PS2 Half-life SPZ sprites are severely different from PC SPR sprites. You can see it by looking on comparison shown on fig. 1.

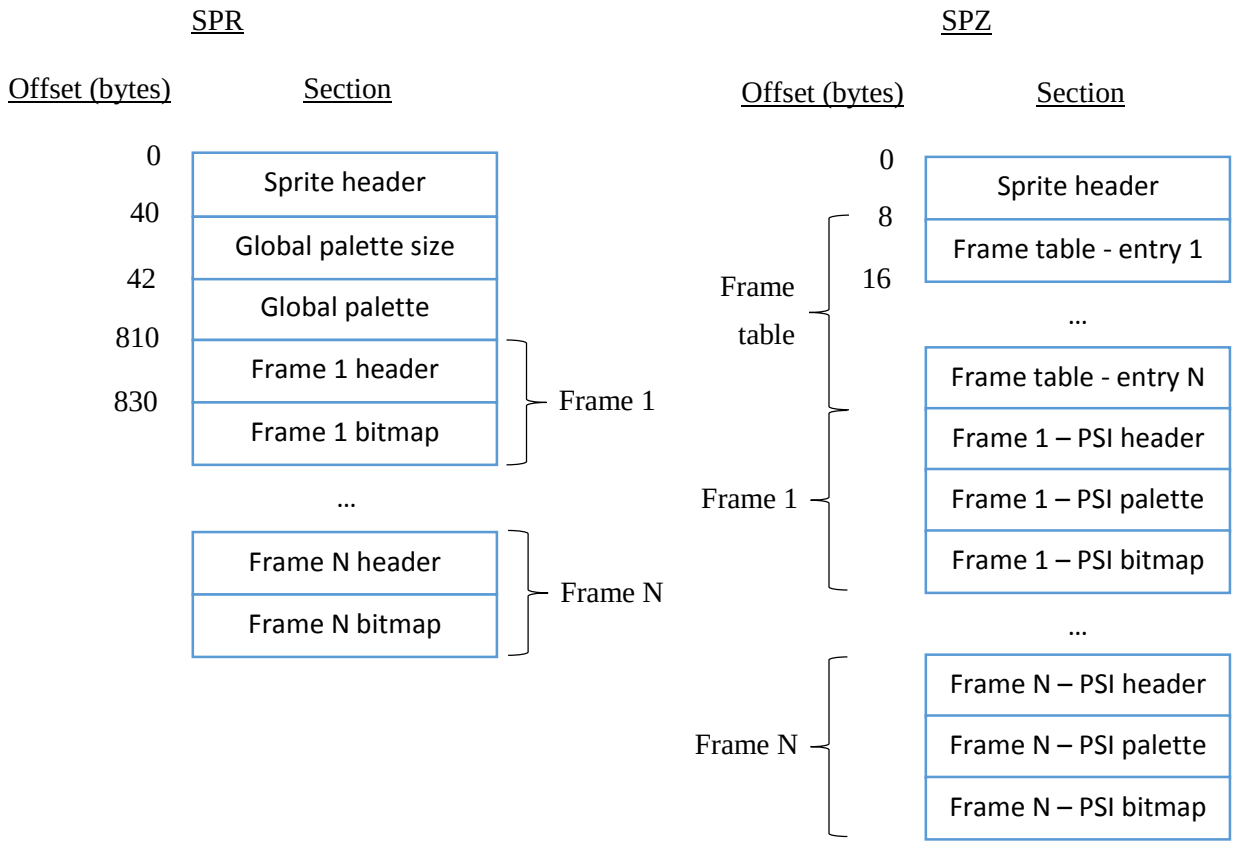

Fig. 1 – Comparison of SPR and SPZ formats

Some parameters of SPR file are not present in SPZ, or represented in different form.

Such parameters as bounding radius, frame origin x, frame origin y are not present in SPZ sprites and must be recalculated:

```
BoundingRadius = sqrtf(((float) NewWidth/2) * ((float) NewWidth/2) +
       + ((float) NewHeight/2) * ((float) NewHeight/2));
FrameOriginX = (-1) * NewWidth / 2;
FrameOriginY =  NewHeight / 2;
```

Sprite transparency is represented differently in SPZ sprites (see chapter 1.3 of this document).

Parameters such as beam length, sync type and frame group are not present in SPZ file at all. So I found out it is safe to set them like that:

BeamLength = 0;

SyncType = 1;

FrameGroup = 0;

1 SPZ file structure

1.1 Header

Structure of header:

```
struct sSPZHeader
{
	char Signature[4];	// "SPAZ" signature
	uchar Type;		// 0x00 - VP_PARALLEL, 0x01 - VP_PARALLEL_UPRIGHT,
				// 0x02 - ORIENTED, 0x03 - VP_PARALLEL_ORIENTED
	uchar Magic;		// = 0
	uchar RAMFlag;		// It is set to 0 everywhere, except GRESTORE.PAK (I think it is
				// flag that indicates that frame offsets are pointing to RAM)
	uchar FrameCount;	// How many frames in sprite
}
```

I managed to find only one sprite with orientation set to 1 – “blood_01.spz”. It is located in Decay’s “PAK0.PAK”. This sprite is not present in PC sprite files, so it is unclear what orientation it has. However, I think that it is “VP_PARALLEL_UPRIGHT”.

1.2 Frame table

Frame table must have entry for each frame. Frame table entry structure:

```
struct sSPZFrameTableEntry
{
	ulong FrameID;		// Frame ID (unique for each frame in all sprite files).
				// It is used only in GRESTORE.PAK, and normally it equals zero
	ulong FrameOffset;	// Location of frame in file (or RAM if RAMFlag is set to 1)
}
```

1.3 Frame

Each frame is regular 8-bit PSI image. Note that red, green, and blue bytes of SPZ palette have maximum value of 127 instead of SPR’s 255. So they must be multiplied by 2 during conversion to SPR format, or divided by 2 during conversion to SPZ format.

It is important to know that all frames inside SPZ sprite must be aligned within 16-byte blocks. Dimensions of each sprite’s frame are limited to those sizes: 16, 32, 64, 128, 256, etc. If input sprite has frames with different dimensions then they should be rescaled and upscale target should be set to original frame size. If sprite has frames with wrong dimensions then it can cause partial game graphics corruption or wrong appearance of sprite.

Also, it is important to know that unlike SPR files, in which transparency type defined by variable, in SPZ files transparency is baked in palette’s alpha channel:

1) NORMAL/ADDITIVE – all alpha bytes in palette are set to 0x80;
2) ALPHATEST– all alpha bytes, except last, are set to 0x80, last one is set to 0x00;
3) INDEXALPHA – alpha bytes in palette are linearly growing from 0x00 in first element to 0x7F in last element.

To determine SPZ file transparency format you can calculate how many alpha bytes are not equal to 0x80:

1) If all bytes are equal to 0x80 then it is additive transparency format.
2) If only one byte is not equal to 0x80 than it is alpha test transparency format.
3) If more than one byte is not equal to 0x80 then it is indexed alpha transparency format.

Written by Alexey Leushin, Novosibirsk, 2017