MechAssault 2 MGF documentation

This document is a work-in-progress documentation of the MGF resource archive file format used in the original Xbox game MechAssault 2: Lone Wolf. It is complete with descriptions of the overall file structure, offsets and lengths of known data in the file, and descriptions of individual files/assets that can be found within the archive.

1. Structure

MGF files are split in to 5 primary sections: the header, file entries, file relationships, directory strings (i.e. file and folder names), and all the body (the file data for every file in the archive). There is one MGF file for each “level” in the game. A level can either be a multiplayer map, a campaign map, or even a cutscene map. MGF files store all the files used for that particular level in the game, so different MGF files will store a lot of the same data such as textures and models. MGF archives do **not** store any audio data; this can be found in the DVDxwb files elsewhere in the game’s root directory.

* 1. Header

The header for MGF archives is 64 bytes long, consisting mostly of 32-bit unsigned integers and some padding.

|  |  |  |
| --- | --- | --- |
| Offset (bytes) | Length (bytes) | Description |
| 0 | 4 | File signature. Always reads “mgf “ (including the whitespace at the end). |
| 4 | 1 | Version number? Always 4 for MechAssault 2 archives and 2 for MechAssault 1 archives. |
| 5 | 1 | Unknown. Value is always 1. |
| 6 | 2 | Unknown. Always reads “ZZ”. |
| 8 | 4 | Unknown. |
| 12 | 4 | Number of files stored in the archive. |
| 16 | 4 | Length of the file entry block in bytes. Because the structs representing file entries are always 32 bytes, this value can also be calculated by multiplying the number of files by 32. |
| 20 | 4 | Offset where the file entry block begins. This is always 64, as the file entry block begins right after the header which is 64 bytes long. |
| 24 | 4 | Number of file relationships stored in the archive. |
| 28 | 4 | Length of the file relationship block in bytes. Relationship structs are 24 bytes long, which means this value can also be calculated with number of file relationships \* 24. |
| 32 | 4 | Offset where the file relationship section begins. |
| 36 | 4 | Length of the directory strings section in bytes. |
| 40 | 4 | Offset where the directory strings section begins. |
| 44 | 4 | Unknown. |
| 48 | 4 | Unknown. Value is always 65536. (This appears many times throughout the archive) |
| 52 | 12 | Padding. |

* 1. File entries

This section of the MGF file stores a list of 32 byte-long structs storing information about each file within the archive. There should be <number of files> of these structs in this section as described by offset 16 in the MGF header. This section starts right after the MGF header.

A breakdown of the file entry structs:

|  |  |  |
| --- | --- | --- |
| Offset (bytes) | Length (bytes) | Description |
| 0 | 4 | Unique identifier. This value appears to be a simple auto-incrementor at first, though there are gaps. The gaps represent folder “entries” in the archive, but folder information is not stored in this section, and is only useful in the following section. |
| 4 | 4 | Unknown. |
| 8 | 4 | Unknown. |
| 12 | 4 | Length of the file data in bytes. |
| 16 | 4 | Length of the file data in bytes (duplicate?) |
| 20 | 4 | Unix timestamp that stores when the last time this file was modified. |
| 24 | 4 | Offset where the file’s data begins. |
| 28 | 4 | Unknown. Only exists in MechAssault 2 archives, and always reads 0x00F71200. |

* 1. File relationships

This section of the MGF file stores a list of 24 byte-long structs storing information about where each file and folder belongs in the folder hierarchy of the archive’s contents. There are <number of relationships> of these structs as described by offset 24 in the MGF header. Each struct represents an “item” in the archive’s directory, and can be either a folder or file, which is unlike the previous section which only stores information on files.

A breakdown of the file relationship struct:

|  |  |  |
| --- | --- | --- |
| Offset (bytes) | Length (bytes) | Description |
| 0 | 4 | Unknown. Variable for folders, always 0 for files. |
| 4 | 4 | Index of the parent folder for this item. The index points to the location of the parent item’s struct as it would appear if all structs were stored in an array. |
| 8 | 4 | If the item is a folder, this value stores a folder ID. Otherwise, if the item is a file, this value is -1 (0xFFFFFFFF). |
| 12 | 4 | Unknown. |
| 16 | 4 | Offset of the string for this item’s file/folder name, relative to the offset the string section. |
| 20 | 4 | If the item is a file, this value stores the file’s ID. Otherwise, if the item is a folder, this value is -1 (0xFFFFFFFF). |

* 1. Directory strings

This section is a simple sequence of null-terminated strings that represent the file and folder names stored in the archive. The offset of each string is referenced from the previous file relationships section.

1. Files
   1. Textures

Texture files are encoded in binary and may store pixel data in many different formats and types. Texture file names are appended with the .tif file extension. Despite the .tif extension, texture are **not** common TIFF files; they are an entirely custom file format.

The header is 149 bytes long and consists of the following:

|  |  |  |
| --- | --- | --- |
| Offset (bytes) | Length (bytes) | Description |
| 32 | 4 | File signature; always reads “PIC “ (including whitespace). |
| 40 | 4 | Total size of file (including header and body) in bytes. |
| 48 | 4 | Version subchunk identifier; always reads “ver “ (including whitespace). |
| 56 | 1 | Version subchunk value. Similar to the archive header, this value is always 4 for MechAssault 2 and 2 for MechAssault 1. |
| 57 | 4 | Flags subchunk identifier; always reads “flgs”. |
| 65 | 4 | File flags. Very important, more details on this soon. |
| 69 | 4 | Texture width subchunk identifier; always reads “wdth”. |
| 77 | 4 | Width of the texture in pixels. |
| 81 | 4 | Texture height subchunk identifier; always reads “hgt “ (including whitespace). |
| 89 | 4 | Height of the texture in pixels. |
| 93 | 4 | Mipmap levels subchunk identifier; always reads “mips”. |
| 101 | 4 | Number of mipmap levels for this texture. |
| 105 | 4 | Texture size subchunk identifier; always reads “size”. |
| 113 | 4 | Size of the texture in bytes. This includes bytes for mipmap levels, extra frames, and is affected by texture compression. |
| 117 | 4 | Texture frames subchunk identifier; always reads “frms”. |
| 125 | 4 | Number of frames for this texture. For normal static textures, this value is always 1. For animated textures, this value is greater than 1. |
| 129 | 4 | Texture depth subchunk identifier; always reads “dpth”. |
| 137 | 4 | Number of layers for this texture. This value is used for multidimensional textures. This value is always 1 for 2D textures. |
| 141 | 4 | Always reads “bits”. |
| 145 | 4 | Number of bytes between the offset of “bits” and the end of the file. In other words, this value is always the value for the texture’s size + 8. |

* + 1. Texture flags

The flags subchunk (identified with “flgs” in the file’s header) provides information on how the texture data is formatted and encoded.

* 1. Models

MechAssault 2’s 3D model files are simple XML documents that describe the materials and meshes that the model is comprised of, and a hierarchical tree structure of XML nodes that map the model’s scene graph.

The XML file itself is appended with the .mgmodel extension. Alongside the XML file are binary files containing vertex and index buffer data for each mesh in the base .mgmodel file. These files are appended with .mgmodel{<meshname> verts} and .mgmodel{<meshname> indices}.

* + 1. XML File structure

All contents of the MGMODEL file are contained with in a root node labelled “xmlres”. Unlike traditional XML files, the initial XML declaration is absent. Below is an overview of an MGMODEL file’s XML structure:

* Xmlres
  + Material (name, attributes)
  + Material (name, attributes)
    - Texture (texture file path, texture properties)
  + Mesh (name, type, material, vertex buffer, index buffer)
  + Mesh (name, type, material, vertex buffer, index buffer)
  + Node (transforms)
    - Node (transforms)
    - Node (transforms)
      * Node 3d object (transforms)
        + Mesh (name)
      * Node
    - Node (transforms)

Materials

There are several different types of material nodes found within the various MGMODEL files, each with varying attributes corresponding to whichever shader they are using. The following discovered materials are:

* Material\_base
* Material\_complex
* Material\_specular\_mask
* Material\_color\_shift
* Material\_solid
* Material\_env
* Material\_detail
* Material\_animated
* Material\_specular\_alpha
* Material\_distort

Most material nodes also have child nodes which point to textures the material will sample from:

* Basetexture – used for colour (diffuse) maps
* Multitexture – used by the material\_complex node
* Distorttexture – used exclusively by the material\_distort node
  1. Strings