Chapter 3 Library Documentation

Overview

Introduction

File formats used by 3D Studio are complex collections of the hundreds of data structures used by 3D Studio. Creating an application to read, write and/or modify these files is an arduous and time-consuming task because of the large number of dissimilar structures involved. A majority of these structures feature constraints and interactions that, if not adhered to, can cause failure when 3D  Studio tries to read or write these files.

The 3D Studio File Toolkit provides an easy and safe abstraction to the complexities of the file format. It allows the application to treat 3D Studio files like databases, assembling the bits and pieces into higher level structures which can be freely added, modified, or removed from the database. Additional functionality is provided to verify and fix databases, making them safe to use with 3D Studio.

Toolkit Concepts

Files

Files are the source or repository for database information. Once opened and associated with a database, they cannot be closed until the database itself is released. Multiple files may be opened, read, and written. During the course of normal toolkit operation, temporary files may be created to store large, unwritten objects (like newly create meshes) in file storage instead of system memory.

Databases

Databases interface to the files by maintaining an index to the many structures contained in the file. Once a file is associated with the database, all subsequent file read and writes are accomplished though requests to the database. A database may be constructed from any number of files, and may even be filled with information that has no physical file representation.

Database Objects

There are several kinds of high level objects that are supported in the database. There are settings, materials, named objects, and object motions. Settings only occur once per database, and have simple methods to retrieve them. Materials, named object, and object motions all have unique names, and the name is used when referring to the object.

Database Queries

A database is created from a file by adding references to the file into the database. Initially, the references contain no data, only a pointer to the file and an offset to the data in the file. When requests are made of a database, say to retrieve a mesh, data is temporarily read into the database and is passed or copied into the object, and then released from the database to keep memory usage at a minimum.

Object Additions

Objects and settings may be added to the database at any time. When an object is added to the database, it is done by copying the objects data into the database, leaving the source object unchanged. Depending on the size of the data, the database may then append the new data to the end of a temporary file.

Object Deletions

Objects and settings may be deleted from the database at any time. When an object is deleted from the database, only the reference to the object is deleted, not the file information. Only if the database is later written back to the file containing the deleted object will the object be deleted from the file.

Object Modifications

Objects and settings may be changed in the database at any time. When a new object has the same name as an existing object, the old object is deleted and the new object is put in its place in the database. As with deletions, the old object isn't truly gone from the file until the file is replaced with the contents of the database.

Object Copying

Objects may be shallow copied or deep copied from one database to another. Shallow copies are copies of the data references, not of the data itself. Two shallow copies in two different databases will share the reference to the same file. A deep copy does not copy the reference, but copies the data into another location and creates new references to it.

Viewport Definitions

The toolkit supports the creation of only a single view layout that is identical for both the 3D Editor and Keyframer. More complex viewport configurations must be copied from existing files.

Database Writing

A database is written to a file by traversing each entry in the database a structure at a time, reading the data from its source file, and writing it into the new file. At the same time, the references in the database are changed to reflect the data's new file location. When writing to a file that has been previously read, the old file is given a temporary name so that other databases may continue to reference that file.

Temporary Files

Temporary files continue to exist until the databases that reference data contained within them cease to do so.

Libraries

The 3D Studio File Toolkit is implemented in four different library files. Each library is intended for use with a different compiler environment. A description of each follows:

1. Use 3dftkm30. lib with MetaWare High C/C++ 3.0. The library contains object files in the Easy OMF-386 format. Link the library using the 386LINK utility in Phar Lap 386.

2. Use 3dftkm32. lib with MetaWare High C/C++ 3.2. The library contains object files in the Easy OMF-386 format. Link the library using the 386LINK utility in Phar Lap 386.

3. Use 3dftkw95. lib with WATCOM C/C++ 32 9.5. The library contains object files in the Easy OMF-386 format. Link the library using the WLINK utility provided with WATCOM C/C++ 32 9.5.

4. Use 3dftkv10. lib with Microsoft Visual C++ 1.0. See the readme.doc file for details on the use of the toolkit with this language.

Include Files

The toolkit comes with two include files:

o 3dsftk.h defines the public data structures, prototypes and constants used in the library. Include it after any standard compiler include files.

o 3dsftkpr.h defines the private undocumented data structures, prototypes and constants used in the library. It is included automatically by 3dsftk.h

Using the Compilers and Linkers

The toolkit supports four different compiler and linker combinations. The following list of example compiler and linker command lines uses [COMPILER NAME] to represent the compiler directory, and [3DSFTK] to represent the toolkit directory. Actual command lines should substitute the correct directory names for these symbols.

Compiler Settings

MetaWare High C/C++ 3.0

The MetaWare High C/C++ 3.0 compiler command line should look like this:

HC386 –I[METAWARE] –I[3DSFTK]\INC –c -f387 -Hpro=3dsftk.pro

The 3dsftk.pro file should contain these settings:

#pragma Memory\_model(Small);#pragma Align\_members(1); #pragma On(Floating\_point); #pragma On(Callee\_pops\_when\_possible); #pragma On(Push\_regsize);

MetaWare High C/C++ 3.2

The MetaWare High C/C++ 3.0 compiler command line should look like this:

HC386 –I[METAWARE] –I[3DSFTK]\INC –c –f387 -Hpro=3dsftk.pro

The 3dsftk.pro file should contain these settings:

#pragma Memory\_model(Small);#pragma Align\_members(1); #pragma On(Floating\_point); #pragma On(Callee\_pops\_when\_possible); #pragma On(Push\_regsize);

WATCOM C/C++ 32 9.5

The WATCOM C/C++ 32 optimizing Compiler and Tools 9.5 command line should look like this:

WCC386P /i=[WATCOM]\H /i=[3DSFTK]\INC /j /ez /mf /fpi /4s /oailt

Linker Settings

MetaWare High C/C++ 3.0

For MetaWare High C/C++ 3.0, the linker command line should look like this:

386LINK objname.obj –s 20480 –nom –lib [METAWARE]\small\hc387.lib -lib [METAWARE]\small\hc386.lib -lib [METAWARE]\small\hcna.lib --lib [3DSFTK]\lib\3dftkm30.lib –exe progname.exp

MetaWare High C/C++ 3.2

For MetaWare High C/C++ 3.2, the linker command line should look like this:

386LINK objname.obj –s 20480 –nom –lib [METAWARE]\small\hc387.lib -lib [METAWARE]\small\hc386.lib -lib [METAWARE]\small\hcna.lib --lib [3DSFTK]\lib\3dftkm32.lib –exe progname.exp

WATCOM C/C++ 32 9.5

For WATCOM C/C++ 32 Optimizing Compiler and Tools 9.5, the linker command line should look like this:

FILE objname.obj OPTION STACK=20480 LIBPATH [WATCOM]\LIB386; [WATCOM]\LIB386\DOS LIBRARY CLIB3S.LIB LIBRARY EMU387.LIB LIBRARY [3DSFTK]\lib\3dftkw95.lib FORMAT PHARLAP OPTION DOSSEG NAME progname.exp

Library Reference

Basic Datatypes, Defines, and Constants

Basic Data Type Definitions

A number of macros are used for the simple data types. The macros ease conflicts with compilers and platforms that use different sizes for their native types.

#define char3ds char

A signed single byte character.

#define uchar3ds unsigned char

An unsigned single byte character.

#define byte3ds char

A signed byte value.

#define ubyte3ds char

An unsigned byte value.

#define short3ds short int

A signed two-byte value.

#define ushort3ds unsigned short int

An unsigned two-byte value.

#define long3ds long int

A signed four-byte value.

#define ulong3ds unsigned long int

An unsigned four-byte value.

#define float3ds float

A four-byte floating point value.

#define double3ds double

An eight-byte floating point value.

#define void3ds void

A typeless definition for near pointers.

Toolkit Revision Level

A macro for testing the version of the toolkit being used is provided.

#define FTKRevisionLevel3ds 2

static const long3ds \_\_FTKRevisionLevel3ds = FTKRevisionLevel3ds;

The revision level increments to the next integer number anytime the 3dsftk.h is changed in a way that effects structures, definitions, or function prototypes.

Booleans

static const int True3ds = (1 == 1);

static const int False3ds = (1 != 1);

Conditionals used in the fields of many structures.

Commonly Used Primitives

Point3ds

Single precision 3D coordinates.

typedef struct

{

float3ds x, y, z;

} point3ds;

float3ds x

???

float3ds y

???

float3ds z

???

Fcolor3ds

Color definition in RGB color space

typedef struct

{

float3ds r;

float3ds g;

float3ds b;

} fcolor3ds;

float3ds r

Red component ranging from 0.0 (darkest) to 1.0 (lightest).

float3ds g

Green component ranging from 0.0 to 1.0.

float3ds b

Blue component ranging from 0.0 to 1.0.

File and Database

File3ds

File control

Used by the toolkit to provide some additional information about a file. The structure should only be manipulated through the OpenFile3ds and CloseAllFiles3ds functions and is documented for reference only.

typedef struct

{

FILE \*file;

char3ds \*filename;

filestate3ds state;

ubyte3ds index;

} file3ds;

FILE \*file

Buffered IO file structure. Defaults to NULL.

char3ds \*filename

Name of open file. Defaults to NULL.

filestate3ds state

Mode of file open. An enumerated type as follows:

typedef enum {      StateNotKnown, /\* File not open yet \*/      ReadFromFile, /\* File open for read \*/      WriteToFile, /\* File open for write \*/      ReadWriteFile /\* File open for read and write \*/} filestate3ds;

ubyte3ds index

The toolkit maintains a list of all open files so that individual chunk3ds structures only need to keep a single byte index to a file instead of a much larger file3ds pointer.

OpenFile3ds

***Prototype:*** file3ds \*OpenFile3ds(const char3ds \*name, const char \*attrib)

Allocates and initializes a file3ds structure and opens a .3ds file for reading or writing.

Arguments:

const char3ds \*name

The complete filename and path of the .3ds file being opened. If no path is specified, the default is the local path.

const char \*attrib

The mode for the file open call, a string containing either of the characters "r" and "w". To open the file for reading, use a single "r". To open the file for reading and writing, use a single "w".

If the call to open the file was successful, then a file3ds \* is returned, if the open fails, then a NULL is returned.

CloseAllFiles3ds

***Prototype:*** void CloseAllFiles3ds()

Closes all .3ds files and deletes any temporary ones. The memory allocated to existing file3ds pointers is freed at this time.

***Important:*** If any databases are still open when this function is called, they are left in an unknown state and should be released at the \*first opportunity.

No arguments:

Returns nothing.

Database3ds

3D Studio database and index

A collection of chunk references that constitute a .3ds file. The references may come from one or more files, or may exist entirely in memory. The database is queried and modified through other functions and should not be manipulated directly. The chunk3ds and the chunklist3ds structures are not documented in this reference as they are for internal toolkit use only.

typedef struct

{

      chunk3ds \*topchunk;

ubyte3ds objlistdirty;

ubyte3ds matlistdirty;

ubyte3ds nodelistdirty;

chunklist3ds \*objlist;

chunklist3ds \*matlist;

chunklist3ds \*nodelist;

} database3ds;

chunk3ds \*topchunk

The topmost chunk, usually the filetype defining chunk, that contains the remainder of the 3ds file.

ubyte3ds objlistdirty

True3ds (default) when a change to objects has been made, and a new objlist needs to be computed.

ubyte3ds matlistdirty

True3ds (default) when a change to the materials has been made and a new matlist needs to be computed.

ubyte3dsnodelistdirty

True3ds (default) when a change to the keyframe nodes has been made and a new nodelist needs to be calculated.

chunklist3ds \*objlist

A cross-reference of camera, light, and mesh names to their chunk records in the database. Used to speed up searches for objects by name or index.

chunklist3ds \*matlist

A cross-reference of material names to their chunk records in the database. Used to speed up searches for materials by name or index.

chunklist3ds \*nodelist

A cross-reference of camera, light, and mesh keyframe node names to their chunk records in the database. Used to speed up searches for nodes by name or index.

InitDatabase3ds

***Prototype:*** void InitDatabase3ds(database3ds \*\*db)

Initializes and optionally allocates memory for a database3ds structure. Initialization needs to be done before the structure can be used by other functions.

Arguments:

database3ds \*\*db

Address of the pointer to the database3ds structure. If the pointer referenced by the address is NULL, then memory is allocated for the database3ds structure, and the pointer is updated with its location.

Returns nothing.

ReleaseDatabase3ds

***Prototype:*** void ReleaseDatabase3ds(database3ds \*\*db);

Frees all memory used by the database structure and its fields.

Arguments:

database3ds \*\*db

Address of the pointer to the database3ds structure. After the memory is freed, then the pointer is reset to NULL.

Returns nothing.

CreateDatabase3ds

***Prototype:*** void CreateDatabase3ds(file3ds \*file, database3ds \*db);

Creates a database from an existing .3ds file. The database's type is set by the type of .3ds file referenced in \*file.

Arguments:

file3ds \*file

A file pointer previously created with read attributes by OpenFile3ds prior to use.

database3ds \*db

The database the file is being built for. The structure must be initialized using InitDatabase3ds prior to use.

Returns nothing.

CreateNewDatabase3ds

***Prototype:*** void CreateNewDatabase3ds(database3ds \*db, dbtype3ds type);

Prepares a new database.

Arguments:

database3ds \*db

The database that is being created new. The structure must be initialized by InitDatabase3ds prior to use.

dbtype3ds type

The type of database being created. Can be one of these values:

typedef enum{      Unknown, /\* Database has no type \*/      MeshFile, /\* Database is a .3DS file \*/      ProjectFile, /\* Database is a .PRJ file \*/      MaterialFile /\* Database is a .MLI file \*/} dbtype3ds;

Of the types listed above, Unknown is not accepted by the function.

Returns nothing.

GetDatabaseType3ds

***Prototype:*** dbtype3ds GetDatabaseType3ds(database3ds \*db);

Determines the type of file contained in the database.

Arguments:

database3ds \*db

The database being queried.

Returns one of the following:

typedef enum{      Unknown, /\* Database has no type or has not been created \*/      MeshFile, /\* Database is a .3DS file \*/      ProjectFile, /\* Database is a .PRJ file \*/      MaterialFile /\* Database is .MLI file \*/} dbtype3ds;

CopyDatabase3ds

***Prototype:*** void CopyDatabase3ds(database3ds \*idb, database3ds \*odb);

Creates a new database from an existing one.

Arguments:

database3ds \*idb

The source database.

database3ds \*odb

The destination database. Must be initialized by InitDatabase3ds prior to use in this function.

Returns nothing.

DisconnectDatabase3ds

***Prototype:*** void DisconnectDatabase3ds(database3ds \*db)

Loads the contents of an entire database completely into local memory and disconnects the database from any associated files. This lets a database survive beyond the CloseAllFiles3ds function.

Arguments:

database3ds \*db

The database to disconnect.

Returns nothing.

WriteDatabase3ds

***Prototype:*** void WriteDatabase3ds(file3ds \*file, database3ds \*db);

Creates a new file from the contents of a database, or rewrites a file opened using OpenFile3ds with file attribute \*attrib = "w".

Arguments:

file3ds \*file

A file pointer previously created with write attributes by OpenFile3ds. After the database is successfully written, its file handle is closed by the toolkit. If the filename is the same as another file opened for reading, the existing, old file is copied to a temporary file so that any existing databases may continue to reference the contents of the old file.

database3ds \*db

The database being written into the file.

Returns nothing.

DumpDatabase3ds

***Prototype:*** void DumpDatabase3ds(FILE \*outfile, database3ds \*db);

Creates a human-readable report of the chunk contents of a database.

Arguments:

FILE \*outfile

A standard file I/O handle to which the report is sent. May be used with the system-defined stdout or stderr file handles without difficulty.

database3ds \*db

The database to be dumped.

NameListEntry3ds

List Entry

Stores the name of an object, material, or keyframe node.

typedef struct {

char3ds \*name;

} namelistentry3ds;

char3ds \*name

Name of object, material, or keyframe node.

When a name list is declared and then used by one of the Get\_\_\_NameList3ds functions, InitNameList3ds is called by the Get function, and does not have to be called separately.

NameList3ds

List Of Names

A list of names for objects, materials or keyframe nodes used to aid searches of the database by name.

typedef struct {

ulong3ds count;

      ulong3ds spaces

namelistentry3ds \*list;

} namelist3ds;

ulong3ds count

Number of entries used in the list.

ulong3ds spaces

Number of entries allocated to the list. This number maybe larger than the number of entries actually used.

namelistentry3ds \*list

A list of names.

InitNameList3ds

***Prototype:*** void InitNameList3ds(namelist3ds \*\*list, long3ds count)

Initializes and allocates memory for the namelist3ds structure.

Arguments:

namelist3ds \*\*list

Address for a pointer to a structure. If the pointer is NULL, then memory is allocated for the structure and its address is stored in the pointer. If memory is already allocated to the namlistentry \*list field, then the existing contents are released before the new memory is allocated.

long3ds count

The minimum number of namelistentry3ds structures to allocate for the list. For efficiency, more entries than requested may be allocated. If initialized, it initializes to the number requested.

Returns nothing.

ReleaseNameList3ds

***Prototype:*** void ReleaseNameList3ds(namelist3ds \*\*list)

Frees memory allocated to a namelist3ds structure.

Arguments:

namelist3ds \*\*list

The list being freed. After the memory is released, a NULL is stored in the pointer.

Returns nothing.

Runtime Error Reporting

Errorid3ds

Runtime error codes generated during the operation of the toolkit.

typedef enum {

      NO\_FTK\_ERRORS,

      ERR\_NO\_MEM,

      ERR\_INVALID\_ARG,

      ERR\_INVALID\_DATA,

  ERR\_INVALID\_CHUNK,

      ERR\_INVALID\_DATABASE,

      ERR\_WRONG\_DATABASE,

      ERR\_UNFOUND\_CHUNK,

      ERR\_WRONG\_OBJECT,

  ERR\_NO\_SUCH\_FILE,

  ERR\_INIT\_FAILED,

  ERR\_OPENING\_FILE,

  ERR\_CLOSING\_FILE,

  ERR\_READING\_FILE,

  ERR\_CREATING\_DATABASE,

  ERR\_READING\_DATABASE,

  ERR\_WRITING\_DATABASE,

  ERR\_WRITING\_FILE,

      ERR\_NO\_STRING\_TOO\_LONG,

  ERR\_TOO\_MANY\_FILES,

  N\_ERRORS

} errorid3ds

NO\_FTK\_ERRORS

There are no file toolkit error pending.

ERR\_NO\_MEM

Not enough memory to complete operation.

ERR\_INVALID\_ARG

The argument passed to the function was invalid. Usually caused by a NULL pointer where or an out of range numeric value.

ERR\_INVALID\_DATA

A structure passed as an argument to the function has bad or incorrect data.

ERR\_INVALID\_CHUNK

An incorrect chunk structure was encountered while reading the database. Usually caused by a corrupt database or file.

ERR\_INVALID\_DATABASE

The database passed as an argument has not been created.

ERR\_WRONG\_DATABASE

The database passed as an argument is the wrong type of database for this function.

ERR\_UNFOUND\_CHUNK

The database is missing important file chunks needed to fill out the data structure. Usually caused by a corrupt database or file.

ERR\_WRONG\_OBJECT

The name passed to the functions exists, but is not the type of object asked for. For example, asking for a mesh object with the GetCameraByName3ds function.

ERR\_NO\_SUCH\_FILE

The file asked for cannot be found.

ERR\_INIT\_FAILED

Failure to initialize structure passed as an argument.

ERR\_OPENING\_FILE

Could not open requested file.

ERR\_CLOSING\_FILE

Could not close requested file.

ERR\_READING\_FILE

Error occurred while reading file associated with database.

ERR\_CREATING\_DATABASE

Error occurred while creating database.

ERR\_READING\_DATABASE

Error occurred while reading database.

ERR\_WRITING\_DATABASE

Error occurred while writing database.

ERR\_WRITING\_FILE

Error occurred writing file.

ERR\_STRING\_TOO\_LONG

String encountered in file, structure, or as an argument was longer than expected. Possibly caused by uninitialized pointer, corrupt file, or database.

ERR\_TOO\_MANY\_FILES

The toolkit has reached its maximum open file limit of 250 files.

N\_ERRORS

There were more errors than could be recorded in the error list.

ErrRec3ds

An error list element.

typedef struct {

errorid3ds id;

const char3ds \*desc;

} ErrRec3ds;

errorid3ds id

The error code of the error.

const char3ds \*desc

The text description of the error message.

Ftkerr3ds

extern byte3ds ftkerr3ds

Global error status. True3ds if there are errors pending. Defaults to False3ds.

Ignoreftkerr3ds

extern byte3ds ignoreftkerr3ds

Ignore error flag. Set to True3ds to cause toolkit to ignore all runtime errors. Errors will continue to be recorded, but they will not cause the toolkit functions to abort. This flag is provided as a work-around for a bad .3ds file or if potentially spurious error messages are received. Defaults to False3ds.

DumpErrList3ds

***Prototype:*** void DumpErrList3ds(FILE \*outfile)

Creates a human-readable report of the last errors encountered.

Arguments:

FILE \*outfile

A standard file I/O handle to which the report is sent. May be used with the system-defined stdout or stderr file handles without difficulty.

Returns nothing.

ClearErrList3ds

***Prototype:*** void ClearErrList3ds()

Clears any pending errors from the error list.

No arguments and returns nothing.

ReturnErrList3ds

***Prototype:*** const ErrRec3ds \*ReturnErrList3ds()

Returns an address to the error list.

No arguments.

Returns **const ErrRec3ds \*** which is a list of error records. The end of the list is terminated with a NO\_FTK\_ERRORS ID. Pointer is to global data and should not be freed by application.

PRINT\_ERRORS\_EXIT

***Description:*** #define PRINT\_ERRORS\_EXIT(file) {if(ftkerr3ds){DumpErrStack3ds(file); if (!ignoreftkerror3ds) exit(-1);}}

A useful macro for testing error status and exiting with a report if there are errors.

Arguments:

FILE \*file

A standard file I/O handle to which the report is sent. May be used with the system-defined stdout or stderr file handles without difficulty.

Mesh Settings

ShadowSets3ds

Global shadow settings

Scene wide shadow settings used in .3ds and .prj files. Corresponds to the 3D Editor/Renderer/Shadows/Global Shadow Control dialog.

typedef struct {

      shadowstyle3ds type;

float3ds bias;

float3ds raybias;

short3ds mapsize;

      float3ds filter;

} shadowsets3ds;

shadowstyle3ds type

Type of shadow to cast, can be one of the following:

typedef enum {      UseShadowMap, /\* Use shadow maps \*/      UseRayTraceShadow /\* Use ray tracing \*/} shadowstyle3ds;

Defaults to UseShadowMap. Corresponds to the Shadow Maps/Ray Trace buttons in the dialog.

float3ds bias

Mapped shadow bias. Valid range is from 0.0 to 99999.0. Defaults to 1.0. Corresponds to the Map Bias field in the dialog.

float3ds raybias

Ray traced shadow bias. Valid range is from 0.0 to 99999.0. Defaults to 1.0. Corresponds to the Ray trace bias field in the dialog

short3ds mapsize

Mapped shadow resolution. Valid range is from 10 to 9999. Defaults to 512. Corresponds to the Map size field in the dialog.

float3ds filter

Mapped shadow filter. Valid range is from 1.0 to 1.0. Defaults to 3.0. Corresponds to the Map sample range field in the dialog.

MeshSet3ds

Global mesh settings

Miscellaneous settings used in .3ds and .prj files.

typedef struct {

      ulong3ds version;

      float3ds masterscale;

shadowsets3ds shadow;

fcolor3ds ambientlight;

      point3ds oconsts;

} meshset3ds;

ulong3ds version

File version. Current version is version 3.

float3ds masterscale

The units-to-inch scale of the objects in the file, a value of 1.0 equals 1 inch per unit. Useful if multiple files with different scales are to be created and merged in 3D Studio. When 3D Studio reads this value, it uses it as a reference to scale the .*3ds* file into whatever units it is using currently. Valid range is from (but not including) 0.0 to maximum allowable for type float3ds. Defaults to 1.0. Roughly corresponds to the Unit Setup option on the Views pull-down menu.

shadowsets3ds shadow

The global shadow settings used in the scene. Corresponds to the 3D Editor/Renderer/Shadows dialog.

fcolor3ds ambientlight

Ambient light color for the scene. Corresponds to the 3D Editor/Lights/Ambient dialog.

point3ds oconsts

The object construction plane used when creating objects in the 3D Editor but has no direct effect on a scene. Corresponds to the 3D Editor/Display/Const menu.

InitMeshSet3ds

***Prototype:*** void InitMeshSet3ds(meshset3ds \*\*mset)

Initializes and optionally allocates memory for the meshset3ds structure.

Arguments:

meshset3ds \*\*sets

The address of a pointer to the meshset3ds structure. If the referenced pointer is NULL then memory is allocated and the new address is put in its place.

Returns nothing.

ReleaseMeshSet3ds

***Prototype:*** void ReleaseMeshSet3ds(meshset3ds \*\*mset)

Frees the memory used by a meshset3ds structure.

Arguments:

meshset3ds \*\*sets

The address of the pointer to the structure. After the structure is freed, then a NULL value is put in its place.

Returns nothing.

GetMeshSet3ds

***Prototype:*** void GetMeshSet3ds(database3ds \*db, meshset3ds \*\*mset)

Gets the settings contained in the database and copies them into a meshset3ds structure. Only databases for .3ds and .prj file contain usable information.

Arguments:

database3ds \*db

The database that is being queried.

meshset3ds \*\*sets

The address of a pointer to the meshset3ds structure. If the referenced pointer is NULL, then a call to InitMeshSet3ds is performed prior to filling out the structure with the new settings.

Returns nothing.

PutMeshSet3ds

***Prototype:*** void PutMeshSet3ds(database3ds \*db, meshset3ds \*sets)

Puts the settings contained in the meshset3ds structure into the database. Overwriting any existing settings with the new ones.

Arguments:

database3ds \*db

The database being written into. The database must be a dbtype3ds of MeshFile or ProjectFile.

meshset3ds \*sets

The settings being written.

Returns nothing.

CopyMeshSet3ds

***Prototype:*** void CopyMeshSet3ds(database3ds \*destdb, database3ds \*srcdb)

Shallow copies the mesh settings from one database into another. Existing mesh settings in the destination database are replaced with the new copy.

Arguments:

database3ds \*destdb

The database to receive the copied settings.

databse3ds \*srcdb

The database containing the original settings.

Returns nothing.

Atmosphere Settings

Fogsettings3ds

Fog Atmosphere Settings

Atmospheric fog settings that correspond to the 3D Editor/Renderer/Setup/Atmosphere/Fog Definition dialog.

typedef struct

{

float3ds nearplane;

float3ds neardensity;

float3ds farplane;

float3ds fardensity;

fcolor3ds fogcolor;

byte3ds fogbgnd;

} fogsettings3ds;

float3ds nearplane

Fogging radius in units from camera to start of fogging effect. Valid range is from 0.0 to the maximum allowable value for type float3ds. Defaults to 0.0. Roughly corresponds to 3D Editor/Cameras/Ranges menu item.

float3ds neardensity

Fogging percentage at the start of the fogging effect. Valid range is from 0.0 (no fogging) to 100.0 (maximum fogging). Defaults to 0.0. Corresponds to the Near  % field in the Fog Definition dialog.

float3ds farplane

Fogging radius in units from camera to end of fogging effect. Valid range is from 0.0 to the maximum allowable value for type float3ds. Defaults to 1000.0. Roughly corresponds to 3D Editor/Cameras/Ranges menu item.

float3ds fardensity

Fogging percentage at the end of the fogging effect. Valid range is from 0.0 (no fogging) to 100.0 (maximum fogging). Defaults to 0.0. Corresponds to the Far  % field in the Fog Definition dialog.

fcolor3ds fogcolor

The fog color. Corresponds to the color sliders in the Fog Definition dialog. Defaults to white (1.0, 1.0, 1.0).

byte3ds fogbgnd

True3ds if the fog effect is applied to the scene background. Defaults to True3ds.

Layerfogsettings3ds

Layer Fog Atmosphere Settings

Atmospheric layer fog settings that corresponds to the 3D Editor/Renderer/Settings/Atmosphere/Layer Fog Definition dialog.

typedef struct{

      float3ds zmin;

      float3ds zmax;

      float3ds density;

fcolor3ds fogcolor;

layerfogfalloff3ds falloff;

byte3ds fogbgnd;

} layerfogsettings3ds;

float3ds zmin

The lower Z elevation of the fog effect. Valid range is from minimum to maximum allowable values for type float3ds. Default is 0.0. Corresponds to the Bottom field in the dialog.

float3ds zmax

The upper Z elevation of the fog effect. Valid range is from minimum to maximum allowable values for type float3ds. Default is 100.0. Corresponds to the Top field in the dialog.

float3ds density

The fogging density percentage. Valid range is from 0.0 to 100.0. Default is 50.0. Corresponds to the Density field in the dialog.

fcolor3ds fogcolor

The fog color. Defaults to white (1.0, 1.0, 1.0). Corresponds to the color sliders in the dialog.

layerflogfalloff3ds falloff

The falloff direction of the effect. One of these values:

typedef enum {      NoFall, /\* No falloff \*/      TopFall, /\* Lowest density at zmax, highest at zmin \*/      BottomFall /\* Lowest density at zmin, highest at zmax \*/} layerfogfalloff3ds;

Default is NoFall. Corresponds to Top/Bottom/None buttons in dialog.

byte3ds fogbgnd

True3ds to fog the background. Default is True3ds. Corresponds to Fog Background button in the dialog.

Dcuesettings3ds

Distance Cue Atmosphere Settings

Atmosphere distance cue settings that correspond to the 3D Editor/Renderer/Setup/Atmosphere/Distance-Cueing dialog.

typedef struct

{

float3ds nearplane;

float3ds neardim;

float3ds farplane;

float3ds fardim;

byte3ds dcuebgnd;

} dcuesettings3ds;

float3ds nearplane

Dimming radius from camera to beginning of the effect. Valid range is 0.0 to maximum allowable value for type float3ds. Default is 0.0.

float3ds neardim

Dimming percentage at the start of the dimming effect. Valid range is from 0.0 to 100.0. Default is 0.0. Corresponds to the Near  % field in the dialog.

float3ds farplane

Dimming radius from camera to end of the effect. Valid range is 0.0 to maximum allowable value for type float3ds. Default is 1000.0.

float3ds fardim

Dimming percentage at the end off the dimming effect. Valid range is from 0.0 to 100.0. Default is 100.0. Corresponds to the Far  % field in the dialog.

byte3ds dcuebgnd

True3ds if the effect is applied to the background. Default is True3ds. Corresponds to the Dim Background button in the dialog.

Atmosphere3ds

Atmosphere Settings

Atmospheric settings that correspond to the 3D Editor/Renderer/Setup/Atmosphere Definition dialog.

typedef struct

{

fogsettings3ds fog;

layerfogsettings3ds layerfog;

dcuesettings3ds dcue;

atmospheretype3ds activeatmo;

} atmosphere3ds;

fogsettings3ds fog

Fog settings that correspond to the Fog Definition dialog.

layerfogsettings3ds layerfog

Layer fog settings that correspond to the Layer Fog Definition dialog.

dcuesettings dcue

Distance Cue settings that correspond to the Distance-Cue dialog.

atmospheretype3ds activeatmo

The atmosphere effect to use in the scene. One of the following values:

typedef enum {      NoAtmo, /\* No atmosphere effects. \*/      UseFog, /\* Using fog effect \*/      UseLayerFog, /\* Using layer fog effect \*/      UseDistanceCue /\* Using distance cue effect \*/} atmospheretype3ds;

Default is NoAtmo. Corresponds the highlighted Fog, Distance Cue, Layered Fog, or None buttons in the dialog.

InitAtmosphere3ds

***Prototype:*** void InitAtmosphere3ds(atmosphere3ds \*\*atmo)

Initializes and optionally allocates memory for the atmosphere3ds structure.

Arguments:

atmosphere3ds \*\*atmo

Address of a pointer to an atmosphere3ds structure. If the pointer is NULL, then memory is allocated for the new structure, and its address is stored in the pointer.

Returns nothing.

ReleaseAtmosphere3ds

***Prototype:*** void ReleaseAtmosphere3ds(atmosphere3ds \*\*atmo)

Frees memory allocated for the atmosphere3ds structure.

Arguments:

atmosphere3ds \*\*atmo

Address of a pointer to an atmosphere3ds structure. After the pointer is freed, a NULL is stored in it.

Returns nothing.

GetAtmosphere3ds

***Prototype:*** void GetAtmosphere3ds(database3ds \*db, atmosphere3ds \*\*atmo)

Copies the atmospheric effect settings from the database into the atmosphere3ds structure. Only databases of the type MeshFile or ProjectFile contain atmosphere settings.

Arguments:

database3ds \*db

The database that is searched for settings information. Should be type MeshFile or ProjectFile.

atmosphere3ds \*\*atmo

Address of a pointer to the atmosphere3ds structure that receives the settings. If the pointer is NULL, then InitAtmosphere3ds is called to create a new structure.

Returns nothing.

PutAtmosphere3ds

***Prototype:*** void PutAtmosphere3ds(database3ds \*db, atmosphere3ds \*atmo);

Copies the atmospheric effect settings from the atmosphere3ds structure into the database. Existing atmosphere settings in the database, if any, are replaced.

Arguments:

database3ds \*db

The database that receives the new settings. Must be type MeshFile or ProjectFile.

atmosphere3ds \*atmo

The atmosphere3ds structure that contains the new settings.

Returns nothing.

CopyAtmosphere3ds

***Prototype:*** void CopyBackground3ds(database3ds \*destdb, database3ds \*srcdb)

Shallow copies the atmospheric effect settings from one database into another. Existing settings in the destination database are replaced with the new copy.

Arguments:

database3ds \*destdb

The database to receive the copied settings.

databse3ds \*srcdb

The database containing the original settings.

Returns nothing.

Background Settings

Bitmapbgnd3ds

Background Bitmap Setting

Bitmap background setting that corresponds to the 3D Editor/Renderer/Background/Bitmap dialog.

typedef struct

{

char3ds \*name;

} bitmapbgnd3ds;

char3ds \*name

The filename of the bitmap being used for a background. Default is NULL.

Solidbgnd3ds

Solid Color Background Setting

Solid color background setting that corresponds to the 3D Editor/Renderer/Background/Solid Color slider dialog.

typedef struct

{

fcolor3ds color;

} solidbgnd3ds;

fcolor3ds color

The color of the solid background. Default is black (0.0, 0.0, 0.0).

Vgradientbgnd3ds

Vertical Color Gradient Background Settings

Vertical gradient background setting that corresponds to the 3D Editor/Renderer/Background/Define Gradient Colors dialog.

typedef struct

{

float3ds gradpercent;

fcolor3ds top;

      fcolor3ds mid;

fcolor3ds bottom;

} vgradientbgnd3ds;

float3ds gradpercent

Position of mid color position. Ranges from 0.0 for the top of the screen to 1.0 for the bottom of the screen. Default is 0.5. Roughly corresponds to the sliding arrow gadget in the dialog.

fcolor3ds top

Color of the gradient top. Default is black (0.0, 0.0, 0.0). Roughly corresponds to the top color swatch in the dialog.

fcolor3ds mid

Color of the gradient middle. Default is black (0.0, 0.0, 0.0). Roughly corresponds to the middle color swatch in the dialog.

fcolor3ds bottom

Color of the gradient bottom. Default is black (0.0, 0.0, 0.0). Roughly corresponds to the bottom color swatch in the dialog.

Background3ds

Background Settings

Background style settings that correspond to the 3D Editor/Renderer/Background Method dialog.

typedef struct {

bitmapbgnd3ds bitmap;

solidbgnd3ds solid;

vgradientbgnd3ds vgradient;

backgroundtype3ds bgndused;

} background3ds;

bitmapbgnd3ds bitmap

Bitmap background settings. Correspond to the Bitmap button in the dialog.

solidbgnd3ds solid

Solid color background settings. Correspond to the Solid Color button in the dialog.

vgradientbgnd3ds vgradient

Vertical color gradient background settings. Correspond to the Gradient button in the dialog.

backgroundtype3ds bgndused

The style of background used in the rendering scene. Can be one of the following values:

typedef enum {      NoBgnd = 0, /\* No background \*/      UseSolidBgnd = 1, /\* Use a solid background \*/      UseVGradientBgnd = 2, /\* Use a gradient background \*/      UseBitmapBgnd = 3 /\* Use a bitmapped background \*/} backgroundtype3ds;

The default setting is NoBgnd. Corresponds to the highlighted Solid Color, Gradient, Bitmap, or None button in the dialog.

InitBackground3ds

***Prototype:*** void InitBackground3ds(background3ds \*\*bgnd)

Initializes and optionally allocates memory for the background3ds structure.

Arguments:

background3ds \*\*bgnd

An address of a pointer to a background3ds structure. If the pointer is NULL, then memory for a new structure is allocated and its address is stored in the pointer.

Returns nothing.

ReleaseBackground3ds

***Prototype:*** void ReleaseBackground3ds(background3ds \*\*bgnd)

Frees the memory allocated to a background3ds structure and its fields.

Arguments:

background3ds \*\*bgnd

Address of a pointer to a background3ds structure. After the structure is freed, the pointer is set to NULL.

Returns nothing.

GetBackground3ds

***Prototype:*** void GetBackground3ds(database3ds \*db, background3ds \*\*bgnd)

Searches the database for background settings and copies the settings into a background3ds structure. Only MeshFile and ProjectFile databases have background settings.

Arguments:

database3ds \*db

Database to search for settings. Should be type MeshFile or ProjectFile.

background3ds \*\*bgnd

The background3ds information that receives the background settings in the database. Prior to being filled in, the structure is initialized by InitBackground3ds.

Returns nothing.

PutBackground3ds

***Prototype:*** void PutBackground3ds(database3ds \*db, background3ds \*bgnd)

Copies the background settings in the background3ds structure into the database. Any existing settings in the database are replaced.

Arguments:

database3ds \*db

Database to receive the new settings. Must be type MeshFile or ProjectFile.

background3ds \*bgnd

Structure containing the background settings.

Returns nothing.

CopyBackground3ds

***Prototype:*** void CopyBackground3ds(database3ds \*destdb, database3ds \*srcdb)

Shallow copies the background settings from one database into another. Existing settings in the destination database are replaced with the new copy.

Arguments:

database3ds \*destdb

The database to receive the copied settings.

databse3ds \*srcdb

The database containing the original settings.

Returns nothing.

Viewport Settings

Viewsize3ds

Viewport size definition.

The viewport size need not match the current 3D Studio state. If it doesn't, then 3D Studio will scale the viewport to fit in the case of ortho or user views, or recalculate the viewport from the camera definition.

typedef struct {

ushort3ds xpos;

ushort3ds ypos;

ushort3ds width;

ushort3ds height;

} viewsize3ds;

ushort3ds xpos

The X axis position in pixels of the upper-left viewport corner.

ushort3ds ypos

The Y axis position in pixels of the upper-left viewport corner.

ushort3ds width

The width of the viewport in pixels.

usrhot3ds height

The height of the viewport in pixels.

Orthoview3ds

Orthographic view definition.

typedef struct {

point3ds center;

float3ds zoom;

} orthoview3ds;

point3ds center

The scene coordinates that appear in the center of the viewport. Defaults to 0.0, 0.0, 0.0.

float3ds zoom

The ratio of pixels to geometry units in the viewport. Defaults to 0.7.

UserView3ds

User view settings

typedef struct {

point3ds center; /\* Center of user view \*/

float3ds zoom; /\* View zoom factor \*/

float3ds horang; /\* Horizontal angle of view \*/

float3ds verang; /\* Veritical angle of view \*/

} userview3ds;

point3ds center

The scene coordinates that appear in the center of the viewport. Defaults to 0.0, 0.0, 0.0.

float3ds zoom

The ratio of pixels to geometry units in the viewport. Defaults to 0.7.

float3ds horang

The view angle in degrees in the XZ axis. Defaults to 30.0

float3ds verant

The view angle in degrees in the YZ axis. Defaults to 20.0 degrees.

Cameraview3ds

Camera view settings.

typedef struct {

char3ds name[11];

} cameraview3ds;

char3ds name[11]

Ten-character camera name on which to base the view. Must be the name of a defined camera.

Viewport3ds

Single viewport definition.

typedef struct {

viewtype3ds type;

orthoview3ds ortho;

userview3ds user;

cameraview3ds camera;

} viewport3ds;

viewtype3ds type

The type of view being defined. Can be one of the following values:

typedef enum {      TopView3ds, /\* Create a top viewport \*/      BottomView3ds, /\* Create a bottom viewport \*/      LeftView3ds, /\* Create a left viewport \*/      RightView3ds, /\* Create a right viewport \*/      FrontView3ds, /\* Create a front viewport \*/      BackView3ds, /\* Create a back viewport \*/      UserView3ds, /\* Create a user viewport \*/      CameraView3ds, /\* Create a camera viewport \*/      SpotlightView3ds /\* Create a spotlight viewport \*/} viewtype3ds;

The default is FrontView3ds.

InitViewport3ds

***Prototype:*** void InitViewport3ds(viewport3ds \*\*vp)

Initializes and optionally allocates memory for the viewport3ds structure.

Arguments:

viewport3ds \*\*vp

The address of a pointer to the viewport3ds structure. If the referenced pointer is omni light then memory is allocated and the new address is put in its place.

Returns nothing.

ReleaseViewport3ds

***Prototype:*** void ReleaseViewport3ds(viewport3ds \*\*vp)

Frees the memory used by a viewport3ds structure.

Arguments:

viewport3ds \*\*vp

The address of the pointer to the structure. After the structure is freed, then a NULL value is put in its place.

Returns nothing.

GetViewport3ds

***Prototype:*** void GetViewport3ds(database3ds \*db, viewport3ds \*\*vp)

Gets the active viewport defined for the Keyframer and it into a viewport3ds structure. Only databases for .3ds and .prj file contain usable information.

Arguments:

database3ds \*db

The database that is being queried.

viewport3ds \*\*vp

The address of a pointer to the viewport3ds structure. If the referenced pointer is NULL, then a call to InitViewport3ds is performed prior to filling out the structure with the viewport.

Returns nothing.

PutViewport3ds

***Prototype:*** void PutViewport3ds(database3ds \*db, viewport3ds \*vp)

Puts the settings contained in the viewport3ds structure into the database. Overwriting any existing settings with the new ones.

Arguments:

database3ds \*db

The database being written into. The database must be a dbtype3ds of MeshFile or ProjectFile.

viewport3ds \*vp

The settings being written.

Returns nothing.

CopyViewport3ds

***Prototype:*** void CopyViewport3ds(database3ds \*destdb, database3ds \*srcdb)

Shallow copies the viewport settings from one database into another. Existing viewport settings in the destination database are replaced with the new copy. Viewport layouts and configurations of any style are copied.

Arguments:

database3ds \*destdb

The database to receive the copied settings.

databse3ds \*srcdb

The database containing the original settings.

Returns nothing.

Material Definitions

Acubic3ds

Automatic Cubic Reflection Map Settings

Automatic cubic and flat reflection map settings that correspond to the Materials/Reflection/Automatic Reflection Map dialog.

typedef struct {

      ubyte3ds firstframe;

      ubyte3ds flat;

ulong3ds size;

ulong3ds nthframe;

} acubic3ds;

ubyte3ds firstframe

True3ds if the automatic reflection map should only be rendered for the first frame. Defaults to False3ds. Corresponds to the First Frame Only button in the dialog.

ubyte3ds flat

True3ds if reflection map is flat mirror, False3ds if cubic reflection. Default is False3ds. Corresponds to the Flat Mirror buttons in the dialog.

ulong3ds size

The resolution of the reflection map. Defaults to 100. Valid range is from 1 to 999999. Corresponds to the Size field in the dialog.

ulong3ds nthframe

The nth frame frequency of reflection rendering. Valid range is from 1 to 32000. Corresponds to the Every Nth Frame field in the dialog.

Bitmap3ds

Material Bitmap Settings

The bitmap settings used for the many different map styles such as textures, opacity, specular, and so on.

typedef struct {

char3ds name[13];

float3ds percent;

tiletype3ds tiling;

byte3ds ignorealpha;

filtertype3ds filter;

float3ds blur;

byte3ds mirror;

byte3ds negative;

float3ds uscale, vscale;

float3ds uoffset, voffset;

float3ds rotation;

tinttype3ds source;

fcolor3ds tint1;

fcolor3ds tint2;

fcolor3ds redtint;

fcolor3ds greentint;

fcolor3ds bluetint;

ulong3ds datasize;

void3ds \*data;

} bitmap3ds;

char3ds name[13]

The bitmap filename. Defaults to an empty string. Corresponds to the filename shown in the Map or Mask columns in the Materials editor.

float3ds percent

The bitmap strength. Valid range is from 0.0 (weakest effect) to 1.0 (maximum effect). Defaults to 0.0. Roughly corresponds to the Amount slider.

tiletype3ds tiling

The tiling style used with the bitmap. Set to one of the following:

typedef enum {    TileMap = 1, /\* Repetitively tile bitmap \*/    DecalMap = 2, /\* Repeat bitmap only once \*/                              /\* with transparent key color \*/    TileDecalMap = 3 /\* Tile and decal the map \*/} tiletype3ds;

The default settings is TileMap. Corresponds to the Tile/Decal/Both radio buttons in the Mapping Parameters dialog.

ubyte3ds ignorealpha

True3ds to ignore a map's alpha channel when decaling. Defaults to False3ds. Corresponds to the Ignore Map Alpha button in the Mapping Parameters dialog.

filtertype3ds filter

Filtering style used on the bitmap. Is one of the following:

typedef enum {      Pyramidal, /\* Pyramidal filtering \*/      SummedArea /\* Summed area filtering \*/} filtertype3ds;

Default is Pyramidal. Corresponds to the Filtering buttons located in the Mapping Parameters dialog.

float3ds blur

Texture map blurring strength. Valid range is 0.0 (no blurring) to 1.0 (maximum blurring). Roughly corresponds to the Blur slider in the Mapping Parameters dialog.

byte3ds mirror

True3ds to mirror the bitmap orientation. Default is False3ds. Corresponds to the Mirror button in the Mapping Parameters dialog.

byte3ds negative

True3ds to negate the bitmaps color space. Default is False3ds. Corresponds to the Negative button in the Mapping Parameters dialog.

float3ds uscale, vscale

Bitmap coordinate rescaling factors. Default to (1.0, 1.0). Correspond to the U Scale and V Scale fields of the Mapping Parameters dialog.

float3ds uoffset, voffset

Bitmap coordinate offset values. Default to (0.0, 0.0). Correspond to the U Offset and V Offset fields of the Mapping Parameters dialog.

float3ds rotation

Bitmap rotation offset. Defaults to 0.0. Corresponds to the Rotation Angle field of the Mapping Parameters dialog box.

tinttype3ds source

Map image source. Can be one of the following:

typedef enum {      RGB, /\* Bitmap's RGB color channels \*/      Alpha, /\* Bitmap's alpha channel \*/      RGBLumaTint, /\* Tinted RGB luminance channel \*/      AlphaTint, /\* Tinted alpha channel \*/      RGBTint /\* Each RBG channel individually tinted \*/} tinttype3ds;

Default is RGB. Not all options are valid for all uses of bitmap. Roughly corresponds to the Source radio buttons in the Mapping Parameters dialog.

fcolor3ds tint1

Start of tinting color ramp used with RGBLumaTint and AlphaTint sources. Default is black (0.0, 0.0, 0.0). Corresponds to the first tinting color swatch in the Mapping Parameters dialog.

fcolor3ds tint2

End of tinting color ramp used with RGBLumaTint and RGBAlphaTint sources. Default is white (1.0, 1.0, 1.0). Corresponds to the second tinting color swatch in the Mapping Parameter dialog.

fcolor3ds redtint

Red channel tinting color used with RGBTint source. Default is red (1.0, 0.0, 0.0). Corresponds to the R color swatch in the Mapping Parameter dialog.

fcolor3ds greentint

Green channel tinting color used with RGBTint source. Default is green (0.0, 1.0, 0.0). Corresponds to the G color swatch in the Mapping Parameter dialog.

fcolor3ds bluetint

Blue channel tinting color used with RGBTint source. Default is blue (0.0, 0.0, 1.0). Corresponds to the B color swatch in the Mapping Parameter dialog.

ulong3ds datasize

The size, in bytes, of the SXP plug in data contained in \*data. Default is 0.

void \*data

SXP plug-in state data. Default is NULL.

Mapset3ds

Map and Mask Settings

Combined pair of map and mask settings.

typedef struct {

bitmap3ds map;

bitmap3ds mask;

} mapset3ds;

bitmap3ds map

The map settings.

bitmap3ds mask

The mask settings.

Rmapset3ds

Reflection Map and Mask Settings

typedef struct {

bitmap3ds map;

byte3ds useauto;

acubic3ds automap;

bitmap3ds mask;

} rmapset3ds;

bitmap3ds map

Reflection map settings. Only the name and percent fields of the bitmap3ds structure are used in reflection maps.

byte3ds useauto

True3ds for automatic reflection mapping. Defaults to False3ds.

acubic3ds automap

Automatic reflection map settings.

bitmap3ds mask

Reflection mask settings.

Material3ds

Material Settings

typedef struct {

char3ds name[17];

fcolor3ds ambient;

      fcolor3ds diffuse;

      fcolor3ds specular;

      float3ds shininess;

float3ds shinstrength;

float3ds blur;

float3ds transparency;

float3ds transfalloff;

float3ds selfillumpct;

float3ds wiresize;

shadetype3ds shading;

byte3ds useblur;

byte3ds usefall;

byte3ds twosided;

byte3ds selfillum;

byte3ds decal;

byte3ds additive;

byte3ds usewire;

  byte3ds usewireabs;

byte3ds facemap;

byte3ds soften;

mapset3ds texture;

mapset3ds texture2;

mapset3ds opacity;

bmapset3ds bump;

mapset3ds specmap;

mapset3ds shinmap;

mapset3ds illummap;

rmapset3ds reflect;

} material3ds;

char3ds name[17]

Material name. Defaults to empty string. Corresponds to the Current Material field in the editor.

fcolor3ds ambient

Material ambient light color. Defaults to black (0.0, 0.0, 0.0). Corresponds to the Ambient color swatch in the editor.

fcolor3ds diffuse

Material diffuse light color. Defaults to black (0.0, 0.0, 0.0). Corresponds to the Diffuse color swatch in the editor.

fcolor3ds specular

Material specular highlight color. Defaults to black (0.0, 0.0, 0.0). Corresponds to the Specular color swatch in the editor.

float3ds shininess

Material shininess. Valid range is from 0.0 to 1.0. Defaults to 0.0. Roughly corresponds to the Shininess slider in the editor.

float3ds shinstrength

Material shininess strength. Valid range is from 0.0 to 1.0. Defaults to 0.0. Roughly corresponds to the Shin. Strength slider.

float3ds blur

Reflection blur. Valid range is from 0.0 to 1.0. Defaults to 0.0. Roughly corresponds to the Reflect.Blur slider in the editor.

float3ds transparency

Material transparency. Valid range is from 0.0 to 1.0. Defaults to 0.0. Roughly corresponds to the Transparency slider in the editor.

float3ds transfalloff

Transparency falloff factor. Valid range is from -1.0 to 1.0. Defaults to 0.0. Roughly corresponds to the Trans. Falloff slider with negative values being equivalent to the In button and positive equivalent to the Out button.

float3ds selfillumpct

Self illumination settings. Valid ranges is from 0.0 to 1.0. Defaults to 0.0. Roughly corresponds to the Self Illum. slider.

float3ds wiresize

Wireframe wire width. Valid ranges is from, but not including, 0.0, to maximum allowable value for type float3ds. Defaults to 1.0. Corresponds to the Thickness field in the Wire Frame Mode dialog.

shadetype3ds shading

The shading level used by the material. Can be one of the following:

typedef enum {      WireShading = 0, /\* Wire frame shading. No longer used in R3 \*/      FlatShading = 1, /\* Flat shading \*/      GouraudShading = 2, /\* Gouraud shading \*/      PhongShading = 3, /\* Phong shading \*/      MetalShading = 4 /\* Metal shading \*/} shadetype3ds;

Default is PhongShading. Corresponds to Flat/Gouraud/Phong/Metal buttons in the editor.

byte3ds useblur

True3ds uses reflection blur. Default is False3ds.

byte3ds usefall

True3ds uses transparency falloff. Default is False3ds.

byte3ds twosided

True3ds uses two-sided rendering. Default is False3ds. Corresponds to 2-Sided button in editor.

byte3ds selfillum

True3ds uses self-illumination. Default is False3ds.

byte3ds additive

True3ds uses additive transparency. Default is False3ds. Corresponds to the Sub/Add buttons in the editor.

byte3ds usewire

True3ds uses wireframe. Default is False3ds. Corresponds to the Wire Frame On button in the Wire Frame Mode dialog.

byte3ds usewireabs

True3ds to measure wireframe width in scene units. Default is False3ds. Corresponds to the Units switch in the Wire Frame Mode dialog.

byte3ds facemap

True3ds uses face mapping. Default is False3ds. Corresponds to the Face Map button in the editor.

byte3ds soften

True3ds uses Phong softening when Phong shading is specified, but does not affect any other type of shading. Defaults is True3ds. Corresponds to the Soften button in the editor. If False3ds, phong shading is not softened.

mapset3ds texture

Texture 1 map settings.

mapset3ds texture2

Texture 2 map settings.

mapset3ds opacity

Opacity map settings.

mapset3ds bump

Bump map settings.

mapset3ds specmap

Specular map settings.

mapset3ds shinmap

Shininess map settings.

mapset3dsillummap

Self Illum map settings.

rmapset3dsreflect

Reflection map settings.

InitMaterial3ds

***Prototype:*** void InitMaterial3ds(material3ds \*\*mat)

Initializes and optionally allocates memory for a material3ds structure.

Arguments:

material3ds \*\*mat

Address of a pointer to a material3ds structure. If pointer is NULL, then memory for a new material3ds structure is allocated and the new address is stored in the pointer.

Returns nothing.

ReleaseMaterial3ds

***Prototype:*** void ReleaseMaterial3ds(material3ds \*\*mat)

Frees memory allocated to a material3ds structure.

Arguments:

material3ds \*\*mat

Address of a pointer to a material3ds structure. After the pointer is freed, a NULL is stored in its place.

Returns nothing.

GetMaterialCount3ds

***Prototype:*** ulong3ds GetMaterialCount3ds(database3ds \*db)

Scans the database and counts the number of material entries it contains.

Arguments:

database3ds \*db

The database to scan.

Returns **ulong3ds** as the total count of material definitions contained in the database.

GetMaterialNameList3ds

***Prototype:*** void GetMaterialNameList3ds(database3ds \*db, namelist3ds \*\*list)

Scans the database for all material names and stores them in a namelist3ds structure.

Arguments:

database3ds \*db

The database to scan.

namelist3ds \*\*list

Address of a pointer to a name list. Prior to filling out the structure, the pointer is passed to InitNameList3ds for initialization and allocation. InitNameList3ds need not be called explicitly.

Returns nothing.

GetMaterialByIndex3ds

***Prototype:*** void GetMaterialByIndex3ds(database3ds \*db, ulong3ds index, material3ds \*\*mat)

Copies the settings of the index(th) material in the database into a material3ds structure. If the index is out of range, than no change is made to the material structure.

Arguments:

database3ds \*db

The database to search.

ulong3ds index

The index of the material in the database. Valid range is from 0 (the first one) to less than the result of GetMaterialCount3ds as applied to \*db.

material3ds \*\*mat

The material structure that receives the settings. Prior to storing the new settings, the pointer is first passed to InitMaterial3ds for initialization and allocation.

Returns nothing.

GetMaterialByName3ds

***Prototype:*** void GetMaterialByName3ds(database3ds \*db, char3ds \*name, material3ds \*\*mat)

Copies the settings for a material with a matching name from the database into the structure. If no match is made, then the structure is left unchanged.

Arguments:

database3ds \*db

Database to be searched.

char3ds \*name

Name of the material. Search is case sensitive.

material3ds \*\*mat

The material that gets the settings. The structure is passed to InitMaterial3ds prior to use.

Returns nothing.

PutMaterial3ds

***Prototype:*** void PutMaterial3ds(database3ds \*db, material3ds \*mat)

Copies the settings from a material structure into a database. Materials in the database with the same name are replaced.

Arguments:

database3ds \*db

The database to store the material in.

material3ds \*mat

The material being added to the database.

Returns nothing.

CopyMaterialByName3ds

***Prototype:*** void CopyMaterialByName3ds(database3ds \*destdb, database3ds \*srcdb, char3ds \*name)

Shallow copies a material from one database into another. Objects in the destination database with the same name as the copied material are replaced.

Arguments:

database3ds \*destdb

The database to receive the copied material.

databse3ds \*srcdb

The database containing the original material.

char3ds \*name

Name of the material to copy. Search is case-sensitive.

Returns nothing.

CopyMaterialByIndex3ds

***Prototype:*** void CopyMaterialByIndex3ds(database3ds \*destdb, database3ds \*srcdb, ulong3ds index)

Shallow copies a material from one database into another. Objects in the destination database with the same name as the copied material are replaced.

Arguments:

database3ds \*destdb

The database to receive the copied material.

databse3ds \*srcdb

The database containing the original material.

ulong3ds index

The index of the material in the source database. Valid range is from 0 (the first one) to one less than the result of GetMaterialCount3ds as applied to \*srcdb.

Returns nothing.

DeleteMaterialByName3ds

***Prototype:*** void DeleteMaterialByName3ds(database3ds \*db, char3ds \*name)

Deletes a material from the database.

Arguments:

database3ds \*db

Database containing the material to delete.

char3ds \*name

Name of the material to delete. Search is case-sensitive.

Returns nothing.

DeleteMaterialByIndex3ds

***Prototype:*** void DeleteMaterialByIndex3ds(database3ds \*db, ulong3ds index)

Deletes a material from the database.

Arguments:

database3ds \*db

Database containing the material to delete.

ulong3ds index

The index of the material in the database. Valid range is from 0 (the first one) to one less than the result of GetMaterialCount3ds as applied to \*db.

Returns nothing.

Mesh Objects

Textvert3ds

Texture map coordinate points

U,V coordinate points are used to determine which portion of a map (texture, opacity, or bump) is rendered on the surface of a face. For every map used, the U,V coordinate for the upper-left corner is 0,0, and the lower-right is slightly less than 1,1 (to enable seamless tiling). U,V values larger than 1 or smaller than 0 are tiled by their fractional portion if the assigned material is not a decal.

typedef struct

{

float3ds u, v;

} textvert3ds;

float3ds u

The horizontal coordinate.

float3ds v

The vertical coordinate.

Face3ds

Face definition and appearance

***Prototype:*** face3ds structure

typedef struct

{

ushort3ds v1, v2, v3, flag;

} face3ds;

ushort3ds v1

Index location of first vertex constructing the face. Valid range 0–65535, but should only reference indices in the vertex list.

ushort3ds v2

Index location of second vertex constructing the face. Valid range 0–65535.

ushort3ds v3

Index location of third vertex constructing the face. Valid range 0–65535.

ushort3ds flag

A set of bits that defines the visibility of the face edges and the wrapping texture coordinates.

static const ushort3ds FaceCAVisable3ds = 0x0001;Flags the edge from v1 to v2 as visible

static const ushort3ds FaceBCVisable3ds = 0x0002;Flags the edge from v2 to v3 as visible.

static const ushort3ds FaceABVisable3ds = 0x0004; Flags the edge from v3 to v1 as visible.

static const ushort3ds FaceUWrap3ds = 0x0008; Flags the face as being at a texture coord u wrap seam.

static const ushort3ds FaceVWrap3ds = 0x0010;Flags the face as being at a texture coord v wrap seam.

Note: Remaining bits should be off.

Mapinfo3ds

Texture-mapping information for the 3D Studio Editor Surface/Mapping/Adjust/Acquire command

typedef struct {

ushort3ds maptype;

float3ds tilex;

float3ds tiley;

float3ds cenx;

float3ds ceny;

float3ds cenz;

float3ds scale;

float3ds matrix[12];

float3ds pw;

float3ds ph;

float3ds ch;

} mapinfo3ds;

ushort3ds maptype

Values for mapping icon selection: 0 for planar icon, 1 for cylindrical icon, 2 for spherical icon. Defaults to 0.

float3ds tilex, tiley

Texture tiling values for horizontal and vertical directions.

float3ds cenx, ceny, cenz

3D location of the map in global coordinates.

float3ds scale

Used to scale equally the size of the planar icon, the radius of the cylindrical icon, and the radius of the spherical icon in global units.

float3ds matrix[12]

3-by-4 transformation matrix for icon orientation.

float3ds pw

Planar icon width.

float3ds ph

Planar icon height.

float3ds ch

Cylindrical icon height.

Objmat3ds

Material to face assignment definition

typedef struct {

char3ds name[17];

ushort3ds nfaces;

ushort3ds \*faceindex;

} objmat3ds;

char3ds name[17]

The name of the material being assigned to the faces. Material must be a valid material name. Defaults to an empty string.

ushort3ds nfaces

The number of faces being assigned the material. Ranges from 0 to the number of faces defined in the mesh object. Defaults to 0.

ushort3ds \*faceindex

An array of face index numbers that can range from 0 to number of faces defined in the mesh object. An array of objmat3ds should not reference the same face twice.

Mesh3ds

Mesh object definition

typedef struct {

char3ds name[11];

byte3ds ishidden;

byte3ds isvislofter;

      byte3ds isvisshaper;

byte3ds ismatte;

byte3ds isnocast;

byte3ds isfast;

      byte3ds isnorcvshad;

      byte3ds isfrozen;

ushort3ds nvertices;

point3ds \*vertexarray;

      ushort3ds nvflags;

ushort3ds \*vflagarray;

      ushort3ds ntextverts;

      textvert3ds \*textarray;

      byte3ds usemapinfo;

mapinfo3ds map;

float3ds locmatrix[12];

ushort3ds nfaces;

face3ds \*facearray;

ulong3ds \*smootharray;

      byte3ds useboxmap;

      ubyte3ds meshcolor;

char3ds boxmap[6][17];

      ushort3ds nmats;

objmat3ds \*matarray;

      byte3ds useproc;

ulong3ds procsize;

char3ds procname[13];

void3ds \*procdata;

} mesh3ds;

char3ds name[11]

Mesh object name of 1–10 characters. Should be unique and non-empty. Defaults to an empty string.

byte3ds ishidden

True3ds to hide the object from display in the 3D Editor and rendering, False3ds (default) to make object visible.

byte3ds isvislofter

True3ds to make the object visible in the 3D Lofter, False3ds (default) to hide it.

byte3ds isvisshaper

True3ds to make the object visible in the 2D Shaper, False3ds (default) to hide it.

byte3ds ismatte

True3ds to make the object a background matting object, False3ds (default) to make it renderable.

byte3ds isnocast

True3ds to prevent the object from casting shadows, a False3ds (default) to make it cast shadows.

byte3ds isfast

True3ds to make the object displayed in fast mode in the 3D Editor, False3ds (default) for complete display.

byte3ds isnorcvshad

True3ds to prevent the object from receiving shadows, False3ds (default) to make it shadowable.

byte3ds isfrozen

True3ds to make the object displayable but not editable, False3ds (default) to make it editable.

ushort3ds nvertices

The number of vertices from which the mesh object is constructed. Valid range is 0–65535. Should be greater than 0. Defaults to 0.

point3ds\*vertexarray

An array of 3D points. Element count must be the same as nvertex. The array can have a maximum of 65535 elements. Must be a valid array. Defaults to NULL.

ushort3ds nvflags

The number of vertex flags used in the mesh object. Valid range is 0 to the value of nvertices. Not required. Defaults to 0.

ushort3ds \*vflagarray

An array of vertex flags. Element count must be the same as nvflags. The array can have a maximum of 65535 elements. Not required. Defaults to NULL.

ushort3ds ntextverts

The number of texture coordinates used in the mesh object. Valid range is 0 to the value of nvertices. Not required. Defaults to 0.

textvert3ds\*textarray

An array of texture coordinates. Element count must be the same as nvertices. The array can have a maximum of 65535 elements. Not required. Defaults to NULL.

byte3ds usemapinfo

True3ds if texture map icon information is associated with the mesh, False3ds (default) if none is available.

mapinfo3ds map

Texture map icon position. Defaults to planar map at (0.0, 0.0, 0.0)

float3ds locmatrix[12]

3-by-4 transformation matrix for object orientation. Only used to record the rotation and translation operations preformed on the vertices of the mesh. Defaults to identity at (0.0, 0.0, 0.0).

ushort3ds nfaces

The number of faces from which the mesh object is constructed. Valid range is 0 to 65535. Must be greater than 0. Defaults to 0.

face3ds \*facearray

An array of faces. Element count must be the same as nfaces. The array can have a maximum of 65535 faces defined. Must be a valid array. Defaults to NULL.

ulong3ds\*smootharray

An array of smoothing group assignments. Element count must be the same as nfaces. The smoothing groups are assigned as bits, with each bit (0–31) representing one smoothing group. The following constants can be used to assign the bits:

static const ulong3ds Smooth01Group3ds = 0x00000001;static const ulong3ds Smooth02Group3ds = 0x00000002;static const ulong3ds Smooth03Group3ds = 0x00000004;static const ulong3ds Smooth04Group3ds = 0x00000008;static const ulong3ds Smooth05Group3ds = 0x00000010;static const ulong3ds Smooth06Group3ds = 0x00000020;static const ulong3ds Smooth07Group3ds = 0x00000030;static const ulong3ds Smooth08Group3ds = 0x00000080;static const ulong3ds Smooth09Group3ds = 0x00000100;static const ulong3ds Smooth10Group3ds = 0x00000200;static const ulong3ds Smooth11Group3ds = 0x00000400;static const ulong3ds Smooth12Group3ds = 0x00000800;static const ulong3ds Smooth13Group3ds = 0x00001000;static const ulong3ds Smooth14Group3ds = 0x00002000;static const ulong3ds Smooth15Group3ds = 0x00004000;static const ulong3ds Smooth16Group3ds = 0x00008000;static const ulong3ds Smooth17Group3ds = 0x00010000;static const ulong3ds Smooth18Group3ds = 0x00020000;static const ulong3ds Smooth19Group3ds = 0x00040000;static const ulong3ds Smooth20Group3ds = 0x00080000;static const ulong3ds Smooth21Group3ds = 0x00100000;static const ulong3ds Smooth22Group3ds = 0x00200000;static const ulong3ds Smooth23Group3ds = 0x00400000;static const ulong3ds Smooth24Group3ds = 0x00800000;static const ulong3ds Smooth25Group3ds = 0x01000000;static const ulong3ds Smooth26Group3ds = 0x02000000;static const ulong3ds Smooth27Group3ds = 0x04000000;static const ulong3ds Smooth28Group3ds = 0x08000000;static const ulong3ds Smooth29Group3ds = 0x10000000;static const ulong3ds Smooth30Group3ds = 0x20000000;static const ulong3ds Smooth31Group3ds = 0x40000000;static const ulong3ds Smooth32Group3ds = 0x80000000;

The flags cannot be combined because only one smoothing group per face is allowed. Not required. Defaults to NULL.

byte3ds useboxmap

True3ds if "box" material assignment is used by the object, False3ds (default) if face assignment is being used.

char3ds boxmap [6] [17]

The material names box assigned to the object. The assignment is as follows:

boxmap[0] is front sideboxmap[1] is back sideboxmap[2] is left sideboxmap[3] is right sideboxmap[4] is top sideboxmap[5] is bottom side

Valid material names must be used if useboxmap is True3ds. Names default to empty strings.

ubyte3ds meshcolor

UI color for mesh object. Valid range is from 0 to 63. Defaults to 0.

ushort3ds nmats

The number of materials assigned to the mesh object. Valid range is from 0 to 255. Not required, defaults to 0.

objmat3ds\*matarray

An array of material-to-face assignment lists. Element count must be the same as nmats. Not required. Defaults to NULL

byte3ds useproc

True3ds to activate the AXP plug-in. Defaults to False3ds.

ulong3ds procsize

The number of bytes of AXP plug-in data. Not required. Defaults to 0.

char3ds procname

The filename of the AXP plug-in. Not required. Defaults to an empty string.

void \*procdata

The data used by the AXP plug-in. Must be the size given in procsize. Not required. Defaults to NULL.

InitMeshObj3ds

***Prototype:*** ushort3ds InitMeshObj3ds(mesh3ds \*\*obj, ushort3ds nvertices, ushort3ds nfaces, ushort3ds initflags)

Initializes and allocates memory for a mesh3ds structure.

Arguments:

mesh3ds \*\*obj

Address of a pointer to a mesh3ds structure. If pointer is NULL, then memory for the structure is allocated and its address is stored in the pointer.

ushort3ds nvertices

The number of vertices in the mesh. Valid range is 0–65537. If number is nonzero, then a point3ds array with nvertices elements is allocated and its pointer stored in the vertexarray field.

ushot3ds nfaces

The number of faces in the mesh. Valid range is 0–65537. If number is nonzero, then a face3ds array with nfaces elements is allocated and its pointer stored in the facearray field.

ushort3ds initflags

Flags for optional allocation of other mesh3ds fields. Can be any combination of the following:

static const ushort3ds InitNoExtras3ds = 0x0000;Do not allocate any extra fields.

static const ushort3ds InitTextArray3ds = 0x0002;Allocate the textarray field.

static const ushort3ds InitSmoothArray3ds = 0x0010;Allocate the smootharray field.

static const ushort3ds InitVFlagArray3ds = 0x0040;Allocate the vflagarray field.

RelMeshObj3ds

***Prototype:*** void RelMeshObj3ds(mesh3ds \*\*obj)

Frees the memory used by a mesh3ds structure.

Arguments:

mesh3ds \*\*obj

The structure to free up. After the structure is freed, then a NULL is stored in the pointer.

Returns nothing.

GetMeshCount3ds

***Prototype:*** ulong3ds GetMeshCount3ds(database3ds \*db)

Scans the database and returns a count of the number of mesh objects in the database.

Arguments:

database3ds \*db

The database to scan.

Returns a **ulong3ds** as the number of mesh objects in the database.

GetMeshNameList3ds

***Prototype:*** void GetMeshNameList3ds(database3ds \*db, namelist3ds \*\*list)

Scans the database for all mesh names and stores them in a namelist3ds structure.

Arguments:

database3ds \*db

The database to scan.

namelist3ds \*\*list

Address of a pointer to a name list. Prior to filling out the structure, the pointer is passed to InitNameList3ds for initialization and allocation.

Returns nothing.

GetMeshByIndex3ds

***Prototype:*** void GetMeshByIndex3ds(database3ds \*db, ulong3ds index, mesh3ds \*\*mesh);

Copies the settings of the index(th) mesh in the database into a mesh3ds structure. If the index is out of range, than no change is made to the mesh structure.

Arguments:

database3ds \*db

The database to search.

ulong3ds index

The index of the mesh in the database. Valid range is from 0 (the first one) to one less than the result of GetMeshCount3ds as applied to \*db.

mesh3ds \*\*mesh

The mesh structure that receives the settings. Prior to storing the new settings, the pointer is first passed to InitMesh3ds for initialization and allocation.

Returns nothing.

GetMeshByName3ds

***Prototype:*** void GetMeshByName3ds(database3ds \*db, char3ds \*name, mesh \*\*mesh)

Copies the settings for a mesh with a matching name from the database into the structure. If no match is made, then the structure is left unchanged.

Arguments:

database3ds \*db

Database to be searched.

char3ds \*name

Name of the mesh. Search is case sensitive.

mesh3ds \*\*mesh

The mesh that gets the settings. The structure is passed to InitMesh3ds prior to use.

Returns nothing.

PutMesh3ds

***Prototype:*** void PutMesh3ds(database3ds \*db, mesh3ds \*mesh)

Copies the settings from a mesh structure into a database. Objects in the database with the same names as the new mesh are replaced.

Arguments:

database3ds \*db

The database in which to store the mesh.

mesh3ds \*mesh

The mesh being added to the database.

Returns nothing.

CopyMeshByName3ds

***Prototype:*** void CopyMeshByName3ds(database3ds \*destdb, database3ds \*srcdb, char3ds \*name)

Shallow copies a mesh from one database into another. Objects in the destination database with the same name as the copied mesh are replaced.

Arguments:

database3ds \*destdb

The database to receive the copied mesh.

databse3ds \*srcdb

The database containing the original mesh.

char3ds \*name

Name of the mesh to copy. Search is case-sensitive.

Returns nothing.

CopyMeshByIndex3ds

***Prototype:*** void CopyMeshByIndex3ds(database3ds \*destdb, database3ds \*srcdb, ulong3ds index)

Shallow copies a mesh from one database into another. Objects in the destination database with the same name as the copied mesh are replaced.

Arguments:

database3ds \*destdb

The database to receive the copied mesh.

databse3ds \*srcdb

The database containing the original mesh.

ulong3ds index

The index of the mesh in the source database. Valid range is from 0 (the first one) to one less than the result of GetMeshCount3ds as applied to \*srcdb.

Returns nothing.

DeleteMeshByName3ds

***Prototype:*** void DeleteMeshByName3ds(database3ds \*db, char3ds \*name)

Deletes a mesh from the database.

Arguments:

database3ds \*db

Database containing the mesh to delete.

char3ds \*name

Name of the mesh to delete. Search is case-sensitive.

Returns nothing.

DeleteMeshByIndex3ds

***Prototype:*** void DeleteMeshByIndex3ds(database3ds \*db, ulong3ds index)

Deletes a mesh from the database.

Arguments:

database3ds \*db

Database containing the mesh to delete.

ulong3ds index

The index of the mesh in the database. Valid range is from 0 (the first one) to one less than the result of GetMeshCount3ds as applied to \*db.

Returns nothing.

Omni Lights and Spotlights

Spotshadow3ds

Spotlight Shadow Settings

The local shadow spotlight settings used in a spotlight that roughly corresponds to the 3D Editor/Lights/Spot/Spotlight Definition dialog.

typedef struct {

byte3ds cast;

shadowstyle3ds type;

byte3ds local;

float3ds bias;

float3ds filter;

ushort3ds mapsize;

float3ds raybias;

} spotshadow3ds;

byte3ds cast

True3ds if the spotlight casts shadows. Default is False3ds. Corresponds to the Cast Shadows button in the dialog.

shadowstyle3ds type

Type of shadow to cast, can be one of the following:

typedef enum {      UseShadowMap, /\* Use shadow maps \*/      UseRayTraceShadow /\* Use ray tracing \*/} shadowstyle3ds;

Defaults to UseShadowMap. Corresponds to the Shadow Maps/Ray Trace buttons in the Local Shadow Control dialog.

byte3ds local

True3ds if local shadow settings are used, False3ds if global settings (from meshset3ds structure) are used. Default is False3ds. Corresponds to the Use Global Settings button in the Local Shadow Control dialog.

float3ds bias

Mapped shadow bias. Valid range is 0.0–99999.0. Defaults to 1.0. Corresponds to the Map Bias field in the Local Shadow Control dialog.

float3ds raybias

Raytraced shadow bias. Valid range is 0.0–99999.0. Defaults to 1.0. Corresponds to the Ray Trace Bias field in the Local Shadow Control dialog

short3ds mapsize

Mapped shadow resolution. Valid range is 10–9999. Defaults to 512. Corresponds to the Map Size field in the Local Shadow Control dialog.

float3ds filter

Mapped shadow filter. Valid range is 1.0–99999.0. Defaults to 3.0. Corresponds to the Map Sample Range field in the Local Shadow Control dialog.

SpotCone3ds

Spotlight Cone Related Settings

typedef struct {

conestyle3ds type;

      float3ds aspect;

byte3ds show;

byte3ds overshoot;

} spotcone3ds;

conestyle type

Shape of light cone used. One of the following values:

typedef enum {      Circular, /\* Circle shaped spot \*/      Rectangular /\* Rectangle shaped spot \*/} conestyle3ds;

Default is Circular. Corresponds to Rectangle and Circle buttons in the Spotlight Definition dialog.

float3ds aspect

Aspect ratio of width/height of a rectangular spotlight cone. Defaults to 1.0. Corresponds to the 3D Editor/Lights/Spot/Aspect menu branch.

byte3ds show

True3ds for spotlight to show cone on editor screens. Defaults to False3ds. Corresponds to the Show Cone button in the dialog.

byte3ds overshoot

True3ds for spotlight overshoot effect. Defaults to False3ds. Corresponds to the Overshoot button on the Spotlight Definition dialog.

Spotprojector3ds

Spotlight Projection Settings

Settings for spotlight projection of bitmaps.

typedef struct {

byte3ds use;

char3ds \*bitmap;

} spotprojector3ds;

byte3ds use

True3ds for using a projection map. Defaults to False3ds. Corresponds to the Projector button in the Spotlight Definition dialog.

char3ds \*bitmap

The name of the projection bitmap. Name must be 12 or fewer characters and free of file path information. Defaults to NULL.

Spotlight3ds

Spotlight Definition

typedef struct

{

point3ds target;

float3ds hotspot;

float3ds falloff;

float3ds roll;

spotshadow3ds shadows;

spotcone3ds cone;

spotprojector3ds projector;

} spotlight3ds;

point3ds target

The spotlight target. Defaults to 1.0, 1.0, 1.0.

float3ds hotspot

The spotlight hotspot angle. Valid range is 0.0–160.0. Defaults to 44.0

float3ds falloff

The spotlight falloff angle. Valid range is 0.0–160.0. Defaults to 45.0

float3ds roll

The spotlight roll angle. Valid range is 0.0–360.0. Defaults to 0.0.

spotshadow3ds shadows

Spotlight shadow settings.

spotcone3ds cone

Spotlight cone settings.

spotprojector3ds project

Spotlight projector settings.

Liteattenuate3ds

Light Attenuation Settings

typedef struct {

byte3ds on;

float3ds inner;

float3ds outer;

} liteattenuate3ds;

byte3ds on

True3ds for light attenuation. Defaults to False3ds. Roughly corresponds to the Attenuate button in the light definition dialogs.

float3ds inner

Radius of inner attenuation radius where light is at full brightness. Default is 10.0. Roughly corresponds to the 3D Editor/Lights/Omni.../Ranges menu branch.

float3ds outer

Radius of outer attenuation radius where light is at its minimum brightness. Default is 100.0. Roughly corresponds to the 3D Editor/Lights/Omni.../Ranges menu branch.

Light3ds

Omni and Spotlight Definition

Used for the definition of Omni lights and Spotlights.

typedef struct {

char3ds name[11];

point3ds pos;

fcolor3ds color;

float3ds multiplier;

byte3ds dloff;

liteattunate3ds attenuation;

liteexclude3ds exclude;

spotlight3ds \*spot;

} light3ds;

char3ds name[11]

A name for the light object with 10 or fewer characters. Default is NULL.

point3ds pos

The position of the light. Defaults to 0.0.

fcolor3ds color

The light color. Defaults to white (0.7, 0.7, 0.7).

float3ds multiplier

The light intensity multiplier. Defaults to 1.0.

byte3ds dloff

True3ds to turn the light off. Defaults to False3ds.

liteattunate3ds attenuation

The attenuation settings for the light.

namelist \*exclude

The excluded objects list.

spotlight3ds \*spot

For spotlights, the spotlight settings, otherwise pointer is NULL.

InitLight3ds

***Prototype:*** void InitLight3ds(light3ds \*\*light)

Initializes and optionally allocates memory for a light3ds structure.

Arguments:

light3ds \*\*light

Address of a pointer to a light3ds structure. If the pointer is NULL, then memory is allocated for the structure and its address is stored in the pointer.

Returns nothing.

InitSpotlight3ds

***Prototype:*** void InitSpotlight3ds(light3ds \*\*spotlight)

Initializes and optionally allocates memory for the light3ds structure and its \*spotfield.

Arguments:

light3ds \*\*spotlight

Address of a pointer to a light3ds structure. If the pointer is NULL, then memory is allocated for the structure and its address is stored in the pointer.

Returns nothing.

ReleaseLight3ds

***Prototype:*** void ReleaseLight3ds(light3ds \*\*light)

Frees memory used by the structure and its fields.

Arguments:

light3ds \*\*light

The structure to be freed. After release, NULL is stored in the pointer.

Return nothing.

GetOmnilightCount3ds

***Prototype:*** ulong3ds GetOmnilightCount3ds(database3ds \*db)

Scans the database for spotlight objects are returns a count of their number.

Arguments:

database3ds \*db

The database to scan for omni lights.

Returns a **ulong3ds** as the number of spotlights in the database.

GetSpotlightCount3ds

***Prototype:*** void GetSpotlightCount3ds(database3ds \*db)

Scans the database for spotlight objects and returns a count of their number.

Arguments:

database3ds \*db

The database to scan for spotlights.

Returns a **ulong3ds** as the number of spotlights in the database.

GetOmnilightNameList3ds

***Prototype:*** void GetOmnilightNameList3ds(database3ds \*db, namelist3ds \*\*list)

Scans the database for omni light objects and returns a list of their names.

Arguments:

database3ds \*db

The database to scan for omni lights.

namelist3ds \*\*list

The list of omni light names found. Pointer gets passed to InitNameList3ds prior to use.

Returns nothing.

GetSpotlightNameList3ds

***Prototype:*** void GetSpotlightNameList3ds(database3ds \*db, namelist3ds \*\*list)

Scans the database for spotlight objects and returns a list of their names.

Arguments:

database3ds \*db

The database to scan for spotlights.

namelist3ds \*\*list

The list of spotlight names found. Pointer gets passed to InitNameList3ds prior to use.

Returns nothing.

GetSpotlightByIndex3ds

***Prototype:*** void GetSpotlightByIndex3ds(database3ds \*db, ulong3ds index, light3ds \*\*spotlight)

Copies the settings of the index(th) spotlight in the database into a spotlight structure. If the index is out of range, then no change is made to the structure.

Arguments:

database3ds \*db

The database being searched.

ulong3ds index

The index of the spotlight in the database. Valid range is from 0 (the first spotlight) to one less the count returned by GetSpotlightCount3ds.

light3ds \*\*spotlight

The light structure that receives the settings. The pointer is passed to InitSpotlight3ds prior to use.

Returns nothing.

GetOmnilightByIndex3ds

***Prototype:*** void GetOmnilightByIndex3ds(database3ds \*db, ulong3ds index, light3ds \*\*light)

Copies the settings of index(th) omni light in the database into a omni light structure. If the index is out of range, then no change is made to the structure.

Arguments:

database3ds \*db

The database being searched.

ulong3ds index

The index of the omni light in the database. Valid range is from 0 (the first omni light) to one less the count returned by GetOmnilightCount3ds.

light3ds \*light

The light structure that receives the settings. The pointer is passed to InitOmnilight3ds prior to use.

Returns nothing.

GetSpotlightByName3ds

***Prototype:*** void GetSpotlightByName3ds(database3ds \*db, char3ds \*name, light3ds \*\*spotlight)

Searches the database for a spotlight with the given name and copies the spotlight's settings into the light3ds structure. If the name is not found, then no change is made to the structure.

Arguments:

database3ds \*db

The database being searched.

char3ds \*name

The light name of 1–10 characters to search for.

light3ds \*\*spotlight

The spotlight structure that receives the settings. The pointer is passed to InitSpotlight3ds prior to use.

Returns nothing.

GetOmnilightByName3ds

***Prototype:*** void GetOmnilightByName3ds(database3ds \*db, char3ds \*name, light3ds \*\*light)

Searches the database for a omni light of the given name and copies the light's settings into a light3ds structure. If the name is not found, then the structure remains unchanged.

Arguments:

database3ds \*db

The database being searched.

char3ds \*name

The light name of 1–10 characters to search for.

light3ds \*\*light

The light structure that receives the settings. The pointer is passed to InitOmnilight3ds prior to use.

Returns nothing.

PutSpotlight3ds

***Prototype:*** void PutSpotlight3ds(database3ds \*db, light3ds \*spotlight)

Copies the spotlight settings from the structure into the database. Lights in the database with the same name are replaced with the new settings.

Arguments:

database3ds \*db

The database to copy the spotlight into.

light3ds \*spotlight

The spotlight being added to the database.

Returns nothing.

PutOmnilight3ds

***Prototype:*** void PutOmnilight3ds(database3ds \*db, light3ds \*light)

Copies the omni light settings from the structure into the database. Lights in the database with the same name are replaced with the new settings.

Arguments:

database3ds \*db

The database to copy the omni light into.

light3ds \*light

The omni light being added to the database.

Returns nothing.

CopySpotlightByName3ds

***Prototype:*** void CopySpotlightByName3ds(database3ds \*destdb, database3ds \*srcdb, char3ds \*name)

Shallow copies a spotlight from one database into another. Objects in the destination database with the same name as the copied spotlight are replaced.

Arguments:

database3ds \*destdb

The database to receive the copied spotlight.

databse3ds \*srcdb

The database containing the original spotlight.

char3ds \*name

Name of the spotlight to copy. Search is case-sensitive.

Returns nothing.

CopySpotlightByIndex3ds

***Prototype:*** void CopySpotlightByIndex3ds(database3ds \*destdb, database3ds \*srcdb, ulong3ds index)

Shallow copies a spotlight from one database into another. Objects in the destination database with the same name as the copied spotlight are replaced.

Arguments:

database3ds \*destdb

The database to receive the copied spotlight.

databse3ds \*srcdb

The database containing the original spotlight.

ulong3ds index

The index of the spotlight in the source database. Valid range is from 0 (the first one) to one less than the result of GetSpotlightCount3ds as applied to \*srcdb.

Returns nothing.

CopyOmnilightByName3ds

***Prototype:*** void CopyOmnilightByName3ds(database3ds \*destdb, database3ds \*srcdb, char3ds \*name)

Shallow copies an omni light from one database into another. Objects in the destination database with the same name as the copied omni light are replaced.

Arguments:

database3ds \*destdb

The database to receive the copied omni light.

databse3ds \*srcdb

The database containing the original omni light.

char3ds \*name

Name of the omni light to copy. Search is case-sensitive.

Returns nothing.

CopyOmnilightByIndex3ds

***Prototype:*** void CopyOmnilightByIndex3ds(database3ds \*destdb, database3ds \*srcdb, ulong3ds index)

Shallow copies an omni light from one database into another. Objects in the destination database with the same name as the copied omni light are replaced.

Arguments:

database3ds \*destdb

The database to receive the copied omni light.

databse3ds \*srcdb

The database containing the original omni light.

ulong3ds index

The index of the omni light in the source database. Valid range is from 0 (the first one) to one less than the result of GetOmnilightCount3ds as applied to \*srcdb.

Returns nothing.

DeleteSpotlightByIndex3ds

***Prototype:*** void DeleteSpotlightByIndex3ds(database3ds \*db, long3ds index)

Deletes a spotlight from the database.

Arguments:

database3ds \*db

Database containing the spotlight to delete.

ulong3ds index

The index of the spotlight in the database. Valid range is from 0 (the first one) to one less than the result of GetSpotlightCount3ds as applied to \*db.

Returns nothing.

DeleteOmnilightByIndex3ds

***Prototype:*** void DeleteOmnilightByIndex3ds(database3ds \*db, long3ds index)

Deletes an omni light from the database.

Arguments:

database3ds \*db

Database containing the omni light to delete.

ulong3ds index

The index of the omni light in the database. Valid range is from 0 (the first one) to one less than the result of GetOmnilightCount3ds as applied to \*db.

Returns nothing.

DeleteSpotlightByName3ds

***Prototype:*** void DeleteSpotlightByName3ds(database3ds \*db, char3ds \*name)

Deletes a spotlight from the database.

Arguments:

database3ds \*db

Database containing the spotlight to delete.

char3ds \*name

Name of the light to delete. Search is case-sensitive.

Returns nothing.

DeleteOmnilightByName3ds

***Prototype:*** void DeleteOmnilightByName3ds(database3ds \*db, char3ds \*name);

Deletes an omni light from the database.

Arguments:

database3ds \*db

Database containing the omni light to delete.

char3ds \*name

Name of the omni light to delete. Search is case-sensitive.

Returns nothing.

Cameras

Camrange3ds

Camera Range Settings

Camera range settings used by atmospherics to determine radius of effect. Roughly corresponds to the 3D Editor/Cameras/Ranges menu branch.

typedef struct {

      float3ds cam\_near;

float3ds cam\_far;

} camranges3ds;

float3ds cam\_near

The near radius from camera position for the start of an atmospheric effect. Defaults to 0.0.

float3ds cam\_far

The far radius from camera position for the end of an atmospheric effect. Defaults to 1000.0.

Camera3ds

Camera Definition

Camera definition structure. Roughly corresponds to the 3D Editor/Cameras menu branch.

typedef struct

{

char3ds name[11];

point3ds position;

point3ds target;

float3ds roll;

float3ds fov;

byte3ds showcone;

camranges3ds range;

} camera3ds;

char3ds name[11]

The name of the camera. Defaults to empty string.

point3ds position

The location of the camera "body". Defaults to 0.0, 0.0, 0.0.

point3ds target

The location of the camera view target. Defaults to 1.0, 1.0, 1.0.

float3ds roll

The angle of camera roll in degrees. Valid range is 0–360.0. Defaults to 0.0

float3ds fov

The Field of View (FOV) in degrees. Valid range is 0.005–175.0. Defaults to 48.0.

byte3ds showcone

True3ds for a visible view cone. Defaults to False3ds.

camranges3ds range

Atmospheric effect ranges.

InitCamera3ds

***Prototype:*** void InitCamera3ds(camera3ds \*\*cam)

Initializes and optionally allocates memory for the camera3ds structure.

Arguments:

camera3ds \*\*cam

Address of a pointer to a camera3ds structure. If the pointer is NULL, then memory is allocated for a new structure and its address is stored in the pointer.

Returns nothing.

ReleaseCamera3ds

***Prototype:*** void ReleaseCamera3ds(camera3ds \*\*cam)

Frees all memory allocated to the camera3ds structure.

Arguments:

camera3ds \*\*cam

The camera structure to be freed. After release, the pointer is set to NULL

Returns nothing.

GetCameraCount3ds

***Prototype:*** ulong3ds GetCameraCount3ds(database3ds \*db)

Scans the database looking for cameras and returns a count of their number.

Arguments:

database3ds \*db

The database to search.

Returns a **ulong3ds** as the number of camera in the database.

GetCameraNameList3ds

***Prototype:*** void GetCameraNameList3ds(database3ds \*db, namelist3ds \*\*list)

Scans the database looking for cameras and returns a list of their names.

Arguments:

database3ds \*db

The database to search.

namelist3ds \*\*list

The list of names. Pointer is prepared by InitNameList3ds prior to use.

Returns nothing.

GetCameraByIndex3ds

***Prototype:*** void GetCameraByIndex3ds(database3ds \*db, long3ds index, camera3ds \*\*cam)

Scans the database looking for the index(th) occurrence of a camera and copies its settings into a camera3ds structure. If the index is out of range, then the structure remains unchanged.

Arguments:

database3ds \*db

The database to search.

long3ds index

The index of occurrence. Valid range is from 0 (the first camera) to one less the count returned by GetCameraCount3ds.

camera3ds \*\*cam

The camera receiving the settings. The pointer is prepared by InitCamera3ds prior to use.

Returns nothing.

GetCameraByName3ds

***Prototype:*** void GetCameraByName3ds(database3ds \*db, char3ds \*name, camera3ds \*\*cam)

Scans the database looking for a camera with a matching name and copies the camera's settings into a camera3ds structure. If the name is not matched, then the structure remains unchanged.

Arguments:

database3ds \*db

The database to search.

char3ds \*name

The camera name of 1–10 characters to search for. The search is case-sensitive.

camera3ds \*\*cam

The camera receiving the settings. The pointer is prepared by InitCamera3ds prior to use.

Returns nothing.

PutCamera3ds

***Prototype:*** void PutCamera3ds(database3ds \*db, camera3ds \*cam)

Copies a camera into the database. If an object of the same name exists, it is replaced with the new camera settings.

Arguments:

database3ds \*db

The database receiving the new camera.

camera3ds \*cam

The camera to add to the database.

Returns nothing.

CopyCameraByName3ds

***Prototype:*** void CopyCameraByName3ds(database3ds \*destdb, database3ds \*srcdb, char3ds \*name)

Shallow copies a camera from one database into another. Objects in the destination database with the same name as the copied camera are replaced.

Arguments:

database3ds \*destdb

The database to receive the copied camera.

databse3ds \*srcdb

The database containing the original camera.

char3ds \*name

Name of the camera to copy. Search is case-sensitive.

Returns nothing.

CopyCameraByIndex3ds

***Prototype:*** void CopyCameraByIndex3ds(database3ds \*destdb, database3ds \*srcdb, ulong3ds index)

Shallow copies a camera from one database into another. Objects in the destination database with the same name as the copied camera are replaced.

Arguments:

database3ds \*destdb

The database to receive the copied camera.

databse3ds \*srcdb

The database containing the original camera.

ulong3ds index

The index of the camera in the source database. Valid range is from 0 (the first one) to one less than the result of GetCameraCount3ds as applied to \*srcdb.

Returns nothing.

DeleteCameraByName3ds

***Prototype:*** void DeleteCameraByName3ds(database3ds \*db, char3ds \*name)

Deletes a camera from the database.

Arguments:

database3ds \*db

Database containing the camera to delete.

char3ds \*name

Name of the camera to delete. Search is case-sensitive.

Returns nothing.

DeleteCameraByIndex3ds

***Prototype:*** void DeleteCameraByIndex3ds(database3ds \*db, ulong3ds index)

Deletes a camera from the database.

Arguments:

database3ds \*db

Database containing the camera to delete.

ulong3ds index

The index of the camera in the database. Valid range is from 0 (the first one) to one less than the result of GetCameraCount3ds as applied to \*db.

Returns nothing.

Common Keyframe Structures

Keyheader3ds

Key temporal and spline attributes

typedef struct

{

ulong3ds time;

ushort3ds rflags;

float3ds tension;

float3ds continuity;

float3ds bias;

float3ds easeto;

float3ds easefrom;

} keyheader3ds;

ulong3ds time

The frame number the key occurs in. Default is 0.

ushort3ds rflags

A set of flags that define which spline terms are in use.

static const ushort3ds KeyUsesNone3ds = 0x00;static const ushort3ds KeyUsesTension3ds = 0x01;static const ushort3ds KeyUsesCont3ds = 0x02;static const ushort3ds KeyUsesBias3ds = 0x04;static const ushort3ds KeyUsesEaseTo3ds = 0x08;static const ushort3ds KeyUsesEaseFrom3ds = 0x10;

Default is KeyUseNone3ds.

float3ds tension

Spline tension value from –1.0 to 1.0. Default is 0.0.

float3ds continuity

Spline continuity value from –1.0 to 1.0. Default is 0.0.

float3ds bias

Spline bias value from –1.0 to 1.0. Default is 0.0.

float3ds easeto

Key ease to value from 0.0 to 1.0. Default is 0.0.

float3ds easefrom

Key ease from value from 0.0 to 1.0. Default is 0.0.

Keyframe Settings

Kfkeyinfo3ds

General animation settings.

Animation settings for length and current frame

typedef struct {

  ulong3ds length;

      ulong3ds curframe;

} kfkeyinfo3ds;

ulong3ds length

Number of frames in the animation. Valid range is from 1 to 32000. Defaults to 30.

ulong3ds curframe

Current frame in the animation. Valid range is from 0 to 32000. Defaults to 0.

Kfsegment3ds

Animation segment settings.

typedef struct {

  byte3ds use;

  ulong3ds begin;

  ulong3ds end;

} kfsegment3ds;

byte3ds use

Use segment flag. True3ds adds segment to database. Defaults to False3ds.

ulong3ds begin

First frame of segment. Valid range is 0 to last frame in animation. Defaults to 0.

ulong3ds end

Last frame of segment. Valid range is 0 to last frame in animation. Defaults to 30.

KfSets3ds

Keyframe settings.

typedef struct

{

kfkeyinfo3ds anim;

kfsegment3ds seg;

} kfsets3ds;

kfkeyinfo3ds anim

Animation settings for length and current frame.

kfsegment3ds seg

Animation segment settings.

InitKeySets3ds

***Prototype:*** void InitKeySets3ds(kfsets3ds \*\*key)

Initializes and optionally allocates memory for a kfsets3ds structure.

Arguments:

kfsets3ds \*\*key

Address of a pointer to a kfsets3ds structure. If pointer is NULL, then memory is allocated for the new structure and its address is stored in the pointer.

Returns nothing.

ReleaseKeySets3ds

***Prototype:*** void ReleaseKeySets3ds(kfsets3ds \*\*key)

Frees all memory allocated to the kfsets3ds structure.

Arguments:

kfsets3ds \*\*key

The settings structure to be freed. After release, the pointer is set to NULL

Returns nothing.

GetKfSets3ds

***Prototype:*** void GetKfSets3ds(database3ds \*db, kfsets3ds \*\*key)

Gets the settings contained in the database and copies them into a kfsets3ds structure. Only databases for .3ds and .prj file contain usable information.

Arguments:

database3ds \*db

The database that is being queried.

kfsets3ds \*\*sets

The address of a pointer to the kfsets3ds structure. If the referenced pointer is NULL, then a call to InitKfSeta3ds is performed prior to filling out the structure with the new settings.

Returns nothing.

PutKfSets3ds

***Prototype:*** void PutKfSets3ds(database3ds \*db, kfsets3ds \*key)

Puts the settings contained in the kfsets3ds structure into the database. Overwriting any existing settings with the new ones.

Arguments:

database3ds \*db

The database being written into. The database must be a dbtype3ds of MeshFile or ProjectFile.

kfsets3ds \*sets

The settings being written.

Returns nothing.

CopyKfSets3ds

***Prototype:*** void CopyKfSets3ds(database3ds \*destdb, database3ds \*srcdb)

Shallow copies the keyframe settings from one database into another. Existing keyframe settings in the destination database are replaced with the new copy.

Arguments:

database3ds \*destdb

The database to receive the copied settings.

databse3ds \*srcdb

The database containing the original settings.

Returns nothing.

Camera Animation

Kfcamera3ds

Camera Animation

Camera animation key structure.

typedef struct {

char3ds name[11];

      char3ds \*parent[22];

      ushort3ds flags1;

ushort3ds flags2;

      ulong3ds npkeys;

ushort3ds npflag;

keyheader3ds \*pkeys;

point3ds \*pos;

      ulong3ds nfkeys;

ushort3ds nfflag;

keyheader3ds \*fkeys;

float3ds \*fov;

      ulong3ds nrkeys;

ushort3ds nrflag;

keyheader3ds \*rkeys;

float3ds \*roll;

      char3ds tparent[22];

ulong3ds ntkeys;

ushort3ds ntflag;

keyheader3ds \*tkeys;

point3ds \*tpos;

ushort3ds tflags1;

ushort3ds tflags2;

} kfcamera3ds;

char3ds name[11]

Name of camera to animate. Defaults to empty string.

char3ds parent[22]

Name of parent node. With meshes the name is a concatenation of the object and instance name separated by a ".". For example, to specify a parent from a mesh instance of "Object01" and an instance name of "Instance01" the string "Object01.Instance01" would be used. Dummy objects as parents are constructed from the special dummy identifier "$$$Dummy", a period, and the dummy name, as in "$$$Dummy.Dummy01".

ushort3ds flags1

Special node flags, present but not supported in this structure.

ushort3ds flags2

Special node flags, any combination of the following:

static const ushortds NodeHasPath3ds = 1;Display the camera motion path. Defaults to 0.

ulong3ds npkeys

Count of position keys in track. Default is 0.

ushort3ds npflag

Track options for position track, can be one of the following:

static const ushort3ds TrackSingle3ds = 0x0000;Track does not repeat or loop.

static const ushort3ds TrackLoop3ds = 0x0003;Track loops.

static const ushort3ds TrackRepeat3ds = 0x0002;Track repeats.

static const ushort3ds TrackLockX3ds = 0x0008;Constrains motion to the X axis.

static const ushort3ds TrackLockY3ds = 0x0010;Constrains motion to the Y axis.

static const ushort3ds TrackLockZ3ds = 0x0020;Constrains motion to the Z axis.

static const ushort3ds TrackNoLinkX3ds = 0x0100;Prevents child from inheriting parent's X axis motion.

static const ushort3ds TrackNoLinkY3ds = 0x0200;Prevents child from inheriting parent's Y axis motion.

static const ushort3ds TrackNoLinkZ3ds = 0x0400;Prevents child from inheriting parent's Z axis motion.

Default is TrackSingle3ds.

keyheader3ds \*pkeys

Array of spline path modifiers for the position keys. Must have same element count as npkeys.

point3ds \*pos

Array of position keys. Element count must be the same number as npkeys.

ulong3ds nfkeys

Count of FOV keys in track. Default is 0.

ushort3ds nfflag

Track options for FOV track. See npflag for their meanings.

keyheader3ds \*fkeys

Array of spline modifiers for the FOV keys. Element count must be the same as nfkeys.

float3ds \*fov

Array of FOV values. Element count must be the same number as nfkeys.

ulong3ds nrkeys

Number of roll keys in the track. Default is 0.

ushort3ds nrflag

Track options for roll track. See npflag for their meanings.

keyheader3ds \*rkeys

Array of spline modifiers for camera roll. Element count must be the same number as nrkeys.

float3ds \*roll

Array of camera roll keys. Element count must be the same number as nrkeys.

char3ds tparent[22]

Parent name for camera target. See parent field for more information.

ulong3ds ntkeys

Number of track position keys. Default is 0.

ushort3ds ntflag

Track options for target keys. See npflag for their meanings.

keyheader3ds \*tkeys

Array of spline path modifiers for target keys. Element count must be the same number as ntkeys.

point3ds \*tpos

Array of target position keys. Element count must be the same number as ntkeys.

ushort3ds tflags1

Special node flags, present but not supported in this structure.:

ushort3ds tflags2

Special node flags, any combination of the following:

static const ushortds NodeHasPath3ds = 1;Display the camera motion path.

InitCameraMotion3ds

***Prototype:*** void InitCameraMotion3ds(kfcamera3ds \*\*cam, ulong3ds npkeys, ulong3ds nfkeys, ulong3ds nrkeys, ulong3ds ntkeys);

Initializes kfcamera3ds structure and optionally allocates memory.

Arguments:

kfcamera3ds \*\*cam

Address of a pointer to a kfcamera3ds structure. If pointer is NULL, then memory is allocated for a new structure and the address stored in the pointer.

ulong3ds npkeys

Allocate memory for specified number of position keys. A value of 0 leaves the position key fields unchanged.

ulong3ds nfkeys

Allocate memory for specified number of FOV keys. A value of 0 leaves the FOV keys fields unchanged.

ulong3ds nrkeys

Allocate memory for specified number of roll keys. A value of 0 leaves the roll keys fields unchanged.

ulong3ds ntkeys

Allocate memory for specified number of target keys. A value of 0 leaves the target key fields unchanged.

Returns nothing.

ReleaseCameraMotion3ds

***Prototype:*** void ReleaseCameraMotion3ds(kfcamera3ds \*\*cam);

Frees memory allocated to a kfcamera3ds structure.

Arguments:

kfcamera3ds \*\*cam

The structure to be freed. After release, the pointer is set to NULL.

Returns nothing.

GetCameraNodeCount3ds

***Prototype:*** ulong3ds GetCameraNodeCount3ds(database3ds \*db);

Scans the database for camera nodes and returns a count of their number.

Arguments:

database3ds \*db

The database to scan for camera nodes.

Returns **ulong3ds** as the count of camera nodes.

GetCameraNodeNameList3ds

***Prototype:*** void GetCameraNodeNameList3ds(database3ds \*db, namelist3ds \*\*list);

Scans the database for camera nodes and returns a list of their names.

Arguments:

database3ds \*db

The database to scan for camera nodes.

namelist3ds \*\*list

The list of camera node names found. Point gets passed to InitNameList3ds prior to use.

Returns nothing.

GetCameraMotionByIndex3ds

***Prototype:*** void GetCameraMotionByIndex3ds(database3ds \*db, ulong3ds index, kfcamera3ds \*\*kfcam)

Searches the database for the index(th) occurrence of a camera node, and copies that node into camera motion structure. If the index is out of range, the structure remains unchanged.

Arguments:

database3ds \*db

The database being searched.

ulong3ds index

The order of occurrence of the node in the database. Valid range is 0 to one less than the count returned by GetCameraMotionCount3ds.

kfcamera3ds \*\*kfcam

The camera node structure that receives the settings. The pointer is passed to InitCameraMotion3ds prior to use.

Returns nothing.

GetCameraMotionByName3ds

***Prototype:*** void GetCameraMotionByName3ds(database3ds \*db, char3ds \*name, kfcamera3ds \*\*kfcam)

Searches the database for a camera node of the given name and copies it into a camera node structure. If a the name is not found, then the structure remains unchanged.

Arguments:

database3ds \*db

The database being searched.

char3ds \*name

The name of 1–10 characters to search for. The search is case-sensitive.

kfcamera3ds \*\*kfcam

The camera node structure that receives the settings. The pointer is passed to InitCameraMotion3ds prior to use.

PutCameraMotion3ds

***Prototype:*** void PutCameraMotion3ds(database3ds \*db, kfcamera3ds \*kfcam)

Copies the camera motion settings from the structure into the database. Camera nodes with the same name are replaced with the new settings.

Arguments:

database3ds \*db

The database to copy the camera node into.

kfcamera3ds \*kfcam

The camera node being added to the database.

Returns nothing.

CopyCameraMotionByName3ds

***Prototype:*** void CopyCameraMotionByName3ds(database3ds \*destdb, database3ds \*srcdb, char3ds \*name)

Shallow copies a camera node from one database into another. Objects in the destination database with the same name as the copied camera node are replaced.

Arguments:

database3ds \*destdb

The database to receive the copied camera node.

databse3ds \*srcdb

The database containing the original camera node.

char3ds \*name

Name of the camera node to copy. Search is case sensitive.

Returns nothing.

CopyCameraMotionByIndex3ds

***Prototype:*** void CopyCameraMotionByIndex3ds(database3ds \*destdb, database3ds \*srcdb, ulong3ds index)

Shallow copies a camera node from one database into another. Objects in the destination database with the same name as the copied camera node are replaced.

Arguments:

database3ds \*destdb

The database to receive the copied camera node.

databse3ds \*srcdb

The database containing the original camera node.

ulong3ds index

The index of the camera node in the source database. Valid range is from 0 (the first one) to one less than the result of GetCameraNodeCount3ds as applied to \*srcdb.

Returns nothing.

DeleteCameraMotionByName3ds

***Prototype:*** void DeleteCameraMotionByName3ds(database3ds \*db, char3ds \*name)

Deletes a camera node from the database.

Arguments:

database3ds \*db

Database containing the camera node to delete.

char3ds \*name

Name of the camera node to delete. Search is case-sensitive.

Returns nothing.

DeleteCameraMotionByIndex3ds

***Prototype:*** void DeleteCameraMotionByIndex3ds(database3ds \*db, ulong3ds index)

Deletes a camera node from the database.

Arguments:

database3ds \*db

Database containing the camera node to delete.

ulong3ds index

The index of the camera node in the database. Valid range is from 0 (the first one) to one less than the result of GetCameraNodeCount3ds as applied to \*db.

Returns nothing.

Ambient Light Animation

Kfambient3ds

Ambient Light Animation Structure

Ambient light animation keys for color.

typedef struct {

      ushort3ds flags1;

ushort3ds flags2;

      ulong3ds nckeys;

ushort3ds ncflag;

keyheader3ds \*ckeys;

fcolor3ds \*color;

} kfambient3ds;

char3ds name[11]

Ambient light name should always be "$AMBIENT$"

short3ds parent

Parent node. Not used.

ushort3ds flags1

Internally used flags.

ushort3ds flags2

Internally used flags.

ulong3ds nckeys

Count of color keys in track. Default is 0.

ushort3ds ncflag

Track options for color track, can be one of the following:

static const ushort3ds TrackSingle3ds = 0x0000;Track does not repeat or loop.

static const ushort3ds TrackLoop3ds = 0x0002;Track loops.

static const ushort3ds TrackRepeat3ds = 0x0003;Track repeats.

Default is TrackSingle3ds.

keyheader3ds \*ckeys

Array of spline path modifiers for the color keys. Element count must be the same as nckeys.

fcolor3ds \*color

Array of color keys. Element count must be the same as nckeys.

InitAmbientLightMotion3ds

***Prototype:*** void InitAmbientLightMotion3ds(kfambient3ds \*\*light, ulong3ds nckeys)

Initializes kfambient3ds structure and optionally allocates memory.

Arguments:

kfambient3ds \*\*light

Address of a pointer to a kfambient3ds structure. If pointer is NULL, then memory is allocated for a new structure and the address stored in the pointer.

ulong3ds nckeys

Allocate memory for specified number of color keys. A value of 0 leaves the color key fields unchanged.

Returns nothing.

ReleaseAmbientLightMotion3ds

***Prototype:*** void ReleaseAmbientLightMotion3ds(kfambient3ds \*\*light);

Frees memory allocated to a kfambient structure.

Arguments:

kfambient3ds \*\*cam

The structure to be freed. After release, the pointer is set to NULL.

Returns nothing.

GetAmbientLightMotionByName3ds

***Prototype:*** void GetAmbientLightMotion3ds(database3ds \*db, kfambient3ds \*\*light)

Searches the database for a ambient light node and copies it into a ambient light node structure. If a node is not found, then the structure remains unchanged.

Arguments:

database3ds \*db

The database being searched.

kfambient3ds \*\*light

The ambient light node structure that receives the settings. The pointer is passed to InitAmbientLightMotion3ds prior to use.

Returns nothing.

PutAmbientLightMotion3ds

***Prototype:*** void PutAmbientLightMotion3ds(database3ds \*db, kfambient3ds \*light)

Copies the ambient light motion settings from the structure into the database. Any existing ambient light node is replaced with the new settings.

Arguments:

database3ds \*db

The database to copy the ambient light node into.

kfambient3ds \*light

The ambient light node being added to the database.

Returns nothing.

Mesh Object Animation

Kfrotkey3ds

Rotation Key

typedef struct {

float3ds angle;

float3ds x;

float3ds y;

float3ds z;

} kfrotkey3ds;

float3ds angle

Angle of rotation in degrees. Defaults to 0.0.

float3ds x

X component of rotation axis. Defaults to 0.0.

float3ds y

Y component of rotation axis. Defaults to 0.0.

float3ds z

Z component of rotation axis. Defaults to 0.0.

Kfmorphkey3ds

Morph Object Key

typedef struct {

char3ds name[11];

} kfmorphkey3ds;

char3ds name[11]

Name of morph target light. Mesh must have same number of vertices as base object.

Kfmesh3ds

Mesh Animation

typedef struct {

char3ds name[11];

      char3ds parent[22];

ushort3ds flags1;

ushort3ds flags2;

      point3ds pivot;

char3ds instance[11];

point3ds boundmin;

point3ds boundmax;

      ulong3ds npkeys;

short3ds npflag;

keyheader3ds \*pkeys;

point3ds \*pos;

      ulong3ds nrkey

short3ds nrflag;

keyheader3ds \*rkeys;

kfrotkey3ds \*rot;

      ulong3ds nskeys;

short3ds nsflag;

keyheader3ds \*skeys;

point3ds \*scale;

      ulong3ds nmkeys;

short3ds nmflag;

keyheader3ds \*mkeys;

kfmorphkey3ds \*morph;

      ulong3ds nhkeys;

short3ds nhflag;

keyheader3ds \*hkeys;

      float3ds msangle;

} kfmesh3ds;

char3ds name[11]

Name of mesh to animate. Special name of "$$$DUMMY" is used for a dummy object. Defaults to empty string.

char3ds parent[22]

Name of parent node. With meshes the name is a concatenation of the object and instance name separated by a ".". For example, to specify a parent from a mesh instance of "Object01" and an instance name of "Instance01" the string "Object01.Instance01" would be used. Dummy objects as parents are constructed from the special dummy identifier "$$$Dummy", a period, and the dummy name, as in "$$$Dummy.Dummy01".

ushort3ds flags1

Special node flags, any combination of the following:

static const ushort3ds KfHideNode3ds = 1<<11;Mesh node is hidden in Keyframer.

static const ushort3ds KfFastNode3ds = 1<<12;Mesh node is fast drawn in Keyframer.

ushort3ds flags2

Special node flags, any combination of the following:

static const ushortds KfNodeHasPath3ds = 1;Display the object motion path.

static const ushort3ds KfNodeAutoSmooth3ds = 1<<1;Mesh object uses automatic smoothing group calculation.

static const ushort3ds KfNodeFrozen3ds = 1<<2;Mesh object is frozen in Keyframer.

static const ushort3ds KfMotionBlur3ds = 1<<4;Mesh object uses motion blur.

static const ushort3ds KfBlurBranch3ds = 1<<5;Mesh object's children are motion blurred.

static const ushort3ds KfMorphMtl3ds = 1<<6;Mesh object's materials are morphed.

static const ushort3ds KfMorphOb = 1<<7;Mesh object is morphed.

point3ds pivot

Rotation pivot point for object node. Defaults to 0.0, 0.0, 0.0.

char3ds instance[11]

Name of object node instance.

point3ds boundmin

Point defining the upper, left, and front extents of the object.

point3ds boundmax

Point defining the lower, right, and back extents of the object.

ulong3ds npkeys

Count of position keys in track. Default is 0.

ushort3ds npflag

Track options for position track, can be one of the following:

static const ushort3ds TrackSingle3ds = 0x0000;Track does not repeat or loop.

static const ushort3ds TrackLoop3ds = 0x0003;Track loops.

static const ushort3ds TrackRepeat3ds = 0x0002;Track repeats.

static const ushort3ds TrackLockX3ds = 0x0008;Constrains motion to the X axis.

static const ushort3ds TrackLockY3ds = 0x0010;Constrains motion to the Y axis.

static const ushort3ds TrackLockZ3ds = 0x0020;Constrains motion to the Z axis.

static const ushort3ds TrackNoLinkX3ds = 0x0100;Prevents child from inheriting parent's X axis motion.

static const ushort3ds TrackNoLinkY3ds = 0x0200;Prevents child from inheriting parent's Y axis motion.

static const ushort3ds TrackNoLinkZ3ds = 0x0400;Prevents child from inheriting parent's Z axis motion.

Default is TrackSingle3ds.

keyheader3ds \*pkeys

Array of spline path modifiers for the position keys. Element count must be the same as npkeys.

point3ds \*pos

Array of position keys. Element count must be the same as npkeys.

ulong3ds nrkeys

Count of rotation keys in track. Default is 0.

short3ds nrflag

Track options for rotation track. See npflag for their meanings.

keyheader3ds \*rkeys

Array of spline path modifiers for the rotation keys. Element count must be the same as nrkeys.

kfrotkey3ds \*rot

Array of rotation keys. Element count must be the same as nrkeys.

ulong3ds nskeys

Count of scale keys in track. Default is 0.

short3ds nsflag

Track options for scale track. See npflag for their meanings.

keyheader3ds \*skeys

Array of spline path modifiers for the scale keys. Element count must be the same as nskeys.

point3ds \*scale

Array of scale keys. Element count must be the same as nskeys.

ulong3ds nmkeys

Count of morph keys in track. Default is 0.

short3ds nmflag

Track options for morph track. See npflag for their meanings.

keyheader3ds \*mkeys

Array of spline path modifiers for the morph keys. Element count must be the same as nmkeys.

kfmorphkey3ds \*morph

Array of morph keys. Element count must be the same as nmkeys.

ulong3ds nhkeys

Count of hide keys in track. Default is 0.

short3ds nhflag

Track options for hide track. See npflag for their meanings.

keyheader3ds \*hkeys

Array of hide keys. Spline modifiers are ignored. Element count must be the same as nhkeys.

float3ds msangle

Angle for morph smoothing angle. Corresponds to the Angle field in the Keyframer/Object/Morph/Options/Morph Options dialog. Defaults to 24.0

InitObjectMotion3ds

***Prototype:*** void InitObjectMotion3ds(kfmesh3ds \*\*obj, ulong3ds npkeys, ulong3ds nrkeys, ulong3ds nskeys, ulong3ds nmkeys, ulong3ds nhkeys);

Initializes kfmesh3ds structure and optionally allocates memory.

Arguments:

kfmesh3ds \*\*obj

Address of a pointer to a kfmesh3ds structure. If pointer is NULL, then memory is allocated for a new structure and the address stored in the pointer.

ulong3ds npkeys

Allocate memory for specified number of position keys. A value of 0 leaves the position key fields unchanged.

ulong3ds nrkeys

Allocate memory for specified number of rotation keys. A value of 0 leaves the rotation key fields unchanged.

ulong3ds nskeys

Allocate memory for specified number of scale keys. A value of 0 leaves the scale key fields unchanged.

ulong3ds nmkeys

Allocate memory for specified number of morph keys. A value of 0 leaves the morph key fields unchanged.

ulong3ds nhkeys

Allocate memory for specified number of hide keys. A value of 0 leaves the hide key fields unchanged.

Returns nothing.

ReleaseObjectMotion3ds

***Prototype:*** void ReleaseObjectMotion3ds(kfmesh3ds \*\*obj);

Frees memory allocated to a kfmesh3ds structure.

Arguments:

kfmesh3ds \*\*obj

The structure to be freed. After release, the pointer is set to NULL.

Returns nothing.

GetObjectNodeCount3ds

***Prototype:*** ulong3ds GetObjectNodeCount3ds(database3ds \*db);

Scans the database for mesh nodes and returns a count of their number.

Arguments:

database3ds \*db

The database to scan for mesh nodes.

Returns **ulong3ds** as the count of mesh nodes.

GetObjectNodeNameList3ds

***Prototype:*** void GetObjectNodeNameList3ds(database3ds \*db, namelist3ds \*\*list);

Scans the database for mesh nodes and returns a list of their names.

Arguments:

database3ds \*db

The database to scan for mesh nodes.

namelist3ds \*\*list

The list of mesh node names found. Pointer gets past to InitNameList3ds prior to use.

Returns nothing.

GetObjectMotionByIndex3ds

***Prototype:*** void GetObjectMotionByIndex3ds(database3ds \*db, short3ds index, kfmesh3ds \*\*obj)

Searches the database for the index(th) occurrence of a mesh node, and copies that node into mesh motion structure. If the index is out of range, the structure remains unchanged.

Arguments:

database3ds \*db

The database being searched.

ulong3ds index

The order of occurrence of the node in the database. Valid range is 0 to one less than the count returned by GetObjectMotionCount3ds.

kfmesh3ds \*\*obj

The mesh node structure that receives the settings. The pointer is passed to InitObjectMotion3ds prior to use.

Returns nothing.

GetObjectMotionByName3ds

***Prototype:*** void GetObjectMotionByName3ds(database3ds \*db, char3ds \*name, kfmesh3ds \*\*obj);

Searches the database for a mesh node of the given name and copies it into a mesh node structure. If the name is not found, then the structure remains unchanged.

Arguments:

database3ds \*db

The database being searched.

char3ds \*name

The name of 1–10 characters to search for. The search is case-sensitive.

kfmesh3ds \*\*obj

The mesh node structure that receives the settings. The pointer is passed to InitObjectMotion3ds prior to use.

PutObjectMotion3ds

***Prototype:*** void PutObjectMotion3ds(database3ds \*db, kfmesh3ds \*obj);

Copies the mesh motion settings from the structure into the database. Object nodes with the same name are replaced with the new settings.

Arguments:

database3ds \*db

The database to copy the mesh node into.

kfmesh3ds \*obj

The mesh node being added to the database.

Returns nothing.

CopyObjectMotionByName3ds

***Prototype:*** void CopyObjectMotionByName3ds(database3ds \*destdb, database3ds \*srcdb, char3ds \*name)

Shallow copies a mesh node from one database into another. Objects in the destination database with the same name as the copied mesh node are replaced.

Arguments:

database3ds \*destdb

The database to receive the copied mesh node.

databse3ds \*srcdb

The database containing the original mesh node.

char3ds \*name

Name of the mesh node to copy. Search is case-sensitive.

Returns nothing.

CopyObjectMotionByIndex3ds

***Prototype:*** void CopyObjectMotionByIndex3ds(database3ds \*destdb, database3ds \*srcdb, ulong3ds index)

Shallow copies a mesh node from one database into another. Objects in the destination database with the same name as the copied mesh node are replaced.

Arguments:

database3ds \*destdb

The database to receive the copied mesh node.

databse3ds \*srcdb

The database containing the original mesh node.

ulong3ds index

The index of the mesh node in the source database. Valid range is from 0 (the first one) to one less than the result of GetObjectMotionCount3ds as applied to \*srcdb.

Returns nothing.

DeleteObjectMotionByName3ds

***Prototype:*** void DeleteObjectMotionByName3ds(database3ds \*db, char3ds \*name)

Deletes a mesh node from the database.

Arguments:

database3ds \*db

Database containing the mesh node to delete.

char3ds \*name

Name of the mesh node to delete. Search is case-sensitive.

Returns nothing.

DeleteObjectMotionByIndex3ds

***Prototype:*** void DeleteObjectMotionByIndex3ds(database3ds \*db, ulong3ds index)

Deletes a mesh node from the database.

Arguments:

database3ds \*db

Database containing the mesh node to delete.

ulong3ds index

The index of the mesh node in the database. Valid range is from 0 (the first one) to one less than the result of GetObjectNodeCount3ds as applied to \*db.

Returns nothing.

Omni Light Animation

Kfomni3ds

Omni light Animation Tracks

typedef struct {

char3ds name[11];

      char3ds parent[22];

      ushort3ds flags1;

ushort3ds flags2;

      ulong3ds npkeys;

ushort3ds npflag;

keyheader3ds \*pkeys;

point3ds \*pos;

      ulong3ds nckeys;

ushort3ds ncflag;

keyheader3ds \*ckeys;

fcolor3ds \*color;

} kfomni3ds;

char3ds name[11];

Name of omni light to animate.

char3ds parent[22]

Name of parent node. With meshes the name is a concatenation of the object and instance name separated by a ".". For example, to specify a parent from a mesh instance of "Object01" and an instance name of "Instance01" the string "Object01.Instance01" would be used. Dummy objects as parents are constructed from the special dummy identifier "$$$Dummy", a period, and the dummy name, as in "$$$Dummy.Dummy01".

ushort3ds flags1

Special node flags, any combination of the following:

static const ushort3ds KfNodeOff3ds = 1<<3;Light node is off in Keyframer.

ushort3ds flags2

Special node flags, any combination of the following:

static const ushortds KfNodeHasPath3ds = 1;Display the object motion path.

ulong3ds npkeys

Count of position keys in track. Default is 0.

ushort3ds npflag

Track options for position track, can be one of the following:

static const ushort3ds TrackSingle3ds = 0x0000;Track does not repeat or loop.

static const ushort3ds TrackLoop3ds = 0x0003;Track loops.

static const ushort3ds TrackRepeat3ds = 0x0002;Track repeats.

static const ushort3ds TrackLockX3ds = 0x0008;Constrains motion to the X axis.

static const ushort3ds TrackLockY3ds = 0x0010;Constrains motion to the Y axis.

static const ushort3ds TrackLockZ3ds = 0x0020;Constrains motion to the Z axis.

static const ushort3ds TrackNoLinkX3ds = 0x0100;Prevents child from inheriting parent's X axis motion.

static const ushort3ds TrackNoLinkY3ds = 0x0200;Prevents child from inheriting parent's Y axis motion.

static const ushort3ds TrackNoLinkZ3ds = 0x0400;Prevents child from inheriting parent's Z axis motion.

Default is TrackSingle3ds.

keyheader3ds \*pkeys

Array of spline path modifiers for the position keys. Element count must be the same as npkeys.

point3ds \*pos

Array of position keys. Element count must be the same as npkeys.

ulong3ds nckeys;

Count of color keys in track. Default is 0.

ushort3ds ncflag;

Track options for color track, can be one of the following. See npflag for their meanings.

keyheader3ds \*ckeys;

Array of spline path modifiers for the color keys. Element count must be the same as nckeys.

fcolor3ds \*color;

Array of color keys. Element count must be the same as nckeys.

InitOmnilightMotion3ds

***Prototype:*** void InitOmnilightMotion3ds(kfomni3ds \*\*light, ulong3ds npkeys, ulong3ds nckeys)

Initializes kfomni3ds structure and optionally allocates memory.

Arguments:

kfomni3ds \*\*obj

Address of a pointer to a kfomni3ds structure. If pointer is NULL, then memory is allocated for a new structure and the address stored in the pointer.

ulong3ds npkeys

Allocate memory for specified number of position keys. A value of 0 leaves the position key fields unchanged.

ulong3ds nckeys

Allocate memory for specified number of color keys. A value of 0 leaves the color key fields unchanged.

Returns nothing.

ReleaseOmnilightMotion3ds

***Prototype:*** void ReleaseOmnilightMotion3ds(kfomni3ds \*\*light)

Frees memory allocated to a kfomni3ds structure.

Arguments:

kfomni3ds \*\*light

The structure to be freed. After release, the pointer is set to NULL.

Returns nothing.

GetOmnilightNodeCount3ds

***Prototype:*** ulong3ds GetOmnilightNodeCount3ds(database3ds \*db)

Scans the database for omni light nodes and returns a count of their number.

Arguments:

database3ds \*db

The database to scan for mesh nodes.

Returns **ulong3ds** as the count of mesh nodes.

GetOmnilightNodeNameList3ds

***Prototype:*** void GetOmnilightNodeNameList3ds(database3ds \*db, namelist3ds \*\*list)

Scans the database for omni light nodes and returns a list of their names.

Arguments:

database3ds \*db

The database to scan for omni light nodes.

namelist3ds \*\*list

The list of omni light node names found. Pointer gets past to InitNameList3ds prior to use.

Returns nothing.

GetOmnilightMotionByIndex3ds

***Prototype:*** void GetOmnilightMotionByIndex3ds(database3ds \*db, short3ds index, kfomni3ds \*\*light)

Searches the database for the index(th) occurrence of a omni light node, and copies that node into omni light motion structure. If the index is out of range, the structure remains unchanged.

Arguments:

database3ds \*db

The database being searched.

ulong3ds index

The order of occurrence of the node in the database. Valid range is 0 to one less than the count returned by GetOmnilightMotionCount3ds.

kfomni3ds \*\*light

The omni light node structure that receives the settings. The pointer is passed to InitOmnilightMotion3ds prior to use.

Returns nothing.

GetOmnilightMotionByName3ds

***Prototype:*** void GetOmnilightMotionByName3ds(database3ds \*db, char3ds \*name, kfomni3ds \*\*light)

Searches the database for a omni light node of the given name and copies it into a omni light node structure. If a the name is not found, then the structure remains unchanged.

Arguments:

database3ds \*db

The database being searched.

char3ds \*name

The name of 1–10 characters to search for. The search is case-sensitive.

kfomni3ds \*\*light

The omni light node structure that receives the settings. The pointer is passed to InitOmnilightMotion3ds prior to use.

PutOmnilightMotion3ds

***Prototype:*** void PutOmnilightMotion3ds(database3ds \*db, kfomni3ds \*light)

Copies the omni light motion settings from the structure into the database. Object nodes with the same name are replaced with the new settings.

Arguments:

database3ds \*db

The database to copy the omni light node into.

kfomni3ds \*light

The omni light node being added to the database.

Returns nothing.

CopyOmnilightMotionByName3ds

***Prototype:*** void CopyOmnilightMotionByName3ds(database3ds \*destdb, database3ds \*srcdb, char3ds \*name)

Shallow copies an omni light node from one database into another. Objects in the destination database with the same name as the copied omni light node are replaced.

Arguments:

database3ds \*destdb

The database to receive the copied omni light node.

databse3ds \*srcdb

The database containing the original omni light node.

char3ds \*name

Name of the omni light node to copy. Search is case-sensitive.

Returns nothing.

CopyOmnilightMotionByIndex3ds

***Prototype:*** void CopyOmnilightMotionByIndex3ds(database3ds \*destdb, database3ds \*srcdb, ulong3ds index)

Shallow copies an omni light node from one database into another. Objects in the destination database with the same name as the copied omni light node are replaced.

Arguments:

database3ds \*destdb

The database to receive the copied omni light node.

databse3ds \*srcdb

The database containing the original omni light node.

ulong3ds index

The index of the omni light node in the source database. Valid range is from 0 (the first one) to one less than the result of GetOmnilightNodeCount3ds as applied to \*srcdb.

Returns nothing.

DeleteOmnilightMotionByName3ds

***Prototype:*** void DeleteOmnilightMotionByName3ds(database3ds \*db, char3ds \*name)

Deletes an omni light node from the database.

Arguments:

database3ds \*db

Database containing the omni light node to delete.

char3ds \*name

Name of the omni light node to delete. Search is case-sensitive.

Returns nothing.

DeleteOmnilightMotionByIndex3ds

***Prototype:*** void DeleteOmnilightMotionByIndex3ds(database3ds \*db, ulong3ds index)

Deletes an omni light node from the database.

Arguments:

database3ds \*db

Database containing the omni light node to delete.

ulong3ds index

The index of the omni light node in the database. Valid range is from 0 (the first one) to one less than the result of GetOmnilightNodeCount3ds as applied to \*db.

Returns nothing.

Spotlight Animation

Kfspot3ds

Spotlight Animation Settings

typedef struct {

char3ds name[11];

      char3ds parent;[22]

      ushort3ds flags1;

ushort3ds flags2;

      ulong3ds npkeys;

ushort3ds npflag;

keyheader3ds \*pkeys;

point3ds \*pos;

      ulong3ds nckeys;

ushort3ds ncflag;

keyheader3ds \*ckeys;

fcolor3ds \*color;

      ulong3ds nhkeys;

ushort3ds nhflag;

keyheader3ds \*hkeys;

float3ds \*hot;

      ulong3ds nfkeys;

ushort3ds nfflag;

keyheader3ds \*fkeys;

float3ds \*fall;

      ulong3ds nrkeys;

ushort3ds nrflag;

keyheader3ds \*rkeys;

float3ds \*roll;

      short3ds tparent;

ulong3ds ntkeys;

ushort3ds ntflag;

keyheader3ds \*tkeys;

point3ds \*tpos;

ushort3ds tflags1;

ushort3ds tflags2;

} kfspot3ds;

char3ds name[11]

Name of spotlight to animate.

char3ds parent[22]

Name of parent node. With meshes the name is a concatenation of the object and instance name separated by a ".". For example, to specify a parent from a mesh instance of "Object01" and an instance name of "Instance01" the string "Object01.Instance01" would be used. Dummy objects as parents are constructed from the special dummy identifier "$$$Dummy", a period, and the dummy name, as in "$$$Dummy.Dummy01".

ushort3ds flags1

Special node flags, any combination of the following:

static const ushort3ds KfNodeOff3ds = 1<<3;Light node is off in Keyframer.

ushort3ds flags2

Special node flags, any combination of the following:

static const ushortds KfNodeHasPath3ds = 1;Display the object motion path.

ulong3ds npkeys

Count of position keys in track. Default is 0.

ushort3ds npflag

Track options for position track, can be one of the following:

static const ushort3ds TrackSingle3ds = 0x0000;Track does not repeat or loop.

static const ushort3ds TrackLoop3ds = 0x0003;Track loops.

static const ushort3ds TrackRepeat3ds = 0x0002;Track repeats.

static const ushort3ds TrackLockX3ds = 0x0008;Constrains motion to the X axis.

static const ushort3ds TrackLockY3ds = 0x0010;Constrains motion to the Y axis.

static const ushort3ds TrackLockZ3ds = 0x0020;Constrains motion to the Z axis.

static const ushort3ds TrackNoLinkX3ds = 0x0100;Prevents child from inheriting parent's X axis motion.

static const ushort3ds TrackNoLinkY3ds = 0x0200;Prevents child from inheriting parent's Y axis motion.

static const ushort3ds TrackNoLinkZ3ds = 0x0400;Prevents child from inheriting parent's Z axis motion.

Default is TrackSingle3ds.

keyheader3ds \*pkeys

Array of spline path modifiers for the position keys. Element count must be the same as npkeys.

point3ds \*pos

Array of position keys. Element count must be the same as npkeys.

ulong3ds nckeys

Count of color keys in track. Default is 0.

ushort3ds ncflag

Track options for color track, can be one of the following. See npflag for their meanings.

keyheader3ds \*ckeys

Array of spline path modifiers for the color keys. Element count must be the same as nckeys.

fcolor3ds \*color

Array of color keys. Element count must be the same number as nckeys.

ulong3ds nhkeys

Count of hot spot keys in track. Default is 0.

ushort3ds nhflag

Track options for hot spot track, can be one of the following. See npflag for their meanings.

keyheader3ds \*hkeys

Array of spline path modifiers for the hot spot keys. Must have same element count as nhkeys.

float3ds \*hot

Array of hot spot keys. Element count must be the same number as nhkeys.

ulong3ds nfkeys

Count of falloff keys in track. Default is 0.

ushort3ds ncflag

Track options for falloff track, can be one of the following. See npflag for their meanings.

keyheader3ds \*fkeys

Array of spline path modifiers for the falloff keys. Element count must be the same as nfkeys.

fcolor3ds \*fall

Array of falloff keys. Element count must be the same as nfkeys.

ulong3ds nrkeys

Number of roll keys in the track. Default is 0.

ushort3ds nrflag

Track options for roll track. Se npflag for their meanings.

keyheader3ds \*rkeys

Array of spline modifiers for spot roll. Element count must be the same as nrkeys.

float3ds \*roll

Array of spot roll keys. Element count must be the same as nrkeys.

char3ds tparent[22]

Parent name for camera target. See parent field for more information.

ulong3ds ntkeys

Number of track position keys. Default is 0.

ushort3ds ntflag

Track options for target keys. See npflag for their meanings.

keyheader3ds \*tkeys

Array of spline path modifiers for target keys. Element count must be the same as ntkeys.

point3ds \*tpos

Array of target position keys. Element count must be the same as ntkeys.

ushort3ds tflags1

Special node flags, present but not used in this structure.

ushort3ds tflags2

Special node flags, any combination of the following:

static const ushortds NodeHasPath3ds = 1;Display the camera motion path.

InitSpotlightMotion3ds

***Prototype:*** void InitSpotlightMotion3ds(kfspot3ds \*\*spot, ulong3ds npkeys, ulong3ds nckeys, ulong3ds nhkeys, ulong3ds nfkeys, ulong3ds nrkeys, ulong3ds ntkeys);

Initializes kfspot3ds structure and optionally allocates memory.

Arguments:

kfspot3ds \*\*spot

Address of a pointer to a kfspot3ds structure. If pointer is NULL, then memory is allocated for a new structure and the address stored in the pointer.

ulong3ds npkeys

Allocate memory for specified number of position keys. A value of 0 leaves the position key fields unchanged.

ulong3ds nckeys

Allocate memory for specified number of color keys. A value of 0 leaves the color key fields unchanged.

ulong3ds nhkeys

Allocate memory for specified number of hot spot keys. A value of 0 leaves the hot spot key fields unchanged.

ulong3ds nfkeys

Allocate memory for specified number of falloff keys. A value of 0 leaves the falloff key fields unchanged.

ulong3ds nrkeys

Allocate memory for specified number of roll keys. A value of 0 leaves the roll key fields unchanged.

ulong3ds ntkeys

Allocate memory for specified number of target position keys. A value of 0 leaves the target position key fields unchanged.

Returns nothing.

ReleaseSpotlightMotion3ds

***Prototype:*** void ReleaseSpotlightMotion3ds(kfspot3ds \*\*spot);

Frees memory allocated to a kfspot3ds structure.

Arguments:

kfspot3ds \*\*spot

The structure to be freed. After release, the pointer is set to NULL.

Returns nothing.

GetSpotlightNodeCount3ds

***Prototype:*** long3ds GetSpotlightNodeCount3ds(database3ds \*db);

Scans the database for spotlight nodes and returns a count of their number.

Arguments:

database3ds \*db

The database to scan for spotlight nodes.

Returns **ulong3ds** as the count of mesh nodes.

GetSpotlightNodeNameList3ds

***Prototype:*** void GetSpotlightNodeNameList3ds(database3ds \*db, chunklist3ds \*\*list);

Scans the database for spotlight nodes and returns a list of their names.

Arguments:

database3ds \*db

The database to scan for spotlight nodes.

namelist3ds \*\*list

The list of spotlight node names found. Pointer gets past to InitNameList3ds prior to use.

Returns nothing.

GetSpotlightMotionByIndex3ds

***Prototype:*** void GetSpotlightMotionByIndex3ds(database3ds \*db, ulong3ds index, kfspot3ds \*spot);

Searches the database for the index(th) occurrence of a spotlight node, and copies that node into spotlight motion structure. If the index is out of range, the structure remains unchanged.

Arguments:

database3ds \*db

The database being searched.

ulong3ds index

The order of occurrence of the node in the database. Valid range is 0 to one less than the count returned by GetSpotlightMotionCount3ds.

kfspot3ds \*\*spot

The spotlight node structure that receives the settings. The pointer is passed to InitSpotlightMotion3ds prior to use.

Returns nothing.

GetSpotlightMotionByName3ds

***Prototype:*** void GetSpotlightMotionByName3ds(database3ds \*db, char3ds \*name, kfspot3ds \*\*spot);

Searches the database for a spotlight node of the given name and copies it into a spotlight node structure. If a the name is not found, then the structure remains unchanged.

Arguments:

database3ds \*db

The database being searched.

char3ds \*name

The name of 1–10 characters to search for. The search is case-sensitive.

kfspot3ds \*\*spot

The spotlight node structure that receives the settings. The pointer is passed to InitSpotlightMotion3ds prior to use.

PutSpotlightMotion3ds

***Prototype:*** void PutSpotlightMotion3ds(database3ds \*db, kfspot3ds \*spot);

Copies the spotlight motion settings from the structure into the database. Object nodes with the same name are replaced with the new settings.

Arguments:

database3ds \*db

The database to copy the spotlight node into.

kfspot3ds \*spot

The spotlight node being added to the database.

CopySpotlightMotionByName3ds

***Prototype:*** void CopySpotlightMotionByName3ds(database3ds \*destdb, database3ds \*srcdb, char3ds \*name)

Shallow copies an spotlight node from one database into another. Objects in the destination database with the same name as the copied spotlight node are replaced.

Arguments:

database3ds \*destdb

The database to receive the copied spotlight node.

databse3ds \*srcdb

The database containing the original spotlight node.

char3ds \*name

Name of the spotlight node to copy. Search is case-sensitive.

Returns nothing.

CopySpotlightMotionByIndex3ds

***Prototype:*** void CopySpotlightMotionByIndex3ds(database3ds \*destdb, database3ds \*srcdb, ulong3ds index)

Shallow copies an spotlight node from one database into another. Objects in the destination database with the same name as the copied spotlight node are replaced.

Arguments:

database3ds \*destdb

The database to receive the copied spotlight node.

databse3ds \*srcdb

The database containing the original spotlight node.

ulong3ds index

The index of the spotlight node in the source database. Valid range is from 0 (the first one) to one less than the result of GetSpotlightMotionCount3ds as applied to \*srcdb.

Returns nothing.

DeleteSpotlightMotionByName3ds

***Prototype:*** void DeleteSpotlightMotionByName3ds(database3ds \*db, char3ds \*name)

Deletes an spotlight node from the database.

Arguments:

database3ds \*db

Database containing the spotlight node to delete.

char3ds \*name

Name of the spotlight node to delete. Search is case sensitive.

Returns nothing.

DeleteSpotlightMotionByIndex3ds

***Prototype:*** void DeleteSpotlightMotionByIndex3ds(database3ds \*db, ulong3ds index)

Deletes an spotlight node from the database.

Arguments:

database3ds \*db

Database containing the spotlight node to delete.

ulong3ds index

The index of the spotlight node in the database. Valid range is from 0 (the first one) to one less than the result of GetSpotlightNodeCount3ds as applied to \*db.

Returns nothing.

////

Common Keyframe Structures

Keyheader3ds

Key temporal and spline attributes

typedef struct

{

ulong3ds time;

ushort3ds rflags;

float3ds tension;

float3ds continuity;

float3ds bias;

float3ds easeto;

float3ds easefrom;

} keyheader3ds;

ulong3ds time

The frame number the key occurs in. Default is 0.

ushort3ds rflags

A set of flags that define which spline terms are in use.

static const ushort3ds KeyUsesNone3ds = 0x00;static const ushort3ds KeyUsesTension3ds = 0x01;static const ushort3ds KeyUsesCont3ds = 0x02;static const ushort3ds KeyUsesBias3ds = 0x04;static const ushort3ds KeyUsesEaseTo3ds = 0x08;static const ushort3ds KeyUsesEaseFrom3ds = 0x10;

Default is KeyUseNone3ds.

float3ds tension

Spline tension value from -1.0–1.0. Default is 0.0.

float3ds continuity

Spline continuity value from -1.0–1.0. Default is 0.0.

float3ds bias

Spline bias value from -1.0–1.0. Default is 0.0.

float3ds easeto

Key ease to value from 0.0 –1.0. Default is 0.0.

float3ds easefrom

Key ease from value from 0.0 –1.0. Default is 0.0.

Camera Animation

Kfcamera3ds

Camera Animation

Camera animation key structure.

typedef struct {

char3ds name[11];

short3ds parent;

      ushort3ds flags1;

ushort3ds flags2;

      ulong3ds npkeys;

ushort3ds npflag;

keyheader3ds \*pkeys;

point3ds \*pos;

      ulong3ds nfkeys;

ushort3ds nfflag;

keyheader3ds \*fkeys;

float3ds \*fov;

      ulong3ds nrkeys;

ushort3ds nrflag;

keyheader3ds \*rkeys;

float3ds \*roll;

      short3ds tparent;

ulong3ds ntkeys;

ushort3ds ntflag;

keyheader3ds \*tkeys;

point3ds \*tpos;

ushort3ds tflags1;

ushort3ds tflags2;

} kfcamera3ds;

char3ds name[11]

Name of camera to animate. Defaults to empty string.

short3ds parent

Parent index. Obsolete and must be replaced with something that works before the product is released.

ushort3ds flags

Special node flags, any combination of the following:

ushort3ds flags2

Special node flags, any combination of the following:

static const ushortds ShowPath3ds = 1;Display the camera motion path.

ulong3ds npkeys

Count of position keys in track. Default is 0.

ushort3ds npflag

Track options for position track, can be one of the following:

static const ushort3ds TrackSingle3ds = 0;Track does not repeat or loop.

static const ushort3ds TrackLoop3ds = 3;Track loops.

static const ushort3ds TrackRepeat3ds = 3;Track repeats.

Default is TrackSingle3ds.

keyheader3ds \*pkeys

Array of spline path modifiers for the position keys. Must have same element count as npkeys.

point3ds \*pos

Array of position keys. Element count must be the same number as npkeys.

ulong3ds nfkeys

Count of FOV keys in track. Default is 0.

ushort3ds nfflag

Track options for FOV track. See npflag for their meanings.

keyheader3ds \*fkeys

Array of spline modifiers for the FOV keys. Element count must be the same as nfkeys.

float3ds \*fov

Array of FOV values. Element count must be the same number as nfkeys.

ulong3ds nrkeys

Number of roll keys in the track. Default is 0.

ushort3ds nrflag

Track options for roll track. Se npflag for their meanings.

keyheader3ds \*rkeys

Array of spline modifiers for camera roll. Element count must be the same number as nrkeys.

float3ds \*roll

Array of camera roll keys. Element count must be the same number as nrkeys.

short3ds tparent

Parent index for camera target.

ulong3ds ntkeys

Number of track position keys. Default is 0.

ushort3ds ntflag

Track options for target keys. See npflag for thier meanings.

keyheader3ds \*tkeys

Array of spline path modifiers for target keys. Element count must be the same number as ntkeys.

point3ds \*tpos

Array of target position keys. Element count must be the same number as ntkeys.

ushort3ds tflags1

Special node flags.

ushort3ds tflags2

Special node flags.

InitCameraMotion3ds

***Prototype:*** void InitCameraMotion3ds(kfcamera3ds \*\*cam, ulong3ds npkeys, ulong3ds nfkeys, ulong3ds nrkeys, ulong3ds ntkeys);

Initializes kfcamera3ds structure and optionally allocates memory.

Arguments:

kfcamera3ds \*\*cam

Address of a pointer to a kfcamera3ds structure. If pointer is NULL, then memory will be allocated for a new structure and the address stored in the pointer.

ulong3ds npkeys

Allocate memory for specified number of position keys. A value of 0 will leave the position key fields unchanged.

ulong3ds nfkeys

Allocate memory for specified number of FOV keys. A value of 0 will leave the FOV keys fields unchanged.

ulong3ds nrkeys

Allocate memory for specified number of roll keys. A value of 0 will leave the roll keys fields unchanged.

ulong3ds ntkeys

Allocate memory for specified number of target keys. A value of j0 will leave the target key fields unchanged.

Returns nothing.

ReleaseCameraMotion3ds

***Prototype:*** void ReleaseCameraMotion3ds(kfcamera3ds \*\*cam);

Frees memory allocated to a kfcamera3ds structure.

Arguments:

kfcamera3ds \*\*cam

The structured to be freed. After release, the pointer is set to NULL.

Returns nothing.

GetCameraNodeCount3ds

***Prototype:*** ulong3ds GetCameraNodeCount3ds(database3ds \*db);

Scans the database for camera nodes and returns a count of their number.

Arguments:

database3ds \*db

The database to scan for camera nodes.

Returns **ulong3ds** as the count of camera nodes.

GetCameraNodeNameList3ds

***Prototype:*** void GetCameraNodeNameList3ds(database3ds \*db, namelist3ds \*\*list);

Scans the database for camera nodes and returns a list of their names.

Arguments:

database3ds \*db

The database to scan for camera nodes.

namelist3ds \*\*list

The list of camera node names found. Point gets past to InitNameList3ds prior to use.

Returns nothing.

GetCameraMotionByIndex3ds

***Prototype:*** void GetCameraMotionByIndex3ds(database3ds \*db, short3ds index, kfcamera3ds \*\*kfcam)

Searches the database for the index(th) occurance of a camera node, and copies that node into camera motion structure. If the index is out of range, the structure remains unchanged.

Arguments:

database3ds \*db

The database being searched.

ulong3ds index

The order of occurance of the node in the database. Valid range is 0 to one less than the count returned by GetCameraMotionCount3ds.

kfcamera3ds \*\*kfcam

The camera node structure that receives the settings. The pointer is passed to InitCameraMotion3ds prior to use.

Returns nothing.

GetCameraMotionByName3ds

***Prototype:*** void GetCameraMotionByName3ds(database3ds \*db, char3ds \*name, kfcamera3ds \*\*kfcam)

Seaches the database for a camera node of the given name and copies it into a camera node structure. If a the name is not found, then the structure remains unchanged.

Arguments:

database3ds \*db

The database being searched.

char3ds \*name

The 1 to 10 character name to search for. The search is case sensitive.

kfcamera3ds \*\*kfcam

The camera node structure that receives the settings. The pointer is passed to InitCameraMotion3ds prior to use.

PutCameraMotion3ds

***Prototype:*** void PutCameraMotion3ds(database3ds \*db, kfcamera3ds \*kfcam)

Copies the camera motion settings from the structure into the database. Camera nodes with the same name are replaced with the new settings.

Arguments:

database3ds \*db

The database to copy the camera node into.

kfcamera3ds \*kfcam

The camera node being added to the database.

Returns nothing.

Ambient Light Animation

Kfambient3ds

Ambient Light Animation Structure

Ambient light animation keys for color.

typedef struct {

char3ds name[11];

short3ds parent;

      ushort3ds flags1;

ushort3ds flags2;

      ulong3ds nckeys;

ushort3ds ncflag;

keyheader3ds \*ckeys;

fcolor3ds \*color;

} kfambient3ds;

char3ds name[11]

Ambient light name should always be "$AMBIENT$"

short3ds parent

Parent node. Not used.

ushort3ds flags1

Various flags. Will be replaced with something better.

ushort3ds flags2

ulong3ds nckeys;

Count of color keys in track. Default is 0.

ushort3ds ncflag;

Track options for color track, can be one of the following:

static const ushort3ds TrackSingle3ds = 0;Track does not repeat or loop.

static const ushort3ds TrackLoop3ds = 3;Track loops.

static const ushort3ds TrackRepeat3ds = 3;Track repeats.

Default is TrackSingle3ds.

keyheader3ds \*ckeys;

Array of spline path modifiers for the color keys. Must have same element count as nckeys.

fcolor3ds \*color;

Array of color keys. Element count must be the same number as nckeys.

InitAmbientLightMotion3ds

***Prototype:*** void InitAmbientLightMotion3ds(kfambient3ds \*\*light, ulong3ds nckeys)

Initializes kfambient3ds structure and optionally allocates memory.

Arguments:

kfambient3ds \*\*light

Address of a pointer to a kfambient3ds structure. If pointer is NULL, then memory will be allocated for a new structure and the address stored in the pointer.

ulong3ds nckeys

Allocate memory for specified number of color keys. A value of 0 will leave the color key fields unchanged.

Returns nothing.

ReleaseAmbientLightMotion3ds

***Prototype:*** void ReleaseAmbientLightMotion3ds(kfambient3ds \*\*light);

Frees memory allocated to a kfambient structure.

Arguments:

kfambient3ds \*\*cam

The structured to be freed. After release, the pointer is set to NULL.

Returns nothing.

GetAmbientLightMotionByName3ds

***Prototype:*** void GetAmbientLightMotion3ds(database3ds \*db, kfambient3ds \*\*light)

Seaches the database for a ambient light node and copies it into a ambient light node structure. If a node is not found, then the structure remains unchanged.

Arguments:

database3ds \*db

The database being searched.

kfambient3ds \*\*light

The ambient light node structure that receives the settings. The pointer is passed to InitAmbientLightMotion3ds prior to use.

PutAmbientLightMotion3ds

***Prototype:*** void PutAmbientLightMotion3ds(database3ds \*db, kfambient3ds \*light)

Copies the ambient light motion settings from the structure into the database. Any existing ambient light node is replaced with the new settings.

Arguments:

database3ds \*db

The database to copy the ambient light node into.

kfambient3ds \*light

The ambient light node being added to the database.

Returns nothing.

Mesh Object Animation

Kfrotkey3ds

Rotation Key

typedef struct {

float3ds angle;

float3ds x;

float3ds y;

float3ds z;

} kfrotkey3ds;

float3ds angle

Angle of rotation in degrees. Defaults to 0.0.

float3ds x

X component of rotation axis. Defaults to 0.0.

float3ds y

Y component of rotation axis. Defaults to 0.0.

float3ds z

Z component of rotation axis. Defaults to 0.0.

Kfmorphkey3ds

Morph Object Key

typedef struct {

char3ds name[11];

} kfmorphkey3ds;

char3ds name[11]

Name of morph target mesh. Mesh must have same number of vertices as base object.

Kfmesh3ds

Mesh Animation

typedef struct {

char3ds name[11];

short3ds parent;

ushort3ds flags1;

ushort3ds flags2;

      point3ds pivot;

char3ds instance[11];

point3ds boundmin;

point3ds boundmax;

      ulong3ds npkeys;

short3ds npflag;

keyheader3ds \*pkeys;

point3ds \*pos;

      ulong3ds nrkeys;

short3ds nrflag;

keyheader3ds \*rkeys;

kfrotkey3ds \*rot;

      ulong3ds nskeys;

short3ds nsflag;

keyheader3ds \*skeys;

point3ds \*scale;

      ulong3ds nmkeys;

short3ds nmflag;

keyheader3ds \*mkeys;

kfmorphkey3ds \*morph;

      ulong3ds nhkeys;

short3ds nhflag;

keyheader3ds \*hkeys;

      float3ds msangle;

} kfmesh3ds;

char3ds name[11];

Name of mesh to animate. Special name of "$$$DUMMY" is used for a dummy object. Defaults to empty string.

short3ds parent

Parent index. Obsolete and must be replaced with something that works before the product is released.

ushort3ds flags

Special node flags, any combination of the following:

ushort3ds flags2

Special node flags, any combination of the following:

static const ushortds ShowPath3ds = 1;Display the object motion path.

point3ds pivot

Rotation pivot point for object node. Defaults to 0.0, 0.0, 0.0.

char3ds instance[11]

Name of object node instance.

point3ds boundmin

Point defining the upper, left, and front extents of the object

point3ds boundmax

Point defining the lower, right, and back extents of the object.

ulong3ds npkeys

Count of position keys in track. Default is 0.

ushort3ds npflag

Track options for position track, can be one of the following:

static const ushort3ds TrackSingle3ds = 0;Track does not repeat or loop.

static const ushort3ds TrackLoop3ds = 3;Track loops.

static const ushort3ds TrackRepeat3ds = 3;Track repeats.

Default is TrackSingle3ds.

keyheader3ds \*pkeys

Array of spline path modifiers for the position keys. Must have same element count as npkeys.

point3ds \*pos

Array of position keys. Element count must be the same number as npkeys.

ulong3ds nrkeys

Count of rotation keys in track. Default is 0.

short3ds nrflag

Track options for rotation track. See npflag for their meanings.

keyheader3ds \*rkeys

Array of spline path modifiers for the rotation keys. Must have same element count as nrkeys.

kfrotkey3ds \*rot

Array of rotation keys. Element count must be the same number as nrkeys.

ulong3ds nskeys

Count of scale keys in track. Default is 0.

short3ds nsflag

Track options for scale track. See npflag for their meanings.

keyheader3ds \*skeys

Array of spline path modifiers for the scale keys. Must have same element count as nskeys.

point3ds \*scale

Array of scale keys. Element count must be the same number as nskeys.

ulong3ds nmkeys

Count of morph keys in track. Default is 0.

short3ds nmflag

Track options for morph track. See npflag for their meanings.

keyheader3ds \*mkeys

Array of spline path modifiers for the morph keys. Must have same element count as nmkeys.

kfmorphkey3ds \*morph

Array of morph keys. Element count must be the same number as nmkeys.

ulong3ds nhkeys

Count of hide keys in track. Default is 0.

short3ds nhflag

Track options for hide track. See npflag for their meanings.

keyheader3ds \*hkeys

Array of hide keys. Spline modifiers are ignored. Element count must be the same number as nhkeys.

float3ds msangle

Angle for morph smoothing angle. Corresponds to the Angle field in the Keyframer/Object/Morph/Options/Morph Options dialogue. Defaults to 24.0

InitObjectMotion3ds

***Prototype:*** void InitObjectMotion3ds(kfmesh3ds \*\*obj, ulong3ds npkeys, ulong3ds nrkeys, ulong3ds nskeys, ulong3ds nmkeys, ulong3ds nhkeys);

Initializes kfmesh3ds structure and optionally allocates memory.

Arguments:

kfmesh3ds \*\*obj

Address of a pointer to a kfmesh3ds structure. If pointer is NULL, then memory will be allocated for a new structure and the address stored in the pointer.

ulong3ds npkeys

Allocate memory for specified number of position keys. A value of 0 will leave the position key fields unchanged.

ulong3ds nrkeys

Allocate memory for specified number of rotation keys. A value of 0 will leave the rotation key fields unchanged.

ulong3ds nskeys

Allocate memory for specified number of scale keys. A value of 0 will leave the scale key fields unchanged.

ulong3ds nmkeys

Allocate memory for specified number of morph keys. A value of 0 will leave the morph key fields unchanged.

ulong3ds nhkeys

Allocate memory for specified number of hide keys. A value of 0 will leave the hide key fields unchanged.

Returns nothing.

ReleaseObjectMotion3ds

***Prototype:*** void ReleaseObjectMotion3ds(kfmesh3ds \*\*obj);

Frees memory allocated to a kfmesh3ds structure.

Arguments:

kfmesh3ds \*\*obj

The structured to be freed. After release, the pointer is set to NULL.

Returns nothing.

GetObjectNodeCount3ds

***Prototype:*** ulong3ds GetObjectNodeCount3ds(database3ds \*db);

Scans the database for mesh nodes and returns a count of their number.

Arguments:

database3ds \*db

The database to scan for mesh nodes.

Returns **ulong3ds** as the count of mesh nodes.

GetObjectNodeNameList3ds

***Prototype:*** void GetObjectNodeNameList3ds(database3ds \*db, namelist3ds \*\*list);

Scans the database for mesh nodes and returns a list of their names.

Arguments:

database3ds \*db

The database to scan for mesh nodes.

namelist3ds \*\*list

The list of mesh node names found. Pointer gets past to InitNameList3ds prior to use.

Returns nothing.

GetObjectMotionByIndex3ds

***Prototype:*** void GetObjectMotionByIndex3ds(database3ds \*db, short3ds index, kfmesh3ds \*\*obj)

Searches the database for the index(th) occurance of a mesh node, and copies that node into mesh motion structure. If the index is out of range, the structure remains unchanged.

Arguments:

database3ds \*db

The database being searched.

ulong3ds index

The order of occurance of the node in the database. Valid range is 0 to one less than the count returned by GetObjectMotionCount3ds.

kfmesh3ds \*\*obj

The mesh node structure that receives the settings. The pointer is passed to InitObjectMotion3ds prior to use.

Returns nothing.

GetObjectMotionByName3ds

***Prototype:*** void GetObjectMotionByName3ds(database3ds \*db, char3ds \*name, kfmesh3ds \*\*obj);

Seaches the database for a mesh node of the given name and copies it into a mesh node structure. If a the name is not found, then the structure remains unchanged.

Arguments:

database3ds \*db

The database being searched.

char3ds \*name

The 1 to 10 character name to search for. The search is case sensitive.

kfmesh3ds \*\*obj

The mesh node structure that receives the settings. The pointer is passed to InitObjectMotion3ds prior to use.

PutObjectMotion3ds

***Prototype:*** void PutObjectMotion3ds(database3ds \*db, kfmesh3ds \*obj);

Copies the mesh motion settings from the structure into the database. Object nodes with the same name are replaced with the new settings.

Arguments:

database3ds \*db

The database to copy the mesh node into.

kfmesh3ds \*obj

The mesh node being added to the database.

Returns nothing.

Omnilight Animation

Kfomni3ds

Omnilight Animation Tracks

typedef struct {

char3ds name[11];

short3ds parent;

      ushort3ds flags1;

ushort3ds flags2;

      ulong3ds npkeys;

ushort3ds npflag;

keyheader3ds \*pkeys;

point3ds \*pos;

      ulong3ds nckeys;

ushort3ds ncflag;

keyheader3ds \*ckeys;

fcolor3ds \*color;

} kfomni3ds;

char3ds name[11];

Name of omnilight to animate. S

short3ds parent

Parent index. Obsolete and must be replaced with something that works before the product is released.

ushort3ds flags

Special node flags, any combination of the following:

ushort3ds flags2

Special node flags, any combination of the following:

static const ushortds ShowPath3ds = 1;Display the object motion path.

ulong3ds npkeys

Count of position keys in track. Default is 0.

ushort3ds npflag

Track options for position track, can be one of the following:

static const ushort3ds TrackSingle3ds = 0;Track does not repeat or loop.

static const ushort3ds TrackLoop3ds = 3;Track loops.

static const ushort3ds TrackRepeat3ds = 3;Track repeats.

Default is TrackSingle3ds.

keyheader3ds \*pkeys

Array of spline path modifiers for the position keys. Must have same element count as npkeys.

point3ds \*pos

Array of position keys. Element count must be the same number as npkeys.

ulong3ds nckeys;

Count of color keys in track. Default is 0.

ushort3ds ncflag;

Track options for color track, can be one of the following. See npflag for their meanings.

keyheader3ds \*ckeys;

Array of spline path modifiers for the color keys. Must have same element count as nckeys.

fcolor3ds \*color;

Array of color keys. Element count must be the same number as nckeys.

InitOmniLightMotion3ds

***Prototype:*** void InitOmniLightMotion3ds(kfomni3ds \*\*light, ulong3ds npkeys, ulong3ds nckeys)

Initializes kfomni3ds structure and optionally allocates memory.

Arguments:

kfomni3ds \*\*obj

Address of a pointer to a kfomni3ds structure. If pointer is NULL, then memory will be allocated for a new structure and the address stored in the pointer.

ulong3ds npkeys

Allocate memory for specified number of position keys. A value of 0 will leave the position key fields unchanged.

ulong3ds nckeys

Allocate memory for specified number of color keys. A value of 0 will leave the color key fields unchanged.

Returns nothing.

ReleaseOmniLightMotion3ds

***Prototype:*** void ReleaseOmniLightMotion3ds(kfomni3ds \*\*light)

Frees memory allocated to a kfomni3ds structure.

Arguments:

kfomni3ds \*\*light

The structured to be freed. After release, the pointer is set to NULL.

Returns nothing.

GetOmniLightNodeCount3ds

***Prototype:*** ulong3ds GetOmniLightNodeCount3ds(database3ds \*db)

Scans the database for omni light nodes and returns a count of their number.

Arguments:

database3ds \*db

The database to scan for mesh nodes.

Returns **ulong3ds** as the count of mesh nodes.

GetOmniLightNodeNameList3ds

***Prototype:*** void GetOmniLightNodeNameList3ds(database3ds \*db, namelist3ds \*\*list)

Scans the database for omni light nodes and returns a list of their names.

Arguments:

database3ds \*db

The database to scan for omni light nodes.

namelist3ds \*\*list

The list of omni light node names found. Pointer gets past to InitNameList3ds prior to use.

Returns nothing.

GetOmniLightMotionByIndex3ds

***Prototype:*** void GetOmniLightMotionByIndex3ds(database3ds \*db, short3ds index, kfomni3ds \*\*light)

Searches the database for the index(th) occurance of a omni light node, and copies that node into omni light motion structure. If the index is out of range, the structure remains unchanged.

Arguments:

database3ds \*db

The database being searched.

ulong3ds index

The order of occurance of the node in the database. Valid range is 0 to one less than the count returned by GetOmniLightMotionCount3ds.

kfomni3ds \*\*light

The omni light node structure that receives the settings. The pointer is passed to InitOmniLightMotion3ds prior to use.

Returns nothing.

GetOmniLightMotionByName3ds

***Prototype:*** void GetOmniLightMotionByName3ds(database3ds \*db, char3ds \*name, kfomni3ds \*\*light)

Seaches the database for a omni light node of the given name and copies it into a omni light node structure. If a the name is not found, then the structure remains unchanged.

Arguments:

database3ds \*db

The database being searched.

char3ds \*name

The 1 to 10 character name to search for. The search is case sensitive.

kfomni3ds \*\*light

The omni light node structure that receives the settings. The pointer is passed to InitOmniLightMotion3ds prior to use.

PutOmniLightMotion3ds

***Prototype:*** void PutOmniLightMotion3ds(database3ds \*db, kfomni3ds \*light)

Copies the omni light motion settings from the structure into the database. Object nodes with the same name are replaced with the new settings.

Arguments:

database3ds \*db

The database to copy the omni light node into.

kfomni3ds \*light

The omni light node being added to the database.

Returns nothing.

Spotlight Animation

Kfspot3ds

Spotlight Animation Settings

typedef struct {

char3ds name[11];

short3ds parent;

      ushort3ds flags1;

ushort3ds flags2;

      ulong3ds npkeys;

ushort3ds npflag;

keyheader3ds \*pkeys;

point3ds \*pos;

      ulong3ds nckeys;

ushort3ds ncflag;

keyheader3ds \*ckeys;

fcolor3ds \*color;

      ulong3ds nhkeys;

ushort3ds nhflag;

keyheader3ds \*hkeys;

float3ds \*hot;

      ulong3ds nfkeys;

ushort3ds nfflag;

keyheader3ds \*fkeys;

float3ds \*fall;

      ulong3ds nrkeys;

ushort3ds nrflag;

keyheader3ds \*rkeys;

float3ds \*roll;

      short3ds tparent;

ulong3ds ntkeys;

ushort3ds ntflag;

keyheader3ds \*tkeys;

point3ds \*tpos;

ushort3ds tflags1;

ushort3ds tflags2;

} kfspot3ds;

char3ds name[11];

Name of spot light to animate.

short3ds parent

Parent index. Obsolete and must be replaced with something that works before the product is released.

ushort3ds flags

Special node flags, any combination of the following:

ushort3ds flags2

Special node flags, any combination of the following:

static const ushortds ShowPath3ds = 1;Display the object motion path.

ulong3ds npkeys

Count of position keys in track. Default is 0.

ushort3ds npflag

Track options for position track, can be one of the following:

static const ushort3ds TrackSingle3ds = 0;Track does not repeat or loop.

static const ushort3ds TrackLoop3ds = 3;Track loops.

static const ushort3ds TrackRepeat3ds = 3;Track repeats.

Default is TrackSingle3ds.

keyheader3ds \*pkeys

Array of spline path modifiers for the position keys. Must have same element count as npkeys.

point3ds \*pos

Array of position keys. Element count must be the same number as npkeys.

ulong3ds nckeys;

Count of color keys in track. Default is 0.

ushort3ds ncflag;

Track options for color track, can be one of the following. See npflag for their meanings.

keyheader3ds \*ckeys;

Array of spline path modifiers for the color keys. Must have same element count as nckeys.

fcolor3ds \*color;

Array of color keys. Element count must be the same number as nckeys.

ulong3ds nhkeys;

Count of hot spot keys in track. Default is 0.

ushort3ds nhflag;

Track options for hot spot track, can be one of the following. See npflag for their meanings.

keyheader3ds \*hkeys;

Array of spline path modifiers for the hot spot keys. Must have same element count as nhkeys.

float3ds \*hot;

Array of hot spot keys. Element count must be the same number as nhkeys.

ulong3ds nfkeys;

Count of falloff keys in track. Default is 0.

ushort3ds ncflag;

Track options for falloff track, can be one of the following. See npflag for their meanings.

keyheader3ds \*fkeys;

Array of spline path modifiers for the falloff keys. Must have same element count as nfkeys.

fcolor3ds \*fall;

Array of falloff keys. Element count must be the same number as nfkeys.

ulong3ds nrkeys

Number of roll keys in the track. Default is 0.

ushort3ds nrflag

Track options for roll track. Se npflag for their meanings.

keyheader3ds \*rkeys

Array of spline modifiers for spot roll. Element count must be the same number as nrkeys.

float3ds \*roll

Array of spot roll keys. Element count must be the same number as nrkeys.

short3ds tparent

Parent index for spot light target.

ulong3ds ntkeys

Number of track position keys. Default is 0.

ushort3ds ntflag

Track options for target keys. See npflag for thier meanings.

keyheader3ds \*tkeys

Array of spline path modifiers for target keys. Element count must be the same number as ntkeys.

point3ds \*tpos

Array of target position keys. Element count must be the same number as ntkeys.

ushort3ds tflags1

Special node flags.

ushort3ds tflags2

Special node flags.

InitSpotLightMotion3ds

***Prototype:*** void InitSpotLightMotion3ds(kfspot3ds \*\*spot, ulong3ds npkeys, ulong3ds nckeys, ulong3ds nhkeys, ulong3ds nfkeys, ulong3ds nrkeys, ulong3ds ntkeys);

Initializes kfspot3ds structure and optionally allocates memory.

Arguments:

kfspot3ds \*\*spot

Address of a pointer to a kfspot3ds structure. If pointer is NULL, then memory will be allocated for a new structure and the address stored in the pointer.

ulong3ds npkeys

Allocate memory for specified number of position keys. A value of 0 will leave the position key fields unchanged.

ulong3ds nckeys

Allocate memory for specified number of color keys. A value of 0 will leave the color key fields unchanged.

ulong3ds nhkeys

Allocate memory for specified number of hot spot keys. A value of 0 will leave the hot spot key fields unchanged.

ulong3ds nfkeys

Allocate memory for specified number of falloff keys. A value of 0 will leave the falloff key fields unchanged.

ulong3ds nrkeys

Allocate memory for specified number of roll keys. A value of 0 will leave the roll key fields unchanged.

ulong3ds ntkeys

Allocate memory for specified number of target position keys. A value of 0 will leave the target position key fields unchanged.

Returns nothing.

ReleaseSpotLightMotion3ds

***Prototype:*** void ReleaseSpotLightMotion3ds(kfspot3ds \*\*spot);

Frees memory allocated to a kfspot3ds structure.

Arguments:

kfspot3ds \*\*spot

The structured to be freed. After release, the pointer is set to NULL.

Returns nothing.

GetSpotLightNodeCount3ds

***Prototype:*** long3ds GetSpotLightNodeCount3ds(database3ds \*db);

Scans the database for spotlight nodes and returns a count of their number.

Arguments:

database3ds \*db

The database to scan for spotlight nodes.

Returns **ulong3ds** as the count of mesh nodes.

GetSpotLightNodeNameList3ds

***Prototype:*** void GetSpotLightNodeNameList3ds(database3ds \*db, chunklist3ds \*\*list);

Scans the database for spotlight nodes and returns a list of their names.

Arguments:

database3ds \*db

The database to scan for spotlight nodes.

namelist3ds \*\*list

The list of spotlight node names found. Pointer gets past to InitNameList3ds prior to use.

Returns nothing.

GetSpotLightMotionByIndex3ds

***Prototype:*** void GetSpotLightMotionByIndex3ds(database3ds \*db, ulong3ds index, kfspot3ds \*spot);

Searches the database for the index(th) occurance of a spotlight node, and copies that node into spotlight motion structure. If the index is out of range, the structure remains unchanged.

Arguments:

database3ds \*db

The database being searched.

ulong3ds index

The order of occurance of the node in the database. Valid range is 0 to one less than the count returned by GetSpotLightMotionCount3ds.

kfspot3ds \*\*spot

The spotlight node structure that receives the settings. The pointer is passed to InitSpotLightMotion3ds prior to use.

Returns nothing.

GetSpotLightMotionByName3ds

***Prototype:*** void GetSpotLightMotionByName3ds(database3ds \*db, char3ds \*name, kfspot3ds \*\*spot);

Seaches the database for a spotlight node of the given name and copies it into a spotlight node structure. If a the name is not found, then the structure remains unchanged.

Arguments:

database3ds \*db

The database being searched.

char3ds \*name

The 1 to 10 character name to search for. The search is case sensitive.

kfspot3ds \*\*spot

The spotlight node structure that receives the settings. The pointer is passed to InitSpotLightMotion3ds prior to use.

PutSpotLightMotion3ds

***Prototype:*** void PutSpotLightMotion3ds(database3ds \*db, kfspot3ds \*spot);

Copies the spotlight motion settings from the structure into the database. Object nodes with the same name are replaced with the new settings.

Arguments:

database3ds \*db

The database to copy the spotlight node into.

kfspot3ds \*spot

The spotlight node being added to the database.