PS2 Half-life DOL models are closely matching PC MDL models, except that textures are stored in 8-bit PSI format and header of DOL model contains additional field for LODs. Comparison of texture formats is shown on fig. 1.

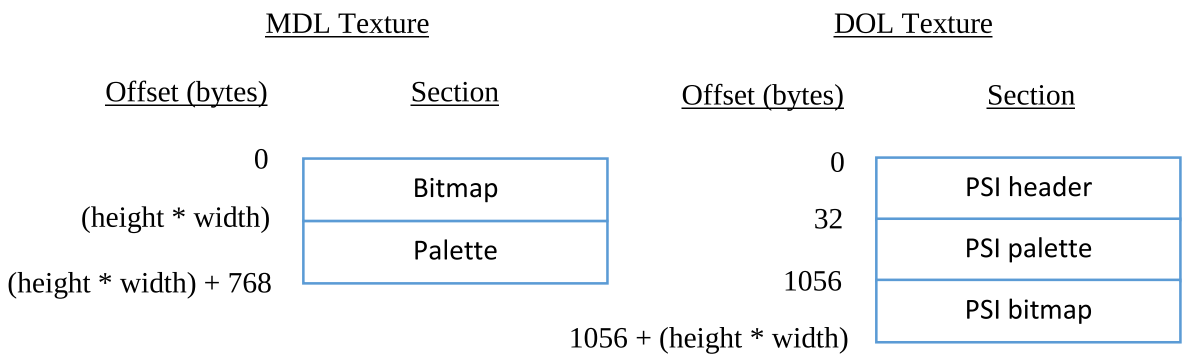

Fig. 1 – Comparison of MDL and DOL texture formats

As you can see, DOL textures are bigger than MDL textures by 288 bytes. So it is quite easy to convert DOL models to MDL format, just rewrite texture data and job is done. But because DOL textures are bigger than MDL textures, to perform conversion from MDL to DOL format you need to update:

1) Model file size.
2) Texture data offset.
3) Texture offsets in texture table.

It is important to know that textures inside DOL models must be aligned within 16-byte blocks. In addition, dimensions of DOL textures are limited to those values: 16, 32, 64, 128, 256, etc. If input model has textures with different dimensions then they should be rescaled, otherwise either model would look corrupted or game would crash.

Additional pain of conversion from MDL to DOL format is that DOL header is bigger in size by 16 bytes. Bad header can cause improper submesh switching (for example, all scientists have Walter (Kleiner) heads or Barney shooting without pistol). If you want to resize header you must update every offset field inside model. However, I managed to find hack that allows proper submesh switching without resizing of header (see chapter 1.1).

To perform more accurate conversion you can also update:

1) Internal model name (change extension)
2) Internal submodel references (change extensions)

It is also important to know that auto-aim function targets last model attachment. So if auto-aim targets something wrong then consider decompiling model and adding attachment to a bone that you wish to target.

1 MDL\DOL file structure

Simplified structure of model file is shown on fig. 2.

Section
Header
Model data
Texture table
Skin table
Textures

Fig. 2 – Simplified structure of model file

1.1 Header

Simplified structure of model header:

```
struct sModelHeader
{
       char Signature[4];                    // "IDST"
       ulong Version;                        // 0xA - GoldSrc model
       char Name[64];                        // Internal model name
       ulong FileSize;                       // Model file size
       char SomeData1[96];                   // Data that is not important for conversion
       ulong SubmodelCount;                  // How many submodels
       ulong SubmodelTableOffset;            // Location of submodel table
       ulong TextureCount;                   // How many textures
       ulong TextureTableOffset;             // Texture table location
       ulong TextureDataOffset;              // Textures location
       ulong SkinCount;                      // How many skins
       ulong SkinEntrySize;                  // Size of entry in skin table (measured in shorts)
       ulong SkinTableOffset;                // Location of skin table
       ulong SubmeshCount;                   // How many submeshes
       ulong SubmeshTableOffset;             // Location of submesh table
       char SomeData2[28];                   // Data that is not important for conversion
       ulong TransitionsOffset;              // Location of transition table

       // Extra section of PS2 DOL models (missing in PC MDL models)
       ulong TransitionsOffset2;             // Same as TransitionsOffset
       uchar SubmeshGroupCount;              // How many submesh groups are present in model
       uchar LODsPerSubmesh;                 // ?How many LODs are related to one submesh?
              // (setting both SubmeshGroupCount and LODsPerSubmesh to 0 disables LODs)
       uchar Magic[10];                      // Filled with zeroes
}
```

The easiest hack to enable proper submesh switching in converted models on PS2 without resizing of header is to set `SubmeshGroupCount` and `LODsPerSubmesh` to 0, despite the fact that it is not header data.

Update: I found out that applying this hack to all IDST models is requirement to run game without crashes on original hardware and to stop messages like "(EE pc:002C2EDC) TLB Miss, addr=0x62c59de6 [load]" in PCSX2.

1.2 Texture table

Texture table must have entry for each texture. Texture table entry structure:

```
struct sTextureTableEntry
{
       char Name[68];              // Texture name
       ulong Width;                // Texture width
       ulong Height;               // Texture height
       ulong Offset;               // Location of texture
}
```

1.3 Skin table

Skin table must have entry for each skin. Skin table entry structure:

```
struct sSkinTableEntry
{
       short Skin[SkinCount];      // Each short contains index of texture,
                                   // associated with this skin
}
```

1.4 Textures

Actual texture data is located in this section. Structure of textures is shown on fig. 1.