# The unofficial DMO format description


## Uwe Girlich

**uwe@planetquake.com**



**v0.0.6, 3/11/1998**



This document describes the DMO file format. This file format is the result of “recording” a game in Duke Nukem 3D or Redneck Rampage. This documentation covers the old Shareware versions 1.0 and 1.1 and the new Shareware version 1.3D of Duke Nukem 3D and the alpha demo version 0.7 ‘moonshine’ of Redneck Rampage. only.


## Table of Contents



# 1. Introduction

To create a DMO file start the game with the command line switch `/r` (and `/l` (level), `/s` (skill) etc.) and play as usual. If you press `F10` the record (and the game) stops. You will find a file `demo1.dmo` in the current directory. To play it back just start the game and hide the menu by pressing `ESC`.

In Duke Nukem 3D 1.0 and 1.1 the skill value is a bit strange: To record a skill *n* (1<=*n*<=4) game note to use `/sn+1`. So `/s2` is “Peace of Cake” and `/s5` is “Damn I’m Good”. The `dn3dhelp.exe` hint (`/sn` with 0<=*n*<=3) is totally wrong.

Multiplayer recordings have always the skill 0. Cooperative recordings won’t play back properly.

It is impossible to record with version 1.0 more than one level. You can’t even record the summary screen at the end of each level.

In version 1.1 it *seems* to be possible (I got it some times). Some other times I got only the recording of the last level.

A DMO file records all player actions. The monster movements, respawn positions etc. are totally deterministic. The messages during a multiplayer game (macros and RemoteRedicule (tm)) do not appear in the DMO.

A DMO file consists of a header with some organizational information and the data area with all the (compressed) game tics.

The term “game tic” comes originally from DOOM and denotes the smallest unit of time during the game. The duration of a game tic is 1/30s. To store a game tic in a file means to store all actions, like movement, open doors, fire weapons and so on, happened during this time.

# 2. Header

## 2.1. old Duke Nukem 3D Header

Duke Nukem 3D 1.0 and 1.1 use a 9 byte main header:

| address | type | contents |
|---|---|---|
| `0x0000` | `long` | number of game tics times numberof players |
| `0x0004` | `byte` | episode (0-3) |
| `0x0005` | `byte` | map (0-8) |
| `0x0006` | `byte` | skill (0-4) |
| `0x0007` | `word` | player number |
| | | |

The first entry in the header (number of game tics) may be zero in version 1.1 recordings. This should mean, that there is more than one level recorded.

All `word` or `long` values in DMO files are Intel ordered (lowest byte first, little endian).

## 2.2. new Duke Nukem 3D Header

Duke Nukem 3D 1.3D uses a 24 byte main header:

| address | type | contents |
|---|---|---|
| `0x0000` | `long` | number of game tics times number of players |
| `0x0004` | `byte` | volume - 1 (/v parameter - 1) |
| `0x0005` | `byte` | level - 1 (/l paramerer - 1) |
| `0x0006` | `byte` | skill (0 .. 4) (/s parameter) |
| `0x0007` | `byte` | MP mode (/c 1 = DukeMatch(spawn), 2 = Coop, 3 = Dukematch(no spawn)) |
| `0x0008` | `short` | player number (1..8) |
| `0x000A` | `short` | `0x01` with /m (nomonsters), `0x00` else |
| `0x000C` | `long` | `0x01` with /t1 (respawn monsters), `0x00` else |
| `0x0010` | `long` | `0x01` with /t2 (respawn items), `0x00` else |
| `0x0014` | `long` | `0x01` with /t3 (respawn inventory), `0x00` else |
| | | |

## 2.3. Redneck Rampage Header

Redneck Rampage uses a 543 byte main header:

| address | type | contents |
|---|---|---|
| `0x0000` | `long` | number of game tics times number of players |
| `0x0004` | `byte` | `0x6C`, it may be a version |
| `0x0005` | `byte` | `0x00`, unknown |
| `0x0006` | `byte` | `0x00`, unknown |
| `0x0007` | `byte` | skill (0...4) |
| `0x0008` | `byte` | MP mode (/c parameter - 1) |
| `0x0009` | `byte` | level (1..11) |
| `0x000A` | `byte` | player number (1..8) |
| `0x000B` | `byte` | `0x00`, unknown |

| | | |
|---|---|---|
| 0x000C | byte | 0x01 with /m (nomonsters), 0x00 else |
| 0x000D | byte | 0x00, unknown |
| 0x000E | long | 0x01 with /t1 (respawn monsters), 0x00 else |
| 0x0012 | long | 0x01 with /t2 (respawn items), 0x00 else |
| 0x0016 | long | 0x01 with /t3 (respawn inventory), 0x00 else |
| 0x001A | long | 0x01 with /a (player AI for fake players), 0x00 else |
| 0x001E | char[0x201] | -name parameter |
| | | |

The episode (better: volume) number is one of the 0x00 bytes in the header.

The players name is a '\0' terminated string. The maximum number of characters for the name is 512.

# 3. Data

The data starts in byte 0x0009 (old Duke Nukem 3D), 0x0018 (new Duke Nukem 3D) or 0x021F (Redneck Rampage) of the DMO file and is organized in several blocks.

## 3.1. Block

Each block consists of a header with some organizational information and the compressed data.

A block header is:

| address | type | contents |
|---|---|---|
| 0x0000 | word | size of the following blockin bytes |
| 0x0002 | word | size of the uncompresseddata block in bytes |
| 0x0004 | word | biggest code for Lempel-Zivdecompression |
| | | |

The compressed data starts in byte 0x0006 of the block. The block size (0x0000/0x0001) includes the rest of the header. This means there are only block size - 4 data bytes.

## 3.2. Compression technique

Duke Nukem 3D uses a modified Lempel-Ziv algorithm (similar to the UNIX command `compress(1)`) to compress the game tics. To reach an even better compression ratio not the game tic itself but the difference to the last one of the same player (difference per byte, without carry bits) will be compressed. This increases the number of 0 bytes enormously and allows long recording in short files.

In fact Steffen Winterfeldt changed the original `compress.c` until we could decompress a DMO file.

If you are really more interested in the compression/decompression routines look in the files `lzw.c` and `unlzw.c` included in LMPC, the Little Movie Processing Centre. You can get it from my Demo Specs page (http://www.planetquake.com/demospecs).

The compression algorithm used in Redneck Rampage seems to be similar but I did not get `lzw.c` and `unlzw.c` to work with Redneck Rampage recordings.

## 3.3. Macro block

Some blocks of data form a macro block. This has to do with both decompression and game tic difference storing. The first game tic in a macro block is the original game tic. All the following game tics are only the byte-per-byte difference game tics to its specific predecessors. There is no special code to signalize the end of a macro block. It is simply the number 2520/(player number) game tics, which makes a macro block. Note that 2520 is divisible by 1,2, ... ,8 without remainder. Only the last macro block may contain less game tics if the file ends before.

## 3.4. Data description

One game tic corresponds to 10 times player number bytes:

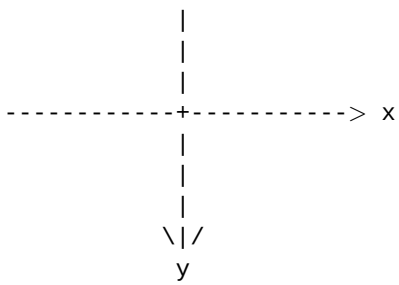
| address | type | player | contents |
|---|---|---|---|
| `0x0000` | `word` | 1 | go in *x*-direction |
| `0x0002` | `word` | 1 | go in *y*-direction |
| `0x0004` | `word` | 1 | turn |
| `0x0006` | `long` | 1 | use/open etc. |
| `0x000a` | `word` | 2 | go in *x*-direction |
| `0x000c` | `word` | 2 | go in *y*-direction |
| . | | | |
| . | | | |
| . | | | |
| | | | |

### 3.4.1. Go *x* and *y*

The 2 words (`signed short`) are duke's speed or the displacement vector in length units per game tic.

To calculate the absolute value of his speed just calculate sqrt($x^2$+$y^2$).

A standard speed is 1280 (with running 2560). The coordinate system used is like this:




### 3.4.2. Turn

The turn word contains 2 bytes (`signed short`) which are the current turning speed or the angle difference per game tic. A positive turning speed means right and a negative means left.

### 3.4.3. Use

There are many “use” actions in Duke Nukem 3D. You can do all at once, because there is a single bit for each action. The apropriate bit is 1 as long as you press the corresponding key.

| bit | purpose | |
|---|---|---|
| 31 | ?? | |
| 30 | Inventory | |
| 29 | Open | |
| 28 | TurnAround | |
| 27 | Inventory_Right | |
| 26 | ?? | |
| 25 | Jetpack | |
| 24 | Holo_Duke | |
| 23 | Mouse_Aiming | new in 1.1 |
| 22 | ?? | |
| 21 | Pause | |
| 20 | Inventory_Left | |
| 19 | Holster_Weapon | |
| 18 | Center_View | |
| 17 | AutoRun | |
| 16 | MedKit | |
| 15 | NightVision | |

| | | |
|---|---|---|
| 14 | Look_Down | |
| 13 | Look_Up | |
| 12 | Steroids | new in 1.1 |
| 8-11 | Weapon number | |
| 7 | ?? | |
| 6 | ?? | |
| 5 | Run | |
| 4 | Aim_Down | |
| 3 | Aim_Up | |
| 2 | Fire | |
| 1 | Crouch | |
| 0 | Jump | |
| | | |

# 4. Version History and Acknowledgements

0.0.0, 8 February, 1996

- First internal version (working paper); never announced.
- Many thanks to Steffen Winterfeldt (Steffen.Winterfeldt@itp.uni-leipzig.de (mailto:Steffen.Winterfeldt@itp.uni-leipzig.de)) for his reverse engineering and programming work.

0.0.1, 10 February, 1996

- First helpful documentation for further research on Duke Nukem 3D; never announced.

0.0.2, 12 February, 1996

- All header bytes decoded, never announced.

0.0.3, 19 February, 1996

- Minimal changes, some remarks to record properly.

0.0.4, 23 February, 1996

- New actions from version 1.1.
- Multi-level recording from 1.1.
- Reorganization of block/macro block description.

0.0.5, 4 May, 1997

- Redneck Rampage info included (incomplete).
- Duke Nukem 3D 1.3D info included.

0.0.6, 12 March 1998

- PlanetQuake is the new home.
- SGML-Tools 1.0.5 used.