УДК 519.688

О.Б. Новікова, аспірант,
Національний авіаційний університет, м. Київ, Україна
novikovaolga88@gmail.com

Lazy Computations як механізм підвищення ефективності фрактальної інтерполяції

Розглянуто використання принципів «лінивих обчислень» для графічного представлення результатів фрактальної інтерполяції. В якості базових функцій обрано фрактальні сплайни, оскільки вони мають легкий та ефективний алгоритм реалізації та адекватно описують фрактальні моделі. Наведено ключові моменти при розробці програмного забезпечення для вирішення поставленої задачі. Результати тестування свідчать, що використання «лінивих обчислень» дозволяє зменшити затримку кадрів при виведенні інформації на екран.

Ключові слова: фрактальна інтерполяція, фрактальний сплайн, графічне представлення даних, lazy computations.

Вступ

Задачі фрактальної інтерполяції даних виникають при вирішенні сучасних фізико-технічних задач (графік температури полум'я, енцефалограми, сейсмограми) та інших розрахунків (коливання валютних курсів, математичні моделі границь природних об'єктів). Математичне розв'язання таких задач тісно пов'язано з комп'ютерною графікою, тому що сучасні технічні системи реалізуються з використанням комп'ютерів та інформаційних технологій. Питання оперативного відображення інтерпольованої графічної інформації на екранах моніторів та її компактного зберігання в пам'яті комп'ютера є наразі досить актуальним.

Усі області застосування інтерполяції пов'язані з обробкою все більших обсягів інформації. В той же час, у реалізації інтерпольованих функцій під час побудови зображень у комп'ютерній графіці з урахуванням сучасних вимог до зменшення затримки оброблюваної інформації при виведенні на екран і до підвищення частоти зміни кадрів, розмірів і складності зображень виникають проблеми, пов'язані з необхідністю істотного зменшення обчислювальної трудомісткості використовуваних алгоритмів інтерполяції. Тому, постійно шукаються нові, більш компактні в реалізації, що вимагають менших обчислювальних ресурсів, і просто більш легкі для освоєння методи й способи відновлення та інтерполяції [3, с. 5].

Дана робота присвячена пошуку шляхів підвищення ефективності обчислень для графічного виведення результатів інтерполяції фрактальних моделей.

Аналіз останніх досліджень і публікацій

Алгоритми машинної графіки, що реалізують задачу візуалізації, є традиційною темою досліджень з питань оптимізації та обробки інформації. Спершу алгоритми оптимізації стосувалися конкретних задач графічного виведення, наприклад, алгоритм Коена-Сазерленда (відсікання відрізків), Брезенхема (представлення прямої на дисплеї) тощо. Сьогодні набуває поширення підхід до оптимізації всього коду програми, коли ключовим моментом становиться *необхідність* тих чи інших операцій. Технологія Lazy Computations для графічних додатків спершу знайшла застосування при маніпуляціях з фотографіями з використанням вейвлетів (роботи Л.Велхо та К.Перліна), у точних геометричних розрахунках (С.Пійон, [2]). Сфера фрактальних моделей досі не була охоплена застосуванням «лінивих обчислень», і дана робота присвячена саме цій темі.

Об'єктом дослідження є графічне представлення результатів фрактальної інтерполяції. **Предметом дослідження** є використання «лінивих обчислень» у алгоритмі фрактальної інтерполяції та графічного виведення її результатів.

Цілі дослідження:

1) дослідити загальні принципи технології «лінивих обчислень» та можливість їх застосування до фрактальних моделей;

2) виявити конкретні прийоми та методи, що дозволяють більш ефективно використовувати системні ресурси при розв'язанні задач фрактальної інтерполяції;

3) перевірити теоретичні положення на практиці шляхом тестування розробленого програмного продукту та проаналізувати отримані результати.

Фрактальні сплайни

Для вирішення задачі фрактальної інтерполяції пропонується використовувати функції, які мають такі властивості:

1. *Однозначність* забезпечує стабільне обчислення алгоритмів декомпозиції та реконструкції.
2. *Регулярність* (неперервність перших похідних) забезпечує гладкість функції та кращу частоту локалізації. Розриви похідних впливають на якість наближення та стиснення зображень.
3. *Симетричність* дозволяє легше оперувати з граничними умовами і дає кращі результати сприйняття відновлених зображень.
4. *Просте аналітичне вираження*.

Фрактальні сплайни мають всі властивості, перераховані вище: однозначність, регулярність, гладкість, симетричність, локальність, просте аналітичне вираження та ефективність у реалізації.

В роботі розглянуто сімейство фрактальних сплайнів, що базуються на кубічних ермітових сплайнах.

Фрактальний сплайн – це функція, що складається з сплайн-функцій різного масштабу, що зберігають самоподібність. Як і звичайний сплайн, фрактальний сплайн характеризується степенем, кількістю вузлів, крайовими умовами. Від фракталу він перейняв такі характеристики, як кількість масштабів і фрактальна розмірність.

Масштабом будемо називати кількість вкладених рівнів самоподібних сплайнів.

Для того, аби сплайн став фракталом, необхідно, щоб кожен із R фрагментів також був сплайном, подібним до оригінального. Тоді сплайн на k-му масштабі буде складатися з R^k фрагментів. Поділ кожного фрагменту сплайна зберігає пропорцію нульового масштабу (рис. 1). У класичному розумінні головна властивість фракталу – самоподібність. Усі його частини подібні одна одній та фракталу в цілому на різних масштабах. Пізніше було запропоновано узагальнену версію самоподібності – самоафінність. Самоафінним називають фрактал, для якого при зміні масштабу стиснення чи розтягнення відбувається не в однаковому відношенні, як у самоподібного фракталу, а по-різному в різних напрямках, тобто виконується умова [4, с. 10]

$$f(t) = \xi^{-1/2} f(\xi \cdot t), \qquad (1)$$

де ξ>0- довільний коефіцієнт.

Оскільки коефіцієнти фрактальних сплайнів на різних масштабах можуть бути неоднаковими, то для описання їх ієрархічної структури будемо користуватися пірамідою фрактальних коефіцієнтів.

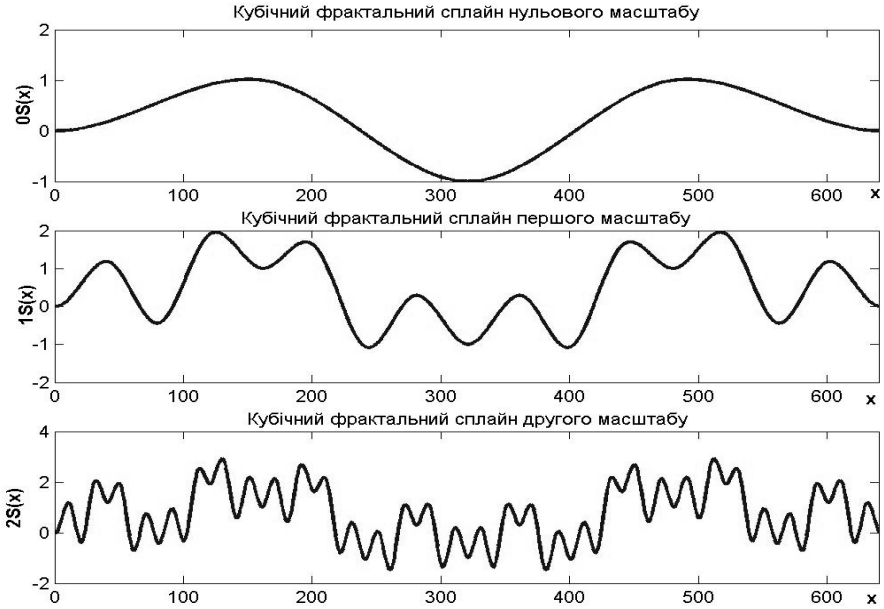

Рисунок 1 – Кубічний фрактальний сплайн з глибиною масштабу k=2.

Lazy Calculations

Більшість сучасних мов програмування (таких як C#, VB.NET, C++, Python і Java) базуються на так званих «негайних розрахунках», тобто операція виконується, як тільки стають відомі значення її операндів. Однак зрозуміло, що негайні обчислення багатьох функцій не завжди потрібні й раціональні з точки зору продуктивності, тому пропонується відкласти ці обчислення до того моменту, коли їх результат буде затребуваний.

Попередником Lazy Computations можна вважати патерн ***Lazy Load*** (завантаження на вимогу, ледаче завантаження), який ввів Мартін Фаулер у свій книзі «Архітектура корпоративних програмних додатків» [5, с. 220]. Суть даного патерна полягає в тому, що об'єкт не містить дані, але знає де їх завантажити, якщо вони йому стануть потрібні.

Розглянемо алгоритм простої програми:

1. Ініціалізація змінних.

2. Зчитування стану пристроїв керування.
3. Відновлення стану об'єктів.
4. Виведення інформації на екран.
5. Повернутися до п.2.

У даному циклі основний процесорний час витрачається на відновлення стану об'єктів і виведення інформації на екран. Оскільки стан об'єкта визначається за допомогою функцій фрактальної інтерполяції, що є складними і тривалими у розрахунку, то критичним є час, який буде приділятися на виведення інформації на екран (частота кадрів). Метод «лінивих обчислень» дозволяє зменшити кількість часу, яку процесор витрачає на відновлення свого стану.

Спершу слід вияснити природу затримок, які виникають при відновленні об'єктів, і обрати більш прийнятний метод відкладених обчислень.

Проблема 1. У програмі час від часу повинен виконуватися ресурсномісткий алгоритм, який вимагає багато процесорного часу. У цей момент спостерігаються затримки, а решту часу процесор «простоює». Така ситуація можлива при виконанні алгоритмів оптимізації, вставки нового вузла тощо.

У даному випадку виконання важкого алгоритму доцільно розбити на декілька частин і виконувати їх не одночасно, а з перервою на малювання кадру й відновлення інших об'єктів. Такий підхід дозволить суттєво знизити час виконання одного конкретного акту відновлення, але вимагає ускладнення коду програми, оскільки необхідно зберігати проміжні результати обчислень на поточному кроці для продовження їх у наступному.

Проблема 2. Має місце велика кількість подій, які виникають (або можуть виникнути) при інкрементації лічильника часу. Одночасна обробка всіх подій може привести до тимчасової затримки роботи програми.

У такому випадку слід мати список усіх подій і виконувати їх не відразу, а з інтервалами. При частоті 20-60 кроків відновлень у секунду, користувач не помітить різниці, яка подія відбулася на першому або на п'ятому кроці (це питання декількох часток секунди), але це дозволить досить сильно розвантажити процесор, щоб збільшити частоту кадрів. Модель обробки подій повинна формувати списки подій (*пріоритетні списки*) і нормувати виконання подій на одному кроці відновлення. Є декілька способів реалізації такого підходу:

– обробляти невелику кількість подій за крок (наприклад, 20% від загальної маси списку, але не менше 50).

– розраховувати загальний час, який витрачається на обробку подій, і при досягненні його критичного значення припиняти обробку й переходити до інших дій. У цьому випадку необхідно розуміти, що підрахунок часу виконання конкретної події також забирає певний час і може гальмувати роботу програми.

Застосування Lazy Calculations у задачах графічного представлення фрактальної інтерполяції

Розглянемо графічний редактор, який оперує з фрактальними часовими рядами, а саме: виводить графік фрактального часового ряду, оцінює його масштаб, розраховує часовий ряд на масштаб більше або менше і виводить його графік. Також графічний редактор дозволяє вирішувати обернену задачу: із заданої «затравки» побудувати фрактал певної складності.

Розглянемо можливість підвищення ефективності роботи такого редактора при виконанні операцій масштабування (zoom). Особливістю математичних фракталів є те, що вони дозволяють необмежене масштабування в усіх напрямках. Тобто щоразу, коли користувач збільшує зображення, то більше деталей він бачить, і щоразу, коли він зменшує зображення, він бачить укрупнені об'єкти. Головна задача в такій системі – це оперативність у зміні зображень.

Якщо користувачу дозволено редагувати зображення (наприклад, ускладнювати фрактал, додавши вузол до «затравки» або перемістивши його), то постає задача ефективної організації розрахунків фрактальної інтерполяції. Для вирішення цих двох задач пропонується використовувати технологію «лінивих обчислень».

Процес багатомасштабного перетворення являє собою цикл, у якому наступні операції повторюються:

– змінити зображення на масштабі x,
– перейти вгору або вниз на рівень.

Зміна зображення відповідає операціям вставки, видалення або переміщення вузла фрактального сплайна, що впливає лише на поточний масштаб. Зміни в зображенні повинні розповсюджуватися на інші масштаби, але лише для видимих компонентів.

Щоб збільшити масштаб, R фрагментів укрупнюються в один. Щоб зменшити масштаб, кожний фрагмент поточного сплайну розпадається на R фрагментів.

Операції масштабування можуть бути вкладені одна в одну. З точки зору користувача, такий інтерфейс поводиться як стек LIFO. Користувач «додає елемент у стек», щоб збільшити графічне представлення, і «забирає елемент зі стеку», щоб повернутися до попереднього представлення.

Реалізація скролінгу, тобто бічного зсуву зображення, також реалізується особливим чином. Для горизонтального або вертикального скролінгу користувачеві необхідно:

1) зменшити масштаб,
2) перевести курсор у нову позицію,
3) збільшити масштаб.

Цей спосіб скролінга має важливе значення для розрахунків і дозволяє включити «ліниві обчислення» в багатомасштабну композицію.

Для реалізації вкладених масштабувань відносно нерухомої точки, було організовано стек зсу-

ву зображення. Зсув для кожного збільшення рівня дає відносну позицію (x, y) зсуву кута зору користувача відносно основного представлення.

Щоразу, коли користувач збільшує масштаб відносно нерухомої точки (x,y) видимої частини зображення, визначається:

$$offset[level] := 2 \times offset[level-1] + \\ + (x - x \bmod 2, y - y \bmod 2) \quad (2)$$

Щоразу, коли користувач зменшує масштаб, повертається попереднє значення offset[level-1].

Управління пам'яттю

Для підвищення ефективності доступу до пам'яті, важливо, щоб локальність на зображення відповідала локалізації в пам'яті: два пікселі, які близько розташовані один до одного на зображенні, повинні з високою ймовірністю розташовуватися поруч у пам'яті [2, с. 315]. Ця умова повинна виконуватися на всіх рівнях деталізації.

Реалізація такої умови можлива у наступний спосіб. Різні масштаби фрактальних сплайнів керуються незалежно. Кожний масштаб s фрактального сплайна управляється як нескінченний двовимірний «віртуальний масив» $W_s[i, j]$ коефіцієнтів фрактального сплайна, де $0 \le i < \infty$ та $0 \le j < \infty$.

Управління пам'яттю кожного віртуального масиву здійснюється за принципом прогресивного підкачування, яке використовується у файловій системі UNIX. Пропонується узагальнений варіант цієї схеми для більш високих роздільних здатностей. Віртуальний масив зберігається у вигляді дерева сторінок. Кожна сторінка містить $res \times res$ пікселів, і складається із двох половин: перша половина містить вказівки на нащадків сторінки, друга половина містить фактичні дані.

Нехай $N = res \times res$, тоді сторінки у віртуальному просторі організовані в такий спосіб (рис. 2):

- Сторінка 0 - перша сторінка,
- Сторінки: 1 ... N-1 розміщені на сторінках, на які вказує перша сторінка.
- Сторінки N... N * N-1 розміщені на сторінках, на які вказують попередні сторінки, і т.д.

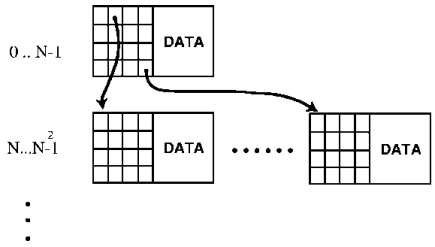

Рисунок 2 – Організація вказівок на різномасштабне представлення даних.

Важливим є, що перша комірка в першому вузлі ніколи не використовується. Ця комірка використовується для зберігання res.

Вважаємо, що всі сторінки віртуального масиву лежать на нескінченній сітці і проіндексовані по I та J, де сторінка I, J містить всі пікселі, такі, що:

$$res \times I \le i < (res \,|\, 1) \times I \quad (3)$$
$$res \times J \le j < (res \,|\, 1) \times J$$

Сторінка [I, J] отримується у два етапи. Спершу розраховується число N, яке визначає шлях вниз по дереву на дану сторінку:

```
k := 0
while I > 0 and J > 0 do
path[k]  :=  mod(I,res)  +  res  *
mod(J,res)
I := I / res
J := J / res
k := k + 1
```

коли шлях визначено, програма слідує ним вниз по дереву:

```
page := firstPage
while k > 0 do
k := k - 1
if page[path[k]] = NULL
page[path[k]] := newPage()
page := page[path[k]]
```

Коли прямокутник пікселів зчитується чи зберігається в пам'ять, то обробляються лише ті сторінки, які його містять (також сторінки можуть створюватися в разі необхідності).

Програмний код генерувався на компіляторі mingw 5.1. Тестування характеристик проводилось на комп'ютері з параметрами: процесор AMD Athlon 64 X2 Dual, частота 2,8 ГГц, ОЗУ 1 Гб, ОС Windows XP SP3. Результати тестування представлені у табл. 1.

Таблиця 1. Порівняння затрат системних ресурсів на виконання деяких операцій графічного виведення з та без використання технології «лінивих обчислень».
1- без «лінивих обчислень», 2 - з «лінивими обчисленнями», 3 – відсоток зміни

Операція	Витрати часу, мс			Витрати пам'яті, Мб		
	1	2	3	1	2	3
Виведення фрактального часового ряду з 2000 точок	200	150	-25%	43	50	+16,27%
Масштабування часового ряду до 400 точок у вікні	700	360	-48,57%	50	68	+36%
Побудова фрактального сплайну з «затравки»	533	300	-43,71%	25	32	+28%
Вставка нового вузла у фрактальний сплайн	1025	760	-25,85%	64	78	+21,87%

Як видно з табл. 1, використання «лінивих розрахунків» для графічного редактора, який оперує фрактальними часовими рядами, дає виграш у часі виконання, але приводить до додаткових витрат пам'яті. Але оскільки сучасні комп'ютери звичайно мають досить великий об'єм оперативної пам'яті, подібні витрати не приведуть до зниження швидкодії. Також слід відмітити, що зростання витрат пам'яті відбувається меншими темпами, ніж прискорення виконання операцій.

Висновки

Організація обчислювального процесу на основі «лінивих розрахунків» дозволяє говорити про перспективність такого підходу з точки зору підвищення гнучкості розв'язку. Використання принципу Lazy Computations для задач фрактальної інтерполяції дозволяє значно підвищити швидкість розрахунків та частоту кадрів.

Наукова новизна роботи полягає в тому, що вперше запропоновано оптимізаційні методи для графічного представлення результатів фрактальної інтерполяції.

Предметом подальших розробок є дослідження алгоритму розпаралелювання в поєднанні з «лінивими обчисленнями».

Список використаної літератури

1. Karezmarczuk Jerzy. Generating Power of Lazy Semantics // Jerzy Karezmarczuk // Theoretical Computer Science. – 1997. – Vol. 187, Issue 1-2. – P. 102-132.
2. Pion Sylvain. A Generic Lazy Evaluation Scheme for Exact Geometric Computations / Sylvain Pion, Andreas Fabri // Science of Computer Programming. – 2011. – №76 (4). – P. 307-323.
3. Бородянский Ю.М. Разработка алгоритмов построения сплайнов на основе дельта-преобразований второго порядка для интерполяции кривых и поверхностей в компьютерной графике: автореф. дис. канд. техн. наук: 05.13.17, 05.13.18 / Юрий Михайлович Бородянский. – Таганрог, 2003. – 20 с.
4. Васильев С.Н. Методы фрактальной интерполяции типа Барнсли / С.Н. Васильев // Известия высших учебных заведений. Серия: Математика. – 2002. – Вып. 9 (484). – С. 3-14.
5. Математические основы машинной графики / Д. Роджерс, Дж. Адамс; пер. с англ. П.А. Монахова, Г.В. Олохтоновой, Д.В. Волкова; под ред. Ю.М. Баяковского. – М.: Мир, 2001. – 604 с.
6. Скопин И.Н. Ленивые и жадные вычисления в числовых задачах [Электронный ресурс]. – Режим доступу: http://www2.sscc.ru/SORAN-INTEL/paper/2010/Skopin-29-04-10a.pdf. – Дата доступу: березень, 2012. – Назва з екрана.

Надійшла до редколегії 02.04.2012

О. НОВИКОВА
Национальный авиационный университет

LAZY COMPUTATIONS КАК МЕХАНИЗМ ПОВЫШЕНИЯ ЭФФЕКТИВНОСТИ ФРАКТАЛЬНОЙ ИНТЕРПОЛЯЦИИ

Рассмотрено использование принципов «ленивых вычислений» для графического представления результатов фрактальной интерполяции. В качестве базовых функций выбраны фрактальные сплайны, имеющие легкий и эффективный алгоритм реализации и адекватно описывающие фрактальные модели. Результаты тестирования показывают, что использование «ленивых вычислений» позволяет уменьшить задержку кадров при выводе информации на экран.

Ключевые слова: фрактальная интерполяция, фрактальный сплайн, графическое представление данных, lazy computations.

O. NOVIKOVA
National Aviation University

LAZY COMPUTATIONS AS A MECHANISM TO ENHANCE THE EFFECTIVENESS OF THE FRACTAL INTERPOLATION

There is considered the usage of "lazy calculations" principles for graphic presentation of fractal interpolation results. As the basic functions fractal splines are chosen because they have easy and efficient algorithm implementation, and adequately describe the fractal model. There are presented key points of software development for this task. Test results prove that applying "lazy calculations" can reduce the delay of frames in information representation on the screen.

Keywords: fractal interpolation, fractal splines, graphical representation, lazy computations.