УДК 681.323

В.М. ГУСЯТИН, канд. техн. наук,
Я.В. ЧАГОВЕЦ, канд. техн. наук,
Д.Г. КОЖУШКО

УПАКОВКА ВЕКТОРНЫХ ТЕКСТУР В ЗАДАЧАХ СИНТЕЗА ИЗОБРАЖЕНИЙ ДЛЯ СИСТЕМ ВИЗУАЛИЗАЦИИ

Розглядається упаковка векторних текстур. Застосування векторних текстур дозволяє суттєво скоротити об'єми пам'яті при збереженні їх із високою роздільною здатністю. Вхідні дані для векторної форми надання перетворюються до деревоподібної структури. Розглянуто алгоритми упаковки як аналітично наданих, так і довільних векторних фігур. Структури, що їх отримано, дозволяють наносити текстуру без попередньої розпаковки до растрової форми надання, що прискорює її відображення. Основною особливістю такої упаковки текстури є можливість усунення аліайзинга в процесі нанесення її на поверхню об'єкта. Нанесення текстури добре реалізується при використанні методу зворотного трасування в системах візуалізації 3D-сцен.

Packing of vector textures is considered. Using vector textures allows to significantly decrease memory size while storing them with high resolution. Input vector data are transformed into tree-like structure. Packing algorithms for both analytical and arbitrary vector textures are considered. Obtained structure allows texture mapping without pre-unpacking it into the raster form which speeds-up its rendering. Main feature of such a texture packing scheme is ability to remove an aliasing during texture mapping. The proposed texture mapping is well implemented if used with ray-tracing in 3D visualization systems.

Постановка проблемы. В процессе синтеза 3D-сцен [1] возникает необходимость оперировать с большими объемами текстур. Кроме того, при отображении текстур возникает проблема устранения алиайзинга. Таким образом, необходимо разработать такие методы упаковки текстур, которые бы позволяли при отображении текстур устранять алиайзинг с минимальными затратами времени и объема памяти.

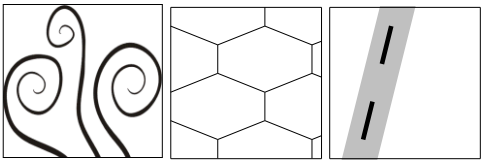
Анализ литературы. В работах [2 – 4] используется метод, основанный на использовании MIP-карт, применение которых позволяет вводить уровни детализации текстуры с целью устранения алиайзинга. В [5] описывается подход к синтезу реалистичной растровой текстуры с большим разрешением. В работах [6 – 7] синтезируется растровая текстура таким образом, чтобы устранить алиайзинг при ее нанесении на криволинейную поверхность. Во всех рассмотренных вариантах предполагается растровое задание текстур. Во многих случаях возникает необходимость работать с текстурами, представленными в векторной форме.

Цель статьи. Разработка форматов упаковки векторных текстур, которые, с одной стороны, позволяют уменьшить объемы текстур, а с другой стороны, имеют такой формат, который позволяет их отображать в реальном времени без перевода в растровую форму с одновременным устранением алиайзинга.

Классификация исходных изображений текстур. Известно деление изображений текстур на растровые и векторные. Векторные изображения характерны тем, что они состоят из видимых областей, разделенных линиями. В зависимости от формы линий, разделим их на два вида: с произвольными линиями раздела и с линиями раздела, имеющими простое аналитическое описание (прямые, эллиптические дуги и некоторые другие). Последние могут иметь линии раздела как равномерно, так и неравномерно распределенные по области изображения. Назовем их для краткости «векторные простые равномерные изображения» и «векторные простые неравномерные изображения».

Растровые изображения. В качестве растровых целесообразно представлять изображения, не имеющие четко выраженных и разделенных областей (полутоновые изображения с плавными градациями цвета, фотографические изображения).

Векторные изображения с произвольными линиями раздела. В данную категорию попадают векторные изображения, границы областей которых сложно описать аналитически. К таким изображениям относятся: изображения рек, полей и т.п. (рис.1, а).



а) б) в)

Рис.1. Виды векторных изображений: а) с произвольными линиями раздела; б) векторные простые равномерные; в) векторные простые неравномерные

Векторные простые равномерные изображения. Изображения, состоящие из областей, границы которых равномерно распределены по площади текстуры (например, изображение орнамента, кирпичная кладка и т.п.) (рис.1, б).

Векторные простые неравномерные изображения. Изображения, состоящие из областей, разделенных прямыми линиями и дугами эллипсов, и неравномерно распределенных по площади текстуры (рис. 1, в). Примеры: искусственные объекты (аэропорты, разметка дороги и т.п.) Под неравномерным распределением понимается сосредоточение детализации в некоторых областях текстуры.

Форматы упакованных текстур. Формат текстуры представляет собой кватернарное дерево (КД), англ. quadtree [8, 9]. Известны две разновидности КД: полное и неполное. Полное кватернарное дерево целесообразно использовать для текстур, в которых детализация равномерно распределена по площади (растровые и векторные простые равномерные изображения). В

литературе известны способы представления растровых текстур в виде полного кватернарного дерева (например, MIP-карты [10]). Неполное кватернарное дерево подходит для случаев, когда текстура имеет неравномерное распределение детализации (векторные с произвольными линиями раздела, векторные простые неравномерные), что позволяет существенно сократить объемы памяти.

Ключевым понятием при рассмотрении упаковки и распаковки текстуры является так называемый классификационный квадрат [9]. Классификационный квадрат (КК) – это область текстуры, имеющая форму квадрата, которая, в свою очередь, может делиться на четыре КК меньшего размера. В конечном итоге текстура состоит из конечного множества КК.

Рассматривается несколько форматов КД, различающихся между собой, с одной стороны, структурой данных, с другой стороны, видом информации, хранимой в этой структуре данных.

Структуры данных. Структура данных различается для полного и неполного кватернарного дерева. Структура неполного кватернарного дерева предполагает хранение информации о связи между родительской и дочерней вершиной. Для полного дерева в хранении такой информации нет необходимости.

Вид информации, хранимой в вершине дерева. Во всех нелистовых вершинах дерева хранится интегральный цвет соответствующего КК.

В листовой вершине возможны следующие виды хранимой информации:

а) цвет;

б) текстура и интегральный цвет КК;

в) информация об областях и их линиях раздела (прямые или дуги эллипсов) и интегральный цвет КК.

В свою очередь, область может быть одноцветной либо текстурированной. Последний случай подразумевает, что векторная текстура содержит в качестве области другую текстуру. Такую текстуру назовем композиционной. Использование композиционной текстуры позволяет разработчику гибко соединять любой набор текстур.

Таким образом, особенностью предлагаемого способа хранения текстур является то, что во всех вершинах хранится интегральный цвет.

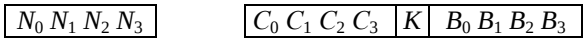
Алгоритмы упаковки текстур. Процесс упаковки растрового изображения полностью соответствует процессу подготовки MIP-карт, который широко известен в литературе [10], поэтому в данной статье не рассматривается.

Упаковка векторного изображения с произвольными линиями раздела.

Подготовка к построению дерева. В начале производится растеризация исходного векторного изображения с получением растрового изображения размером 2^N на 2^N элементов. Для этого могут быть использованы средства растеризации, входящие в состав стандартных графических пакетов, таких как CorelDraw. Выбор значения N производится исходя из требуемой точности.

Алгоритм упаковки. На основе растрового представления векторного изображения строится неполное кватернарное дерево. В литературе известны алгоритмы его построения [9]. Предлагается для уменьшения объема дерева и в дальнейшем для уменьшения времени нанесения текстуры с одновременным устранением алиайзинга внести дополнительные особенности в структуру дерева и алгоритм упаковки, описание которых приведено ниже:

а) Последний уровень дерева состоит из палитры цветов и списка КК. Каждый КК из этого списка имеет формат 1 (рис. 2, а), что соответствует хранению цвета в листовой вершине. Палитра позволяет по индексу получить соответствующий ему цвет. Она содержит все цвета, которые присутствуют в исходном растровом изображении. В данной реализации размер палитры не может превышать 256, так как на хранение каждого индекса отводится 1 байт.



а) б)

Рис. 2. Формат КК: а) формат 1, б) формат 2,
N_i – номер i-го индекса в палитре; C_i – усредненный цвет i-го дочернего КК; B_i – признак делимости i-го дочернего КК; K – номер первого делимого дочернего КК в списке уровня.

б) Каждый последующий верхний уровень является списком КК в формате 2 (рис. 2, б).

в) В процессе упаковки с целью разгрузки оперативной памяти промежуточные данные хранятся в файлах КК. Каждый такой файл содержит матрицу, элементами которой являются КК в формате 2 (рис. 2, б).

Для упаковки текстуры используется функция **PackTree**:

```
void PackTree(in Растровый_файл, inout Текстура){
    PackLastLevel(Растровый_файл, Текстура, Файл_КК);
    while(не достигнут верхний уровень){
        Исходный_файл_КК = Файл_КК;
        PackLevel(Исходный_файл_КК,Текстура,Файл_КК);} }
```

Функция работает в два этапа. На первом этапе упаковывается последний уровень (функция **PackLastLevel**), а на втором – все остальные верхние уровни дерева (функция **PackLevel**).

Функция **PackLevel** принимает в качестве параметров: исходный файл КК (рис.3), ссылку на текстуру, для которой будет добавлен очередной уровень дерева, ссылку на файл КК, в который будет записана информация, необходимая для построения следующего уровня дерева:

```
void PackLevel(in Исходный_файл_КК,inout Текстура,out Файл_КК){
    for_each(Строка из Файла_КК){
        считываем две строки Исходного_файла_КК;
        for_each(текущий_КК из Строки){
            Выделяем дочерние_КК;
            for_each(дочерний_КК){
                if(дочерний_КК содержит хотя бы одно дерево
                    или содержит различные цвета){
```

```
        Устанавливаем в текущем_КК признак
        делимости данного дочернего_КК;
        Вписываем Дочерний_КК в соотв уровень
        Текстуры;     }
    назначаем усредненный цвет дочернего_КК в
    соответствующую ячейку текущего_КК; } } } }
```

Функция **PackLastLevel** аналогична функции **PackLevel**, за исключением того, что она учитывает специфику формирования последнего уровня. Работа функции **PackTree** проиллюстрирована на рис. 3.

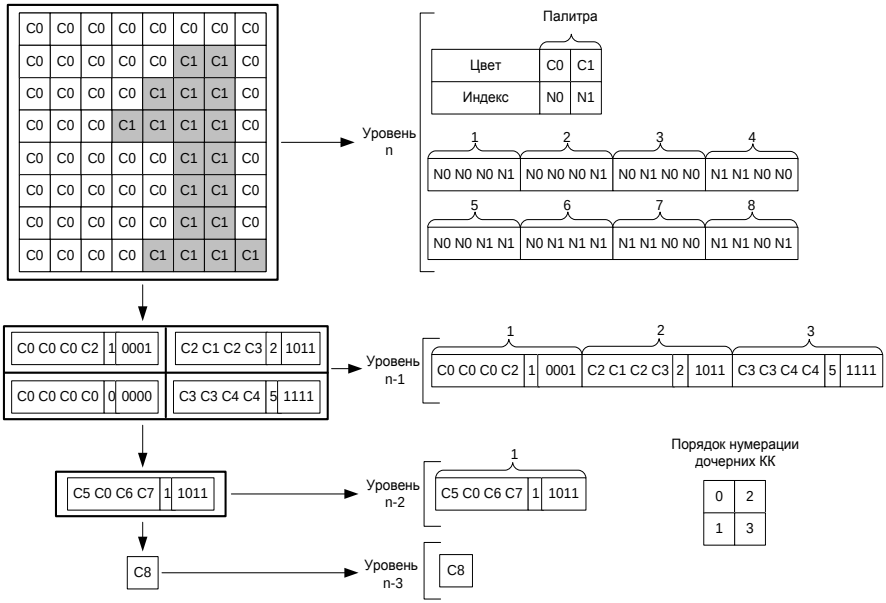


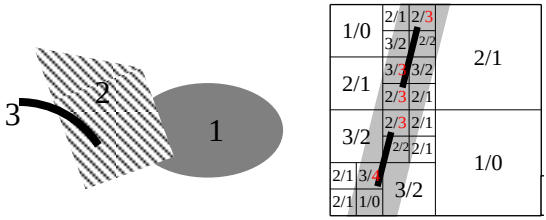
Рис. 3. Иллюстрация процесса упаковки векторного изображения со сложными линиями

Было проведено моделирование, в результате которого исходное растровое представление векторной текстуры фюзеляжа самолета (размер исходного растрового представления 201 326 646 байт) было упаковано данным алгоритмом, при этом размер текстуры составил 1 825 834 байт.

Упаковка векторного простого неравномерного изображения

Подготовка к построению дерева. Исходное векторное изображение готовится в пакете CorelDraw и оттуда экспортируется в файл формата PostScript. Экспортированный файл содержит упорядоченный набор фигур. Каждая фигура может быть одного из следующих типов (рис. 4, а): выпуклый многоугольник; площадь, ограниченная эллипсом; дуга эллипса заданной ширины (мы ограничивались фигурами лишь тех типов, для которых нами был найден соответствующий алгоритм распаковки). Изображение,

закодированное таким образом, можно получить, прочертив все фигуры в порядке возрастания номеров (рис. 4, а).



а) б)

Рис. 4. Элементы векторного простого изображения

Алгоритм упаковки. Текстура состоит из списка фигур и дерева. Каждая фигура в списке содержит информацию о своих границах и заливке. Дерево в каждой своей вершине хранит информацию о четырех КК, каждый из которых имеет формат рис. 5, а.

R	K	C

а)

F_0	...	F_M	L_0	...	L_{M-1}

б)

Рис. 5. а) формат КК; б) формат R

Здесь K – номер соответствующего делимого дочернего КК в дереве, либо «–1», если данный КК – листовой; C – интегральный цвет КК, R – запись, в которой:

F_i – номер фигуры, $i = 0,\ldots,M$;

L_j – номер линии, $j = 0,\ldots,M-1$.

Данный формат соответствует хранению информации об областях, их линиях раздела и интегральном цвете КК.

Выбор количества линий M влияет на:

1. Размер записи R (рис. 5, б).

2. Глубину упакованного дерева: чем больше M, тем меньше глубина.

3. Сложность алгоритма отображения текстуры: чем больше M, тем сложнее алгоритм отображения.

Глубину дерева на этапе упаковки ограничивают некоторым количеством уровней, которое выбирают исходя из требуемой точности. Следует учитывать, что в этом случае на последнем уровне дерева будет производиться отброс областей, если количество линий для КК превышает M.

Производится загрузка списка фигур из PostScript-файла. Для построения кватернарного дерева по списку фигур используется рекурсивная функция **PackTree1**. Функция принимает в качестве исходных параметров: координаты

и размеры КК, текущий уровень дерева, которому соответствует КК, ссылку на текстуру, для которой должно быть построено дерево (добавлено поддерево).

Данная функция предназначена для построения дерева (или поддерева), соответствующего данному КК. Она разбивает текущий КК на 4 дочерних, а затем для каждого из них:

а) выполняет функцию **PrepareSquare**;

б) в зависимости от признака необходимости деления КК помечает текущий КК либо как листовой, либо как нелистовой. В последнем случае производится рекурсивный вызов **PackTree1** для дальнейшего построения поддерева;

в) записывает дочерний КК в дерево.

Функция **PrepareSquare** предназначена для (1) определения делимости вершины дерева, соответствующей заданному КК и (2) заполнения записи *R*, характеризующей векторное изображение данного КК.

```
Признак_необх_деления_КК PrepareSquare(in КК, in Список_Фигур, out
Запись){
    for_each(Фигура из Список_Фигур){
        if(Фигура имеет общие части с КК){
            IntersectFigure(КК, Фигура);
            if(Фигура охватывает КК){
                Завершить_запись(Запись, Фигура);
                return «КК не требуется делить»;    }
            //Фигура пересекает КК несколькими линиями
            Внести данные о линиях, пересекающих КК, в Запись;
            if(Запись переполнена)
                return «КК подлежит делению»; } }
    }
```

На рис. 4, б показано разбиение исходного изображения на КК. Числитель дроби обозначает количество областей, знаменатель – количество линий, попавших в КК.

Было проведено моделирование, в результате которого исходное изображение, представляющее собой план аэропорта Борисполь (количество фигур 500), было упаковано данным алгоритмом. Количество уровней было выбрано равным 20, при этом размер упакованного файла текстуры составил 4 850 580 байт. Эквивалентная по точности и разрешению растровая текстура заняла бы, по меньшей мере, 10^{12} байт.

Упаковка векторного простого равномерного изображения

Упаковка данного изображения производится в два этапа. На первом этапе производится выбор количества N уровней дерева и формируется нижний уровень. Выбираем N таким образом, чтобы каждая ячейка получившейся матрицы содержала не более M линий. Нижний уровень дерева представляет собой матрицу размером 2^N на 2^N элементов. Каждый элемент матрицы содержит информацию о КК в формате, приведенном на рис. 5, б. Заполнение информации о КК (записи R) производится функцией **PrepareSquare**, описанной выше. После построения последнего уровня дерева назначаем интегральный цвет каждому КК на остальных уровнях.

Достоинства данного формата представления текстуры:

1. Объем памяти структуры меньший, чем при упаковке алгоритмом векторной простой неравномерной текстуры, за счет отсутствия ссылок.

2. Использование полного кватернарного дерева позволяет производить отображение текстуры более простым алгоритмом, с меньшей затратой времени.

Выводы. В результате исследований для каждого из описанных видов изображения текстуры предложены форматы и алгоритмы ее упаковки. Достоинства предложенных форматов текстур:

1. Обеспечивают компактность представления в памяти.

2. Сохраняют такое достоинство векторного представления, как высокое разрешение.

3. Применение композиционных текстур позволяет гибко комбинировать векторные и растровые текстуры.

4. Позволяют быстро отображать текстуры без перевода их в растровую форму с одновременным устранением алиайзинга.

В дальнейшем предполагается серия публикаций, в которых описываются методы отображения текстур с одновременным устранением алиайзинга.

Список литературы: **1**. *Гусятин В.М.* Алгоритм геометрических преобразований изображения в системах визуализации тренажеров транспортных средств / Авиационно-космическая техника и технология. Труды ХАИ им. Н.Е. Жуковского за 1997. – С. 467–471. **2.** *Williams L.* Pyramidal Parametrics, S1GGRAPH 83. – P. 1–11. **3.** *Heckbert P.S.* Filtering by Repeated Integration, SIGGRAPH 86. – P. 315–321. **4.** *Crow F.C.* Summed-Area Tables for Texture Mapping, SIGGRAPH 84. – P. 207–212. **5.** *Interrante V.* Harnessing natural textures for multivariate visualization, IEEE CG-M Nov-Dec 2000. – P. 6–11. **6.** *Gorla G., Interrante V., Sapiro G.* Texture synthesis for 3D shape representation, IEEE T-VCG, 2003. – P. 512–524. **7.** *Ying L., Hertzmann A., Biermann H., Zorin D.* Texture and Shape Synthesis on Surfaces // Proc. 12th Eurographics Workshop Rendering, 2001. **8.** *Samet H.* The quadtree and related hierarchical data structures. ACM Computing Surveys. – 16 (2), 1984. – P. 187–260. **9.** *Samet H., Webber R.E.* Hierarchical Data Structures and Algorithms for Computer Graphics, Part I: Fundamentals, CG&A, 1988. – P. 48–68. **10.** *Foley J.D., van Dam A., Feiner S.K., Hughes J.F.* Computer Graphics (principles and practice) by Addison-Wesley Publishing Company, Inc., 1996. – P. 1175

Поступила в редакцию 11.10.2005