Structure of XPR file
An xpr file is a dataset, containing a directory of sorts, and multiple “files”
The types of files that can be contained in an xpr file are:
Models (which contain Objects, which contain materials)
Vertexbuffers (which have 3 dimensional models0
Textures (dds graphics files)

The xpr file starts with a header

```
typedef struct _xprheader {
	dword magic;
	dword filesize;
	dword headersize;
} xprheader;
```

The magic in my experience is not consistent might be encrypted.
Filesize does not always match the actual file size.
Headersize is the size of the header data for the xpr file.

```
0000 D3 9E B6 49 00 EC 11 00  00 28 00 00 00 00 00 80  ╚Pts╣I.∞◄..(.....Ç
0010 C0 23 00 00 4D 44 4C 00  03 00 00 00 06 00 00 00  └#..MDL.♥...♠...
0020 00 00 00 00 03 00 00 00  38 00 00 00 D2 0C 00 00  ....♥...8...╥...
0030 D6 11 00 00 00 00 00 00  4F 42 4A 00 01 00 00 00  ╓◄......OBJ.☺...
```

After the xpr header, there is are the 3 types of “files”
First there is a magic number
0x80000000 for a model
0x00040001 for a texture
0x00800001 for a vertex buffer

So assuming we have a model

```
0000 D3 9E B6 49 00 EC 11 00  00 28 00 00 00 00 00 80  ╚Pts╣I.∞◄..(.....Ç
0010 C0 23 00 00 4D 44 4C 00  03 00 00 00 06 00 00 00  └#..MDL.♥...♠...
0020 00 00 00 00 03 00 00 00  38 00 00 00 D2 0C 00 00  ....♥...8...╥...
0030 D6 11 00 00 00 00 00 00  4F 42 4A 00 01 00 00 00  ╓◄......OBJ.☺...
0040 04 00 00 00 D9 04 00 00  00 00 00 00 80 40 AE 3A  ♦...┘♦......Ç@«:
```

This tells us that what follows is a model header

```
typedef struct _mdlheader {
	dword size;   // size of chunk
	dword magic;  // model magic
	dword numobj; // # of objects in the model
	dword numtxt; // # of textures in the model
	dword pad0;   // 00 fill
	dword num1;   // ?? $ of vertex buffers ?
} mdlheader;
```

Following the model header is an array of dwords the count is in the model header “numobj” in this example 3.
So the 3 object offsets are
38 00 00 00
D2 0C 00 00
D6 11 00 00

These offsets are from the beginning of the file
So looking at offset 38 we see the first object block

```
typedef struct _vertexinfo { // total size 32 bytes
    dword numverts;
    dword vertex_offset;
    dword numfaces;
    dword index_offset;
    dword num0;
    dword num1;
    dword num2;
    dword num3;
} vertexinfo;

typedef struct _objheader { // total size 160 bytes
        dword magic;
        dword vertex_type;
        dword num0;
        dword numfaces;
        float x,y,z,w; // 16 bytes
        vertexinfo vi[4]; //32 * 4 = 128 bytes
} objheader;
```

```
0030 D6 11 00 00 00 00 00 00  4F 42 4A 00 01 00 00 00  ╓◄......OBJ.☺...
0040 04 00 00 00 D9 04 00 00  00 00 00 00 80 40 AE 3A  ♦...┘♦......Ç@«:
0050 00 62 74 BC FD 8C 5F 3E  15 02 00 00 00 28 00 00  .bt╝²ì_>§☻...(..
0060 D9 04 00 00 20 03 00 00  00 00 00 00 00 00 00 00  ┘♦.. ♥.........
0070 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
0080 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
0090 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00A0 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
        Not all of vi is shown
```

Following the object header, are the material headers.
A material header has a magic number of 0x80000000 for a standard material and 0x80000001 for a 256 bit transparent material.

```
00D0 00 00 00 00 00 00 00 00  00 00 00 80 90 00 00 00  ............ÇÉ...
00E0 00 00 00 00 00 00 00 00  00 00 00 00 A8 00 49 BC  ..............¿.I╝
00F0 96 79 24 3D E8 47 1F 3E  00 00 80 3F 00 00 80 3F  ûy$=ΦG▼>..Ç?..Ç?
0100 00 00 80 3F 00 00 80 3F  00 00 80 3F 00 00 80 3F  ..Ç?..Ç?..Ç?..Ç?
0110 00 00 80 3F 00 00 80 3F  BD C4 F8 3E 2A 58 B3 3E  ..Ç?..Ç?╝─°>*X│>
0120 1B 85 E4 3E 00 00 80 3F  00 00 00 00 00 00 00 00  .àΣ>..Ç?........
0130 00 00 00 00 00 00 00 00  00 00 48 42 01 00 00 00  ..........HB☺...
0140 03 00 00 00 7B 59 71 FF  01 00 00 00 00 00 00 00  ♥...{Yq ☺.......
0150 00 00 00 00 00 04 00 00  20 00 00 00 00 00 00 00  .....♦.. ........
0160 5C 01 00 00 02 00 00 00  00 00 00 80 90 00 00 00  \☺..☻.....ÇÉ...
0170 00 00 00 00 00 00 00 00  00 00 00 00 94 BE 27 BC  ............ö┘'╝
0180 9B 6E 4E BD 37 D6 27 3E  00 00 80 3F 00 00 80 3F  ¢nN╝7╓'>..Ç?..Ç?
0190 00 00 80 3F 00 00 80 3F  00 00 80 3F 00 00 80 3F  ..Ç?..Ç?..Ç?..Ç?
01A0 00 00 80 3F 00 00 80 3F  BD C4 F8 3E 2A 58 B3 3E  ..Ç?..Ç?╝─°>*X│>
01B0 1B 85 E4 3E 00 00 80 3F  00 00 00 00 00 00 00 00  .àΣ>..Ç?........
01C0 00 00 00 00 00 00 00 00  00 00 48 42 01 00 00 00  ..........HB☺...
01D0 03 00 00 00 7B 59 71 FF  02 00 00 00 00 00 00 00  ♥...{Yq ☻.......
01E0 00 00 00 00 00 04 00 00  20 00 00 00 5E 01 00 00  .....♦.. ...^☺..
01F0 DC 01 00 00 02 00 00 00  01 00 00 80 90 00 00 00  ▄☺..☻...☺..ÇÉ...
0200 00 00 00 00 00 00 00 00  00 00 00 00 F0 43 F6 BB  ............≡C÷╗
0210 00 62 74 BC 43 B9 5A 3E  00 00 80 3F 00 00 80 3F  .bt╝C╣Z>..Ç?..Ç?
```

```
typedef struct _materialblock { // 144 bytes
```

```
        dword header;
        dword size;
        dword pad02;
        dword number;
        float mat_color0[4]; // diffuse color ?
        float mat_color1[4];
        float mat_color2[4];
        float mat_color3[4];
        float mat_color5[4];
        float power; // specular power ??
        dword pad25; // 1
        dword pad26; // 1
        dword ffff; // 0xffffffff
        dword tex_no; // texture used by this material
        dword pad29;
        dword transparency; // this is the dxt1 transparency flag
        dword pad31;
        dword pad32;
        dword indexoffset; // index buffer - offset
        dword indexsize; // index buffer - num
        dword pad35;
} materialblock;
```

There is quite a bit of “stuff” between the objects, I don’t know what it is so I’m not documenting it.

The objects in this file would then also occur at offset D20C0000 and D6110000

Now by moving to the offset “size” listed in the original model here at location 0010 we will then point to the next “file”