Generating graphic visualizations using programming language tools with direct access to video memory

Oleksandr Zhyrytovskyi, Roman Zubko
Open International University of Human Development "Ukraine"
Kyiv, Ukraine
i.am.zhirik@gmail.com, rzubko@ukr.net

Abstract. Previously, programming languages such as BASIC and PASCAL allowed direct access to video memory. This made it possible to experiment with various visualizations on the computer screen. Today, processors are much faster, and video adapters can display images at higher resolutions. However, due to the lack of direct access to video memory in modern programming languages, algorithms for working with graphics have not seen significant development. The purpose of this article is to demonstrate the types of visualizations that can be created with direct access to video memory.

***Keywords*: computer graphics, visualizations, fractals, 3d rendering, texture mapping, mesh, triangualtion.**

Формування графічних візуалізацій за допомогою засобів мови програмування з прямим доступом до відеопам'яті

Жиритовський Олександр, Зубко Роман
Відкритий міжнародний університет розвитку людини «Україна»
Київ, Україна
i.am.zhirik@gmail.com, rzubko@ukr.net

Анотація. Раніше мови програмування такі, як BASIC чи PASCAL, мали можливість прямого доступу до відеопам'яті. Як наслідок, можна було експериментувати з відображенням різноманітних візуалізацій на екрані комп'ютера. На сьогодні процесори стали набагато швидшими і відеоадаптери здатні відтворювати зображення з більшою роздільною здатністю. Проте, за відсутності засобів прямого доступу до відеопам'яті у сучасних мовах програмування, алгоритми для роботи з графікою не зазнали значного розвитку. Метою даної статті є показ візуалізацій, які можуть бути розроблені за наявності прямого доступу до відеопам'яті.

Ключові слова: комп'ютерна графіка, візуалізації, фрактали, 3d рендеринг, накладання текстури, каркасна сітка, триангуляція.

Вступ

Формування графічної візуалізації передбачає задання певного кольору кожному пікселю на екрані. Маючи мову програмування з можливістю прямого доступу до відеопам'яті, можна створювати візуалізації на екрані комп'ютера у реальному часі.

Фрактали

Фрактал — це геометрична фігура або структура, яка має властивість самоподібності, тобто повторюється в зменшеному масштабі на різних рівнях. Це означає, що якщо збільшити якусь частину фракталу, вона виглядатиме подібно до цілого. Розглянемо один із фракталів, який називається множина Мандельброта [1]. В основі хаотичності даного фракталу лежить послідовність

$$Z_i = Z_{i-1}^2 + C \qquad (1)$$

При умові, що початкове значення Z_0 та константа C лежать в межах від -2 до 2, ця послідовність починає поводити себе дуже хаотично. А саме, вона спочатку коливається на цьому проміжку певний час, а уже потім наближається до 0 чи прямує до нескінченності. Кількість ітерацій, за яку відбувається вихід за межі діапазону від -2 до 2, можна відобразити у вигляді градацій яскравості кольору. Для створення двовимірної картинки, замість цілих чисел, потрібно використати комплексні числа. Приклад множини Мандельброта наведено на рис. 1.

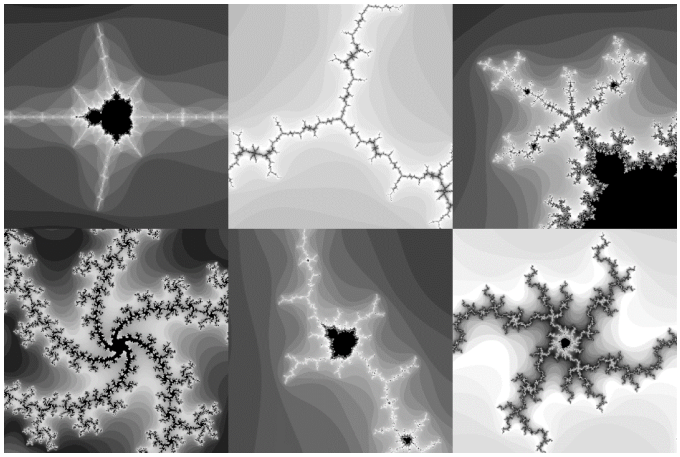

Рис. 1. Множина Мандельброта

Множина Жюліа будується так само, як і множина Мандельброта. Відмінністю є тільки те, що константа *C* у ній задається не поточною координатою точки, а певним сталим значенням. Приклад множини Жюліа [2] наведено на рис. 2.

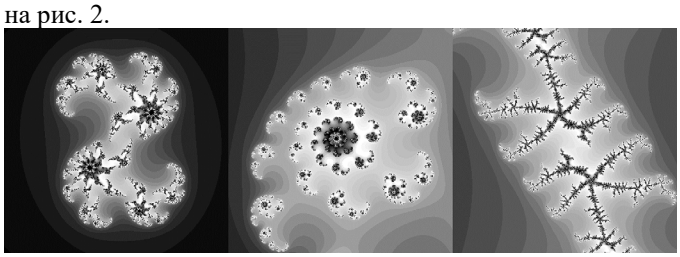

Рис. 2. Множина Жюліа

Фрактальні дерева

В основу побудови фрактальних дерев покладена функція побудови гілки дерева. Ця функція приймає координати *x* та *y* початку гілки, кут нахилу гілки та рівень гілки. Стовбур має перший рівень, довгі гілки мають другий рівень, коротші гілки – третій рівень і так далі. Знаючи координати та кут відносно цих координат, можна намалювати нову гілку уже вищого рівня під потрібним кутом відносно поточного. Кожна з гілок дерева будується подібним чином. Приклад етапів побудови фрактального дерева зображено на рис. 3. Кожен наступний етап тут доповнюється наступним рівнем гілок.

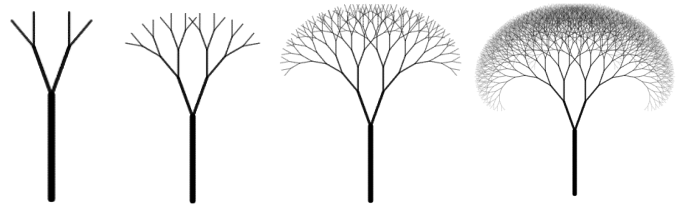

Рис. 3. Етапи побудови фрактального дерева

Як результат, можна побудувати фрактальне дерево з різними довжинами гілок, що нахилені під різними кутами та використовуючи дві або три криві лінії. Криві лінії тут описуються кубічними сплайнами. Кожна наступна гілка є сплайном, що будується як продовження попереднього сплайну. Приклади таких дерев зображені на рис. 4.

Рис. 4. Фрактальні дерева

Для створення більшої реалістичності доцільно намалювати дерево з більш товстим стовбуром. Для цього можна використати сплайни, які мають різну товщину ліній на їх кінцях. Приклади зображень таких дерев з товстим стовбуром побудованих на основі даного підходу наведено на рис. 5.

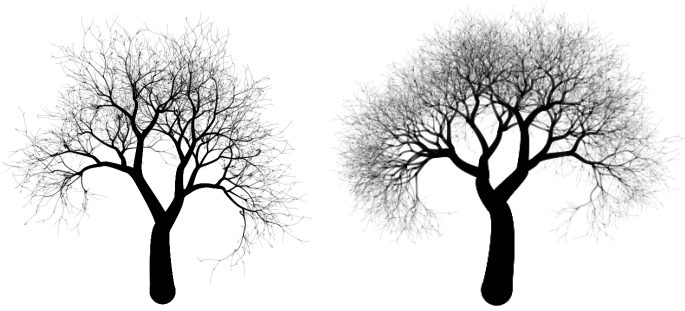

Рис. 5. Фрактальні дерева побудовані на основі кривих змінної товщини

Ще одним типом фрактальних дерев є Н-дерево. Воно має таку назву, оскільки схоже на літеру «Н». Його дві гілки можна уявити у вигляді двох годинникових стрілок. На кінцях кожної з цих двох годинникових стрілок знаходяться ще дві годинникові стрілки, повернуті на такі самі кути і так далі. В залежності від того під якими кутами знаходяться гілки будуть отримані різноманітні фігури. Приклади фігур, що базуються на обертанні гілок Н-дерева зображено на рис. 6.

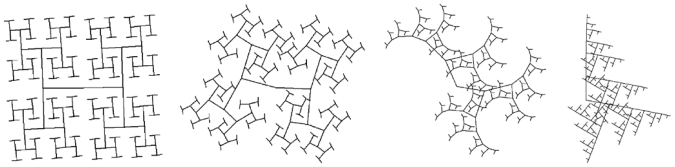

Рис. 6. Фігури побудовані на основі обертання гілок Н-дерева

Хаотичні функції

Особливістю хаотичних функцій є те, що кожне наступне значення функції рахується на основі попереднього. В основі хаотичного розподілу значень таких функцій лежить формула кола, але змінна *y* тут рахується за допомогою її попереднього значення:

$$\begin{aligned} x_i &= \cos(\alpha) \\ y_i &= \sin(\alpha + y_{i-1} \cdot t) \end{aligned} \qquad (2)$$

де:

α – кут, що змінюється від 0 до 2π;

t – певна константа, яка за необхідності може змінюватися у часі;

x, y – координати кожної точки графіка функції, що відображається на екрані.

Функцію за такою формулою зображено на рис.7.

Рис. 7. Хаотична функція за формулою (2)

На рис. 8 наведено приклад хаотичної функції, що будується за формулою:

$$x = \cos\left(\alpha + 2 \cdot \sqrt{0.1 + (x+t)^2 + (y + t/(x+y+t))^2}\right)$$
$$y = \sin\left(\alpha + 2 \cdot \sqrt{0.1 + (x+t)^2 + (y+t)^2}\right) \quad (3)$$

де:

α – кут, що змінюється від 0 до 2π;

t – певна константа, яка за необхідності може змінюватися у часі;

x, y – координати кожної точки графіка функції, що відображається на екрані.

Рис. 8. Хаотична функція за формулою (3)

Фігури Ліссажу

В основі побудови фігур Ліссажу [3] лежить формула кола, у яку додані два коефіцієнти. Вона має наступний вигляд:

$$x = \cos(k_1 \alpha)$$
$$y = \sin(k_2 \alpha) \quad (4)$$

де:

α – кут, що змінюється від 0 до 2π;

k_1, k_2 – коефіцієнти, що задають зображення функції;

x, y – координати кожної точки графіка функції, що відображається на екрані.

Комбінації двох коефіцієнтів даватимуть різні фігури. Приклади таких фігур зображено на рис. 9.

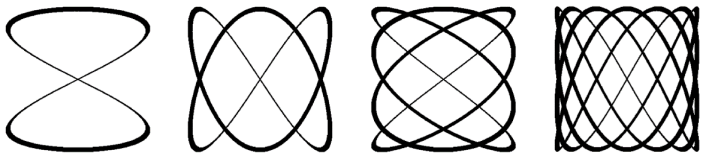

Рис. 9. Фігури Ліссажу

Троянди Гвідо Гранді

Троянди Гвідо Гранді, так само як і фігури Ліссажу, будуються за двома коефіцієнтами, але їх формула інша і має наступний вигляд:

$$x = \cos(k_1 \alpha) \cdot \cos(k_2 \alpha)$$
$$y = \sin(k_2 \alpha) \cdot \cos(k_1 \alpha) \quad (5)$$

де:

α – кут, що змінюється від 0 до 2π;

k_1, k_2 – коефіцієнти, що задають зображення функції;

x, y – координати кожної точки графіка функції, що відображається на екрані.

Приклади троянд Гвідо Гранді зображено на рис. 10.

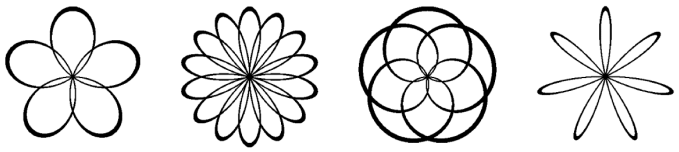

Рис. 10. Троянди Гвідо Гранді

Генерація лабіринтів

Для генерації лабіринту спочатку будується рамка, що є чотирма стінами по краям. В середині цієї рамки кожна нова стінка малюється починаючи від довільної існуючої стінки таким чином, щоб її кінець ні до якої іншої стінки не примикав. Результат побудованого таким способом лабіринту зображено на рис. 11.

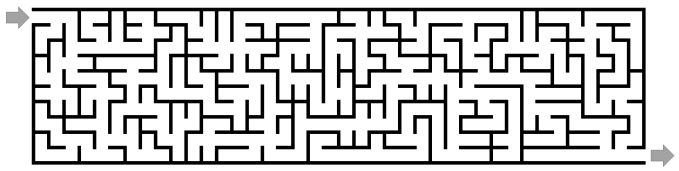

Рис. 11. Лабіринт

Візуалізація руху хвиль на воді

В даній моделі кожна точка екрану має колір, що характеризує висоту хвилі в цій точці. Оскільки хвилі накладаються, то висота хвилі в точці є сумою висот хвиль які її перетинають (рис. 12).

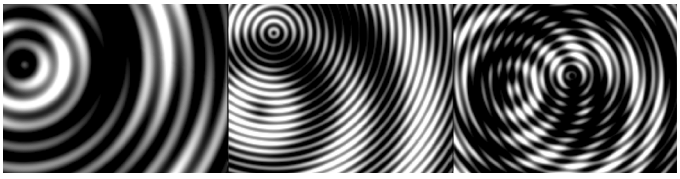

Рис. 12. Візуалізація руху хвиль на воді

Побудова тривимірних фігур на основі точок

В основі побудови фігур у тривимірному просторі лежить формула тривимірного повороту, яка має вигляд:

$$x' = x_c + x\cos\alpha - y\sin\alpha$$
$$y' = y_c + (x\sin\alpha + y\cos\alpha)\sin\beta + z\cos\beta \quad (6)$$

де:

x, y, z – координати точки у тривимірному просторі;

α, β – кути повороту тривимірної площини;

x_c, y_c – координати центру екрану;

x', y' – двовимірні координати точки на екрані.

Використовуючи формулу (6) можна ставити точку на екрані у довільних координатах. Як приклад на рис. 13 намальовано тривимірні зображення квіток жоржини та троянди.

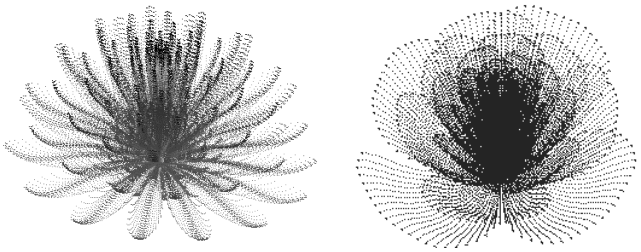

Рис. 13. Зображення квіток жоржини та троянди побудованих за допомогою малювання точок у тривимірному просторі

Побудова тривимірної фігури за допомогою каркасної сітки

В основі побудови тривимірної фігури є сітка з точок. Для побудови каркасу фігури сусідні точки у ній з'єднуються прямими лініями. Для побудови суцільної фігури кожна клітинка сітки заміщується двома трикутниками. Результати таких перетворень зображено на рис. 14.

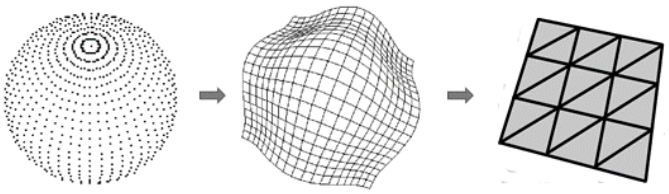

Рис. 14. Побудова тривимірної фігури за допомогою каркасної сітки

Накладання текстури на грань

Кожна трикутна грань фігури, у найпростішому випадку, може мати суцільний колір. Більш складним варіантом є плавне переливання трьох кольорів між вершинами трикутника. Накладання текстури на трикутну грань виконується повністю ідентично за тими самим формулами, як і переливання трьох кольорів. Тільки замість плавної зміни кольорів відбувається плавна зміна координат точок зображення текстури (рис. 15).

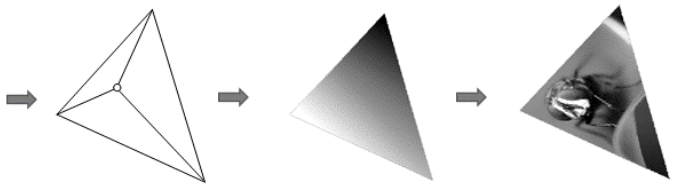

Рис. 15. Накладання текстури на грань

Накладання текстури на довільну форму

Якщо побудувати сітку з трикутників, що має певну форму, то для кожного трикутника можна окремо вказати координати текстури. Як наслідок, зображення текстури можна накласти на довільну форму [4] (рис. 16).

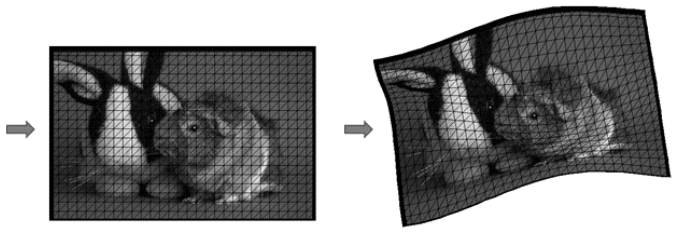

Рис. 16. Накладання текстури на довільну форму

Системи часток

Система часток [5] передбачає наявність великої кількості часток, які поводять себе за певними законами. Частка представляє собою певну нечітку фігуру, що має визначений колір, розмір, поточні координати, напрямок руху і швидкість. Також частка має фіксовану тривалість життя, після чого може просто зникнути, або перетворитися на набір інших часток. Для симуляції вибухів феєрверків будемо вважати, що є певна частка (ракета), яка у процесі польоту створює за собою слід із часток. У певний момент ракета зникає перетворюючись на набір часток у вигляді іскор, які розлітаються у різні сторони. Приклад симуляції вибухів феєрверків, використовуючи систему часток, зображено на рис. 17.

Рис.17. Симуляція вибухів феєрверків

Імітація вогню

В основі імітації вогню є безперервний цикл малювання помаранчевих крапок внизу екрана та розмиття. Розмиття відбувається присвоєнням точці кінцевого зображення середнього значення сусідніх точок початкового зображення. Що правда у такому випадку розмір полум'я буде лише у декілька пікселів. Для того щоб підняти полум'я угору, можна задати певні коефіцієнти для розмиття. Тоді замість формули середнього арифметичного (7)

$$\frac{a+b+c}{3} \quad (7)$$

використовується формула середнього зваженого (8).

$$\frac{1\cdot a+2\cdot b+3\cdot c}{1+2+3} \tag{8}$$

Приклад коефіцієнтів та зображення полум'я наведено на рис. 18.

$$\begin{pmatrix} 10 & 1 & 10 \\ 1 & 1 & 1 \\ 1 & 100 & 1 \end{pmatrix}$$

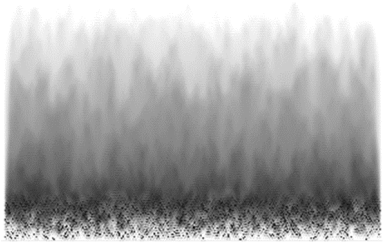

Рис. 18. Коефіцієнти розмиття та зображення імітованого вогню

Псевдо-тривимірні деформаційні візуалізації

Ідея створення псевдо-тривимірних деформаційних візуалізацій [6] полягає у тому, що кожній точці кінцевого зображення ставиться у відповідність певна точка початкового зображення. Потім кінцеве зображення стає початковим і цей цикл повторюється. Координати точки кінцевого зображення можуть бути дійсними числами. Тому можна використати техніку білінійного згладжування. А саме, для цього у кожної з координат треба відокремити цілу та дійсну частини чисел. Після чого потрібно взяти фрагмент початкового зображення розміром 2x2 пікселі з цілими координатами. Потім змішати ці чотири кольори пікселів в один у пропорції, що задаватиметься дійсними частинами чисел поточних координат. Циклічне малювання та зміщення точок у нові координати, разом з білінійним згладжуванням, може сформувати візуалізацію галактики (рис. 19).

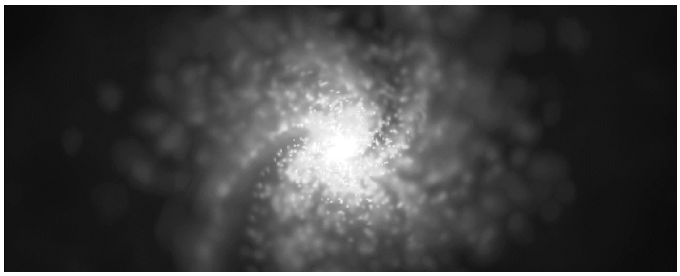

Рис. 19. Візуалізація галактики

Графічне відображення сніжинок

Сніжинка генерується як шість променів, рівномірно розташованих під кутом 60°. Кожен промінь має чотири відгалуження з випадковою довжиною, що розміщені під таким самим кутом. Саме ці відгалуження визначають унікальну форму сніжинки. Приклад сніжинок, побудованих за таким алгоритмом, наведено на рис. 20.

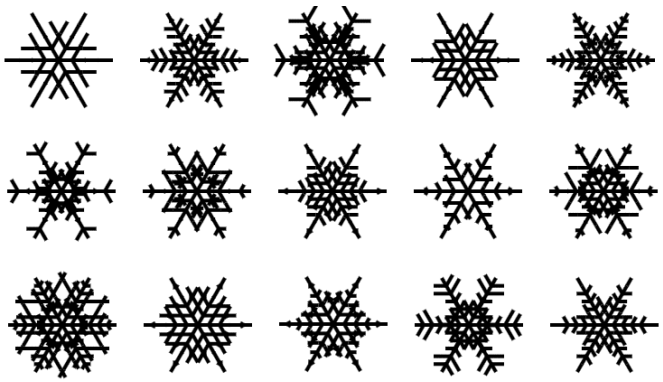

Рис. 20. Графічне відображення сніжинок

Висновки

Візуалізації у даній статті були згенеровані за допомогою прямого доступу до відеопам'яті. Можна зробити висновок, що написання алгоритмів, які виконуються на процесорі комп'ютера з прямим доступом до відеопам'яті є потужним механізмом для формування різноманітних візуалізацій на екрані. Для подальшого розвитку алгоритмів побудови візуалізацій актуальним є наявність у мовах програмування вбудованих засобів для прямого доступу до відеопам'яті.

Література

1. Mandelbrot B.B. The Fractal Geometry of Nature / B.B. Mandelbrot. – San Francisco: W.H. Freeman and Company, 1982. – 468 p.
2. Barnsley, M. F. Fractals Everywhere. – London : Academic Press Inc., 1988. – 396 p. – P. 269.
3. Greenslade, T. B. Jr. Adventures with Lissajous Figures / T. B. Jr. Greenslade. – San Rafael, CA : Morgan & Claypool Publishers, 2018. – 89 p.
4. Botsch, M., Kobbelt, L., Pauly, M., Alliez, P., & Lévy, B. Polygon Mesh Processing. – Natick, Massachusetts: AK Peters, 2010. – 364 p.
5. https://shawnhargreaves.com/egg/ (дата звернення 05.05.2025).
6. https://www.geisswerks.com/geiss/ (дата звернення 05.05.2025)