```
=========================================
    T H E   " U N - O F F I C I A L"
      PLAYSTATION DEVELOPMENT FAQ

             LIBGPU
           CONFERENCE
=========================================


           Release v1.1
      Last Updated: August, 31, 1995
```

DISCLAIMER

TRADEMARK INFORMATION

COPYRIGHT NOTICE

[1.] Library GPU (LIBGPU)

[1.1.]:How can the texture distortion be avoided that occurs when a texture-mapped rectangle polygon is displayed with transparent perspective conversion?

GPU in PlayStation supports the linear texture mapping. This means that, only as for the 'texture', the middle points of lines before conversion will be the middle points even after transparent perspective conversion. Also, since a rectangle is composed of 2 triangles, the following problem will take place.

```
  A         B
 „¬„ª„ª„ª„ª„ª„–    If this rectangle is altered in shape to a trapezoid
 „«         _^„«   with AB // CD, the line, 1 - 4, will be straight.
But,
 „«1 2   _^   „«   if it is altered freely, the line, 1- 4, will be bent.
 „«+ +  _^* * „«
 „«    _^  3 4 „«
 „«_^         „«
 „ -_„ª„ª„ª„ª„ª„®
  C            D
(++ and ** are texture patterns.)
```

Since this is the specification, the effective use of the software only can make it possible to avoid this problem.

The following ways may be suggested.

  1) Dividing the rectangle ABCD into small pieces to reduce the bend of the line 1 - 4. However, this will increase the registered polygons and the overhead, and as a result, the number of polygons for other characters will be reduced.

 2) Letting the distortion look more natural by manipulating the texture pattern. However, the success depends on trial and error.

  3) Switching some prepared patterns of the texture every view point if the change pattern is fixed. The texture is distorted beforehand assuming the distortion after conversion.   However, this consumes texture area substantially, and it is difficult to prepare a lot of patterns beforehand in fact.

Therefore, the best way will be 1). Try to avoid this situation by linking these ways.

[1.2.]:I would like to know about the texture cache specification.

The size of the texture cache depends on the texture modes as shown below.

```
 Texture      Cache size
  ---------------------------
   4-bit       64_~64
   8-bit       32_~64
```

```
  16-bit       32_~32
  ---------------------------
```

When the texture pattern used by a primitive is within this range, the texture will be cached, and high-speed drawing will be feasible. Also, only 1 entry of a CLUT will be cached. Thus, as far as the CLUT used by the primitive isn't switched, the CLUT will be cached and high-speed drawing will be feasible. The texture cache and the CLUT cache will be flushed automatically after the access to the frame buffer.

[1.3.]:How can the shape of a sprite be altered?

It is easy to alter the shape freely with libgpu. With the POLY_FT4 structure, after the initialization by SetPolyFT4() and setTPage, execute the AddPrim() function by substituting any value into the structure member.
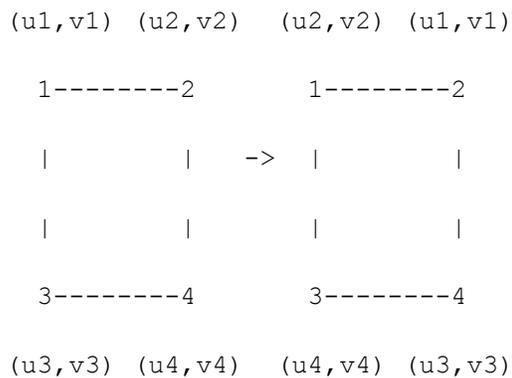
[1.4.]:How can flicker caused by the large-sized sprite rotation be avoided?

GPU performance depends on the number of drawing pixels and the depth of drawing. Therefore, even if there are only 10 sprites, but if the size is 255 x 255, the processing will be beyond the CPU performance to be incomplete. According to the report, it is guessed that quite large 16- or 8-bit images are rotated. The internal arithmetic functionality of GPU is used for the distortion/rotation, but this is also the limited specification. Thus, if the extreme distortion/rotation/scaling are carried out, the

internal arithmetic speed will not be able to keep up with it. Use GPU
in consideration of the smaller number of drawing pixels, and the
effective cache-hit.

[1.4.1.]:How can a sprite pattern be flipped?

Use POLY_FT4 instead of SPRITE to flip 'u' and 'v' corresponding to each
other. For example, if flipping (u, v) on y-axis, it is y-axis flipping.

```
  (u1,v1) (u2,v2)    (u2,v2) (u1,v1)

    1--------2         1--------2

    |        |   ->    |        |

    |        |         |        |

    3--------4         3--------4

  (u3,v3) (u4,v4)    (u4,v4) (u3,v3)
```

However, since the processing speed of POLY_FT4 is half of that of
SPRITE, use SPRITE for the patterns which don't have to be flipped.