YAFA

COLLABORATORS			
	TITLE : YAFA		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		August 5, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

YAFA

1.1 YAFA

```
                                    Y A F A
            An IFF Format For Animations


             Introduction

             Description

             Contact
                            Release V1.0 (26-May-96)
```

1.2 YAFA/Introduction

```
The animation file format YAFA provides features not seen in other
formats, and its main advantage is the great number of different ways
to store the animation frames.

Here is a list of the main features:
 -bit depth 1...8 (2...256 colors, HAM6 and HAM8)
 -planar or chunky frames
 -optional timecode-per-frame
 -optional palette-per-frame
 -optional delta-compression (similar to IFF ANIM)
 -optional XPK-compression
 -annotations

Using the compression options it is possible to optimize a YAFA file
for maximum playback speed or minimum storage size. Chunky frame data
usually compresses better and can be displayed directly on gfx boards.
```

1.3 YAFA/Description

YAFA files are made of chunks just like specified in the IFF standard.
In the text below the special chunks defined for YAFA and their structure are explained. Please note that there may very well be some extensions in the future. If you are interested in including your own ideas in YAFA please let
us
know.
(things to think about: depth>8, other (better) compression ...)

```
FORM YAFA
.INFO        general information
.DRGB        palette (useless with palette-per-frame)
.DLTX        delta expansion (useless without delta compression)
.PROF        frame offsets (useless without any compression)
.TTBL        timecodes for each frame (used if present)
.ANNO        annotation
.BODY        frames (and palette-per-frame)
```

The order of the chunks is not important except that the BODY chunk must be the last chunk in the file.
Chunks that are useless in a special file should be omitted.

INFO

This chunk is mandatory for YAFA files. It contains data about the frames and specifies which optional features are used.

```
 UWORD width       (in pixels - must be a multiple of 16)
 UWORD height      (in pixels)
 UWORD depth       (currently supported: 1...8)
 UWORD speed       (playback speed, each anim frame is displayed this number
                    of video frames, frames-per-second can be calculated by
                    dividing video-frames-per-second (ie. PAL: 50) by speed)
 UWORD frames      (number of frames)
 UWORD frametype   (currently supported are:
                    0 planar
                    1 planar with XPK compression
                    3 chunky 8bit with XPK compression
                    4 chunky 8bit
 UWORD flags       (set bits have following meanings:
                    BIT 0 Hold-And-Modify mode (HAM)
                    BIT 1 palette-per-frame ('dynamic palette')
                    BIT 2 delta compression, the next two bits specify
                          the type:
                                    BIT 4 set - LONG
                                    else BIT 3 set - WORD
                                    else BYTE
                   other bits are unused, clear them for future compatibility)
```

DRGB

This chunk contains the palette used for all frames. It can be omitted when dynamic palette is enabled.
The palette is stored in a loadrgb structure which can be used as a parameter for graphics/LoadRGB32() for instance.

```
struct loadrgb
{
UWORD count;      (number of colors)
UWORD first;      (number of the color register to receive the first color
                   in this structure)
ULONG colors []; (array of color values, each color value consists of three
                   longwords containing the Red, Green and Blue components in
                   their highest bytes)
};
```

DLTX

This chunk is only useful in files with delta-compressed frames. It contains ULONG dltx that is the number of bytes by which the largest delta-compressed frame is larger than a uncompressed frame. If there is no delta frame larger than an uncompressed frame then dltx is NULL.
Also if the frames are delta-compressed and the DLTX chunk is missing the value for dltx should be considered as NULL.
This value is needed for allocating memory for frame buffers, you have to take (uncompressed size + dltx) bytes.

PROF

This chunk contains an array of ULONG values that are the offsets of each frame's end from the beginning of the BODY. In YAFA files without compression all frames have the same size and the offsets can easily be calculated when they are needed, the PROF is obsolete in this case.
If you need to know the size of a specific frame (ie. for loading) simply calculate: size=PROF[i]-PROF[i-1]. (i = 1...number_of_frames-1)
If you need to seek to a specific frame (ie. when skipping some frames) get the offset from the beginning of the BODY out of the PROF: offset=PROF[i-1].
For the first frame you have to use: size=PROF[0] and offset=0.

TTBL

This chunk contains an array of UWORDs that are the speed values for each individual frame. There is no explicit way to enable these timecodes, if the chunk is present in a YAFA file it is used. Anyway, the global speed in the INFO chunk is not obsolete, it must always be set to a legal value (>0). In the TTBL values of 0 are allowed, they just mean not to take a new speed but to use the one from the previous frame instead.
So you might want to add a TTBL chunk to all YAFA anims you create. If they are not needed currently they can all be set to 0, so they have no effect at all, but they can be edited later.

ANNO

This chunk contains some additional information included by the creator
of the YAFA file. In order to allow the content to have an odd length and
to be able to extract exactly this odd length from the ANNO, there is
an ULONG value at the beginning of the chunk containing the real length
of the content. Behind the content there is a pad byte if needed to make
the chunk length even.
The content itself should be a printable zero-terminated ASCII-string. This
leaves the possibility to include binary files too, you just have to make
sure that there is a NULL byte in front of the binary stuff.

BODY

This chunk contains the frames one after another. It is not structured,
which means that you need the information stored in the other chunks to know
how to deal with the BODY chunk.

*planar:
 simply raw bitplanes

*chunky:
 one byte represents one pixel, the value is the color number,
 valid for depth = 5...8 (unused bits are cleared ofcourse)

*XPK compression:
 every frame is compressed individually,
 use xpkmaster.library's functions: XpkPack(), XpkUnPack()

*Delta compression:
 The delta compression used in YAFA is similar to the one in IFF ANIM7, but
 in YAFA files the first two frames are uncompressed. Compressed frames
 look like this:
  -8 pointers to opcode lists
  -8 pointers to data lists
  -opcode lists (elements are bytes)
  -data lists (elements are bytes/words/longwords)
 Unused pointers are set to NULL. (De-)Compression is performed on a
 bitplane-by-bitplane basis. The bitplane is splitted into vertical columns
 that are 8/16/32 pixels wide. (for byte/word/long compression)
 The opcode lists consist of opcodes for every column. A column starts with
 an opcode count. If the opcode count is zero it means that the complete
 column remains unchanged. The opcodes are of three classes and refer to a
 varying amount of data (to fetch from the corresponding data list) depending
 on the class:
  -skip op (non-zero, hi bit clear): skip this number of rows in the
   destination bitplane
  -uniq op (non-zero, hi bit set): clear hi bit, the remainder is the
   number of data items to copy (each item to the next destination row)
  -same op (zero): followed by a count byte, that says how many destination
   rows are to be set to the same data item (the next item in the data list)

*combinations:

```
 All combinations of the above mentioned features are possible. When chunky
 data is to be delta-compressed it is treated like it was planar with eight
 bitplanes.
 When reading a YAFA file do following steps (if necessary):
   -read frame (using PROF for size and offset)
   -XpkUnpack()
   -delta-decompress
   -chunky to planar
 When writing a YAFA file do the analogous steps in reverse order.

*dynamic palette:
 colors are stored directly behind the frame they belong to in a loadrgb
 structure (see DRGB chunk),
 in case of XPK compression frame data and palette are compressed together,
 you have to get the uncompressed size from XPK in order to be able to
 calculate the start of the palette:
            palette_pt = frame_pt + size - colorsize
       with  colorsize = number_of_colors * 12 + 4
```

1.4 YAFA/Contact

```
The YAFA animation format is developed by WK-Artworks and Infect.
We are also programming our own player and processor/converter using
YAFA which are already available (ie. on Aminet). In order to make YAFA
more popular we ask programmers of image/animation software to support
this new format in their programs.
If you have any questions and/or suggestions you are welcome to contact:
```


```
 Michael Henke (aka Smack/Infect)
 Praetoriusstr. 1/205
 06124 Halle/Saale
 Germany
 EMail: epgbd@cluster1.urz.uni-halle.de

 or

 Andreas Maschke (aka WK-Artworks)
 Zenkerstraße 5
 06108 Halle/Saale
 Germany
 Phone: ++49 (0)345/5170331
 EMail: epgbc@cluster1.urz.uni-halle.de
```