

=====

T H E " U N - O F F I C I A L "

PLAYSTATION DEVELOPMENT FAQ

LIBGTE

CONFERENCE

=====

Release v1.1

Last Updated: August 31, 1995

DISCLAIMER

This FAQ is to aid in informing the licensed game developer about the development environment provided by Sony Computer Entertainment.

The Development System Tool to which this manual relates is supplied pursuant to and subject to the terms of the Sony Playstation Licensed Developer Agreement.

This FAQ is intended for distribution to and use only by Sony PlayStation Licensed Developers in accordance with the Sony Playstation Licensed Developer Agreement. The information in this manual is subject to change without notice.

The content of this manual is Confidential Information of Sony for the purposes of the Sony PlayStation Licensed Developer Agreement and otherwise.

TRADEMARK INFORMATION

PlayStation and Sony Computer Entertainment names and logos are trade names and/or trademarks and/or copyright artwork of Sony Corporation(or its subsidiaries).

All specific names included herein are trademarks and are so acknowledged: IBM, Microsoft, MS-DOS. Any trademarks not mentioned here are still hypothetically acknowledged.

COPYRIGHT NOTICE

[1.] Library GTE (LIBGTE)

[1.1.]: What function can set FogNear and FogFar at the same time?

SetFogNearFar() can.

We have Japanese document only.

```
<begin Japanese>
_ $BL>>N_ (J
    SetFogNearFar    _ $B%U%)%0%Q%i%a!<%?$r@_Dj$9$k_ (J
_ $B7A<0_ (J
    void SetFogNearFar(a,b,h)
    long    a,b,h;
_ $B2r@b_ (J
    _ $B;kE@$H%9%/j!<%s$N5wN%$,_(Jh_ $B$N$H$-!"_ (J
    _ $B%U%)%0#0!s$H$J$k#ZCM$r_ (J a _ $B$K@_Dj$9$k!#_ (J
    _ $B%U%)%0#1#0#0!s$H$J$k#ZCM$r_ (J b _ $B$K@_Dj$9$k!#_ (J
    0<a,b<65536
    (b-a)>=100
    <_ $B0z?t%U%)!<%^%C%H_ (J>
    a : (0,32,0)
    b : (0,32,0)
    h : (0,32,0)
_ $BJV$jCM_ (J
    _ $B$J$7_ (J
<end Japanese>
```

[1.2.]: Which is a faster divider to use CPU or to use GTE?

It is faster to use CPU divider because of the overhead of register settings of GTE.

[1.3.]: How does GTE calculate 'p' (depth queue parameters)

$$p = DQB + DQA * (h/sz)$$

h: projection.

sz: z value in the screen coordinate.

DQB,DQA: set by SetFog*() functions

[1.4.]: Which is faster for 32bitx32bit or 32bitx16bit multiplier.

ApplyMatrixLV() is a 16bitx32bit multiplier using GTE. No samples for 32bitx32bit multiplier using GTE.

Generally GTE is faster for matrix or vector calculation, and CPU is faster for the other case such as sigle 32bitx16bit multiplier.

[1.5.]: Which is faster for 64bitx64bit multiplier.

CPU is faster.

[1.6.]: Is there any functions for anti-aliasing ? (by Acclaim)

We can make anti-aliasing functions. please tell me the specifications.

- 1) Anti-aliasing for the edge of each polygon ?
- 2) Anti-aliasing for texture pattern?
- 3) Or do you want a simple LPF (Low Pass Filter) function?
- 4) What is the texture resolution (4bit/8bit/16bit)?

[1.7.]: When we can use new in-line GTE functions.

In-line GTE functions are built-in type libgte functions. These functions is fast because no stack access or PC (program counter) change is required when they are called.

Our in-line GTE functions are different from conventional c++ inline functions. The conventional in-line functions needs source code of the functions. but ours does not needs them. The linker attached the in-line functions in object code level. Therefore what you need is the new linker and *.obj code in libgte.

This in-line functions are working in R&D level on some UNIX (especially Sony NEWS-OS) environment, then now we are coverting to the PC environment which everyone uses.

So it takes for a month or so. we are trying to release the first sample in 7/E.

[1.8.]: Which is faster to use in-line RotTransPers or MESH ?

It depends on the case.

[1.9.]: The syntax of the MESH functions?

*_B#1#7!%_(JMesh functions

**17.1.

NAME

RotMeshPrimS_F3
RotMeshPrimS_G3
RotMeshPrimS_FC3
RotMeshPrimS_GC3
RotMeshPrimS_FT3
RotMeshPrimS_GT3
RotMeshPrimS_FCT3
RotMeshPrimS_GCT3
RotMeshPrimS_T3



FORMAT

```

RotMeshPrimS_F3(msh,prim,ot,otlen,dpq,backc)
RotMeshPrimS_G3(msh,prim,ot,otlen,dpq,backc)
RotMeshPrimS_FC3(msh,prim,ot,otlen,dpq,backc)
RotMeshPrimS_GC3(msh,prim,ot,otlen,dpq,backc)
RotMeshPrimS_FT3(msh,prim,ot,otlen,dpq,backc)
RotMeshPrimS_GT3(msh,prim,ot,otlen,dpq,backc)
RotMeshPrimS_FCT3(msh,prim,ot,otlen,dpq,backc)
RotMeshPrimS_GCT3(msh,prim,ot,otlen,dpq,backc)
RotMeshPrimS_T3(msh,prim,ot,otlen,dpq,backc)

```

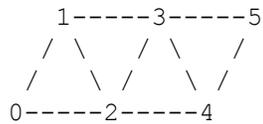
```

TMESH *msh;      /*pointer to TMESH data*/
POLY_?3 *prim;   /*pointer to GPU packet*/
u_long *ot;      /*pointer to ordering table*/
u_long otlen;    /*length of ordering table*/
long dpq;        /*depth quing ON/OFF(dpq=0:OFF,1:ON)*/
long backc;      /*backface clip ON/OFF(backc=0:ON,1:OFF)*/

```

EXPALNATION

Rotation, Transposition, Perspective & Link to OT of strip type mesh data(smash) s.t.



There is 9 drawing modes.

```

Flat                ...F
Gouraud             ...G
FlatColor           ...FC
GouraudColor        ...GC
FlatTexture         ...FT
GouraudTexture      ...GT
FlatColorTexture    ...FCT
GouraudColorTexture ...GCT
texture             ...T

```

Flat_\$(B!')_(JFlat Shading by one of vertex color_\$(B!J_(Jlighting ON_\$(B!K_(J

Gouraud_\$(B!')_(JGouraud Shading by vertex colors_\$(B!J_(Jlighting ON_\$(B!K_(J

Flat Color_\$(B!')_(JFlat rendering by one of vertex color_\$(B!J_(JLighting OFF_\$(B!K_(J

Gouraud Color_\$(B!')_(JSmooth rendering by vertex colors_\$(B!J_(JLighting OFF_\$(B!K_(J

Flat Texture_\$(B!')_(JTexture mapping & Flat Shading_\$(B!J_(Jlighting ON_\$(B!K_(J

Gouraud Texture_\$(B!')_(JTexture mapping & Gouraud Shading_\$(B!J_(Jlighting ON_\$(B!K_(J

Flat Color Texture_\$(B!')_(JTexture mapping & Flat rendering_\$(B!J_(Jlighting OFF_\$(B!K_(J

```

    Gouraud Color Texture_$(J)Texture mapping & Gouraud
    rendering(lighting OFF)
    Texture_$(J)Texture mapping_$(J)lighting OFF_$(J)K_$(J)

```

NOTE

In case of FT,GT,FCT,GCT and T texture address should be preset in GPU packet.

FCT and GCT don't have depth quing option.

RETURN VALUE

NONE

**17.2.

NAME

```

    RotMeshPrimR_F3
    RotMeshPrimR_G3
    RotMeshPrimR_FC3
    RotMeshPrimR_GC3
    RotMeshPrimR_FT3
    RotMeshPrimR_GT3
    RotMeshPrimR_FCT3
    RotMeshPrimR_GCT3
    RotMeshPrimR_T3

```

FORMAT

```

    RotMeshPrimR_F3 (msh,prim,ot,otlen,dpq,backc)
    RotMeshPrimR_G3 (msh,prim,ot,otlen,dpq,backc)
    RotMeshPrimR_FC3 (msh,prim,ot,otlen,dpq,backc)
    RotMeshPrimR_GC3 (msh,prim,ot,otlen,dpq,backc)
    RotMeshPrimR_FT3 (msh,prim,ot,otlen,dpq,backc)
    RotMeshPrimR_GT3 (msh,prim,ot,otlen,dpq,backc)
    RotMeshPrimR_FCT3 (msh,prim,ot,otlen,dpq,backc)
    RotMeshPrimR_GCT3 (msh,prim,ot,otlen,dpq,backc)
    RotMeshPrimR_T3 (msh,prim,ot,otlen,dpq,backc)

```

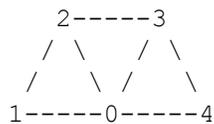
```

    TMESH *msh;      /*pointer to TMESH data*/
    POLY_?3 *prim;   /*pointer to GPU packet*/
    u_long *ot;      /*pointer to ordering table*/
    u_long otlen;    /*length of ordering table*/
    long dpq;        /*depth quing ON/OFF(dpq=0:OFF,1:ON)*/
    long backc;      /*backface clip ON/OFF(backc=0:ON,1:OFF)*/

```

EXPLANATION

Rotation,Transposition,Perspective & Link to OT of round type mesh data(rmesh) s.t.



There is 9 drawing modes.

```

Flat          ...F
Gouraud       ...G
FlatColor     ...FC
GouraudColor  ...GC
FlatTexture   ...FT
GouraudTexture ...GT
FlatColorTexture ...FCT
GouraudColorTexture ...GCT
texture       ...T

```

```

Flat_$(JFlat Shading by one of vertex color_$(Jlighting
ON_$(K_$(J
Gouraud_$(JGouraud Shading by vertex colors_$(Jlighting
ON_$(K_$(J
Flat Color_$(JFlat rendering by one of vertex
color_$(Jlighting OFF_$(K_$(J
Gouraud Color_$(JSmooth rendering by vertex
colors_$(Jlighting OFF_$(K_$(J
Flat Texture_$(JTexture mapping & Flat
Shading_$(Jlighting ON_$(K_$(J
Gouraud Texture_$(JTexture mapping & Gouraud
Shading_$(Jlighting ON_$(K_$(J
Flat Color Texture_$(JTexture mapping & Flat
rendering_$(Jlighting OFF_$(K_$(J
Gouraud Color Texture_$(JTexture mapping & Gouraud
rendering(lighting OFF)
Texture_$(JTexture mapping_$(Jlighting OFF_$(K_$(J

```

NOTE

In case of FT,GT,FCT,GCT and T texture address should be preset in GPU packet.
 FCT and GCT don't have depth quing option.

RETURN VALUE
 NONE

**17.3.

NAME

RotMeshPrimQ_T

FORMAT

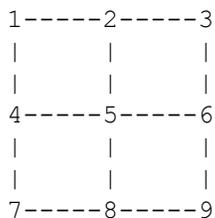
RotMeshPrimQ_T(msh,prim,ot,otlen,dpq,backc,SCLIP,line_sxy)

```

QMERH *msh;          /*pointer to QMESH data*/
POLY_FT4 *prim;      /*pointer to GPU packet*/
u_long *ot;          /*pointer to ordering table*/
u_long otlen;        /*length of ordering table*/
long dpq;            /*depth quing
ON/OFF(dpq=0:OFF,1:ON)*/
long backc;          /*backface clip
ON/OFF(backc=0:ON,1:OFF)*/
SCLIP *SCLIP;        /*screen clipping area*/
LINE_BUF *line_sxy  /*1 line buffer for internal
calculation*/
_$(B2r@b_$(J

```

Rotation, Transposition, Perspective, Link to OT and
 Screen Clipping by screen coordinates (X, Y, Z) of
 2 dimensional type mesh data (qmesh) s.t.



There is 1 drawing mode.

Texture ...T

Texture_\$(J) (JTexture mapping_\$(J) (Jlighting OFF_\$(K) (J

NOTE

Vertex number of H direction should be multiple of 3. (msh->lenh=3*n)

In case of FT, GT, FCT, GCT and T texture address should be preset in GPU packet

This function uses following structures.
 More than 1H+3vertices (msh->lenh+3) line buffer is necessary.
 Scratchpad as line buffer will speed up the calculation.

```

typedef struct {
    long    sminX;
    long    smaxX;
    long    sminY;
    long    smaxY;
    long    sminZ;
    long    smaxZ;
} SCLIP;

typedef struct {
    long    sxy;
    long    code;
} LINE_BUF;

```

RETURN VALUE
 NONE

[1.10.]: Is it possible to perform coordinate conversion and transparent perspective conversion separately in libgte?

The RotTrans() function performs coordinate conversion only. However, transparent perspective conversion can not be performed singly because of the hardware specification.

[1.11.]: How can a matrix be rotated in order of Z-, X-, and Y-axis in libgte?

Use the `RotMatrixYXZ()` function instead of `RotMatrix()`.

[1.12.]: Is the screen coordinate value obtained by the `RotTransPers()` function returned with the offset added?

Yes. The screen coordinate is returned with the offset added.