DARK FORCES SPECS

Table des matières

Title Page

UNOFFICIAL SPECIFICATIONS
v 3.01

Welcome Page

Quoting DF Specs v. 1.00:
"As you will rapidly see, there are still a lot of unknown things in these specs. But as they are diminishing very quickly, I found it was time to write them down in a "formal" way, which can serve as a reference."

So this is version 3.01 of the DF Specs! I (Jereth) have been very privileged to have been asked by Yves to become a co-author. Hopefully now there will be more frequent updates since there are now four of us working at it, but as nearly everything about DF has been discovered, and Jedi Knight is rapidly approaching, updates may not be a necessity after all!

And I (Yves) am very happy to have those knowledgeable friends share the work with me :-)
Jereth is completely right : those Specs are now nearly finished (after nearly two years !), and I believe the following updates will be more oriented to the Reference section than to real new discoveries.

However I (Alexei) tend to disagree with my collegues: there's still quite a few blanks in DF Specs: iMuse commands and EXE hack to name a few. Sadly, some of these blanks are critical: like we can't patch in-level music without knowing internal iMuse commands.
I hope we'll fix it soon.

And your servitors look forward to beginning work on the Jedi Knight Unofficial Specifications :-)

We will try to explain differences with DOOM level making where applicable, so if you have experience in that domain, look for the **Doom note** hyper jumps in the header of some pages.
Removing the Doom notes from the body of the text will make life simpler for newcomers, who have been buried in two jargons mixed together until now.

As always, nothing is as good as seeing how professionals do things, so don't hesitate to go and see an example of how the LucasArts team implemented what you want to do.

May The Force Be With You,
Yves Borckmans, Jereth Kok, Alexei Novikov and David Lovejoy

Note
We will frequently use the following abbreviations :

DF	Dark Forces
LEC	LucasArts Entertainment Company
SC	sector
WL	wall
VX	vertex
OB	object
TX	texture

What's New Page

Version 3.01
- introduces as new authors people who have done so much for DF Specs
- matches the HTML Specs 3.01 exactly
- corrects some minor errors in 3.00

File Formats

- Made some important corrections to the JEDI.LVL description.
- Made a very minor note about Walk: in the Adjoin/Mirror/Walk Mechanism description.
- Made some additions and corrections to Sector flags and Wall flags
- Updated Object Sequences and Logics, Generators, and the Full Logics list
- Made an addition to MSG files
- Completely overhauled the INF section
- Made additions and changes to BM, WAX and 3DO descriptions.
- Added a CUTMUSE.TXT section.
- Changed and added to the GOL file description.
- Updated the DFBRIEF.LFD contents list.
- Alex Novikov gave us some extra info for the GMD description.
- Added a VOC file format.

Reference

- Made some additions and corrections to Limitations on objects.
- Added the Metrics topic.
- Added Textures.gob (A-N) file list.
- Added Textures.gob (R-Z) file list.
- Added the Resources Cross Reference lists.

Author and Credits

Note -- there have been changes to Jereth Kok's, Carlos Gomez, David Lovejoy's and Serge Debroeyer's e-mail addresses.

Authors

Yves BORCKMANS (`Yves.Borckmans@ping.be`)
Jereth KOK (`jereth@alphalink.com.au`)
Alexei NOVIKOV (`anoviko@emory.edu`)
David LOVEJOY (`dlovejoy@nucleus.com`)

Credits

HELP WANTED
If you have deciphered information that isn't covered in the Specs, don't hesitate to send it to us.
We'll include it in the next release.

Serge DEBROEYER (`sdeb@rtbf.be`)
for adjoin/mirror tips, some flags, some INF tips, complex sectors errors, ...

Don SIELKE (`DSielke@aol.com`)
for the complete flags list, the -u option, ...

Florian ENGELHARDT (`100442.2012@compuserve.com`)
for information on GMD files

Paul NEMESH (`PNem@aol.com`)
for information and text on VUE files

Randy GREENE (`rgreene@val.net`)
for information on event_masks

Carlos GOMEZ (`CarGo95@aol.com`)
Jonathan WISE (`TheRedDeth@aol.com`)
for information on frames and waxes

Nicola SALMORIA (`MC6489@mclink.it`)
for information and texts on dfbrief.lfd and briefings.lst

Michael TAYLOR (`MichaelLTa@aol.com`)
for information and texts on cutscene.lst and the 3DO description

Carl KENNER (`Andrew.Kenner@Unisa.Edu.Au`)
for information and text on the FILM LFD resources

Len BOWERS (`len@lenbow.demon.co.uk`)
for information on the LEV PALETTE entry, a sound list and for relaying other info to me

Blake CROSBY (`bcrosby@interlog.com`)
for a jedisfx.lfd sound list

Peter KLASSEN (`101336.145@compuserve.com`)
for information on briefings, and finding and helping to figure out several new INF functions

Anthony HALL (`Ehhbetsy@aol.com`)
for finding the INF texture function, and some metrics.

Paulius Stepanas (`PStepana@VTRLMEL1.TRL.OZ.AU`)
for his textures descriptions.

Very special thanks to Daron, Ingar and Ray. You're the best !

May The Force Be With all of You...

Copyrights

DARK FORCES	is (c) LucasArts Entertainment Company
DOOM	is (c) iD Software

None of these have anything to do with these totally unofficial specs, and shouldn't be bothered with them in any way.

Thanks for those GREAT games...

DF SPECS	are (c) Yves Borckmans, Jereth Kok, Alexei Novikov, David Lovejoy

Accelerator Page 1

File Formats
Reference

In 3D Engine
GOB
LEV O
INF GOL
BM FME WAX
3DO VUE
PAL CMP
FNT
VOC GMD
MSG

Out of 3D Engine
LFD
Jedi.lvl
Briefing.lst
Cutscene.lst
Cutmuse.txt
ANIM DELT FILM
PLTT
FONT
VOIC GMID
MSG

Accelerator Page 2

File Formats

Reference

General Description

General Description

General Description - Containers & Patching

Containers & Patching

Doom Note

Containers

Dark Forces contains a huge amount of files, and these are grouped by type in what are called GOB files and LFD Files.
They serve as **containers** for other files which in turn contain information.
GOB Files mainly contain data for the DF engine, while LFD Files contain the data needed for the Landru system (cutscenes, menus, briefings).

Patching

Files can be extracted from the GOB and worked upon, then be used in the game.
LucasArts made a very good job of this and prepared a path for users to modify levels.
If you want to make DF accept new or modified files, create a GOB file with them (say mylevel.gob).
Then use the following command line: **dark -umylevel.gob**

The order in which DF looks for a file is as follows:
1) as a file in the installed directory
2) in the GOB specified by -u...
3) in its normal GOB in the installed directory
4) in its normal GOB on the CD

LFD resources can also be extracted and worked upon, but there isn't any facility to load them.
What you have to do is recompose the patched LFD, and set it in the installed directory, or in the LFD subdirectory of the installed directory. A LFD in the installed directory will be taken in preference to one in the LFD subdirectory with the same name.

Putting a patched LFD in a GOB and loading it with -u doesn't work.

General Description - General File Descriptions

General File Descriptions

Doom Note

Files critical to DF:

JEDI.LVL	List of the levels
TEXT.MSG	In-game text messages
BRIEFING.LST	List of the briefings
CUTSCENE.LST	List of the cutscenes
CUTMUSE.TXT	Cutscene musics

The levels themselves are each composed of 6 files, found in dark.gob:

name.LEV	geometry (static)
name.INF	workings (dynamic)
name.GOL	goals
name.O	objects
name.PAL	palette
name.CMP	palette mappings

Resources:

Textures	are stored in .BM files, as are the weapons display, and so on.
Sounds	are stored in .VOC files (normal Creative Labs format).
Music	are stored in .GMD files (type 2 midi files)

Objects are stored in the following files depending on their type:

obj.3DO	3D	object (real 3D)
obj.FME	FRAME	(a "one view" object)
obj.WAX	SPRITE	(i.e. all the enemies)
obj.VOC	SOUND	(any sound)

3D object motions are stored in VUE files (normal 3D Studio format).

File Formats

	In 3D Engine	**Out of 3D engine**
Container/Distribution		
	GOB	LFD
Level Related		
	LEV	JEDI.LVL
	Sector Flags	BRIEFING.LST
	Wall Flags	CUTSCENE.LST
	INF	CUTMUSE.TXT
	GOL	
	O	
Graphics Related		
	BM	ANIM
	FME	DELT
	WAX	FILM
	3DO	
	VUE	
	PAL	PLTT
	CMP	
	FNT	FONT
Sound Related		
	VOC	VOIC
	GMD	GMID
Text Messages Related		
	MSG	MSG

GOB Files

Doom Note

GOB files are a repository for many other files, and are by far the best way to distribute add-on levels.
They contain a header with a signature, a data part and an index part.

```
GOB_Header IS
{
 GOB_MAGIC     char[4]          // 'GOB' followed by 0x0A
 MASTERX       long             // offset to MASTERN
}
```

The embedded files follow, then comes the index.

```
GOB_Index IS
{
 MASTERN       long             // number of files in the GOB
 INDEXES       GOB_Ix_Entry[n]  // one index entry per file
}
```

Where:

```
GOB_Ix_Entry IS
{
 IX             long              // pointer to start of the file
 LEN            long              // length of the file
 NAME           char[13]          // name of the file,
                                  // null terminated
}
```

LFD Files

LFD files contain various resources, mostly sound and graphics.
You don't need them to create a new level, but for things like cutscenes and briefings.
The Dark Forces LFD format is completely compatible with X-Wing and Tie Fighter LFD files.

```
LFD_Ix_Entry IS
{
 TYPE           char[4]           // type of the resource
 NAME           char[8]           // name of the resource
 LENGTH         long              // length of the resource
}
```

Then LENGTH bytes follow the header.

The first index entry is of type RMAP, and contains the list of all the sections in the .LFD file.
This is similar to the GOB Master Index.

The other sections can be:

Section	**Description**
ANIM	animation, this is a collection of DELT
DELT	static image in delta format
FILM	'script' referencing the other resources in the LFD
FONT	font
GMID	General Midi music
PLTT	palette used for ANIM and DELT
VOIC	VOC (standard Creative Labs format)

Jedi.lvl

Contains a list of the levels in DF.

```
| LEVELS 14
```

This is the number of entries in JEDI.LVL

```
| Secret Base,        SECBASE,
\ L:\LEVELS\SECBASE\;L:\LEVELS\;L:\LEVELS\BM-GEN\;
\ L:\LEVELS\PALETTES\;L:\LEVELS\BM-IMPER\;
\ L:\LEVELS\HOLDER\
| ...
```

This is the definition for a level. The first entry is the description (eg. "Secret Base") to be shown in the mission menu in DF.

The second entry is the name of the level (e.g. SECBASE). It will be applied in the following areas:

levname.LEV
levname.O
levname.INF
levname.GOL
levname.PAL
levname.CMP

DELTlevname
ANIMlevname

LEVELNAME entries in headers of LEV, O and INF files.

LEV entry in BRIEFING.LST

To successfully change the name of a level, its name must be changed in all of these as well as in JEDI.LVL.

The paths stored are unused, and were most probably referring to the LucasArts file server at development time.

The remainder of this file contains the names of all the levels in Dark Forces.

LEV Files

LEV Files

LEV files contain a complete level geometry. They are in a quite complex text format.
They are also huge (generally > 600K), but this isn't a problem, as you really cannot edit them as a text file, because of the many dependencies between the geometry elements.

Geometry Elements

Doom note

The basic geometry elements of a DF level are :

VERTEX	a point in a 2 dimensions projection (X and Z)
WALL	a line joining 2 vertices
SECTOR	a collection of walls generally closed, can contain "gaps" or other sectors

As the game works with a two dimensions projection, the third (Y) dimension is coded at the sector level by a floor altitude and a ceiling altitude.
Note that this imply that floors and ceilings of a sector are always **FLAT**.

Sectors can however be **layered** on top of one another to give a "full 3D" feeling.

Each sector is coded with its walls and vertices, and is completely self contained

The relation between sectors is done at the wall level by the adjoin/mirror/walk mechanism.

Adjoin/Mirror/Walk Mechanism

Doom note

```
1--------2    Sector 1 has 5 vertices (0 to 4) marked 0 1 2 A B
|        |                 5 walls    including AB (wall 3)
|   S1   |
|        |    Sector 2 has 4 vertices (0 to 3) marked 0 1 B A
0---B====A                 4 walls    including BA (wall 2)
    | S2 |
    1----0
```

It is VERY important to note that there are 2 vertices at point A, two vertices at point B and 2 walls marked ==== . As I said earlier, sectors are self contained.

So, to come back to the adjoin/mirror/walk mechanism, if S1 and S2 must be connected, an adjoin/mirror relation must be established.

```
+----1---+
|        |
0        2
| 4    3 |
+---+====+
    1   2 3
    +--0-+
```

This is quite simple : the adjoin is the number of the connected sector, and the mirror is the number of the connection wall.

So we would need to set:
    in S1 : W3.adjoin = S2 and W3.mirror = 2
    in S2 : W2.adjoin = S1 and W2.mirror = 3

If there is no adjoin/mirror relationship, the values for adjoin and mirror will be -1.

Walk values seem to have no effect at all in a level, but they are mostly set to the same value as adjoin.

A Quick Note on Texturing

Doom note

When you have adjoined sectors:

TOP	is ABOVE the ceiling of the other SC
BOT	is BELOW the floor of the other SC
MID	is everywhere you can see through to the other SC

Of course, the MID texture is not shown when walls are adjoined, so that you can see through!
(Note: WL flag 1, bit 1 forces it back in place. See Wall Flags)

LEV File Format

The LEV file is composed of 3 parts:
- Magic, Version number and general level info
- Texture Table
- Geometry Description i.e. sectors, walls, vertices data

The following **comments** are accepted:
- # comment
- DATA # comment

LEV Magic, Version number and General Level Info

Magic and version number

This is trivial.

```
| LEV 2.1
```

General Level Info

This part contains the following data (sample from secbase.lev):

```
| LEVELNAME SECBASE
| PALETTE   SECBASE.PAL
| MUSIC     AVENGE.GMD
| PARALLAX  1024.0000 1024.0000
```

It seems that LEVELNAME isn't used at all by DF.
MUSIC is also unused, because musics are hardcoded in dark.exe.
(AVENGE.GMD doesn't even exist in DF, I think it is a Tie Fighter music !)

PALETTE determines the palette (PAL) used in the level, you may change it.

PARALLAX determines how much the "exterior" backgrounds scroll as you turn.
1024 1024 means as you turn around 360 degrees, you will see 1024 pixel columns of background sky.
Vertical PARALLAX is similar, although of course you can't pitch 360 degrees in DF.

LEV Texture Table

As there is a lot of TX information in a level, a texture table is created to avoid storing TX names in full at each occurrence.

Coding sample :

```
| TEXTURES 85         # number of textures
| TEXTURE: TEX00.BM   # texture 0
| TEXTURE: TEX01.BM   # texture 1
| ...
| TEXTURE: TEX84.BM   # texture 84
```

Afterwards, all the textures are referred to by their 0 based index in this texture table.

Note that changing TX names in the TX table may be an ultra fast way to relook a level !

LEV Geometry Description

The first data is the total number of sectors in the level :

```
| NUMSECTORS number_of_sectors
```

Then each sector is described, with its vertices and walls.
Please note that the wall data is on ONE line, but has been split here for visual convenience.

```
| SECTOR scnum
|  NAME              sector_name
|  AMBIENT           20
|  FLOOR TEXTURE     80 -0.38 -0.06 2
|  FLOOR ALTITUDE     0.00
|  CEILING TEXTURE   0  0.00  0.00 2
|  CEILING ALTITUDE -12.00
|  SECOND ALTITUDE    0.00
|  FLAGS 0 0 0
|  LAYER             1
|
| VERTICES numvx
|  X: 252.00 Z: 224.00 # a vx
|  ...
|
| WALLS numwl
|  WALL LEFT:  0 RIGHT:  1
\       MID:   0  0.00  0.00 0
\       TOP:   1  0.00  0.00 0
\       BOT:   2  0.17  0.00 0
\       SIGN: -1  0.00  0.00
\       ADJOIN: 57 MIRROR: 0 WALK: 57
\       FLAGS: 0 0 0
\       LIGHT: 5
|  ...
```

Hmmm... heavy information!
Click a section to take it apart, it's not too difficult.

LEV Geometry - Sector

```
| SECTOR scnum
```

This is the sector number, it is zero based.

```
|  NAME             sector_name
```

This is both a link to the .INF file and a useful reminder.

```
|  AMBIENT          20
```

Ambient light level in this sector.
Note that this value is used in GROMAS to indicate an amount of red fog, not a light level.
This is a good demonstration of the use of the CMP files.

```
|  FLOOR TEXTURE    80 -0.38 -0.06 2
```

The TX to apply to the floor of the SC as an index in the TX table.
The following two floats are the X and Z offsets by which the TX must be moved before being mapped.
The third (int) value is unused.
It seems that floor textures must be 64x64, or the game engine does strange things.

```
|  FLOOR ALTITUDE    0.00
```

The altitude of the floor of this SC. Note that the Y axis goes "down", so higher altitudes have lower values.

```
|  CEILING TEXTURE  0  0.00  0.00 2
|  CEILING ALTITUDE -12.00
```

Same as floor.

```
|  SECOND ALTITUDE   0.00
```

This is used to indicate a second "floor" altitude in a sector. For instance, a second altitude of 4 will make you "enter into the floor" 4 deep. It will in addition make the sector water like and generate a splashing sound. If you set a negative second altitude, you will be able to walk higher on the sector, provided you also enter the sector higher.
This is the way platforms are created (the platform object is only a visual clue).

```
|  FLAGS            0 0 0
```

Three flags, the second of which is never used in the 14 original levels.
Change various things in the sector. See Sector Flags.

```
|  LAYER            1
```

The layer on which the SC is (positive, 0 or negative).
This value is used in the game to make different maps corresponding to zones of altitude.
Note that this is only a logical grouping, but is also used by the map in the game.

LEV Geometry - Vertices

```
| VERTICES vxnum
```

This is the number of vertices that this SC has.

```
|  X: 252.00 Z: 224.00 # a vx
```

List of the vertices.
X and Z are trivial.

LEV Geometry - Walls

```
| WALLS wlnum
```

This is the number of walls that this SC has.

```
|  WALL LEFT:  0 RIGHT:  1
```

These are the origin and destination vertices for this wall.

```
\       MID:   0  0.00  0.00 0
```

The TX to apply to the middle of the WL as an index in the TX table.
The following two floats are the X and Y offsets by which the TX must be moved before being mapped (remember Y goes down).
The third (int) value is unused.

```
\       TOP:   1  0.00  0.00 0
\       BOT:   2  0.17  0.00 0
```

Same as MID

```
\       SIGN: -1  0.00  0.00
```

A sign is a second TX on the same WL, its main use is to place switches.
First is the TX to apply to a sign on the WL as an index in the TX table.
The following two floats are the X and Y offsets by which the TX must be moved before being mapped (remember Y goes down). Also note that this is relative to the texturing of the wall. So if you offset the WALL, you have to add this offset to that of the SIGN.

```
\       ADJOIN: 57 MIRROR: 0 WALK: 57
```

See The Adjoin/Mirror/Walk mechanism

```
\       FLAGS: 0 0 0
```

Three flags.
Change various things in the wall. See Wall Flags.

```
\       LIGHT: 5
```

Relative modification of the luminosity on this specific WL.

LEV Sector Flags

FLAG 1

Bit	Description	Comment
1	EXTERIOR - NO CEIL. (SKY)	Note: actual ceiling limit will be the ceiling altitude + 100
2	DOOR	instant door
4	SHOT REFLECTION / MAG.SEAL	walls, floor and ceiling reflect weapon shots
8	EXTERIOR ADJOIN	will adjoin adjacent skies
16	ICE FLOOR (SKATING)	
32	SNOW FLOOR	no apparent effects
64	EXPLODING WALL/DOOR	instant exploding door
128	EXTERIOR - NO FLOOR (PIT)	Note: actual floor limit will be the floor altitude - 100
256	EXTERIOR FLOOR ADJOIN	will adjoin adjacent pits
512	CRUSHING SECTOR	vertically moving elevators will crush the player
1024	NO WALL DRAW / "HORIZON"	removes walls of a sector (sector must be sky and pit to work properly)
2048	LOW DAMAGE	
4096	HIGH DAMAGE	both can be combined for GAS
8192	NO SMART OBJECT REACTION	
16384	SMART OBJECT REACTION	
32768	SUBSECTOR	no apparent effects
65536	SAFE SECTOR	
131072	RENDERED	
262144	PLAYER	
524288	SECRET SECTOR	increments the %secret when entered

Note on the Smart Objects:
Smart Object Reactions will cause doors and CERTAIN elevator classes to react to enemies. There are two values, not a toggle, because Flag doors by default react to smart objects, and INF elevators by default don't react.

These are the elevators than can react to smart objects:
```
basic
inv
basic_auto
morph_move1
morph_move2
morph_spin1
morph_spin2
move_wall
rotate_wall
door
door_mid
door_inv
```

FLAG 2

is unused.

FLAG 3

When "message: system lights" is sent (e.g. in TALAY when you turn on the generator), the engine copies the value here to the Ambient of the sector.

LEV Wall Flags

FLAG 1

Bit	Description	Comment
1	ADJOINING MID TX	the MID TX is NOT removed
2	ILLUMINATED SIGN	
4	FLIP TEXTURE HORIZONTALLY	
8	ELEV CAN CHANGE WALL LIGHT	
16	WALL TX ANCHORED	
32	WALL MORPHS WITH ELEV	
64	ELEV CAN SCROLL TOP TX	
128	ELEV CAN SCROLL MID TX	
256	ELEV CAN SCROLL BOT TX	
512	ELEV CAN SCROLL SIGN TX	
1024	HIDE ON MAP	
2048	SHOW AS NORMAL ON MAP	i.e. light green
4096	SIGN ANCHORED	
8192	WALL DAMAGES PLAYER	
16384	SHOW AS LEDGE ON MAP	i.e. dark green
32768	SHOW AS DOOR ON MAP	i.e. yellow

FLAG 2

is unused.

FLAG 3

Bit	Description	Comment
1	CAN ALWAYS WALK	Player will climb any height
2	PLAYER & ENEMIES CANNOT WALK THROUGH WALL	
4	ENEMIES ONLY CANNOT WALK THROUGH WALL	
8	CANNOT FIRE THROUGH WALL	

INF Files

INF Files

INF files control the dynamic workings of a level. They are text files written in "The INF programming language".

INFs accept C like /* */ comments.

They are made up of **item** definitions, which are linked
to the SCs via the SC names
to the WLs via the SC names and WL number

INF File Format
The INF programming language

INF File Format

Here is the header of the INF file:

```
|  INF 1.0
|  LEVELNAME SECBASE
|
|  items 2
```

INF File version and level name, followed by total number of items in the file.
Don't forget to change this value when you add or remove items in an INF.

Then follow the items:

```
|  item: sector  name: secname
|    seq
|     ........
|    seqend
```

A Sector Item.

```
|  item: sector  name: secname  num: #wallnum
|    seq
|     ........
|    seqend
```

A Wall Item.

etc.

See also item: level

Each item follows the same format, structured by the **seq** and **seqend** statements, between which the definitions are contained.

Note:
More than one class statement is allowed per item.

INF item: level

This is used to play entire level ambient sounds. Quite useless, but it might be needed if you've got a level with lots of water, wind or machinery and it saves you from putting lots of sound objects all over the place.

This is never successfully used in the original levels. There is, however, a failed attempt in EXECUTOR.INF which is where I found out about it from.

usage:

```
| item: level
|  seq
|     amb_sound: [voc file] [num] [num]
|  seqend
```

I'm not sure what the 2 nums do, but including them seems to stop the sound from playing.

INF Programming Language

Each item (apart from an item: level) will have one or more **classes**. There are 3 types of classes:

elevators	They dynamically modify sectors and walls
triggers	When triggered, they trigger something in their clients
teleporter chute	A very special item used to "fall" to another sector

Each class will have several variables that can be customized to change how the class functions.

Messages can be sent around a level to modify sectors, walls, and INF items.

There are a few special functions that can be executed: create an adjoin:, page: a sound, and display a text: message.

See also some new INF functions that weren't used in the original levels, but were found in DARK.EXE

INF Elevators

Elevators make sectors and walls dynamic. They can obviously be used to create lifts, platforms, doors etc., but you often also need dummy (i.e. non-accessible) elevators for level control purposes.

Elevators will usually have stops, which are different values the elevator can arrive at.

Elevators may also have slaves copying their actions.

Here are the elevator classes:
elevator change_light
elevator basic
elevator inv

elevator move_floor
elevator move_ceiling
elevator move_fc
elevator scroll_floor
elevator scroll_ceiling
elevator move_offset
elevator basic_auto

elevator change_wall_light
elevator morph_move1
elevator morph_move2
elevator morph_spin1
elevator morph_spin2
elevator move_wall
elevator rotate_wall
elevator scroll_wall

elevator door
elevator door_mid
elevator door_inv

INF Triggers

Triggers send a message to a client sector when triggered. They can be used to create switches, tripwires etc. Triggers can also be used to display text.

Note: if no message is specified, then the default message (m_trigger) will be sent to the client(s).

Here are the trigger classes:
trigger standard
trigger
trigger switch1
trigger single
trigger toggle

INF Teleporter Chutes

Teleporter chutes are a special class of their own. Their function is to teleport the player directly up or down to another sector.

Dark Forces teleporter chutes are not deliberate teleporters like in DOOM. They are usually not intended to be noticed, and are intended to make it look like the player has just fallen through a chute into a layer below, for example, in the Robotics Facility where you fall into the gas room, and Jabba's Ship where you fall into the area where you rescue Jan. These cases need to use teleporter chutes because it is impossible to use the same sector in both layers - its walls would need to be given double adjoins!

Because teleporter chutes send you to the same X and Z coordinates, the target sector MUST occupy the same physical space of the teleporter chute, or it may be possible to teleport outside of a sector. Of course your Y coordinate can change.

usage:

```
| class: teleporter chute
|  target: [target sectorname]
```

INF Variables

Variables set how elevators and triggers function.

master:
event_mask:
event:
entity_mask:
speed:
start:
center:
angle:
key:
flags:
sound:
object_mask:

INF Messages

Messages are sent from triggers when they are triggered and elevators when they arrive at stops.
They are sent to other triggers and elevators, and in some cases just regular sectors and lines (except **message: lights**, which is sent to the **system**). They do various things to their recipients.
Messages are placed in the sequence of elevators and triggers.

Messages all have these general syntax:

(sent from an elevator)

```
| message: [stop number] [receiver] [message] [parameters]
```

(sent from a trigger)

```
| client: [receiver]
| message: [message] [parameters]
```

`[receiver]` is the receiver of a message. Can be one of the following:

`[sectorname]`	receiver is a sector
`[sectorname([wallnum])]`	receiver is a wall
`SYSTEM`	receiver is the SYSTEM (message: lights only)

`[parameters]` are parameters specific to the type of message.

Here are the messages:
m_trigger
goto_stop
next_stop

prev_stop
master_on
master_off
clear_bits
set_bits
complete
done
wakeup
lights

Remember that when you look at an INF file and you see something like :

```
| class: elevator eeeee
|  stop: 0
|   message: 0 mmmmm
|  stop: 1
|   message: 1 mmmmm
```

it's only a visual clue, and you could group all the messages in one place and in any order.
Important : if you add a stop, you have to renumber !

Notes:
- When a specific message is not specified, the default message is m_trigger.
- When messages are sent from an elevator, they are sent when it ARRIVES at a stop.
- For some reason, messages can't be sent from "terminate" or "complete" stops.

New INF functions

Elevators:
elevator move_offset
elevator basic_auto
elevator move_wall
elevator door_inv

Variables:
object_mask:

Special Functions:
texture:

Here are some INF keywords that were found in DARK.EXE but as yet are not understood.
We would appreciate it if people could help work out these as they may be usable!

stop_y:
trigger_action:
condition:

enclosed
mid

entity_enter
move:

GOL Files

GOL files control the functioning of the objective screen in the PDA. They contain a list of mission goals which are shown to be completed in the objective screen when they are fired. Note that the objective screen is an ANIM in DFBRIEF.LFD. It is composed of a number of DELTs (with yellow text) which are overlaid on the first embedded DELT (which has green text) as goals are completed.

```
| GOL 1.0
| GOAL: 0       ITEM:   5       # DT weapon
| GOAL: 1       TRIG:   1
```

Each GOAL: can be a TRIG: -- a goal trigger, or an ITEM: -- a goal item.

Goal triggers

These are goals that are fired by the .INF file when a "complete" message is sent. The message will fire the appropriate goal in the GOL file.

For instance, "message: [stop] [recipient] complete 1" will say that "TRIG: 1" is complete!

Goal items

These are goals which are fired when you pick up a goal item. The logics of the goal items fire an internal message to the GOL when the item is picked up.

Each goal item has a num of its own:

Goal item	*Description*	*Num*
`LOGIC: PLANS`	Death Star plans	0
`LOGIC: PHRIK`	Phrik metal	1
`LOGIC: NAVA`	Nava Card	2
`LOGIC: DATATAPE`	data tapes	4
`LOGIC: DT_WEAPON`	broken DT weapon	5
`LOGIC: PILE`	Your Gear	6

For instance, picking up a broken DT weapon will say that "ITEM: 5" is complete!
Notice this implies that you can only use **one** of each goal item in each level.

Note: the goal items will also move an elevator called "complete" to its next stop when picked up.

Managing Goals

The best way to handle goals is to use elevator "complete" only for mission goal/completion handling. It should have a number of "hold" stops and a final "complete" stop. Each goal you accomplish will move elevator "complete" one stop forward, until accomplishing the final goal moves it onto its "complete" stop, completing the level. Goal triggers will move elevator "complete" if the "complete" message is sent to elevator "complete" (because the "complete" message also moves its recipient to its next stop). Goal items automatically move elevator "complete" when they are picked up, as mentioned above.

Don't get confused with the 3 different "completes" ! One is a message, one is the name of an elevator, and one is a stop option just like "hold".

Final note: don't assume the goals will happen in the .GOL order ! Ordering completion of goals is something you need to do yourself as part of your level design !

O Files

O Files

O files contain all the level objects. They are in text format.

There are many different object types in Dark Forces:

Type	**File**	**Description**
SPIRIT	[none]	an object not linked to a viewable file (i.e. invisible) Its main use is for the PLAYER, but you can create other invisible items.
SAFE	[none]	a restart point after the player died. You should put SAFEs in your levels, to allow the player to restart not far from where he died.
SPRITE	WAX	fully animated objects such as enemies.
FRAME	FME	"one view" objects such as energy power ups.
3D	3DO	3D objects such as mousebots.
SOUND	VOC	an ambient sound around the object position.

File Format

O File Format

They are composed of 3 parts :
- Magic and Version number and level name
- Objects Tables
- Object Descriptions
  - Sequences and Logics
  - Generators
  - Full Logics list

They accept C like /* */ **comments**.

O Magic, Version Number and Level Name

```
| O 1.1
```

This is trivial.

```
| LEVELNAME SECBASE
```

I'm not sure this level name is used in DF !

O Object Tables

As there is a lot of OB information in a level, 4 object tables are created to avoid storing OB names in full at each occurrence.

```
| PODS 3             # These are the "3D" objects
|  POD: DEATH.3DO    # 00
|  ...
|
| SPRS 10            # These are the SPRITES
|  SPR: OFFCFIN.WAX  # 00
|   ...
|
| FMES 6             # These are the FRAMES
|  FME: IENERGY.FME  # 00
|  ...
|
| SOUNDS 1           # These are the SOUNDS
|  SOUND: BANG.VOC   #00
|  ...
```

Afterwards, all the objects are referred to by their 0 based index in the object tables. The object CLASS determines in which table to look.

O Object Descriptions

The first data is the total number of objects in the level :

```
| OBJECTS 185
```

Then each object is described.
Please note that the object data first line has been split here for visual convenience.

```
| CLASS: SPIRIT  DATA:  0 X: 131.00 Y:    0.00    Z: 210.00
\                       PCH:   0.00 YAW: 176.34 ROL:   0.00
\                       DIFF: 1
|  SEQ
|   LOGIC:     PLAYER
|   EYE:       TRUE
|  SEQEND
|
| CLASS: SPRITE  DATA:  0 X: 320.62  Y:  20.00   Z:  275.64
\                       PCH: 0.00  YAW: 270.00 ROL:    0.00
\                       DIFF: 1
|  SEQ
|    TYPE:       I_OFFICER
|   SEQEND
```

CLASS is the type of object, and DATA is the offset in the corresponding object table.
(SPIRIT and SAFE have DATA = 0).

X, Y, Z are trivial.

PCH, YAW, ROL are classic spatial orientation, but only YAW is really used (DOOM equivalent is THING orientation). It takes a value in degrees where 0 is at the "top of the screen when you look at the map". The value increases clockwise. PCH and ROL are only needed for 3D objects.

DIFF is the difficulty level at which the object appears.

DIFF	EASY	MED	HARD
-3	X	X	X
-2	X	X	
-1	X		
0	X	X	X
1	X	X	X
2		X	X
3			X

O Object Sequences and Logics

Doom note

SEQ and SEQEND are delimiters for a series of options/modifiers to apply to the object, which determine its behavior.

The basic thing that all entities will have is a LOGIC: that controls it (eg. for an enemy, tells it what direction to walk in, when to shoot and so on). Logics are hardcoded in DARK.EXE and also determine things like how fast an enemy moves, how it attacks, how strong it is, what sounds it makes, what weapon it drops when it dies etc. In addition, logics will control what the sprite appears to be doing (i.e. what frames in the WAX that are shown).

See Full Logics list.

The same viewable file may be used to create 'different' objects. For instance, OFFCFIN.WAX may be used with a LOGIC: I_OFFICER or LOGIC: I_OFFICERR (note the second 'R') which will generate a red key then killed instead of the usual ammo clip. Or you can use it with LOGIC: STORM1 and although the enemy will appear like an officer, it will behave as a stormtrooper, take as many shots to kill as a stormtrooper etc.

The keywords TYPE: and LOGIC: are freely exchangeable, and the ITEM keyword is optional before item logics.

Combined Logics

You can combine logics freely. LOGIC: ANIM is frequently combined with many of the item and scenery logics to animate the sprites.

If you combine enemy logics, the first LOGIC: is in this case the primary logic, which means that to kill the object, you have to use the firepower needed to kill its first LOGIC: . Very strange things may happen when combining LOGICs, and some combinations don't work, or even don't work every time!
Try Mousebot + Barrel, or Player + Mousebot...

O Generators

Generators cause enemies to appear mid-way through a level. Here is a quite self explaining example:

```
| CLASS: SPRITE  DATA:  4 X: 396.88   Y:  -2.00   Z: 217.48
\                      PCH:   0.00 YAW:   0.00 ROL:   0.00
\                     DIFF: 1
|  SEQ
|   LOGIC:     GENERATOR STORM1
|   DELAY:     30
|   INTERVAL:  20
|   MIN_DIST:  70
|   MAX_DIST:  200
|   MAX_ALIVE: 3
|   NUM_TERMINATE: 8
|   WANDER_TIME: 40
|  SEQEND
```

All generated enemies will use the sprite defined, and will appear "awake" (i.e. walking around, not standing still) from the X, Y and Z coordinates of the generator.

LOGIC: is the logic that the generated sprites will have. Note the GENERATOR keyword. Note also that only the following logics are allowed to be generated (generating others will cause problems and usually crash the game!)

```
I_OFFICER and key variations
TROOP
STORM1
COMMANDO
BOSSK
G_GUARD
REE_YEES
REE_YEES2
SEWER1
INT_DROID
PROBE_DROID
REMOTE
```

DELAY: is the time in seconds that needs to pass from the start of a level before the generator starts operating.

INTERVAL: is the time in seconds between each generation.

For an enemy to be generated, the **player** must be at a distance from the generator that is between MIN_DIST and MAX_DIST.

MAX_ALIVE: is the maximum number of enemies from the generator allowed alive at the same time.

NUM_TERMINATE: is the number of enemies to be generated. When this is reached, the generator deactivates. If set to -1, an infinite amount will be generated, and the generator will never deactivate.

WANDER_TIME: is the time in seconds that a generated sprite walks around before becoming inactive.

Note: in DARK.EXE, there is a keyword "PLUGIN:" among the above generator keywords. Its usage is still unknown.

Sprites aren't generated when the generator is able to see you, however (otherwise it would look like the enemies were walking out of thin air!). The best way to observe a generator working is therefore on the map by using the LACDS cheat.

Also note that you can set MASTER: OFF on a generator (not to be confused with the INF master variable!), and activate it by sending a "master_on" message to the sector that contains it.

O All Object Logics

This is a list of all the objects and other modifiers that can be used in the sequences of objects.

Please also see the end of this section for some unknowns found in DARK.EXE.

Player

```
| LOGIC:  PLAYER
| EYE:    TRUE
```

These should always be used together. Technically though, the LOGIC: PLAYER is the entity that you will control and move around, while EYE: TRUE is the object from whose point of view the level is viewed from. So yes, you can try following enemies and mousebots around with the eye......

Items

Remember that you can use ITEM keyword before these logics.
Message is the message number from TEXT.MSG that is displayed when you pick up the item (just in case you want to patch).

Logic:	**Description:**	**Message:**
General -		
`| LOGIC: SHIELD`	20 shield units	114
`| LOGIC: BATTERY`	battery unit	211
`| LOGIC: CLEATS`	ice cleats	304
`| LOGIC: GOGGLES`	infra red goggles	303
`| LOGIC: MASK`	gas mask	305
`| LOGIC: MEDKIT`	med kit	311
Weapons -		
`| LOGIC: RIFLE`	Blaster rifle / 15 energy units	100 / 101
`| LOGIC: AUTOGUN`	Repeater Rifle / 30 power units	103 / 104
`| LOGIC: FUSION`	Jeron fusion cutter / 50 power units	107 / 108
`| LOGIC: MORTAR`	Mortar Gun / 3 mortar shells	105 / 106
`| LOGIC: CONCUSSION`	Concussion Rifle / 100 power units	110 / 111
`| LOGIC: CANNON`	Assault cannon / 30 plasma units	112 / 113
Ammo -		
`| LOGIC: ENERGY`	15 energy units	200
`| LOGIC: DETONATOR`	1 thermal detonator	203
`| LOGIC: DETONATORS`	5 thermal detonators	204
`| LOGIC: POWER`	10 power units	201
`| LOGIC: MINE`	1 mine	207
`| LOGIC: MINES`	5 mines	208
`| LOGIC: SHELL`	1 mortar shell	205
`| LOGIC: SHELLS`	5 mortar shells	206
`| LOGIC: PLASMA`	20 Plasma units	202
`| LOGIC: MISSILE`	1 missile	209
`| LOGIC: MISSILES`	5 missiles	210
Bonuses -		

`| LOGIC: SUPERCHARGE`	weapon supercharge	307
`| LOGIC: INVINCIBLE`	shield supercharge	306
`| LOGIC: LIFE`	extra life	310
`| LOGIC: REVIVE`	revive	308
Keys -		
`| LOGIC: BLUE`	blue key	302
`| LOGIC: RED`	red key	300
`| LOGIC: YELLOW`	yellow key	301
`| LOGIC: CODE1`	code key 1	501
`| LOGIC: CODE2`	code key 2	502
`| LOGIC: CODE3`	code key 3	503
`| LOGIC: CODE4`	code key 4	504
`| LOGIC: CODE5`	code key 5	505
`| LOGIC: CODE6`	code key 6	506
`| LOGIC: CODE7`	code key 7	507
`| LOGIC: CODE8`	code key 8	508
`| LOGIC: CODE9`	code key 9	509
Goal items -		
`| LOGIC: DATATAPE`	data tapes	406
`| LOGIC: PLANS`	Death Star plans	400
`| LOGIC: DT_WEAPON`	broken DT weapon	405
`| LOGIC: NAVA`	Nava Card	402
`| LOGIC: PHRIK`	Phrik metal	401
`| LOGIC: PILE`	Your Gear	312

Enemy logics

Logic:	Description:
Imperials -	
`| LOGIC: I_OFFICER`	Imperial officer
`| LOGIC: I_OFFICERR`	Officer with red key
`| LOGIC: I_OFFICERB`	Officer with blue key
`| LOGIC: I_OFFICERY`	Officer with yellow key
`| LOGIC: I_OFFICER1`	Officer with code key 1
`| LOGIC: I_OFFICER2`	Officer with code key 2
`| LOGIC: I_OFFICER3`	Officer with code key 3
`| LOGIC: I_OFFICER4`	Officer with code key 4
`| LOGIC: I_OFFICER5`	Officer with code key 5
`| LOGIC: I_OFFICER6`	Officer with code key 6
`| LOGIC: I_OFFICER7`	Officer with code key 7
`| LOGIC: I_OFFICER8`	Officer with code key 8
`| LOGIC: I_OFFICER9`	Officer with code key 9
`| LOGIC: TROOP`	Stormtrooper
`| LOGIC: STORM1`	Stormtrooper
`| LOGIC: COMMANDO`	Imperial Commando
Aliens -	
`| LOGIC: BOSSK`	Bossk
`| LOGIC: G_GUARD`	Gammorean Guard
`| LOGIC: REE_YEES`	ReeYees with thermal detonators
`| LOGIC: REE_YEES2`	ReeYees w/o thermal detonators
`| LOGIC: SEWER1`	Sewer creature
Robots -	
`| LOGIC: INT_DROID`	Interrogator droid
`| LOGIC: PROBE_DROID`	Probe droid
`| LOGIC: REMOTE`	Remote
Bosses -	
`| LOGIC: BOBA_FETT`	Boba Fett
`| LOGIC: KELL`	Kell Dragon
`| LOGIC: D_TROOP1`	Phase 1 Dark Trooper

| LOGIC: D_TROOP2	Phase 2 Dark Trooper
| LOGIC: D_TROOP3	Phase 3 Dark Trooper (Mohc)

Special sprite logics

Note: The WAX files used for the explosions of the Barrel and Land Mine are hardcoded.

Logic:	Description:
| LOGIC: SCENERY	Displays first cell of wax 0, then all of wax 1 when attacked
| LOGIC: ANIM	Displays wax 0 over and over
| LOGIC: BARREL	Power Generating unit
| LOGIC: LAND_MINE	Land mine

3D object logics

Logic:	Description:
| LOGIC: TURRET	gun turret
| LOGIC: MOUSEBOT	mousebot
| LOGIC: WELDER	welding arm

3D object motion logics

There are 2 logics for giving motions to a 3D object:
LOGIC: UPDATE to perpetually rotate a 3D, and
LOGIC: KEY to give a VUE motion to the 3D

Rotation on X-axis
```
| LOGIC:    UPDATE
| FLAGS:    8
| D_PITCH:  [speed]
```

Rotation on Y-axis
```
| LOGIC:    UPDATE
| FLAGS:    16
| D_YAW:    [speed]
```

Rotation on Z-axis
```
| LOGIC:    UPDATE
| FLAGS:    32
| D_ROLL:   [speed]
```

Speed is the speed at which the 3D object rotates from -999 (max anti-clockwise) to 999 (max clockwise).

VUE object

```
| LOGIC:      KEY
| VUE:        filename.VUE "id"
| VUE_APPEND: filenam2.VUE "id"
| PAUSE:      TRUE
| FRAME_RATE: [frame rate]
```

filename.VUE is the name of the VUE file to use.

"id" is the name of the identifier within the VUE file to use.

VUE_APPEND: is an optional VUE to be played after the first VUE.

PAUSE: TRUE will cause the VUE to pause each time it is played until a "wakeup" message is sent to the sector containing the 3D object. Objects with "PAUSE: TRUE" will also be "woken up" if their RADIUS is shot.

Frame rate is in frames per second.

Other sequence modifiers

```
| BOSS:  TRUE
```

This can be set to the following logics:

```
    BOBA_FETT
    KELL
    D_TROOP1
    D_TROOP2
    D_TROOP3
```

When you kill the enemy, an elevator called "boss" will move to its next stop (unless it is LOGIC: D_TROOP3, where the elevator must be called "mohc"). This is similar to the movement of "complete" when a goal item is picked up. Using this modifier, you can cause something to happen when the player has killed the boss, for instance the player could be locked in a certain area until he has killed the boss and then a door will be opened letting him out.

```
| RADIUS:   [horizontal distance]
```

This defines the size of an invisible circle around the object where the PLAYER cannot enter or shoot through. Frames and sprites have radiuses by default, but 3D objects don't, so you have to set one unless you want the PLAYER to walk right through. You can use this with a Spirit to create an invisible obstacle.

```
| HEIGHT:   [vertical distance]
```

Similar to radius, height defines an area above (positive value) or below (negative value) an object where you can't walk or fire through. Therefore, using radius and height together, you can effectively create an impenetrable cylinder-shaped area around an object.

A further note:
RADIUS and HEIGHT, if used with objects having a logic, will also affect how the logic interacts with the player. If used with items, they determine the distance Kyle has to be from the item to pick it up. If used with enemies and "LOGIC: SCENERY", they determine the distance from the enemy that laser bolts etc. have to come within to damage the enemy.

Unknown

These are found in DARK.EXE. It is likely that some are only used internally by the DF engine. We would appreciate any help working out any possible usable ones!

```
VISIBLE:
SHADED:
LIGHT:
PARENT:
D_X:
D_Y:
D_Z:
D_VIEW_PITCH:
D_VIEW_YAW:
D_VIEW_ROLL:
VIEW_PITCH:
```

```
VIEW_YAW:
VIEW_ROLL:
EYE_D_XYZ:
EYE_D_PYR:
SYNC:
PLUGIN:

STORM
DISPATCH
THINKER
FOLLOW
FOLLOW_Y
RANDOM_YAW
MOVER
SHAKER
PERSONALITY
```

PAL Files

PAL Files

PAL files are 768 bytes long, and store a single 256 colors palette as 256 x 3 RGB bytes.

```
PAL_File IS
{
colors       RGB_Color[256]    // 256 colors from 0 to 255
}
```

Where:

```
RGB_Color Is
{
R            byte              // Red intensity
G            byte              // Green intensity
B            byte              // Blue intensity
}
```

Note that these intensities range from 0 to 63 (limit of VGA mode 0x13) in the PAL files.

Each level in Dark Forces can have its own palette, specified in the LEV header.

CMP Files

CMP files store palette mappings (like DOOM COLORMAP), which are the way the colors behave under different light levels.

Each byte is an index into the palette, and the CMP file is in fact an array that can be used for shading.

Given a color value [C] and a light value [L] the correct color to draw can be determined with the following formula:

```
DrawColor = [start_of_CMP_file]+(256*L)+C
```

There are 128 added bytes at the end of the file generally forming a slow gradient from 0x00 to 0x1F.
Those serve to modify light values when you use the headlight (or when firing a weapon lights the area).
The first of the 128 bytes controls the area right next to you and each one after that control an area progressively further away.
0x00 is the maximum illumination while 0x1f is minimum for the headlight.
Values above 0x1f and up to 0x27 serve to suppress the weapon lighting effect too.

The only use I see for those is to set them all to 0x1f to suppress the headlight altogether.
It doesn't seem logical to suppress the weapon lighting, although it can be done too...

FNT Files

These files store a proportional character set. FNT files are found in DARK.GOB
This set may not be complete. The font used to display the ammo, for instance contains only the numbers and the ':' character.

```
FNT_Header IS
{
 Magic      char[4]              // 'FNT' + 15h (21d)
 Height     byte                 // Height of the font
 u1         byte                 // Unknown
 DataSize   int                  // Data after header
 First      byte                 // First character in font
 Last       byte                 // Last character in font
 pad1       byte[22]             // 22 times 0x00
}
```

Then follow the characters.
There is (Last-First+1) FNT_Character blocks (one per character).

```
FNT_Character IS
{
 Width      byte                 // Width of the character
 Picture    byte[Width*Height]   // Bytes describing the character,
                                 // encoded by columns from bottom to top
                                 // Each byte is an index in the
                                 // current PAL palette
}
```

BM Files

BM files store textures used in a variety of ways in DF.
They serve as wall textures, as floor and ceiling textures (in which case they must be 64*64), as 3DO facet textures, as weapons, and as the Heads Up Display.

Here is the data structure for the BM file header.

```
BM_Header IS
{
MAGIC          char[4]     // = 'BM ' + 0x1E
SizeX          int         // if = 1 then multiple BM in the file
SizeY          int         // EXCEPT if SizeY also = 1, in which case
                           // it is a 1x1 BM
idemX          int         // unused by engine
idemY          int         // unused by engine
Transparent    byte        // 0x36 for normal
                           // 0x3E for transparent
                           // 0x08 for weapons
logSizeY       byte        // logSizeY = log2(SizeY)
                           // logSizeY = 0 for weapons
Compressed     int         // 0 = not compressed
                           // 1 = compressed (RLE)
                           // 2 = compressed (RLE0)
DataSize       long        // Data size for compressed BM
                           // excluding header and columns starts table
                           // If not compressed, DataSize is unused
pad1           byte[12]    // 12 times 0x00
}
```

Please note that BM must have height and width which are powers of 2 (except weapons).
The data follows, encoded by COLUMNS from the bottom to the top.

See also:
- Transparent BM
- Multiple BM
- Compressed BM

BM Transparent, Multiple

You can transform any BM in a transparent BM by changing its Transparent value from 0x36 to 0x3E.
The color 0 will 'disappear' and you will be able to see through it if it is a MID texture on an adjoined wall.
Note that this isn't the same as DOOM transparent textures (which use something very similar to RLE0).

Note that weapons BM use 0x08 for their transparent value, so maybe the transparent byte is a collection of flags, where the bit 3 means transparent.

BM Multiple

If SizeX = 1 (EXCEPT if SizeY = 1 in which case it is a 1*1 BM) the BM file is multiple.

The header of multiple BMs is different from that of a normal BM.

```
BM_Multiple_Header IS
{
```

```
MAGIC          char[4]     // = 'BM ' + 0x1E
SizeX          int         // = 1
SizeY          int         // = length of file - 32
idemX          int         // = -2
idemY          int         // number of 'sub' BMs
Transparent    byte
logSizeY       byte
Compressed     int
DataSize       long
pad1           byte[12]    // 12 times 0x00
}
```

Straight after the Multiple BM header are two bytes:
- The first is either the **frame rate** (in frames per second) of an **animated texture**, or is **0** to designate a **switch**. You may alter this value if you want.
- The second byte is 2.

Then follows a table of offsets to the 'sub' BM composed of idemY long.
The simple fact that this table exists tells us that sub BMs of different sizes may be stored.

Each 'sub' BM then has its own header, slightly different from the BM_Header:

```
BM_SUBHeader IS
{
SizeX          int         // horizontal size
SizeY          int         // vertical size
idemX          int         // unused by engine
idemY          int         // unused by engine
DataSize       long        // unused (no compression allowed)
logSizeY       byte        // logSizeY = log2(SizeY)
pad1           byte[3]
u1             byte[3]     // these are always filled, but they seem //
                           to be unused
pad2           byte[5]
Transparent    byte        // 0x36 for normal
                           // 0x3E for transparent
pad3           byte[3]
}
```

Important notes
1) There is no MAGIC field.
2) For a multiple BM to work correctly, it must be made a SIGN, and for switches there MUST also exist a corresponding trigger in the .INF Else, switches will be displayed wrong (as a single column) and the animated will display correctly, but static.
This means that you cannot do animated floors and ceilings this way !
3) The multiple BMs are limited to 64K in size because SizeY contains the size of the file - 32 and is an int.
Although it should never be a problem with switches, this means that you must use animated BMs for small textures only.

A solution that allows animated walls of any size AND animated floors and ceilings is to compose a huge texture with your multiple images pasted next to each other. Then use INF elevators to scroll wall or scroll floor/ceiling using the offsets of the images as stops. If you set a speed of 0, the change will be instantaneous, and the effect will be the same. An added bonus is that you'll also have complete control on starting/stopping the animation.

BM Compressed

If Compressed = 1 or 2, the BM is compressed.
These existed in the DEMO (buyit.bm, Compressed = 1; wait.bm, Compressed = 2), but there aren't any in the full game.
The engine still supports them however, so here are their descriptions.
Note that Multiple BMs don't allow compression.
(thanks to Alex Novikov for corrections and improvements on these notions).

The heart of the data is a **columns starts** table, with the start addresses of each of the columns. It is at the end of the file, at offset DataSize, and has one long entry per column containing this column start address.
This start address is calculated without the 32 bytes BM header (i.e. read the header in a struct, then the data in a **huge buffer** at offset 0).

Compressed = 1 (RLE)

The coding of one column follows (in pseudo code format).

```
while(end of data for this column not reached)
{
 if(buffer[address] <= 128)
   the FOLLOWING n bytes are direct values
 else
   the FOLLOWING byte is a color byte to repeat n-128 times
}
```

So, for example, the following hex values ...88 02 17 28 82... mean:
write 8 pixels of color 02, then write 17 pixels with colors 28, 82, etc.

This should be the format of choice for non-transparent BMs.

Compressed = 2 (RLE0)

The coding of one column follows (in pseudo code format).

```
while(end of data for this column not reached)
{
 if(buffer[address] <= 128)
   the FOLLOWING n bytes are direct values
 else
   skip n-128 transparent (background) pixels
}
```

So, for example, the following hex values ...88 02 17 28 82... mean:
skip 8 background pixels, then write two pixels with colors 17 and 28, then skip 2 background pixels, etc.

This should be the format of choice for transparent BMs.

FME Files

They contain the frames, which are the "one view" objects (you can turn around them, and you always see the same image).

Here are the data structures for the FME file headers.

```
FME_Header1 IS
{
InsertX       long        // Insertion point, X coordinate
```

```
                              // Negative values shift the FME left
                              // Positive values shift the FME right
InsertY        long           // Insertion point, Y coordinate
                              // Negative values shift the FME up
                              // Positive values shift the FME down
Flip           long           // 0 = not flipped
                              // 1 = flipped horizontally
Header2        long           // pointer to FME_Header2
UnitWidth      long           // Unused
UnitHeight     long           // Unused
pad3           long           // Unused
pad4           long           // Unused
}


FME_Header2 IS
{
SizeX          long           // Size of the FME, X value
SizeY          long           // Size of the FME, Y value
Compressed     long           // 0 = not compressed
                              // 1 = compressed
DataSize       long           // Datasize for compressed FMEs,
                              // equals length of the FME file - 32
                              // If not compressed, DataSize = 0
ColOffs        long           // Always 0, because columns table
                              // follows just after
pad1           long           // Unused
}
```

If Compressed = 0, the data follows, encoded by COLUMNS from the bottom to the top.

Compressed FME

Compressed FMEs are very similar to compressed BMs (RLE0).

After FME_Header2 follows a table of offsets to the starts of the columns data.
Those are offsets from the start of FME_Header2.

Then follow the columns data.

The coding of one column follows (in pseudo code format).

```
while(end of data for this column not reached)
{
 if(buffer[address] <= 128)
   the FOLLOWING n bytes are direct values
 else
   skip n-128 transparent (background) pixels
}
```

So, for example, the following hex values ...88 02 17 28 82... mean:
skip 8 background pixels, then write two pixels with colors 17 and 28, then skip 2 background pixels, etc.

WAX Files

They contain the sprites.
(samples : STORMTROOPERS, BONUS LIVES ...)

They are a collection of embedded CELLS (FME files stripped of their FME_Header1).

```
WAX file structure:

WAX_Header IS
{
 Version       long        // constant = 0x00100100
 Nseqs         long        // number of SEQUENCES
 Nframes       long        // number of FRAMES
 Ncells        long        // number of CELLS
 Xscale        long        // unused
 Yscale        long        // unused
 XtraLight     long        // unused
 pad4          long        // unused
 WAXES         long[32]    // pointers to WAXES
                           // = different actions
}

WAX IS
{
 Wwidth        long        // World Width
 Wheight       long        // World Height
 FrameRate     long        // Frames per second
 Nframes       long        // unused = 0
 pad2          long        // unused = 0
 pad3          long        // unused = 0
 pad4          long        // unused = 0
 SEQS          long[32]    // pointers to SEQUENCES
                           // = views from different angles
}
```

Note: World Width and World Height are values which define how big the sprite actually appears in-game.

```
SEQUENCE IS
{
 pad1          long        // unused = 0
 pad2          long        // unused = 0
 pad3          long        // unused = 0
 pad4          long        // unused = 0
 FRAMES        long[32]    // pointers to FRAMES
                           // = the animation frames
}

FRAME IS
{
InsertX        long        // Insertion point, X coordinate
                           // Negative values shift the cell left
                           // Positive values shift the cell right
InsertY        long        // Insertion point, Y coordinate
                           // Negative values shift the cell up
                           // Positive values shift the cell down
Flip           long        // 0 = not flipped
                           // 1 = flipped horizontally
Cell           long        // pointer to CELL
```

```
                             // = single picture
UnitWidth      long          // Unused
UnitHeight     long          // Unused
pad3           long          // Unused
pad4           long          // Unused
}

CELL_Header IS
{
SizeX          long          // Size of the CELL, X value
SizeY          long          // Size of the CELL, Y value
Compressed     long          // 0 = not compressed
                             // 1 = compressed
DataSize       long          // Datasize for compressed CELL,
                             // equals length of the CELL
                             // If not compressed, DataSize = 0
ColOffs        long          // Always 0, because columns table
                             // follows just after
pad1           long          // Unused
}
```

An explanation of how it all works:

The 32 WAXes pointed to by the .WAX file header are 32 possible states that the sprite can be in (usually only up to 14 are used). The logic controls what WAX is shown when, so that the sprite appears to be doing what the logic actually is doing.

All enemies apart from the REMOTE follow this general pattern:

WAX #	**state**
0	moving -- eg. walking, floating
1	attacking (primary)
2	dying (from punch)
3	dying (from shot or explosion)
4	lying dead
5	staying still (i.e. not sited player yet)
6	follow through of primary attack -- eg. kick from gun
7	secondary attack -- eg. TD for reeyees, green junk for int. droid, ...
8	follow through of secondary attack
9	jump (Kell Dragon)
10	
11	
12	getting injured (dianoga looking around)
13	special action Using shield for D_TROOP1, flying for D_TROOP2 and D_TROOP3, submerging for dianoga, ...

Note: The Phase 3 varies from this pattern quite a bit.
Where a state doesn't apply for a particular enemy logic, the WAX will usually just be the enemy walking or moving towards you. It won't be called for by the logic.

The remote has 4 states:

WAX #	**state**
0	moving
1	staying still -- before siting player
2	dying
3	dying

LOGIC: SCENERY has 2 simple states:

WAX #	**state**
0	normal
1	attacked

LOGIC: BARREL has 2 states:

WAX #	**state**
0	normal
1	exploding

LOGIC: ANIM, as well as weapon projectiles, explosions, splashing water etc. have 1 continuous state.

The 32 pointers to SEQUENCES in each WAX structure point to the view of the WAX (state) from 32 different angles as you move around it (0, 11.25, 22.50....348.75). The first pointer (angle 0) is when the logic is facing you.

The pointers to FRAMEs in each SEQUENCE structure point to the FRAMEs that make up an animation sequence for each point of view. FRAMEs are the header 1 of FME files.
The SEQUENCE consists of 32 FRAME entries. Usually no more than 5 are used, but the dianoga has 27 frames of animation for one of its states (WAX 12, when it looks around for you) !
The entries = 0 are unused.

Each FRAME points to a CELL, which is a picture with the same format as .FME files with header 2 of FME files.

3DO Files

They contain the "3D" objects. (samples : MOUSEBOT, the DEATH STAR HOLOGRAM, ...)

They are text files containing a geometric description of a full 3D object, and are converted from 3D Studio .ASC format. They accept # comments.

3DO format: [by Michael Taylor]

```
| 3DO 1.2
```

Magic and Version Number: this is the word "3DO" followed by a version #, either 1.2, 1.20, or 1.30.

Next comes several lines of header data. Included is the picture name, number of objects in the file, number of vertice, number of polygons, palette used, and number of textures.

```
| 3DONAME  cube
| OBJECTS  00001
| VERTICES 00008
| POLYGONS 00006
| PALETTE  METAL.PAL

| TEXTURES 0
```
or
```
| TEXTURES 1
|   TEXTURE: IPDTENGR.BM
```

The palette file doesn't appear to relate to any PAL file found in the GOB directory.
[Could this be the type of rendering (metal, phong, ...) used in 3DS ? [Yves]]
[It is probably the palette used by LEC when testing the 3DOs out-of-game. In DF, the palette of the currently loaded level is used - Jereth]
Please note that textures are a little different and will be explained below.

If any textures are used then below the TEXTURES # line is additional lines defining each texture file. It creates a zero based array of textures for later usage by the objects.

See Object Definitions

3DO Object Definitions

[by Michael Taylor]

After the header data comes each object's definition. Each one starts with an object header and then the data. The object header is the word "OBJECT" followed by the object's name in double quotes. The object names seem irrelevant provided they are unique within the 3DO file. Next is the word "TEXTURE" followed by the texture used for this object. If no texture is used then the value of -1 is used else an index into the texture table defined in the header data is given.

```
| OBJECT "shuttle"
| TEXTURE 0     # Index into texture array
|               # IFOCTGR.BM
```

After the texture information, starts the actual geometric description of the object.

First comes the vertices. The initial line is the word VERTICES followed by the number of vertices defined. Then the vertices are listed starting with 0 and going up to the number of vertices listed on the VERTICES line. Each vertex is defined by 3 numbers; x, y, and z. They represent relative locations on a 3-D graph. They are taken to 3 decimal places.

```
| VERTICES 8
|    0:   0.000   2.000  -0.050
|    1: -10.000   2.000  -5.550
| ...
```

After the vertex information, comes the polygonal information. Each object may be made up of either triangles or quads.
The appropriate header and number of polygons defined are listed, TRIANGLES for triangles and QUADS for quadrilaterals.
The polygons are described with a number starting at 0, then the vertex number for each end point is given (3 for triangles and 4 for quadrilaterals). Then a color is given to each polygon (0 to 255). Finally comes the shading used for each polygon.

Note that in order to use a texture for a polygon, you must set its shading to TEXTURE.

[Here is a list and quick explanation of each of the shading types:

FLAT	Normal, flat surface
GOURAUD	Gouraud shading on surface
VERTEX	Display only vertexes of polygon (like Death Star holo)
TEXTURE	filled with a texture
GOURTEX	filled with a texture, plus gouraud shading on the texture
PLANE	texture on a horizontal plane (acts same as floor and ceiling textures -- must be 64*64, affected by flr and ceil txoffsets, and scrolled by

elevators scroll_floor and scroll_ceiling)

- Jereth]

```
| TRIANGLES 12
| # Num  V1  V2  V3  Color  Shading
|   0:   1   2   3     0     PLANE
|   1:   0   1   3     0     PLANE
|   2:   5   1   0    62     FLAT
| ...
```

Also note that the vertices are listed in clockwise order if you are facing directly at the polygon.
[This simplifies hidden lines/surfaces algorithm, as you may determine the facet orientation with 3 of them [Yves]]

[end of Michael's section]

Here is a description of TEXTURE VERTICES and TEXTURE QUADS/TRIANGLES, which Michael didn't fully cover.

If textures are used (TEXTURE, GOURTEX or PLANE shading), then texture vertices and texture triangles/quads also needed to be defined.

TEXTURE VERTICES:

These are a set of points defined on an X-Y plane, where X and Y coordinate values are >=0 and <= 1. These points define relative positions on the texture being used for the current object, eg. for a 16 x 8 texture, the following TEXTURE VERTICE...

```
|#      num:    <x>     <y>
|       0:      0.5     0.25
```

....defines a point on the texture at (8, 2) in geometry units, or (64, 8) in pixels.

TEXTURE QUADS / TEXTURE TRIANGLES:

These link texture vertices into a 3 or 4 sided polygon, hence deciding which portion of the texture is to be placed on the polygon.

For example, if you have an 16 x 8 texture, and the following 4 TEXTURE VERTICES:

```
|0:     0.00    0.00
|1:     0.00    0.50
|2:     1.00    0.50
|3:     1.00    0.00
```

and the following TEXTURE QUAD:

```
|0:     0       1       2       3
```

....the bottom half of the texture will be placed onto QUAD 0 of the object (i.e. up to an X value of 16, but only up to a Y value of 4) with the first vertice of the TEXTURE QUAD being placed on the first vertice of the QUAD, the second vertice on the second, and so on. So you can also orientate the portion of texture on the polygon any way you want by keeping the TEXTURE VERTICES pointed to in the same order, but varying the starting vertice, flip it by reversing the order of TEXTURE VERTICES pointed to, or even deform the texture by varying the order of the TEXTURE VERTICES pointed to.

```
  |     1:  2    1    0    3
```

In this example, the texture will be flipped horizontally, and be on its side relative to TEXTURE QUAD 0 (the first example).

Of course, this section of the texture will need to be scaled to cover the whole polygon, so if the polygon is, say, a 64 by 32 rectangular QUAD, the texture will be expanded by a factor of 4 for the above example. If the polygon doesn't have dimensions of the same ratio as the portion of texture, the texture portion will be warped, eg. if the polygon for the above example is shaped like a regular trapezium, the top part of the texture will be squashed and the bottom part stretched.

It is okay to point to the same texture vertices over and over again if you for example want to put the same section of a texture on more than one polygon in the object.

Note: TEXTURE VERTICES and TEXTURE QUADS / TRIANGLES are also needed for PLANE fill, although you can't decide what part of a texture is to be placed on a PLANE polygon. Hence the TEXTURE VERTICES pointed to by the TEXTURE QUAD / TRIANGLE are unused.

TEXTURE QUADS / TRIANGLES correspond with the polygons (having TEXTURE, GOURTEX or PLANE fill) that they are linked to. So if QUAD 0 and 2 of an object have a texture fill, but QUAD 1 is just gouraud or flat or otherwise, then TEXTURE QUAD 0 and 2 will be used, but TEXTURE QUAD 1 must also be defined even though it isn't used. So to be economical, you should have all polygons filled with a texture defined first within each object of the 3DO file.

VUE Files

[Slightly edited extract from VUE.TXT by Paul Nemesh]
[changed the references to "object name" to "id" which is what is used in the OFFSTVUE tool]

This what a sample .VUE looks like:

```
| vue 1
| transform "id" #1 #2 #3 #4 #5 #6 #7 #8 #9 #10 #11 #12
| transform "id" ......
```

"id" is the identifier (referenced by the .o file, see below). So you can store more than one set of 3D object motions within the one VUE, each with a different identifier.

The values for #1 through #9 are the coefficients of the rotating and scaling matrix that is used by DF to determine how to draw the .3do. [...]

The formulas are:
```
#1: Scale x [cos(H) x cos(R)]
#2: Scale x [-sin(H) x cos(P) + cos(H) x sin(R) x sin(P)]
#3: Scale x [-sin(H) x sin(P) - cos(H) x sin(R) x cos(P)]
#4: Scale x [sin(H) x cos(R)]
#5: Scale x [cos(H) x cos(P) + sin(H) x sin(R) x sin(P)]
#6: Scale x [cos(H) x sin(P) - sin(H) x sin(R) x cos(P)]
#7: Scale x [sin(R)]
#8: Scale x [-cos(R) x sin(P)]
#9: Scale x [cos(R) x cos(P)]
```

The values for #10 through #12 are:
```
#10: X coordinate
#11: Z coordinate
#12: -Y coordinate
```

The .o file should have the following logic associated with the .3do:

```
SEQ
LOGIC: KEY                          /* This always needs to be present. */
VUE: FILENAME.VUE "ID"              /* This is the filename of the .VUE, with
                                    the identifier in quotes. */
VUE_APPEND: FILENAM2.VUE "ID"       /* Same as the previous line, except this
                                    will be run directly after the first .VUE is
                                    finished. */
PAUSE: TRUE                         /* If this line is used, the .VUE will run
                                    exactly once (like Kyle's ship taking off).
                                    If this line is omitted, the .VUE will
                                    continuously repeat itself. */
SEQEND
```

[End of extract]

Apparently, the very best way to generate VUE files is to use 3D Studio, as .VUE is a standard 3DS file format, used to describe objects motion. By the way, 3DS .ASC is the base format for the 3DOs, after which the LEC team converted them.

VOC Files

These are standard .VOC files in the **Creative Labs** format.

The DF engine only accepts MONO 8-bit 11KHz (11025 Hz) .VOC files.

Note that sounds are looped (eg. the water and wind) using REPEAT/END REPEAT markers.

[by galt@dsd.es.com]

```
Creative Voice File (VOC) Format:

    HEADER (bytes 00-19)
    Series of DATA BLOCKS (bytes 1A+) [Must end w/ Terminator Block]

-----------------------------------------------------------------

HEADER:
=======
     byte #     Description
     ------     ------------------------------------------
     00-12      Creative Voice File
     13-15      1A 1A 00  (eof to abort printing of file)
     16-17      Version number (minor,major) (VOC-HDR puts 0A 01)
     18-19      2's Comp of Ver. # + 1234h (VOC-HDR puts 29 11)

-----------------------------------------------------------------

DATA BLOCK:
===========

   Data Block:  TYPE(1-byte), SIZE(3-bytes), INFO(0+ bytes)
   NOTE: Terminator Block is an exception -- it has only the TYPE byte.

      TYPE   Description     Size (3-byte int)   Info
    ----   -----------     -----------------    -----------------------
    00     Terminator      (NONE)               (NONE)
    01     Sound data      2+length of data     *
    02     Sound continue  length of data       Voice Data
    03     Silence         3                    **
    04     Marker          2                    Marker# (2 bytes)
    05     ASCII           length of string     null terminated string
    06     Repeat          2                    Count# (2 bytes)
    07     End repeat      0                    (NONE)

    *Sound Info Format:         **Silence Info Format:
     ---------------------        ----------------------------
     00   Sample Rate             00-01  Length of silence - 1
     01   Compression Type        02     Sample Rate
     02+  Voice Data


Marker#            -- Driver keeps the most recent marker in a status byte
Count#             -- Number of repetitions + 1
                        Count# may be 1 to FFFE for 0 - FFFD repetitions
                        or FFFF for endless repetitions
Sample Rate        -- SR byte = 256-(1000000/sample_rate)
Length of silence  -- in units of sampling cycle
Compression Type   -- of voice data
                        8-bits    = 0
                        4-bits    = 1
                        2.6-bits  = 2
                        2-bits    = 3
                        Multi DAC = 3+(# of channels) [interesting--
                                      this isn't in the developer's manual]
```

GMD Files

They contain the musics.

[by Alex Novikov]

The header of GMD file (or the LFD GMID resource) consists of two fields:

```
GMD_Header IS
{
 Magic         char[4]     // the string 'MIDI'
 Size          long        // Size of the whole file excluding header
                           // inverted byte order
}
```

The order of bytes in the Size field is inverted: the first byte is the highest byte, the 4th byte is the lowest byte of the value (this order is normal for Mac, but inverted for PC).

Then follow a variable number of chunks in format:

```
GMD_Chunk IS
{
 Type          char[4]     // chunk type
 Size          long        // Size of the chunk excluding header
```

```
                                  // inverted byte order
 }
```

The field Size has the inverted order of bytes - same as the field Size of the file header.

The following Chunks are encountered:

MDpg
Varied length, usually 14 (0Eh)
Very strange content - mostly doesn't change from file to file, but if it does - some new byte is INSERTED between usual ones (with chunk size preserved, so the last byte of chunk goes).

MThd
6 bytes long.
Normal MIDI header. Indicates MIDI format 2.

```
 MTHD_CHUNK IS
 {
Format        INVERTED_INT     // always 2 (MIDI2 format)
NTracks       INVERTED_INT     // Number of tracks in the file
Division      INVERTED_INT     // always 1E0h (tempo constant)
 }
```

INVERTED_INT is an INT with inverted byte order.

MTrk
Normal MIDI format 0(2) track data with the exception that "running status" (i.e. if one MIDI event followed by the same MIDI event with different parameters, the MIDI event code can omitted) is not used/supported. You cannot omit MIDI event codes.This basically means that GMD MTrk data are compatible with the MIDI standard, but MTrk from external MIDIs can be (and often are) incompatible with the GMD standard. See SMF (Standard MIDI File) specs for more info on MTrk chunk content.

The additional data in GMD's MTrk chunks is internal iMuse commands. Internal iMuse commands are stored as SysEx (System Exclusive) messages. They usually look like:

```
 F0 Size 7D 03 TEXT 00 F7

F0           identifier of SysEx message
Size         value of message size in MIDI variable length format
7D 03        probably an identifier of iMuse message
TEXT         a text string of several characters
00           string terminator
F7           SysEx message terminator
```

The encountered messages are (TEXT part):

```
start new
stalk trans #           // # is a number appears to be a float
fight trans #,#
engage trans #
from fight #,#
from stalk #,#,#
from boss #
clear callback
to X                    // X= A,B,C...
to Xslow                // X= A,B,C...
```

The number of parameters may vary. And, actually, the effect of these messages is not really known.

There are also iMuse messages beginning with 7D 01 whose format is unknown.

They seem to have something to do with looping the in-level music.

MSG Files

They contain the text messages used in the game.

text.msg	Contains in-game text messages. You can create new messages or patch existing ones. New messages can be displayed with the "TEXT:" INF function.
local.msg	Contains run-time error messages and should be left untouched.
hotkeys.msg	Contains menu hotkeys and should be left untouched

General format:

```
| MSG 1.0
|
| MSGS 119
|
| # internal game messages
| 0    0:  "Joystick Off"
| ...
| END
```

MSGS is the number of messages. Don't forget to update it if you add messages.

I found no problems by adding messages to TEXT.MSG at 900 and more.

eg.

```
| 900  1:  "Hurry up !"
```

The number followed by a colon (eg. 1:) rates the importance of the message relative to other messages in the MSG file. '0: ' is the most important, and as the number increases, the message becomes less important. If a message is currently on screen, it can be immediately overwritten with one of the same or more importance, otherwise if the incoming message is less important, it won't be shown. So for example, you will probably want the pickup message of a goal item to be more important than the pickup message of a shield or clip.

The 'cheat messages' are from 700 onwards.
Just so you know where to insert a few 'Cheater!' and 'Chicken Mode ON' ... :-)

ANIM (ANM Files)

ANIM (ANM Files)

Those LFD resources contain animations played in the cutscenes, the missions objective screens that appear in the PDA, and the game menus.
ANIMs are quite logically a collection of DELTs.

Note: the .anm extension is a convention adopted by add-on developers when writing conversion programs, there are no real ANM files in DARK FORCES.

Their format is quite simple:

```
ANIM_Header IS
{
 NbDELT       int          // number of embedded DELTs
}
```

Then follows each DELT, encoded as :

```
ANIM_DELTData IS
{
 DELTSize      long        // the size of the embedded DELT
 aDELT         bytes[n]    // a complete DELT resource
}
```

DELT (DLT Files)

A DELT LFD resource codes a static image.
They are generally used as backgrounds for ANIMs, but their most important use is in the briefings 'texts' (the scrollable section of the briefing screen) which are a DELT stored in dfbrief.lfd for each level

Note: the .dlt extension is a convention adopted by add-on developers when writing conversion programs, there are no real DLT files in DARK FORCES.

```
DELT_Header IS
{
 OffsX       int       // X offset
 OffsY       int       // Y offset
 SizeX       int       // X size - 1 !
 SizeY       int       // Y size - 1 !
}
```

After the header, a variable number of **line descriptors** follow.
They are composed of an header and some data.

```
DELT_Line
{
SizeAndType  int       // size and compression of the line
StartX       int       // X position of line start
StartY       int       // Y position of line start
}
```

StartX and StartY indicate the point where to start the drawing. You can start in the middle of a line, and draw a portion of it. Lines need not be in consequential order. You can split one line in more than one section. Portions not covered are, of course, transparent.

Bits 1-15 of SizeAndType indicate the number of pixels described in this section.

If bit 0 of SizeAndType is 0, the byte following the header contains the number of bytes to copy.
Those bytes follow.

If bit 0 is 1, data compressed with RLE follows.

This data may be composed of copy and RLE parts, which is indicated by **bit 0** of the count byte.

FILM (FLM Files)

A FILM LFD resource handles the scripting of a scene.
It specifies the PLTT to use as palette, the DELT to use as background, the ANIM to play, and the VOIC to play during the ANIM.
The PLTT, DELT and ANIM are in the same LFD as the FILM, while all the VOIC seem to refer to jedisfx.lfd

Note: the .flm extension is a convention adopted by add-on developers when writing conversion programs, there are no real FLM files in DARK FORCES.

First comes a header :

```
FILM_Header IS
{
 u1          int         // unknown
 u2          int         // unknown
 nbENTRIES   int         // number of entries
}
```

Then follow a series of entries:

```
FILM_Entry IS
{
 TYPE          char[4]          // type of the resource
 NAME          char[8]          // name of the resource
 LENGTH        long             // length of the resource
                                // including this structure
}
```

Note that this structure is identical to the LFD_IX_Entry structure.

Each entry may be:

Section	**Description**
ANIM	animation, this is a collection of DELT
DELT	static image in delta format
PLTT	palette used for ANIM and DELT
VOIC	VOC (standard Creative Labs format)
VIEW	First entry in the FILM
CUST	Custom

The first entry is of type VIEW.
The PLTT entry seems to be of fixed size, and the ANIM entry depends on the number of frames in the animation.
See Carl Kenner's Description for more details on these entries.

Then comes a trailer:

```
FILM_Trailer IS
{
 MAGIC       char[4]     // = 'END' + 0x00
 u1          int         // unknown
 u2          int         // unknown
 u3          int         // unknown
}
```

See Carl Kenner's description

FILM (FLM Files)

Here is Carl Kenner's much more complete description of FILMs.

```
FILM File Specs (DOS Name = .FLM)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Addresses mentioned are hexadecimal. Values are decimal.

Film files contain the directions of what to do in a cutscene.
Although they can also be used for dialog boxes, this is rare and should not
bother you.

They are part of the LANDRU system developed by Ed "Kill'em" Kilham, and as
such are only found in .LFD files. They are used in Dark Forces, X-Wing,
Imperial Pursuit, B-Wing, TIE Fighter and Defender of the Empire.

Here is the format of the header:

00:    Magic  (Integer)         Always equals 4
02:    FilmLength (Integer)     In clock ticks (about 1/10 of a second)
04:    ObjectCount (Integer)    Not including END

"Magic" may mean something, but it probably just identifies it as a FILM file.

A series of ObjectCount object blocks follows.
Here is the format of each object block:

00:    Extension (4 chars)        Block Type Name (see table)
04:    Name (8 chars)             File Name (see table)
0C:    TotalLength (Long Int)     Total length of Block (BlockLength + 22)
10:    BlockType (Integer)        (See Table)
12:    NumberOfCommands (Integer) Number Of Commands (including End command)
14:    BlockLength (Integer)      TotalLength - 22 (don't ask me why)
16:    ===== Command List ======  (see below)

If the object file doesn't exist you will get an error in a dialog box saying
"Unable to load all items in cutscene _________"

============================
Block Types
----------------------------
01:    END/0
02:    VIEW
03:    DELT   ANIM   CUST
04:    PLTT
05:    VOIC
============================

================================================================
BlockNames
----------------------------------------------------------------
VIEW:    "UNTITLED"  \  Maybe you can give the film a title,
```

```
END:     "UNTITLED"  /  but nobody ever does, so I don't either.
CUST:    "CUSTOM"
Otherwise it is the filename
=================================================================

Here is the format of each command:

00: CommandLength (Integer)   Total length of the command
02: Command (Integer)         (See Table)
04: ParameterList (Integers)  (CommandLength-4) / 2 parameters

=====================================================
Commands (decimal not hex)
---------------- General Commands -------------------
0:  Unused ???
1:  Unused ???
2:  END ()
3:  *TIME* (timeframe)
---------------- Type 3 Commands --------------------
4:  MOVE (x, y, 0, 0)
5:  SPEED (horizontal, vertical, 0, 0)
6:  LAYER (z)
7:  FRAME (n, ?0?)
8:  ANIMATE (direction, ?0?)
9:  CUE (n)
10: VAR (v) ???
11: WINDOW (xMin, yMin, xMax, yMax)
12: ?
13: SWITCH (OnOff)
14: ???? (1, 0/1)
---------------- Palette commands -----------------
15: PALETTE (0)
16: ?
17: ?
------------------ View Commands ------------------
18: CUT (c, t)
19: ?
------------------ Sound Commands -----------------
20: LOOP (0)
21: ?
22: ?
23: ?
24: PRELOAD (2/1)
25: SOUND (OnOff, volume, 0, 0)
26: ?
27: ?
28: STEREO (OnOff, volume, 0, 0, PanPosition, 0, 0)
=====================================================

All .FILM files must have one VIEW block and it must be the first.
It's name should be UNTITLED.
There is also a END block at the end of the file. It is not counted in
NumberOfObjects. It contains only the first part of the object block header.
It has the same name as the VIEW block.

One or Two CUST blocks both named CUSTOM are optional. They are not associated
with files.


Command Descriptions:
```

```
~~~~~~~~~~~~~~~~~~~~~

END ()
======
Length: 4
Number: 2
Syntax: END

This command is always the last command for an object.

*TIME* ()
=========
Length: 6
Number: 3
Syntax: *TIME* x
    or  *TIME* x.x

This command is always the first command for an object. It tells LANDRU when
to do the following commands up to the next *TIME* command.
The next *TIME* command tells it when to do the commands following it, etc.
Any commands between 2 *TIME* commands will be done simultaneously (almost).
*TIME* commands must come in chronological order otherwise the LANDRU system
will hang (or give an error message?).

x is the time in clock ticks (about 1/10th of a second).
x.x is the time in seconds approximately (decimal number).

-----------------------------------------------------------------------------
Type 3 Commands
-----------------------------------------------------------------------------
These commands may only be used on graphical objects or a CUSTOM object.

MOVE (x, y, 0, 0)
=================
length: 12 or 18
number: 4
Syntax: MOVE x y 0 0
    or  MOVE x y 0 0 0 0 0
    or  MOVETO x y 0 0
    or  MOVETO x y 0 0 0 0 0

Moves the object to the coordinates (x,y).
All objects are at the origin (0,0) at the start.

SPEED (right, down, 0, 0)
=========================
length: 12 or 18
number: 5
Syntax: SPEED right down 0 0
    or  SPEED right down 0 0 0 0 0

Changes the objects horizontal speed to <right> and its veritcal speed to
<down>. Negatives mean left and up respectively.
The units are approximately decapixels per time frame, or something similar.
Objects are stationary by default.

LAYER (z)
=========
length: 6
number: 6
```

```
Syntax: LAYER z

Changes the object's layer to z. The smaller or more negative <z> is the
further forward it is. Objects with a low <z> move in front of objects with a
high <z>.
Objects always start on layer 0.
100 is usually the background.
*** I think that layer zero is done like the text crawl for scene #30 ***

FRAME (n, ?0?)
==============
length: 8
number: 7
Syntax: FRAME n 0
? or ?  FRAME n 128

Displays the frame number <n> of a .ANIM object.
If n is odd then frame <n>-1 will be drawn first then frame <n> will be drawn
on top. If <n> is higher than the number of frames in a .ANIM then you will
get an error message:
" XACTOR.C: Value out of bounds. "
or something similar.
Animations start at frame 0.

ANIMATE (direction, ?0?)
========================
length: 8
number: 8
Syntax: ANIMATE direction 0
??? or  ANIMATE direction 128

Direction may be one of the following:
 0, OFF
 1, ON, FORWARDS
-1, BACKWARDS

This command starts a .ANIM object animating in the appropriate direction.
.ANIMs start, by default, inanimate on frame 0.

CUE (n)
=======
length: 6
number: 9
Syntax: CUE n

If used in a CUST object in Dark Forces then it sends a cue to iMuse to start
the music. This corresponds to the cue number (which is only a comment) in the
CUTMUSE.TXT file under the SEQUENCE specified in the CUTSCENE.LST file. Music
is not a part of Landru so it is found in GOB files not LFD files. This makes
it HARD to add music to cutscenes.

If used in a CUST object in X-Wing or TIE Fighter then it handles all sorts of
goodies, such as speech, text and music. The VAR command also plays an
important role.

If used in a graphical object then it probably does nothing useful.

VAR (n)
=======
length: 6
```

```
number: 10
Syntax: VAR n

Unknown. Used mainly in X-Wing, TIE Fighter CUST objects.

WINDOW (xMin, yMin, xMax, yMax)
===============================
length: 12
number: 11
Syntax: WINDOW xMin yMin xMax yMax

Probably clips and limits the displayed image to the specified region.
Useful to make stars fit a window, when other parts are transparent.

SWITCH (OnOff)                          \    |    /
==============                            \  |  /
length: 6                              -  -  *  -  -
number: 13                                /  |  \
Syntax: SWITCH OnOff                    /    |    \

OnOff may be one of the following:
0, OFF
1, ON

This command is VERY important. It switches the graphic on or off.
When graphics are switched off they are not displayed.
Graphics are SWITCHED OFF BY DEFAULT.
Objects should always be switched on at the start and switched off at the end
(NumberOfFrames-1).

???? (1, 0/1)
=============
length: 8
number: 14
Syntax ???? 1 0
   or  ???? 1 1

This command is quite common is some games, but I have no idea what it does.
It can't do anything important because cutscenes work fine without it.

-----------------------------------------------------------------------------
Palette Commands
-----------------------------------------------------------------------------
These commands may only be used on palette objects. (Type 4)

PALETTE (0)
===========
Length: 6
Number: 15
Syntax: PALETTE 0

Sets the palette to the palette in this palette file.

-----------------------------------------------------------------------------
View Commands
-----------------------------------------------------------------------------
These commands may only be used on the View Block.
A VIEW block must always be present, but may contain no commands.

CUT (how, type)
```

```
===============
Length: 8
Number: 18
Syntax: CUT how type

I'm not sure exactly what this does, but it is definately a cut of some sort.

<how> may be one of the following:
1, SWAP
2, CLEAR
3, DIRTY
12, FadeRight
13, FadeLeft
14, FadeUp
15, FadeDown
21, FadeUpDown
2333, FadeToBlack
23, Stop

<type> may be one of the following:
2, Old
3, End
4, New

-----------------------------------------------------------------------------
Sound Commands
-----------------------------------------------------------------------------
These commands may only be used on sound objects. (Type 5)

SOUND (OnOff, Volume, 0, 0)
===========================
Length: 12
Number: 25
Syntax: SOUND 1 volume% 0 0
   or : SOUND 0 0 ? ?

Plays the sound or switches it off depending on the value of OnOff.

STEREO (OnOff, Volume%, 0, 0, PanPosition, 0, 0)
================================================
Length: 18 or 24
Number: 28
Syntax: STEREO 1 volume% 0 0 PanPosition 0 0
   or   STEREO 0 0 ? ? 0 ? ?

Plays a sound in stereo or switches it off.
PanPosition is 0-255.
0 = left
255 = right
128/127 = center

LOOP (0)
========
Length: 6
Number: 20
Syntax: LOOP 0

Breaks out of the current repeating loop. (I think)

PRELOAD (2/1)
```

```
=============
Length: 6
Number: 24
Syntax: PRELOAD 1
   or   PRELOAD 2

Unknown. Probably has something to do with loading?
```

PLTT (PLT Files)

PLTT LFD resources are of variable size, and store a (possibly incomplete) palette used by ANIM and DELT resources.

Note: the .plt extension is a convention adopted by add-on developers when writing conversion programs, there are no real PLT files in DARK FORCES.

```
PLTT_File IS
{
First        byte              // first color in the palette
Last         byte              // last color in the palette
colors       RGB_Color[n]      // n = Last - First + 1
pad1         byte              // unused = 0
}
```

Where:

```
RGB_Color Is
{
R            byte              // Red intensity
G            byte              // Green intensity
B            byte              // Blue intensity
}
```

Note that contrary to the PAL files, the intensities range from 0 to 255 in the PLTT resources.

FONT (FON Files)

These LFD resources store a proportional character set, which may be incomplete.
I found two examples : font6 and font8.

Note: the .fon extension is a convention adopted by add-on developers when writing conversion programs, there are no real FON files in DARK FORCES. There are FNT files however, which are quite different !

```
FON_Header IS
{
First     int                   // First character in font
Last      int                   // Last character in font
u1        int                   // 8, could be bits per char line
Height    int                   // Height of Chars
u2        int                   // could be average Width
                                // or the minimal Width to use
pad1      byte[2]               // 2 times 0x00
}
```

Then follows a block of (Last-First+1) bytes (one per character), which code the width of the corresponding character.

```
FON_Characters_Widths IS
{
Widths    byte[Last-First+1]    // each byte is the width of one
                                // character
}
```

Then each character is described in turn:

```
FON_Character IS
{
Bitmap    Byte[Height]          // Height bytes for each character
}
```

Now the funny part: each of these bytes is a bitmap representation of a line of the character. A bit set correspond to a pixel drawn on the screen.

For example, if the bytes are 48h, FCh, 48h, FCh, 48h, 00h
this gives

```
48h    .X..X...
FCh    XXXXXX..
48h    .X..X...
FCh    XXXXXX..
48h    .X..X...
00h    ........
```

Which is the # character.
Note that the width as referenced in the FON_Characters_Widths array would be 6 for this character.
In fact, FON_Characters_Widths must be used to determine where on the screen to draw the next character.

VOIC (VOC Files)

Those LFD resources store .VOC files, in the Creative Labs format.
It seems that all the VOIC resources are in the jedisfx.lfd file.

GMID (GMD Files)

Those LFD resources store .GMD general midi files.

BRIEFING.LST

BRIEFING.LST

[by Nicola Salmoria]

In DARK.GOB we have the file BRIEFINGS.LST, its contents are trivial:

LEV	name of the level
LFD	name of the .LFD file to take the briefing from
ANI	name of the ANIM (in the .LFD file) to use as background
PAL	the palette to use

[It seems that LFD containing briefings and objective screens **must** be called DFBRIEF.LFD, or the PDA won't work properly - Jereth]

CUTSCENE.LST

[by Michael Taylor]

The cutscene.lst file contains the necessary information to display the various scenes in between missions. This includes the starting logos and the ending credits. The file is integrated with the various LFD files that make up the cutscenes. Following is the complete file format and the descriptive notes I have made.

File Format

Firstly, comments are denoted by the pound sign (#). Everything after the sign is ignored until the end of the line.

First comes the magic 'CUT' and a version number.
```
| CUT 1.0
```

Next is the keyword CUTS and the number of cutscenes defined in the file.
```
| CUTS 39
```

Then starts the cutscene information:
```
| 550:  gromas1.lfd  gromas1   10  0  0  3  100
```

Lets decipher this line. The first number is an id number used by the LFD file. It seems that each mission is given its own id number and therefore you cannot change the first id number but you may add id numbers which in effect adds more cutscenes to a file. More on this later.
Note: as stated in the file, don't change the ftextcra.lfd scenes id number, it is hardcoded into the program.

[There are in fact id numbers which are hardcoded to be played before and after each level. Cutscene 100 is played before level 1, 150 is played after level 1, 200 is played before level 2, 250 is played after level 2 and so on. Also note that cutscene 10 is played just after loading up DF, and 1500 when the game has been completed. And for your interest, the reason the textcrawl is hardcoded at 30 is because cutscene 30 was specially designed to have the text scroll into the distance -- giving the textcrawl another id number will cause it to scroll straight up -Jereth]

Next comes the resource file that contains the cutscene and following this a scenes file. You can move most of the resource files around and effectively change the scenes provided you also change the scenes file. I received an error message when I tried to change the scenes file while leaving the resource file the same. It is possible to swap files, for example you can have the credits displayed prior to the Dark Forces logo if you swap lines 41 and 40. This is not a good way to swap scenes though. I'll show you a better way later.

[Actually, the resource file is the .LFD to take the cutscene from, while the scenes file is the FILM within the .LFD to use as a script for the cutscene. So they do **not** actually need to have the same name - Jereth]

Next comes the speed at which the scene is showed. It must be in the range of 5-20 else an error message is displayed. 20 is the fastest. I use this to speed up the starting logos when I'm testing things in the cutscene.lst file.

The next number is the scene id that should be displayed when this one ends. Zero means that this is the last scene to display. This is by far the best way to swap scenes. For example if you want to show the credits prior to the Dark Forces logo then you would change the next scene number in line 30 to 41 and then change the next scene number in line 41 to 40. Finally, change the next scene number in line 40 to 0. If you forget to change the last scene's next scene number to 0 then it will get into a loop. By changing the next scene number, you can also add your own cutscenes.

The next number is the skip scene number. This number determines which scene to be displayed if ESC [or Enter] is hit. In most files, it should be zero which means to go to the menu or the next mission. But it can contain an id number for a scene.

Next comes the SEQ number for the cutmuse.txt file. This links the appropriate music with the scenes.

Finally is the volume at which the sound is played. 100 is normal.

CUTMUSE.TXT

This file controls the music to be played during cutscenes.
CUTMUSE.TXT accepts // comments.

File Format

```
SEQUENCE: 1

// cue 1
CUE: star-thm
0 0 0 0

// cue 2
CUE: star-thm
B 2 B 2
...

SEQUENCE: 2

// cue 1
CUE: execmus
0 0 0 0

...
```

Note: there is no header.

As you can see, the file is split into a number of **Sequences** which correspond to the "SEQ number" in CUTSCENE.LST. Sequences are nothing more than logical groupings of cutscenes that are played together, for example the starting sequence, the long sequence before TALAY, the ending sequence. The whole point of a sequence is that in CUTSCENE.LST, only the first cutscene in each sequence of cutscenes needs to point to the corresponding music sequence in CUTMUSE.TXT -- the rest can have "SEQ number" set to 0 as the same music sequence selected at the first cutscene will apply throughout the remainder of the cutscene sequence.

Sequences each have a number of **Cues** which are fired by the CUST objects in FILMs of cutscenes. Cues define a .GMD file (note - without the extension) to play the music from, what chunk within it to play, and how and when to play the chunk.

Note: the numbering of CUEs in CUTMUSE.TXT are just comments -- they are not actually defined with numbers.

[Thanks to Alex Novikov for lots of help in figuring out the following]

Cues point to the chunk to be played like this:

%c %d %c %d

The two characters refer to MTrk chunks within the GMD. Capital letters are used, i.e. A, B, C, D, E..... where A is the first track, B is the second...... The numbers seem to refer to a point in the track -- larger numbers will start the track from further on. They maybe refer to a number of patterns or an interval of time (seconds or beats?), from the start of the track.

Now, the overall meaning seems to be something like this: the first character and number refer to a certain point in the music, which when reached, will change the music to a point defined by the second character and number. So "C 7 D 2" possibly means: when the music reaches track C time/pattern 7, then change to track D time/pattern 2. All this will happen when a FILM CUST object fires the Cue.

There are also a few exceptions:

"0 0 0 0" seems to be the equivilant of "give no command", so the music will just play on through unless it gets into a melody loop.

"1 0 0 0" usually means start the next track, but it has varying effects in different cutscenes, and sometimes will bring the music out of a loop, but sometimes won't.

". 0 0 0" will fade the music away.

A lot of this seems to be dependant on the internal iMUSE commands within GMD tracks, whose workings are unfortunately still very much unknown.

Reference

Reference

DF executable and DF engine

Descriptions Lists

Cheat Codes

Just in case you haven't found them anywhere else !
I've also shown the equivalent or nearest cheat for DOOMers.

DF CHEAT	NOTES	DOOM CHEAT
LABUG	bug mode	
LACDS	map with things	IDDT x2
LADATA	coordinates & %secret	IDMYPOS
LAIMLAME	total invincibility	IDDQD
LAMAXOUT	get everything	IDKFA
LANTFH	teleport (*)	IDSPISPOPD/IDCLIP
LAPOGO	allow to climb any height	IDSPISPOPD/IDCLIP w/o walking thru
LAPOSTAL	get weapons and ammo	IDFA
LARANDY	weapon super charge	
LAREDLITE	freeze enemies	
LASKIP	finishes current level	
LAUNLOCK	get all the keys	the 'K' of IDKFA
LAlevelname	jump to level	IDCLEV

(*) To use this, press TAB to show the map, then press and hold the key just under the Escape key.
Now use the cursor keys, and a red dot will move accordingly, starting at your current position.
Set it where you want to go, and type LANTFH

Please note that (just like in Doom) these codes are directly scanned from the keyboard, and so correspond to a QWERTY keyboard disposition. So a French user on an AZERTY keyboard would have to type 'LQI,LQ,E' on his keyboard instead of 'LAIMLAME'.

dark.exe command line

Here are all the command line parameters for the Dark Forces executable:
(ordered by probable usefulness)

Parameter	Description
-ugob	use an user gob file (where gob=your gob file)

-shots	enable screen shot mode (use Print Screen key)
-c	disable cutscenes
-llevel	play a particular level (where level=yourlevel)
-xd	specify CD-ROM drive letter (where d=drive)
-f	don't check to see if FILES= is set high enough
-t	autotest mode (runs all levels briefly)
-g	create a text file with a list of all files that were opened during the running of the game

A few words about the DF engine scene rendering

[by Daron Stinnett, DF Project Leader]

To the engine, a window refers to a clipping region that is created for every adjoining wall in the current view. If the camera is in one sector looking at an adjoin into another sector, one window would be created. The window is used to clip the drawing of all walls and objects in the adjoining sector. Every adjoin becomes a window (clip region) for the drawing of all objects and walls that are viewed through the adjoin.

The engine has a hard limitation of 40 active windows. Active windows build up when adjoins are viewed through other adjoins. For example, given a single sector that has adjoins to three sectors along one side, if all three adjoins are viewed at once, there would only ever be one active adjoin. This is because none of the adjoins is viewed through another adjoin. However, all four sectors were stacked end to end and there was an adjoin between adjacent sectors, and the camera was placed so that it was in one of the end sectors and could view all three adjoins, this would result in 3 active windows. So a long hallway made of 42 consecutive sectors that could all be seen from one end, would cause the maximum active windows to be exceeded, resulting in the smearing effect at the far end of the hallway.

Something to watch out for is the effect of sub-sectors. Every edge (wall) of a sub-sector creates a window. So a sub-sector has the effect of splitting a scene up into pieces, often multiplying the number of windows in a scene. This is especially problematic when the windows created by a sub-sector split the drawing of 3D objects into several pieces. Since a window is a clip region for drawing all walls and objects, an object that is partially viewed through two or more windows will be drawn as many times as there are windows overlapping the 3D object. This can really slow down the engine.

Managing windows is key to creating speedy levels. Often the speed of a scene is very closely related to the number of windows and their orientation to each other. A long hallway made up of 30 consecutive windows is much easier on the engine than a simpler room with a complex sub-sector (or set of sub-sectors) that creates 30 windows. A good example of a high window situation is looking over the low wall at the city early in the Talay level. However, it works out well because most windows are created further back in the scene. If the situation were reversed - one way to do that would be to split the low wall into many small walls - the engine would bog down in a big way.

Limitations on objects

[by Daron Stinnett, DF Project Leader]

There is no maximum for objects in a sector.
However there can only be a total of 512 objects in the level.
Also you can only load up 64 each of PODs, FMEs, and WAXs.

So you could have 1 FME and 512 objects that use that FME.

Notes
This limitation has probably been removed, because you **can** use more than 512 objects in a level and all work well.

On the other hand, there is a limit to the number of objects that can be **displayed** (or active ?) at a given time.
When you reach it objects begin to flicker in and out, enemies don't appear but do fire at you, etc.
It begins at around 500 objects too…

Metrics

This collection of numerical data should be a useful reference for level designers asking themselves questions like:

- will the player be able to climb this stairs smoothly ?
- will the player be able to jump this ?
- will the player die from this fall ?
- will the player pass through this gap ?
- etc.

MAXIMUM WALKABLE HEIGHT : 3.50

Note
- Don't forget that any height can be made walkable by setting Wall Flag 3 bit 1.

MAXIMUM JUMPABLE HEIGHT : 9.65

Notes
- It doesn't matter if you're running or not
- It doesn't matter if you're crouching or not
- Don't forget that any height can be made walkable by setting Wall Flag 3 bit 1.

DAMAGE FROM FALLING :

HEIGHT	**MIN**	**MAX**	**MEAN**	**Notes**
36	0	0	0	(1)
37	0	1	0	(2)
40	4	9	6	
45	19	23	21	
50	31	35	33	
55	43	48	44	
60	54	58	56	
65	67	71	69	
70	77	78	77	
75	87	91	89	
79	93	98	95	(3)
80	95	++	98	(4)
81	98	++	++	(5)
82	++	++	++	(6)

Notes
(1) Maximum "no damage" fall
(2) 1 point of damage happened twice in 30 falls.
(3) Maximum "no death" fall
(4) % death : 3/11 = 27%
(5) % death : 12/19 = 63%
(6) Minimum "sure kill"

- Shields or Supershield are of no help
- LAIMLAME totally protects you (at least up to 3000).
- Crouching doesn't affect the damage taken.
- Jumping up before the fall does of course add to it.
- Sprinting when hitting the ground doesn't change anything.
- The current health doesn't affect damage.

Effects of second altitude:
1) A positive second altitude (water) must be added to the height of the fall.
I.e. the water doesn't break the fall at all, it increases the damage :-)
2) A negative second altitude (platform) must be subtracted from the height of the fall.
3) In both cases the results are consistent with the equivalent fall from the sum or difference of heights.

[All falls tested between 10 and 30 times.]

MINIMUM WALKABLE WIDTH : 4.90

Notes
- This is a width between two angles of columns in a room.
  Passing between those two isn't exactly the same as walking in a 4.9 wide corridor !
- When running you may sometimes pass through a gap as little as 4.6
  I strongly believe this is a problem in the engine collision detection.

MINIMUM WALKABLE HEIGHT : 6.80

Note
The generally adopted rule of thumb of 1 DF unit = 1 foot would make Kyle very big (207 cm).
I believe we should use **1DF unit = 25 cm** instead.

MINIMUM CROUCH HEIGHT : 3.00

LONGJUMPS

Standing	~14
Walking	~20
Running	~40

Notes
- These values assume that the start and end altitudes of the jump are the same.
- DF levels must be set on low gravity worlds :-)

SOUNDS.GOB

Sounds in this list may be used in INF with the page: command.

In case you wonder about such names as M01KYL01.VOC, here is the decomposition :

M	Mission	
01	Mission Number (Arc Hammer is 16!)	
KYL	Speaker	
	KYL	Kyle
	JAN	Jan Ors
	IMP	Imperial
	JAB	Jabba

	MMA	Mon Mothma
	MOC	General Mohc
	NAR	Narrator
	REB	Rebel
	VDR	Vader;
01	first speech for this mission (A1 is an alternate recording)	

[by David Lovejoy and Len Bowers (double submission :-))]

AXE-1	VOC	ATTACK	Gammoreean guards axe sound
CREATUR2	VOC	ATTACK	Sewer monster loud growl
INTSTUN	VOC	ATTACK	Interrogator droid shooting
PROBFIR1	VOC	ATTACK	Probe droid firing
REMOTE-2	VOC	ATTACK	ppssshh REMOTE
BOBA-1	VOC	BOBA	Boba Fett hah hah when sees you
BOBA-2	VOC	BOBA	Boba Fett firing
BOBA-3	VOC	BOBA	Boba Fett duck sound when hurt
BOBA-4	VOC	BOBA	Boba Fett getting killed
ROCKET-2	VOC	BOBA	Boba Fett jet pack (flying)
KEY	VOC	BONUS	Pick up key sound
BONUS	VOC	BONUS	Bonus pick up sound
COMPLETE	VOC	BONUS	Mission complete
DOOR	VOC	DOOR	Standard door opening hissing sound
DOOR2-1	VOC	DOOR	Large door double thump open
DOOR2-2	VOC	DOOR	large door running
DOOR2-3	VOC	DOOR	thump same as door 1-3
ELEV2-1	VOC	DOOR	click and a clunk
ELEV2-2	VOC	DOOR	loud running noise
ELEV2-3	VOC	DOOR	loud clunk
LOCKED-1	VOC	DOOR	locked key door sound
SWITCH3	VOC	DOOR	Switch flip standard
BIGREFL1	VOC	DT1	WHEN hit by plasma from Dark trooper
PHASE1A	VOC	DT1	Phase 1 DT neaaahhh when sees you
PHASE1B	VOC	DT1	Phase 1 DT aaahhh hurt
PHASE1C	VOC	DT1	Phase 1 DT Dying
SWORD-1	VOC	DT1	Phase 1 DT sword sound
PHASE2A	VOC	DT2	Phase 2 DT ahhgglllloooklok when sees you
PHASE2B	VOC	DT2	Phase 2 DT Phutt DIE hurt
PHASE2C	VOC	DT2	Phase 2 DT DYING
ROCKET-1	VOC	DT2,DT3,WPN	Dark trooper flying jetpack
PHASE3A	VOC	DT3	metallic sound with boba fett laugh (MOHC) when sees you
PHASE3B	VOC	DT3	Phase 3 DT mrp ughh ughh hurt
PHASE3C	VOC	DT3	Phase 3 DT dying ooohhhhhhh
BOSSKDIE	VOC	DYING	bossk dying
CREATDIE	VOC	DYING	Sewer monster diying
GAMOR-1	VOC	DYING	Gammorean guard pig squeal
REEYEE-3	VOC	DYING	yooooohhhh ughghggh dying died
ST-DIE-1	VOC	DYING	Stormtrooper/commando/officer/ dying
EEEK-3	VOC	EXP/MOUSE	mouse bot dying
EX-SMALL	VOC	EXP/WEAPON	Thermal detonator explosion & int droid explosion
EX-TINY1	VOC	EXP/WEAPON	laser ,repeater shot hitting wall
PROBALM	VOC	EXPLOSION	Probe droid about to explode alarm
TURRET-1	VOC	EXPLOSION	Turret Shot
SCRSHOT	VOC	HEADER	PICTURE taking sound (screen shot)
BOSSK-3	VOC	HURT	bossk higher pitched squeal hurt
CREATHRT	VOC	HURT	Sewer monster hurt
GAMOR-2	VOC	HURT	Gammorean Guard Pig squeal louder hurt
REEYEE-2	VOC	HURT	yoooooohhhh hurt
ST-HRT-1	VOC	HURT	Stormtrooper/commando/officer/ hurt by laser

KELL-1	VOC	KELL	Kell Dragon roar used on first siting Order in exe kell-1,-8,-5,jump,7
KELL-5	VOC	KELL	Kell dragon hitting with tail /biting
KELL-7	VOC	KELL	Kell dragon loud then soft errgggr / used when kell killed
KELL-8	VOC	KELL	Kell dragon roar/used when hurt
KELLJUMP	VOC	KELL	Kell dragon jumping
CHOKE	VOC	KYLE	Kyle Choking in gas
CRUSH	VOC	KYLE	Kyle getting crushed
FALL	VOC	KYLE	Yaaaaahhhhhhhhhh
GOGGLES1	VOC	KYLE	Goggles ON
GOGGLES2	VOC	KYLE	Goggles battery run down
HEALTH1	VOC	KYLE	Ugh health loss ???? when used
JUMP-1	VOC	KYLE	Kyle jump
KYLEDIE1	VOC	KYLE	Big heart pumping sound (Kyle Dying)
LANDING1	VOC	KYLE	The Crow landing
MASK1	VOC	KYLE	Gas mask breathe in sound
MASK2	VOC	KYLE	Gas mask breathe out sound
SHIELD1	VOC	KYLE	Something hitting shield ??????
SWIM-IN	VOC	KYLE/WEAPON	Kyle jumping into water
CLAYMOR1	VOC	KYLEWPN	Laying mine
CONCUSS1	VOC	KYLEWPN	Concussion rifle empty
CONCUSS5	VOC	KYLEWPN	Concussion rifle firing
CONCUSS6	VOC	KYLEWPN	Concussion rifle empty ?????
FUSION1	VOC	KYLEWPN	Fusion cutter single shot
FUSION2	VOC	KYLEWPN	Fusion cutter empty
MISSILE1	VOC	KYLEWPN	Kyle firing missile
MORTAR2	VOC	KYLEWPN	Mortar gun empty
MORTAR4	VOC	KYLEWPN	Mortar gun firing
MORTAR9	VOC	KYLEWPN	Mortar chamber rotate
PISTOL-1	VOC	KYLEWPN	Bryar pistol shot
PISTOUT1	VOC	KYLEWPN	Bryar pistol out of ammo
PLAS-EMP	VOC	KYLEWPN	Plasma cannon empty
PLASMA4	VOC	KYLEWPN	Plasma cannon firing
REP-EMP	VOC	KYLEWPN	repeater empty
REPEAT-1	VOC	KYLEWPN	Repeater gun shot
REPEATER	VOC	KYLEWPN	repeater rapid fire
RIFLE-1	VOC	KYLEWPN	Rifle single shot
RIFLOUT	VOC	KYLEWPN	Rifle empty
SWING	VOC	KYLEWPN	FIST SWING sound
WEAPON1	VOC	KYLEWPN	Weapon pickup sound
QUARTER	VOC	KYLEWPN/KYLE	5 quick beeps
EEEK-1	VOC	MOUSE	Mouse bot
EEEK-2	VOC	MOUSE	Mouse bot hit/hurt Dedmouse.fme
ICMDO-1	VOC	n/u	He's over here stop that man (commando)
IOFFIC-1	VOC	n/u	Halt hold it right there (Officer)
KELL-2	VOC	n/u	Kell dying ??? couldn't confirm use, not called by exe
REEYEE1	VOC	N/U	hey hold up who's there
REEYEE2	VOC	N/U	yooohhh
REEYEE3	VOC	N/U	yoooooghg ughghh dying
REEYEE4	VOC	N/U	yoooooagagah dying
STORM-1	VOC	n/u	There he is get him Stormtrooper not used
BOSSK-1	VOC	SIGHT	bossk hissth
CREATUR1	VOC	SIGHT	Sewer monster low growl
GAMOR-3	VOC	SIGHT	Gammor Guard grunt
INTALERT	VOC	SIGHT	Interrogator droid uwmmmwha
PROBE-1	VOC	SIGHT	Probe droid enemy escape advance
RANOFC02	VOC	SIGHT	Stop where you are

			Officer used on first sighting
RANOFC04	VOC	SIGHT	Your not authorised in this area
			used on second siting
RANOFC05	VOC	SIGHT	You're in violation of imperial law
			used on third siting
RANOFC06	VOC	SIGHT	Halt
			used on fourth siting
RANSTO01	VOC	SIGHT	There he is stop him
			Stormtrooper first sighting
RANSTO02	VOC	SIGHT	You there, stop where you are
			second sighting
RANSTO03	VOC	SIGHT	Stop Rebel scum
RANSTO04	VOC	SIGHT	You're not authorised in this area
RANSTO05	VOC	SIGHT	Surrender immediately
RANSTO06	VOC	SIGHT	Halt
RANSTO07	VOC	SIGHT	Set blasters on full
RANSTO08	VOC	SIGHT	Blast him
REEYEE-1	VOC	SIGHT	Hey hold up who's there
			used on logic REEYEE2 only
BEEP-10	VOC	WEAPON	Mine triggering in secondary mode
BOLTREF1	VOC	WEAPON	When hit by laser from storm/commando/officer
EMISBY	VOC	WEAPON	When hit by missile from Dark trooper
EX-LRG1	VOC	WEAPON	loud explosion (mine)(concussion rifle)
EX-MED1	VOC	WEAPON	kyle's missile, plasma, mortar explosions
			Dark trooper plasma, DT3 tracker balls explosions
FIREBALL	VOC	WEAPON	??? sounds like a fireball
LASRBY	VOC	WEAPON	Laser shot miss
PUNCH	VOC	WEAPON	Kyle's fist hitting something
THERMAL1	VOC	WEAPON	Thermal Detonator Bounce
TRACKER	VOC	WEAPON	Mechanical noise made by Dark Trooper 3
WELD-1	VOC	WELD	Welder moving short spark.wax Weld in EXE order
			weld-2,-1,sht,hrt,die
WELD-2	VOC	WELD	Welder Moving longer
WELD-DIE	VOC	WELD	Welder dying
WELDHRT	VOC	WELD	Welder hurt
WELDSHT1	VOC	WELD	Welder hitting Kyle
AMMO	VOC		Loading ammo
BEEP-01	VOC		Shrieking Beep
BOSS-05	GMD		
BOSS-08	GMD		
BOSS-10	GMD		
BOSS-11	GMD		
BOSS-14	GMD		
BOSSK-2	VOC		bossk squeal
BULLET	VOC		almost sounds like wind
BUTT1	VOC	n/u	Fist sound
BUTT2	VOC	n/u	Fist hit
CARGO	GMD		
CHUCKL-1	VOC		Boba Fett chuckling hah hah
CLEAT	VOC		walking with ice cleats on
CLOSCRED	GMD		
CONVEYER	VOC		conveyor belt running
CRIXMUS	GMD		
DEFAULT	GMD		
DEFV0000	VOC		empty voc ????
DOOR-04	VOC		big door close/open
DOOR1-1	VOC		Large hissing door
DOOR1-2	VOC		Large door running
DOOR1-3	VOC		Thump
DOOR3-1	VOC		Large hollow sounding door open thump

DOOR3-2	VOC		loud running door
DOOR3-3	VOC		low thump
EEEK2	VOC		faster mouse bot hurt
EEEK3	VOC		faster mouse bot dying
ELECTRIC	VOC		Sounds just like a real arc welder
ELEV1-1	VOC		click
ELEV1-2	VOC		high pitched running noise
ELEV1-3	VOC		click with a motor turning for a sec
ELEV3-1	VOC		loud hiss several clicks
ELEV3-2	VOC		low rumbling running noise
ELEV3-3	VOC		louder clunk
ELEVOFF3	VOC		loud clunk same as elev 2-3
ELEVRUN2	VOC		running elevator
EMISBY1	VOC	n/u	same as emisby 1
EXECMUS	GMD		
FIGHT-01	GMD		
FIGHT-02	GMD		
FIGHT-03	GMD		
FIGHT-04	GMD		
FIGHT-05	GMD		
FIGHT-06	GMD		
FIGHT-07	GMD		
FIGHT-08	GMD		
FIGHT-09	GMD		
FIGHT-10	GMD		
FIGHT-11	GMD		
FIGHT-12	GMD		
FIGHT-13	GMD		
FIGHT-14	GMD		
FLAME-1	VOC		Something like a flame-thrower
FRIGMUS	GMD		
GROMAS1	GMD		
GROMAS2	GMD		
GROOVE2	GMD		
HEALTH2	VOC		MIFT health / door won't open ?????
JABBAMUS	GMD		
LAND-1	VOC		Kyle Landing after jump
LANDING2	VOC		Kind of a low loud rumble
LASRBY1	VOC		weird mechanical noise
LASRFLY	VOC		same as Lasrby1.voc
LOGOMIX	VOC		LA logo sound
M01IMP01	VOC		Primary dropline engage Dropline one, two nine release
M01KYL01	VOC		This is too easy. now to get back to my ship
M01KYL02	VOC		This looks like it could be a normal Imperial attack .
M01KYL03	VOC		A new stormtrooper weapon that can take out a base that easy !
M01KYL04	VOC		This could be interesting.
			OK I'm in but I think I'll need some help
M02JAN01	VOC		Go ahead Kyle
M02JAN02	VOC		Get back to the landing pad and I'll meet you there
M02KYL01	VOC		Jan
M02KYL02	VOC		Looks like I've found something that could help us out
M03JAN01	VOC		You're the boss Kyle
M03KYL01	VOC		Jan I've found Mof Rebus I'm ready to get out of this mess
M04JAN01	VOC		That's all we need Lets get out of here I'm getting nervous
M04KYL01	VOC		I've found interesting looking metal. I think this may offer us some
M05JAN01	VOC		OK Kyle sounds good to me

M05KYL01 VOC	Kyle to Jan , charge set ready to clear
M05KYL02 VOC	Jan ,you better get me out of here .I think i just finished off a Dark
M05KYL03 VOC	If that thing down there is any indication of what were dealing
M06JAN01 VOC	Don't hang around. lets get out of here before any more Dark
M06KYL01 VOC	OK Jan I've rescued Madine
M07JAN01 VOC	Picking up the signal ,looks like were done here
M07JAN02 VOC	OK Kyle lets see where those e smugglers are headed
M07KYL01 VOC	Tracking devices secured
M08KYL01 VOC	Charge one set
M08KYL02 VOC	Charge two set
M08KYL03 VOC	All charges set
M08KYL04 VOC	Woman after my own heart
M08KYL05 VOC	Ah Sh.......
M09JAN01 VOC	Those must be smuggler routes to the ARC hammer. I think it's
M09JANA1 VOC	Those must be smuggler routes to the ARC hammer. I think it's
M09KYL01 VOC	Jan I've found an imperial Nava card
M10JAN01 VOC	Thanks I thought I was done for
M10KYL01 VOC	Jabba what have you done with Jan if any harm comes to her I'll
M10KYL02 VOC	I wish you were here too Jabba there nothing like roast Kell dragons
M10KYL03 VOC	no time for hugs lets get out of here
M11JAN01 VOC	Good job Kyle but your not done yet
M11JAN02 VOC	Beautiful Kyle now get the data tape and get your mercenary hide
M11JAN03 VOC	Kyle something strange is going on down here. Get back here I
M11JAN04 VOC	Oh no Kyle you better lookout I just saw
M11JAN05 VOC	Kyle where are you .I'm back at the landing pad
M11JAN06 VOC	I had Tie fighters all over me. I had to properly dispose of them
M11JANA6 VOC	I had Tie fighters all over me. I had to properly dispose of them
M11KYL01 VOC	Jan I've cracked the central lock I'm in
M11KYL02 VOC	Nava card inserted and decoding
M11KYL03 VOC	Data tapes in hand I'm on my way out
M11KYL04 VOC	Where are you Jan ?
M12IMP01 VOC	Smuggler ship, your flight path is clear begin your docking procedure
M12JAN01 VOC	good job Kyle
M12JAN02 VOC	Good luck Kyle and may the force be with you
M12KYL01 VOC	OK Jan smuggler ship secure
M12KYL02 VOC	Now launching I'll see you on the dark side , Jan
M13KYL01 VOC	here we go
M16KYL01 VOC	That's one
M16KYL02 VOC	that's two
M16KYL03 VOC	one more left
M16KYL04 VOC	Jan would be proud
M16KYL05 VOC	There is no glory in war MOHC
M16KYL06 VOC	For freedom
M16MOC01 VOC	Its been a long time since I've challenged a man in battle
MACHINE1 VOC	Gromas mines machine sounds
MACHINE2 VOC	Louder machine sound
MO8JAN01 VOC	Good job lets blow this ice cube

NOTELOOP	GMD		
OPENCRED	GMD		
POWER1	VOC		Low mechanical noise
PROBALM1	VOC	n/u	high pitched beep beep
PROBALM2	VOC	n/u	Can't explain it //???
REVIVAL	VOC		REVIVE
ROBOT1	GMD		
ROBOT2	GMD		
RUMBLE	GMD		
RUMBLE1	VOC		LOW RUMBLE SOUND
SHIELD2	VOC		Something hitting ?????????
SMOFFICE	GMD		
SNOW	VOC		Kyle walking in snow
SPLASH1	VOC		Kyle jumping into water
STALK-01	GMD		
STALK-02	GMD		
STALK-03	GMD		
STALK-04	GMD		
STALK-05	GMD		
STALK-06	GMD		
STALK-07	GMD		
STALK-08	GMD		
STALK-09	GMD		
STALK-10	GMD		
STALK-11	GMD		
STALK-12	GMD		
STALK-13	GMD		
STALK-14	GMD		
STAR-THM	GMD		
SURFIN	GMD		
SWIM	VOC		Kyle swimming
SWIM-OUT	VOC		Kyle leaving water
SWITCH1	VOC		Switch flip
SWITCH2	VOC		Switch clicking
TAKEOFF1	VOC		Kyle's ship taking off
TAKEOFF2	VOC		Kyle's ship taking off second part
TEMP	GMD		
TEST1	GMD		
TEST2	GMD		
VICTORY	GMD		
WATER1	VOC		Running water
WATER2	VOC		Running water
WIND1	VOC		Wind Noise

Note file :	MO8JAN01.VOC its "O" (letter) instead of "0" (zero)
REEYEES	logic reeyees &reeyees2 use files reeyee-1>3.voc Files reeyee1>4.voc not used
StormTrooper	logic storm1 &troop use st-die-1.voc logic storm1 &troop use st-hrt-1.voc files Ransto01.voc >ransto08.voc are use in sequence
Commando	logic commando uses st-die1.voc logic commando uses st-hrt-1.voc files Ransto01.voc >ransto08.voc are use in sequence
Officer	logic officin uses st-die-1.voc logic officin use st-hrt-1.voc

		files Ranofc02.voc >Ranofc06.voc used in sequence
Kell Dragon		kell-1.voc > kelljump.voc couldn't confirm when kell-2.voc used
Groups		
	ATTACK	attacking sounds group misc logics
	BOBA	Boba Fett group sounds includes rocket-2 voc
	BONUS	bonus pickup, key, complete
	DOOR	door, elev. switch sounds
	DT1,DT2,DT3	sounds used in three groups includes associated weapon noise
	DYING	dying sounds of misc logics
	EXP	Exploding type noises
	HURT	Various logics hurt sounds
	HEADER	I found this near front of exe file hence the name header
	KELL	Kell dragon group
	KYLE	Kyle's associated sounds with things he does
	KYLEWPN	Kyle's weapons these are in the same order as the keyboard keys
	MOUSE	Mousebot group noises
	SIGHT	Various logics first reaction sounds to kyle's presence
	WEAPON	last group in exe with associated fme, wax files for weapon reactions
	WELD	Welder noises

SPRITES.GOB

[by David Lovejoy]

ASHTRAY.FME	Ashtray	
BARREL.WAX	Barrel	
BEERPIP.FME	Beer Pipe	
BOBABALL.WAX	Yellow Boba Fett Ball	Boba Fett Shots
BOBAFETT.WAX	Boba Fett	
BOSSK.WAX	Bossk	
BULLET.FME	Blue Bullet	Autogun Shooting
BULLEXP.WAX	Small Yellow Explosion	Bullet Explosion
CARDS.FME	Cards	
CFLAME.WAX	Yellow Flame	
CHAIN.FME	Hanging Chain	
CHAIR.WAX	Chair	
COMMANDO.WAX	Commando	
CONCEXP.WAX	Big Blue Explosion	Concussion Explosion
CRIX.WAX	Crix Madine	
CUP1.FME	Green Cup	
CUP2.FME	Blue Cup	
DEDBODY1.FME	Dead Body	
DEDBODY2.FME	Dead Body	
DEDBODY3.FME	Dead Body	
DEDMOUSE.FME	Dead Mouse	Called By Exe In Voc Section
DEFAULT.WAX	Cone Head Small	
DET_CODE.FME	Blank Det Code	

DETEXP.WAX	Big Red Td Explosion	Detonator Explosion
DFLAME.WAX	Small Yellow Flame	
EMISEXP.WAX	Fusion Explosion	Fusion Cutter Explosion
EMSCULP.WAX	Emperor's Sculpture	
EWOK86.WAX	Ewok	
EXPTINY.WAX	Yellow Tiny Explosion	
FROGBOWL.WAX	Bowl	
FROGBWL2.FME	Bowl	
GAMGUARD.WAX	Gamorrean Guard	
GENEXP.WAX	White Explosion	
GFPIPES1.FME	Pipes	
GFVENTDN.FME	Vent Pipe Floor Model	
GFVENTUP.FME	Vent Pipe Ceiling Model	
HANGLIT.WAX	Hanging Lamp	
IARMOR.WAX	Shield	
IAUTOGUN.FME	Autogun (Repeater)	
IBATTERY.FME	Battery	
ICANNON.FME	Plasma Cannon	
ICEILIT2.WAX	Ceiling Lamp	
ICHARGE.FME	Supercharge	
ICLEATS.FME	Ice Cleats	
ICONCUS.FME	Concussion Rifle	Bossk Weapon Drop
IDATA.FME	Data Card	
IDET.FME	Detonator	
IDETS.FME	Thermal Detonators	Reeyees Weapon Drop
IDPLANS.WAX	Death Star Plans	
IDTGUN.FME	Broken DT Weapon	
IDTGUN.WAX	Same As Default .Wax	
IENERGY.FME	Energy Cell	Officin Weapon Drop
IFLRLIT.WAX	Floor Lamp	
IFUSION.FME	Fusion Cutter	
IGOGGLES.FME	Goggles	
IINVINC.WAX	Invincible	
IKEYB.FME	Blue Key	
IKEYR.FME	Red Key	
IKEYY.FME	Yellow Key	
ILIFE.WAX	Life	
IMASK.FME	Gas Mask	
IMEDKIT.FME	Med Kit	
IMINE.FME	Mine	
IMINES.FME	Land Mines	
IMORTAR.FME	Mortar	
IMSL.FME	DT Missile	
IMSLS.FME	DT Missiles	DT2 Weapon Drop
INAVA.WAX	Nava Card	
INTDROID.WAX	Interrogation Droid	
IOBCAP6.FME	Round Cap Pipe	
IOBPIP4.FME	Round Pipe	
IOBVALV1.FME	Round Vent	
IPHRIK.FME	Phrik Metal	
IPHRIK.WAX	Phrik Metal	
IPILE.FME	Kyle's Kit	
IPLAZMA.FME	Plasma Cell	DT2 Weapon Drop
IPOWER.FME	Power Cell	Concussion/Repeater
IREVIVE.WAX	Revive	
ISHELL.FME	Mortar Shell	
ISHELLS.FME	Mortar Shells	
IST-GUNI.FME	Laser Rifle Horiz	Wpn Dropped By Troop &Commando
IST-GUNU.FME	Laser Rifle Vertical	
JAN.FME	Jan Ors	

KELL.WAX	Kell Dragon	
LANDMINE.FME	Landmine	
LIT1.WAX	Short Standing Lamp	
LIT2.WAX	Short Standing Lamp	
LIT3.WAX	White Round Lamp Floor	
LIT4.FME	Short White Light	
MINEEXP.WAX	Huge White Explosion	Mine Explosion
MISSEXP.WAX	Yellow And Green Dot	Missile Explosion
MOFREBUS.FME	Moff Rebus Guy	
MORTEXP.WAX	Huge White Explosion	Mortar Explosion
OFFCFIN.WAX	Officer	
PHASE1.WAX	Phase 1 DT	
PHASE2.WAX	Phase 2 DT	
PHASE3X.WAX	Phase 3 DT	
PLASEXP.WAX	Blue Plasma Explosion	Assault Cannon Explosion
PROBE.WAX	Probe Droid	
REDLIT.WAX	Hanging Red Lamp	
REEYEES.WAX	Reeyees	
REMOTE.WAX	Remote Ball	
ROCK.WAX	Rock	
SEWERBUG.WAX	Sewer Bug (Creature)	
SMALITE1.FME	Top Lighted Dome Lamp	
SMALITE2.FME	Short Blue Light Floor	
SPARK.WAX	Electrical Spark	
SPLASH.WAX	Water Splash	Swim-In.Voc
STORMFIN.WAX	Stormtrooper	
TABLE.WAX	Table	
TALLIT1.WAX	Tall Standing Lamp	
TALLIT2.WAX	Tall Standing Lamp	
TBLELIT.WAX	Small Yellow Globe	
TRIPLT.WAX	Tall Multicolor Lamp	
WDET.FME	Thermal Detonator	Throwing TD
WDT3MSL.WAX	Phase 3 DT Yellow Balls	
WEMISS.WAX	Fusion Ball	Shot From Fusion Cutter
WIDBALL.WAX	Green Ball	
WLMINE.FME	Mine On Floor	On Floor
WMINE.FME	Mine On Floor With Light	On Floor
WMSL.WAX	Assault Cannon Missile Fly	Shot From Missile Launcher
WPLASMA.WAX	Assault Cannon Blue Ball	Shot From Assault Cannon
WSHELL.WAX	Mortar Shell Flying	Shot From Mortar Gun

TEXTURES.GOB (A-N)

[by Paulius Stepanas]

X and Y are the width and height of the texture in game units (multiply by 8 for the size in pixels).

Texture	XxY	Description
CESUNSET.BM	32x32	Sky; purple sunset over mountains.
CFWATER4.BM	8x8	Floor; blue, swirling water.
CPCARPT1.BM	8x8	Floor; blue carpet squares.
DEFAULT.BM	8x8	Wall; mottled with red word DEFAULT (colour varies).
ENGSTEXT.BM	2x4	Wall, exterior; grey with peeling, orange strips.
ENGTEXTS.BM	4x8	Wall; grey with vertical streaks of orange paint.
GDJAML1Y.BM	2x16	Wall; red with vertical panels.

GDJMIN1Y.BM	2x8	Wall; orange, stained, with diagonal stripes and controls.
GDJMIN2Y.BM	2x8	Wall; orange, stained, with diagonal stripes and red lights.
GDMINESM.BM	8x8	Door; pale red with grating in centre.
GEGROSKY.BM	16x32	Sky; red, burning.
GPDIRTDK.BM	8x8	Wall; red, mottled.
GPDIRTRD.BM	8x8	Floor; orange, mottled.
GPGRIDSM.BM	8x8	Floor; light grey grating over coloured wiring.
GPMINE02.BM	8x8	Wall; red with horizontal stripes and floor grates.
GPMINE1X.BM	8x8	Wall; red with horizontal stripes.
GPMINE2Y.BM	8x8	Wall; red with vertical bands.
GPMINE5.BM	8x8	Door; red with horizontal stripes with riveted border.
GPMINE6.BM	8x8	Wall; red with horizontal bands.
GPPIPES7.BM	8x8	Floor; pale red, speckled.
GPZIGZ1X.BM	8x8	Floor; red with orange arrow stripes to the right and red grill.
GPZIGZ1Y.BM	8x8	Floor; red with orange arrow stripes to the top and red grill.
GPZIGZ2X.BM	8x8	Floor; red with orange arrow stripes to the right, half red lights.
GPZIGZ2Y.BM	8x8	Floor; red with orange arrow stripes to the top, half red lights.
GPZIGZ3X.BM	8x8	Floor; red with orange arrow stripes to the right.
GPZIGZ3Y.BM	8x8	Floor; red with orange arrow stripes to the top.
GPZIGZ4X.BM	8x8	Floor; red with orange arrow stripes to the right and red lights.
GPZIGZ4Y.BM	8x8	Floor; red with orange arrow stripes to the top and red lights.
GWANO.BM	32x64	Wall; red with speckles, pipes, red lights and crossed grate; parts can be used separately.
GWBIGASS.BM	16x128	Wall; red with pipes and exchange coupling; 4 parts of 32 (can be divided to 16), with and without exchange coupling, with (2) and without sequencer charge.
GWBIGCLF.BM	16x64	Wall; dark red, mottled.
GWDIRTDK.BM	16x8	Wall; dark red, mottled.
GWDIRTLT.BM	16x8	Wall; orange, mottled.
GWDIRTMD.BM	16x8	Wall; red, mottled.
GWDRILL1.BM	4x16	Wall; red side of drill.
GWDRILL4.BM	4x16	Wall; red tubing.
GWDRILL5.BM	16x16	Wall; red planks behind speckled, pale red border.
GWDRILL6.BM	2x16	Door track; speckled pale red with triangle patterns.
GWMINE01.BM	16x8	Wall; red with horizontal stripes, floor grates and ribbing.
GWMINE02.BM	16x8	Wall; red with horizontal stripes and floor grates.
GWMINE03.BM	16x8	Wall; red with horizontal stripes, floor grates and hanging wires.
GWMINE06.BM	8x16	Wall; red with horizontal stripes with riveted border.
GWPIPES1.BM	8x16	Wall; red with tubing.
GWPIPES2.BM	16x16	Wall; pale red with tubing and grating.
GWPIPES3.BM	8x16	Wall; pale red with tubing and grating.
GWPIPES4.BM	16x16	Wall; red with tubing and grating.
GWPIPES5.BM	4x16	Wall; pale red, speckled with recessed tubes.
GWPIPES6.BM	32x16	Wall; pale red, speckled with recessed tubes; in two parts (do not use for walls longer than 16).
GWPIPES7.BM	8x16	Wall; pale red, speckled with recessed tubes.
GWPIPES8.BM	16x16	Wall; pale red, speckled with recessed tubes.
GWSTRIPE.BM	4x8	Wall; orange, stained, with diagonal stripes.
HOLOGRAM.BM	8x8	Floor; circular hologram projector.
IAFAN.BM	8x16	Wall; fan behind grey grating with light; two positions.
IAFANSH.BM	8x16	Wall; grey shadows of fan; two positions.
IASWBLUE.BM	2x8	Switch; blue hour glass, dark and light (each 2x4).
IATRKDEV.BM	8x16	Switch; engine port on dark grey with and without tracking device (each 8x8);
ICDELEV.BM	8x8	Wall; grey grating.
ICDET6.BM	8x8	Wall; light grey with vertical grating and peeling paint.
ICFUEL1.BM	8x8	Floor; grey, circular grill with white lights.
ICJAMLRX.BM	8x8	Floor; dark grey with horizontal strip of red lights.
ID24X16.BM	32x16	Door; grey concrete with vertical grating and red Imperial circle.
ID8X8.BM	8x8	Door; dark grey with embossed Imperial circle.
IDALONG1.BM	8x16	Door; red panelling with white V surrounded by grey panelling with light grey piping.
IDASMAL1.BM	8x8	Door; red with light grey ribbing.
IDASMAL2.BM	8x8	Door; red with white band and light grey column.
IDASMAL3.BM	8x8	Door; red with white band.
IDBLKDOR.BM	16x16	Door; grey with embossed circular grid and red and white lights.
IDCMPCTR.BM	8x16	Door; dark grey with features (can use just top 12).

IDDET.BM	8x32	Wall; dark grey panelling with features and white ceiling light.
IDDET1.BM	8x8	Door; dark grey with features and white light.
IDDOOR1.BM	8x8	Door; standard grey with ribbing.
IDDTENTW.BM	16x16	Arch; light grey.
IDFUEL1.BM	8x16	Wall; grey with yellow warning strip and red lights.
IDFUEL2.BM	8x16	Wall; grey with yellow warning strips and ribbing.
IDGROOVE.BM	1x1/4	Door track; dark grey with lighter edges.
IDHATCH1.BM	8x4	Wall; grey with ribbing and two yellow stripes.
IDHATCH2.BM	8x4	Wall; grey with ribbing and two yellow stripes and red light.
IDISO.BM	16x32	Door; dark grey, embossed Imperial circle with silver edging and radiating, beige bands.
IDJAMLCX.BM	8x8	Floor; dark grey with horizontal strip of white lights.
IDJAMLCY.BM	8x8	Floor; dark grey with vertical strip of white lights.
IDJAMLRD.BM	4x1	Door track; dark grey with red light.
IDJAMLW.BM	4x1	Door track; dark grey with white light.
IDJAMPNL.BM	4x8	Door track; grey with white lights and controls.
IDLOGOGN.BM	16x16	Door; grey with embossed Imperial circle and green lights.
IDLOGORD.BM	32x32	Door; grey with red Imperial circle and green lights.
IDMARBB1.BM	8x8	Door; mottled black with patterned, silver side panels.
IDMARBY1.BM	8x8	Door; mottled yellow with patterned, silver side panels.
IDMGCRT2.BM	8x8	Crate; blue grey with blast hole.
IDMGCRT3.BM	8x8	Crate; dark yellow with blast hole.
IDMGDCRT.BM	8x8	Crate; dark beige with blast hole.
IDSECB1.BM	16x16	Door; light grey with red markings in two columns.
IDSECB2.BM	8x8	Crate; dark beige, no markings.
IDSECB3.BM	16x8	Door, sliding; dark beige, half ribbed, half plain with red light.
IDSECBA.BM	8x8	Door, sliding; dark beige with ribbing and switch (left side).
IDSECBB.BM	8x8	Door, sliding; dark beige with ribbing (centre, slash).
IDSECBC.BM	8x8	Door, sliding; dark beige with ribbing (centre, backslash).
IDSHLDLX.BM	8x8	Door track; vertical white light with borders; used as hangar air shield.
IDSHLDLY.BM	8x8	Door track; horizontal white light with borders; used as hangar air shield.
IDTSTB1.BM	8x8	Door; light grey with dark panel and red markings.
IEDETSKY.BM	32x32	Sky; night sky through dark clouds.
IERAMSKY.BM	32x32	Sky; bright blue.
IESTARS.BM	16x16	Sky; black with scattered stars.
IETSTSKY.BM	32x32	Sky; orange with cloud furrows.
IF1.BM	8x8	Floor; grey, cubist doodles.
IF2.BM	8x8	Floor; grey plasma.
IF3.BM	8x8	Floor; dark grey panelling.
IFDSHDOW.BM	8x8	Wall; grey shadow of grating.
IFFUEL1.BM	8x8	Wall; grey shadow of circular grating.
IFORCFLD.BM	8x8	Floor; four vertical strips of green lights (force field).
IFRCFLD2.BM	8x8	Floor; four horizontal strips of green lights (force field).
IPACONVX.BM	8x8	Floor; vertical, grey beam with circles and frills with blue and red pipes.
IPACONVY.BM	8x8	Floor; horizontal, grey beam with circles and frills with blue and red pipes.
IPADARK3.BM	8x8	Floor; grey, plasma tile.
IPADARK4.BM	8x8	Floor; grey cubist.
IPADARK6.BM	8x8	Floor; vertical bands of grey and dark grey.
IPAFLOR1.BM	8x8	Floor; grey panelling.
IPAGRD1Y.BM	8x8	Floor; grating of crossed grey and light grey bars.
IPAGRD2.BM	8x8	Floor; dark grey, triangular grating.
IPAGRD3X.BM	8x8	Floor; dark grey, triangular grating over two horizontal, silver pipes.
IPALITE0.BM	8x8	Ceiling; dark grey panelling with white light down the right side.
IPALITE1.BM	8x8	Ceiling; dark grey panelling with white light across the top.
IPALITE2.BM	8x8	Ceiling; grey bars in union jack with circular light, extinguished.
IPALITE3.BM	8x8	Ceiling; grey bars in union jack with circular, yellow light.
IPATEC1.BM	8x8	Wall; exposed tubing behind light grey bars and panelling.
IPATEC3Y.BM	8x8	Wall; exposed tubing behind light grey bars.
IPATEC4Y.BM	8x8	Wall; exposed tubing behind light grey bars.
IPATEC5Y.BM	8x8	Wall; exposed tubing and light grey bars.
IPBIGLT.BM	8x8	Ceiling; grey with oval, white light.
IPBRIDGE.BM	8x8	Wall; light and dark grey, rectangular patterns.
IPCOMSLT.BM	8x8	Floor; grey, horizontal planking.
IPCOMTOP.BM	8x8	Floor; black circuit board with white lights.
IPCPAN2.BM	8x32	Wall; beige with embossed, trapezoidal panels.

IPCPAN3.BM	16x32	Door; grey bulkhead door.
IPCPAN8.BM	32x32	Wall; grey panelling with some exposed tubing.
IPCUPLER.BM	16x16	Wall; large, vertical, silver tube over beige panelling with yellow hazard stripes.
IPDET1.BM	8x8	Floor; natural, light grey stone.
IPDETEL2.BM	8x8	Wall; light grey, bordered, cross-hatch grating with white lights.
IPDETPIP.BM	8x8	Wall; light grey ribbed columns with white lights.
IPDETPNL.BM	8x8	Wall; coloured computer readout.
IPDETSQR.BM	8x8	Wall; dark grey panelling with grey border.
IPDTENBL.BM	8x8	Wall; dark grey stripes/columns.
IPDTENGR.BM	8x8	Floor; beige, filled grating.
IPDTENRD.BM	8x8	Floor; beige grating with red lights.
IPDUCTC.BM	8x8	Wall; grey beige bars over grey panelling with white light.
IPDUCTG.BM	8x8	Ceiling; light grey air ducts between grey panels.
IPEXCEIL.BM	8x8	Ceiling; light grey panelling with oval, white light.
IPEXELV.BM	8x8	Wall; grey ribbing with recessed, vertical tubes.
IPEXFLR.BM	8x8	Floor; grey cubist.
IPEXFLR2.BM	8x8	Floor; white cubist.
IPEXFLR3.BM	8x8	Floor; light grey cubist.
IPGATE1Y.BM	8x8	Door; dark grey, patterned, barred gate with red lights.
IPGRDBLX.BM	8x8	Floor; grey tiles with horizontal highlight.
IPGRDBLY.BM	8x8	Floor; grey tiles with vertical highlight.
IPGRDGRY.BM	8x8	Floor; light grey tiles with vertical highlight.
IPGREYC2.BM	8x8	Floor; light grey, mottled.
IPHANGR1.BM	8x8	Ceiling; very dark grey panelling with red and white lights.
IPJAMLRX.BM	8x8	Floor; dark grey with horizontal strip of red lights.
IPJAMLRY.BM	8x8	Floor; dark grey with vertical strip of red lights.
IPMONTRS.BM	8x8	Wall; computer screens in grey panelling.
IPOCTGR.BM	8x8	Floor; grey pattern of octagonal wheels.
IPOVAL.BM	4x2	Door track; dark grey with oval cavity.
IPOVAL2.BM	8x8	Wall; dark grey with oval cavities.
IPPIPEX.BM	8x8	Ceiling; light grey horizontal pipe in grey panels, and grate.
IPPIPEY.BM	8x8	Ceiling; light grey vertical pipe in grey panels, and grate.
IPPOOLBL.BM	8x8	Floor; grey, circular highlight.
IPPOOLGR.BM	8x8	Floor; light grey, circular highlight.
IPRAM1.BM	8x8	Wall; grey panelling with crossing supports.
IPRAM2.BM	8x8	Floor; white square with grey border.
IPRAM3.BM	8x8	Wall; grey panelling with horizontal bar.
IPRAM4.BM	8x8	Wall; grey grating over red and grey pipes.
IPRAMBLT.BM	8x8	Wall; silver, pinched columns.
IPRECTGR.BM	8x8	Floor; light grey rectangle in white.
IPRMCRT1.BM	8x8	Crate; dark beige, marked 3K.
IPRMCRT2.BM	8x8	Crate; dark yellow, marked Danger Bio-Test.
IPRMCRT3.BM	8x8	Crate; dark yellow, marked 3K.
IPRMCRT4.BM	8x8	Crate; dark beige, marked with Imperial circle.
IPRMCRT5.BM	8x8	Crate; dark beige, marked Class 5 Explosives.
IPRMCRT6.BM	8x8	Crate; dark beige, marked 1 Ton Rubber Duck.
IPRMCRT7.BM	8x8	Crate; blue grey, marked Shields.
IPRMCRT8.BM	8x8	Crate; blue grey, markings obscured.
IPRMCRT9.BM	8x8	Crate; dark yellow, marked Power Cells.
IPSDENGN.BM	16x16	Wall; grey with horizontal cylinder and yellow warning stripes.
IPSEC1.BM	8x8	Floor; light grey, mottled.
IPSEC1B.BM	16x16	Wall, exterior; light grey, mottled, with blast hole in centre.
IPSEC3.BM	8x8	Wall; grey panelling.
IPSQAR2.BM	8x8	Ceiling; small, light grey cross-hatch on grey with white lights.
IPSQUAR1.BM	8x8	Ceiling; light grey cross-hatch on dark grey with white lights.
IPSTGRID.BM	8x8	Ceiling; grey, concrete union jack.
IPTRSHC1.BM	8x8	Floor; metal garbage in brown sewerage.
IPTSTB1.BM	8x8	Floor; black tiles with silver edging.
IPTSTB2.BM	8x8	Wall; light grey, panelled columns with red stripe.
IPTSTB3.BM	8x8	Ceiling; dark grey panelling with light.
IPTSTB4.BM	8x8	Ceiling; white with grey border and light blue light.
IPTSTB5.BM	8x8	Ceiling; white with grey border.
ISECBSKY.BM	32x32	Sky; black night with stars and clouds.
IW8DIGIT.BM	4x16	Switch; eight red letters, each 2x2.

IWABGWHT.BM	32x16	Wall; white panelling, red-striped column, grey panel and exposed tubing.
IWACONV1.BM	8x16	Wall; grey with grill and large, red hazard stripes.
IWACONV2.BM	8x16	Wall; grey with grill, red hazard stripes and grey beam with black rectangles.
IWACONV3.BM	8x16	Wall; grey with grill, red hazard stripes and grey beam.
IWACONV4.BM	8x16	Wall; grey with grill.
IWADARK1.BM	8x16	Wall; dark grey panelling.
IWADARK3.BM	8x16	Wall; grey, mottled.
IWADARK4.BM	8x16	Wall; grey plasma.
IWAPIPE1.BM	16x16	Wall; grey, mottled with horizontal, silver pipe.
IWAPIPE2.BM	16x16	Wall; grey, mottled with silver pipes.
IWAPIPE3.BM	16x16	Wall; grey, mottled with silver pipes.
IWAPIPE4.BM	32x16	Wall; grey, mottled with horizontal, silver pipes.
IWAPIPE5.BM	8x16	Wall; grey, mottled with two vertical, silver pipes.
IWAPIPE6.BM	16x16	Wall; grey panelling with light grey pipes and covering panels.
IWAPIST1.BM	32x16	Wall; grey with four, moving pistons and orange light.
IWARC1.BM	16x16	Wall; blue grey panelling with red paint streak and features.
IWARED0.BM	4x16	Wall; light grey with red hazard stripes.
IWARED4.BM	8x16	Wall; grey panel with red hazard stripes and exposed tubing.
IWASEQUE.BM	8x32	Switch; four panels (each 8x8): red, striped wall with exchange coupling with (3) and without (1) sequencer charge.
IWATEC3.BM	8x16	Wall; grey with exposed, light grey machinery.
IWATEC4.BM	4x16	Wall; dark grey, horizontal panelling with white pipes.
IWAVGEXT.BM	16x32	Wall; light grey with trapezoidal panel and white light.
IWBGFUEL.BM	16x32	Wall; light grey with crossed ribbing and white centre light.
IWBGFUL2.BM	16x32	Wall; grey panelling with horizontal, silver pipe.
IWBGFUL3.BM	16x16	Wall; grey panelling with crossing supports.
IWBGPIPE.BM	16x16	Wall; mauve pipe and panelling.
IWBLKHED.BM	16x16	Arch; light grey, circular (only lower 12).
IWBRWIN2.BM	16x8	Window; grey panelling around five-sided window.
IWCMPCTR.BM	8x16	Wall; grey concrete with rust stripes.
IWCOMBLO.BM	8x16	Wall; black circuit board with white lights.
IWCOMSLT.BM	8x16	Wall; blue grey with patterned, vertical panels.
IWCOMWAL.BM	8x16	Wall; grey, vertical grating with white lights.
IWDET1.BM	8x16	Wall; two light grey squares.
IWDET10.BM	16x16	Wall; mottled, grey concrete with vertical grating.
IWDET2.BM	8x16	Wall; grey panelling with ceiling skirting.
IWDETAL2.BM	8x16	Wall; grey, large-ribbed with minor rust.
IWDETEL3.BM	8x16	Wall; grey with vertical strip of small, red lights.
IWDETELV.BM	16x16	Door; grey with embossed hour glass, vertical runners and controls.
IWDHUGEZ.BM	16x32	Wall, exterior; light grey, ribbed concrete edifice.
IWDKHALL.BM	8x16	Wall; blue grey with sunken, squared oval.
IWDMGED1.BM	8x16	Wall; cracked, grey concrete with rust stripes.
IWDMGED2.BM	8x16	Wall; grey concrete with rust stripes and blast hole.
IWDMGED3.BM	8x16	Wall; grey panelling with riveted crack.
IWDMGED4.BM	8x16	Wall; grey panelling with blast hole.
IWDNLIT.BM	8x16	Wall; grey panelling with white light and features.
IWDNLITI.BM	16x16	Wall; grey panelling with white light and features.
IWDSHALL.BM	16x16	Wall; grey panelling with vertical strips of white light.
IWDSTALL.BM	8x32	Wall; grey panelling.
IWDTCHRT.BM	8x8	Sign; picture of path through lifts on grey (actually 6x6).
IWDTEN.BM	4x16	Wall; dark grey with sunk panel below horizontal ribbing.
IWELPANL.BM	2x4	Sign; grey floor indicator panel for lifts.
IWENGBAC.BM	32x16	Wall, exterior; dark grey engine exhausts in dark grey wall.
IWENGSID.BM	64x16	Wall, exterior; side of ships engines in grey wall.
IWEX1.BM	16x16	Wall; grey panelling.
IWFRAME1.BM	16x4	Wall, short; grey panelling with some lights.
IWFRAME2.BM	16x4	Wall, short; grey panelling with computer screens.
IWFTANK.BM	16x32	Wall; grey panelling with column and crossed struts on each panel.
IWFUEL1.BM	16x16	Wall; grey with red, silver and yellow pipes and circular ceiling pattern.
IWFUEL2.BM	16x16	Wall; grey with red and yellow pipes, valve and circular ceiling pattern.
IWFUEL3.BM	8x16	Wall; grey with yellow pipes and circular ceiling pattern.
IWFUEL4.BM	8x16	Wall; grey with red and yellow pipes behind diagonal grating and circular ceiling pattern.
IWFUEL5.BM	8x16	Wall; beige air duct with circular ceiling pattern.
IWFUEL7.BM	8x16	Wall; grey with red, silver and yellow pipes and switches.

IWFUEL8.BM	8x16	Wall; screens and gratings between light grey ribbing.
IWFUEL9.BM	16x16	Wall; beige air duct with yellow switch and circular ceiling pattern.
IWFUELS.BM	8x16	Sign; screen showing rotation of docking corridor in red and green (2 panels, each 8x8).
IWGRAN5.BM	16x32	Wall, exterior; light grey, vertically mottled.
IWGRAN6.BM	16x32	Wall, exterior; light grey, vertically mottled.
IWGREYC2.BM	16x16	Wall, exterior; light grey, mottled with horizontal divider.
IWLIFTER.BM	16x32	Wall; dirty, grey panelling with vertical tubes emerging and yellow lightning marking.
IWLITE.BM	2x4	Door track; light grey with white light strips.
IWLOGORD.BM	8x32	Wall; tall, grey panel with features and white edge lights.
IWLOWCON.BM	8x16	Wall; three light grey rectangles.
IWMNTRS.BM	8x16	Wall; grey panelling with computer screens.
IWNOLIT.BM	8x16	Wall; grey, speckled panelling with embossed hour glass.
IWPANEL1.BM	16x8	Wall; light grey control panel with many lights.
IWPANEL2.BM	16x8	Wall; light grey control panel with dual screen and lights.
IWPANEL3.BM	16x8	Wall; light grey control panel with many lights.
IWRAM1.BM	16x32	Wall; dark grey panelling with grates, spars and rust.
IWRAM2.BM	16x16	Wall; white panelling with small computer screens.
IWRAM3.BM	8x16	Wall; dark grey panels with horizontal red and white strips.
IWRAM4.BM	8x16	Wall; grey panelling with grates and blue and white tubing.
IWRAM5.BM	16x16	Wall; white panelling with multiple, small screens.
IWRAM6.BM	16x16	Wall; white panelling with blue light panels.
IWRAM7.BM	4x16	Wall; plain, white panelling.
IWRAM8.BM	4x16	Wall; plain, white panelling.
IWRAMELV.BM	8x8	Wall; dark grey with side gratings and small, white lights.
IWRAMGL1.BM	8x8	Wall; screen with red, conical schematic.
IWRAMGL2.BM	8x8	Wall; screen with red, circular schematic.
IWRAMON1.BM	8x8	Wall; brown screen with red, orbital schematic.
IWRAMON2.BM	16x8	Wall; brown screen with green, planetary schematic.
IWRAMON3.BM	8x8	Wall; brown screen with red, conical schematic.
IWRCPORT.BM	16x16	Wall; mauve panelling with circular grating and waste exhausts.
IWRDBL.BM	2x4	Door track; dark grey with red light and blue light strips.
IWREACT1.BM	16x16	Wall; grey panelling with vertical, white light strips.
IWRISER.BM	2x1	Door track; dark grey with red light.
IWSCONBL.BM	8x16	Wall; grey panelling with small, fluorescent light.
IWSCONGR.BM	8x16	Wall; light grey panelling with small, fluorescent light.
IWSECB2.BM	8x16	Wall; grey with centred panels and protruding top panel.
IWSECB3.BM	8x16	Wall; light grey with centred panel and vertical ribs.
IWSECB4.BM	8x16	Wall; grey panelling with vertically striped centre panel.
IWSECB5.BM	8x16	Wall, exterior; brown stone blocks.
IWSECBS1.BM	8x32	Wall; tall, grey panel with features.
IWSHIP0.BM	64x16	Wall, exterior; grey side of ship with peeling, red stripe.
IWSTRIP.BM	4x8	Wall; light grey, vertical strip grating.
IWTALL1.BM	16x32	Wall; light grey with trapezoidal panel and white floor light.
IWTALL2.BM	8x32	Wall; tall, grey panel with features.
IWTSTB1.BM	32x16	Wall; grey panelling with light and light grey skirting at ceiling with red stripes.
IWTSTB2.BM	8x32	Wall; grey panelling with two light grey horizontals with red stripes.
IWTSTB4.BM	8x16	Wall; light grey panelling with pattern of red stripes.
IWTSTBRC.BM	16x32	Wall; grey panelling with light, tubing and duct vent.
IWURB1.BM	16x32	Wall; dark grey, mottled vertical with grey, mottled side panels.
IWURB2.BM	16x32	Wall; dark grey, angled panels with grey, mottled side panels.
IWURB3.BM	16x32	Wall; dark grey, mottled vertical with yellow light and grey, mottled side panels.
IWWRHSE1.BM	8x16	Wall; light grey, part ribbed with centre strip of white lights.
JDBIGDR1.BM	16x16	Door; silver door with exposed machinery and yellow half circle.
JDDOOR1.BM	8x8	Door; dirty, grey with riveted panels, six circular holes in two columns and red light.
JDDOOR2.BM	8x8	Door; grey with horizontal bands and orange light.
JDLILDR1.BM	16x8	Door; silver door with exposed machinery and yellow half circle.
JDLILDR2.BM	16x8	Wall; orange panel with white lamps and exposed, grey tubing.
JPBLLT.BM	8x8	Wall; top orange, bottom blue.
JPCIRCL1.BM	8x8	Door; orange with centred circle.
JPCIRCL2.BM	8x8	Floor; orange with spoked wheel.
JPCOMB5.BM	8x8	Wall; orange with blue band and strip of orange lights.
JPDIMON1.BM	8x8	Floor; orange circles and triangles.
JPELEV1Y.BM	8x8	Wall; two vertical, grey pipes separated by diagonal grating, over orange.
JPLINES1.BM	8x8	Floor; orange panelling.

JPLINES2.BM	8x8	Floor; orange panelling with diagonal band.
JPLINES3.BM	8x8	Floor; orange panelling.
JPMACH1.BM	8x8	Wall; orange panelling with exposed machinery.
JPSQUAR1.BM	8x8	Floor; dark orange, speckled with cuneiform border.
JWBIG01.BM	16x32	Wall; light grey panelling with exposed tubing below orange, with ornate roof skirting.
JWCIRCL1.BM	8x16	Wall; orange with blue stripes and embedded, grey circle connected to control panel.
JWCIRCL2.BM	4x16	Wall; orange with blue stripes and vertical, grey pipe.
JWCIRCL3.BM	4x16	Wall; orange with blue stripes.
JWCOLUMN.BM	8x16	Wall; orange with column and grey, barred grating.
JWCOMB1.BM	8x16	Wall; orange with patterning and light grey panelling with orange light.
JWCOMB2.BM	4x16	Wall; orange with patterning and light grey panelling.
JWCOMB3.BM	4x16	Wall; orange with blue stripes and grey piping.
JWCOMB4.BM	8x16	Wall; orange and dark grey with grey piping.
JWFANCY2.BM	16x8	Wall; orange with archaic patterns and hanging, grey pipes.
JWGRATE1.BM	8x16	Wall; orange with patterning and grey, barred grating.
JWPIGHED.BM	8x16	Wall; orange with patterning and assorted, grey tubes and bars.
JWPLAIN.BM	8x16	Wall; orange with patterning.
JWRELIEF.BM	32x16	Wall; brown bass relief of Jabba in his throne room.
JWSHPEXT.BM	32x16	Wall; dark orange with grey and dark grey pipes and panels.
NDBROWN.BM	8x8	Door; painted, orange door with ribbing and diagonal cross.
NDCORUG1.BM	16x16	Wall; grey roll-a-door with red, circular graffiti (check offset when using).
NDCORUG2.BM	16x16	Wall; grey roll-a-door with red graffiti and blast hole (check offset when using).
NDJAMS1Y.BM	2x8	Door track; light and dark grey stripes with red markings.
NDJAMS2X.BM	8x2	Door track (horizontal); grey with white light.
NDJAMS2Y.BM	2x8	Door track; grey with white light.
NDLIGHT.BM	8x8	Door; grey with red hazard stripes and white light.
NDPIPES4.BM	16x16	Door; grey with ribbing, exposed tubing and yellow hazard stripes.
NDREDOTS.BM	8x8	Door; grey with inverted Y panelling and red lights.
NDSTRONG.BM	8x8	Door; dark grey with ribbing and partially exposed lock bars.
NDWARN7.BM	16x16	Door; dark grey hatch with yellow hazard stripes (door is centre 10x12).
NENARSKY.BM	32x64	Sky; window-lit sky scrapers with coloured star field.
NPBPIPE1.BM	8x8	Wall; two large, grey, vertical pipes with red markings.
NPBPIPE2.BM	8x8	Wall; two large, grey, vertical pipes with red markings and crossed, grey ribbing.
NPBRN02.BM	8x8	Floor; circular, rust-red pattern.
NPBRN04.BM	8x8	Wall; orange and brown with air ducts and two red lights.
NPBRN05.BM	8x8	Floor; brown, ducted grating.
NPGREY01.BM	8x8	Wall; grey, vertically mottled.
NPGREY02.BM	8x8	Wall; grey, vertically mottled with lighter stains.
NPGRYBAR.BM	8x8	Floor; horizontal, grey beam on dark grey.
NPGRYLT.BM	8x8	Floor; grey with faint, vertical stripes.
NPLIT1.BM	8x8	Floor; dirty, grey grating with two vertical strips of white lights.
NPLIT4.BM	8x8	Ceiling; circular, white light on dark grey.
NPLIT4SH.BM	8x8	Floor; shadow of circular light on grey.
NPPANEL1.BM	8x8	Wall; dirty, grey panelling.
NPPIPES1.BM	8x8	Wall; mess of white pipes.
NPPIPES2.BM	8x8	Wall; mess of white pipes, obscured by grey panelling.
NPSUPORT.BM	8x8	Wall; plain light grey crossed by dirty, grey struts.
NPVENTS1.BM	8x8	Wall; dirty, grey, horizontal gratings with decorative strip.
NPVENTS2.BM	8x8	Floor; dirty, grey, horizontal gratings.
NPWRNFAD.BM	8x8	Floor; light grey with diagonal, yellow stripes.
NSSIGN01.BM	8x8	Sign; red, Imperial circle with red writing.
NSSIGN03.BM	8x8	Sign; white poster with red, Rebel symbol, black writing and blue stripes.
NSSIGN04.BM	8x8	Sign; blue and red writing, yellow and red stripes, on grey.
NSSIGN05.BM	4x8	Sign; dark yellow with hazard border.
NSSIGN06.BM	4x4	Sign; white with red left arrow and split circle in blue and white.
NSSIGN07.BM	4x4	Sign; white with red right arrow and winged, black marking.
NSSIGN09.BM	4x4	Sign; white with black writing, yellow circle and red box.
NSSIGN10.BM	4x4	Sign; dark grey with blue, red and white writing and circular marking.
NUM6-OFF.BM	4x4	Switch; red switch handle on white, down position.
NWARCH1.BM	8x16	Wall; lit, grey panel on dark grey, with red writing.
NWARCH2.BM	8x16	Wall; lit, grey panel on dark grey, with red writing and strip of triangular grating.
NWBEVEL1.BM	4x16	Wall; dark grey with striped panelling, slime and protruding, red light.
NWBEVEL3.BM	4x16	Wall; dark grey with grating and protrusions.
NWBEVEL4.BM	4x16	Wall; dark grey with olive stripe and protrusions.

NWBEVEL5.BM	8x16	Wall; dark grey with white light and protrusion.
NWBIGGIE.BM	32x64	Wall; dirty, grey panelling with yellow light strips.
NWBPIPE2.BM	16x16	Wall; dirty, grey panelling with two large, grey pipes with red markings.
NWBPIPE3.BM	16x16	Wall; dirty, grey panelling with two large, grey pipes with red markings and grey ribbing.
NWBRACE.BM	16x16	Wall; dirty, grey panelling with grey struts and two bands, in olive and grey.
NWBRN01L.BM	8x16	Wall; dark red and brown with white light strip and features.
NWBRN02.BM	8x16	Wall; dark orange with grating, red lights and strange pattern.
NWBRN03L.BM	8x16	Wall; dark orange with grating and yellow floor light.
NWEXT01.BM	16x16	Wall; dirty, grey panelling.
NWEXT02.BM	16x16	Wall; dirty, grey panelling with large, grey panel.
NWEXT06.BM	16x16	Wall; dirty, grey panelling with grating and grey bricks.
NWEXT07D.BM	32x16	Wall; dirty, grey panelling with grating, grey bricks and colourful graffiti.
NWEXT08D.BM	32x16	Wall; dirty, grey panelling with grating, grey bricks, colourful graffiti and white, Rebel poster.
NWEXT09.BM	16x16	Door; dark grey with ribbing.
NWEXT2.BM	16x16	Wall; dirty, grey panelling.
NWGREY01.BM	8x16	Wall; grey, vertically mottled.
NWGREY02.BM	8x16	Wall; grey, mottled with indentations at top and bottom.
NWGRN03.BM	8x16	Wall; dark grey with olive hazard stripes.
NWGRN06.BM	16x16	Wall; dirty, grey panelling with large, green panel and red band.
NWGRN08.BM	16x16	Wall; dirty, grey and green panelling with grating.
NWGRN09.BM	16x16	Wall; grey and green.
NWGRN10.BM	8x16	Wall; grey with green, panelled pattern.
NWGRN12.BM	8x16	Wall; grey with green, panelled cross.
NWNOTCH.BM	8x16	Wall; black, trapezoidal panelling.
NWPIPES2.BM	16x16	Wall; grey panelling with band of exposed piping.
NWRISER.BM	8x2	Door track (horizontal); grey with oval cavities.
NWTV1.BM	8x16	Screen; three panels of green writing over red Imperial circle (can be used separately).
NWTV2.BM	8x16	Screen; three panels of red circle and blue writing expanding on quartered yellow & orange background (can use separately).
NWTV3.BM	8x32	Screen; six panels, one black, five of white static (can be used separately).
NWWARN1.BM	8x16	Wall; dirty, grey with two bands of yellow hazard stripes.
NWWARN2.BM	8x16	Wall; dirty, grey with two bands of yellow hazard stripes and panelling.
NWWARN4.BM	8x16	Wall; dirty, grey with two bands of yellow hazard stripes and triangular ribbing.
NWWARN4L.BM	4x16	Wall; dirty, grey with two bands of yellow hazard stripes and triangular ribbing (left end).
NWWARN4R.BM	4x16	Wall; dirty, grey with two bands of yellow hazard stripes and triangular ribbing (right end).
NWWHT01.BM	16x16	Wall; white with gratings and diagonal, green stripe.

TEXTURES.GOB (R-Z)

[by Paulius Stepanas]

X and Y are the width and height of the texture in game units (multiply by 8 for the size in pixels).

Texture	XxY	Description
RDBIG2.BM	16x16	Door; white with Y panel and features.
RDBIG4.BM	16x16	Door; light grey with gratings, ribbing and rust.
RDJAML1Y.BM	2x16	Door track; light grey with black, hollow centre.
RDRED01.BM	8x8	Door; red with grey Y panel.
RDRED02.BM	8x16	Door; red with grey Y panel (can use lower 10).
RDRED03.BM	8x8	Door; red with grey Y panel, mirrored across middle.
RDRED04.BM	16x8	Door; red with grey Y panel, green delta and side panels.
RDREDJX.BM	8x8	Wall; red panelling with green lights, divisions horizontal.
RDREDJY.BM	8x8	Wall; red panelling with green lights, divisions vertical.

RDTUBE.BM	16x16	Wall; light grey with gratings, ribbing and rust.
RESKY01.BM	16x32	Sky; pale blue with clouds low.
RESKY02.BM	16x32	Sky; pale blue with clouds high.
RFICE01.BM	8x8	Floor; pale blue, mottled (ice).
RFICYDAG.BM	8x8	Floor; pale blue, diagonally mottled (ice).
RFICYEW.BM	8x8	Floor; pale blue, horizontally mottled (ice).
RPACID01.BM	8x8	Floor; dark yellow, mottled.
RPCONVBX.BM	8x8	Floor; grey grating on left, blue and red tubes on right.
RPCONVBY.BM	8x8	Floor; grey grating at top, blue and red tubes at bottom.
RPDANGR2.BM	4x8	Sign; yellow and red warning with skull.
RPGRIDDK.BM	8x8	Floor; dark grey, triangular grating.
RPGRIDMD.BM	8x8	Floor; grey, triangular grating.
RPIBEAM.BM	8x8	Ceiling; white, horizontal beam over grey, crossed supports.
RPIBEAM2.BM	8x8	Ceiling; white, vertical beam over grey, crossed supports.
RPICE02.BM	8x8	Floor; blue, mottled (ice).
RPMIXTOP.BM	8x8	Floor; grey and blue planks with slime.
RPSNOW3.BM	8x8	Floor; white, mottled (snow).
RPURINE1.BM	8x8	Floor; white with embossed union jack around drain.
RWACID02.BM	8x64	Wall; white panelling with red hazard stripes, in four parts with slime at the base.
RWBIGGO.BM	16x32	Wall; white with red hazard stripes, multiple beams and tubes.
RWCLEAN1.BM	8x16	Wall; white panelling with red band.
RWCLEAN2.BM	4x16	Wall; white panelling.
RWCLEAN3.BM	8x16	Wall; white panelling overlaid with grey plating.
RWCOLENR.BM	2x16	Wall; grey with grill (end piece for RWCOLTEC.BM).
RWCOLTEC.BM	16x16	Wall; grey with red stripe, grill, tubing and white lights.
RWCOLUM1.BM	4x16	Wall; grey with grill.
RWCOLUM3.BM	4x16	Wall; grey with red stripe and grill.
RWCONV1L.BM	8x16	Wall; light grey with red hazard stripes and slime.
RWEXTI01.BM	8x16	Wall; white with dripping slime.
RWEXTI03.BM	8x32	Wall; white with dripping slime.
RWFAN.BM	64x8	Wall; white panelling with grey fin (used for fan turbine).
RWFANSWT.BM	8x128	Switch; grey wall with red stripe, grill and white lights, with turbine switch; 8 panels (1 off, 4 on, 3 without switch).
RWGEARS.BM	8x16	Wall; white with grating strip (can be used in two parts).
RWIBEAM1.BM	8x64	Wall; grey, vertical beam with rust and snow at base.
RWIBEAM2.BM	8x16	Wall; grey, vertical beam with rust.
RWIBEAM3.BM	32x16	Wall; white, hydraulic support with red stripe, grey, vertical beams and snow at base.
RWICE01.BM	16x16	Wall; blue, rippled ice.
RWICEFAL.BM	8x16	Wall; white waterfall.
RWMIXER1.BM	16x32	Wall; toothed side of mixer with slime on lower half.
RWRWALL1.BM	4x16	Wall; grey with grill and white lights.
RWRWALL2.BM	4x16	Wall; grey with vertical light strips and aquamarine panelling.
RWSEQUEN.BM	8x64	Switch; white panelling with red band, with and without (1) exchange coupling, with (2) and without (1) sequencer charge.
RWURINE1.BM	16x16	Wall; white with red-striped panel.
RWURINE2.BM	4x8	Wall; grey urinal with drainage slots and accumulated scum.
SALTBULB.BM	8x16	Ceiling; circular, white light on dark grey, both on and off.
SDGATE1.BM	8x8	Door; corroded, grey sluice gate.
SDGRATE1.BM	8x8	Door; corroded, grey sluice gate with grating.
SDJAM1Y.BM	2x8	Door track; corroded, grey door track.
SDJAMLRG.BM	2x16	Wall; grey panelling with vertical strips and red rust.
SDROUNDM.BM	8x8	Arch; corroded grey with red lights.
SDROUNDT.BM	8x1	Arch top; top piece for SDROUNDM.BM
SESEWSKY.BM	32x32	Sky; orange with cloud furrows.
SPBARS1.BM	8x8	Wall; grey, metal bars with red rust.
SPBARS2.BM	8x8	Wall; grey, metal bars with red rust and hung wiring.
SPBARS3.BM	8x8	Floor; grey, cross-hatched, metal bars with red rust.
SPCMENT1.BM	8x8	Floor; light grey flagstone with green mould.
SPCMENT2.BM	8x8	Floor; light grey, mottled with orange rust.
SPDRAIN1.BM	8x8	Floor; dark grey, diamond drain with rust.
SPGRATE1.BM	8x8	Floor; dark grey grate with circular holes and rust.
SPMETAL1.BM	8x8	Floor; grey and red rust, mottled.
SPPIT4.BM	8x8	Wall; grey and light grey horizontals with serious rust.
SPSEWGE1.BM	8x8	Floor; green and brown, bubbling sewerage.

SPSEWGE2.BM	8x8	Floor; brown, flowing sewerage.
SPSEWGE3.BM	8x8	Wall; brown, water-falling sewerage.
SPSTRIPX.BM	8x8	Floor; horizontal, grey and dark planks with rust.
SPSTRIPY.BM	8x8	Floor; vertical, grey and dark planks with rust.
SWCNTROL.BM	16x8	Wall; stained, grey with recessed, vertical piping and switch panel.
SWPIT1.BM	8x16	Wall; light grey with features and green stains.
SWPIT2.BM	8x16	Wall; light grey with grey band and rust stains.
SWPIT3.BM	8x32	Wall; light grey with two grey bands and rust stains.
SWPIT4.BM	8x16	Wall; light grey with two grey bands and rust stains.
SWPIT5.BM	8x64	Wall; light grey with grey bands and rust stains; top 24 may be used as wall with floor skirting.
SWREBAR1.BM	16x8	Wall; grey, horizontal ribbing with lighter filling and rust.
SWREBAR2.BM	16x8	Wall; grey, horizontal ribbing with rust.
SWSLANT1.BM	16x8	Wall; light grey, slanted blocks with green stains.
SWSWITCH.BM	4x16	Switch; five-position dial, each 2 high.
SWWALKW1.BM	8x2	Door track (horizontal); light grey with stains.
SWWALKW2.BM	8x4	Door track (horizontal); light grey with stains.
TDBIGDR4.BM	16x16	Door; silver with three panels and embossed circles, red lights and shell damage.
TIEWNG.BM	16x16	Wall; side of TIE Fighter wing (black with grey ribbing).
TPCOMP2.BM	8x8	Wall; white control panel with screens and red strip.
TPCRAKS1.BM	8x8	Floor; grey, irregular paving.
TPDIMNDY.BM	8x8	Floor; dark grey, diamond tiles.
TPGREY1.BM	8x8	Floor; beige, mottled.
TWBARLT1.BM	8x16	Wall; grey with white, central light strip and floor skirting.
TWBARLT3.BM	32x16	Wall; grey with light strip, skirting and damaged machinery.
TWCAP01.BM	16x32	Wall, exterior; light grey, featured edifice.
TWCAP03.BM	8x32	Wall, exterior; light grey, featured edifice.
TWCAP07.BM	8x32	Wall, exterior; light grey, featured edifice with bullet holes.
TWCAP08.BM	32x32	Wall, exterior; light grey edifice with shell-stripped concrete.
TWCOMP2D.BM	8x16	Wall; white control panel with broken screens and red strip.
TWDTOP1.BM	8x8	Wall, exterior; light grey cement with soot.
TWEXT01A.BM	16x16	Wall, exterior; shell-damaged cement with minor rust.
TWEXT01B.BM	16x16	Wall, exterior; cracked cement with minor rust.
TWEXT05A.BM	8x16	Wall, exterior; grey with blast hole.
TWEXT05B.BM	8x16	Wall, exterior; grey with shell damage.
TWEXT05C.BM	16x16	Wall, exterior; grey with bullet holes.
TWEXT06B.BM	8x16	Wall, exterior; grey with diagonal panels and bullet holes.
TWEXT06D.BM	16x16	Wall, exterior; grey with diagonal panels.
TWHANGR1.BM	8x16	Wall; dark grey horizontals with white supports.
TWILI10A.BM	8x16	Wall; orange with light grey panels.
TWILI10B.BM	8x16	Wall; orange with light grey, cracked panels.
TWILI10C.BM	8x16	Wall; orange with light grey, shock-stripped panels.
TWILIT01.BM	16x16	Wall; dark grey panelling with destroyed lights.
TWINT08A.BM	8x16	Wall; light grey panelling with destroyed switch.
TWINT08B.BM	8x16	Wall; light grey panelling with blast hole.
TWINT09A.BM	8x16	Wall; orange panelling with destroyed switch.
TWINT09B.BM	8x16	Wall; orange panelling with bullet holes.
TWINT12A.BM	16x16	Wall; light grey with soot.
TWINT12B.BM	16x16	Wall; light grey with soot and bullet holes.
WHFILL.BM	1/2x1/2	Filler; small, white square.
ZAEYE.BM	4x4 x2	Switch; open or closed, blue eye.
ZANAV.BM	8x8 x6	Sign, animated; blue nava card being decoded on grey with red light strips (each image actually 6x6).
ZASPIN.BM	8x8 x7	Sign, animated; rotating white light on grey with yellow-striped border.
ZASWIT00.BM	4x4 x2	Switch; red or green circle around white centre.
ZASWIT01.BM	4x4 x2	Switch; red or blue rectangle on light grey.
ZASWIT02.BM	4x4 x2	Switch; light grey dial with red or green light.
ZASWIT03.BM	4x2 x2	Switch; orange or blue hand.
ZASWIT04.BM	2x4 x2	Switch; red or blue light; one off, one on.
ZASWIT05.BM	2x4 x2	Switch; lever with coloured lights.
ZASWIT06.BM	4x4 x2	Switch; red switch handle on white.
ZASWIT07.BM	4x4 x2	Switch; red switch handle on grey with red or green light.
ZASWIT08.BM	4x8 x2	Switch; small switch on grey grill with lights.
ZASWIT09.BM	4x4 x2	Switch; coloured panel with red cross with or without red circle.

ZASWIT10.BM	4x4 x2	Switch; turning handle on grey with yellow strip.
ZASWIT11.BM	2x8 x2	Switch; lever on grey with red or green lighted border and down arrow.
ZASWIT12.BM	4x4 x2	Switch; red or blue rectangle on light grey.
ZASWIT14.BM	4x4 x2	Switch; red lever on green and coloured lights.
ZDARMOR2.BM	8x8	Door; light grey with vertical, hydraulic locking cylinders.
ZDARMOR3.BM	8x8	Door; light grey with vertical, hydraulic locking cylinders and rust (red in GROMAS).
ZDARMOR4.BM	8x8	Door; grey with vertical, hydraulic locking cylinders and rust (red in GROMAS).
ZDBIGDR1.BM	16x16	Door; light grey with two vertical, grill panels.
ZDBIGDR2.BM	16x16	Door; light grey with embossed hourglass.
ZDBIGDR3.BM	16x16	Door; silver with three panels and embossed circles.
ZDBIGDR4.BM	16x16	Door; silver with three panels and embossed circles, and red lights.
ZDEXT1.BM	8x8	Door; orange in grey frame.
ZDILOGO1.BM	16x16	Door; dark grey, embossed Imperial circle with silver edging.
ZDIMPER1.BM	8x8	Door; standard grey with ribbing.
ZDIMPER4.BM	16x16	Door; black with red circle.
ZDIMPER5.BM	16x16	Door; black with red circle and embossed features.
ZDIMPLG4.BM	16x16	Door; grey with red Imperial circle.
ZDIND1.BM	8x8	Door; grey with indentations at top and bottom and red writing.
ZDJAML1Y.BM	2x16	Wall; light grey with vertical panels (red in GROMAS).
ZDJAML2Y.BM	2x16	Door track; grey with blue light panels and switch (red in GROMAS).
ZDJAML3Y.BM	2x16	Door track; grey with blue light panels.
ZDJAMS2Y.BM	2x8	Door track; grey with vertical panels (red in GROMAS).
ZDJAMSM1.BM	2x8	Door track; red with vertical panels (same as ZDJAMS2Y).
ZDJMIN1X.BM	8x8	Floor; grey with blue light panels.
ZDMETAL1.BM	8x8	Door; grey with side grills and rust.
ZDMINEBG.BM	16x16	Door; light grey with dark, patterned grating and rust (red in GROMAS).
ZDREBL1.BM	8x8	Door; white with green light.
ZDREBL2.BM	8x8	Door; white with vertical stripes.
ZDREBL3.BM	8x8	Door; orange with embossed features.
ZDREBLT1.BM	8x8	Door; white with two panels and red lights.
ZDROBDR1.BM	16x8	Door; red with grey Y panel and side panels.
ZFBGGRID.BM	8x8	Ceiling; crossed bands of white light (vertical) and grey.
ZFBGRID2.BM	8x8	Ceiling; crossed bands of white light (horizontal) and grey.
ZMGRATE1.BM	8x8	Wall; white struts with exposed centre.
ZMJABMD1.BM	2x8	Door track; grey bars with joined circles at both ends.
ZMTUBE.BM	16x16	Arch; grey wall with tubes and white lights coming in from four directions to a circular hole.
ZPBLACK1.BM	8x8	Floor; black tiles.
ZPBLACK2.BM	8x8	Floor; black tiles with grey cross-hatch.
ZPBOLTD1.BM	8x8	Wall, exterior; grey, concrete blocks with riveted, metal border and rust.
ZPBOLTD3.BM	8x8	Door; dark grey metal bars.
ZPBRICK1.BM	8x8	Floor; white bricks.
ZPBRICK2.BM	8x8	Floor; orange bricks.
ZPBRICK3.BM	8x8	Floor; grey cobble stones.
ZPCIRCL1.BM	8x8	Floor; grey discs on white.
ZPCIRCL2.BM	8x8	Floor; four light grey circles between square tiles.
ZPCMENT1.BM	8x8	Floor; light grey, speckled with yellow stains.
ZPCMENT2.BM	8x8	Floor; grey, mottled, linked and riveted plates.
ZPCMENT6.BM	8x8	Wall, exterior; light grey cement blocks with dark border.
ZPCRUNFL.BM	8x8	Floor; light grey, vertical grating with frilled edges.
ZPDIAGS1.BM	8x8	Floor; white, speckled with grey diagonal.
ZPDIAGS2.BM	8x8	Floor; light grey, speckled with grey diagonal.
ZPDIMNDX.BM	8x8	Floor; grey, diamond tiles.
ZPDIRT01.BM	8x8	Floor; dark red, speckled.
ZPGPIPE1.BM	8x8	Wall; dark grey mess of pipes.
ZPGPIPE2.BM	8x8	Ceiling; dark grey mess of pipes with cross bars (red in GROMAS).
ZPGPIPE3.BM	8x8	Ceiling; crossed, red bars over dark mess of pipes.
ZPGPIPE4.BM	8x8	Ceiling; dark mess of pipes with red cross bars with two horizontal planks.
ZPGRASS3.BM	8x8	Floor; green, speckled (grass).
ZPGRDGRY.BM	8x8	Floor; light grey tiles with reflected, vertical highlight.
ZPGREY.BM	8x8	Floor; grey, mottled (red in GROMAS).
ZPGREY2.BM	8x8	Floor; grey/beige, mottled cubist.
ZPGREYP1.BM	8x8	Wall; light grey panelling with hung tube.
ZPGROMS1.BM	8x8	Floor; dirty, red tiles with rivets and interlocks.

ZPGRSPAT.BM	8x8	Floor; mottled, brown bricks (red in GROMAS).
ZPGRTE1X.BM	8x8	Floor; tight, grey grating.
ZPGRTE1Y.BM	8x8	Floor; tight, grey grating.
ZPGRTE2Y.BM	8x8	Floor; tight, grey grating with rust (red in GROMAS).
ZPGRTE3X.BM	8x8	Floor; loose, grey grating.
ZPGRTE3Y.BM	8x8	Floor; loose, grey grating.
ZPGRTE4Y.BM	8x8	Floor; loose, grey grating with rust.
ZPGRTE5Y.BM	8x8	Floor; silver grating with oval holes.
ZPGRTE6Y.BM	8x8	Wall; white, vertical stripes.
ZPGRTE8.BM	8x8	Floor; square, grey grate over pipe.
ZPGRYFIL.BM	1/8x1/8	Filler; small, dark grey square (red in GROMAS).
ZPHTEC2.BM	8x8	Wall; light grey roll-a-door.
ZPHTEC3.BM	8x8	Floor; orange.
ZPHTECX.BM	8x8	Floor; horizontal, orange planking.
ZPHTECY.BM	8x8	Floor; vertical, orange planking.
ZPHTECZ1.BM	8x8	Floor; diagonal, orange planking.
ZPIND1.BM	8x8	Wall; dark grey, mottled, with ribbing.
ZPIND3.BM	8x8	Wall; dark grey, mottled.
ZPINT19.BM	8x8	Wall; bright orange, vertical stripes.
ZPINT20.BM	8x8	Wall; white with string art.
ZPINT21X.BM	8x8	Floor; yellow and brown, horizontal stripes.
ZPJABMES.BM	8x8	Floor; beige mesh with highlight.
ZPJABPA1.BM	8x8	Wall; light grey jumble of riveted panels.
ZPJABPI1.BM	8x8	Wall; exposed tubing and light grey bars.
ZPJABPI2.BM	8x8	Wall; exposed tubing behind light grey bars.
ZPJABPI3.BM	8x8	Wall; dark grey with light grey tubes and blue marks.
ZPJABPI4.BM	8x8	Wall; grey jumble of riveted panels.
ZPLBLU1X.BM	8x8	Ceiling; dark grey quarters with horizontal, fluorescent tube.
ZPLBLU2Y.BM	8x8	Ceiling; grey with two vertical strips of fluorescent lights.
ZPLIT01X.BM	8x8	Floor; horizontal, white panelling with white strip light.
ZPLIT01Y.BM	8x8	Floor; vertical, white panelling with white strip light.
ZPLIT03X.BM	8x8	Ceiling; black with strips and horizontal, white strip lights.
ZPLIT03Y.BM	8x8	Ceiling; black with strips and vertical, white strip lights.
ZPLIT04.BM	8x8	Ceiling; large, circular, white light on dark grey.
ZPLIT06.BM	8x8	Ceiling; four circular, white lights on grey.
ZPLIT08.BM	8x8	Ceiling; bank of four square, white lights.
ZPLIT08D.BM	8x8	Ceiling; diagonally quartered, fluorescent, white lights.
ZPLIT09.BM	8x8	Ceiling; black grills, light grey tiles and square, white light.
ZPLIT10.BM	8x8	Floor; light grey, quartered by white strips.
ZPLIT15.BM	8x8	Ceiling; dark grey and white light squares divided by grey.
ZPMARBB1.BM	8x8	Wall; dark grey, vertical strips/panels.
ZPMARBB2.BM	8x8	Floor; black, mottled tile.
ZPMARBB3.BM	8x8	Floor; black, mottled with veins of blue.
ZPMARBE2.BM	8x8	Floor; black, mottled tile.
ZPMARBG3.BM	8x8	Floor; grey plasma.
ZPMARBL1.BM	8x8	Floor; white, mottled.
ZPMARBL2.BM	8x8	Floor; grey plasma.
ZPMARBR2.BM	8x8	Floor; grey, plasma bricks.
ZPMARBW2.BM	8x8	Floor; white plasma.
ZPMARBWF.BM	8x8	Wall; light grey Emperors head.
ZPMARBY1.BM	8x8	Floor; dark yellow, mottled.
ZPMAT.BM	8x8	Floor; grating with circular holes.
ZPPANEL1.BM	8x8	Floor; light grey, riveted panel.
ZPPANEL3.BM	8x8	Floor; white, cubist panelling.
ZPPOOLG1.BM	8x8	Floor; dark grey with circular highlight (red in GROMAS).
ZPPOOLG2.BM	8x8	Floor; dark grey with circular highlight and square-curved tiling.
ZPPOOLG4.BM	8x8	Floor; dirty, dark grey with circular highlight.
ZPPOOLG5.BM	8x8	Floor; dirty, dark grey with diamond pattern and circular highlight.
ZPPOOLWT.BM	8x8	Floor; white, circular highlight.
ZPRIVET1.BM	8x8	Floor; mottled grey with riveted plates.
ZPROCK02.BM	8x8	Floor; light grey, rough concrete.
ZPSEWRL1.BM	8x8	Ceiling; grey, corroded cross bars with yellow light.
ZPSEWRL3.BM	8x8	Ceiling; four grey, corroded cross bars with yellow lights.
ZPSEWRL4.BM	8x8	Wall; grey ribbing with exposed tubes and yellow light.

ZPSEWRL6.BM	8x8	Wall; grey ribbing with exposed tubes and red light.
ZPSGRTE1.BM	8x8	Floor; grey, cross-hatched grating with rust.
ZPSHINY1.BM	8x8	Floor; white, mottled.
ZPSIMP06.BM	8x8	Wall; dirty, grey cubist with dark, crossing bands.
ZPSIMP09.BM	8x8	Floor; grey/mauve cubist pattern.
ZPSIMP11.BM	8x8	Floor; light grey panelling with crossed bands.
ZPSIMP12.BM	8x8	Floor; light grey concrete.
ZPSLOT1Y.BM	8x8	Floor; grey with two vertical strips of holes.
ZPSLOT2Y.BM	8x8	Floor; grey with two vertical strips of holes and rust (red in GROMAS).
ZPSLOT3X.BM	8x8	Floor; grey with two light, horizontal strips.
ZPSLOT3Y.BM	8x8	Floor; grey with two light, vertical strips.
ZPSLOT4X.BM	8x8	Floor; grey with two light, horizontal strips and rust.
ZPSLOT6X.BM	8x8	Floor; grey with two horizontal strips of red lights.
ZPSLOT6Y.BM	8x8	Floor; grey with two vertical strips of red lights.
ZPSLOT7Y.BM	8x8	Floor; grey with four vertical strips of holes.
ZPSPATBL.BM	8x8	Floor; mauve and grey, speckled bricks.
ZPSPATGR.BM	8x8	Floor; brown, speckled bricks.
ZPSTONGR.BM	8x8	Floor; four white, square tiles.
ZPTILE3.BM	8x8	Floor; blue grey, mottled tile with double border.
ZPTILE4.BM	8x8	Floor; light grey, mottled tile with grey border.
ZPTUBECX.BM	8x8	Ceiling; grey with horizontal tubes and strips of blue lights.
ZPTUBECY.BM	8x8	Ceiling; grey with vertical tubes and strips of blue lights.
ZPTUBEFX.BM	8x8	Ceiling; grey with horizontal tubes and grey strips.
ZPTUBEFY.BM	8x8	Ceiling; grey with vertical tubes and grey strips.
ZPVEINED.BM	8x8	Floor; light grey with white strip pattern.
ZPWHTLIT.BM	8x8	Floor; white bricks.
ZPYFILL.BM	1/8x1/8	Filler; small, orange square.
ZSBANNLG.BM	8x16	Sign; red, Imperial banner.
ZSBLK-A.BM	4x4	Sign; white I in green circle.
ZSBLK-B.BM	4x4	Sign; white n in red circle.
ZSBLK-C.BM	4x4	Sign; white G in dark blue circle.
ZSBLK-D.BM	4x4	Sign; white A in orange circle.
ZSBLK-E.BM	4x4	Sign; white, dotted A in white circle.
ZSBLK-F.BM	4x4	Sign; white, double-dotted A in light blue circle.
ZSLTSTR1.BM	8x1	Door track (horizontal); grey with green lights.
ZSLTSTR2.BM	8x1	Door track (horizontal); grey with red lights.
ZSPANEL1.BM	4x4	Wall; dark grey grating panel into duct.
ZSREBEL3.BM	8x8	Sign; red Rebel symbol.
ZW4WAY.BM	4x16	Sign; four panels of yellow and red lights indicating any of three sections open or all closed.
ZWARCH1.BM	4x16	Wall; white with embossed arch.
ZWBAND1.BM	4x16	Wall; dark grey, mottled with vertical, silver bands.
ZWBARLT1.BM	8x16	Wall; grey, mottled with horizontal, white light strip.
ZWBARLT2.BM	8x16	Wall; grey, mottled with horizontal, white light strip and floor skirting.
ZWBARPAD.BM	8x16	Wall; orange panelling with rounded corners.
ZWBARS.BM	2x4	Door track; vertical, grey bar.
ZWBRIDGE.BM	16x16	Wall, exterior; light grey support mechanism of bridge, with rust.
ZWCARGP1.BM	8x16	Wall; beige with embossed, trapezoidal panels.
ZWCARGP2.BM	4x16	Wall; dark grey, horizontal planking with soot stains.
ZWCARGP3.BM	8x16	Door; grey bulkhead door.
ZWCARGP4.BM	8x16	Wall; beige with embossed, trapezoidal panels and red light.
ZWCARGP5.BM	8x16	Door; grey bulkhead door with grill.
ZWCARGP6.BM	8x16	Door; grey, circular bulkhead door.
ZWCARGP7.BM	16x16	Wall; beige panelling with exposed machinery.
ZWCARGP8.BM	32x2	Door track (horizontal); grey with blue strip.
ZWCARGP9.BM	16x16	Door; beige, panelled bulkhead door.
ZWCHOMP1.BM	8x8	Door; white with vertical panel (used for body of turbine blade).
ZWCLIFLT.BM	16x16	Wall, exterior; white, stone blocks with rust.
ZWCLIFMD.BM	16x16	Wall, exterior; light grey, stone blocks with rust.
ZWCMENT1.BM	16x8	Wall, exterior; white cement blocks between grey horizontals and rust.
ZWCMENT2.BM	16x8	Wall, exterior; white cement blocks with grey lintel and rust.
ZWCMENT3.BM	16x8	Wall, exterior; white blocks with interlocking, grey border.
ZWCMENT4.BM	16x8	Wall, exterior; white blocks with interlocking, grey border and white centre.
ZWCMENT5.BM	16x8	Wall, exterior; white blocks with interlocking, grey border and yellow light.

ZWCOLUM3.BM	32x16	Wall; orange, mottled and patterned with white, classical columns.
ZWCOMP1.BM	8x8	Wall; white with control panel.
ZWCOUPL1.BM	16x32	Wall; large, vertical, silver tube over beige panelling with yellow hazard stripes.
ZWCOUPL2.BM	16x16	Wall; large, vertical, silver tube over beige panelling with yellow hazard stripes.
ZWCRUNCH.BM	8x16	Wall; silver grating with frilled bottom (used as stamping machine.
ZWCSIDE.BM	8x2	3DO edge; dark grey, triangle-ribbed.
ZWDARK1.BM	8x16	Wall; black with vertical stripes.
ZWDARK2.BM	8x16	Wall; black, Y-shaped panelling with circular indentation.
ZWDARK4.BM	8x16	Wall; dark grey, gothic cavity.
ZWDARK5.BM	8x16	Wall; dark grey, gothic arch.
ZWDKBARS.BM	4x16	Wall; dark grey with two bands of vertical bars.
ZWEXT01.BM	4x16	Wall, exterior; white with dark mottling at base.
ZWEXT01D.BM	16x32	Wall, exterior; white with dark mottling and rust at base.
ZWEXT02.BM	4x16	Wall, exterior; white with dark mottling at base.
ZWEXT03.BM	4x16	Wall, exterior; white with embossed panel.
ZWEXT04.BM	4x16	Wall, exterior; white with dark mottling at base.
ZWEXT05.BM	4x16	Wall, exterior; grey.
ZWEXT06.BM	4x16	Wall, exterior; white.
ZWEXT07.BM	4x16	Wall, exterior; grey with ribbing.
ZWEXT08.BM	8x16	Wall; mauve horizontals with white supports.
ZWEXT09L.BM	8x16	Wall, exterior; grey with two horizontal strips of blue lights.
ZWFACIST.BM	8x16	Wall; grey with vertical light strip and S-shape.
ZWGASTNK.BM	8x8	Crate; white octagon with grey struts.
ZWGRENL1.BM	4x16	Wall; grey glass shape with green light at top.
ZWGRUV1.BM	4x16	Wall; white, stone column.
ZWHALL1L.BM	8x16	Wall; dark grey with vertical, white light strips, grating and skirting.
ZWHALL2L.BM	4x16	Wall; dark grey with vertical, white light strips and skirting.
ZWHALL3.BM	4x16	Wall; dark grey with grating and skirting.
ZWHALL7.BM	4x16	Wall; dark grey with oval features..
ZWHALL7S.BM	4x16	Wall; dark grey with oval features.
ZWHITEC1.BM	16x16	Wall; grey grill with orange panelling and white column.
ZWHITEC2.BM	16x16	Wall; grey grill with blue panelling and grey column.
ZWIBEAM1.BM	8x8	Wall; white struts over grey planking.
ZWILIT01.BM	4x16	Wall; grey panel with rectangular, white light.
ZWILIT03.BM	4x16	Wall; roughed, orange panel with white border.
ZWILIT06.BM	8x16	Wall; white with vertical, white light strip and features.
ZWILIT08.BM	8x16	Wall; white with vertical, white light strip.
ZWILIT12.BM	8x16	Wall; grey, upwards-pointing arrow with white light.
ZWILIT14.BM	8x16	Wall; black grill with horizontal, light grey strips.
ZWILIT18.BM	4x16	Wall; roughed, grey panel with white border and lamp.
ZWILOGO2.BM	32x16	Door; black, mottled with embossed Imperial circle and red stripe.
ZWILOGO6.BM	16x16	Door; white, mottled with red Imperial circle.
ZWIMP15.BM	8x16	Wall; dark grey with vertical, sunk panels and silver strip.
ZWIMP19.BM	8x16	Wall; dirty, grey, mottled.
ZWIMP20.BM	16x16	Wall; dirty, grey with dark grey struts and ribbing.
ZWIMP27.BM	16x16	Wall; black with trapezoidal panels.
ZWIMP28.BM	8x16	Wall; black with trapezoidal panels.
ZWINDSP1.BM	16x16	Wall; dark grey, angled horizontals.
ZWINT01.BM	8x16	Door; white hatch with green lights.
ZWINT02.BM	8x16	Wall; white, horizontal panels.
ZWINT04.BM	4x16	Wall; white with vertical strips.
ZWINT05.BM	4x16	Wall; orange with vertical strips.
ZWINT06.BM	4x16	Wall; white with rounded, vertical strips.
ZWINT07.BM	4x16	Wall; orange with rounded, vertical strips.
ZWINT08.BM	8x16	Wall; white panelling.
ZWINT09.BM	8x16	Wall; orange panelling.
ZWINT10.BM	4x16	Wall; white with rounded, vertical strips.
ZWINT11.BM	4x16	Wall; orange with rounded, vertical strips.
ZWINT12.BM	4x16	Wall; white panelling.
ZWINT13.BM	4x16	Wall; orange panelling.
ZWINT14.BM	4x16	Wall; white, cubist panelling.
ZWINT15.BM	4x16	Wall; orange, cubist panelling.
ZWINT16.BM	4x16	Wall; white panelling with truncated corners.
ZWINT21.BM	16x16	Wall; yellow and brown grill-like strips.

ZWINT22.BM	8x16	Wall; silver with polygonal pattern.
ZWJABEXT.BM	32x16	Wall; grey with dark grey mechanicals and swept back pipe.
ZWJABPI1.BM	8x16	Wall; light grey panelling.
ZWLIT03X.BM	4x4	Wall; black grill with horizontal, white light strip at base.
ZWLOUNG.BM	8x16	Wall; dark grey with horizontal, white light and six red lights.
ZWLYSM.BM	8x16	Wall; black with two bands of bars and horizontal, yellow light strips.
ZWMARBB1.BM	16x16	Wall; black, mottled.
ZWMARBB3.BM	16x16	Wall; black, mottled with gold starburst panel.
ZWMARBG1.BM	16x16	Wall; grey with red Imperial circle and angled ribbing.
ZWMARBG2.BM	16x16	Wall; grey with Emperor and angled ribbing.
ZWMARBW1.BM	16x16	Wall; black panelling with white and blue marble skirtings.
ZWMARBW3.BM	4x16	Wall; black panelling with white and blue marble column and skirtings.
ZWMARBY0.BM	16x16	Wall; dark grey, random weave.
ZWMARBY2.BM	16x16	Wall; dark grey, random weave with dark yellow, mottled skirtings.
ZWMARBY4.BM	8x16	Wall; dark yellow, mottled with two light grey, horizontal bands.
ZWMARBY5.BM	16x16	Wall; dark yellow, mottled with two light grey, horizontal bands and Emperors head.
ZWMARBY6.BM	4x16	Wall; black, mottled with dark yellow, mottled skirtings and white light.
ZWMARBY7.BM	8x16	Wall; black, mottled with dark yellow, mottled skirtings and circle of white light.
ZWMARBY9.BM	8x16	Wall; rough, dark yellow panel with radiating, yellow panels.
ZWMAZE.BM	16x16	Sign; hexagonal, green and red map of computer core.
ZWNAVSW.BM	8x8	Switch; grey space to insert nava card for decoding with white lights (actually 6x6).
ZWOVAL1Y.BM	8x16	Wall; grey panelling with white floor and ceiling lamps.
ZWPANEL4.BM	4x16	Wall, exterior; light grey, concrete panel.
ZWPANLC1.BM	8x16	Wall, exterior; white, protruding panel in light grey concrete.
ZWPANLC3.BM	8x16	Wall, exterior; white and orange, protruding panel in light grey concrete.
ZWPIPES1.BM	8x16	Wall; white panels with grate and pipes.
ZWPIPES2.BM	16x8	Wall; light grey panelling with dark piping at floor and ceiling and rust.
ZWPORTHL.BM	8x16	Wall; black panelling with circular, white light.
ZWSMUGGL.BM	16x64	Wall; grey panelling.
ZWSTONEW.BM	16x16	Wall; light grey, diagonally fitted, stone wall.
ZWSTRIP1.BM	2x16	Door track; light grey with circular lock bars.
ZWSTRIP2.BM	2x16	Door track; red with circular lock bars and rust.
ZWSTRIP3.BM	1x8	Door track; light grey with dark hollows.
ZWSTRIP5.BM	1x8	Door track; light grey with horizontal grate.
ZWSTRIP6.BM	8x2	Door track (horizontal); light grey with large, rectangular lock bars.
ZWSTRIP8.BM	8x1	Door track (horizontal); light grey with vertical grate.
ZWTUBE.BM	4x16	Wall; grey with horizontal tubes and circular, white lights.
ZWTUBEND.BM	16x16	Door; grey wall with tubes and white lights coming in from four directions to darker door.
ZWWHITE0.BM	2x16	Wall; grey grill with orange panelling.

Cutscenes LFD Files

[by Michael Taylor]

Here is what is shown for each resource file.

idresource LFD	scene
10:logo.lfd	Lucas Arts logo
20:swlogo.lfd	Star Wars logo
30:ftextcra.lfd	scrolling text
40:1e.lfd	ship flyby
41:darklogo.lfd	Dark Forces logo and credits
200:kflyby.lfd	flyby from planet
209:execx.lfd	star destroyer
210:execcomp.lfd	Darth Vader and Gen. Mohc talk
211:arcext.lfd	Arc Hammer external

215:tubecomp.lfd	DT loading and lauch
216:succes.lfd	Darth Vader continues
220:neb1.lfd	rebel fleet
225:brief1.lfd	Mon Montha briefing 1
230:holocu.lfd	Admiral's report
235:brief2.lfd	Mon Montha briefing 2
240:exitneb.lfd	Kyle takes off
500:gromas1.lfd	flying to Gromas
550:gromasx.lfd	leaving Gromas
600:arcfly.lfd	Arc Hammer
605:madine1.lfd	General's report on Madine
610:boba.lfd	Boba Fett
800:rob1.lfd	flying to robotics facility
850:robotx.lfd	leaving robotics facility
1000:jabba1.lfd	Kyle takes off
1010:jabba2.lfd	tractor beam gets Kyle
1020:jabba3.lfd	Jabba's ship takes off
1030:pit.lfd	Jabba and Kyle
1050:jabescp.lfd	escaping from Jabba
1400:cargo1.lfd	cargo goes from Executor to Arc Hammer
1410:cargo2.lfd	cargo docks
1450:exp1xx.lfd	Arc Hammer explosion
1451:exp2x.lfd	Kyle flies by
1452:exp3xx.lfd	Darth Vader's comments
1460:award1.lfd	rebel fleet
1470:award2.lfd	show medal
1472:award3.lfd	Kyle leaves hangar
1475:award4.lfd	Kyle flies in and out of fleet
1480:endfly.lfd	Kyle flies away
1500:fullcred.lfd	credits

DFBRIEF.LFD

The following files must always be present in DFBRIEF.LFD:

PLTTbrf-jan	Palette to use for briefings
DELTcursor	Cursor for briefings and PDA
ANIMguns	Weapons screen in PDA
ANIMitems	Items screen in PDA

There must also be one or more briefing backgrounds as necessary:

ANIMbrf-jan	Jan
ANIMbrf-mon	Mon Mothma
ANIMbrf-nil	Jabba

Briefings are stored in DELT sections of dfbrief.lfd, named after the level.
The width of the scrollable region seems to be hardcoded in the game, so the only field we'll want to change is SizeY.

Objective screens are stored in ANIM sections of dfbrief.lfd, also named after the level.
The first DELT in the ANIM has all the goals in green text. The following DELTs have one goal each in yellow text.
They are overlaid on the first DELT when the goal has been completed. See GOL file

See also BRIEFING.LST

JEDISFX.LFD

Sounds in this list may not be used in INF, they are for use in briefings and cutscenes.

SFX are pure sound effects
DLG are pure dialogs
DLX are dialogs including sound effects

In case you wonder about such names as M01KYL01.VOC, here is the decomposition :

M	Mission	
01	Mission Number (Arc Hammer is 16!)	
KYL	Speaker	
	KYL	Kyle
	JAN	Jan Ors
	IMP	Imperial
	JAB	Jabba
	MMA	Mon Mothma
	MOC	General Mohc
	NAR	Narrator
	REB	Rebel
	VDR	Vader;
01	first speech for this mission (A1 is an alternate recording)	

[by Blake Crosby]

VOC Name	**Type**	**Description**
AX-SHING.VOC	SFX	Sound Of An Axe
BEAM-1C.VOC	SFX	Low Pitch Beam Sound
BEAM-2A.VOC	SFX	High Pitch Beam Sound
BEEP-01.VOC	SFX	Default Beep
BEEP-3.VOC	SFX	Warning Horn
BEEP-6.VOC	SFX	Buzzer
BREATH-1.VOC	SFX	Darth Vader Inhaling
BREATH-2.VOC	SFX	Darth Vader Exhaling
BUCKLE1.VOC	SFX	Attaching A Buckle
BUTTON-1.VOC	SFX	Default Button Sound
DFLOCK.VOC	SFX	Distant Sound (Explosion???)
DFLOGO.VOC	SFX	Sound Of Df Logo
DISTRESS.VOC	DLX	Tak Base Distress Call
DIVE1.VOC	SFX	Dark Trooper Sound
DIVE2.VOC	SFX	Dark Trooper Sound
DIVE3.VOC	SFX	Dark Trooper Sound
DOOR-1.VOC	SFX	Door
DS-LP-2.VOC	SFX	Star Destroyer Engines Passing By
DT-DOOR2.VOC	SFX	Big Door
DTLAUNCH.VOC	SFX	Dark Trooper Launcher (Explosion??)
DT-LOWER.VOC	SFX	Mecanical Sound
ELEV1-1.VOC	SFX	Elevator Sound
EN-ZP-4.VOC	SFX	Futuristic Sound
EX-BIG-2.VOC	SFX	Explosion
EXEC.VOC	SFX	Star Destroyer's Engines
EX-FL-2.VOC	SFX	Explosion
EX-GR-2.VOC	SFX	Explosion
EX-GROM1.VOC	SFX	Distant Explosion
GOGGLES1.VOC	SFX	Wistleing Sound
GOGGLES2.VOC	SFX	Wistleing Sound
GUNCOCK.VOC	SFX	Gun Cocking Sound

HEART-1.VOC	SFX	Medical Machine Beep
HOLO1.VOC	SFX	Futuristic Sound
HYD-CL-1.VOC	SFX	Click Sound
HYD-CL-3.VOC	SFX	Deeper Clicking Sound
HYDROL-3.VOC	SFX	Elevator Starting
HYP-IN-8.VOC	SFX	Entering Hyperspace
INTCOM3.VOC	DLX	Dark Trooper Release, Mark 1...
INTCOM4.VOC	DLX	Dark Trooper Test, Mark1...
INTCOM5.VOC	DLX	Medic 1, Medic 2, Medic 3
INTCOM6.VOC	DLX	Launch Test 2, Launch Test 3
INTNARA1.VOC	DLG	Scrolling Text (First Third)
INTNARB1.VOC	DLG	Scrolling Text (Middle Third)
INTNARC1.VOC	DLG	Scrolling Text (Last Third)
JABAHOLO.VOC	SFX	Futuristic Sound
JABGONE.VOC	SFX	Futuristic Sound
JHALO-ON.VOC	SFX	Futuristic Sound
LOGOMIX.VOC	SFX	Lucasarts Logo
M01IMP01.VOC	DLG	Primary Drop Line Engage. Dropline One, Two Nine Release.
M01KYL01.VOC	DLG	This Is Too Easy, Now To Get To My Ship
M01KYL02.VOC	DLG	Interesting, This Sounds Like It Could Be An Imperial Attack... Except For Those Sounds.
M01KYL03.VOC	DLG	A New Stormtrooper That Can Take Out A Base That. Easy! I Should've Kept Working For The Empire.
M01KYL04.VOC	DLG	This Could Be Interesting, All Right I'm In But I Think I'll Need Some Help On This One. I Want Jan Ors As My Mission Officer.
M01MMA01.VOC	DLG	Thank You Commander For Responding On Such Short...
M01MMA02.VOC	DLG	5 Days Ago The Empire Attacked One Of Our Secret...
M01MMA03.VOC	DLG	Tak Base Was Destroyed Within Minutes, Many...
M01MMA04.VOC	DLG	Very Perceptive Commander, I Know You Understand That...
M01MMA05.VOC	DLG	This Imperial Officer, Crix Madine Wishes To Defect...
M01MMA06.VOC	DLG	The Rebel Command Is Not Taking This Lightly, They...
M01MMA07.VOC	DLG	Certainly, Then I Will Let Jan Further Brief You On...
M01MOC01.VOC	DLG	Thank You Lord Vader, What I Will Unveil For You...
M01MOC02.VOC	DLG	With Pleasure
M01MOC03.VOC	DLG	Dark Trooper Release
M01MOC04.VOC	DLG	Certainly Lord Vader
M01MOC0A.VOC	DLG	Thank You Lord Vader, What I Will Unveil Today...
M01MOC0B.VOC	DLG	We Will Be Able To Decimate The Rebels Just As We...
M01NAR01.VOC	DLG	Kyle Delivers The Plans To The Rebel Alliance...
M01REB01.VOC	DLG	This Is Tak Base To Anybody Out There, Please We...
M01REB02.VOC	DLG	Total Devastation, They Broke Through Our Shields...
M01VDR01.VOC	DLG	The Emperor Has Approved Of Your Test Demonstration...
M01VDR02.VOC	DLG	A Noble Cause General, I Hope The Demonstration...
M01VDR03.VOC	DLG	Very Impressive General The Emperor Will Be Most...
M02JAN01.VOC	DLX	Go Ahead Kyle
M02JAN02.VOC	DLX	Get Back To The Landing Pad And I Will Meet You There
M02KYL01.VOC	DLX	Jan ?
M02KYL02.VOC	DLX	Looks Like I Found Somthing That Could Help Us Out
M03JAN01.VOC	DLX	You're The Boss, Kyle
M03KYL01.VOC	DLX	Jan, I Found Moff Rebus; I'm Ready To Get Out Of This Mess
M04JAN01.VOC	DLX	That's All We Need, Lets Get Out Of Here I'm Getting Nervous
M04KYL01.VOC	DLX	I Found Some Interesting Looking Metal I Think This May Offer Us Some Important Clues.
M05JAN01.VOC	DLX	Ok Kyle, Sounds Good To Me
M05KYL01.VOC	DLX	Kyle To Jan, Charge Set Ready To Clear
M05KYL02.VOC	DLX	Jan You Better Get Me Out Of Here I Think I Just Finished Off A Dark Trooper. I Don't Want To Find Out If There Are Any More Around.
M05KYL03.VOC	DLX	If That Thing Down There Is Any Indication Of What We Are Dealing With, We're Going To Need More Fire Power.
M05MOC01.VOC	DLG	This Contemptible Excuse For An Officer Will No Longer...
M05MOC02.VOC	DLG	I Understand The Threat Lord Vader, Katarn Was Once An...
M05VDR01.VOC	DLG	Katarn Will Not Be As Easy To Deal With, He Is Very...

M06JAN01.VOC	DLX	Don't Hang Around Let's Get Out Of Here Before Any More Dark Troopers Arrive.
M06KYL01.VOC	DLX	Ok Jan, I Rescued Madine
M07JAN01.VOC	DLX	Picking Up The Signal, Looks Like We Are Done Here
M07JAN02.VOC	DLX	Ok Kyle, Let's See Where These Smugglers Are Headed
M07KYL01.VOC	DLX	Tracking Device Is Secured
M08KYL01.VOC	DLX	Charge One Set
M08KYL02.VOC	DLX	Charge Two Set
M08KYL03.VOC	DLX	All Charges Set
M08KYL04.VOC	DLX	Woman After My Own Heart
M08KYL05.VOC	DLX	Ah Sh {Static}
M09JAN01.VOC	DLX	Those Must Be Smuggler Routes To The Arc Hammer I Think It's Time To Get Out Of Here.
M09JANA1.VOC	DLX	Those Must Be Smuggler Routes To The Arc Hammer I Think It's Time To Get Out Of Here.
M09KYL01.VOC	DLX	Jan I Found The Imperial Nava Card
M10JAB01.VOC	DLX	Jabba Speaking
M10JAB02.VOC	DLX	Jabba Speaking
M10JAB03.VOC	DLX	Jabba Speaking
M10JAB04.VOC	DLX	Jabba Laughing And Speaking
M10JAB05.VOC	DLX	Jabba Speaking
M10JAB06.VOC	DLX	Jabba Speaking (Mad)
M10JAN01.VOC	DLG	Thanks I Thought I Was Done For
M10KYL01.VOC	DLX	Jabba, What Have You Done With Jan? If Any Harm Comes To Her I'll Personally Shove My Blaster Down Your Slimy Throat.
M10KYL02.VOC	DLX	I Wish You Were Here Too Jabba, There Is Nothing Like Roasted Kell Dragon
M10KYL03.VOC	DLG	No Time For Hugs, Lets Get Out Of Here
M11JAN01.VOC	DLX	Good Job Kyle But You're Not Done Yet
M11JAN02.VOC	DLX	Beautiful Kyle. Now Get That Data Tape And Get Your Mercenary Hide Out Of There. I Can't Stay Out Here Too Long Before Imperial Security..
M11JAN03.VOC	DLX	Kyle Something Strange Is Going Down Over Here! Get Back Here, I Mean It!
M11JAN04.VOC	DLX	Oh No! Kyle You Better Look Out I Just Saw {Static}
M11JAN05.VOC	DLX	Kyle Where Are You? I'm Back At The Landing Pad
M11JAN06.VOC	DLX	I Had Tie Fighters All Over Me, I Had To Properly Dispose Of Them
M11JANA6.VOC	DLX	I Had Tie Fighters All Over Me, I Had To Properly Dispose Of Them
M11KYL01.VOC	DLX	Jan, I Cracked The Central Lock, I'm In
M11KYL02.VOC	DLX	Nava Card Inserted And Decoding
M11KYL03.VOC	DLX	Data Tape Is In Hand I'm On My Way Out
M11KYL04.VOC	DLX	Where Were You Jan?
M12IMP01.VOC	DLG	Smuggler's Ship, Your Flight Path Is Clear Begin Your Docking Procedure
M12JAN01.VOC	DLX	Good Job Kyle
M12JAN02.VOC	DLX	Good Luck Kyle, And May The Force Be With You
M12KYL01.VOC	DLX	Ok Jan, Smuggler's Ship Secured
M12KYL02.VOC	DLX	Now Launching, I'll See You On The Dark Side Jan
M13KYL01.VOC	DLG	Here We Go
M16KYL01.VOC	DLG	That's One
M16KYL02.VOC	DLG	That's Two
M16KYL03.VOC	DLG	One More Left
M16KYL04.VOC	DLG	Jan Would Be Proud
M16KYL05.VOC	DLG	There Is No Glory In War Mohc
M16KYL06.VOC	DLG	For Freedom
M16MOC01.VOC	DLX	It's Been A Long Time Since I Have Challenged A Man In Battle I'm Glad My Opponent Is So Worthy.
M16MOC02.VOC	DLG	You Were An Excellent Adversary, Commander; The Warrior's Flame Burns In...
M16MOC03.VOC	DLG	It Is Unfortunate That You Do Not Appreciate What I Am Building Here...
M16MOC04.VOC	DLG	No Glory? Then Why Do You Engage In This War?
M16MOC05.VOC	DLG	You Delude Yourself Commander, We All Fight For Freedom; To Bad You Will...
M16VDR01.VOC	DLG	This Is An Unfortunate Set Back, The Force Is Strong With Katarn
MEDISCAN.VOC	SFX	Soft Buzzing Sound
MEDSHIP3.VOC	SFX	A Frigate's Engines
MIL-NF-1.VOC	SFX	Swishing Sound
MO8JAN01.VOC	DLX	Good Job, Let's Blow This Ice Cube!
PIGPUSH.VOC	SFX	Burping Sound

RANTRO01.VOC	DLG	There He Is Stop Him!
RANTRO02.VOC	DLG	You There! Stop Where You Are!
RANTRO03.VOC	DLG	Stop! Rebel Scum
RANTRO04.VOC	DLG	You Are Not Authorized In This Area
RANTRO05.VOC	DLG	You Are In Violation Of Imperial Law, Surrender Immediately
RANTRO06.VOC	DLG	Halt!
RANTRO07.VOC	DLG	Set Blasters On Full!
RANTRO08.VOC	DLG	Blast Him!
RANTRO09.VOC	DLG	Release Charges 1, 7, 11 And 9
RANTRO10.VOC	DLG	There Is An Intruder On The Premises, All Forces On Alert!
RANTRO11.VOC	DLG	Condition Red, Intruder Is Onboard
RAY.VOC	SFX	Futuristic Sound
RAY-OFF.VOC	SFX	Futuristic Sound
REMOTE-2.VOC	SFX	Swishing Sound
REV-UP-1.VOC	SFX	Futuristic Sound
REV-UP-2.VOC	SFX	Futuristic Sound
SD-LP-1.VOC	SFX	Star Destroyer Engines
SHIPLOCK.VOC	SFX	Locking Sound
SH-NF-1.VOC	SFX	Crow's Engines Passing By
SLEEVE-2.VOC	SFX	Swishing Sound
SNORT.VOC	SFX	Snorting Sound
STEADY.VOC	SFX	Engine Sound
STRAP1.VOC	SFX	Clicking Sound
TGT-02.VOC	SFX	Soft Beeps
TGT-LP-7.VOC	SFX	Soft Beep
TREPDO-2.VOC	SFX	Swishing Sound
TUBE1.VOC	SFX	Elevator Moving/Doors Opening
XW-NF-1.VOC	SFX	X-Wing Flying By
ZOOM1.VOC	SFX	Futuristic Sound

Resources Cross Reference

The patcher's paradise !
Do you want to know which .voc plays when Boba Fett dies ? Here it is !

[by David Lovejoy]

Weapons
Assault Rifle
Autogun
Bryar Pistol
Concussion Rifle
Plasma Cannon / Missile Launcher
Fusion Cutter
Kyle's Fists
Mines
Mortar
Thermal Detonator

Kyle Katarn
Infra Red Goggles
Ice Cleats
Gas Mask
Actions
Misc. Pickup Stuff
Misc. Pickup Goals

Imperial Enemies
Commando
Officer
Interrogator Droid
Probe Droid
Remote Droid
Mousebot
Phase 1 Dark Trooper

Other Enemies
Boba Fett
Bossk
Gamorrean Guard
Kell Dragon
Reeyees
Dianoga (Sewer bug)

Phase 2 Dark Trooper
Phase 3 Dark Trooper (Mohc)
Turret
Welder

Sprites.gob	iautogun.fme	autogun
"	ipower.fme	ammo for autogun
"	bullet.fme	autogun bullet
"	bullexp.wax	bullet explosion
Textures.gob	autogun1.bm	autogun at rest
"	autogun2.bm	autogun firing
"	autogun3.bm	autogun chamber rotating
Sounds.gob	ex-tiny.voc	gun shot hitting object
"	repeater.voc	rapid fire autogun
"	repeat-1.voc	single shot autogun
"	rep-emp.voc	autogun empty

Sprites.gob	iconcus.fme	concussion rifle
"	ipower.fme	ammo
"	concexp.wax	concussion blue explosion
Textures.gob	concuss1.bm	concussion rifle at rest
"	concuss2.bm	firing
"	concuss3.bm	firing
Sounds.gob	concuss1.voc	concussion rifle empty
"	concuss5.voc	concussion rifle firing
"	concuss6.voc	concussion rifle empty ???
"	ex-lrg1.voc	large explosion sound

Sprites.gob	icannon.fme	plasma cannon
"	iplasma.fme	plasma power cells, dropped by DT
"	imsl.fme	missile pickup
"	imsls.fme	missiles pickup
"	wplasma.wax	plasma cannon blue shot
"	wmsl.wax	missile flying
"	missexp.wax	missile explosion
"	plasexp.wax	plasma explosion
Textures.gob	assault1.bm	plasma cannon at rest
"	assault2.bm	plasma cannon firing
"	assault3.bm	missile launcher at rest
"	assault4.bm	missile launcher firing
Sounds.gob	missile1.voc	missile firing
"	plas-emp.voc	plasma cannon empty
"	plasma4.voc	plasma cannon firing
"	ex-med1.voc	plasma/missile explosion
"	bigrefl1.voc	hit by DT plasma cannon /missile when supershield or laimlame on

Sprites.gob	ifusion.fme	fusion cutter
"	ipower.fme	ammo for fusion
"	weimiss.wax	fusion ball shot
"	emisexp.wax	fusion shot explosion
Textures.gob	fusion1.bm	fusion cutter at rest
"	fusion2.bm	barrel #1 firing
"	fusion3.bm	barrel #2 firing
"	fusion4.bm	barrel #3 firing
"	fusion5.bm	barrel #4 firing
"	fusion6.bm	all barrels firing
Sounds.gob	ex-tiny1.voc	fusion shot explosion sound
"	fusion1.voc	fusion cutter shot
"	fusion2.voc	fusion cutter empty

Textures.gob	rhand1.bm	right hand
"	punch1.bm	left hand
"	punch2.bm	left hand extending fully
"	punch3.bm	left hand partially extended
Sounds.gob	punch.voc	Kyle's fist hitting something
"	swing.voc	fist swinging in empty air

Sprites.gob	landmine.fme	looks like a candle mine
"	imine.fme	one upright mine pickup
"	imines.fme	mine pack pickup
"	wmine.fme	mine on floor with light
"	wlmine.fme	mine on floor no light
"	mineexp.wax	large white mine explosion
Textures.gob	clay1.bm	mine in hand no light
"	clay2.bm	mine in hand with light
Sounds.gob	beep-10.voc	beeps before exploding
"	ex-lrg1.voc	large explosion sound
"	claymor1.voc	laying mine sound

For exploding mines use wlmine.fme, wmine.fme, landmine.fme set logic to Land_mine.
Landmine.fme will appear in level as wmine.fme if set to Land_mine logic.

Sprites.gob	imortar.fme	mortar gun
"	ishell.fme	1 mortar shell pickup
"	ishells.fme	mortar shells pickup
"	wshell.wax	flying mortar shell
"	mortexp.wax	mortar explosion
Textures.gob	mortar1.bm	mortar gun at rest
"	mortar2.bm	mortar gun firing
"	mortar3.bm	mortar gun firing
"	mortar4.bm	mortar gun chamber rotating
Sounds.gob	ex-med1.voc	mortar explosion sound
"	mortar4.voc	mortar gun firing

"	mortar2.voc	mortar gun empty
"	mortar9.voc	mortar gun chamber rotating sound

Dark.gob	wrbolt.3do	red laser shot
Sprites.gob	exptiny.wax	laser explosion
"	ienergy.fme	ammo for pistol
Textures.gob	pistol1.bm	pistol at rest
"	pistol2.bm	pistol firing
"	pistol3.bm	pistol going to rest position
Sounds.gob	ex-tiny1.voc	laser explosion sound
"	lasrby.voc	missed laser shot
"	pistol-1.voc	pistol shot sound
"	pistout1.voc	pistol empty
"	boltref1.voc	laser shot hitting Kyle
		only when laimlame or supershield is on

Dark.gob	wrbolt.3do	red laser shot
Sprites.gob	ist-guni.fme	horizontal laser rifle pickup
"	ist-gunu.fme	vetical laser rifle pickup
"	ienergy.fme	ammo for laser rifle
"	exptiny.wax	laser shot explosion
Textures.gob	rifle-1.bm	laser rifle at rest
"	rifle-2.bm	laser rfile firing
Sounds.gob	ex-tiny1.voc	laser explosoin sound
"	laserby.voc	missed shot
"	rifle-1.voc	rifle single shot
"	riflout.voc	rifle empty
"	boltref1.voc	laser hits Kyle
		only when laimlame or supershield on

Sprites.gob	idet.fme	1 thermal detonator pickup
"	idets.fme	thermal detonators pickup
"	wdet.fme	thermal detonator for throwing
"	detexp.wax	thermal detonator explosion
Textures.gob	therm1.bm	thermal detonator in hand at rest
"	therm2.bm	thermal detonator in right hand
"	therm3.bm	empty right hand
Sounds.gob	ex-small.voc	thermal detonator explsion sound
"	thermal1.voc	thermal detonator bounce

Dark.gob	wrbolt.3do	red laser shot
Sprites.gob	ist-guni.fme	laser rifle horizontal position
"	exptiny.wax	laser explosion
"	commando.wax	commando
Sounds.gob	ransto01.voc	There he is stop him
"	ransto02.voc	You there stop where you are

"	ransto03.voc	Stop rebel scum
"	ransto04.voc	You're not authorized in this area
"	ransto05.voc	Surrender immediately
"	ransto06.voc	Halt
"	ransto07.voc	Set blasters on full
"	ransto08.voc	Blast him
"	st-hrt-1.voc	commando hurt
"	st-die-1.voc	commando die
"	ex-tiny1.voc	gun shot hitting wall
"	boltref1.voc	shot that hits Kyle from laser type weapon
		only when supershield or laimlame used
"	lasrby.voc	laser shot miss
"	rifle-1.voc	rifle single shot

Files ransto01 - 08.voc are used in order each time a stormtrooper notices you.
Used only once in each sector but may also be used from an adjoining sector.

Dark.gob	wrbolt.3do	red laser shot
Sprites.gob	officin.wax	officer
"	exptiny.wax	shot explosion
"	ienergy.fme	ammo dropped by officin logic
"	ikeyr.fme	red key dropped by logic officinr
"	ikeyb.fme	blue key dropped by logic officnb
"	ikeyy.fme	yellow key dropped by logic officiny
"	det_code.fme	blank det_code logic officin1-9
Sounds.gob	ranofc02.voc	Stop where you are
"	ranofc04.voc	You're not authorized in this area
"	ranofc05.voc	You're in violation of imperial law
"	ranofc06.voc	Halt
"	st-hrt-1.voc	officer hurt
"	st-die-1.voc	officer dying
"	ex-tiny1.voc	laser shot explosion sound
"	lasrby.voc	laser shot miss
"	boltref1.voc	shot hitting Kyle
		only when laimlame or supershield is on

logics 1-5 are normaly used in game, Dfbrief1.lfd must be modified to include detcodes 6-9
TEXT.MSG already has coding added for detcodes 6-9

The ranofc02-6.voc files are used in order when ever an officer sees you, usually only once per sector, but will be used when seen from another adjoining sector

Sprites.gob	ipower.fme	Ammo dropped by droid
"	widball.wax	green ball shot
"	emisexp.wax	droid shot explosion
"	intdroid.wax	Interogator droid
Sounds.gob	intalert.voc	droid ummmwwaa sound
"	instun.voc	droid stunning at close range sound
"	probfir1.voc	droid firing
"	ex-small.voc	int droid exploding
"	ex-tiny1.voc	shot exploding
"	bigrefl1.voc	used when kyle hit by droid shot
		only when supershield or laimlame on
"	emisby.voc	droid shot missing kyle

Dark.gob	wrbolt.3do	red laser shot
Sprites.gob	probe.wax	probe droid
"	ipower.fme	ammo dropped by probe droid
"	genexp.wax	probe droid exploding
"	exptiny.wax	probe droid laser shot explosion
Sounds.gob	probe-1.voc	enemy escape advance used when probe sees you
"	probfir1.voc	probe droid firing
"	probalm.voc	probe about to explode
"	ex-tiny1.voc	laser shot explosion sound
"	lasrby.voc	laser shot missed Kyle
"	ex-med1.voc	probe exploding sound
"	boltref1.voc	laser shot hitting Kyle
		used when laimlame or supershield on

Dark.gob	wgbolt.3do	green laser shot
Sprites.gob	remote.wax	remote droid
"	exptiny.wax	laser explosion
Sounds.gob	probfir.voc	remote shooting
"	remote-2.voc	remote psshhttt
"	ex-tiny1.voc	laser shot explosion sound
"	lasrby.voc	Laser shot misses kyle sound
"	boltref1.voc	laser hits kyle
		only when laimlame or supershield on

Dark.gob	mousebot.3do	mouse bot
Sprites.gob	ibattery.fme	battery
"	dedmouse.fme	deadmouse
Sounds.voc	eeek-1.voc	mouse squack
"	eeek-2.voc	mouse hit/hurt
"	eeek-3.voc	mouse dying

Sprites.gob	phase1.wax	phase1 dark trooper
Sounds.voc	phase1a.voc	neaahh used when sighted by phase1
"	phase1b.voc	aaagh used when phase1 hurt
"	phase1c.voc	phase1 dying
"	sword-1.voc	sword sound

Sprites.gob	phase2.wax	phase2 dark trooper
"	iplasma.fme	plasma power cells pickup
"	imsls.fme	missiles pickup
"	wmsl.wax	flying missile
"	wplasma.wax	blue plasma shot flying

"	missexp.wax	missile explosion
"	plasexp.wax	plasma explosion
Sounds.gob	phase2a.voc	phase 2 ahhgggiioooklok used when sighting kyle
"	phase2b.voc	phase 2 phutt die used when hit
"	phase3c.voc	phase 2 dying
"	rocket-1.voc	phase 2 flying (jetpack)
"	plasma4.voc	plasma cannon firing
"	missile1.voc	missile firing
"	ex-med1.voc	plasma/missile explosion
"	emisby.voc	missed shot
"	bigrefl1.voc	used when kyle hit by plasma or missile only when laimlame or supershield is on

Sprites.gob	phase3x.wax	phase3 dark trooper (mohc)
"	wdt3msl.wax	phase3 yellow balls fyling
"	plasexp.wax.	plasma explosion
"	missexp.wax.	tracker balls explosion
"	wplasma.wax	blue plasma shot flying
Sounds.gob	missile1.voc	tracker balls launch sound
"	plasma4.voc	plasma cannon firing
"	tracker.voc	mechanical noise when tracker balls about to be launched
"	rocket-1.voc	phase 3 flying (jetpack)
"	emisby.voc	missed plasma shot
"	ex-med1.voc	plasma and tracker ball explosion
"	phase3a.voc	mohc laugh used in arc.inf, sector mohc_laugh also used when first sighting Kyle
"	phase3b.voc	phase 3 hurt mrp oghh
"	phase3c.voc	phase 3 dying ooooooggghhhh
"	bigrefl1.voc	used when Kyle hit by plasma or missile only when laimlame or supershield is on
"	m16moc01.voc	"It's been a long time since I challenged a man in battle" used by arc.inf sector: voclev

Dark.gob	gun.3do	turret gun
"	base.3do	turret base
"	wgbolt.3do	green laser
Sprites.gob	genexp.wax	turret explosion
"	exptiny.wax	laser shot explosion
Sounds.gob	turrent-1.voc	turret firing
"	ex-med1.voc	turret exploding sound
"	lasrby.voc	laser shot miss
"	ex-tiny1.voc	laser shot explosion

Dark.gob	weldarm.3do	welder arm
"	weldbase.3do	welder base
Sprites.gob	genexp.wax	welder arm exploding
Sounds.gob	ex-med1.voc	welder arm exploding sound
"	weld-1.voc	welder arm moving long distance
"	weld-2.voc	welder arm moving short distance

"	weldsht1.voc	welder arm hitting Kyle
"	weldhrt.voc	welder arm hurt
"	weld-die.voc	welder arm dying

Sprites.gob	bobaball.wax	Boba Fett shots yellow balls
"	genexp.wax	Boba Fett ball explosion
"	bobafett.wax	Boba Fett
Sounds.gob	fireball.voc	Boba Fett ball missed shot
"	ex-med1.voc	Boba Fett ball explosion sound
"	boba-1.voc	Boba Fett laughing
"	boba-2.voc	Boba Fett firing weapon
"	boba-3.voc	Boba Fett hit
"	boba-4.voc	Boba Fett dying
"	rocket-2.voc	Boba Fett flying

Sprites.gob	iconcus.fme	concussion rifle
"	bossk.wax	bossk
"	concexp.wax	concussion rifle explosion
"	bossk-1.voc	hisstth when bossk sees you
"	bossk-3.voc	bossk hurt
Sounds.gob	bosskdie.voc	bossk dying
"	ex-lrg1.voc	large concussion explosion sound
"	concuss5.voc	concussion rifle firing
"	bossk-2.voc	not used

Sprites.gob	gamguard.wax	Gamorrean guard
Sounds.gob	gamor-3.voc	guard sighting you grunt
"	gamor-2.voc	guard squeal when hurt
"	gamor-1.voc	guard dying
"	axe-1.voc	Axe sound

Sprites.gob	kell.wax	Kell dragon
Sounds.gob	kelljump.voc	Kell jumping
"	kell-1.voc	Roar used when Kyle first spotted
"	kell-2.voc	not used Kell dragon
"	kell-5.voc	used when biting Kyle
"	kell-7.voc	Kell dying
"	kell-8.voc	Kell hit or hurt

Sprites.gob	reeyees.wax	reeyees
"	idets.fme	thermal detonators pickup
"	wdet.fme	thermal detonator thrown

"	detexp.wax	thermal detonator explosion
Sounds.voc	reeyee-1.voc	"hey hold up who's there"
"	reeyee-2.voc	yooooggh reeyees hurt
"	reeyee-3.voc	youuuuggghh reeyees dying
"	ex-small.voc	thermal detonator explosion
"	reeyee1.voc	not used
"	reeyee2.voc	not used
"	reeyee3.voc	not used
"	reeyee4.voc	not used
"	thermal1.voc	thermal detonator bounce sound when thermal detonator hits ceiling or floor

Sprites.gob	sewerbug.wax	Dianoga
Sounds.gob	creatur1.voc	Dianoga low growl when it sees Kyle
"	creatur2.voc	Dianoga attacking
"	creathrt.voc	Dianoga hurt
"	creatdie.voc	Dianoga dying

Sprites.gob	igoggles.fme	ir goggles
"	ibattery.fme	battery
Sounds.gob	goggles1.voc	goggles on
"	goggles2.voc	goggles battery run down

Sprites.gob	icleats.fme	ice cleats
Sounds.gob	cleat.voc	walking with cleats on not used
"	snow.voc	walking in snow not used

Sprites.gob	imask.fme	gas mask
Texures,gob	gmask.bm	gas mask on face
Sounds.gob	mask1.voc	breathe in sound
"	mask2.voc	breathe out sound
"	choke.voc	choking in gas

Sprites.gob	splash.wax	splash in liquid surface
Sounds.gob	health1.voc	used when health points go down
"	shield1.voc	used when shield points go down
"	crush.voc	getting crushed
"	fall.voc	falling yaaaaaahhhhh
"	kyledie1.voc	Kyle dying
"	jump-1.voc	jumping
"	land-1.voc	not used
"	splash1.voc	not used splash

"	swimin.voc	splash
"	weapon1.voc	weapon pickup sound
"	scrshot.voc	screen shot (with dark -shots)
"	key.voc	key pickup sound
"	comlete.voc	completion sound
"	bonus.voc	bonus pickup sound

Sprites.gob	iinvinc.wax	invincible
"	icharge.fme	weapon super charge
"	irevive.wax	revive
"	ilife.wax	extra life
"	imedkit.fme	medkit
"	ipile.fme	Kyle's kit used in jabbship level
"	iarmor.wax	shield power up
"	ikeyr.fme	red key
"	ikeyb.fme	blue key
"	ikeyy.fme	yellow key
"	det_code.fme	blank det code
Sounds.gob	quarter.voc	used when invincible and supercharge running out

Sprites.gob	idplans.wax	DeathStar plans
"	iphrik.wax	Phrik metal
"	inava.wax	Nava card
"	idtgun.wax	default wax
"	idtgun.fme	broken dark trooper weapon
"	phrik.fme	Phrik metal
"	idata.fme	Data card
"	jan.fme	Jan Ors
"	mofrebus.fme	Moff Rebus
"	crix.wax	Crix Madine

GOBS are quite similar in principle to DOOM WAD files, but at a higher level.
In fact, WADS directly contain the information or resources, but GOBS also contain complex files, themselves still containing multiple resources.

dark -umygob.gob is nearly equivalent to doom -file mywad.wad
You cannot however load multiple gobs in DF as you can load multiple wads in DOOM.

Here is the correspondence between DARK FORCES and DOOM level components:

DF FILE	**USE**	**DOOM EQUIVALENT**
name.LEV	geometry (static)	SECTORS, LINEDEFS/SIDEDEFS, VERTEXES
name.INF	workings (dynamic)	none, except the TAG concept
name.GOL	goals	none
name.O	objects	THINGS
name.PAL	palette	PLAYPAL 0 (not the 'hit' palettes)
name.CMP	palette mappings	COLORMAP

:
The GOB structure is quite similar to that of a WAD file, the small difference is that in wads the MASTERN field is at the beginning of the file, between WAD_MAGIC and MASTERX.
Of course, the WAD_MAGIC is 'PWAD' or 'IWAD', not 'GOB' followed by 0x0A.
This distinction between master (IWAD) and patches (PWAD) doesn't exist in DF.

Layers are a whole new concept for DOOM levels designers.
It is possible in DF to have many sectors one above the others.
This is completely impossible in DOOM.

The DF sectors are self-contained, by opposition to DOOM vertex and linedef sharing.

In Doom, the equivalent to an Adjoin/Mirror is the sharing of two vertices and a linedef between two sectors.
Contrary to a DF Wall, the linedef doesn't contain texturing information, this one being coded at the sidedef level.

There is no node building (BSP) to do on these levels.
There certainly are checks at level loading time, but a few tests on complicated sectors seem to show 'errors' in texturing or HOM problems.
In fact, it seems that big non-convex sectors are problematic.
Maybe you just have to try and create a few 'more convex' sectors instead.

This is exactly the same as DOOM texturing, just note that there are two different walls, not one linedef with two sidedefs.

Unlike in DOOM, objects and logics are completely separate things in DF. The object will only be a visual thing -- it is the logic given to it that determines how it behaves. An object can be given any suitable logic, so the same viewable object could behave in different ways.

A stop is a value that an elevator can arrive at. This value varies depending on the class of elevator, and can be floor altitude, ceiling altitude, ambience, degrees etc. Stops can be used practically, such as different heights a lift stops at, or can be used purely for level control as elevators can also send a message, page a sound, or create an adjoin upon arriving at a stop.

Note: Elevators can have any number of stops. If no stops are given, the elevator will start at value 0 and keep increasing its value throughout the entire level. This may be appropriate for an "elevator scroll_floor", but not for an "elevator move_floor" !!!

Note: Door elevators should NOT be given stops. They will have automatic stops set depending on the altitudes of the floor and ceiling of their sector.

Usage:
```
| stop: [value1] [value2]
```

The first value can be given in three ways:

`| [num]`	absolute stop
`| @[num]`	relative stop

`| [sectorname]`	equal the value of the sector [sectorname]

The second value can be given in 4 ways:

`| [time]`	time in sec that elevator remains at stop
`| hold`	elevator will remain at stop indefinitely
`| terminate`	elevator will stay at the stop permanently
`| complete`	mission will be complete when elev arrives at stop

A slave of an elevator will follow whatever the elevator does does. However, if relative stops are used, the slave may not necessarily have the exact same actions. For example, a sector with "elevator move_floor" may have a floor altitude of 0 and a slave of it may have a floor alt of 4. When the elevator moves to "stop: @5" the slave will move to altitude 9.

Usage:
```
| slave: [slave sectorname]
```

Changes the AMBIENT of a sector, i.e. changes the light
level in a sector.

Stop values are sector ambience.

Changes FLOOR ALTITUDE of a sector.

Stop values are the altitude of the floor.

Changes the CEILING ALTITUDE of a sector.
Often used for making doors (as you can set Smart Object Reactions).

Stop values are the altitude of the ceiling.

Changes the FLOOR ALTITUDE of a sector.
The difference from "elevator basic" is that the smart object flag does not affect this elevator.

Stop values are the altitude of the floor.

Changes the CEILING ALTITUDE of a sector.
The difference from "elevator inv" is that the smart object flag does not affect this elevator.

Stop values are the altitude of the ceiling.

Changes both the FLOOR ALTITUDE and CEILING ALTITUDE of a sector, i.e. the floor and ceiling will move up and down together.

Stop values are the altitude of the floor.

Scrolls the floor texture of a sector. Player moves with the floor texture by default, but see the FLAGS variable.

Stop values are distances in pixels (x by 8 to get distances in level geometry units).

Scrolls the ceiling texture of a sector.

Stop values are distances in pixels (x by 8 to get distances in level geometry units).

Changes the SECOND ALTITUDE of a sector.

Stop values are second altitude.

Changes the FLOOR ALTITUDE of a sector, but returns to altitude 0 after cycling through all its stops. From there, its event needs to be triggered twice to move it to its first stop again. Otherwise, seems to be the same as elevator_basic.

Stop values are floor altitude.

Changes the LIGHT of any walls in the sector with flag 1 bit 8 (allow change wall light), i.e. the relative light level of a wall to the sector will change.

Note: this elevator won't work if the sector's AMBIENT is 31, in the same way that the LIGHT field of walls won't work in the same case.

Stops values are wall light.

Translates the VERTEX positions of any walls in the sector with flag 1 bit 32 (wall morph with sector). The entire wall will translate on the X-Z plane.

If the walls are adjoined, their mirrors also need to move so should also be set with flag 1 bit 32.

The PLAYER will by default not move with the walls. (but see the FLAGS variable).

Stop values are distances on the X-Z (horizontal) plane relative to the starting location of the wall.

Same as elevator morph_move1 except the PLAYER will by default move relative to the walls if it is in the sector (but see the FLAGS variable).

Rotates the VERTEX positions of any walls in the sector with flag 1 bit 32 (wall morph with sector). The entire wall will rotate on the X-Z plane.

If the walls are adjoined, their mirrors also need to move so should also be set with flag 1 bit 32.

The PLAYER will by default not spin with the walls (but see the FLAGS variable).

Stop values are angles in degrees.

Same as elevator morph_spin1 except the PLAYER will spin relative to the walls if it is in the sector (but see the FLAGS variable).

This is the same as elevator morph_move1 except that it has a default event_mask of 0.

This is the same as elevator morph_spin1 except that it has a default event_mask of 0.

Scrolls texture(s) of any walls in the sector with flag 1 bit 64/128/256/512 (allow scroll mid/top/bot/sign texture). Stop values are distances in pixels (x by 8 to get distances in level geometry units)

Instant door. Note, that it is easier to just use flag 1 bit 2 on a sector for an instant door. Elevator door is only really needed if you want to alter variables, for example, create a key door.

Stops and event_mask are set automatically - just make sure that the ceiling altitude of the sector is when the door is OPEN.

Instant 2 part door (opens upwards AND downwards). Information for the top and bottom parts are specified individually (so if you want to set a key, you have to set it to both halves of the door).

i.e.

```
| class: elevator door_mid
|  addon: 0
|  [info for the top part]
|  addon: 1
|  [info for the bottom part]
```

Stops and event_mask are set automatically - just make sure the floor and ceiling altitudes of the sector are of the door when it is OPEN.

A door that opens downwards. Otherwise, the same as any other door elevator.

Stops and event_mask are set automatically - just make sure that the floor altitude of the sector is when the door is OPEN.

Used with triggers, client defines which sector(s) a message is sent to when the trigger is triggered.

Usage:
```
| client: [client sectorname]
```

This can be applied to a sector (entering, leaving, nudging it etc.) or a line (crossing, nudging it etc.). Can't be used with switches, or you get a single vertical line where the sign should be.

Exactly the same as trigger standard as far as I know (maybe because trigger standard is the default trigger? So if there's a default trigger and a default message than what's the default elevator???)

This is used specifically for switches. Remember, the wall containing the switch must have a sign which is a switch texture. When the switch is pressed, the first texture will change to the second texture in the multiple bm. The second texture can't be pressed - a message: done must be sent to change the switch back to the first texture. This can be done as many times as you like.

This is a trigger that is used with switches. The switch can only be pressed ONCE. Once the switch is on its second texture, it will remain there even if a "message: done" is sent.

This is a trigger that is used with switches. The switch can be pressed while showing either texture, so there's no need for "message: done".

Determines whether an elevator or trigger is able to function.

Usage:
```
| master: on|off
```

Default:
```
| master: on
```

Determines what event will operate an elevator or trigger. The event, when carried out, will move an elevator to its next stop, or trigger a trigger.

event_mask values

1	Cross line from front side
2	Cross line from back side
4	Enter sector
8	Leave sector
16	Nudge line from front side / Nudge sector from inside
32	Nudge line from back side / Nudge sector from outside
64	Explosion
256	Shoot or punch line (see entity_mask)
512	Land on floor of sector

(The above are bit values, so are added up when more than one are needed.)

* or -1	All bits set
Custom values	Triggered by triggers with an event: or by certain messages with the custom value as a parameter.

Usage:
```
| event_mask: [value]
```

See Defaults

(elevs basic, inv, basic_auto)
```
| event_mask: 52
```

(elevs morph_move1, morph_move2, morph_spin1, morph_spin2)
```
| event_mask: 60
```

(other elevators)
```
| event_mask: 0
```

(all triggers)
```
| event_mask: *
```

Creates a custom event value for a trigger. The trigger will then only affect an elevator class with this event value set in its event_mask. The custom value should be a bit value (i.e. a power of 2) so that it can be added with the other custom and preset bits (this works fine). LEC always uses 65536 onwards so I suggest you do this too. However, it seems that you do not HAVE to use a bit value because in EXECUTOR.INF LEC uses 2621444 (note the extra 4) and it works OK. But I don't suggest you do this.

Event: is needed with multi-class elevators or triggers where each class is controlled by a separate trigger. For example in the Research Facility (level 4), the sector called "Corecat" (spins around the Phrik metal) is two classes of elevator - elevator move_fc and elevator morph_spin2. Two switches control these classes individually. If the "event" variable was not used, both switches would move both classes of elevator to its next stop at the same time.

Utilising the event variable, it is made possible to have one switch control the spinning and the other control the moving floor/ceiling.

Usage:
```
| event: [value]
```

Defines the entity that triggers a trigger

entity_mask values

1	Enemy
8	Weapon
2147483648	Player

(The above are bit values, so are added up when more than one are needed.)

* or -1	All bits set

Note: Enemies and weapons (laser bolts, rockets etc.) can enter and leave sectors and cross lines just like the PLAYER, but can't nudge or land on the floor. i.e. you can use entity_mask values 1 and 8 with event_mask values 1, 2, 4 and 8 but NOT with 16, 32 and 512.

Usage:
```
| entity_mask: [value]
```

Default:
```
| entity_mask: 2147483648
```

Determines the speed that an elevator moves between stops. If speed: 0 is set the elevator will move between stops instantaneously.

Usage:
```
| speed: [value]
```

Default:
Different for each type of elevator.

Determines which stop an elevator starts at, right at the start of a level after it has loaded up.

Usage:
```
| start: [stopnum]
```

Default:
```
| start: 0
```

Used with rotating elevators, center defines the coordinates of the center of revolution.

Usage:
```
| center: [x coord] [z coord]
```

Default:
```
| center: 0 0
```

Used with texture-scrolling or horizontally moving elevators, angle defines the direction in which the texture will scroll or the sector will move. For scrolling walls, angle: 0 is down. For scrolling floors, scrolling ceilings and moving sectors, angle: 0 is north.

Usage:
```
| angle: [value in degrees]
```

Default:
```
| angle: 0
```

Defines which key is needed to manually trigger the event of an elevator. Key is optional, of course.

Usage:
```
| key: red|blue|yellow
```

Determines whether or not the player moves with a morphing or a horizontally scrolling elevator.

flag values

1	Move on floor
2	Move on 2nd altitude

These are bit values, so can be added (3) for moving on both the floor AND 2nd alt.

Note: In FUELSTAT.INF (I think) you may find "flags: 7". This suggests that there is a value for 4 as well.

Note: In some places in the original levels, flags is set on vertically moving elevators like move_floor and move_fc. I'm not sure whether this is a mistake, or if flags do something different here.

Usage:
```
| flags: [value]
```

See Defaults

(elevs scroll_floor, morph_move2, morph_spin2)
```
| flags: 3
```

(elevs scroll_ceiling, morph_move1, morph_spin1, move_wall, rotate_wall)
```
| flags: 0
```

Note: all slaves will have flags set to 0.

Sets the sound effects of the elevator or trigger. Elevators have 3 sound effects - leaving a stop (1), moving between stops (2), and arriving at a stop (3). Triggers only have one sound - when triggered.

Usage (elevators):

```
| sound: [sound value (1, 2 or 3)] [VOC file]
```

Usage (triggers):
```
| sound: [VOC file]
```

Note: Setting "0" in place of [VOC file] makes the sound effect silent.

See Defaults

(elevs change_light, change_wall_light, scroll_floor, scroll_ceiling, and scroll_wall)
```
| sound: 1 0
| sound: 2 0
| sound: 3 0
```

(elevs move_floor, move_fc, basic, basic_auto, change_offset, door_inv and bottom half of door_mid)
```
| sound: 1 elev2-1.voc
| sound: 2 elev2-2.voc
| sound: 3 elev2-3.voc
```

(elevs move_ceiling, inv, morph_move1, morph_move2, morph_spin1, morph_spin2, move_wall, rotate_wall and top half of door_mid)
```
| sound: 1 door2-1.voc
| sound: 2 door2-2.voc
| sound: 3 door2-3.voc
```

(elevator door)
```
| sound: 1 door.voc
| sound: 2 0
| sound: 3 0
```

(trigger standard)
```
| sound: 0
```

(triggers switch1, single and toggle)
```
| sound: switch3.voc
```

This seems to work like event_mask when used with an elevator, and like entity_mask when used with a trigger.

Triggers the event of an elevator or trigger (no matter what its event_mask value is).
An elevator will be moved to its next stop, and a trigger will be triggered.

Sent from an elevator.

Sent to a line trigger or sector trigger or an elevator.

Parameters:

`[event value]`	--	optional -- the message will only be sent to the class with this event value set in its event_mask

Sends an elevator to a specified stop.

Sent from a trigger or an elevator.

Sent to an elevator.

Parameters:

`[num]` --	Stop number to send elevator to.

Sends an elevator to its next stop.

Sent from an elevator or trigger.

Sent to an elevator.

Parameters:

`[event value]`	optional -- the message will only be sent to the class with this event value set in its event_mask

Sends an elevator to its previous stop.

Sent from an elevator or trigger.

Sent to an elevator.

Parameters:

`[event value]`	optional -- the message will only be sent to the class with this event value set in its event_mask

Turns an elevator's or trigger's master on.
This message also turns on all generators in the recipient sector with "master: off" set in the .O file.

Sent from an elevator or trigger.

Sent to an elevator or trigger (or a normal sector with generators in it).

Parameters:

`[event value]`	optional -- the message will only be sent to the class with this event value set in its event_mask

Turns an elevator's or trigger's master off.

Sent from an elevator or trigger.

Sent to an elevator or trigger.

Parameters:

`[event value]`	optional -- the message will only be sent to the class with this event value set in its event_mask

Sets specified flag bits to a sector or wall. To set more than one bit, add up the bit values that you want to be set.

Sent from a trigger or elevator.

Sent to a sector or wall.

Parameters:

`[flagnum]` --	flag number (1, 2 or 3)
`[bitnum]` --	bit value to set

Clears specified flag bits from a wall or sector. To clear more than one bit, add the bit values up that you want cleared.

Sent from a trigger or elevator.

Sent to a sector or wall.

Parameters:

`[flagnum]` --	flag number (1, 2 or 3)
`[bitnum]` --	bit value to clear

Tells the GOL file that a trigger goal has been completed, updating the objective screen. Also moves recipient elevator to its next stop.

Sent from a trigger or an elevator.

Sent to an elevator (preferably to your "complete" elevator as it will also be moved one stop closer to its complete stop).

Parameters:

`[num]` --	refers to the "TRIG: [num]" in the GOL file. The corresponding goal in your PDA will then be shown to be complete (if your ANIM is done correctly, that is!!!)

Puts a switch on its first texture - it can be pressed again (UNLESS it is a trigger single).

Sent from an elevator.

Sent to a line trigger.

Parameters: none

VUEs with "PAUSE: TRUE" will be played through once when this message is sent to the sector containing the 3DO object.

Sent from an elevator.

Sent to a sector.

Parameters: none

Toggles the ambience of ALL sectors in the level between their original setting and the value of sector flag 3. Using sector flag 3 bits 1, 2, 4, 8, and 16 it is possible to make any ambient level from 0 to 31.

Sent from an elevator or trigger.

Sent to the SYSTEM (treat it like a sector with name "system", but make sure there are NO actual sectors called "system" anywhere in your level!

Parameters: none

Plays a sound effect when an elevator arrives at a stop. "Page:" is placed in an elevator's sequence.

Usage:
```
| page: [stopnum] [VOC file]
```

Displays a text message from TEXT.MSG when a trigger is triggered. "Text:" is placed in a trigger's sequence.

Usage:
```
| text: [text number in text.msg]
```

Adjoins a line to another line when an elevator arrives at a stop, removing any adjoins it had with a previous line. This is required if you need a line to remove its adjoin with one line and adjoin with another line midway through a level. "Adjoin" is placed in an elevator's sequence.

For example, in level 6 (detention center), you may notice that the 2 main lifts have a door on each layer adjoined to it on the same line. Since a line cannot be adjoined to more than one other line at once, the following occurs: midway through moving up between 2 layers, the elevator move_floor arrives at a stop which it remains at for 0 seconds. At this stop, a line of the lift sector is adjoined to a line of the door sector on the layer above, at the same time removing its adjoin with a line of the door sector on the layer below. The lift's doors all appear to be directly on top of each other.

Usage:
```
| adjoin: [stopnum] [sector1] [line1] [sector2] [line2]
```
#**texture:**

[by Anthony Hall]

The texture: command's format is like this:

```
texture: [stopnum] [flag] [donor]
```

This command will copy the texturing from one specified sector to another one. It must be used in the INF entry of the sector that will be changed. [donor] is the sector to copy a texture from. The flag tells whether to copy ceiling to ceiling texture or floor to floor. If the flag starts with a letter then ceiling textures will be used. If it's a number then

# 20260KM

floors will be used.

I haven't been able to get it to work with walls or in trigger INF entries though.