УДК 004.932

Заречанський Я. О., ст. ІІІ курсу факультету кібернетики; науковий керівник – к.техн.н, доцент Шпортько О. В. (Міжнародний економіко-гуманітарний університет імені академіка Степана Дем'янчука, м. Рівне)

РЕАЛІЗАЦІЯ АЛГОРИТМУ ГЕНЕРАЦІЇ ФРАКТАЛЬНИХ ДЕРЕВ МОВОЮ ПРОГРАМУВАННЯ C#

Анотація. *У статті розкрито особливості реалізації алгоритму генерації фрактальних дерев мовою програмування C# в середовищі MS Visual Studio. Доведено, що формування фрактальних дерев найкраще виконувати рекурсивно, зміщуючи початок координат та кут нахилу координатних осей. Наочно наведено фрагменти програмного коду, проміжні та остаточні результати виконання етапів розробленої програми. Обґрунтовано переваги графічної обробки зображень мовою програмування C#.*
Ключові слова: *фрактальне дерево, рекурсивні виклики методів.*

Abstract. *The article describes the features of the implementation of the algorithm for the generation of fractal trees in the C# programming language in MS Visual Studio. Formation of fractal trees is suggested to be performed recursively, shifting the origin to the angle of inclination of the coordinate axes. The code snippets and intermediate and final results of the successive stages of the developed program are given. The benefits of graphic image processing in C# programming language are justified.*
Keywords: *fractal tree, recursive method calls.*

Останнім часом неабиякої популярності набули фрактальні зображення, які вирізняються серед інших, з одного боку, незвичною красою, а з іншого – відносною простотою побудови засобами обчислювальної техніки. Як відомо, фрактал – це нескінченно самоподібна геометрична фігура, кожен фрагмент якої повторюється при зменшенні масштабу [1]. Такий об'єкт називають самоподібним, коли збільшені його частини схожі на сам об'єкт і одна до одної [2]. Фрактали дають змогу швидко формувати складні зображення засобами векторної комп'ютерної графіки, тому реалізація оптимізованих алгоритмів генерації фрактальних зображень засобами різних мов програмування є нині та залишатиметься в найближчому майбутньому актуальним завданням.

Слово «фрактал» у перекладі з латинської означає «складається з фрагментів». Воно було запропоноване Бенуа Мандельбротом у 1975 році [3] для позначення нерегулярних, слабкоподібних структур, якими він займався, хоча у його роботах були використані наукові результати інших вчених,

які працювали у 1875–1925 роках (Анрі Пуанкаре, Фату П'єр, Жюліа Гастон Моріс, Георг Кантор, Фелікс Гаусдорф) [2]. Але тільки в наш час результати їх роботи вдалося об'єднати в єдину систему [4]. Фрактальними властивостями володіють, наприклад, узбережжя, хмари, крони дерев, сніжинки [1]. Найвідомішими фрактальними об'єктами, що генеруються засобами обчислювальної техніки, є дерева (рис. 1): від кожної гілки відходять менші, схожі на неї, від них – ще менші, але теж подібні і т. д. За окремою гілкою математичними методами можна відслідкувати властивості всього дерева.

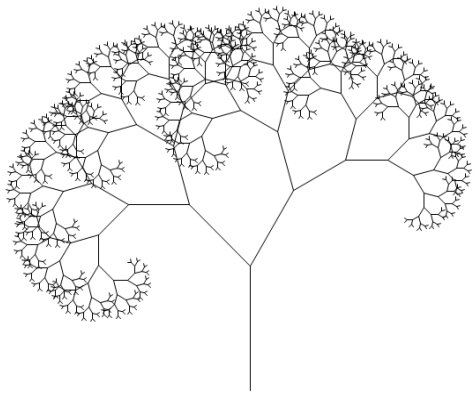

Рис. 1. Приклад фрактального дерева

Взагалі, описати подібні об'єкти фракталів можна всього лише декількома однаковими математичними рівняннями [2], і саме це дає змогу швидко генерувати фрактальні зображення за допомогою комп'ютерних технологій. Приклад реалізації саме такого підходу наведено у цьому дослідженні.

Метою нашої статті є дослідження та обґрунтування способу реалізації алгоритму генерації фрактального дерева з використанням рекурсивних викликів [5] засобами мови C# для виводу його на форму програми середовища розробки програмного забезпечення Microsoft Visual Studio.

Опис основних кроків пропонованої реалізації алгоритму генерації фрактального дерева [5]. Виведення початку стовбур дерева на площину форми. Оскільки початок координат на екрані по замовчуванню починається з верхнього лівого кутка, то його необхідно перемістити поряд з центром нижнього краю форми (рис. 1). Сам стовбець стовбура потрібно зобразити прямою лінією на екрані. Одна з вершин цієї ліні знаходиться в перенесеному початку координат, а друга – на 200 пікселів вище від неї (змінна довжини *len* ініціалізується значенням 200). Після цього початок координат переноситься до другої точки. Далі генеруються «гілки дерева» відносно перене-

сених початків координат, використовуючи два рекурсивні виклики функції генерації фрагменту зображення, подібного до стовбура, але з меншою довжиною та іншими кутами нахилу координатних осей. Ці виклики необхідно виконувати, поки довжина поточної гілки перевищує два пікселя (*len* > 2).

Наведемо рекурсивний алгоритм генерації «гілок» детальніше. Для реалізації першої половини цього алгоритму, за умови, що довжина лінії поточної гілки перевищує два пікселя, потрібно послідовно виконати такі дії:

1. Вивести лінію поточної «гілки» дерева з параметрами, заданими на попередній ітерації (для першої ітерації це «стовбур дерева»);

2. Перенести початок координат до другої точки виведеної лінії;

3. Запам'ятати поточні параметри координатних осей;

4. Перетворити поточну довжину лінії «гілки» в проміжну змінну (*prom* = *len*);

5. Повернути координатні осі для наступної ітерації на 45° *вправо;*

6. Зменшити довжину лінії «гілки» для наступної ітерації на 33 % відносно попередньої ітерації (*len* *= 0,67);

7. Рекурсивно викликати функцію виводу правого відгалуження «гілки» зі зменшеною довжиною.

Коли довжина поточної «гілки» стане не більшою двох пікселів (*len* <= 2), то, керуючись алгоритмом, необхідно повернутися до попередньої ітерації та виконати дії його другої половини:

1. Відновити збережені перед рекурсивним викликом параметри кординатних осей;

2. Відновити збережену довжину лінії «гілки» (*len* = *prom*);

3. Повернути координатні осі на 45° *вліво;*

4. Зменшити довжину лінії «гілки» на 33 % (*len* *= 0,67);

5. Рекурсивно викликати функцію виводу лівого відгалуження «гілки» зі зменшеною довжиною;

6. Повторно відновити збережені параметри координатних осей та довжини лінії;

7. Завершити поточний рекурсивний виклик.

Після виконання цих рекурсивних дій на формі програми появиться симетричне фрактальне дерево.

Практична реалізація алгоритму генерації фрактального дерева. Запропоновані кроки алгоритму необхідно реалізувати засобами мови програмування C# в середовищі розробки програмного забезпечення Microsoft Visual Studio. Для початку потрібно створити рішення з шаблону *Application Windows Forms* і в автоматично сформованій формі згенерувати обробник події *Paint*. Дальше необхідно перейти до коду форми і для початку створити глобальні змінні, які знадобляться в програмі:

```
Pen p = new Pen(Color.FromArgb (255, 255, 255, 255), 2);
int len = 200,
```

де *p* – характеристики лінії для виводу стовбура та «гілок» (в нашому випадку колір лінії білий, а її товщина – 2 пікселя), *len* – довжина стовбура в пікселях. Глобальні змінні описуються в середині класу форми, оскільки в програмі можуть одночасно генеруватися декілька дерев.

Дальше у функції реакції на подію *Paint* необхідно здійснити зміщення початку координат поряд з центром нижнього краю форми і вивести першу лінію-стовбур, яка в подальшому буде також рекурсивно виводити «гілки»:

```
private void Form1_Paint(object sender, PaintEventArgs e)
{if (len == 200) // переміщення початку координат при виводі стовбура
e.Graphics.TranslateTransform ((int) (Width/ 2), Height – 50);
e.Graphics.DrawLine (p, 0, 0, 0, -len) // вивід лінії
// запам’ятовуємо поточне положення координатних осей
GraphicsState transState = e.Graphics.Save();
```

Рядок методу *DrawLine* цього коду виводить лінію між точками з координатами (0; 0) та (0; -*len*) з заданими раніше характеристиками відносно зміщеного початку координат.

Тепер реалізуємо першу половину рекурсивного алгоритму:

```
// переміщуємо початок координат до кінця виведеної лінії
e.Graphics.TranslateTransform (0, -len);
if (len > 2) // потрібне задання параметрів для «гілок»
{int prom = len; // зберігаємо поточну довжину лінії «гілки»
e.Graphics.RotateTransform(45); // повертаємо осі вправо
en = (int)(len * 0,67); // зменшуємо довжину лінії
// рекурсивно викликаємо цю ж функцію для виводу «гілок» справа
Form1_Paint (sender, e);
```

Формуючи зображення, потрібно уважно слідкувати за змінними, які діляться або множаться на дійсні числа, адже вивід у формі виконується лише по цілочисельних координатах пікселів. При виконані лише наведеної, першої половини реалізації алгоритму отримаємо лінії, які з’єднані між собою і послідовно відхиляються вправо (рис. 2).

Рис. 2. Результат виконання програми з реалізованою першою половиною рекурсивного алгоритму

В наведеному коді немає відновлення збережених перед рекурсивним викликом параметрів координатних осей. Справа в тому, що для виводу ліній, що послідовно відхиляються вправо, таке відновлення не потрібне. Але в процесі реалізації всього алгоритму можна використати відновлення положення координатних осей для забезпечення коректного виводу усіх розгалужень фрактального дерева:

```
// відновлюємо положення осей та довжину лінії після виводу справа
e.Graphics.Restore(transState);
len = prom;
e.Graphics.RotateTransform (–45); //повертаємо осі вліво
len = (int) (len * 0.67); //зменшуємо довжину лінії
// рекурсивно викликаємо цю ж функцію для виводу «гілок» зліва
Form1_Paint(sender, e);
e.Graphics.Restore (transState); // знову відновлюємо положення осей
len = prom; //відновлюємо довжину лінії
}} // завершення рекурсії та процедури
```

Змінну класу *GraphicsState* можна створити після відкриття простору імен *Drawing2D*, вписавши на початку файла коду форми

```
using System.Drawing.Drawing2D.
```

Фрактальне дерево, що отримується в результаті виконання повної наведеної реалізації описаного алгоритму, зображене на рис. 3. Інші варіанти рекурсивних дерев можна отримати, змінюючи кути нахилу «гілок» та їх довжини.

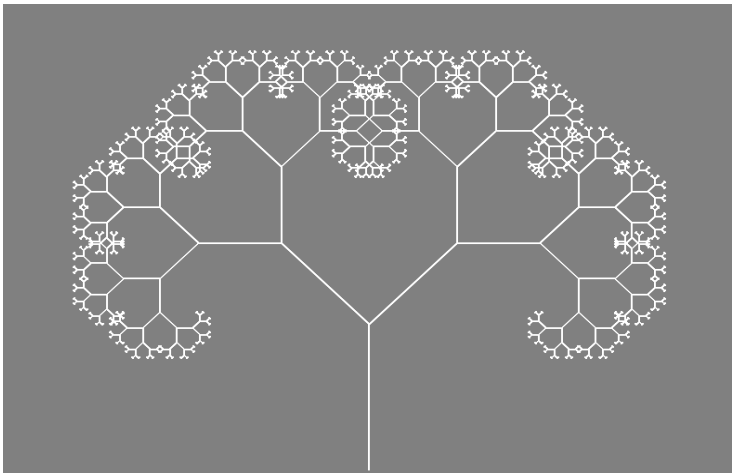

Рис. 3. Фрактальне дерево, отримане за результатами виконання наведеного програмного коду

З наведених результатів дослідження можна зробити такі висновки:

1. Для генерування фрактальних об'єктів доцільно використовувати рекурсивні виклики методів, зменшуючи щоразу висоту окремого фрагмента до досягнення граничного значення;

2. Реалізуючи повороти фрагментів графічних об'єктів, варто не розраховувати їх положення самостійно, а використати стандартні методи повороту осей та зміщення початку координат обраної мови програмування;

3. Середовище MS Visual Studio має в своєму арсеналі потужні сучасні засоби обробки графічних зображень, що реалізовані мовою програмування C#, які варто використовувати в практиці програмування.

1. Геометрія в природі: фрактали. URL: http://juliagalant.blogspot.com/2012/09/blog-post_30.html (дата звернення: 14.10.2019). **2.** Фрактальна графіка. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Фрактальна_графіка (дата звернення: 14.10.2019). **3.** Фрактал. Вікіпедія. URL:https://uk.wikipedia.org/wiki/Фрактал (дата звернення: 14.10.2019). **4.** Гринченко В. Т., Мацыпура В. Т., Снарский А. А. Фракталы: от удивления к рабочему инструменту. Київ, 2013. 270 с. **5.** Coding Challenge #14: Fractal Trees – Recursive. URL: https://www.youtube.com/watch?v=0jjeOYMjmDU&t=830s (дата звернення: 14.10.2019).