УДК 519.213:257, 519.657

Є.В. Крилов, В.К. Анікін, О.Є. Булацан

ОПТИМІЗАЦІЯ ТРИВИМІРНИХ МОДЕЛЕЙ З МЕТОЮ ЇХ ВИКОРИСТАННЯ У СЕРЕДОВИЩІ ІНТЕРНЕТ

Введення

Представлення тривимірних сцен та об'єктів на екрані монітора вимагає наявності на комп'ютері користувача достатніх розрахункових ресурсів та потужного графічного чіпу. Ця проблема вже практично вирішена, так як сучасне покоління комп'ютерів та загальний стан обчислювальної техніки вже дозволяють вільно працювати із тривимірною графікою.

На сьогоднішній день не існує єдиного загального стандарту для представлення тривимірної графіки в мережі Інтернет. За останні 30 років було сформовано декілька основних концепцій, по ним були прийняті відповідні стандарти: VRML (ISO/IEC 14772-1:1997), X3D(ISO/IEC 19775). Через відсутність єдиного стандарту на представлення тривимірної графіки у мереж Інтернет, більшість успішних, в рамках продуктивності, комерційних продуктів використовують свої власні програми для перегляду – плагіни. На жаль, на даний час не існує повної узгодженості чи домовленостей між розробниками плагінів та браузерів (програм для перегляду Інтернет-середовища). Це спричиняє додаткові труднощі для користувачів та сповільнює масовий розвиток технології.

Мета роботи

Метою роботи та даної статті є дослідження та спроба вирішення однієї із ключових проблем в даній сфері – зменшення розміру (об'єму) вихідного файлу з описом тривимірного об'єкта чи сцени, із збереженням при цьому достатньої реалістичності об'єкту. Саме вищезазначений вихідний файл має завантажуватися на комп'ютер користувача для перегляду тривимірної сцени.

Критеріями для оптимізації файлу виступають два параметри:

1. Розмір (об'єм) вихідного файлу;
2. Ступень реалістичності оптимізованого файлу відповідно оригінального;

Об'єм файлу вимірюється в байтах даних. В ідеальному випадку, розмір оптимізованого файлу має становити не більше 1 МБ (1024 КБ = 1048567 байт). В такому випадку при швидкості доступу до Інтернету 56Кб/сек., час завантаження складе не більше 20 секунд. По даним авторитетних світових Інтернет-компаній на 2009 рік середня швидкість доступу в розвинених країнах вже значно перевищує 56Кб/сек. (256Кб/сек. та більше), при таких умовах швидкість завантаження файлу до 1Мб

має становити 2-3 секунди, що є сприятливим результатом для даного дослідження. При таких умовах поріг максимально об'єму файлу можна підняти до 3 МБ. При швидкості 256Кб/сек. (або більше) час завантаження такого фалу становитиме не більше 10 сек.

Степінь реалістичності вимірюватиметься через коефіцієнт $K_{\text{реал}}$, і визначатиметься як: $K_{\text{реал}} = N_o + N_{но}$. Де $N_{но}$ кількість вершин трикутників у не оптимізованого тривимірного об'єкта, а N_o – відповідна кількість вершин у оптимізованого. Даний механізм пов'язаний із тим, що для побудови будь-якого тривимірного об'єкта в даний момент використовується система взаємопов'язаних трикутників. Зменшення вершин (і відповідно, трикутників) призводить до зменшення об'єму вихідного файлу, але й до погіршення зовнішнього вигляду тривимірного об'єкту. Експериментальним шляхом було встановлено, що при величині коефіцієнта більше 0.4 (тобто 40% від кількості трикутників оригінального об'єкта) – як правило, реалістичність об'єкта залишається допустимою.

Оптимізація за допомогою алгоритму Хаффмана

Оскільки файл з усіма даними стосовно тривимірного об'єкту містить у собі певний набір символів, що описують даний об'єкт, можна скористатися певним алгоритмом, який закодує цю послідовність символів та зменшить розмір файлу. При цьому файл тривимірного об'єкту зменшиться у розмірі, а зовнішній вигляд об'єкту залишиться таким самим як і оригінал – оскільки кількість вершин (мешів) не буде змінена. Для відображення тривимірного об'єкту у вікні браузера користувача, знадобиться спеціальний плагін (програвач), який декодує послідовність символів та інтерпретує їх у тривимірний об'єкт.

У світі існує багато спеціальних технік, які дозволяють закодувати та відображати з мінімальними затратами бітові поля змінної довжини, що використовуються, наприклад, для відображення геометрії тривимірних об'єктів. У даному дослідженні для оптимізації таких об'єктів було застосовано всесвітньо відомий алгоритм Хаффмана – адаптивний алгоритм оптимального префіксного кодування з мінімальною збитковістю, розроблений у 1952 році доктором Масачусетского технологічного інституту Девідом Хаффманом.

На відміну від алгоритму Шеннона-Фано, алгоритм Хаффмана залишається завжди оптимальним і для вторинних алфавітів m_2 з більш ніж двома символами [4].

Цей метод кодування складається із двох основних етапів:

1. Побудова оптимального кодового дерева;
2. Побудова відображення код-символ на основі побудованого дерева;

Алгоритм Хаффана має наступний вигляд:

1. Підраховуються ймовірності появи символів первинного алфавіту у вихідному тексті (якщо вони не задані заздалегідь);
2. Символи первинного алфавіту m_1 виписуються у порядку убування ймовірностей;

 ISSN 1562-9945

3. Останні n_0 символів поєднуються у новий символ, ймовірність якого дорівнює сумарній ймовірності цих символів, ці символи видаляються, а новій символ вставляється у список на відповідне місце (по ймовірності). n_0 обчислюється із системи (1):

$$\begin{cases} 2 \leqslant n_0 \leqslant m_2 \\ n_0 = m_1 - a\,(m_2 - 1)\,, \end{cases} \tag{1}$$

де a – ціле число, m_1 і m_2 – потужність первинного й вторинного алфавіту відповідно.

1. Останні m_2 символів знову поєднуються в один і вставляються на відповідну позицію. Символи, що ввійшли в об’єднання, попередньо видаляються.
2. Попередній крок повторюється доти, доки сума всіх m_2 символів не буде доривнювати 1.

Цей процес можна представити як побудова дерева, корінь якого - символ з ймовірністю 1, отриманий після об’єднання символів на останньому кроці, де m_2 – символи з попереднього кроку й т.д.

Кожні m_2 елементів, що знаходяться на одному рівні, нумеруються від 0 до $m_2 - 1$. Коди отримують з шляхів (від першого нащадка кореня й до листка). При декодуванні можна використовувати те ж саме дерево. Воно зчитується по одній цифрі, далі робиться крок по дереву, поки не досягається “лист”. Після цього виводиться символ, що знаходиться на даному “листі” й виконується повернення в корінь.

Бінарне дерево, що відповідає коду Хаффмана, називається деревом Хаффмана.

Можна також стверджувати, що побудова коду Хаффмана рівносильна побудові відповідного йому дерева.

Загальна схема побудови дерева Хаффмана виглядає наступним чином:

1. Складеться список символів, що кодуються (при цьому кожний символ розглядається як одноелементне бінарне дерево, вага якого дорівнює вазі символу).
2. Зі списку вибирається 2 вузли з найменшою вагою (під вагою можна розуміти частоту використання символу - чим частіше використовується, тим більше важить).
3. Формується новий вузол, до нього приєднується у якості дочірніх два інших вузли обраних зі списку. При цьому вага сформованого вузла залишається рівною сумі ваг дочірніх вузлів.
4. Новий сформований вузол додається до списку.
5. Якщо в списку більше одного вузла, пункти 2–5 повторюються. [5]

Алгоритм Хаффмана дає змогу закодувати набір символів.

Розглянемо приклад кодування файлу с описом тривимірного об’єкту. Маємо файл об’ємом 100 байт, що містить 6 різних символів. Підрахувавши входження кожного із символів у файл (Таблиця 1), одержуємо наступне :

Таблиця 1

Входження кожного із символів у файл

Символ	**A**	**B**	**C**	**D**	**E**	**F**
Число входжень	10	20	30	5	25	10

На наступному етапі ці числа називаємо частотою входження для кожного символу. Упорядкуємо числа входжень символів по спаданню (Таблиця 2).

Таблиця 2

Упорядковані по спаданню числа входжень

Символ	**C**	**E**	**B**	**F**	**A**	**D**
Число входжень	30	25	20	10	10	5

Візьмемо з останньої таблиці символи з найменшою частотою. У нашому випадку це D (5) і один з символів F або A (10), можна взяти будь-який з них наприклад A. Сформуємо з “вузлів” D і A новий “вузол”, частота входження для якого буде дорівнювати сумі частот D і A (рис. 1):

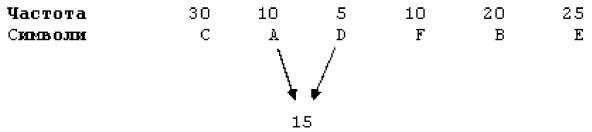

Рис. 1 – Формування вузла з вузлів A і D

Номер у рамці - сума частот символів D і A. Тепер ми знову шукаємо два символи з найнижчими частотами входження. Виключаючи із перегляду D і A і розглядаючи замість них новий “вузол” із сумарною частотою входження. Найнижча частота тепер в F і нового “вузла”. Знову виконаємо операцію злиття вузлів (рис. 2) :

Розглядаємо таблицю знову для наступних двох символів (B і E). Ми продовжуємо виконувати ці дії, поки все “дерево” не сформоване, тобто поки всі елементи не зведуться до одного єдиного вузла (рис. 3):

Коли дерево створене, можемо кодувати файл. Необхідно завжди починати із кореня (Root). Кодуючи перший символ (лист дерева C), ми простежуємо вверх по дереву всі повороти гілок і якщо ми робимо лівий поворот, те запам’ятовуємо 0-й біт, і аналогічно 1-й біт для правого повороту. Так для C, ми будемо йти вліво до 55 (і запам’ятаємо 0), потім

 ISSN 1562-9945

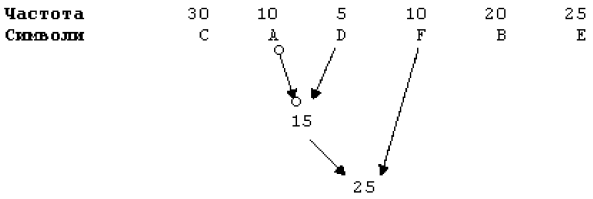

Рис. 2 – Формування вузла з вузлів A D і F

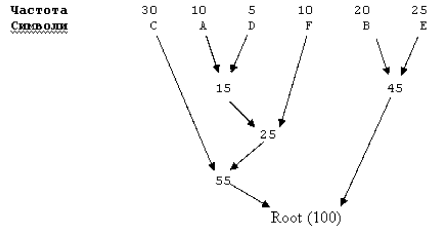

Рис. 3 – Формування останнього вузла

знову вліво (0) до самого символу. Код Хаффмана для нашого символу C – 00. Для наступного символу (A) у нас виходить - ліво, право, ліво, ліво , що виливається в послідовність 0100. Виконавши вище сказане для всіх символів одержимо

C = 00 (2 біти)
A = 0100 (4 біти)
D = 0101 (4 біти)
F = 011 (3 біти)
B = 10 (2 біти)
E = 11 (2 біти)

Кожний символ спочатку може бути представлений 8-ома бітами (один байт). Оскільки ми зменшили число бітів, необхідних для подання кожного символу, розмір вихідного файлу також зменшився. Результати стиснення вираховуються наступним чином (Таблиця 3):

Початковий розмір файлу 100 байт (800 біт). Розмір стислого файлу 30 байт (240 біт). Коефіцієнт стиснення цього файлу 70%.

Для відновлення первісного файлу, ми повинні мати декодувальне дерево, тому що кожен файл матиме своє унікальне дерево. Отже ми повинні зберігати дерево разом з файлом. Це в певній незначній мірі збільшує розмір вихідного файлу.

У даній методиці у кожному вузлі перебувають 4 байти покажчика, тому повна таблиця для 256 байт буде мати розмір приблизно 1 Кбайт.

Таблиця в даному прикладі має 5 вузлів плюс 6 вершин (де й перебувають символи) , усього 11.4 байта 11 разів - 44. Якщо ми додамо після невеликої кількості байтів, для збереження положення вузла, ще

Таблиця 3

Результати стиснення

Частота	Початкова величина	Стиснена величина	Різниця
C 30	30 x 8 = 240	30 x 2 = 60	180
A 10	10 x 8 = 80	10 x 3 = 30	50
D 5	5 x 8 = 40	5 x 4 = 20	20
F 10	10 x 8 = 80	10 x 4 = 40	40
B 20	20 x 8 = 160	20 x 2 = 40	120
E 25	25 x 8 = 200	25 x 2 = 50	150

й деяку іншу статистику - наша таблиця буде приблизно 50 байтів по величині [2].

Додавши до 30 байтів стислої інформації, 50 байтів таблиці одержуємо, що розмір загального архівного файлу - до 80 байт. З огляду на , що первісна величина файлу в розглянутому прикладі була 100 байт – ми одержали 20% стиск інформації.

В результаті, початковий файл розміром у 256 байт було стиснуто за допомогою коду Хаффмана до 195 байтів, або на 33%. Для префіксного коду , коефіцієнт стиснення може складати до 70%. Необхідно додати, що ключем до побудови префіксних кодів служить звичайне бінарне дерево і якщо уважно розглянути попередній приклад з побудовою дерева, можна переконається , що всі одержувані там коди – префіксні.

Висновки

Тенденція використання технологій представлення тривимірної графіки на сторінках глобальної мережі Інтернет неухильно займає своє місце, поступово витісняючи звичні суспільству статичні і нецікаві елементи та схеми. Перспективність технології осмислюють все більше компаній в різних країнах світу, підтвердженням цього є драматичне збільшення кількості відповідних проектів та розробок у 2006–2009 роках. Світова фінансова криза стала додатковим стимулом для компаній у пошуку нових методів, технологій та схем збільшення об'ємів продажу своїх товарів.

В даній роботі було проведено дослідження та зроблена спроба вирішення однієї із ключових проблем, що заважають масовому впровадження технології. В рамках дослідження було проведено ряд експериментів по оптимізації розміру (об'єму) вихідного файлу з описом тривимірних сцен за допомогою алгоритму оптимального префіксного кодування з мінімальною збитковістю – алгоритму Хаффмана.[1]

Використання алгоритму Хаффмана для оптимізації тривимірних моделей дали дуже продуктивні результати. На базі алгоритму та серії інших досліджень компанією TurnTool ApS® у 1998 було створено проект TurnTool™. Станом на 2009 рік TurnTool™ є однією з найкращих програм, що дозволяють розміщувати елементи тривимірної графіки на

 ISSN 1562-9945

Інтернет сторінках. На базі TurnTool™ студією веб-дизайну ClearDesign було створено власний проект – "Тривимірні моделі для вашого сайту", що дозволяють користувачу оглянути та взаємодіяти із фото реалістичними тривимірними об'єктами нерухомості, автомобілями та домашньою фурнітурою. Таким чином студія ClearDesign® однією з перших приватних компаній в Україні, що почали надавати можливість розміщення тривимірних інтерактивних моделей на сайті клієнта.

Автори статті вважають, що вже зовсім скоро суспільство у світі зможе повною мірою користатися перевагами тривимірної технології – купувати товари, автомобілі, нерухомість, подорожувати відомими та найяскравішими місцями на планеті, побачити найкращі архітектурні та культурні споруди, не відходячи від екрану монітора свого комп'ютера.

Литература

1. Alexander J.W. The combinatorial theory of complexes. Annals of Mathematics
2. Alliez P. and Desbrun. M. Valence–driven connectivity encoding of 3D meshes. In urographics, pages 480–489. Blackwell, 2001.
3. Alliez P., Cohen–Steiner D., Devillers O., Levy B. and Desbrun M. Anisotropic polygonal remeshing. In Siggraph. ACM, 2003.
4. Wheeler F.W. Adaptive arithmetic coding source code.
5. Huffman. D.A. A method for the construction of minimum redundancy codes. In I.R.E, pages 1098–1102, 1952.
6. Baumgart B.G. Winged edge polyhedron representation. Technical Report AIM-179 (CS-TR-74-320), Stanford University, 1972.

Отримано 25.03.2009 р.