The internal format

Things work in a very simple way. We have one main header, follow by header from each frame and then, the data from each frame. Here's a how it works:

- File Header
- Frame 1 Header
- Frame 2 Header
- ...
- Frame N Header
- Frame 1 Data
- Frame 2 Data
- ...
- Frame N Data

File Header

The File Header is very simple and consists only of few variables (unsigned 2 bytes integers):

- Reserved word (must be zero)
- Width
- Height
- NumberOfFrames

Frame Header

Each Frame comes with a relevant amount of data that will help to render it.

- X[uint16] (Horizontal position of the 0,0)
- Y[uint16] (Vertical position of the 0,0)
- Width[uint16] (Width of the frame. Note that X + Width < FileHeader.Width)
- Height[uint16] (Height of the frame. Note that Y + Height < FileHeader.Height)
- Flags[uint8] Special flags. It can be any number
- Align[3] 3 bytes align to dword
- Color[uint32] Some color (Can be Transparent color)
- Reserved2[uint32] (A bunch of zero. Unused by Westwood. That's a kind of space - you can use to add... things like password for it).
- Offset[uint32] (Location, inside the file, of the frame data. Use this value - with Seek to reach the frame data. If offset equal 0 it is NULL "frame"

Frame Data

The frame data is stored in one of two types, determined by the "flags" byte in the frame header. If the flag data have second bit clear, the frame data is stored as (FrameHeader.Height * FrameHeader.Width) bytes, each byte being an index of the palette

If the flag data second bit is set the data is stored using RLE compression. See below for more details.

Compression

If FrameHeader.Flag have second bit set, compression is used.

```
if (info->flag & 0x2)
{
  // with compression code
}
else
{
  // without compression code
}
```

First method:
The first two bytes of the frame data indicate how many bytes are in the line (that is, how many bytes of the frame data represent a whole row of pixels). This count includes the two bytes that say how many bytes in the line (for example if the value was 8, there would be 6 bytes of pixel information until the next line).
After this, there is a series of bytes indicating which index of the PAL file the colour is. In any case where the colour is 0x00, then the following byte indicates how many pixels of colour 0x00 (transparent) are to be written. For example, a line might read:

```
07 00 00 09 53 00 9C
```

This would be 9 pixels of transparent colour, colour 0x53 in the pallette, and then another 0x9C pixels of transparent. This format repeats until all the lines in the image have been accounted for.