -\frac{2*far*near}{far-near} \\ 0 & 0 & -1 & 0 \end{pmatrix} \quad (2)$$

Значення fov, far, near та aspect задають та перемножують нові координати у вигляді вектора на матрицю. Принцип роботи зображено на рис. 4.

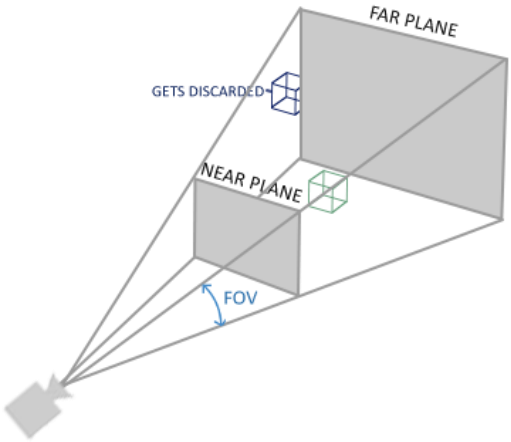

Рис. 4. Принцип роботи перспективної проекції

Різноманітні матриці можуть бути використані і для трансформації об’єктів, а саме: масштабування, переносу та повороту. Перша з них розраховується таким чином: $\begin{pmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$, де S_x, S_y та S_z – це

коефіцієнти зміни розміру по кожній осі.

Матриця перенесення має наступний вигляд: $\begin{pmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$, де T_x, T_y та T_z – це зсув по кожній осі.

Для повороту об'єкта використовують декілька різних матриць: для трансформації по окремим осям та об'єднану.

Поворот по осі X: $\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$.

Поворот по осі Y: $\begin{pmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$.

Поворот по осі Z: $\begin{pmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$.

Об'єднана матриця повороту:

$$\begin{pmatrix} \cos\theta + R_x^2(1-\cos\theta) & R_xR_y(1-\cos\theta) - R_z\sin\theta & R_xR_z(1-\cos\theta) - R_y\sin\theta & 0 \\ R_yR_z(1-\cos\theta) - R_z\sin\theta & \cos\theta + R_y^2(1-\cos\theta) & R_yR_z(1-\cos\theta) - R_x\sin\theta & 0 \\ R_zR_x(1-\cos\theta) - R_y\sin\theta & R_zR_y(1-\cos\theta) - R_x\sin\theta & \cos\theta + R_z^2(1-\cos\theta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

В усіх матрицях кут повороту задається через θ, а в об'єднаній матриці R_x, R_y та R_z визначають коефіцієнти повороту по відповідним осям.

Таким чином, лінійна алгебра дозволяє проводити будь-які трансформації над об'єктом в зручній системі координат.

Останнє на що ми маємо звернути увагу, це керування камерою. Неможливо щось зробити з віртуальним простором, наданим графічним API. Це ж саме відноситься і до грані куба, з якої відбувається спостереження. Тому, немає прямих інструментів керування віртуальною камерою. Для обходження цих правил використовуємо той же прийом, що і з перспективою – замість повороту камерою, повертаємо всі точки всередині віртуального простору. Тому, використання матриць трансформації у всіх точках віртуального простору імітуватиме переміщення та поворот камери.

Висновок

Таким чином, використання лінійної алгебри в комп'ютерній графіці є потужним інструментом для перетворення простих програмних API.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. OpenGL Wiki [Електронний ресурс] – Режим доступу до ресурсу: https://www.khronos.org/opengl/wiki/.
2. Learn OpenGL, extensive tutorial resource for learning Modern OpenGL [Електронний ресурс] – Режим доступу до ресурсу: https://learnopengl.com/.

Миргородський Андрій Вікторович – студент групи 3ПІ-18б, факультет інформаційних технологій та комп'ютерної інженерії, Вінницький національний технічний університет, м. Вінниця.

Воронін Євген Сергійович - студент групи 3ПІ-18б, факультет інформаційних технологій та комп'ютерної інженерії, Вінницький національний технічний університет, м. Вінниця.

Науковий керівник: ***Прозор Олена Петрівна*** – доцент кафедри вищої математики, Вінницький національний технічний університет, м. Вінниця.

Myrhorodskyi Andrii V. – student of 3PI-18b, Faculty for Information Technologies and Computer Engineering, Vinnytsia National Technical University, Vinnytsia.

Voronin Yevhen S. - student of 3PI-18b, Faculty for Information Technologies and Computer Engineering, Vinnytsia National Technical University, Vinnytsia.

Supervisor: ***Prozor Olena P.*** – Cn. Sc. (Eng), Assistant Professor of the Department of Higher Mathematics, Vinnytsia National Technical University, Vinnytsia, email: el.przr@gmail.com.