УДК 004.932.2+004.932.72'1

И. А. Бобров, Н.Е. Губенко
Донецкий национальный технический университет, г. Донецк
кафедра компьютерных систем мониторинга

АНАЛИЗ СТРУКТУРЫ БИНАРНОГО ФАЙЛА 3DS ДЛЯ ТРЕХМЕРНЫХ ПРИЛОЖЕНИЙ

Аннотация

Бобров И.А., Губенко Н.Е. Анализ структуры бинарного файла 3DS. *Выполнен анализ структуры бинарного файла 3DS, который хранит 3d сцену целиком, сохраняя при этом данные об объектах, источниках света, камерах, и применненных материалах.*

Ключевые слова: *3d объект, координаты, нормали, текстурные координаты.*

Постановка проблемы. При создании 3д приложений используются трехмерные объекты, созданные в крупных пакетах трехмерного моделирования – например 3D Studio Max. Экспортируя трехмерный объект из пакета трехмерного моделирования, получаем файл, который хранит геометрические данные, текстурные координаты, источники света и т.д. в зависимости от настроек экспорта и самого формата файла.

Самым распространенным, но и самым сложно структурированным является формат 3DS. В отличии от других форматов, он хранит данные об 3d объектах, текстурных координатах, источниках света, камерах, материалах и текстур.

Цель статьи – провести анализ структуры бинарного файла 3DS, для создания 3д приложений. Выделить недостатки и преимущества.

Решение задач и результаты исследований. Для создания 3d приложений необходимо иметь данные о координатах точек, из которых состоят объекты, о свете и о камерах, использованных в сцене. Проведенный анализ структуры 3DS файла позволяет сделать вывод о том, что все необходимые данные для создания 3d приложений можно получить из этого файла**.**

Все данные, хранящиеся в формате 3DS, расположены в чанках(chank). Чанк – это некоторый блок данных. Чанк имеет нелинейную структуру, это значит, что элементы, находящиеся по структуре внутри других, читаются только после того, как будут прочитаны (будет получен к ним доступ) элементы их содержащие. Структура 3DS формата имеет вид дерева.

Каждый чанк состоит из 4х полей:

- - идентификатор - это число, определяющее тип чанка, записанное в шестнадцатеричном формате, размером 2 байта;
- - размер чанка - число размером 4 байта;
- - данные чанка – данные хранимые в чанке, это поле может иметь разную длину;
- - подблоки – неопределенное количество чанков, для которых данный чанк является родителем, это поле может иметь разную длину;

Структура чанка представлена на табл.1.

Таблица 1 – Структура чанка

Отступ	**Длинна**	**Описание**
0	2	Идентификатор
2	4	Размер чанка (2+4+n+m)
6	n	Данные чанка
6+n	m	Подблоки

Иерархия и код наиболее полезных чанков в файле:

```
MAIN CHUNK 0x4D4D
 3D EDITOR CHUNK 0x3D3D
  OBJECT BLOCK 0x4000
   TRIANGULAR MESH 0x4100
    VERTICES LIST 0x4110
    FACES DESCRIPTION 0x4120
     FACES MATERIAL 0x4130
    MAPPING COORDINATES LIST 0x4140
     SMOOTHING GROUP LIST 0x4150
    LOCAL COORDINATES SYSTEM 0x4160
   LIGHT 0x4600
    SPOTLIGHT 0x4610
   CAMERA 0x4700
  MATERIAL BLOCK 0xAFFF
   MATERIAL NAME 0xA000
    AMBIENT COLOR 0xA010
    DIFFUSE COLOR 0xA020
    SPECULAR COLOR 0xA030
    TEXTURE MAP 1 0xA200
    BUMP MAP 0xA230
   REFLECTION MAP 0xA220
   [SUB CHUNKS FOR EACH MAP]
    MAPPING FILENAME 0xA300
    MAPPING PARAMETERS 0xA351
```

Рисунок 1 – Иерархическая структура 3DS формата

Для чтения 3d объекта необходимо:

1. **Чтение блока VERTICES LIST.** Этот чанк содержит массив триад координат точек, из которых состоит объект. Каждая триада имеет тип float и занимает 4 байта. На табл.2 представлена структура чанка VERTICES LIST, а на рис.2 программный код чтения данного блока.

Таблица 2 – Структура блока VERTICES LIST

VERTICES LIST	
Идентификатор	0x4110
Размер чанка	Варьируется
Родитель чанка	TRIANGULAR MESH
Подблоки	Нет
Данные чанка	Количество точек (unsigned short) Массив триад координат. (3*float)

```
case 0x4110:
l_qty = binr.ReadUInt16();
float[] Verts = new float[l_qty * 3];
for (i = 0; i < l_qty; i++)
{
    Verts[i * 3] = binr.ReadSingle();
    Verts[i * 3 + 2] = binr.ReadSingle();
    Verts[i * 3 + 1] = binr.ReadSingle();
}
    break;
```

Рисунок 2 – Чтение блока VERTICES LIST

2. Чтение блока FACES DESCRIPTION. Этот чанк содержит массив-дескриптор, в котором содержатся номера точек из блока VERTICES LIST в том порядке, в котором состоят полигоны этого объекта.

На табл.3 представлена структура чанка FACES DESCRIPTION, а на рис.3 программный код чтения данного блока.

Таблица 3 – Структура блока FACES DESCRIPTION

FACES DESCRIPTION	
Идентификатор	0x4120
Размер чанка	Варьируется
Родитель чанка	TRIANGULAR MESH
Подблоки	FACES MATERIAL
Данные чанка	Количество полигонов (unsigned short) Массив триад точек (3* unsigned short)

```
case 0x4120:
l_qty = binr.ReadUInt16();
float[] Verts _face = new float[l_qty * 3];
for (i = 0; i < l_qty; i++)
{
Verts _face [i * 3] = binr.ReadSingle();
Verts _face [i * 3 + 2] = binr.ReadSingle();
Verts _face [i * 3 + 1] = binr.ReadSingle();
}
    break;
```

Рисунок 3 – Чтение блока FACES DESCRIPTION

3. Чтение блока MAPPING COORDINATES LIST. Этот чанк содержит массив из 2х текстурных координат для всех точек объекта. Каждая текстурная координата имеет тип float и занимает 4 байта.

На табл.4 представлена структура чанка VERTICES LIST, а на рис.4 программный код чтения данного блока.

Таблица 4 – Структура блока MAPPING COORDINATES LIST

MAPPING COORDINATES LIST	
Идентификатор	0x4140
Размер чанка	Варьируется
Родитель чанка	SMOOTHING GROUP LIST
Подблоки	Нет
Данные чанка	Количество точек (unsigned short) Массив текстурных координат. (2*float)

```
case 0x4140:
l_qty = binr.ReadUInt16();
float[] MapCoord = new float[l_qty * 2];
for (i = 0; i < l_qty; i++)
{
MapCoord[i * 2] = binr.ReadSingle();
MapCoord[i * 2 + 1] = binr.ReadSingle();
}
    break;
```

Рисунок 4 – Чтение блока MAPPING COORDINATES LIST

4. Расчет нормалей. Файл 3DS не содержит информацию о нормалях объекта, их необходимо рассчитывать самостоятельно. Это является минусом данного формата.

Вектор нормали - это вектор единичной длины, перпендикулярный к поверхности в данной точке этой поверхности. Вектор нормали имеет три проекции на оси координат X,Y,Z, которые называют координатами вектора. Координаты вектора нормали присваиваются каждой вершине отдельно и являются её атрибутом. Вектор нормали очень важен для правильного расчета освещения поверхности, т.к. определяет направление отражённого света.

Для расчета нормали необходимо иметь три точки на поверхности. Из трёх точек можно составить два вектора, которые всегда будут лежать в одной плоскости. Проведем из точки 1 в точку 2 вектор A, из точки 1 в точку 3 вектор B. Вектор N, перпендикулярный A и B рассчитывается как векторное произведение $N = [AxB]$:

$$Nx = Ay * Bz - By * Az = (y2 - y1) * (z3 - z1) - (y3 - y1) * (z2 - z1)$$
$$Ny = Bx * Az - Ax * Bz = (x3 - x1) * (z2 - z1) - (x2 - x1) * (z3 - z1)$$
$$Nz = Ax * By - Bx * Ay = (x2 - x1) * (y3 - y1) - (x3 - x1) * (y2 - y1) \quad (1)$$

Далее нужно вычислить длину вектора N и его координаты Nx,Ny,Nz поделить на длину, т.е. нормализовать вектор N.

Чтение координат источников света(LIGHT). Данный чанк содержит информацию о источнике света.

На табл.5 представлена структура чанка LIGHT, а на рис.5 программный код чтения данного блока.

Таблица 5 – Структура блока LIGHT

LIGHT	
Идентификатор	0x4600
Размер чанка	Варьируется
Родитель чанка	EDIT OBJECT
Подблоки	Дополнительная информация
Данные чанка	Координаты положения. (3 float)

```
case 0x4600:
float x = binr.ReadSingle();
float y = binr.ReadSingle();
float z = binr.ReadSingle();
    break;
```

Рисунок 5 – Чтение блока LIGHT

Чтение блока CAMERA. Данный чанк содержит информацию о камере. На табл.6 представлена структура чанка CAMERA, а на рис.6 программный код чтения данного блока.

Таблица 6 – Структура блока CAMERA

CAMERA	
Идентификатор	0x4700
Размер чанка	Варьируется
Родитель чанка	EDIT OBJECT
Подблоки	Нет
Данные чанка	Координаты положения. (3 float) Координаты направления. (3 float) Угол поворота. (float) Коэффициент зуммирования. (float)

```
case 0x4700:
    float pX = binr.ReadSingle();
    float pY = binr.ReadSingle();
    float pZ = binr.ReadSingle();
    float tX = binr.ReadSingle();
    float tY = binr.ReadSingle();
    float tZ = binr.ReadSingle();
    float bank = binr.ReadSingle();
    float lens = binr.ReadSingle();
    break;
```

Рисунок 6 – Чтение блока CAMERA

Чтение блока MATERIAL BLOCK. Данный чанк содержит информацию о материале: имя материала, диффузный цвет, цвет отражения, цвет преломления, текстуры.

На табл.7 представлена структура чанка MATERIAL BLOCK, а на рис.7 программный код чтения данного блока.

Таблица 7 – Структура блока MATERIAL BLOCK

MATERIAL BLOCK	
Идентификатор	0xAFFF
Размер чанка	Варьируется
Родитель чанка	EDIT OBJECT
Подблоки	MATERIAL NAME 0xA000 AMBIENT COLOR 0xA010 DIFFUSE COLOR 0xA020 SPECULAR COLOR 0xA030 TEXTURE MAP 1 0xA200 BUMP MAP 0xA230 REFLECTION MAP 0xA220
Данные чанка	Нет

```
case 0xA000:
String str_name = ReadString(binr, 20);
read.GetMaterials().Add(new Material());
read.GetCurrentMaterials().SetName(str_name);
break;
case 0xA010:
MatType = "Ambient";
read.GetCurrentMaterials().AddMap(MatType, new
MaterialOptions());
break;
case 0xA020:
MatType = "Diffuse";
read.GetCurrentMaterials().AddMap(MatType, new
MaterialOptions());
break;
case 0xA030:
MatType = "Specullar";
read.GetCurrentMaterials().AddMap(MatType, new
MaterialOptions());
break;
case COL_RGB:
float ar = binr.ReadSingle();
float ag = binr.ReadSingle();
float ab = binr.ReadSingle();
read.GetCurrentMaterials().GetMapOption(MatType).Set
24bit(ar, ag, ab);
break;
case COL_TRU:
int ar2 = binr.ReadByte();
int ag2 = binr.ReadByte();
int ab2 = binr.ReadByte();
read.GetCurrentMaterials().GetMapOption(MatType).Set
RGB(ar2, ag2, ab2);
break;
```

Рисунок 7 – Чтение блока MATERIAL BLOCK

Была разработана программа для считывания данных об объектах, освещении, камерах и материалах с 3DS файла. Диаграмма вариантов использования отображена на рис.8.

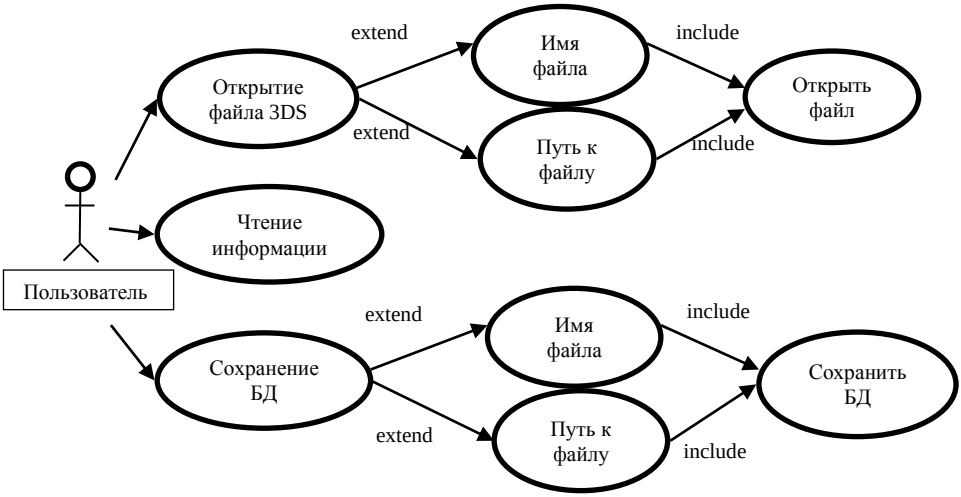

Рисунок 8 – Диаграмма вариантов использования

Выводы. Проведен анализ структуры бинарного файла 3DS с целью использования сложных 3d моделей в графических приложениях. Выявлены следующие преимущества: большое количество данных о сцене и структурированность этих данных и недостатки: нет информации о нормалях объекта, их необходимо рассчитывать и отсутствие полной документации формата. Была написана программа позволяющая считать все данные о 3d сцене и сохранить эти данные в базу данных для дальнейшего использования в 3d приложениях.

Список литературы

1. Разработка 3d приложений Android OpenGL ES./ Интерет-ресурс. – http://andmonahov.blogspot.com/2012/04/opengl-es.html /.3ds- Загл. с экрана.
2. Формат 3d studio mesh, взгляд изнутри./ Интернет-ресурс. - http://www.render.ru/books/show_book.php?book_id=470- Загл. с экрана.
3. 3D-Studio File Format (.3ds) / Интернет-ресурс. - http://paulbourke.net/dataformats/3ds/ – Загл. с экрана.
4. 3D Studio File Format Information/ Интерет-ресурс. – http://cse.csusb.edu/tong/courses/cs520/notes/3DSINFO.TXT- Загл. с экрана.
5. 3DS Max file/ Интерет-ресурс. – http://en.wikipedia.org/wiki/.3ds- Загл. с экрана.
6. Создание 3ds loader'a. / Интерет-ресурс. – http://savardge.narod.ru/bin/article_bin1.html /.3ds- Загл. с экрана.
7. Создание 3ds loader'a. / Интерет-ресурс. – http://savardge.narod.ru/bin/article_bin1.html /.3ds- Загл. с экрана.