УДК 004.92

***Чаговец Я.В., Дудник Б.А.***

# УПАКОВКА ВЕКТОРНЫХ ТЕКСТУР В ЗАДАЧАХ СИНТЕЗА ИЗОБРАЖЕНИЙ ДЛЯ СИСТЕМ ВИЗУАЛИЗАЦИИ

*Предложен формат и алгоритм упаковки текстуры на основе заполняющих пространство кривых (Гильберта, Мортона). Формат характеризуется высокой степенью упаковки за счёт свойства локализации указанных кривых и позволяет производить выборку текселей без распаковки. Предложен также алгоритм быстрой выборки текселей текстуры на основе бинарного поиска, что позволяет отображать текстуры с минимальными требованиями к объему памяти и затратами времени.*

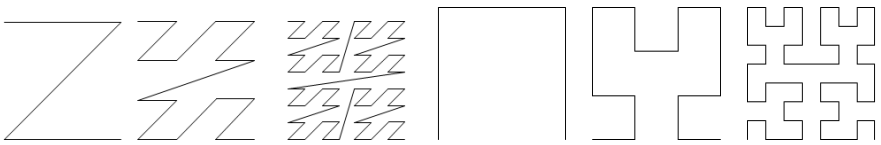
***Анализ литературы.*** В работах [2], [3], [5] описаны реализации алгоритмов сканирования поверхностей. В работе [4] описан процесс сжатия изображений на основе заполняющих пространство кривых. Областью использования данного сканирования в представленных работах являлись полутоновые монохромные изображения. Были представлены реализации построения кривой Гильберта рекурсивным алгоритмом. В [6] изложены принципы построения и использования данных кривых, а в [7] приведены усовершенствованные алгоритмы программной реализации генерации кривых. В [8] описываются существующие алгоритмы сжатия без потерь, используемые в медицинских изображениях. *В* анализируемой литературе не было описания сжатия векторных текстур с использованием генерации заполняющих пространство кривых и эффективных структур данных для хранения и работы с упакованными текстурами. В работах [1, 9, 10] рассмотрены методы упаковки векторных текстур, а также алгоритмы распаковки. Требуется дальнейшее увеличение эффективности упаковки и наложения текстур.

***Цель статьи.*** Разработка формата структуры данных для упакованных текстур, алгоритма упаковки и алгоритма доступа к пикселям текстуры. Также необходимо разработать формат файла упакованной текстуры. Полученные результаты проверить с помощью программной реализации.

## 1. Заполняющие пространство кривые

С помощью кривых, заполняющих пространство [2-8] представляется возможным сканирование изображения. Причём данный обход изображения обладает свойством локализации, в отличие от традиционного построчного сканирования, которое является крайне неэффективным из-за постоянной загрузки фрагментов одних и тех же объектов, которые сканировались в предыдущих строчках. Заполняющие пространство кривые Гильберта и Мортона (Z-кривая) (см. рис.1), обладают искомым

свойством локализации. Они рекурсивно сканируют квадрант за квадрантом и не делают длинных переходов из одного квадранта в другой.



а) б)

Рис.1 Заполняющие пространство кривые: а) кривая Мортона; б) кривая Гильберта.

Кривая Гильберта:

Основывается на делении каждой стороны единичного квадрата на четыре меньших квадрата. Затем каждый из четырех получившихся квадратов, в свою очередь, аналогично делится на четыре меньших квадрата и т.д. На каждой стадии такого деления строится кривая, которая обходит имеющиеся квадраты. Кривая Гильберта представляет собой предельную кривую, полученную в результате такого построения.

Z-кривая:

Важным свойством Z-кривой (кривой Мортона) является легкость ее построения. Если представить координаты X и Y в следующем виде: $X = \sum_{i=0}^{n-1} x_i 2^i$, $Y = \sum_{i=0}^{n-1} y_i 2^i$, то значение шага кривой определяется по формуле Z = $\sum_{i=0}^{n-1} (x_i + 2y_i) 2^{2i}$. Для подобного упорядочивания оказывается, что понятия "соседства" в смысле положения на кривой и "соседства" в обычном двумерном пространстве очень близки.

## 2. Исходные изображения текстур

Подробная классификация векторных текстур представлена в работе [9]. В рамках данной классификации разрабатываемый формат ориентирован на векторные простые равномерные и векторные простые неравномерные изображения. Формат рассчитан на текстуры большого объема, для которых целесообразно использовать другие способы представления в памяти, отличные от битовой матрицы.

### 2.1. Структура данных упакованной текстуры

Общая структура данных представляет собой следующее:

1. Заголовок (header). Хранит размер текстуры в текселях и тип кривой, использовавшийся при упаковке.
2. Палитра цветов. Таблица, адресом в которой является индекс, а значением – цвет, соответствующий данному индексу.
3. Блок данных текстуры. Представляет собой последовательность данных по отрезкам, записанным в порядке обхода изображения по соответствующей кривой (Гильберта или Мортона). Последовательность отрезков представляется как D0, D1, …, Dn. Здесь блок данных Di представляет собой кортеж Di = <Si, Pi>, где Si –

номер шага вдоль кривой, соответствующий последнему текселю i-го отрезка, Pi – индекс цвета для i-го отрезка.

Проиллюстрируем структуру данных на примере. Для этого воспользуемся изображением 8x8, в котором используется 6 различных цветов:

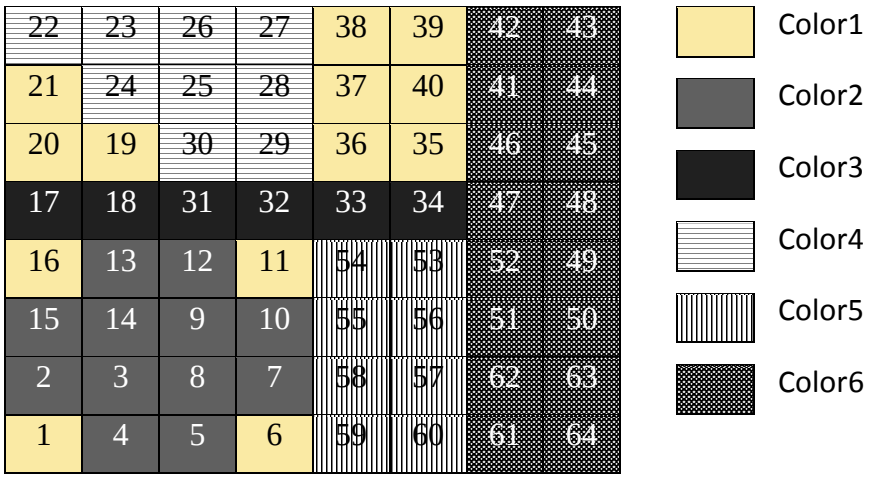



Рис. 2 Пример текстуры. Порядок сканирования кривой Гильберта обозначен числами (номера шагов).

В результате упакованная структура будет иметь вид:

Таблица 1 - Заголовок упакованной текстуры

| Width | Height | Curve type |
|---|---|---|
| 8 | 8 | Hilbert |

Таблица 2 - Палитра цветов (P – индекс цвета)

| P | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Color | color1 | color2 | color3 | color4 | color5 | color6 |

Таблица 3 - Пример упакованной текстуры (S – номер шага, P–индекс цвета)

| S | 1 | 5 | 6 | 10 | 11 | 15 | 16 | 18 | 21 | 30 | 34 | 40 | 52 | 60 | 64 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| P | 1 | 2 | 1 | 2 | 1 | 2 | 1 | 3 | 1 | 4 | 3 | 1 | 5 | 6 | 5 |

В памяти индекс цвета представлен 1 байтом, номер шага кривой – 4 байтами, что позволяет индексировать в 2-х мерном пространстве точки в пределах 65536x65536 пикселей.

### 2.2. Алгоритм упаковки текстур

Вначале производится анализ текстуры и приведение её в случае необходимости к требуемому типу (256 цветов, 24 – битный цвет). Затем текстура сканируется

выбранной кривой обхода и анализируется линейная развертка изображения. Она представляет собой последовательности (от 1 и больше) пикселей одинакового цвета. Каждая такой отрезок записывается в памяти парой: последний пиксель отрезка (номер шага кривой) и индекс цвета.

Упаковка текстуры в необходимую структуру данных происходит следующим образом:

1. Запись заголовка файла;
2. Создание палитры;
3. Формирование блока данных.

Алгоритм формирования блока данных описывается псевдокодом:

```
void Convert (in входная_текстура, out блок_данных)
{
   текущий_цвет = входная_текстура ->цвет_пикселя (0,0);
   unsigned номер_шага = 1;
   do
   {
      номер_шага->(x,y);
      if ((x,y) в пределах текстуры)
      {
         количество_обработанных_пикселей++;
         следующий_цвет = входная_текстура ->цвет_пикселя(x,y);
         if (следующий_цвет != текущий_цвет)
         {
            блок_данных->добавить_след_отрезок(текущий_цвет, номер_шага);
            текущий_цвет = следующий_цвет;
          }
      }
      номер_шага++;
   }while(все_пиксели_не_обработаны);
   блок_данных->Добавить_след_отрезок(текущий_цвет, номер_шага);
}
```

### 2.3. Алгоритм доступа к текселям в упакованной текстуре

Данная структура данных позволяет быстро определить цвет необходимого нам текселя. Так как Si > Si-1, то для получения цвета текселя есть возможность воспользоваться бинарным поиском. Это происходит следующим образом: по известным координатам искомого текселя определяется шаг кривой обхода (x,y) -> S, затем бинарным поиском определяется номер отрезка, на котором расположен тексель и берётся соответствующий данному отрезку цвет.

Доступ к текселям текстуры:

```
Цвет getColor(in x, in y)//координаты текселя
{
```

```
    номер_шага = Получить_номер_шага_по_координатам(x,y);
    индекс_цвета = BinarySearch(блок_данных, номер_шага);
    цвет = палитра-> цвет (индекс_цвета);
    return цвет;
}
```

Таким образом, при получении цвета текселя по заданным координатам происходит преобразование последних в соответствующий номер шага используемой кривой, затем с помощью бинарного поиска определяется, какому из отрезков принадлежит тексель с заданными координатами. В конечном итоге, когда искомый отрезок найден, возвращается соответствующий ему цвет.

Возможность бинарного поиска в блоке данных упакованной текстуры позволяет очень быстро определять цвет искомого текселя.

### 2.4. Формат файла

Формат упакованной текстуры, сохраняемой в файле, немного отличается от того, что находится в памяти при непосредственном использовании. Вместо значения шага кривой в файл пишется длина отрезка. Количество байт для записи длины отрезка зависит от величины последнего. Есть 7 значимых бит и 8-й для определения окончания числа (если «1» – конец числа, если «0» – продолжить чтение в следующем байте). Это сделано для большего сжатия текстуры, сохранённой в файл.

### 2.5. Реализация алгоритма

Для проверки предложенных концепций была создана программная реализация описанных алгоритмов на языке программирования C++ в среде разработки MS Visual Studio, с использованием Qt Framework. Также были созданы вспомогательные утилиты для визуализации и тестирования.

В созданной реализации производится упаковка текстур, цвета которых можно представить палитрой из 256 элементов (индекс составляет 1 байт) с глубиной цвета 24 бита. В ином случае сначала изображение будет преобразовано в данный формат, и затем будет произведена упаковка. Максимальный размер упаковываемой текстуры составляет 65536x65536 текселей.

### 2.6. Сравнение с аналогами

Текстура в упакованном формате удовлетворяет требованиям малого объёма, занимаемого в памяти и быстрого доступа к произвольным текселям текстуры. Большинство других форматов после чтения из сохранённого файла изображения представляются в памяти битовой матрицей.

В качестве сравнения с существующими форматами был выбран формат [10] на основе неполного кватернарного дерева. Данный формат был выбран, поскольку рассчитан на решение идентичных задач упаковки векторных текстур большого объёма. Сравнение производилось по следующим критериям: размер текстуры, сохраненной на диске, время доступа к произвольным текселям текстуры.

Для сравнения были выбраны две различные текстуры аэропорта, обе размером 8192x8192 текселей.

Таблица 4 – Результаты сравнительного анализа

| Исходные данные | Критерий | Формат на основе заполняющих пространство кривых | Формат на основе неполного кватернарного дерева |
|---|---|---|---|
| Текстура №1 | Размер упакованной текстуры, байт | 330870 | 1314065 |
| | Среднее время доступа к произвольному текселю, мкс | 6.7670 | 43.1789 |
| Текстура №2 | Размер упакованной текстуры, байт | 78582 | 407669 |
| | Среднее время доступа к произвольному текселю, мкс | 4.6080 | 35.4988 |

## Выводы

Были предложены формат и алгоритм для упаковки векторных текстур, а также алгоритм доступа к её элементам. За счёт свойства локализации заполняющих пространство кривых удалось достичь высоких степеней сжатия. Разработанная структура данных упакованной текстуры даёт возможность быстрого доступа к произвольным текселям без распаковки.

Было произведено тестирование и сравнительный анализ с существующим форматом, ориентированным на такой же класс изображений. По основным критериям, а именно: среднему времени доступа к текселям упакованной текстуры и размеру, занимаемому на диске, предложенный формат показал лучшие показатели, чем существующий. Однако следует отметить, что в предложенном формате отсутствует возможность устранения алиайзинга, в отличие от формата, работающего на основе кватернарного дерева, с которым производилось сравнение.

В дальнейшем планируется усовершенствование формата и алгоритма с возможностью устранения 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. L. Velho, J. Gomez. Digital Halftoning with Space Filling Curves.Computer Graphics, Volume 25, Number 4, July 1991.
3. J. K. Lawder and P. J. King.Using Space-filling Curves for Multi-dimensional indexing. Lecture Notes in Computer Science, 2000, Volume 1832/2000, p. 20-35.

4. S. Biswas. Hilbert Scan and Image Compression. Pattern Recognition, 2000. Proceedings. 15th International Conference. **p**. 207 - 210 Volume 3
5. A. J. Cole. Compaction Technique for Raster Scan Graphics Using Space-filling Cureves. Computer journal, 30:87-92, 1987.
6. D. Hilbert: Über die stetige Abbildung einer Linie auf ein Flächenstück. Mathematische Annalen 38 (1891), 459–460.
7. H. Warren. Hakker's Delight. Addison Wesley, p. 284, 2004.
8. JY Liang, CS Chen, CH Huang. Lossless Compression of Medical Images Using Hilbert Space-Filling Curves. **Computerized Medical Imaging and Graphics**. Volume 32, Issue 3 , Pages 174-182, April 2008.
9. Гусятин В.М., Чаговец Я.В., Кожушко Д.Г. Упаковка векторных текстур в задачах синтеза изображений для систем визуализации. // Вісник Національного технічного університету "ХПІ". Збірник наукових праць. Тематичний випуск: Інформатика і моделювання. -Харків: НТУ "ХПІ". - 2005. -№56. С.9-16
10. Гусятин В.М., Чаговец Я.В, Кожушко Д.Г. Метод расчёта цвета проекции пикселя в задачах нанесения текстур, представленных неполным кватернарным деревом. // Радіоелектронні та комп'ютерні системи. – Харків, „ХАІ", 2008. – Вип.2(29). – С.59-62.