STRIKE EAGLE EYE

REFERENCE MANUAL

DAVID O'SHEA

STRIKE EAGLE EYE

STRIKE EAGLE EYE: REFERENCE MANUAL

David O'Shea

Version 0.0038

TABLE OF CONTENTS

LIST OF TABLES

INTRODUCTION

This document describes Strike Eagle Eye™, a set of open-source tools for the computer game F-15 Strike Eagle III™ developed and published by MicroProse®. Initially, this document will only cover the file formats used by the game, but over time a description of some of the internal data stored in memory by the game will be added, followed by information on the open-source tools developed to interface with the game and its files.

WARNING: The information in this document is a work in progress. It is possible that the information in this document only applies to certain versions or releases of the game, certain configurations of those versions or releases or different hardware or software platforms. It is possible that attempting to use this information with other versions or releases of the game, other configurations or other hardware or software platforms will result in damage to the game or your computer, so use this information at your own risk.

ABOUT THIS DOCUMENT

The latest release version of this document is always available at the Strike Eagle Eye website [http://strikeeagleeye.sourceforge.net/]. The website also provides a link to the Strike Eagle Eye project page on SourceForge [http://sourceforge.net/projects/strikeeagleeye/] which provides details of the Subversion repository used to configure this document and accepts bug reports regarding this document.

This document was created in DocBook XML [http://www.docbook.org/] format using the Visual Editor for XML (Vex) [http://vex.sourceforge.net/] plugin for the Eclipse Platform [http://www.eclipse.org/] and then transformed into HTML for publishing on the World Wide Web using The SAXON XSLT Processor from Michael Kay [http://saxon.sourceforge.net/] and the DocBook XSL stylesheets [http://wiki.docbook.org/topic/DocBookXslStylesheets]. The transformation was directed by Apache Ant [http://ant.apache.org/].

The following free open-source software was used to gather the information contained in this document: Bed [http://bedlinux.tripod.com/] binary editor, bochs [http://bochs.sourceforge.net/] PC emulator, DOSBox [http://www.dosbox.com/], Eigen [http://eigen.tuxfamily.org/] C++ template library for linear algebra, GIMP - The GNU Image Manipulation Program [http://www.gimp.org/], GNU Emacs [http://www.gnu.org/software/emacs/], HT Editor [http://hte.sourceforge.net/] hex editor, The Mesa 3D Graphics Library [http://www.mesa3d.org/], MinGW - Minimalist GNU for Windows - and MSYS [http://www.mingw.org/], the Python programming language [http://www.python.org/], SDL - Simple DirectMedia Layer [http://www.libsdl.org/], VBinDiff - Visual Binary Diff [http://www.cjmweb.net/vbindiff/].

CHAPTER 1. CONCEPTS

I. CO-ORDINATES

The following co-ordinate system is used in this document, with the axes assigned based on the order that co-ordinates are stored in the data files:

- the X axis is the east-west axis, with greater X co-ordinate values being further east;
- the Y axis is the vertical axis, with greater Y co-ordinate values being at higher altitudes; and
- the Z axis is the north-south axis, with greater Z co-ordinate values being further south.

In many of the game's data files, 2D coordinates are used, with no Y co-ordinate value specified.

II. OBJECTS

An object consists of a 3D model which can appear at any number of locations on the ground, plus associated attributes and behaviour. The model may appear flat, or consist of a number of visually separate 3D parts, such as a number of unconnected buildings, but still consists of a single 3D model[1].

Examples of objects include air traffic control towers, patrol boats, oil rigs, SAM radar sites and camels.

There may be multiple instances of each object in the world, each of which:

- generally appears on the ground (the section called “Object placement information” describes how each tile in the Level 1 tile set has a list of objects associated with it; note that they are placed using only horizontal coordinates, without a facility to specify altitude);
- generates a radar return;
- can often be destroyed by the player (FIXME: does the section called “Object and Sub-Object Data” have hit point information to control this?); and
- may be one of the player's objectives.

Each object:

- is referred to within the game data files by its index into the list of objects (FIXME: is object names the main thing to determine how many objects there are?);
- has a name, which is displayed when the player destroys it (see the section called “Object Names”);
- has additional data, such as the number of points received for destroying the object, associated with it in (see the section called “Object and Sub-Object Data”);
- has 3D models of two different detail levels associated with it (see the section called “Main `.3DX` File Names” for details of which files contain which level of detail, and the section called “Model Level of Detail Distances” for per-object distance thresholds); and
- has a scaling factor associated with its 3D model (see the section called “Model Downscaling Factors”).

[1] Presumably due to limitations on 3D model sizes, battleships consist of two separate models, one for the superstructure and one for the rest of the ship, but technically these are two separate objects, "Battleship" and "BB Top".

When an object visually consists of separate parts, those parts may be sub-objects which have their own names and which may be destroyed separately; the section called “Object and Sub-Object Data” describes how sub-objects are associated with objects. When the player destroys an object (or part of it if it has sub-objects), the same 3D model is still displayed at the same location, but some parts of the model may be rendered differently to simulate damage (see the section called “Condition Prefix Sub-Element”).

ATTACKING THE PLAYER

Whilst objects can be attacked by the user, objects other than AAA guns are passive and do not attack the user or paint them using radar.

As described in the section called “Enemy/Home Plate Data”, the locations at which TEWS radar threats appear, and from which SAMs are launched, are specified in terms of coordinates within specific tiles in the Level 1 tile set. Typically, these will be set so that the location within a level 1 tile at which threats appear and SAMs are launched coincides with the location of a SAM object as specified in the section called “Object placement information”, but the SAM behaviour is not actually associated with a particular object in the data files, and it would be possible for there to be no object at the location specified in the section called “Enemy/Home Plate Data”.

In the case of AAA, placement of an AAA gun object (with the correct flag set as described in the section called “Object and Sub-Object Data”) is all that is required for AAA to be automatically be fired from the gun when the player is flying nearby. In order to cause an AAA-related radar to appear in the TEWS, the same scheme as described above is required (i.e. see the section called “Enemy/Home Plate Data”).

CHAPTER 2. F-15 STRIKE EAGLE III FILE FORMATS

I. WORLD (.WLD) FILE

A world file contains the following information about a theater of war:

- a square bitmap for the surface of the earth in the theater, with each edge measuring one mebipixel (i.e. 1024^2) for a total area of one tebipixel (i.e. 1024^4), but utilising hierarchical levels of heavily repeated patterns to consume only 144 kibibytes on disk;
- ground placement information for Objects ; and
- the VGA palette used for the bitmap plus all other video output by the game while the world is visible on the screen (i.e. while in the cockpit).

The game ships with three world files, `DS0.WLD`, `KO0.WLD` and `PA0.WLD`. Each time a mission is created for a theater (FIXME:xref), a modified version of the `.WLD` file without the `0` (e.g. `DS.WLD`) is created[1]. In these modified versions of the world files, some enemy units are moved, for example SAM sites can appear and disappear (FIXME: or are they moved? does the unit count always stay the same?) and naval units move.

TABLE 2.1. WORLD (.WLD) FILE FORMAT

Offset (bytes)	Length (bytes)	Description
0x0	0x10000	Level 0 (topmost) tile (256 * 256 * 1 byte)
0x10000	0x6000	Level 2 tile set (96 tiles of size 16 * 16 * 1 byte)
0x16000	0x6000	Level 1 tile set (96 tiles of size 16 * 16 * 1 byte)
0x1C000	0x8000	Level 3 tile set (128 tiles of size 16 * 16 * 1 byte)
0x24000	0x2400	Object placement information (96 lists of 16 objects with 6 bytes per object)
0x26400	0x300	VGA palette (256 entries with 3 bytes per entry)
0x26700		End-of-file

[1] If a modified version already exists (i.e. a mission has previously been created for the theater and the user has not removed the modified version), it is not clear whether the new modified version is based on the existing modified version or the base version shipped with the game.

CO-ORDINATES

All tiles used in the world bitmap are two-dimensional row-major arrays of bytes. The arrays use a natural layout with the row index forming the Z co-ordinate and the column index forming the X co-ordinate. The axes are defined in Section I, “Co-ordinates”; note that Y co-ordinates are not referenced in this file.

LEVEL 0 (TOPMOST) TILE

As there are a number of levels, each containing multiple tiles, it is useful to consider this top level of the bitmap to be a single tile, although it is never repeated. This tile is a 256 * 256 array of bytes with values less than 96, with each byte being an index into the level 1 tile set. The byte also forms an index for the Object placement information.

In the theaters supplied with the game, it can often be observed that a large number of bytes in this level 0 tile have the same value. This is because a large portion of these theaters is water or desert, and of this, a large portion will probably never be flown over by the player due to the relatively large size of the theater compared to the area in which missions take place.

In the Korea theater, it is observed that it is **not possible to fly or slew** into the four northern rows, the six southern rows, the two western columns or the four eastern columns of this tile. When the aircraft is near one of these edges, at certain zoom levels, it is possible to see wrapping in the *TSD* (moving map), for example when the aircraft is as far north as possible, the top row of pixels in the TSD is from the southernmost part of the world.

LEVEL 1 TILE SET

NOTE: As shown in Table 2.1, “World (`.WLD`) File Format”, this tile set *follows* the level 2 tile set in the file.

This contains a set of 96 tiles, each of which is a 16 * 16 array of bytes with values less than 96, with each byte being an index into the level 2 tile set. The index of the tile within *this* tile set also forms an index for the Object placement information - objects are placed on the surface of the earth based on which of the level 1 tiles covers that portion of the earth - and the FIXME:reference(data and mission) files also refer to these indices.

In the theaters supplied with the game, the level 1 tiles can generally be grouped into the following categories:

- open ocean and plain ground in various patterns;
- coastlines running east-west, north-south, diagonally and with various corners and bays;
- ground with creeks and rivers running east-west, north-south and with various curves and forks;
- ground with roads running east-west, north-south, diagonally and with various corners and intersections;
- ground with a road intersecting with a creek or river where the object placement information would cause a bridge object to appear at the intersection;
- ground with roads for interconnection within a SAM site where the object placement information would cause a SAM radar and launchers to appear at appropriate locations;
- ground with many roads and a dark gray appearance simulating a town or city where the object placement information would cause buildings and possibly SAM/ AAA defences to appear;
- ground with a runway and taxiways and perhaps roads and/or part of a town or city where the object placement information would cause a control tower and hangars to appear at appropriate locations, possibly along with other buildings and SAM/ AAA defences; and

- ground with or without a road where the object placement information would cause tents or tanks to appear to simulate infantry or amour concentrations.

These tiles are placed adjacent to each other in the level 0 tile such that coastlines, creeks, rivers, roads, etc. are aligned properly.

In the game, if a *TSD* is zoomed out as far as possible and then zoomed in three times, the slew keys will move the length or width of a level 1 tile with each step. When zoomed in as far as possible, it requires 16 slew steps to move the length or width of a level 1 tile.

LEVEL 2 TILE SET

This contains a set of 96 tiles, each of which is a 16 * 16 array of bytes with values less than 128, with each byte being an index into the level 3 tile set.

In the theaters supplied with the game, the level 2 tiles are often smaller versions of the level 1 tiles and are repeated to generate the level 1 tiles. For example, a level 2 tile may contain a short section of road and be repeated in a level 1 tile to form a longer stretch of road.

LEVEL 3 TILE SET

This contains a set of 128 tiles, each of which is a 16 * 16 array of bytes with any valid byte value (i.e. values less than 256), with each byte being a VGA palette entry index that is written into video RAM by the game.

In the theaters supplied with the game, the level 3 tiles are often solid colours, stippled colours giving a textured effect, or geometrical shapes - generally diagonally-split tiles with one colour in one half and one in another used to form diagonal roads or coastlines.

In the Korea theater, it is observed that not all of the level 3 tiles are referenced by the level 2 tiles (FIXME: verify that all the higher-level tiles are referenced since we are inferring that with this statement).

Not all of the palette entries should be referenced by the the level 3 tiles, and some palette entries appear to be treated specially. In the Korea theater for example, it has been noted that when a level 3 tile contains the byte 0x10, which corresponds to a dark red colour in the palette, the value used on-screen as seen in a screen shot is actually 0x7F, which corresponds to a medium green colour in the palette. It appears that other palette entries are also treated in this way.

OBJECT PLACEMENT INFORMATION

This forms an array of 96 object lists, indexed by the index into the level 1 tile set, specifying the Objects to be placed on the earth's surface in the area covered by any level 1 tile with the given index. Each object list consists of 16 object specifiers, so each level 1 tile can have up to 16 3D objects in it, although some 3D objects may be compound objects with sub-parts. Each entry in the object list is 6 bytes long and is of the following format:

TABLE 2.2. OBJECT LIST ENTRY FORMAT

Offset (bytes)	Length (bytes)	Description
0x0	0x2	Object index (see Section II, “Objects”); set to 0xFFFF if this object list entry should be ignored
0x2	0x2	X co-ordinate at which the object should be placed
0x4	0x2	Z co-ordinate at which the object should be placed

All values in the entry are *little-endian* . If the user attacks and destroys the object, the

`[Friendly]` *`object name`* `destroyed by` *`weapon name`*

message is shown using the name of the object whose index is specified here, regardless of what terms were used in any mission objective, although (FIXME: see `.MIS` file - if tile and index into object list are right it will say "PRIMARY ACHIEVED" just before the message about the destruction or something like that). Co-ordinates are *world co-ordinates* relative to the level 1 tile in which the object is being placed. This means that they reference a square grid with each dimension being 16 * 16 * 16 = 4096, so the four-most significant bits are normally 0, the next four are *tile co-ordinates* within the level 1 tile, the next four are the tile co-ordinates within the level 2 tile and the four least-significant bits are the tile co-ordinates within the level 3 tile. In some cases, the four most-significant bits are non-zero. An object can be placed on the eastern or southern edge of the tile by setting the relevant co-ordinate to 0x1000 (4096). A co-ordinate or co-ordinates greater than 0x1000 may be used to cause an object to appear within the area covered by a level 1 tile to the east and/or south of the tile to which the object list applies (FIXME: does game use 0x1000 or more in any base maps?), but note that when the player attacks the object, the object appears to not receive splash damage from bombs landing in level 1 tiles that are not adjacent to the tile whose object list the object is specified in. For example, given a level 1 tile whose tile co-ordinates in the level 0 tile are (0, 0), if the co-ordinates for an object in that tile's object list are (0x2000, 0), i.e. the object lies on the boundary between the level 1 tiles whose level 0 tile co-ordinates are (0, 1) and (0, 2), then if the player attempts to bomb the object and the bomb lands in level 1 tile (0, 1), the object may be destroyed, but if the bomb lands in level 1 tile (0, 2), it will not be destroyed. It is still possible to destroy an object with a co-ordinate slightly greater than 0x2000, e.g. 0x2010, but since it requires the user to not bomb the object directly but instead bomb the edge of the adjacent tile, the user is likely to have difficulty in destroying the object, as they would for a value less than but approaching 0x2000, so the co-ordinate should probably not approach 0x2000 if in fact there is good reason to use a co-ordinate greater than 0x1000. For co-ordinate values significantly larger than 0x2000, e.g. 0x3100, the object begins to exhibit strange effects such as not being visible from all angles.

VGA PALETTE

This is a standard VGA palette with entries for 256 colours, with each 3-byte entry consisting of red, green and blue components in that order, and each component being a byte in the range 0 (minimum intensity) through 63 (maximum intensity) inclusive[2].

As noted in the section called “Level 3 tile set”, some palette entries cannot be referenced by the level 3 tiles. Some palette entries are used for the interior and exterior of the aircraft. It is expected that there would be two palette entries, one used for the aircraft's exhaust that would be a light blue colour and one used for the aircraft's tracers, missile exhausts and burning objects which would be red/orange/yellow, and that each of these palette entries would be cycled rapidly by the game to give a flame-like effect. It is likely that none of these palette entries may be referenced by the level 3 tiles.

II. ENEMY (.NMY) FILE

The enemy file is a square bitmap covering the surface of the earth in the theater indicating which parts of the theater are controlled by the enemy. When the user destroys an object, this bitmap is used to determine whether the object is considered "Friendly" or not. This bitmap also determines which of the level 1 tile indices specified in (FIXME: reference `.DAT` file's enemy SAM/airbase information - specifically assume that this file is used to determine which tile indices referenced in `??0.DAT` should be randomly selected from when generating `??.DAT`) should "attack" the user (via radar "spikes", launching SAMs or launching aircraft from a runway) and which of the AAA guns specified in the Object placement information for the applicable level 1 tiles should fire on the user; the non-enemy SAM/AAA/airbases are completely passive.

[2]The range 0 through 63 corresponds to 6 bits of resolution per colour channel as is supported by the VGA standard.

This file is 512 bytes long and can be viewed as a 64 * 64 row-major array of bits covering the theater such that one bit in the array corresponds to a 4 * 4 block of level 1 tiles in the level 0 tile. Each row consists of 8 bytes, for a total of 64 bits, where the most-significant bit of the first byte has index 0 and the least-significant bit of the last byte has index 63. The file consists of 64 such rows, with the first having index 0 and the last having index 63. (FIXME: reference co-ordinates section which says X co-ordinate increases to the east, etc.)

A bit in the array is set to zero if the corresponding 4 * 4 block of level 1 tiles in the level 0 tile is enemy-controlled, and set to one if the block is friendly.

III. THEATER DATA (.DAT) FILE

This file contains various information about the theater. As for the `.WLD` file (see Section I, "World (`.WLD`) File"), the game ships with three of these files, `DS0.DAT`, `KO0.DAT` and `PA0.DAT`, and each time a mission is created for a theater, a modified version of the file without the `0` (e.g. `DS.DAT`) is created. However, in the case of the `.DAT` file, there is some difference in the format of the data in the `??0.DAT` and `??.DAT` files. This difference applies only to the enemy/home plate data and is described in the section called "Co-ordinates and Tile Indices in Enemy/Home Plate Records". It appears that it is only in this section that the `??0.DAT` and `??.DAT` files differ.

TABLE 2.3. THEATER DATA (.DAT) FILE FORMAT

Length (bytes)	Description
`6 + 10 * n`	See the section called "Enemy/Home Plate Data"
`2 + 38 * (m + 2)`	See the section called "Unknown 1"
variable	See the section called "Object Names"
2	Number of objects (equal to the sum of the numbers of models in `??.3DX` and `??MOB.3DX`)
2	Number of sub-objects (equal to the number of object names minus the number of objects)
`12 * (number_of_objects + number_of_sub_objects)`	See the section called "Object and Sub-Object Data"
`number_of_objects`	See the section called "Model Downscaling Factors"
`2 * number_of_objects`	See the section called "Model Level of Detail Distances"
96	See the section called "Level 1 Tile Lights Flags"
256	See the section called "TSD Palette"
56	Unknown 3
512	See the section called "Model Palette"
`2 + 2 * k`	See the section called "Enemy Aircraft Types"
2	Unknown 5
10	See the section called "Latitude/Longitude Conversion Data"

ENEMY/HOME PLATE DATA

The enemy/home plate data specifies the locations of:

- enemy SAM and/or radar sites;
- enemy airbases; and
- (friendly) *home plates*.

This data begins at offset `0x0` in the file and consists of a header whose format is described in Table 2.4, “Theater Data Enemy/Home Plate Header Format” followed by a list of records whose format is described in Table 2.6, “Theater Data Enemy/Home Plate Record Format” . The header defines the number of records in the list and, as shown in Table 2.5, “Theater Data Enemy/Home Plate Record Indices” , also divides the list into a number of distinct, contiguous groups based on their indices.

TABLE 2.4. THEATER DATA ENEMY/HOME PLATE HEADER FORMAT

Offset (bytes)	Length (bytes)	Description
`0x0`	`0x2`	`nRecords`, the number of enemy/home plate records following this header
`0x2`	`0x2`	`firstEnemyAirbase`, the index of the first record corresponding to an enemy airbase
`0x4`	`0x2`	`firstHomePlate`, the index of the first record corresponding to a *home plate*

TABLE 2.5. THEATER DATA ENEMY/HOME PLATE RECORD INDICES

First index	Last index	Co-ordinates relative to level 1 tile in `??0.DAT`[a]	Description
`0`	`2`	n/a	all fields in these records are set to `0`
`2`	`firstEnemyAirbase - 1`	yes	enemy SAM and/or radar sites
`firstEnemyAirbase`	`firstHomePlate - 1`	yes	enemy airbases
`firstHomePlate`	`nRecords - 1`	no	*home plates*

[a]See the section called “Co-ordinates and Tile Indices in Enemy/Home Plate Records” .

NOTE: Through testing, it has been determined that the game (at least in version 4108.01) supports the case where `firstEnemyAirbase` is equal to `firstHomePlate` , i.e. where there are no enemy airbase records. This is supported in both of the following cases:

- where there are one or more enemy airbase records in the `??0.DAT` file, but the referenced tile indices are designated as friendly in the `??.NMY` file, in which case the game determines that there are no enemy airbases and writes no records for these to the `??.DAT` file; and
- where there are no enemy airbase records in the `??0.DAT` file and hence the game writes none to the `??.DAT` file.

TABLE 2.6. THEATER DATA ENEMY/HOME PLATE RECORD FORMAT

Offset (bytes)	Length (bytes)	Description
0x0	0x2	X co-ordinate (see the section called "Co-ordinates and Tile Indices in Enemy/Home Plate Records")
0x2	0x2	Z co-ordinate (as above)
0x4	0x2	SAM and/or radar type (see Table 2.7, "Enemy SAM and/or Radar Types"), set to 0 for records not corresponding to a SAM and/or radar site
0x6	0x1	set to 0 if this record corresponds to a SAM and/or radar site, 1 for a friendly/enemy airbase/home plate
0x7	0x1	home plate type (FIXME describe), set to 0 for records not corresponding to a home plate
0x8	0x1	for SAM and/or radar and enemy airbase records when they appear in the ??0.DAT file, non-zero tile index in level 1 tile set to which this record applies, otherwise set to 0
0x9	0x1	padding byte always set to 0

TABLE 2.7. ENEMY SAM AND/OR RADAR TYPES

Value	*TEWS* radar indicator[a]	SAM type launched	Notes
0x0	n/a	n/a	this value indicates no SAM and/or radar
0x1	9	*none*	GCI radar
0x2	9	*none*	GCI radar
0x3	1	"SA-2"	
0x4	1	"SA-3"	not used [b]
0x5	2	"SA-5"	
0x6	2	"Hawk"	
0x7	3	"SA-6"	
0x8	4	"SA-8B"	
0x9	8	"SA-14"	
0xA	4	"SKYGRD"	
0xB	4	"ROLAND"	

Value	*TEWS* radar indicator[a]	SAM type launched	Notes
0xC	*none*	"SA-7"	
0xD	*none*	"SA-13"	
0xE	8	*none*	not used [b]

[a]The number shown within a box in the *authentic mode* TEWS display to indicate the radar type.

[b]These values are not used in any of the ??0.DAT files distributed with version 4108.01 of the game or the DS.DAT file distributed with the version 4108.03 patch.

CO-ORDINATES AND TILE INDICES IN ENEMY/HOME PLATE RECORDS

The enemy/home plate record may specify the location(s) that it refers to in one of two ways:

1. specifying a set of locations by specifying a non-zero level 1 tile index and *world co-ordinates* relative to the level 1 tile; or
2. specifying a single location by specifying the *home plate co-ordinates* of the location (with the level 1 tile index set to 0).

The first scheme is used for enemy SAM and/or radar and enemy airbase records in ??0.DAT . The second scheme is used for home plate records in ??0.DAT and for all record types in ??.DAT .

NOTE: When the first method is used in the ??0.DAT file, it has not yet been determined whether the game uses *all* enemy (see Section II, "Enemy (.NMY) File") level 1 tiles with the given index when generating the records in the ??.DAT file. It can been seen however that the number of enemy/home plate records in the ??.DAT file is much greater than that in the ??0.DAT file.

HOME PLATES

A home plate may be either an airbase or a tanker track. The game selects one home plate as the starting point and *intended* finishing point for the mission; the player can choose to land at an alternate airbase at the end of the mission.

At the start of the mission, the aircraft is always heading north, so all runways are oriented north-south and the KC-10 tankers are always heading north.

The co-ordinates in a home plate record appear to specify the middle of the runway or, in the case of a tanker track, the exact location of the KC-10 tanker - this is the location of the home plate marker in the *TSD*. However, the actual location of the aircraft at the start of the mission is approximately two minutes of latitude south of this position, which corresponds to the southern end of the runway or, in the case of a tanker track, a position behind the KC-10 tanker.

Airbases are further divided into two types, those with one runway where type 0x04 is used and those with two where type 0x0C is used. In the latter case, the home plate co-ordinates should specify the eastern runway; the game will have two F-15s performing touch-and-go landings on the western runway. It is assumed that the distance between the two runways is somehow set on a per-theater basis as in the Korean theater, where type 0x0C is not used in the KO0.DAT file shipped with the game, the F-15s appear to use the runway that the home plate is on instead of a (non-existent) runway to the west. The two F-15s do not appear immediately after the start of the mission. When they first appear, they are quite close together and their first landing is aligned well with the runway, but on subsequent landings the distance between the planes increases and their touchdowns are unrealistic and close to the sides of the runway.

The home plate records are used by the game to determine where it is appropriate for the player to land the plane at the end of the mission; confusingly for the player, they cannot simply land on any runway in friendly territory as it is not the presence of a runway texture on the ground that allows them to land. When type 0x0C is used, the player is able to land on either of the two runways.

Home plate names as shown on the whiteboard in the briefing room appear to be stored in RAP.EXE. The home plate type does not affect the home plate's name - after changing a home plate from type 0x14 to 0x04 it is still shown as “Tanker Track” on the whiteboard - but the use of type 0x14 causes the home plate's name (whether it be an airbase name or “Tanker Track”) to be followed by two hyphens and a random callsign (which also appears to be taken from RAP.EXE).

TABLE 2.8. HOME PLATE TYPE

Value	Description
0x00	Indicates that a record is not a home plate record
0x04	Airbase with one runway
0x0C	Airbase with two runways
0x14	Tanker track

UNKNOWN 1

It is not known what the purpose of this section is. It appears to consist of a 16-bit count m , which is set to 11 in all ??0.DAT files supplied with the game, followed by (m + 2) records of size 38 bytes, for a total of 2 + 38 * (11 + 2) = 496 bytes.

NOTE: It has not been verified that the value 11 actually relates to the amount of data following, i.e. that changing this value results in a change in the file size expected by the game.

OBJECT NAMES

This part of the file specifies the names of the various objects that can appear on the ground in the theater. These object names appear during a mission in messages of the form

[Friendly] *object name* destroyed by *weapon name*

and during the debrief in messages of the form “Hit *object name*” .

This data is formatted as a 2-byte unsigned integer n followed by n bytes of variable-length, NUL-terminated (ASCIIZ) strings.

Some objects in the game consist of a number of sub-objects which can be independently destroyed by the player. All objects and sub-objects have names specified in this list of names. However, as shown in the section called “Case Study”, at some times the game shows the name of the object whose sub-objects have been destroyed and at other times shows the names of individual sub-objects that have been destroyed.

Names are assigned sequentially to objects and sub-objects in the following order:

- models in ??[M].3DX ordered by model index from least to greatest; then
- as above for ??MOB[M].3DX; then
- sub-models as identified by *used* conditional magic values (FIXME: xref) 0x92 to 0x99 in models in ??[M].3DX, grouped by model index from least to greatest and ordered within each grouping by conditional magic value from least to greatest; then
- as above for ??MOB[M].3DX.

Note that all of the models in the ??[M].3DX and ??MOB[M].3DX files must be read in order to determine the number of sub-models in each model. For an example of the mapping from names to models and sub-models, see Table 2.9, “Assignment of Object Names in DS0.DAT to Models and Sub-Models” .

TABLE 2.9. ASSIGNMENT OF OBJECT NAMES IN DS0.DAT TO MODELS AND SUB-MODELS

Object name index (0-based)	Assigned to
0 to 29	DS[M].3DX model indices 0 to 29
30 to 61	DSMOB[M].3DX model indices 0 to 31
62 to 65	DS[M].3DX model index 4 conditional magic values 0x92 to 0x95
66 to 69	DS[M].3DX model index 5 conditional magic values 0x92 to 0x95
70 to 73	DS[M].3DX model index 6 conditional magic values 0x92 to 0x95
74 to 76	DS[M].3DX model index 19 conditional magic values 0x92 to 0x94
77 to 80	DS[M].3DX model index 23 conditional magic values 0x92 to 0x95
81 to 88	DS[M].3DX model index 27 conditional magic values 0x92 to 0x99
89 to 91	DSMOB[M].3DX model index 4 conditional magic values 0x92 to 0x94
92 to 95	DSMOB[M].3DX model index 7 conditional magic values 0x92 to 0x95
96 to 99	DSMOB[M].3DX model index 13 conditional magic values 0x92 to 0x95
100 to 102	DSMOB[M].3DX model index 14 conditional magic values 0x92 to 0x94
103 to 104	DSMOB[M].3DX model index 16 conditional magic values 0x92 to 0x93
105 to 106	DSMOB[M].3DX model index 23 conditional magic values 0x92 to 0x93

OBJECT AND SUB-OBJECT DATA

This part of the file contains various data for each object and sub-object. It consists of an array of records with indices corresponding to the sequence of model names described above. The format of each record is described in Table 2.10, “Object and Sub-Object Data Record Format”.

TABLE 2.10. OBJECT AND SUB-OBJECT DATA RECORD FORMAT

Offset (bytes)	Length (bytes)	Description
0x0	1	Unknown flags[a]
0x1	1	Unknown, may relate to the type of weapon required to destroy the object or sub-object
0x2	2	Signed integer specifying the number of points the player receives for destroying the target, set to 0 for objects that have sub-objects

Offset (bytes)	Length (bytes)	Description
0x4	4	Unknown, probably contains a number of separate bytes or words
0x8	2	Signed X world co-ordinate relative to the model's origin of the centre of the object or sub-object; generally set to 0 for objects except those whose models are not centred on the origin, non-zero for most sub-objects
0xA	2	Signed Z world co-ordinate relative to the model's origin corresponding to the above

[a]This byte appears to contain flag bits. The following observations have been made in the DS0.DAT file:

- the bit 0x01 is set for records corresponding to objects or sub-objects that are AAA guns, tanks, ZSU-23s, ZSU-57s or objects that contain one or more sub-objects of those types, e.g. it is set for "City" which contains four "AAA Gun" sub-objects, and setting this will cause any object to fire AAA at the player;
- the bit 0x02 is set for records corresponding to naval vessels; and
- the bit 0x80 is set:
    1. for a record corresponding to an object only if the object is one with sub-objects, in which case the byte at offset 0x1 gives the sub-object index of the first sub-object in the object, where 0 is the index of the first sub-object in the theater; and
    2. for a record corresponding to a sub-object only if the sub-object is the last sub-object in its parent object.

NOTE: Co-ordinates are relative *world co-ordinates*, not model co-ordinates, and hence are not scaled by the model downscaling factor.

MODEL DOWNSCALING FACTORS

This part of the file consists of an array of unsigned bytes, one for each object (sub-objects are not included as they do not have their own models). Each value is used to scale down the size of its corresponding ??[M].3DX or ??MOB[M].3DX model. All vertex co-ordinate values in a model in one of these files are divided by the model's downscaling factor to convert them into world co-ordinates. No downscaling factor in the original files supplied with the game is greater than 8. The value 0 is used in some cases where the corresponding model contains no vertices and hence no division by zero would actually be performed.

MODEL LEVEL OF DETAIL DISTANCES

This part of the file consists of an array of 16-bit unsigned integers, one for each object (sub-objects are not included as they do not have their own models). The game determines whether a higher-detail ??[MOB].3DX model or a lower-detail ??[MOB]M.3DX model should be drawn by comparing the distance between the camera and the object to the threshold distance for the model as specified in this array.[3]

NOTE: This distance is likely to be in world co-ordinates, but this has not yet been confirmed.

[3]If the player configures Options → Object Detail → Low, models from ??[MOB]M.3DX will always be used.

LEVEL 1 TILE LIGHTS FLAGS

This part of the file consists of an array of bytes, one for each level 1 tile, specifying whether or not the tile has world-level lights that are not associated with objects. These lights are specified in the ??LTSF.3DX file, which contains one model for each level 1 tile. A 1 value indicates that lights are provided for the corresponding tile, i.e. the corresponding model in the ??LTSF.3DX file has one or more vertices. A 0 value indicates that lights are not provided, i.e. the model has zero vertices.

TSD PALETTE

This part of the file provides a 256-entry palette mapping from the VGA palette indices specified in the level 3 tiles in the .WLD file (see the section called “Level 3 tile set”) to the VGA palette indices to be used in the *TSD*. The map shown in the TSD is generated from the .WLD tile data using LOD (FIXME: this document does not yet describe how LOD works for the .WLD file or even mention it in the .WLD file section) as for the ground texture and the resulting pixel values are then mapped using this palette onto VGA palette entries appropriate for use in the TSD.

In the files supplied with the game, it appears that entries corresponding to VGA palette indices that are not used by the level 3 tiles use an identity mapping, i.e. entry i has value i. Other entries map to a small number of VGA palette entries which the game does not modify the brightness of based on the time of day, unlike the VGA palette entries referenced by the level 3 tiles.

MODEL PALETTE

This part of the file provides a 256-entry palette specifying the colours/textures to be used when displaying models from any of the .3DX files used in the theater. Tracers from AAA guns are also affected by this palette, but tracers from the player's F-15 and enemy aircraft are not. Single-pixel dots used to represent aircraft and missiles at long distances from the viewer are also not affected by this palette. The colour values used in the sub-elements in the .3DX files are used to index into the array of palette entries. Each palette entry consists of 2 bytes as described in Table 2.11, “Model Palette Entry Interpretation” .

TABLE 2.11. MODEL PALETTE ENTRY INTERPRETATION

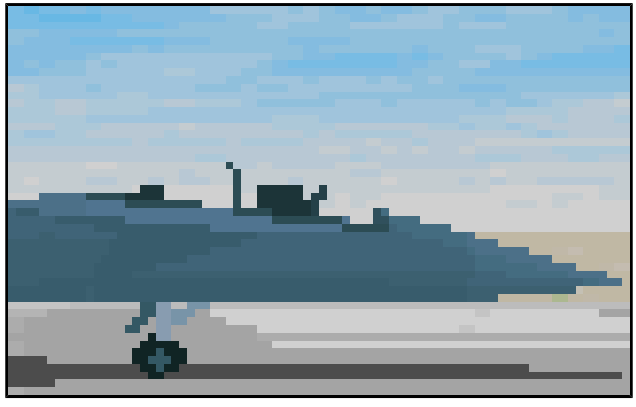
Second byte value	Colour or texture drawn [a]	Use in supplied ??0.DAT files [a]
0xFA	Partially transparent; in most cases, the value of a seems to have no effect, but for some values such as 81 and 153 it results in the cockpit glass appearing to be very dark gray with all shapes in the cockpit appearing dark green	For entries 128 and 145 only, with a always set to 0
0xFB	Completely transparent; a appears to be ignored although all possible values for it have not been tested	For entry 10 only, with a always set to 0xFB as well
0xFC	Dithering of transparency and a	In the DS theater, colour index 28 which is used for the GCI radar antennae is a dithering of transparency and dark green (VGA palette index 175)

Second byte value	Colour or texture drawn [a]	Use in supplied ??0.DAT files [a]
0xFD	Transparent, with a used to adjust palette indices of other surfaces viewed through the surface as shown by the following test results where the value of a was set to one of the following values and the cockpit of the player's F-15 was observed: • 0: the cockpit glass is completely transparent • 2: the shapes in the cockpit are drawn using VGA palette index 62 instead of 60 and the sky texture visible through the cockpit glass appears to be drawn with VGA palette indices 2 greater than before as well • 4: the change in the sky is more obvious than with 2, but the shapes in the cockpit still use palette index 62 suggesting that the game is aware of the gradients in the palette and avoids changing the hue when using a to adjust palette indices • 64: the hue of the shapes in the cockpit is changed to brown, suggesting that the game only supports a particular range of values for a when attempting to avoid hue changes.	Not used
0xFE	Solid a	Not used
0xFF	Solid a	Approximately 90% of entries are of this type
other	Dithering of a and b	For entries 137 to 143 inclusive as referenced by ??MTNS.3DX in all theaters, although in the KO and PA theaters, most entries use equal values for a and b, causing a solid colour to appear instead of dithering

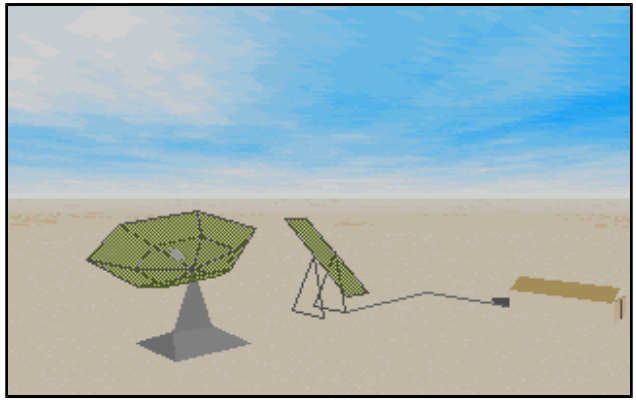
[a] a and b refer to the first and second bytes in the palette entry respectively.

F-15 COCKPIT SHOWING USE OF UNMODIFIED PALETTE ENTRY WITH 0XFA FOR GLASS

F-15 COCKPIT SHOWING USE OF MODIFIED PALETTE ENTRY WITH 0XFB FOR GLASS



GCI RADAR IN DS THEATER SHOWING USE OF PALETTE ENTRY WITH 0XFC



SPECIAL PALETTE ENTRIES

Palette index 146 is handled specially by the game. This special handling occurs when this value is used as the colour for a sub-element in a .3DX file, but not when it appears in a palette entry in the .DAT file or is used in the level 3 tile set in a .WLD file. The contents of the corresponding palette entry are ignored by the game and the behaviour is instead as follows:

1. When used in F15AIR3.3DX, PLANE1.3DX, PLANE2.3DX (i.e. all model files used for aircraft) and .3DX files for weapons and other stores (e.g. FUEL_RW.3DX), the game first draws the corresponding sub-element with VGA palette index 147 (bright red), then with 146 (dark red), then does not draw the element, draws it using palette index 146 again and then repeats the cycle, causing the sub-element to appear as a flashing red light. This .3DX file defines point sub-elements on the leading edges of both wings and on the tip of the starboard vertical stabiliser, causing flashing red lights to appear in these places on the player's F-15 when the player has turned on the running lights using **Shift**+**L**.
2. When used in ??[MOB][M].3DX (i.e. all model files used for ground objects), the game behaves as for 1 except that the steps where VGA palette index 146 is used are removed, i.e. the sub-element alternates between bright red and not drawn, and the flashing is of a lower frequency.
3. When used in ??MTNS.3DX, the game always draws the sub-element using VGA palette index 146, i.e. the element does not flash. Note however that the palette entry with index 146 is still ignored.

ENEMY AIRCRAFT TYPES

This part of the file consists of a 16-bit unsigned integer giving the number of different enemy aircraft types that will appear in the theater followed by a list of 16-bit unsigned integers giving the aircraft type indices. (FIXME: cross-reference aircraft type data in `F15.EXE`)

LATITUDE/LONGITUDE CONVERSION DATA

This part of the file specifies how the world co-ordinates used within the game are converted to latitudes and longitudes for presentation to the player in the briefing and the up-front controller.

TABLE 2.12. LATITUDE/LONGITUDE CONVERSION DATA FORMAT

Offset (bytes)	Length (bytes)	Description
`0x0`	`0x2`	`lat0`, the latitude in minutes corresponding to the northernmost part of the map (i.e. Z world co-ordinate 0 relative to the level 0 tile)
`0x2`	`0x2`	`long0`, the longitude in minutes corresponding to the westernmost part of the map (i.e. X world co-ordinate 0 relative to the level 0 tile)
`0x4`	`0x2`	`latInWldCoord`, the increase in Z world co-ordinate per minute increase in latitude
`0x6`	`0x2`	`longInWldCoord`, the increase in X world co-ordinate per minute increase in longitude
`0x8`	`0x1`	ASCII character prefix applied to latitudes, i.e. “N” or “S”[a]
`0x9`	`0x1`	ASCII character prefix applied to longitudes, i.e. “E” or “W”

[a] “S” is not used in any of the theaters shipped with the game as they are all in the northern hemisphere.

Given world co-ordinates `x` and `z`, the latitude `lat` and longitude `long` in minutes can be calculated using the values specified in Table 2.12, “Latitude/Longitude Conversion Data Format” in this way:

lat = lat0 + z / latInWldCoord

long = long0 + x / longInWldCoord

Note that for theaters in the northern hemisphere, `latInWldCoord` will be negative since an increase in latitude corresponds to a more northerly location and hence a decrease in Z world co-ordinate. Similarly, theaters in the western hemisphere will have a negative `longInWldCoord`.

IV. RANGE (.RNG) FILE

The range file specifies, for each *home plate*, the area in the world from which targets can be selected when that home plate is used for a mission. Each area is specified in the form of a bitmap with a resolution of one level 1 tile. These bitmaps always have an X dimension of 256 level 1 tiles but have a variable Z dimension.

TABLE 2.13. RANGE (.RNG) FILE FORMAT

Length (bytes)	Description
2	Number of home plates n
2 * n	n records of the format described in Table 2.14, “Bitmap Z Co-ordinate Record Format”
32 - 2 * n	Padding (all 0)
variable	Bitmap data

NOTE: Although it appears that the size of the padding between the records and the bitmap data is calculated to ensure that there are 32 bytes between the home plate count and the first bitmap, it has not been verified that this is the case for values of n other than 8 or 9, i.e. other than those values that are used in the original game files.

TABLE 2.14. BITMAP Z CO-ORDINATE RECORD FORMAT

Offset (bytes)	Length (bytes)	Description
0x0	1	Z tile co-ordinate relative to the level 0 tile corresponding to the *first* row in the bitmap
0x1	1	Z tile co-ordinate relative to the level 0 tile corresponding to the *last* row in the bitmap

Each bitmap can be viewed as a 256 * Z row-major array of bits. Each row consists of 32 bytes, for a total of 256 bits, where the most-significant bit of the first byte has index 0 and the least-significant bit of the last byte has index 255. A bit in the array is set to 1 if the corresponding level 1 tile is within range of the home plate or 0 if it is not. As expected, all tile co-ordinate rows in the level 0 tile that lie outside of the range of Z tile co-ordinates specified in the record are not considered to be within range of the home plate.

The level 1 tiles selected by the bitmaps in this file should correspond to those marked as enemy tiles in the .NMY file (see Section II, “Enemy (.NMY) File”), otherwise the game will select mission objectives which will be treated as friendly when destroyed by the player.

V. AVAILABLE MISSIONS (.MIS) AND SELECTED MISSIONS (MISSIONS.F15) FILES

The available missions file lists the available missions for the theater. Each mission entry references a level 1 tile index to which the mission can apply, so if there is for example a “SAM SITE” mission, when generating the mission the game can select from one of many SAM sites in the theater.

The selected missions file lists the three missions selected for the user's current sortie plus additional information about the sortie.

TABLE 2.15. AVAILABLE MISSIONS (.MIS) FILE FORMAT

Offset (bytes)	Length (bytes)	Description
0x0	0x2	Number of missions n (little-endian, greater than 2)
0x2	0x9E * n	n mission records (FIXME:xref)
0x2 + 0x9E * n	0x9E	Empty mission (FIXME: what values does it have?)
0x2 + 0x9E * (n + 1)		End-of-file

TABLE 2.16. SELECTED MISSIONS (MISSIONS.F15) FILE FORMAT

Offset (bytes)	Length (bytes)	Description
0x0	0x2	Un-terminated ASCII string giving theater identifier, e.g. (FIXME: format) "KO"
0x2	0x4	Unknown
0x6	0x2	As for offset 0x0
0x8	FIXME:?	Unknown
0x64	0x2	X *home plate co-ordinate* of the *home plate* (selected from the (FIXME: reference .dat file))
0x66	0x2	Z home plate co-ordinate of the home plate
0x68	FIXME:?	Unknown
0x88	0xC	FIXME: identical to bytes at 0x64 onwards?
FIXME:?	FIXME:?	Unknown
0xBE	0x1DA	3 mission records (FIXME: xref)

TABLE 2.17. MISSION RECORD FORMAT

Offset (bytes)	Length (bytes)	Description
0x0	0x1	Level 1 tile index containing objective

Offset (bytes)	Length (bytes)	Description
0x1	0x1	Mission Weapon Type
0x2	0x2	FIXME: unknown
0x4	0x50	Targeted objects (10 records of size 8 bytes and structure as defined in Table 2.18, “Mission Target Record Format”; last 2 records are never used but appear to be usable)
0x54	0x2	FIXME: number of targeted objects that must be destroyed (assume this is a little endian value and not a byte followed by something else - it makes sense as 16-bit, value never more than 4, although values up to 10 work)
0x56	0x2	Displayed X *world co-ordinate* relative to the level 1 tile containing the objective. This co-ordinate is used in all cases where the objective's location is made visible to the player, including specifying the point about which the reconnaissance video in the briefing rotates, the location of the designator in radar displays and the *TSD* and the distance to the sequence point shown in the up-front controller. In order to complete the objective, the player must destroy the previously-noted number of targeted objects from the list rather than destroy the specific object at these co-ordinates.
0x58	0x2	FIXME: Z co-ordinate (as above)
0x5A	0x1	Always zero in `.MIS` file, X *tile co-ordinate* within the level 0 tile of the level 1 tile containing the objective in `MISSIONS.F15`
0x5B	0x1	Always zero in `.MIS` file, Z *tile co-ordinate* within the level 0 tile of the level 1 tile containing the objective in `MISSIONS.F15`
0x5C	0x2	FIXME: unknown, set to zero in `.MIS` file, non-zero in `MISSIONS.F15`
0x5E	0x40	ASCIIZ objective description shown in handwriting on the board in the briefing room, in a normal font when you click on the board, and in the up-front controller in the cockpit; typically all in uppercase (FIXME: can use lowercase?);

Offset (bytes)	Length (bytes)	Description
		hyphens may be used, typically padded with spaces after the NUL character
0x9E		End-of-record

TABLE 2.18. MISSION TARGET RECORD FORMAT

Offset (bytes)	Length (bytes)	Description
0x0	0x1	Index into the object list (see Object placement information) of the selected level 1 tile of an object that must be destroyed, where the first object list entry is given an index equal to 1
0x1	0x3	Always 0
0x4	0x1	FIXME: unknown, but non-zero if record is valid
0x5	0x1	Always 0
0x6	0x1	FIXME: unknown
0x7	0x1	Always 0

If the level 1 tile index at offset 0x0 occurs many times in the level 0 tile, it will take the game some time to generate the user's mission - this will be observed as a delay entering the mission briefing room.

TABLE 2.19. MISSION WEAPON TYPE

Value	Air-to-Ground weapon selected by default	Used for[a]
0	Harpoon, no AA missiles	Not used
1	Mk82 (500lb)	Headquarters, port facilities
2	Mk84 (2000lb)	Hardened hangars, bunkers, C&C bunker, SHC tunnel, bio weapon plant, chemical plant, nuke plant, commo tower, bridge, power plant, ammo dump
3	CBU-87 CEM	SAM site, SAM radar, oil storage tanks, ZSU-23 site, ZSU-57 site, propane tanks, SCUD site, SA-5 site, GCI radar, truck depot, AAA guns, DPRK guard towers, infantry group, ammo dump
4	Rockeye	Tank group
5	Harpoon	Bio weapons ship, oil tanker, submarine, chem weapons ship, supply ship, DPRK supply ship, DPRK spy ship, missile boats, Najin gun boat
6	Durandal	Runway

[a]The information about the mission objectives that the weapon is used for is based on investigation of the missions available in the Korean theater.

VI. CAMPAIGN (.CMP AND .EXP) FILES

These files are only created by the game when a player starts a campaign. The base file name consists of the theater designator, e.g. DS, followed by the player's roster index in the range 0 to 7, so if the player with the right-most locker in the roster screen is playing a campaign in the Desert Storm theater, the file names DS7.CMP and DS7.EXP would be used.

CAMPAIGN DATA (.CMP) FILE

This file is first created by the game when the player first enters the briefing room after selecting campaign mode.

CAMPAIGN DESTROYED TARGETS (.EXP) FILE

This file records the objects that the player has destroyed during the campaign, whether or not they were mission objectives. At the end of each mission, the game updates this file to reflect the new targets destroyed during the mission. The number of records stored at offset 0x0 is incremented by the number of targets destroyed and, as documented in the section called “Campaign destroyed targets are overwritten”, the existing records at the start of the file are overwritten with the new data and new records with all fields set to 0 are added to the end of the file.

TABLE 2.20. CAMPAIGN DESTROYED TARGETS (.EXP) FILE FORMAT

Offset (bytes)	Length (bytes)	Description
0x0	0x2	Number of records n
0x2	0x6 * n	n records of format described in Table 2.21, “Campaign Destroyed Target Record Format”
0x2 + 0x6 * n		End-of-file

TABLE 2.21. CAMPAIGN DESTROYED TARGET RECORD FORMAT

Offset (bytes)	Length (bytes)	Description
0x0	0x1	X tile co-ordinate within the level 0 tile
0x1	0x1	Z tile co-ordinate within the level 0 tile
0x2	0x2	0-based index into the object list for the level 1 tile referenced by the above co-ordinates
0x4	0x2	Bit list indicating which sub-models of the object were destroyed or 0xFFFF if the object does not have any sub-models. In the case where this is a bit list, it is expected that the most significant 8 bits are all set to 0 as no model supplied with the game has more than 8 sub-models. Each bit

Offset (bytes)	Length (bytes)	Description
		number n (with 0 being the *LSB*) corresponds to the model's conditional magic (FIXME: xref to model information) with value 0x92 + n where the conditional should be set/true if and only if the bit is set to 1. If this field is set to 0xFFFF, the conditional magic with value 0x81 (if it exists) is used instead.

CASE STUDY

In version 4108.03 of the game, during a campaign in the Desert Storm theater, the primary objective for a mission is “AAA EMPLACMENTS (sic) N3449 E4517” . The primary objective information in MISSIONS.F15 reveals that:

- the level 1 tile index containing the objective is 0x37;
- the tile co-ordinates within the level 0 tile of the level 1 tile containing the objective are (0x95, 0x28);
- only one mission target record has a 1-based index into the object list which is non-zero and this value is 6 (the last four, unknown bytes of the record are FF000200); and
- the number of targeted objects that must be destroyed is 2.

It can be confirmed that the byte at offset 0x2895 in DS.WLD is equal to 0x37 . In the same file, the object placement information (see the section called “Object placement information”) for level 1 tile index 0x37 contains the value 27 at index 5. In DS.DAT , the object description for object 27 is “City” . Model index 27 in DS.3DX has 8 buildings, 4 of which appear to have AAA guns on top of them. When set, conditional magic values (FIXME: xref) 0x92 to 0x99 each cause one of these buildings to appear destroyed, with the even-numbered values corresponding to the buildings with AAA guns on them. The object descriptions in DS.DAT for objects 81-88 appear to apply to these sub-models; these descriptions alternate between “AAA Gun” and “Building” .

When the primary objective is attacked using CBU-87 CEM, the following messages appear:

```
(WSO) City destroyed by CBU87
(WSO) PRIMARY MISSION ACHIEVED!
```

Inspection of the target area using reverse padlock (**F10**) view reveals that not all of the buildings appear destroyed although due to the explosion animations it is very difficult to determine exactly how many are destroyed. In the debrief, “Hit AAA Gun” appears 4 times with no other “Hit” messages shown.

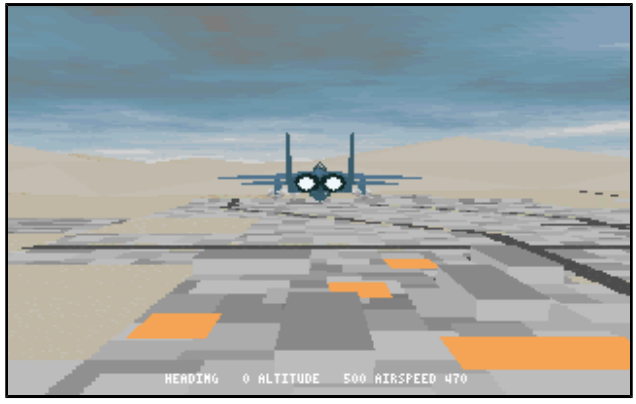
Inspection of the DS7.EXP file reveals that the number of records is increased by 1 compared to the state of the file prior to the mission, a new record consisting entirely of 0 s is appended to the file, and the first record in the file contains: [4]

- tile co-ordinates within the level 0 tile (0x95, 0x28) which match the co-ordinates from MISSIONS.F15;
- 0-based index into the object list for the level 1 tile equal to 5 which is consistent with the object list entry referenced in MISSIONS.F15;[5] and
- bit list equal to 0x55, equal to 01010101 in binary where bits 0, 2, 4 and 6 are set, which is consistent with the even-numbered conditional magic values for the model corresponding to the buildings which appear to have AAA guns on them and is also consistent with the object descriptions in DS.DAT.

As a result of these values being recorded in the DSP7.EXP file, during the next mission these objects still appear destroyed (without the explosion animations) as shown below:

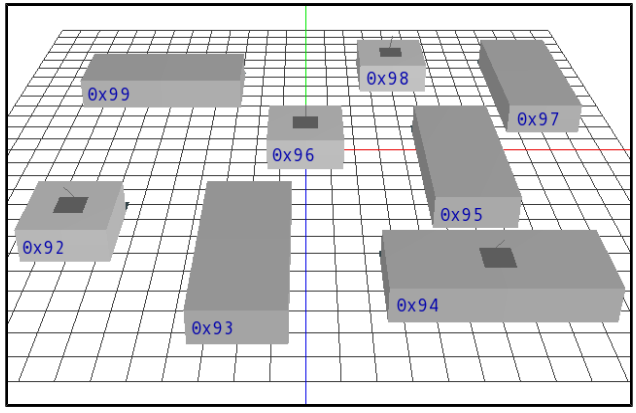
[4] the section called “Campaign destroyed targets are overwritten” covers the storage of new data in the first records of the file and the appending of records with all values set to 0.
[5] This value is offset by 1 from the value in MISSIONS.F15 due to the differing array index base.

DESTROYED "AAA GUNS" IN THE "CITY" DUE TO THE .EXP FILE



For comparison, the following figure shows the "City" model with none of the buildings destroyed and with the conditional magic value for each building superimposed over the building:

"CITY" ANNOTATED WITH CONDITIONAL MAGIC NUMBERS[6]



VII. RECONNAISSANCE VIDEO (BRFPIC.F15) FILE

This file contains the animation shown on the monitor in the briefing room when the player is viewing the map in that room. This animation is played in a loop and alternately shows the primary and secondary objectives in black-and-white. The file is generated when the player first selects the map in the briefing room.

This file consists of two 16-frame animations, one animation for each objective, with no file header. Each frame is of size 4060 bytes: 70 pixels wide, 58 pixels high and 8 bits per pixel. The top-left pixel of each frame is drawn at screen location (3, 121) by the game. The palette used for display of the video differs from that used when the player is in the cockpit.

VIII. AUDIO WAVEFORM (.8 AND F153.BIN) FILES

These files are 8kHz 8-bit mono raw PCM wave files with no header information. The `.8` files are used in the title animation with one file for each spoken phrase. The `F153.BIN` file contains all the wave audio used in the game with phrases generated by playing back multiple clips from the file; a separate table containing the offset and length of each word/phrase/phrase part must be present in some other part of the game.

[6] Image rendered by prototype `.3DX` file viewer.

IX. FIXME (RAP.EXE) FILE

This executable program file appears to be the main executable program that implements the non-simulation part of the game, including the main screen, theater selection, the briefing room and the arming screen.

This file is compressed using FIXME. After decompression using FIXME, the following data can be found in it:

TABLE 2.22. RAP.EXE FILE FORMAT

Offset by game version (bytes)		Length (bytes)	Description
4108.01	**4108.03**		
FIXME	FIXME	variable	FIXME: add home plate names

X. SIMULATION ENGINE (F15.EXE) FILE

This executable program file appears to be the main executable program that implements the simulation part of the game, i.e. the part where the player is flying the plane.

This file is compressed using FIXME. After decompression using FIXME, the following data can be found in it:

TABLE 2.23. F15.EXE FILE FORMAT

Offset by game version (bytes)		Length (bytes)	Description
4108.01	**4108.03**		
0x05B44A	0x0597CA	92	See the section called “Main .3DX File Names”
0x05B4A6	0x059826	unknown	unknown
0x05B72A	0x059B88	936	See the section called “Weapon Data”
0x05BAD2	0x059F30	1200	See the section called “Aircraft Data”
0x05BF82	0x05A3E0	unknown	unknown
0x05FB26	0x05DF84	782	See the section called “Stores .3DX File Names”
0x05FE34	0x05E292	unknown	unknown

MAIN .3DX FILE NAMES

This part of the file contains a list of variable-length, NUL-terminated (ASCIIZ) .3DX file names. Where xx occurs in these file names, it is replaced with the two-character code for the theater.

If the first file name is assigned index 1, the model file indices used in the section called “Weapon Data” and the section called “Aircraft Data” below (in the file supplied with the game, indices 5 through 7 are used) correspond to the names in this list and index 0 is used to indicate the absence of a model.

TABLE 2.24. MAIN .3DX FILE INDICES AND NAMES

Index	File name	Description of contents	Downscaling Factor(s)
1	xx.3DX	Ground objects	Specified per model in the .DAT file, see the section called “Model Downscaling Factors”
2	xxMOB.3DX	Ground objects	
3	xxM.3DX	Low-detail versions of the models in xx.3DX	
4	xxMOBM.3DX	Low-detail versions of the models in xxMOB.3DX	
5	F15AIR3.3DX	F-15 models with a number of levels of detail plus cockpit model used in padlock (**F8**) view	4[a]
6	PLANE1.3DX	US-built fixed-wing aircraft plus a bomb	unknown
7	PLANE2.3DX	Non-US-built fixed-wing aircraft plus a parachute, an aircraft's shadow and a missile	
8	xxLTSF.3DX	Ground lights, with one model for each level 1 tile	0.5
9	xxMTNS.3DX	Mountains (see the section called “Mountain 3D Model (??MTNS.3DX) Files”)	0.25

[a]This has been verified for the first two models in this file only. It is possible that other models may differ.

STORES .3DX FILE NAMES

As for the part of the file described in the section called “Main .3DX File Names” , this part of the file contains a list of variable-length, NUL-terminated (ASCIIZ) .3DX file names. The files referenced by this list all contain models for external stores on the player's aircraft including air-to-air missiles, air-to-ground weapons and fuel tanks.

NOTE: Many file names are repeated in this list. The significance of this repetition and the ordering of the entries has not been investigated.

WEAPON DATA

This part of the file contains data for the following types of weapons:

- SAMs;
- AAMs;
- a number of items called “GUN_1” through “GUN_5” whose use is unknown; and
- a number of air-to-ground weapons including “HARM” and “GBU-15”, which do not reference the models used for the player's weapons and hence are most likely related to a removed feature of the game where AI aircraft could launch ground attacks.

These records are assigned indices starting with `0` for the first record (which contains the name “None”) and are referenced using these indices by the data described in the section called “Aircraft Data” .

TABLE 2.25. WEAPON RECORD FORMAT

Offset (bytes)	Length (bytes)	Description
`0`	`8`	ASCIIZ name
`8`	`14`	unknown
`22`	`1`	model index in `.3DX` file, or `0` if there is no model
`23`	`1`	model file index (see the section called “Main `.3DX` File Names”), or `0` if there is no model

AIRCRAFT DATA

This part of the file contains data for:

- enemy fixed-wing aircraft as referenced by the `.DAT` files (see the section called “Enemy Aircraft Types”), including the “F-16” which appears in the Panama theater;
- the “F15E” which only appears at home plates with home plate type `0x0C` (i.e. airbases with two runways, see the section called “Home Plates”);
- the “KC-10” which only appears at home plates with home plate type `0x14` (i.e. tanker tracks);
- other friendly fixed-wing aircraft, including “F-14” and “F-117”, which do not appear in the game; and
- US and enemy helicopters which do not appear in the game and for which models are not specified or available except in the playable demo version of the game.

Some enemy fixed-wing aircraft names (e.g. “MIG-23”) appear in more than one record. This is most likely because the record includes the missile loadout of the aircraft and the same aircraft can appear in the game with different loadouts as shown in [1] in its description of the enemy forces in each theater.

TABLE 2.26. AIRCRAFT RECORD FORMAT

Offset (bytes)	Length (bytes)	Description
0	8	ASCIIZ name
8	2	unsigned integer specifying the maximum speed of the aircraft; the aircraft's normal speed in knots shown in the air-to-air radar's *DTWS* mode display is approximately 85% of the value specified here
10	2	unsigned integer specifying the maximum banking factor when turning, ranging from 1 for low-performance transportation aircraft to 5 for high-performance fighters; a value of 0 results in the aircraft generally never turning toward the player's aircraft and higher values such as 10 result in the aircraft banking far too much - until it is upside-down - and failing to turn
12	2	see Table 2.27, “Aircraft Radar Types”
14	1	model index in .3DX file, or 255 if there is no model
15	1	model file index (see the section called “Main .3DX File Names”), or 0 if there is no model
16	2	index of the weapon record (see the section called “Weapon Data”) of the radar-guided missiles carried, 0 for “None”
18	2	index of the weapon record (see the section called “Weapon Data”) of the heat-seeking missiles carried, 0 for “None”
20	2	number of radar-guided missiles carried, 0 if the corresponding weapon record index is 0
22	2	number of heat-seeking missiles carried, 0 if the corresponding weapon record index is 0

TABLE 2.27. AIRCRAFT RADAR TYPES

Value	*TEWS* radar indicator[a]	Notes
0x00	n/a	this value indicates that the aircraft has no radar
0x14	9	used in supplied file only for “AWACS” aircraft which is not expected to appear in the game
0x15	9	used in supplied file only for “IL-76” aircraft

Value	*TEWS* radar indicator[a]	Notes
0x16	3	
0x17	1	
0x18	4	
0x19	2	
0x1A	5	

[a]The number shown within a diamond in the *authentic mode* TEWS display to indicate the radar type.

XI. 3D MODEL (.3DX) FILES

A large number of .3DX files are provided with the game. Each file contains a number of 3D models that can be drawn during the mission. The files containing all aircraft, objects and lights that appear on the ground and mountains are referenced by the data described in the section called “Main .3DX File Names” and the files containing the external stores for the player's aircraft are referenced by the data described in the section called “Stores .3DX File Names”.

Each file begins with a header (see the section called “File Header”) followed by variable-length data for each model (see the section called “Model Data”) and finally a file terminator record (see the section called “File Terminator Record”).

This file format, with some variations, is also used by Microprose F-117A™ Stealth Fighter 2.0.

FILE HEADER

Each .3DX file begins with the following header:

TABLE 2.28. .3DX FILE HEADER FORMAT

Offset (bytes)	Length (bytes)	Description
0x0	2	Number of models n
0x2	2 * n	For each of the n models, absolute file offset of the first byte of the model data (see the section called “Model Data”)
0x2 + 2 * n	2	Absolute file offset of the file terminator record (see the section called “File Terminator Record”)

MODEL DATA

For each model in the file, the following data is present: FIXME xrefs: header, element normal vectors, element vertex index lists, element rendering sequence tree (if present), vertex co-ordinates, edges, element appearance data

FIXME describe elements, edges and vertices; cover fact that element indices 0-254 are available but 255 is reserved as a special value in Table 2.32, “Element Rendering Sequence Tree Node Record Format”

MODEL DATA HEADER

TABLE 2.29. MODEL DATA HEADER FORMAT

Offset (bytes)	Length (bytes)	Description
`0x0`	2	Signature set to `0x8001`
`0x2`	2	Absolute file offset of the byte following this absolute file offset
`0x4`	2	unknown
`0x6`	2	unknown, but appears to relate to the number of parts of the model that can move independently (e.g. a rotating radar dish)
`0x8`	8 * previous unknown value if it is greater than 2, otherwise `0`	unknown, but appears to contain origin for independently-moving parts
variable	2	number of vertices (see FIXME)
variable	2	see the section called “Element count and rendering sequence tree presence flag”

ELEMENT COUNT AND RENDERING SEQUENCE TREE PRESENCE FLAG

This signed 2-byte value encodes both:

- the number of elements (see FIXME) as the absolute value; and
- a boolean indication of whether or not a rendering sequence tree (see FIXME) is present for the model based on the sign of the value.

TABLE 2.30. ELEMENT COUNT AND RENDERING SEQUENCE TREE PRESENCE FLAG DECODING

Condition on value x	Derivation of element count	Derived rendering sequence tree presence flag
`x >= 0`	`x`	False
`x < 0`	`-1 * x`	True

ELEMENT NORMAL VECTORS

This part of the model data consists of one record of the following format for each element. Each record may have an optional 2-byte prefix which is a value in the range `0x8000` through `0x8003` inclusive or `0x8100`. FIXME: meaning of prefix

TABLE 2.31. ELEMENT NORMAL VECTOR RECORD FORMAT

Offset (bytes)	Length (bytes)	Description
0x0	2	X component of vector (if value read from file is in the range 0x8000 through 0x8003 inclusive or is equal to 0x8100, this value is an optional prefix as described above and is not part of the record)
0x2	2	Y component of vector
0x4	2	Z component of vector
0x6	4	Constant factor FIXME

This data is used to determine the visibility of the element, i.e. whether or not the viewer is “in front of” the element, or equivalently whether or not the element is facing toward the viewer.

ELEMENT VERTEX INDEX LISTS

This part of the model data contains one vertex index list for each element. Each vertex index list starts with a one-byte vertex count giving the number of vertex indices in the list n. This is followed by n bytes, where each byte is a 0-based vertex index.

ELEMENT RENDERING SEQUENCE TREE

As noted in the section called “Element count and rendering sequence tree presence flag” , this part of the model data may nor may not be included. If the flag indicates that it is *not* included, this part of the model data is not present and consumes no space in the file.

This data starts with a one-byte, 0-based element index giving the root element of the tree. Following this byte is one tree node record of the format described in Table 2.32, “Element Rendering Sequence Tree Node Record Format” for each element. These nodes are arranged into a binary tree where each tree node contains two references to sub-trees in the form of element indices. Note that there is exactly one tree node for each element and therefore tree nodes are referred to by their element indices.

TABLE 2.32. ELEMENT RENDERING SEQUENCE TREE NODE RECORD FORMAT

Offset (bytes)	Length (bytes)	Description
0x0	1	0-based element index of the root node of the left-hand sub-tree, or 0xFF if there is no sub-tree
0x1	1	as above, but for the right-hand sub-tree

This tree specifies the order in which elements should be drawn in order to ensure that elements that are closer to the viewer are drawn as closer to the viewer than elements that are further away. To draw an element A closer to the viewer than an element B implies that element A must be rendered *after* element B, i.e. a rendering sequence is implied.

When the model is rendered, a traversal of all tree nodes (i.e. all elements) is performed, starting at root node. The following processing is performed for each node:

- if the corresponding element is visible (as determined using the information described in the section called “Element Normal Vectors”):
    1. the right-hand sub-tree is recursively processed;
    2. the corresponding element is drawn; and
    3. the left-hand sub-tree is recursively processed;
- otherwise (the corresponding element is not visible)
    1. the left-hand sub-tree is recursively processed;
    2. the corresponding element is *not* drawn (as it is not visible); and
    3. the right-hand sub-tree is recursively processed.

Therefore, any elements that are “in front of” a given element (i.e. on the side of the given element that its normal vector extends from) must be part of the element's left-hand sub-tree and any elements that are “behind” a given element (i.e. on the opposite side of the given element from its normal vector) must be part of the element's right-hand sub-tree. Note that in some cases, such as when considering the faces of a convex solid, it is not possible for one element to obscure another. In such cases, it would not matter what sequence the elements were rendered in, so following of the rule described here would not be required but it would not be detrimental either.

VERTEX CO-ORDINATES

This part of the model data consists of one record of the following format for each vertex. As for the data described in the section called “Element Normal Vectors”, each record may have an optional 2-byte prefix which is a value in the range `0x8000` through `0x8003` inclusive or `0x8100`. FIXME: meaning of prefix

TABLE 2.33. VERTEX CO-ORDINATES RECORD FORMAT

Offset (bytes)	Length (bytes)	Description
`0x0`	2	X co-ordinate of vertex (if value read from file is in the range `0x8000` through `0x8003` inclusive or is equal to `0x8100`, this value is an optional prefix as described above and is not part of the record)
`0x2`	2	Y co-ordinate of vertex
`0x4`	2	Z co-ordinate of vertex

EDGES

This part of the file starts with a 2-byte edge count n followed by n edge records. Each edge record consists of 2 bytes, each a `0`-based vertex index for one end of the edge. It appears that the edges are undirected and therefore the order in which the two vertices in each edge are specified does not matter.

ELEMENT APPEARANCE DATA

This data specifies the visual appearance of the elements by defining, for each element, one or more sub-elements consisting of points, lines or polygons. Each sub-element can have conditions attached to it, allowing the game to selectively render certain sub-elements to simulate a normal vs. destroyed appearance, turn lights on and off, etc.

This part of the file can take one of three forms:

1. the general case where there is a non-zero element count (and by implication a non-zero vertex count as it is impossible to define an element without vertices);
2. the degenerate case where the numbers of vertices and elements are both zero, resulting in this data having a total length of zero; and
3. a special case where the element count is zero but the vertex count is non-zero, in which case there is data in this section for one implicit element not otherwise referenced in the file, but no file offsets table (FIXME XREF) is present[7].

The data consists of an optional list of absolute file offsets of the element appearance data followed by the actual data.

ELEMENT APPEARANCE DATA OFFSETS

This part of the file consists of one 2-byte absolute file offset for each element. This file offset references the start of the element appearance record for the given element.

This part of the file is not present in the special case where the element count is zero but the vertex count is non-zero.

ELEMENT APPEARANCE RECORDS

This part of the file consists of one element appearance record for each element or, in the special case where the element count is zero but the vertex count is non-zero, exactly one element appearance record.

Each element appearance record consists of a variable number of sub-element appearance records, each of which is of variable length, ending with a sub-element appearance record marked as a terminator. The first byte of a sub-element appearance record indicates the length of the record and the general appearance of the sub-element.

TABLE 2.34. SUB-ELEMENT APPEARANCE FIRST BYTE VALUE

Value of first byte x	Sub-element appearance record section	
`0x00` < x <= `0x1F`	FIXME: polygon inside edges	
`0x20` < x <= `0x3F`	FIXME: points at vertices	
`0x40` < x <= `0x5F`	FIXME: lines along edges	
`0x56`	FIXME: unknown - could be 0x16 lines	
`0x80` <= x <= `0x9A`	FIXME: conditions	
`0xFF`	FIXME: terminator	

[7] In the files supplied with the game, this special case is only seen in the `??LTSF.3DX` files.

POLYGON SUB-ELEMENT

This sub-element appearance record type defines a one-sided polygon bounded by a sequence of edges. The element's normal extends from the side that is drawn and the edges are specified using *clockwise winding* . The record consists of the following bytes:

1. a byte containing the number of edges that bound the polygon n such that n is no less than 3[8] and no greater than 31[9];
2. n bytes, each specifying an edge index, with indices specified in clockwise order; and
3. a byte containing the model palette index (see the section called “Model Palette”) with which the polygon is drawn.

POINTS SUB-ELEMENT

This sub-element appearance record type defines a set of points drawn at specified vertex indices. Due to the low resolution used, in the game each point has a size of one pixel regardless of viewing distance.

This sub-element type is most frequently used for lights, where palette indices for red, blinking red or white are normally used. Note that whereas in real life a point light is normally viewable from almost all angles except when it is obscured by other objects, this set of points is rendered by the game based on whether the element's normal vector indicates that it is facing the viewer and, if one is present, based on the element rendering sequence tree. To have a point visible from all angles, it would be necessary to use two separate elements with normals facing in opposite directions.

The record consists of the following bytes:

1. a byte equal to `0x20` plus the number of points n such that n is no less than 1[10] and no greater than 31[11];
2. a byte containing the model palette index (see the section called “Model Palette”) with which the points are drawn; and
3. n bytes, each specifying the index of a vertex at which a point is to be drawn.

LINES SUB-ELEMENT

This sub-element appearance record type defines a set of lines drawn along specified edge indices. (FIXME: always have count of 1 in F15SE3, but higher in F117A?) Due to the low resolution used, in the game each line has a width of one pixel regardless of viewing distance.

The uses of this sub-element type include a submarine's communication (FIXME: whip?) antenna, beams making up communications towers and edges of partially-transparent/stippled polygons such as radar dishes and the player's cockpit. Just as in (FIXME: reference points sub-element), note that whereas in real life an antenna or beam is normally viewable from almost all angles except when it is obscured by other objects, this set of lines is rendered by the game based on whether the element's normal vector indicates that it is facing the viewer and, if one is present, based on the element rendering sequence tree. To have a line visible from all angles, it would be necessary to use two separate elements with normals facing in opposite directions.

The record consists of the following bytes:

1. a byte equal to `0x40` plus the number of lines n such that n is no less than 1[12] and no greater than 31[13];
2. a byte containing the model palette index (see the section called “Model Palette”) with which the lines are drawn; and
3. n bytes, each specifying the index of an edge along which a line is to be drawn.

---

[8] It has not been verified that this is the minimum value, but it is most likely the minimum, as even though it would be possible to omit one edge from the polygon and infer that it connects the two vertices that each appear in only one of the polygon's edges, this is not done in the files provided with the game.
[9] This upper limit has not been verified.
[10] It has not been verified that this is the minimum value, but a value of `0` does not make sense as it specifies that nothing will be rendered.
[11] This upper limit has not been verified.
[12] It has not been verified that this is the minimum value, but a value of `0` does not make sense as it specifies that nothing will be rendered.
[13] This upper limit has not been verified. FIXME need to investigate 0x56

CONDITION PREFIX SUB-ELEMENT

This sub-element does not itself specify that a shape or shapes be rendered, but instead indicates that the following sub-element should be rendered conditionally based on the result of a certain test. This sub-element is never followed by another condition prefix sub-element or a terminator sub-element, i.e. it always immediately precedes a sub-element for polygons, points or lines.

The record consists of the following bytes:

1. a byte equal to `0x80` plus the condition number n such that n is no less than `0` and no greater than `26`[14]; and
2. a byte containing the value used in the conditional comparison.

The condition numbers and values have special meanings defined by the game engine. These meanings appear to be specific to certain `.3DX` files or sets of them as some values are used in two different files for different purposes.

For many of the condition numbers, the sub-elements affected by (following) the conditional sub-elements for different values are mutually exclusive. For example, in `??[MOB][M].3DX` the sub-elements following conditional number `1` with value `0` represent an undestroyed part of a model whereas for value `1` the sub-elements represent the same part of the model when it is destroyed. It would not make sense to draw both sets of sub-elements as the same part of a model cannot be both destroyed and not at the same time, so at any one point in time only the sub-elements for one of the two values will be drawn. Similarly, the sub-elements following conditional number `10` with value `1` are only drawn when it is dark, whereas when it is not dark these elements are not drawn (it is not known if there are any sub-elements prefixed with conditional number `10` with value `0`). This case can be considered to be similar to the previous, where at any point in time only the sub-elements for one of the two values are drawn, except in this case there are no elements to draw if the value is `0`.

Based on this, it seems reasonable to assume that as the game engine renders each model, it associates a single value with each condition number and only renders those sub-elements with matching condition numbers and values. In the case of condition number `26`, this would mean that even if the model contained more than one gun, a muzzle flash could only be drawn for one at a time, but as the game engine never appears to render *any* muzzle flashes this assumption cannot be proved or disproved based on this condition number.

FIXME document known values

FILE TERMINATOR RECORD

This record consists of two bytes with value `0` and consumes the last two bytes of the file. Aside from the fact that it is always referenced by the last 2-byte file offset in the file header (see the section called “File Header”), it can be distinguished from model data (see the section called “Model Data”) as the first two bytes of model data are a non-zero signature.

MOUNTAIN 3D MODEL (??MTNS.3DX) FILES

Each theather has a file containing 3D models of mountains. Each file contains 32 models. When the models are assigned indices 0 through 31, the binary representation of each model's index provides information about the model as shown in Table 2.35, “Model Indices in Mountain 3D Model (??`MTNS.3DX`) Files”. The models are divided into two sets of 16 which vary in their appearance, with the MSB selecting between the two sets. Some of the models are designed to be placed adjacent to other mountain models to form a continuous mountain range, with the other 4 bits indicating in which cardinal direction(s) the model is intended to be used with adjacent mountain models. Each of these bits is set if the mountain is open in the given direction, i.e. there should be another mountain model in that direction, or unset if the mountain model is attached to the ground in the given direction.

[14] This is the largest value observed in the files supplied with the game. It is possible that a larger value could be used if (FIXME: larger number of sub-objects is specified in .dat file)

TABLE 2.35. MODEL INDICES IN MOUNTAIN 3D MODEL (??MTNS.3DX) FILES

Bit	0 (MSB)	1	2	3	4 (LSB)
Purpose	set number	N	E	S	W

CHAPTER 3. APPENDICES

I. F-15 STRIKE EAGLE III VERSIONS

TABLE 3.1. THE RELEASED VERSIONS OF OF F-15 STRIKE EAGLE III, LEAST TO MOST RECENT

Version number[a]	Release date	Description
(version number not available)	01-Oct-1992[b]	Playable demo that appears to be based on a much earlier version of the code, distributed as `F15IIPD.ZIP`
4108.01	08-Dec-1992[c]	As provided on original installation floppy diskettes
4108.02	unknown	It is not known if this version was released, but it is listed in the `README.TXT` file supplied in `F15303.ZIP`
4108.03	23-Apr-1993[d]	Version provided in the `F15303.ZIP` patch and *most likely* on the floppy diskette provided with [1] [e]

[a] As shown in FILE → About F-15 III

[b] The date stamp of the most recent file (`READ.ME`) in the `.ZIP` file

[c] The date stamp of all the files on the installation floppy diskettes

[d] The date stamp of the most recent file (`README.1ST`) in the patch

[e] From the back cover of [1] :

> **BONUS DISK!**
>
> Includes the latest F-15 Strike Eagle III update files *plus* all new historical missions based on the U.S. air strikes on Iraq in 1993!

As the `README.TXT` file from `F15303.ZIP` indicates that 4108.03 is the version in which the "IRAQ '93" missions were added, it is likely that this floppy diskette included the same version.

COMPARISON OF PLAYABLE DEMO AND VERSION 4108.01

The following table shows *some* of the differences between these two versions of the game:

TABLE 3.2. FILE DIFFERENCES BETWEEN PLAYABLE DEMO AND VERSION 4108.01

File name by version		Differences
Playable demo	**4108.01**	
THEM.3DX	PLANE2.3DX	Models for helicopters only present in playable demo
US.3DX	PLANE1.3DX	

COMPARISON OF VERSIONS 4108.01 AND 4108.03

This section contains a *partial* analysis of the differences between these two versions of the game.

DIFFERENCES IN F15.EXE

As shown in Section X, “Simulation Engine (F15.EXE) File”, file offsets differ between the two files, suggesting differences in the program code or unknown data. However, *no* differences exist in the data discussed in the section called “Main .3DX File Names”, the section called “Stores .3DX File Names”, the section called “Weapon Data” and the section called “Aircraft Data”.

II. F-15 STRIKE EAGLE III KNOWN ISSUES

GAME HANGS “WHEN FLYING NEAR TO THE CARRIER GROUP OR THE BATTLESHIP GROUP”

In version 4108.01 of the game, DSMOB.3DX model index 20 incorrectly has the (FIXME: reference .3dx file format) value at offset 0x8b1f set to 0. From inspection of the data in the file, it appears that 2 is the correct value. An incorrect value in this location would lead to the game attempting to read data from incorrect file offsets and hence getting garbage.

This model is described in DS0.DAT as “Guided Missile Frigate” and appears to be used in the carrier and battleship groups. Flying close to either of these with Options → Object Detail → High set would cause this model to be loaded. It has also been observed that, with the same setting, viewing the carrier battle group in reverse padlock (**F10**) mode also causes the game to hang or crash. As the DSMOBM.3DX file does not have a similar error, it is assumed that with Options → Object Detail → Low set no problem would occur as the erroneous model would not be loaded.

README.TXT in F15303.ZIP states under “Version 4108.03” , “FIXES” :

> When flying near to the carrier group or the battleship group, the game will no longer hang.

The .ZIP file contains a new DSMOB.3DX file. The only difference between the version of this file shipped with version 4108.01 and this new version is that the byte at offset 0x8b1f is changed from 0 to 2 .

CAMPAIGN DESTROYED TARGETS ARE OVERWRITTEN

On completion of a mission in versions 4108.01 and 4108.03 of the game, when the `.EXP` file (see the section called “Campaign Destroyed Targets (`.EXP`) File”) is updated, the number of records stored at offset `0x0` is increased as expected, but the new records are written over the top of the initial records. Records containing all `0` values are appended to the file where it would be expected that the new records should have been written.

The result of this behaviour is that targets that were destroyed earlier in the campaign will no longer be destroyed later in the game. It is possible that the game was designed to cause some or all destroyed targets to be “repaired” (i.e. revert to their pre-destruction state) over time. However, the following factors suggest that this is a bug in the game:

1. The aforementioned appended records with all values set to `0` serve no purpose, which suggests that there is a bug in the handling of this file.
2. Given that the first record in the file is always the first to be overwritten on *every* mission, assuming that at least one target is destroyed on each mission, then this first record would record the first target destroyed in mission n and cause that target to appear destroyed in mission (n + 1). However, during that mission (n + 1), the first record would be overwritten, so that by mission (n + 2) the first target destroyed in mission n no would longer appear destroyed. Assuming a rate of one mission per day, this behaviour would simulate the case where, for at least the first target destroyed in each mission, the enemy takes only one day to repair the damage. In the case where more targets are destroyed on the first mission than in any subsequent mission, however, some records will never be overwritten, simulating the case where some targets destroyed in the first mission are never repaired. These do not appear to be reasonable parameters for a simulation; it would make more sense that the intention was for the game to not simulate the repair of any destroyed targets.

It would be possible for an external tool to back up the `.EXP` files before the game is run and, assuming that only one mission is flown per campaign whilst the game is running, compare these to the modified versions after the game exits to re-construct the complete set of records in order to work around this issue.

NOTE: Implementing such a workaround would greatly increase the number of valid records in the `.EXP` file by the end of a campaign. It may be the case that the game was never tested with large numbers of *valid* records in this file and that this workaround could cause issues in the game.

III. GAME DATA FILE STATISTICS

TABLE 3.3. THEATER STATISTICS

Datum	Theater		
	DS	K0	PA
`??0.DAT` home plate count	8	9	9
`??0.DAT` object name count	107	116	102
`??[M].3DX` model count	30	29	29
`??MOB[M].3DX` model count	32	31	29

IV. GNU FREE DOCUMENTATION LICENSE

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
D. Preserve all the copyright notices of the Document.
E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
H. Include an unaltered copy of this License.
I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by

the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/ [http://www.gnu.org/copyleft/].

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: HOW TO USE THIS LICENSE FOR YOUR DOCUMENTS

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright (C) YEAR YOUR NAME.
> Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

GLOSSARY

A

Authentic mode: The game allows various systems in the aircraft to be modelled using one of two different levels of realism (and therefore difficulty). This mode is the most realistic of the two.
See Also Standard mode.

C

Clockwise winding: Specifies that the vertices or (in the case of this game) edges defining a one-sided polygon (i.e. a polygon that is only rendered when observed from a particular side) are specified in an order that, when observed from the side that is rendered, is clockwise.

D

Designated Track With Scan: A mode provided by the *authentic mode* air-to-air radar.
DTWS: See Designated Track With Scan.

H

Home plate: The friendly air base or tanker track that a mission starts and ends at.
Home plate co-ordinate: A modified form of *world co-ordinate* used for storing the co-ordinates of a *home plate*. A home plate co-ordinate is a world co-ordinate divided by 32 and is stored in 16 bits with the most significant bit always 0. (FIXME: rename as these co-ordinates have been found to be used elsewhere)

L

LSB: Least-Significant Bit

S

Standard mode: The game allows various systems in the aircraft to be modelled using one of two different levels of realism (and therefore difficulty). This mode is the least realistic of the two.
See Also Authentic mode.

T

Tactical Electronic Warfare System: In the game, this provides a cockpit display of ground and airborne radar threats and missiles. In authentic mode, ground and airborne radar threat icons include a number which provides additional information about the type of radar.
Tactical Situation Display: A cockpit display showing a moving map with icons showing the locations of the primary and secondary targets and the home plate. In *standard mode*, it also shows the locations of ground and airborne threats and all friendly airbases.
TEWS: See Tactical Electronic Warfare System.
Tile co-ordinate: A co-ordinate within a tile that, if any sub-tiles exist, treats them as having a size of one and hence references a sub-tile rather than an actual location in the world. In other words, this co-ordinate always forms an array index into the 2D array that makes up the tile.
TSD: See Tactical Situation Display.

W

World co-ordinate: A co-ordinate within a tile that accounts for the sizes of all the levels of sub-tiles, if any exist. A world co-ordinate may be relative to a particular tile at a particular level, in which case co-ordinate (0, 0) refers to the top-left corner of that tile, or may be relative to the whole world (i.e. the level 0 tile), in which case co-ordinate (0, 0) refers to the top-left corner of the level 0 tile.

REFERENCES

[1] RUSSELL, Lawrence T. *F-15 Strike Eagle III: The Official Strategy Guide*. Rocklin, CA, USA 95677-1260: Prima Publishing, 1993. ISBN 1-55958-197-2.

INDEX

A