

3D-Studio File Format (.3ds)  
Autodesk Ltd.

Document Revision 0.93 - Januari 1997

Rewritten by Martin van Velsen (email: vvelsen@ronix.ptf.hro.nl )  
and Robin Feroq ( 3ds-bin + mli )(email: robin@msrwww.fc-net.fr)  
Based on documentation by Jim Pitts ( email: jim@miconetics.com )  
Source update provided by:  
Albert Szilvasy (email: szilvasy@almos.vein.hu )

A lot of the chunks are still undocumented if you know what they do  
please email me Martin van Velsen, Robin Feroq or Jimm Pitts.  
As I get more information on the file format, I will document it for  
everyone to see. I will post this regularly to alt.3d and alt.3d-studio  
and I can be contacted there if my email does not work.

(Also see the 3d-studio material .mli documentation by Robin Feroq.)

Disclaimer.

This document describes the file format of the 3ds files generated by  
3d-studio by Autodesk. By using the information contained within, you  
agree not to hold any of the authors liable if, from its use, you  
f^Hmuck something up. OK?

Autodesk has at not yet released the official specifications of the  
3d-studio formats. You will therefor receive NO support from Autodesk  
or any company related to Autodesk concerning the nature and contents  
of the 3d-studio binary .3ds and .mli formats.

A warning beforehand. This docs describes the format of 3ds files  
produced by version 3.0 and higher of 3d-studio. You can find this  
version information at byte 29 in the binary file.

This document can be found on the regular newsgroups:  
alt.3d and alt.3d-studio  
It can also be found at: "http://www.mediatel.lu"

-----  
Contents

1. Introduction
2. Chunks anyone ?
3. 3D Editor chunks
4. Keyframer chunks
5. CODE

1. Introduction

=====

The 3ds file format is made up of chunks. They describe what information  
is to follow and what it is made up of, it's ID and the location of the  
next block. If you don't understand a chunk you can quite simply skip it.  
The next chunk pointer is relative to the start of the current chunk and  
in bytes. The binary information in the 3ds file is written in a special  
kind of way. Namely the lest significant byte comes first in an int.  
For example: 4A 5C (2 bytes in hex) would be 5C high byte and 4A low  
byte. In a long it is: 4A 5C 3B 8F where 5C4A is the low word and 8F 3B  
is the high word. And now for the chunks. A chunk is defined as:

```
start end size name
0      1  2   Chunk ID
2      5  4   Pointer to next chunk relative to the place where
               Chunk ID is, in other words the length of the chunk
```

Chunks have a hierarchy imposed on them that is identified by it's ID.  
A 3ds file has the Primary chunk ID 4D4Dh. This is always the first  
chunk of the file. Within the primary chunk are the main chunks.

to give you a preview and a reference to the hierarchy of chunks,  
below is a diagram to show the different chunk ID's and their place  
in the file. The chunks are given a name because below the diagram is  
a list which defines the names to the actual chunk id's. This makes it  
easier to put it in some source code ( how convenient that some sample  
code is included )

```
MAIN3DS (0x4D4D)
|
+---EDIT3DS (0x3D3D)
| |
| | +---EDIT_MATERIAL (0xAFFF)
| | |
| | | +---MAT_NAME01 (0xA000) (See mli Doc)
```

```

|
+--EDIT_CONFIG1 (0x0100)
+--EDIT_CONFIG2 (0x3E3D)
+--EDIT_VIEW_P1 (0x7012)
|
|   +--TOP (0x0001)
|   +--BOTTOM (0x0002)
|   +--LEFT (0x0003)
|   +--RIGHT (0x0004)
|   +--FRONT (0x0005)
|   +--BACK (0x0006)
|   +--USER (0x0007)
|   +--CAMERA (0xFFFF)
|   +--LIGHT (0x0009)
|   +--DISABLED (0x0010)
|   +--BOGUS (0x0011)
|
+--EDIT_VIEW_P2 (0x7011)
|
|   +--TOP (0x0001)
|   +--BOTTOM (0x0002)
|   +--LEFT (0x0003)
|   +--RIGHT (0x0004)
|   +--FRONT (0x0005)
|   +--BACK (0x0006)
|   +--USER (0x0007)
|   +--CAMERA (0xFFFF)
|   +--LIGHT (0x0009)
|   +--DISABLED (0x0010)
|   +--BOGUS (0x0011)
|
+--EDIT_VIEW_P3 (0x7020)
+--EDIT_VIEW1 (0x7001)
+--EDIT_BACKGR (0x1200)
+--EDIT_AMBIENT (0x2100)
+--EDIT_OBJECT (0x4000)
|
|   +--OBJ_TRIMESH (0x4100)
|   |
|   |   +--TRI_VERTEXL (0x4110)
|   |   +--TRI_VERTEXOPTIONS (0x4111)
|   |   +--TRI_MAPPINGCOORS (0x4140)
|   |   +--TRI_MAPPINGSTANDARD (0x4170)
|   |   +--TRI_FACEL1 (0x4120)
|   |   |
|   |   |   +--TRI_SMOOTH (0x4150)
|   |   |   +--TRI_MATERIAL (0x4130)
|   |   |
|   |   +--TRI_LOCAL (0x4160)
|   |   +--TRI_VISIBLE (0x4165)
|   |
|   +--OBJ_LIGHT (0x4600)
|   |
|   |   +--LIT_OFF (0x4620)
|   |   +--LIT_SPOT (0x4610)
|   |   +--LIT_UNKNWN01 (0x465A)
|   |
|   +--OBJ_CAMERA (0x4700)
|   |
|   |   +--CAM_UNKNWN01 (0x4710)
|   |   +--CAM_UNKNWN02 (0x4720)
|   |
|   +--OBJ_UNKNWN01 (0x4710)
|   +--OBJ_UNKNWN02 (0x4720)
|
+--EDIT_UNKNW01 (0x1100)
+--EDIT_UNKNW02 (0x1201)
+--EDIT_UNKNW03 (0x1300)
+--EDIT_UNKNW04 (0x1400)
+--EDIT_UNKNW05 (0x1420)
+--EDIT_UNKNW06 (0x1450)
+--EDIT_UNKNW07 (0x1500)
+--EDIT_UNKNW08 (0x2200)
+--EDIT_UNKNW09 (0x2201)
+--EDIT_UNKNW10 (0x2210)
+--EDIT_UNKNW11 (0x2300)
+--EDIT_UNKNW12 (0x2302)
+--EDIT_UNKNW13 (0x2000)
+--EDIT_UNKNW14 (0xAFFF)
|
+--KEYF3DS (0xB000)
|
|   +--KEYF_UNKNWN01 (0xB00A)
|   +--KEYF_FRAMES (0xB008)
|   +--KEYF_UNKNWN02 (0xB009)
|   +--KEYF_OBJDES (0xB002)
|   |
|   |   +--KEYF_OBJHIERARCH (0xB010)
|   |   +--KEYF_OBJDUMMYNAME (0xB011)
|   |   +--KEYF_OBJJUNKWN01 (0xB013)
|   |   +--KEYF_OBJJUNKWN02 (0xB014)
|   |   +--KEYF_OBJJUNKWN03 (0xB015)

```

```

+--KEYF_OBJPIVOT      (0xB020)
+--KEYF_OBJUNKWN04    (0xB021)
+--KEYF_OBJUNKWN05    (0xB022)

```

A chunk type which you can find all through the file are the color chunks which are called:

```

COL_RGB
COL_TRU
COL_UNK

```

## 2. Chunks anyone ?

```

=====

```

Now for the actual numbers, as you will see I used the define, because I ripped that part right out of my code. Be carefull however because there are a lot of new chunk types which were not documented in the original paper by Jim Pitts.

```

//>----- Primary chunk

#define MAIN3DS      0x4D4D

//>----- Main Chunks

#define EDIT3DS      0x3D3D // this is the start of the editor config
#define KEYF3DS      0xB000 // this is the start of the keyframer config

//>----- sub defines of EDIT3DS

#define EDIT_MATERIAL 0xAFFF
#define EDIT_CONFIG1  0x0100
#define EDIT_CONFIG2  0x3E3D
#define EDIT_VIEW_P1  0x7012
#define EDIT_VIEW_P2  0x7011
#define EDIT_VIEW_P3  0x7020
#define EDIT_VIEW1    0x7001
#define EDIT_BACKGR   0x1200
#define EDIT_AMBIENT  0x2100
#define EDIT_OBJECT   0x4000

#define EDIT_UNKNW01  0x1100
#define EDIT_UNKNW02  0x1201
#define EDIT_UNKNW03  0x1300
#define EDIT_UNKNW04  0x1400
#define EDIT_UNKNW05  0x1420
#define EDIT_UNKNW06  0x1450
#define EDIT_UNKNW07  0x1500
#define EDIT_UNKNW08  0x2200
#define EDIT_UNKNW09  0x2201
#define EDIT_UNKNW10  0x2210
#define EDIT_UNKNW11  0x2300
#define EDIT_UNKNW12  0x2302
#define EDIT_UNKNW13  0x3000
#define EDIT_UNKNW14  0xAFFF

//>----- sub defines of EDIT_OBJECT
#define OBJ_TRIMESH  0x4100
#define OBJ_LIGHT    0x4600
#define OBJ_CAMERA   0x4700

#define OBJ_UNKNWN01  0x4010
#define OBJ_UNKNWN02  0x4012 //>----- Could be shadow

//>----- sub defines of OBJ_CAMERA
#define CAM_UNKNWN01  0x4710
#define CAM_UNKNWN02  0x4720

//>----- sub defines of OBJ_LIGHT
#define LIT_OFF       0x4620
#define LIT_SPOT      0x4610
#define LIT_UNKNWN01  0x465A

//>----- sub defines of OBJ_TRIMESH
#define TRI_VERTXL    0x4110
#define TRI_FACEL2    0x4111
#define TRI_FACEL1    0x4120
#define TRI_SMOOTH    0x4150
#define TRI_LOCAL     0x4160
#define TRI_VISIBLE   0x4165

//>----- sub defs of KEYF3DS

#define KEYF_UNKNWN01 0xB009
#define KEYF_UNKNWN02 0xB00A
#define KEYF_FRAMES   0xB008
#define KEYF_OBJJDES  0xB002

//>----- these define the different color chunk types
#define COL_RGB       0x0010

```

```

#define COL_TRU  0x0011
#define COL_UNK  0x0013

//>>----- defines for viewport chunks

#define TOP      0x0001
#define BOTTOM   0x0002
#define LEFT    0x0003
#define RIGHT   0x0004
#define FRONT   0x0005
#define BACK    0x0006
#define USER    0x0007
#define CAMERA  0x0008 // 0xFFFF is the actual code read from file
#define LIGHT   0x0009
#define DISABLED 0x0010
#define BOGUS   0x0011

```

### 3. 3D Editor Chunks

=====

So far for the quick stuff now the more detailed info.

#### \* Main chunks

The main chunk ( the **primary chunk of 0x4D4D that is** ) is actually the **complete file**. So the size of this chunk is the size of the file minus the main chunk header.

There are **two more main** chunks, the **3d-editor** chunk and the **keyframer chunk**:

id

**3D3D** Start of Editor data ( this is also the place where the objects are )  
**B000** Start of Keyframer data

Directly after a Main chunk is another chunk. This could be any other type of chunk allowable within its main chunks scope. ( see diagram )

#### \* Subchunks of 3D3D

id	Description
0100	Part of configuration
1100	unknown
1200	Background Color
1201	unknown
1300	unknown
1400	unknown
1420	unknown
1450	unknown
1500	unknown
2100	Ambient Color Block
2200	fog ?
2201	fog ?
2210	fog ?
2300	unknown
3000	unknown
<b>3D3E</b>	Editor configuration main block
<b>4000</b>	Definition of an Object
<b>AFFF</b>	Start of material list

#### \* Subchunks of AFFF - Start of material list

\* A000 - material name  
 - This chunk contains the name of the material which is an ASCIIZ string

(  
 More material chunks are explained in the doc about 3d-studio .mli files. The chunk types mentioned in that doc are exactly the same as in the .3ds file  
 )

#### \* Subchunks of 3D3E - Editor configuration

id	Description
7001	Start of viewport indicator
7011	Viewport definition ( type 2 )
7012	Viewport definition ( type 1 )
7020	Viewport definition ( type 3 )

The **3D3E chunk** is a curious one because it **contains a lot of redundant data**. ( or so it seems ) The most important chunk is **7020**, this chunk **describes the 4 viewports which are active in the editor**. I assume that U are using the 4 normal viewport layout, because I have not tested it with other configurations. The editor confid will contain 5x chunk 7020 and 5x chunk 7011. only the first 4 7020 chunks are important for how the viewports look like. I guess that the other chunks only give additional info, but I am not sure. The things you are looking for in these chunks is at byte: 6 & 7 ( as offset from the first 6 bytes chunk header and pointer ) these bytes ( unsigned int ) contain the info at to what view is used, with the following id's:

id	Description
0001	Top
0002	Bottom
0003	Left
0004	Right
0005	Front
0006	Back
0007	User
FFFF	Camera
0009	Light
0010	Disabled

\* Subchunks of 4000 - Object description Block

- first item of Subchunk 4000 is an ASCIIZ string of the objects name.  
ASCIIZ means a string of characters ended by a zero.

Remember an Object can be a Camera a Light or a mesh

id	Description
4010	unknown
4012	shadow ?
4100	Triangular Polygon List ( Contains only subchunks )
4600	Light
4700	Camera

( Mapping:

These chunks are optional. They stand just after the vertex list when the object is mapped. )

\* Subchunks of 4100 - Triangular Polygon List

id	Description
4110	Vertex List
4111	Vertex Options
4120	Face List
4130	Face Material
4140	Mapping Coordinates
4150	Face smoothing group
4160	Translation Matrix
4165	Object visible/invisible
4170	Standard Mapping

\* 4110 - Vertex List

start	end	size	type	name
0	1	2	unsigned int	Total vertices in object
2	5	4	float	X-value
6	9	4	float	Y-value
10	13	4	float	Z-value

bytes 2..13 are repeated times the total amount of vertices in the object

\* 4111 - Vertex Options

First 2 bytes: number of vertices.

Then a short int for each vertex:

bit 0-7	0
bit 8-10	x
bit 11-12	0
bit 13	vertex selected in selection 3
bit 14	vertex selected in selection 2
bit 15	vertex selected in selection 1

bit 8-10 are just like random. From a save to another of the same scene it may change.

Other bits (0-7 and 11-12) have effects on visibility of vertex.

The 4111 chunk can be deleted without much influence, 3ds will still load the file all right.

\* 4120 - Face list

start	end	size	type	name
0	1	2	unsigned int	total polygons in object (numpoly)
2	3	2	unsigned int	number of vertex A
4	5	2	unsigned int	number of vertex B
6	7	2	unsigned int	number of vertex C
8	9	2	unsigned int	face info (*)

repeats 'numpoly' times for each polygon.

The first three ints are the three vertices of the face.  
0 stands for the first vertex defined in the vertex list.  
The order has a purpose: to give the direction for the normal of each face.  
If you turn a screw (standard screw) in the way the vertices indicate you will find the normal.

If vertices given in order are A B C:



This means unscrewing => the normal points out of the screen.

(\*) this number is a binary number which expands to 3 values.  
 for example 0x0006 would expand to 110 binary. The value should be read as 1 1 0 .This value can be found in 3d-studio ascii files as AB:1 BC:1 AC:0 .Which probably indicated the order of the vertices. For example AB:1 would be a normal line from A to B. But AB:0 would mean a line from B to A.

```

bit 0      AC visibility
bit 1      BC visibility
bit 2      AB visibility
bit 3      Mapping (if there is mapping for this face)
bit 4-8    0 (not used ?)
bit 9-10   x (chaotic ???)
bit 11-12  0 (not used ??)
bit 13     face selected in selection 3
bit 14     face selected in selection 2
bit 15     face selected in selection 1

```

#### \* 4130 - Face Material Chunk

If the object is all default material there is no 4130 chunk.  
 In fact, there is one 4130 chunk for each material present on the object.

Each 4130 face material chunks begins with an asciiz of a material, then after the null character is a short int that gives the number of faces of the object concerned by this material, then there is the list itself of these faces. 0000 means the first face of the (4120) face list.

```

***** Read the Doc on MLI files for more info on *****
***** Mapping and Materials *****

```

#### \* 4140 Mapping coordinates.

First 2 bytes: number of vertices.

Then, for each vertex 2 floats that give the mapping coordinates.  
 That is, if a point is at the center of the map it will have 0.5 0.5 as mapping coordinates.

#### \* 4150 - Face Smoothing Group

**nfaces\*4bytes**

If read as long int, the nth bit indicate if the face belongs or not to the nth smoothing group.

#### \* 4160 Local axis

Local axis info.

The **three first blocks of three floats** are the definition (in the absolute axis) of the local axis X Y Z of the object.  
 And the **last block of three floats is the local center of the object.**

#### \* 4170 Standard mapping

First 2 bytes: type of mapping

```

0 => plannar or specific (in this case, like mapping from the lofter,
the information of this chunk is irrelevant)
1 => cylindrical
2 => spherical

```

then come 21 floats that describe the mapping.

#### \* 4600 - Light

```

start end size type  name
0      3      4 float  Light pos X
4      7      4 float  Light pos Y
8     11      4 float  Light pos Z

```

after this structure check for more chunks.

```

id      Description ( full description later )
0010    RGB color
0011    24 bit color
4610    Light is a Spot light
4620    Light is off/on ( Boolean )

```

#### \* 4610 - Spot Light

```

start end size type  name
0      3      4 float  Target pos X

```

```

4      7      4  float  Target pos X
8     11     4  float  Target pos X
12    15     4  float  Hotspot
16    19     4  float  Falloff

```

## \* 0010 - RGB Color

```

start end size type  name
0      3      4  float  Red
4      7      4  float  Green
8     11     4  float  Blue

```

## \* 0011 - RGB Color - 24 bit

```

start end size type  name
0      1      1  byte  Red
1      1      1  byte  Green
2      2      1  byte  Blue

```

## \* 4700 - Camera

Describes the details of the camera in the scene

```

start end size type  name
0      3      4  float  Camera pos X
4      7      4  float  Camera pos Y
8     11     4  float  Camera pos Z
12    15     4  float  Camera target X
16    19     4  float  Camera target X
20    23     4  float  Camera target X
24    27     4  float  Camera bank ( rotation angle )
28    31     4  float  Camera lens

```

## 4. Keyframer Chunks

```
=====
```

## \* Keyframer chunk

```

id      Description
B00A    unknown
7001    See first description of this chunk
B008    Frames
B009    unknown
B002    Start object description

```

## \* B008 - Frame information

simple structure describing frame info

```

start end size type  name
0      3      4  unsigned long  start frame
4      7      4  unsigned long  end frame

```

## \*B002 - Start of Object info

Subhunks

```

id      Description
B010    Name & Hierarchy
B011*   Name Dummy Object
B013    unknown
B014*   unknown
B015    unknown
B020    Objects pivot point ?
B021    unknown
B022    unknown

```

## \* B010 - Name &amp; Hierarchy descriptor

```

start end size type  name
0      ?      ?  ASCIIZ      Object name
?      ?      2  unsigned int  unknown
?      ?      2  unsigned int  unknown
?      ?      2  unsigned int  Hierarchy of Object

```

The object hierarchy is a bit complex but works like this.

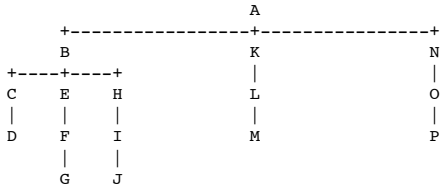
Each Object in the scene is given a number to identify its order in the tree. Also each object is ordered in the 3ds file as it would appear in the tree.

The root object is given the number -1 ( FFFF ).

As the file is read a counter of the object number is kept. Is the counter increments the objects are children of the previous objects. But when the pattern is broken by a number what will be less than the current counter the hierarchy returns to that level.

for example.

object name	hierarchy	
A	-1	
B	0	This example is taken from 50pman.3ds
C	1	
D	2	
E	1	I would really reccomend having a look at one of the examples with the hierarchy numbers to help work it out.
F	4	
G	5	
H	1	
I	7	
J	8	
K	0	
L	10	
M	11	
N	0	
O	13	
P	14	



Still not done with this chunk yet !  
 If the object name is \$\$\$DUMMY then it is a dummy object  
 and therefore you should expect a few extra chunks.

\* B011 - Dummy objects name.  
 Names a dummy object. ASCIIZ string.

\* B020 - Pivot Point ?  
 The Objects pivot point. Not quite sure what the first five floats do yet  
 ( ideas ?).

start	end	size	type	name
0	3	4	float	unknown
4	7	4	float	unknown
8	11	4	float	unknown
12	16	4	float	unknown
16	19	4	float	unknown
20	23	4	float	unknown
24	27	4	float	Pivot Y
28	32	4	float	Pivot X

5. CODE  
 =====

!!!!!! The source code is not UP-TO-DATE it is written after rev 0.9  
 of this Doc It will be updated when I can find the time ( or maybe YOU can)  
 If the code looks like it has been written by a a twelve year old, then  
 looks do deceieve, I like very simple and easy to read source code. All that  
 matters is that it does the trick.

```

-----8< cut here >8-----

/*-----*\
This is a lib which reads 3d-studio binary files from version 3.0
and higher
(v1.05)
author: Martin van Velsen
( and some great help by Gert van der Spoel )
email: vvelsen@ronix.ptf.hro.nl

If you happen to come across some variables with strange names, then
that will possible be Dutch names, sorry for that :)
\*-----*/
#ifdef __3DSBIN_H__
#define __3DSBIN_H__

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <conio.h> // IF you are on a dos system
#include <dos.h> // IF you are on a dos system
  
```



```

//>----- tools
#define __DEBUG__      0
#define TRUE          0
#define FALSE         1

//>----- Id Chunk
#define MAIN3DS       0x4D4D

//>----- Main Chunks
#define EDIT3DS       0x3D3D // this is the start of the editor config
#define KEYF3DS       0xB000 // this is the start of the keyframer config

//>----- sub defines of EDIT3DS
#define EDIT_MATERIAL 0xAFFF
#define EDIT_CONFIG1 0x0100
#define EDIT_CONFIG2 0x3E3D
#define EDIT_VIEW_P1 0x7012
#define EDIT_VIEW_P2 0x7011
#define EDIT_VIEW_P3 0x7020
#define EDIT_VIEWW1 0x7001
#define EDIT_BACKGR 0x1200
#define EDIT_AMBIENT 0x2100
#define EDIT_OBJECT 0x4000

#define EDIT_UNKNW01 0x1100
#define EDIT_UNKNW02 0x1201
#define EDIT_UNKNW03 0x1300
#define EDIT_UNKNW04 0x1400
#define EDIT_UNKNW05 0x1420
#define EDIT_UNKNW06 0x1450
#define EDIT_UNKNW07 0x1500
#define EDIT_UNKNW08 0x2200
#define EDIT_UNKNW09 0x2201
#define EDIT_UNKNW10 0x2210
#define EDIT_UNKNW11 0x2300
#define EDIT_UNKNW12 0x2302 // new chunk type
#define EDIT_UNKNW13 0x3000
#define EDIT_UNKNW14 0xAFFF

//>----- sub defines of EDIT_MATERIAL
#define MAT_NAME01 0xA000 //> includes name (see mli doc for materials)

//>----- sub defines of EDIT_OBJECT
#define OBJ_TRIMESH 0x4100
#define OBJ_LIGHT 0x4600
#define OBJ_CAMERA 0x4700

#define OBJ_UNKNWN01 0x4010
#define OBJ_UNKNWN02 0x4012 //>>---- Could be shadow

//>----- sub defines of OBJ_CAMERA
#define CAM_UNKNWN01 0x4710 // new chunk type
#define CAM_UNKNWN02 0x4720 // new chunk type

//>----- sub defines of OBJ_LIGHT
#define LIT_OFF 0x4620
#define LIT_SPOT 0x4610
#define LIT_UNKNWN01 0x465A

//>----- sub defines of OBJ_TRIMESH
#define TRI_VERTEXL 0x4110
#define TRI_FACEL2 0x4111 // unknown yet
#define TRI_FACEL1 0x4120
#define TRI_SMOOTH 0x4150
#define TRI_LOCAL 0x4160
#define TRI_VISIBLE 0x4165

//>>----- sub defs of KEYF3DS
#define KEYF_UNKNWN01 0xB009
#define KEYF_UNKNWN02 0xB00A
#define KEYF_FRAMES 0xB008
#define KEYF_OBJDES 0xB002

#define KEYF_OBJHIERARCH 0xB010
#define KEYF_OBJDUMMYNAME 0xB011
#define KEYF_OBJJUNKNWN01 0xB013
#define KEYF_OBJJUNKNWN02 0xB014
#define KEYF_OBJJUNKNWN03 0xB015
#define KEYF_OBJJUNKNWN04 0xB020
#define KEYF_OBJJUNKNWN05 0xB021
#define KEYF_OBJJUNKNWN06 0xB022

//>>----- these define the different color chunk types
#define COL_RGB 0x0010
#define COL_TRU 0x0011

```

```

#define COL_UNK 0x0013 // unknown

//>----- defines for viewport chunks

#define TOP 0x0001
#define BOTTOM 0x0002
#define LEFT 0x0003
#define RIGHT 0x0004
#define FRONT 0x0005
#define BACK 0x0006
#define USER 0x0007
#define CAMERA 0x0008 // 0xFFFF is the code read from file
#define LIGHT 0x0009
#define DISABLED 0x0010
#define BOGUS 0x0011

//>----- global vars

char *viewports [11]={
    "Bogus",
    "Top",
    "Bottom",
    "Left",
    "Right",
    "Front",
    "Back",
    "User",
    "Camera",
    "Light",
    "Disabled"
};

FILE *bin3ds;
unsigned long current_chunk=0L;
unsigned char views_read=0;
unsigned int numb_faces=0,numb_vertices=0;
char temp_name [100];
float trans_mat [4][4]; // translation matrix for objects

#endif

-----8< cut here >8-----

/*-----*\
   This is a lib which reads 3d-studio binary files from version 3.0
   and higher
   (v1.05)
   author: Martin van Velsen
          ( and some great help by Gert van der Spoel )
   email: vvelsen@ronix.ptf.hro.nl

   If you happen to come across some variables with strange names, then
   that will possible be Dutch names, sorry for that :)
*\-----*/
#ifndef __3DSBIN_C__
#define __3DSBIN_C__

#include "3ds_bin.h"

/*-----*\
unsigned char ReadChar (void)
{
    return (fgetc (bin3ds));
}

//>----- if you want to add some code to create a progress bar, then
//>----- I suggest you do it here. This is the only function which
//>----- reads from disk
*\-----*/
unsigned int ReadInt (void)
{
    unsigned int temp = ReadChar();
    return ( temp | (ReadChar () << 8));
}

/*-----*\
unsigned long ReadLong (void)
{
    unsigned long temp1,temp2;
    unsigned long temp3,temp4;

    temp1=ReadInt ();
    temp2=ReadInt ();

    return (temp3+(temp4*0x10000L));
}

/*-----*\
unsigned long ReadChunkPointer (void)
{
    return (ReadLong ());
}

```

```

}
/*-----*/
unsigned long GetChunkPointer (void)
{
return (ftell (bin3ds)-2); // compensate for the already read Marker
}
/*-----*/
void ChangeChunkPointer (unsigned long temp_pointer)
{
fseek (bin3ds,temp_pointer,SEEK_SET);
}
/*-----*/
int ReadName (void)
{
unsigned int teller=0;
unsigned char letter;

strcpy (temp_name,"Default name");

letter=ReadChar ();
if (letter==0) return (-1); // dummy object
temp_name [teller]=letter;
teller++;

do
{
letter=ReadChar ();
temp_name [teller]=letter;
teller++;
}
while ((letter!=0) && (teller<12));

temp_name [teller-1]=0;

#ifdef __DEBUG__
printf (" Found name : %s\n",temp_name);
#endif
return (0);
}
/*-----*/
int ReadLongName (void)
{
unsigned int teller=0;
unsigned char letter;

strcpy (temp_name,"Default name");

letter=ReadChar ();
if (letter==0) return (-1); // dummy object
temp_name [teller]=letter;
teller++;

do
{
letter=ReadChar ();
temp_name [teller]=letter;
teller++;
}
while (letter!=0);

temp_name [teller-1]=0;

#ifdef __DEBUG__
printf ("Found name : %s\n",temp_name);
#endif
return (0);
}
/*-----*/
unsigned long ReadUnknownChunk (unsigned int chunk_id)
{
unsigned long current_pointer;
unsigned long temp_pointer;

chunk_id=chunk_id;

current_pointer=GetChunkPointer ();
temp_pointer =ReadChunkPointer ();

ChangeChunkPointer (current_pointer+temp_pointer);
// move to the new chunk position
return (temp_pointer);
}
/*-----*/
unsigned long ReadRGBColor (void)
{
float rgb_val [3];

for (int i=0;i<3;i++)
fread (&(rgb_val [i]),sizeof (float),1,bin3ds);

#ifdef __DEBUG__
printf (" Found Color (RGB) def of: R:%5.2f,G:%5.2f,B:%5.2f\n",
rgb_val [0],

```

```

        rgb_val [1],
        rgb_val [2]);
#endif

return (12L);
}
/*-----*/
unsigned long ReadTrueColor (void)
{
    unsigned char true_c_val [3];

    for (int i=0;i<3;i++)
        true_c_val [i]=ReadChar ();

#ifdef __DEBUG__
    printf ("      Found Color (24bit) def of: R:%d,G:%d,B:%d\n",
            true_c_val [0],
            true_c_val [1],
            true_c_val [2]);
#endif

    return (3L);
}
/*-----*/
unsigned long ReadBooleanChunk (unsigned char *boolean)
{
    unsigned long current_pointer;
    unsigned long temp_pointer;

    current_pointer=GetChunkPointer ();
    temp_pointer  =ReadChunkPointer ();

    *boolean=ReadChar ();

    ChangeChunkPointer (current_pointer+temp_pointer); // move to the new chunk position
    return (temp_pointer);
}
/*-----*/
unsigned long ReadSpotChunk (void)
{
    unsigned long current_pointer;
    unsigned long temp_pointer;
    float target [4];
    float hotspot,falloff;

    current_pointer=GetChunkPointer ();
    temp_pointer  =ReadChunkPointer ();

    fread (&(target [0]),sizeof (float),1,bin3ds);
    fread (&(target [1]),sizeof (float),1,bin3ds);
    fread (&(target [2]),sizeof (float),1,bin3ds);
    fread (&hotspot,sizeof (float),1,bin3ds);
    fread (&falloff,sizeof (float),1,bin3ds);

#ifdef __DEBUG__
    printf ("      The target of the spot is at: X:%5.2f Y:%5.2f Z:%5.2f\n",
            target [0],
            target [1],
            target [2]);
    printf ("      The hotspot of this light is : %5.2f\n",hotspot);
    printf ("      The falloff of this light is : %5.2f\n",falloff);
#endif

    ChangeChunkPointer (current_pointer+temp_pointer);
    // move to the new chunk position
    return (temp_pointer);
}
/*-----*/
unsigned long ReadLightChunk (void)
{
    unsigned char end_found=FALSE,boolean;
    unsigned int temp_int;
    unsigned long current_pointer;
    unsigned long temp_pointer;
    unsigned long tellertje=6L; // 2 id + 4 pointer
    float light_coors [3];

    current_pointer=GetChunkPointer ();
    temp_pointer  =ReadChunkPointer ();

    fread (&(light_coors [0]),sizeof (float),1,bin3ds);
    fread (&(light_coors [1]),sizeof (float),1,bin3ds);
    fread (&(light_coors [2]),sizeof (float),1,bin3ds);

#ifdef __DEBUG__
    printf ("      Found light at coordinates: X: %5.2f, Y: %5.2f,Z: %5.2f\n",
            light_coors [0],
            light_coors [1],
            light_coors [2]);
#endif

    while (end_found==FALSE)
    {

```

```

temp_int=ReadInt ();

switch (temp_int)
{
case LIT_UNKNWN01 :
#ifdef __DEBUG__
printf (">>>> Found Light unknown chunk id of %0X\n",LIT_UNKNWN01);
#endif
tellertje+=ReadUnknownChunk (LIT_UNKNWN01);
break;

case LIT_OFF :
#ifdef __DEBUG__
printf (">>>> Light is (on/off) chunk: %0X\n",LIT_OFF);
#endif
tellertje+=ReadBooleanChunk (&boolean);
#ifdef __DEBUG__
if (boolean==TRUE)
printf ("      Light is on\n");
else
printf ("      Light is off\n");
#endif
break;

case LIT_SPOT :
#ifdef __DEBUG__
printf (">>>> Light is SpotLight: %0X\n",TRI_VERTEXL);
#endif
tellertje+=ReadSpotChunk ();
break;

case COL_RGB :
#ifdef __DEBUG__
printf (">>>> Found Color def (RGB) chunk id of %0X\n",temp_int);
#endif
tellertje+=ReadRGBColor ();
break;

case COL_TRU :
#ifdef __DEBUG__
printf (">>>> Found Color def (24bit) chunk id of %0X\n",temp_int);
#endif
tellertje+=ReadTrueColor ();
break;

default :
break;
}

tellertje+=2;
if (tellertje>=temp_pointer)
end_found=TRUE;
}

ChangeChunkPointer (current_pointer+temp_pointer);
// move to the new chunk position
return (temp_pointer);
}
/*-----*/
unsigned long ReadCameraChunk (void)
{
unsigned long current_pointer;
unsigned long temp_pointer;
float camera_eye [3];
float camera_focus [3];
float rotation,lens;

current_pointer=GetChunkPointer ();
temp_pointer =ReadChunkPointer ();

fread (&(camera_eye [0]),sizeof (float),1,bin3ds);
fread (&(camera_eye [1]),sizeof (float),1,bin3ds);
fread (&(camera_eye [2]),sizeof (float),1,bin3ds);

#ifdef __DEBUG__
printf ("      Found Camera viewpoint at coordinates: X: %5.2f, Y: %5.2f,Z: %5.2f\n",
camera_eye [0],
camera_eye [1],
camera_eye [2]);
#endif

fread (&(camera_focus [0]),sizeof (float),1,bin3ds);
fread (&(camera_focus [1]),sizeof (float),1,bin3ds);
fread (&(camera_focus [2]),sizeof (float),1,bin3ds);

#ifdef __DEBUG__
printf ("      Found Camera focus coors at coordinates: X: %5.2f, Y: %5.2f,Z: %5.2f\n",
camera_focus [0],
camera_focus [1],
camera_focus [2]);
#endif

fread (&rotation,sizeof (float),1,bin3ds);
fread (&lens,sizeof (float),1,bin3ds);
#ifdef __DEBUG__
printf ("      Rotation of camera is: %5.4f\n",rotation);
printf ("      Lens in used camera is: %5.4fmm\n",lens);
#endif
}

```

```

if ((temp_pointer-38)>0) // this means more chunks are to follow
{
#ifdef __DEBUG__
printf ("      **** found extra cam chunks ****\n");
#endif
if (ReadInt ()==CAM_UNKNWN01)
{
#ifdef __DEBUG__
printf ("      **** Found cam 1 type ch ****\n");
#endif
ReadUnknownChunk (CAM_UNKNWN01);
}
if (ReadInt ()==CAM_UNKNWN02)
{
#ifdef __DEBUG__
printf ("      **** Found cam 2 type ch ****\n");
#endif
ReadUnknownChunk (CAM_UNKNWN02);
}
}

ChangeChunkPointer (current_pointer+temp_pointer);
// move to the new chunk position
return (temp_pointer);
}
/*-----*/
unsigned long ReadVerticesChunk (void)
{
unsigned long current_pointer;
unsigned long temp_pointer;
float vertices [3]; // x,y,z
unsigned int numb_v;

current_pointer=GetChunkPointer ();
temp_pointer =ReadChunkPointer ();
numb_vertices =ReadInt ();

#ifdef __DEBUG__
printf ("      Found (%d) number of vertices\n",numb_vertices);
#endif

for (int i=0;i<numb_vertices;i++)
{
fread (&(vertices [0]),sizeof (float),1,bin3ds);
fread (&(vertices [1]),sizeof (float),1,bin3ds);
fread (&(vertices [2]),sizeof (float),1,bin3ds);

#ifdef __DEBUG__
printf ("      Vertex nr%d: X: %5.2f Y: %5.2f Z:%5.2f\n",
i,
vertices [0],
vertices [1],
vertices [2]);
#endif
}

ChangeChunkPointer (current_pointer+temp_pointer);
// move to the new chunk position
return (temp_pointer);
}
/*-----*/
unsigned long ReadSmoothingChunk ()
{
unsigned long current_pointer;
unsigned long temp_pointer;
unsigned long smoothing;

current_pointer=GetChunkPointer ();
temp_pointer =ReadChunkPointer ();

for (int i=0;i<numb_faces;i++)
{
smoothing=ReadLong();
smoothing=smoothing; // compiler warnig depressor *)
#ifdef __DEBUG__
printf ("      The smoothing group for face [%5d] is %d\n",i,smoothing);
#endif
}

ChangeChunkPointer (current_pointer+temp_pointer);
// move to the new chunk position
return (temp_pointer);
}
/*-----*/
unsigned long ReadFacesChunk (void)
{
unsigned long current_pointer;
unsigned long temp_pointer;
unsigned int temp_diff;
unsigned int faces [6]; // a,b,c,Diff (Diff= AB: BC: CA: )

current_pointer=GetChunkPointer ();
temp_pointer =ReadChunkPointer ();

```

```

numb_faces    =ReadInt ();
#ifdef __DEBUG__
printf ("      Found (%d) number of faces\n",numb_faces);
#endif

for (int i=0;i<numb_faces;i++)
{
  faces [0]=ReadInt ();
  faces [1]=ReadInt ();
  faces [2]=ReadInt ();
  temp_diff=ReadInt () & 0x000F;
  faces [3]=(temp_diff & 0x0004) >> 2;
  faces [4]=(temp_diff & 0x0002) >> 1;
  faces [5]=(temp_diff & 0x0001);

#ifdef __DEBUG__
printf ("      Face nr:%d, A: %d B: %d C:%d , AB:%d BC:%d CA:%d\n",
        i,
        faces [0],
        faces [1],
        faces [2],
        faces [3],
        faces [4],
        faces [5]);
#endif
}

if (ReadInt ()==TRI_SMOOTH)
  ReadSmoothingChunk ();
#ifdef __DEBUG__
else
  printf ("      No smoothing groups found, assuming autosmooth\n");
#endif

ChangeChunkPointer (current_pointer+temp_pointer);
// move to the new chunk position
return (temp_pointer);
}
/*-----*/
unsigned long ReadTranslationChunk (void)
{
  unsigned long current_pointer;
  unsigned long temp_pointer;
  current_pointer=GetChunkPointer ();
  temp_pointer  =ReadChunkPointer ();

  for (int j=0;j<4;j++)
  {
    for (int i=0;i<3;i++)
      fread (&(trans_mat [j][i]),sizeof (float),1,bin3ds);
  }

  trans_mat [0][3]=0;
  trans_mat [1][3]=0;
  trans_mat [2][3]=0;
  trans_mat [3][3]=1;

#ifdef __DEBUG__
printf ("      The translation matrix is:\n");
for (int i=0;i<4;i++)
  printf ("      | %5.2f %5.2f %5.2f %5.2f |\n",
        trans_mat [i][0],
        trans_mat [i][1],
        trans_mat [i][2],
        trans_mat [i][3]);
#endif

ChangeChunkPointer (current_pointer+temp_pointer);
// move to the new chunk position
return (temp_pointer);
}
/*-----*/
unsigned long ReadObjChunk (void)
{
  unsigned char end_found=FALSE,boolean=TRUE;
  unsigned int temp_int;
  unsigned long current_pointer;
  unsigned long temp_pointer;
  unsigned long tellertje=6L; // 2 id + 4 pointer

  current_pointer=GetChunkPointer ();
  temp_pointer  =ReadChunkPointer ();

  while (end_found==FALSE)
  {
    temp_int=ReadInt ();

    switch (temp_int)
    {
      case TRI_VERTEXL :
#ifdef __DEBUG__
printf (">>>>> Found Object vertices chunk id of %0X\n",
        temp_int);
#endif

```

```

        #endif
        tellertje+=ReadVerticesChunk ();
        break;
    case TRI_FACEL1 :
        #ifdef __DEBUG__
        printf (">>>> Found Object faces (1) chunk id of %0X\n",
            temp_int);
        #endif
        tellertje+=ReadFacesChunk ();
        break;
    case TRI_FACEL2 :
        #ifdef __DEBUG__
        printf (">>>> Found Object faces (2) chunk id of %0X\n",
            temp_int);
        #endif
        tellertje+=ReadUnknownChunk (temp_int);
        break;
    case TRI_LOCAL :
        #ifdef __DEBUG__
        printf (">>>> Found Object translation chunk id of %0X\n",
            temp_int);
        #endif
        tellertje+=ReadTranslationChunk ();
        break;
    case TRI_VISIBLE :
        #ifdef __DEBUG__
        printf (">>>> Found Object vis/invis chunk id of %0X\n",
            temp_int);
        #endif
        tellertje+=ReadBooleanChunk (&boolean);

        #ifdef __DEBUG__
        if (boolean==TRUE)
            printf ("    Object is (visible)\n");
        else
            printf ("    Object is (not visible)\n");
        #endif
        break;
    default:
        break;
}

```

```

tellertje+=2;
if (tellertje>=temp_pointer)
    end_found=TRUE;
}

```

```

ChangeChunkPointer (current_pointer+temp_pointer);
// move to the new chunk position
return (temp_pointer);
}

```

```

/*-----*/

```

```

unsigned long ReadObjectChunk (void)
{
    unsigned char end_found=FALSE;
    unsigned int temp_int;
    unsigned long current_pointer;
    unsigned long temp_pointer;
    unsigned long tellertje=6L; // 2 id + 4 pointer

```

```

    current_pointer=GetChunkPointer ();
    temp_pointer =ReadChunkPointer ();

```

```

    if (ReadName ()==-1)
    {
        #ifdef __DEBUG__
        printf (">>>>* Dummy Object found\n");
        #endif
    }

```

```

    while (end_found==FALSE)
    {

```

```

        temp_int=ReadInt ();

```

```

        switch (temp_int)
        {

```

```

            case OBJ_UNKNWN01:tellertje+=ReadUnknownChunk (OBJ_UNKNWN01);break;
            case OBJ_UNKNWN02:tellertje+=ReadUnknownChunk (OBJ_UNKNWN02);break;
            case OBJ_TRIMESH :

```

```

                #ifdef __DEBUG__
                printf (">>>> Found Obj/Mesh chunk id of %0X\n",
                    OBJ_TRIMESH);
                #endif

```

```

                tellertje+=ReadObjChunk ();
                break;

```

```

            case OBJ_LIGHT :
                #ifdef __DEBUG__
                printf (">>>> Found Light chunk id of %0X\n",
                    OBJ_LIGHT);
                #endif

```

```

                tellertje+=ReadLightChunk ();
                break;

```

```

            case OBJ_CAMERA :
                #ifdef __DEBUG__

```



```

        printf (">>> Found Camera chunk id of %0X\n",
                OBJ_CAMERA);
    #endif
    tellertje+=ReadCameraChunk ();
    break;
default:
    break;
}

tellertje+=2;
if (tellertje>=temp_pointer)
    end_found=TRUE;
}

ChangeChunkPointer (current_pointer+temp_pointer);
// move to the new chunk position
return (temp_pointer);
}
/*-----*/
unsigned long ReadBackgrChunk (void)
{
    unsigned char end_found=FALSE;
    unsigned int temp_int;
    unsigned long current_pointer;
    unsigned long temp_pointer;
    unsigned long tellertje=6L; // 2 id + 4 pointer

    current_pointer=GetChunkPointer ();
    temp_pointer =ReadChunkPointer ();

    while (end_found==FALSE)
    {
        temp_int=ReadInt ();

        switch (temp_int)
        {
            case COL_RGB :
                #ifdef _DEBUG_
                printf (">> Found Color def (RGB) chunk id of %0X\n",
                        temp_int);
                #endif
                tellertje+=ReadRGBColor ();
                break;

            case COL_TRU :
                #ifdef _DEBUG_
                printf (">> Found Color def (24bit) chunk id of %0X\n",
                        temp_int);
                #endif
                tellertje+=ReadTrueColor ();
                break;

            default:
                break;
        }

        tellertje+=2;
        if (tellertje>=temp_pointer)
            end_found=TRUE;
    }

    ChangeChunkPointer (current_pointer+temp_pointer);
    // move to the new chunk position
    return (temp_pointer);
}
/*-----*/
unsigned long ReadAmbientChunk (void)
{
    unsigned char end_found=FALSE;
    unsigned int temp_int;
    unsigned long current_pointer;
    unsigned long temp_pointer;
    unsigned long tellertje=6L; // 2 id + 4 pointer

    current_pointer=GetChunkPointer ();
    temp_pointer =ReadChunkPointer ();

    while (end_found==FALSE)
    {
        temp_int=ReadInt ();

        switch (temp_int)
        {
            case COL_RGB :
                #ifdef _DEBUG_
                printf (">>>> Found Color def (RGB) chunk id of %0X\n",
                        temp_int);
                #endif
                tellertje+=ReadRGBColor ();
                break;

            case COL_TRU :
                #ifdef _DEBUG_
                printf (">>>> Found Color def (24bit) chunk id of %0X\n",
                        temp_int);
                #endif
                tellertje+=ReadTrueColor ();
                break;
        }
    }
}

```

```

        default:      break;
    }

    tellertje+=2;
    if (tellertje>=temp_pointer)
        end_found=TRUE;
}

ChangeChunkPointer (current_pointer+temp_pointer);
// move to the new chunk position
return (temp_pointer);
}
/*-----*/
unsigned long FindCameraChunk (void)
{
    long temp_pointer=0L;

    for (int i=0;i<12;i++)
        ReadInt ();

    temp_pointer=11L;
    temp_pointer=ReadName ();

#ifdef __DEBUG__
    if (temp_pointer!=-1)
        printf (">>>>* No Camera name found\n");
#endif

    return (temp_pointer);
}
/*-----*/
unsigned long ReadViewPortChunk (void)
{
    unsigned long current_pointer;
    unsigned long temp_pointer;
    unsigned int port,attribution;

    views_read++;

    current_pointer=GetChunkPointer ();
    temp_pointer =ReadChunkPointer ();

    attribution=ReadInt ();
    if (attribution==3)
    {
#ifdef __DEBUG__
        printf ("<Snap> active in viewport\n");
#endif
    }
    if (attribution==5)
    {
#ifdef __DEBUG__
        printf ("<Grid> active in viewport\n");
#endif
    }
}

for (int i=1;i<6;i++) ReadInt (); // read 5 ints to get to the viewport

port=ReadInt ();
if ((port==0xFFFF) || (port==0))
{
    FindCameraChunk ();
    port=CAMERA;
}

#ifdef __DEBUG__
printf ("Reading [%s] information with id:%d\n",viewports [port],port);
#endif

ChangeChunkPointer (current_pointer+temp_pointer);
// move to the new chunk position
return (temp_pointer);
}
/*-----*/
unsigned long ReadViewChunk (void)
{
    unsigned char end_found=FALSE;
    unsigned int temp_int;
    unsigned long current_pointer;
    unsigned long temp_pointer;
    unsigned long tellertje=6L;

    current_pointer=GetChunkPointer ();
    temp_pointer =ReadChunkPointer ();

    while (end_found==FALSE)
    {
        temp_int=ReadInt ();

        switch (temp_int)
        {
            case EDIT_VIEW_P1 :
#ifdef __DEBUG__

```

```

        printf (">>>> Found Viewport1 chunk id of %0X\n",
                temp_int);
        #endif
        tellertje+=ReadViewPortChunk ();
        break;
    case EDIT_VIEW_P2 :
        #ifdef __DEBUG__
        printf (">>>> Found Viewport2 (bogus) chunk id of %0X\n",
                temp_int);
        #endif
        tellertje+=ReadUnknownChunk (EDIT_VIEW_P2);
        break;
    case EDIT_VIEW_P3 :
        #ifdef __DEBUG__
        printf (">>>> Found Viewport chunk id of %0X\n",
                temp_int);
        #endif
        tellertje+=ReadViewPortChunk ();
        break;
    default
    :break;
}

tellertje+=2;
if (tellertje>=temp_pointer)
    end_found=TRUE;

if (views_read>3)
    end_found=TRUE;
}

ChangeChunkPointer (current_pointer+temp_pointer);
// move to the new chunk position
return (temp_pointer);
}
/*-----*/
unsigned long ReadMatDefChunk (void)
{
    unsigned long current_pointer;
    unsigned long temp_pointer;

    current_pointer=GetChunkPointer ();
    temp_pointer =ReadChunkPointer ();

    if (ReadLongName ()===-1)
    {
        #ifdef __DEBUG__
        printf (">>>>* No Material name found\n");
        #endif
    }

    ChangeChunkPointer (current_pointer+temp_pointer);
    // move to the new chunk position
    return (temp_pointer);
}
/*-----*/
unsigned long ReadMaterialChunk (void)
{
    unsigned char end_found=FALSE;
    unsigned int temp_int;
    unsigned long current_pointer;
    unsigned long temp_pointer;
    unsigned long tellertje=6L;

    current_pointer=GetChunkPointer ();
    temp_pointer =ReadChunkPointer ();

    while (end_found==FALSE)
    {
        temp_int=ReadInt ();

        switch (temp_int)
        {
            case MAT_NAME01 :
                #ifdef __DEBUG__
                printf (">>>> Found Material def chunk id of %0X\n",
                        temp_int);
                #endif
                tellertje+=ReadMatDefChunk ();
                break;

            default:break;
        }

        tellertje+=2;
        if (tellertje>=temp_pointer)
            end_found=TRUE;
    }

    ChangeChunkPointer (current_pointer+temp_pointer);
    // move to the new chunk position
    return (temp_pointer);
}
/*-----*/
unsigned long ReadEditChunk (void)

```

```

{
unsigned char end_found=FALSE;
unsigned int temp_int;
unsigned long current_pointer;
unsigned long temp_pointer;
unsigned long tellertje=6L;

current_pointer=GetChunkPointer ();
temp_pointer  =ReadChunkPointer ();

while (end_found==FALSE)
{
temp_int=ReadInt ();

switch (temp_int)
{
case EDIT_UNKNW01:tellertje+=ReadUnknownChunk (EDIT_UNKNW01);break;
case EDIT_UNKNW02:tellertje+=ReadUnknownChunk (EDIT_UNKNW02);break;
case EDIT_UNKNW03:tellertje+=ReadUnknownChunk (EDIT_UNKNW03);break;
case EDIT_UNKNW04:tellertje+=ReadUnknownChunk (EDIT_UNKNW04);break;
case EDIT_UNKNW05:tellertje+=ReadUnknownChunk (EDIT_UNKNW05);break;
case EDIT_UNKNW06:tellertje+=ReadUnknownChunk (EDIT_UNKNW06);break;
case EDIT_UNKNW07:tellertje+=ReadUnknownChunk (EDIT_UNKNW07);break;
case EDIT_UNKNW08:tellertje+=ReadUnknownChunk (EDIT_UNKNW08);break;
case EDIT_UNKNW09:tellertje+=ReadUnknownChunk (EDIT_UNKNW09);break;
case EDIT_UNKNW10:tellertje+=ReadUnknownChunk (EDIT_UNKNW10);break;
case EDIT_UNKNW11:tellertje+=ReadUnknownChunk (EDIT_UNKNW11);break;
case EDIT_UNKNW12:tellertje+=ReadUnknownChunk (EDIT_UNKNW12);break;
case EDIT_UNKNW13:tellertje+=ReadUnknownChunk (EDIT_UNKNW13);break;

case EDIT_MATERIAL :
#ifdef __DEBUG__
printf (">>> Found Materials chunk id of %0X\n",
temp_int);
#endif
tellertje+=ReadMaterialChunk ();
break;
case EDIT_VIEW1 :
#ifdef __DEBUG__
printf (">>> Found View main def chunk id of %0X\n",
temp_int);
#endif
tellertje+=ReadViewChunk ();
break;
case EDIT_BACKGR :
#ifdef __DEBUG__
printf (">>> Found Backgr chunk id of %0X\n",
temp_int);
#endif
tellertje+=ReadBackgrChunk ();
break;
case EDIT_AMBIENT :
#ifdef __DEBUG__
printf (">>> Found Ambient chunk id of %0X\n",
temp_int);
#endif
tellertje+=ReadAmbientChunk ();
break;
case EDIT_OBJECT :
#ifdef __DEBUG__
printf (">>> Found Object chunk id of %0X\n",
temp_int);
#endif
tellertje+=ReadObjectChunk ();
break;
default:
break;
}

tellertje+=2;
if (tellertje>=temp_pointer)
end_found=TRUE;
}

ChangeChunkPointer (current_pointer+temp_pointer);
// move to the new chunk position
return (temp_pointer);
}
/*-----*/
unsigned long ReadKeyfChunk (void)
{
unsigned char end_found=FALSE;
unsigned int temp_int;
unsigned long current_pointer;
unsigned long temp_pointer;
unsigned long tellertje=6L;

current_pointer=GetChunkPointer ();
temp_pointer  =ReadChunkPointer ();

while (end_found==FALSE)
{
temp_int=ReadInt ();

```

```

switch (temp_int)
{
case KEYF_UNKNWN01 :tellertje+=ReadUnknownChunk (temp_int);break;
case KEYF_UNKNWN02 :tellertje+=ReadUnknownChunk (temp_int);break;
case KEYF_FRAMES :
#ifdef __DEBUG__
printf (">>> Found Keyframer frames chunk id of %0X\n",
temp_int);
#endif
tellertje+=ReadUnknownChunk (temp_int);
break;
case KEYF_OBJDES :
#ifdef __DEBUG__
printf (">>> Found Keyframer object description chunk id of %0X\n",
temp_int);
#endif
tellertje+=ReadUnknownChunk (temp_int);
break;
case EDIT_VIEW1 :
#ifdef __DEBUG__
printf (">>> Found View main def chunk id of %0X\n",
temp_int);
#endif
tellertje+=ReadViewChunk ();
break;
default:
break;
}

tellertje+=2;
if (tellertje>=temp_pointer)
end_found=TRUE;
}

ChangeChunkPointer (current_pointer+temp_pointer);
// move to the new chunk position
return (temp_pointer);
}
/*-----*/
unsigned long ReadMainChunk (void)
{
unsigned char end_found=FALSE;
unsigned int temp_int;
unsigned long current_pointer;
unsigned long temp_pointer;
unsigned long tellertje=6L;

current_pointer=GetChunkPointer ();
temp_pointer =ReadChunkPointer ();

while (end_found==FALSE)
{
temp_int=ReadInt ();

switch (temp_int)
{
case KEYF3DS :
#ifdef __DEBUG__
printf (">> Found *Keyframer* chunk id of %0X\n",KEYF3DS);
#endif
tellertje+=ReadKeyfChunk ();
break;
case EDIT3DS :
#ifdef __DEBUG__
printf (">> Found *Editor* chunk id of %0X\n",EDIT3DS);
#endif
tellertje+=ReadEditChunk ();
break;
default:
break;
}

tellertje+=2;
if (tellertje>=temp_pointer)
end_found=TRUE;
}

ChangeChunkPointer (current_pointer+temp_pointer);
// move to the new chunk position
return (temp_pointer);
}
/*-----*/
int ReadPrimaryChunk (void)
{
unsigned char version;

if (ReadInt ()==MAIN3DS)
{
#ifdef __DEBUG__
printf ("> Found Main chunk id of %0X\n",MAIN3DS);
#endif
//>----- find version number
fseek (bin3ds,28L,SEEK_SET);
version=ReadChar ();
if (version<3)

```

```
{
#ifdef __DEBUG__
printf ("Sorry this lib can only read 3ds files of version 3.0 and higher\n");
printf ("The version of the file you want to read is: %d\n",version);
#endif
return (1);
}
fseek (bin3ds,2,SEEK_SET);
ReadMainChunk ();
}
else
return (1);

return (0);
}
/*-----*/
/*                               Test Main for the 3ds-bin lib                               */
/*-----*/
int main (int argc,char **argv)
{
argc=argc;

bin3ds=fopen (argv [1],"rb");
if (bin3ds==NULL)
return (-1);

#ifdef __DEBUG__
printf ("\nLoading 3ds binary file : %s\n",argv [1]);
#endif
while (ReadPrimaryChunk ()==0);

return (0);
}
/*-----*/
#endif
```