VDX

[Jump to navigation](https://wiki.multimedia.cx/index.php/VDX#mw-head)[Jump to search](https://wiki.multimedia.cx/index.php/VDX#searchInput)

* Extension: VDX
* Company: [Trilobyte](https://wiki.multimedia.cx/index.php?title=Trilobyte&action=edit&redlink=1" \o "Trilobyte (page does not exist))
* Decoder: <http://scummvm.svn.sourceforge.net/viewvc/scummvm/scummvm/trunk/engines/groovie/vdx.cpp>

VDX is a full motion video format used in the game [The 7th Guest](http://www.mobygames.com/game/dos/7th-guest).

**Contents**

* [1File format](https://wiki.multimedia.cx/index.php/VDX#File_format)
* [2General chunk format and decompression](https://wiki.multimedia.cx/index.php/VDX#General_chunk_format_and_decompression)
* [3Chunk type 0x00 (Replay information?)](https://wiki.multimedia.cx/index.php/VDX#Chunk_type_0x00_(Replay_information?))
* [4Chunk type 0x20 (Image)](https://wiki.multimedia.cx/index.php/VDX#Chunk_type_0x20_(Image))
* [5Chunk type 0x25 (Delta frame)](https://wiki.multimedia.cx/index.php/VDX#Chunk_type_0x25_(Delta_frame))
* [6Chunk type 0x80 (Sound)](https://wiki.multimedia.cx/index.php/VDX#Chunk_type_0x80_(Sound))

File format

Let the following be conventions throughout this document:

* All data in a VDX file is stored in little-endian (Intel) notation.
* byte - 8 bits unsigned number.
* word - 16 bits unsigned number.
* dword - 32 bits unsigned number.
* rgb - a 3 byte structure representing the R/G/B values of a single colour.

A VDX file has a short dedicated header:

word identifier

byte unknown[6]

* identifier - Should be 0x6792.

The rest of the file is completely chunk-oriented, i. e. following the header there will be an arbitrary number of chunks until the file ends. In general, the first chunk of every VDX file will contain a still image.

Video sequences within a VDX file are intended to be played at 15 frames per second.

General chunk format and decompression

Each chunk is structured as follows:

byte chunkType

byte unknown

dword dataSize

byte lengthMask

byte lengthBits

byte data[]

* chunkType - Determines the type of data.
* unknown - Probably contains replay (synchronization?) commands.
* dataSize - Size of the chunk without the header information.
* lengthMask - Compression information. Not zero when the data has been compressed.
* lengthBits - Compression information. Not zero when the data has been compressed.
* data - The pure chunk data; its lengths is always determied by dataSize.

The chunk data can optionally be compressed by a common variant of the LZSS alogrithm. This is the case if and only if both of the values lengthMask and lengthBits are not equal to zero. Decompression will take place using a circular history buffer and a sliding window with the following parameters:

bufferSize = 1 << (16 - lengthBits)

windowSize = 1 << lengthBits

threshold = 3

All references are relative to the current write position. Initially, writing starts at bufferSize - windowSize. The lengthMask value of the header can be used to isolate the length portion of a buffer reference, though this information seems a bit redundant since the number of bits used (lengthBits) is also known.

Chunk type 0x00 (Replay information?)

The purpose of this chunk is not entirely clear. Its data size is (always?) zero and only the second header byte is set to a non-zero value. The values observed here are 0x67 and 0x77. When converting a VDX file, synchronization between audio and video can apparently be achieved by repeating the last video frame each time this chunk is seen with a value of 0x67.

Chunk type 0x20 (Image)

This chunk contains a compressed bitmap image following a header:

word numXTiles

word numYTiles

word colourDepth

rgb palette[]

byte image[]

The image is split into tiles of 4x4 pixels in size.

* numXTiles - Number of tiles in horizontal direction.
* numYTiles - Number of tiles in vertical direction.
* colourDepth - Colour depth in bits. Only 8 observed.
* palette - As many RGB values as needed for a palette implied by colourDepth. Only 256 observed.
* image - A sequence of (numXTiles \* numYTiles) structures (each 4 bytes in size) describing the image (top left to bottom right, line-wise). A single tile structure looks as follows:

byte colour1

byte colour0

word colourMap

The colourMap is a field of 16 bits determining the colours of the pixels within the 4x4 tile. A pixel will be coloured as indicated by the palette index colour1 if and only if the respective bit is set to 1. Else, colour0 is used:

+----+----+----+----+

| 15 | 14 | 13 | 12 |

LSB (0) MSB (15) +----+----+----+----+

| | | 11 | 10 | 9 | 8 |

XXXXXXXXXXXXXXXX --> +----+----+----+----+

| 7 | 6 | 5 | 4 |

+----+----+----+----+

| 3 | 2 | 1 | 0 |

+----+----+----+----+

Chunk type 0x25 (Delta frame)

If a VDX file contains a video sequence, the first frame of this sequence is the picture contained in the 0x20 chunk. Afterwards, each new frame is represented by a chunk of this type, indicating the changes to be applied to the last frame and its palette. The data starts with a header:

word localPalSize // if zero, the header ends here

word palBitField[16]

rgb localColours[]

byte image[]

* localPalSize - Determines the number of palette entries to be changed for this frame.
* palBitField - A field of 256 bits indicating which palette entries should be changed.
* localColours - A sequence of RGB values (as many as there are bits set in palBitField) indicating the new colours to be inserted into the palette.
* image - A sequence of byte opcodes, each followed by a varying number of byte parameters, describing the delta information.

All palette adaptations are performed on last frame's palette:

LSB ... MSB "i": 0 15 0 15 0 15

| | | | | |

X ... X X ... X X ... X

| | |

palBitFieldIdx: 0 1 ... 15

|

|

V

palette index: 0 1 15 16 31 ... 255

| | | | | |

(palBitFieldIdx,i): (0,15) (0,14) ... (0,0) (1,15) ... (1,0) (15,0)

The image adaptations are applied to the tiles of the last frame and start in the top left corner, again working from left to right.

The opcodes can be devided into seven classes according to their function:

* 0x00 .. 0x5f - Followed by two parameters, which are hereby referred to as colours colour1 and colour0. The current tile will be altered in the same fashion the still image in the 0x20 chunk is decompressed. The colour map used is extracted from a predefined array of word values containing 96 entries. The correct entry can be identified by directly using the opcode as the element index. The map of predefined values is given here byte-wise:

0x00, 0xc8, 0x80, 0xec, 0xc8, 0xfe, 0xec, 0xff, 0xfe, 0xff, 0x00, 0x31, 0x10, 0x73, 0x31, 0xf7,

0x73, 0xff, 0xf7, 0xff, 0x80, 0x6c, 0xc8, 0x36, 0x6c, 0x13, 0x10, 0x63, 0x31, 0xc6, 0x63, 0x8c,

0x00, 0xf0, 0x00, 0xff, 0xf0, 0xff, 0x11, 0x11, 0x33, 0x33, 0x77, 0x77, 0x66, 0x66, 0xcc, 0xcc,

0xf0, 0x0f, 0xff, 0x00, 0xcc, 0xff, 0x76, 0x00, 0x33, 0xff, 0xe6, 0x0e, 0xff, 0xcc, 0x70, 0x67,

0xff, 0x33, 0xe0, 0x6e, 0x00, 0x48, 0x80, 0x24, 0x48, 0x12, 0x24, 0x00, 0x12, 0x00, 0x00, 0x21,

0x10, 0x42, 0x21, 0x84, 0x42, 0x00, 0x84, 0x00, 0x88, 0xf8, 0x44, 0x00, 0x32, 0x00, 0x1f, 0x11,

0xe0, 0x22, 0x00, 0x4c, 0x8f, 0x88, 0x70, 0x44, 0x00, 0x23, 0x11, 0xf1, 0x22, 0x0e, 0xc4, 0x00,

0x3f, 0xf3, 0xcf, 0xfc, 0x99, 0xff, 0xff, 0x99, 0x44, 0x44, 0x22, 0x22, 0xee, 0xcc, 0x33, 0x77,

0xf8, 0x00, 0xf1, 0x00, 0xbb, 0x00, 0xdd, 0x0c, 0x0f, 0x0f, 0x88, 0x0f, 0xf1, 0x13, 0xb3, 0x19,

0x80, 0x1f, 0x6f, 0x22, 0xec, 0x27, 0x77, 0x30, 0x67, 0x32, 0xe4, 0x37, 0xe3, 0x38, 0x90, 0x3f,

0xcf, 0x44, 0xd9, 0x4c, 0x99, 0x4c, 0x55, 0x55, 0x3f, 0x60, 0x77, 0x60, 0x37, 0x62, 0xc9, 0x64,

0xcd, 0x64, 0xd9, 0x6c, 0xef, 0x70, 0x00, 0x0f, 0xf0, 0x00, 0x00, 0x00, 0x44, 0x44, 0x22, 0x22

* 0x60 - 16 parameters which represent individual palette entries. The 16 pixels of the current tile will be filled by the corresponding colours.
* 0x61 - No parameters. This opcode simply skips one line as a carriage return/line feed would do, i. e. the new position will be 4 pixels (1 tile) further down on the left border of the frame.
* 0x62 .. 0x6b - No parameters. These opcodes skip tiles within a line. The current position is moved (Opcode - 0x62) tiles to the right. (This would seem to indicate that 0x62 is some kind of a NOP and thus without apparent purpose?)
* 0x6c .. 0x75 - One parameter defining a palette entry. The next (Opcode - 0x6b) tiles in the current line will be solidly filled with the corresponding colour.
* 0x76 .. 0x7f - (Opcode - 0x75) parameters. Each defines a palette entry and is used to solidly fill a whole tile within the current line.
* 0x80 .. 0xff - 3 Parameters. The opcode itself and the following byte represent a colour map, the next two bytes are the palette entries colour1 and colour0, respectively. The tile will be coloured using the usual method.

Chunk type 0x80 (Sound)

A chunk of this type contains raw 8 bit mono wave data sampled at 22,050 Hz.