**Описание структуры GIF файла, работы алгоритма класса GifEncoder**

В классе GifEncoder описаны константы, которые указывают на соответствующие параметры. Например, константы FileType и FileVersion указывают на формат файла и его версию для идентификации приложениями этого файла как Gif. Другие же константы указывают на длины и позиции блоков байтов (листинг 1).

//Установка хэдера и констант для GIF файла

#region Header Constants

private const string FileType = "GIF";

private const string FileVersion = "89a";

private const byte FileTrailer = 0x3b;

private const int ApplicationExtensionBlockIdentifier = 0xff21;

private const byte ApplicationBlockSize = 0x0b;

private const string ApplicationIdentification = "NETSCAPE2.0";

private const int GraphicControlExtensionBlockIdentifier = 0xf921;

private const byte GraphicControlExtensionBlockSize = 0x04;

private const long SourceGlobalColorInfoPosition = 10;

private const long SourceGraphicControlExtensionPosition = 781;

private const long SourceGraphicControlExtensionLength = 8;

private const long SourceImageBlockPosition = 789;

private const long SourceImageBlockHeaderLength = 11;

private const long SourceColorBlockPosition = 13;

private const long SourceColorBlockLength = 768;

#endregion

Листинг 1 – Константы

Формат графического обмена (Graphics Interchange Format, GIF) представляет собой формат растрового изображения. GIF поддерживает до 8 бит на пиксель для каждого изображения, что позволяет изображению ссылаться на собственную палитру до 256 различных цветов, выбранных из 24-битного пространства RGB. Что немаловажно, GIF поддерживает анимацию и её цикличное воспроизведение, а также прозрачность, использование которой позволяет не сохранять очередной кадр целиком, а только изменения относительно предыдущего. Изображения в GIF сжимаются с помощью LZW-компрессии.

GIF состоит из фиксированной области в начале файла, после которой располагается переменное число блоков, и заканчивается файл завершителем изображения (рисунок 1).



Рисунок - Структура файла gif

В начале каждого файла GIF находится заголовок, который состоит из 6 байт и содержит в себе текст «GIF87a» или «GIF89a», в зависимости от версии (рисунок 2).

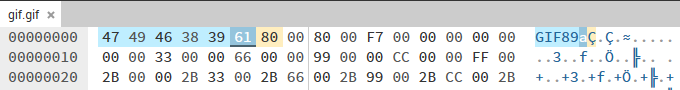


Рисунок – Записанные данные о версии файла Gif

После заголовка идёт логический дескриптор экрана, который включает в себя данные о ширине и высоте отображаемого экрана в пикселях, информацию об экране и цветовой таблице, индекс цвета фона и коэффициент сжатия пикселов (рисунок 3).



Рисунок – Дескриптор глобального экрана

Ширина и высота экрана представлены первыми 4 байтами. 5 байт является служебным и содержит в себе следующие значения:

* Pixel – занимает первые 3 бита и определяет число цветов в изображении;
* бит 3 является служебным со значением по умолчанию 0;
* Cr – занимает 4 по 6 биты и определяет битовую глубину цвета – 1;
* M – флаг, определяющий существует глобальная цветовая таблица или нет.

6 байт выделяется под цвет фона, а 7 определяет соотношение сторон в пикселях (по умолчанию равен нулю и соотношение равно 1:1).

На рисунке 4 выделены 4 байта, содержащие данные о высоте и ширине экрана, а также значение 5-го служебного байта.

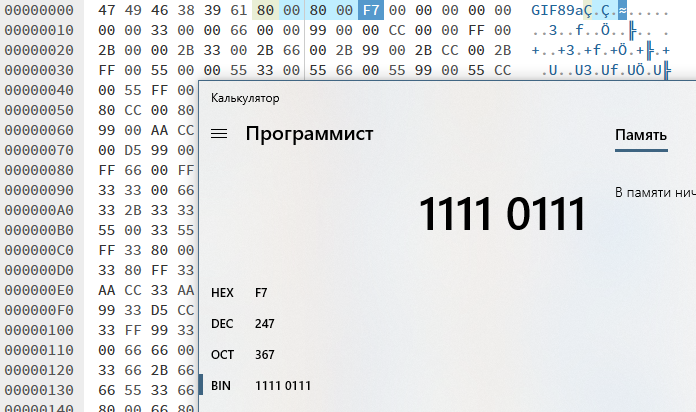


Рисунок – HEX представление дескриптора служебного экрана

После дескриптора глобального экрана идёт глобальная цветовая таблица, если таковая определена в предыдущем блоке (рисунок 5). Цвета в ней представлены в формате RGB, где на каждый оттенок отводится 1 байт. В максимуме цветовая таблица может располагаться с 13 и до 778 байта. Её запись происходит при добавлении первого изображения через метод WriteColorTable (листинг 2).



Рисунок – Глобальная цветовая таблица

//Данный код читает таблицу цветов из исходного файла GIF и записывает ее в выходной поток в текущей позиции

private void WriteColorTable(Stream sourceGif)

{

//Устанавливаем позицию записи таблицы цветов

sourceGif.Position = SourceColorBlockPosition;

//Устанавливаем размер таблицы цветов

var colorTable = new byte[SourceColorBlockLength];

//Считываем из исходного файла в массив

sourceGif.Read(colorTable, 0, colorTable.Length);

//Записываем в файл таблицу цветов

stream.Write(colorTable, 0, colorTable.Length);

}

Листинг 2 – Метод записи таблицы цветов в файл

После заголовка, дескриптора глобального экрана и глобальной цветовой таблицы идёт переменная часть gif. Каждое изображение в файл записывается отдельно со своими блоками расширений, дескрипторами изображения и цветовой таблицей (рисунок 6).



Рисунок – Структура изображений в GIF

Блоки расширения представляют собой необязательные блоки, позволяющие настраивать данные изображения (рисунок 7). Расширения состоят из:

1. Начало расширения – байт 0x21, обозначающий начало расширения.
2. Код расширения – содержит информацию о типе расширения:

* 0x1 – расширение простого текста;
* 0xF9 – расширение управления графикой;
* 0xFE – расширение комментария;
* 0xFF – расширение программы.

В основном используются 2 типа расширений: программы и управления графикой.

1. Заголовок – байт, содержащий информацию о размере следующего блока данных в байтах.



Рисунок – Структура блоков расширений

Расширение программы содержит следующие блоки данных и их значение:

* 0x0B – размер блока в байтах;
* первые 7 байт – (NETSCAPE) идентификатор приложения, которому принадлежит это расширение;
* следующие 3 байта – (2.0) код приложения. С его помощью приложение проверяет, действительно ли это расширение принадлежит ему.
* 0x03 – размер блока в байтах;
* 0x01 – фиксированное значение;
* 0x[] 0x[] – значение 0..65535. Беззнаковое целое в формате little-endian. Определяет, сколько раз должен повторяться цикл. Если 0 – бесконечно;
* 0x00 – конец расширения.

Вид в HEX редакторе этих блоков представлен на рисунке 8.



Рисунок – Представление блока расширения

Метод для записи всего сказанного выше называется InitHeader и вызывается единожды в том случае, если изображение является первым (листинг 3).

//Если кадра является первым, то мы должны внести в файл базовые метаданные GIF файла

private void InitHeader(Stream sourceGif, int w, int h)

{

//Вносим информацию о типе и версии файла

WriteString(FileType);

WriteString(FileVersion);

//Устанавливаем данные о ширине и высоте отображаемого экрана в пикселях

WriteShort(width.GetValueOrDefault(w));

WriteShort(height.GetValueOrDefault(h));

// Устанавливаем позицию в исходном файле GIF для чтения информации о глобальной цветовой палитре

sourceGif.Position = SourceGlobalColorInfoPosition;

// Записываем служебный 5й байт

WriteByte(sourceGif.ReadByte());

//Зарезервированные байты

WriteByte(0);

WriteByte(0);

//Записываем таблицу цветов

WriteColorTable(sourceGif);

//Запись блока расширения в таблицу

WriteShort(ApplicationExtensionBlockIdentifier);

WriteByte(ApplicationBlockSize);

//Запись идентификатора приложения, которому принадлежит это расширение

WriteString(ApplicationIdentification);

//Размер блока в байтах

WriteByte(3);

//Фиксированное значение

WriteByte(1);

//Записываем количество повторений

WriteShort(repeatCount.GetValueOrDefault(0));

//Конец расширения

WriteByte(0);

isFirstImage = false;

}

 Листинг 3 – Метод для инициализации первых блоков файла

Расширение управления графикой содержит один блок данных, состоящий из:

* 0x04 – размер следующего блока данных;
* Служебный байт (рисунок 9), где младший бит указывает на то, будет ли какой-либо цвет использоваться как прозрачный, бит флага ввода указывает на то, требуется ли реакция пользователя на продолжения обработки изображения, последние 3 бита зарезервированы и равны нулю. Метод обработки определяет, что делать после отображения:

а) 000 – к картинке не будет применяться никакой обработки;

б) 001 – картинка останется без изменений;

в) 010 – картинка затрется фоном;

г) 011 – восстановится изображение под картинкой;

д) 100 -111 – не определены.

* 0x[] 0x[] – отводятся под время задержки в анимации в мс;
* 0x[] – индекс цвета прозрачности;
* 0x0 – конец расширения.

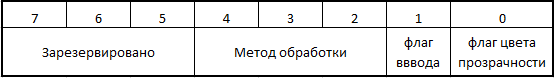


Рисунок – Структура служебного байта расширения управления графикой

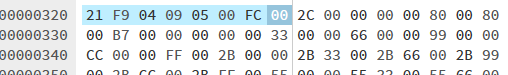


Рисунок – HEX вид блока управления графикой

Служебный байт равен 9 (в двоичном виде 0000 1001), а значит флаг цвета прозрачности равен 1, флаг ввода 0, метод обработки 010 (картинка затрётся фоном).

//Данный код читает блок графического управления из исходного файла GIF и записывает его в

//выходной поток в соответствии с определенной структурой и форматом блока графического управления GIF

private void WriteGraphicControlBlock(Stream sourceGif, TimeSpan frameDelay)

{

//Устанавливаем позицию записи блока графического управления

sourceGif.Position = SourceGraphicControlExtensionPosition;

//Считываем заголовок блока графического управления

var blockhead = new byte[SourceGraphicControlExtensionLength];

sourceGif.Read(blockhead, 0, blockhead.Length);

//Записываем данные о коде расширения

WriteShort(GraphicControlExtensionBlockIdentifier);

//Записываем размер блока

WriteByte(GraphicControlExtensionBlockSize);

//Записываем служебный байт

WriteByte(blockhead[3] & 0xf7 | 0x08);

//Запись задержки кадра в милисекундах

WriteShort(Convert.ToInt32(frameDelay.TotalMilliseconds / 10));

//Записываем индекс цвета прозрачности

WriteByte(blockhead[6]);

WriteByte(0);

}

 Листинг 4 – Метод для записи блока управления графикой

После блока расширений идёт блок данных изображения (рисунок 11)



Рисунок – Структура блока данных изображения

В заголовок входят следующие байты:

* первые 4 байта, определяющие координаты верхнего левого угла логического экрана (по умолчанию 0, 0);
* следующие 4 байта определяют ширину и высоту изображения в пикселях;
* служебный байт, в котором определяются флаги, указывающие на присутствие локальной таблицы для поиска цветов и определения последовательности высвечивания пикселов: флаг локальной цветовой таблицы (бит 0), флаг чередования (бит 1), флаг сортировки (бит 2), размер элемента локальной цветовой таблицы (биты 5 - 7). Биты 3 - 4 зарезервированы;
* 0x03 – минимальный размер кода в LZW;
* 0x[] – размер следующего блока в байтах;
* блок данных, сжатых алгоритмом LZW;
* 0x00 – окончание блока данных.



Рисунок – HEX представление заголовка блока

Для записи блока изображения используется метод WriteImageBlock, в который передаётся исходное изображение, флаг наличия встроенной цветовой, позицию изображения и его размеры (листинг 5).

//Данный код читает заголовок и данные блока изображения из исходного файла GIF и записывает их в

//выходной поток в соответствии с определенной структурой и форматом блока изображения GIF.

private void WriteImageBlock(Stream sourceGif, bool includeColorTable, int x, int y, int h, int w)

{

//Устанавливаем позицию записи блока изображения

sourceGif.Position = SourceImageBlockPosition;

//Считываем заголовок блока изображения

var header = new byte[SourceImageBlockHeaderLength];

sourceGif.Read(header, 0, header.Length);

//Записываем хэдер блока (в данном случае его обозначение 0x2C)

WriteByte(header[0]);

//Координаты относительно экрана

WriteShort(x);

WriteShort(y);

//Высоту и ширину изображения

WriteShort(h);

WriteShort(w);

//Если это не первое изображение, то включаем сюда цветовую палитру изображения

if (includeColorTable)

{

//Устанавливаем позицию цветовой палитры для чтения её из исходного изображения

sourceGif.Position = SourceGlobalColorInfoPosition;

//Записываем служебный байт

WriteByte(sourceGif.ReadByte() & 0x3f | 0x80);

//Записываем цветовую таблицу

WriteColorTable(sourceGif);

}

else

//Записываем служебный байт

WriteByte(header[9] & 0x07 | 0x07);

WriteByte(header[10]);

sourceGif.Position = SourceImageBlockPosition + SourceImageBlockHeaderLength;

//Записываем данные сжатые алгоритмом LZW из исходного изображения

var dataLength = sourceGif.ReadByte();

while (dataLength > 0)

{

var imgData = new byte[dataLength];

sourceGif.Read(imgData, 0, dataLength);

//Записываем размер следующего блока

stream.WriteByte(Convert.ToByte(dataLength));

//Записываем данные о изображении в поток

stream.Write(imgData, 0, dataLength);

//Считываем новый размер блока изображения

dataLength = sourceGif.ReadByte();

}

//Завершаем блок изображения

stream.WriteByte(0);

}

 Листинг 5 – Метод для записи данных блока изображения

Для удобства использования были добавлены методы для записи байтов, 2 байтов и строки в поток (листинг 6).

//Записывает в поток байт

private void WriteByte(int value)

{

stream.WriteByte(Convert.ToByte(value));

}

//Записывает в поток 2 байта

private void WriteShort(int value)

{

stream.WriteByte(Convert.ToByte(value & 0xff));

stream.WriteByte(Convert.ToByte((value >> 8) & 0xff));

}

//Записывает строку в поток

private void WriteString(string value)

{

stream.Write(value.ToArray().Select(c => (byte)c).ToArray(), 0, value.Length);

}

 Листинг 6 – Вспомогательные методы

После завершения записи данных всех изображений идёт байт-терминатор (завершитель) gif-файла, который равен 0x3B (код символа «;»). Завершитель воспринимается декодером, как сигнал об остановке обработки изображения. Запись завершителя происходит при вызове метода Dispose (листинг 7).

//Метод записывает завершающий байт в поток и выполняет сброс буфера,

//чтобы убедиться, что все данные были записаны в выходной поток

public void Dispose()

{

WriteByte(FileTrailer);

stream.Flush();

}

 Листинг 7 – Метод для завершения работы с файлом