

Sample of pseudo environment mapping sample

Jan 12, 1998

Apr 21, 1998

Copyright(C) 1998 Sony Computer Entertainment Inc.  
Rights Reserved.

#### <Overview of pseudo environment mapping>

With pseudo environment mapping driver, the following two methods are allowed to use.

##### 1) "1-D mapping"

"1-D mapping" is a method of implementing the metallic material by making the texture address which depends on normal vector's Y value of screen coordinate when the transparent perspective conversion of primitive driver is performed.

"1-D mapping" sample texture uses gradation in X direction, which makes the U value of texture address which depends on normal vector's Y value of screen coordinate.

##### 2) "2-D mapping" (reflection and refraction)

"2-D mapping" is a method of implementing a material have reflection or refraction effect by making the texture address which depends on normal vector's X and Y values of screen coordinate when the transparent perspective conversion of primitive driver is performed.

In library 4.2, pseudo environment mapping driver is released as beta version. Spec is possible to be changed without any announce in the future.

#### <Pseudo environment mapping primitive drivers>

Primitive drivers currently available:

##### 1) Driver Function

Aliases	ID	Primitive types
-----	-----	-----
GsUSCAL2	0x06000100	envmap shared calculate
GsUE1G3	0x0600100c	envmap 1D gour tri
GsUE1G4	0x06001014	envmap 1D gour quad
GsUE1SG3	0x0600110c	envmap 1D shared gour tri
GsUE1SG4	0x06001114	envmap 1D shared gour quad
GsUE2LG3	0x0600200c	envmap 2D reflect gour tri
GsUE2LG4	0x06002014	envmap 2D reflect gour quad
GsUE2RG3	0x0600300c	envmap 2D refract gour tri
GsUE2RG4	0x06003014	envmap 2D refract gour quad
GsUE2RLG3	0x0600400c	envmap 2D both gour tri
GsUE2RLG4	0x06004014	envmap 2D both gour quad
GsUE2OLG3	0x0600500c	envmap 2D org+reflect gour tri
GsUE2OLG4	0x06005014	envmap 2D org+reflect gour quad
*****		

##### 2) Macro definition for LAB File(envmap.def)

```
#define CTG_ENV      0x06      /* category ID */
```

```

#define ENV_SHARED 0x0100 /* envmap shared ID */
#define ENV1D      0x1000 /* envmap 1D */
#define ENV2DL     0x2000 /* envmap 2D reflect */
#define ENV2DR     0x3000 /* envmap 2D refract */
#define ENV2DRL    0x4000 /* envmap 2D both */
#define ENV2DOL    0x5000 /* envmap 2D org+reflect */

```

<Data creation for pseudo environment mapping>

The following three changes are needed.

#### 1) Environment mapping Texture

For example, use the texture under data/scei/envmap/texture directory.  
Add image primitive by using HMD dis-assembler/assembler.

```

ENV1D?.TIM    for 1-D mapping
ENV2D?1.TIM   for 2-D mapping reflection
ENV2D?2.TIM   for 2-D mapping refraction

```

#### 2) Primitive type change(for 1-D mapping)

Change primitive type to environment mapping primitive by using HMD dis-assembler/assembler.

##### 1. When environment mapping is assigned to gouraud triangle:

```

DEV_ID(SCE)|CTG(CTG_POLY)|DRV(0)|PRIM_TYPE(IIP|TRI);
-> DEV_ID(SCE)|CTG(CTG_ENV)|DRV(0)|PRIM_TYPE(ENV1D|IIP|TRI);

```

##### 2. When shared calculation primitive is changed to one for environment mapping use:

```

DEV_ID(SCE)|CTG(CTG_SHARED)|DRV(0)|PRIM_TYPE(0);
-> DEV_ID(SCE)|CTG(CTG_ENV)|DRV(0)|PRIM_TYPE(ENV_SHARED);

```

##### 3. When shared polygon primitive is changed to one for environment mapping use:

```

DEV_ID(SCE)|CTG(CTG_SHARED)|DRV(0)|PRIM_TYPE(TRI|IIP);
-> DEV_ID(SCE)|CTG(CTG_ENV)
   |DRV(0)|PRIM_TYPE(ENV_SHARED|ENV1D|TRI|IIP);

```

#### 3) Environment mapping primitive header section

When pseudo environment mapping driver is used, environment mapping primitive header section is required. Replace primitive header by using HMD dis-assembler/assembler. The following is an example of environment mapping primitive header section.

Note: Size or parameter and the like of environment mapping primitive header section is subject to change in future.

##### 1. 1-D mapping only

\*\*\*\*\*

PolyPrimHdr\_0000:

```

14; /* hdr size */
M(Poly_0000 / 4);
M(Vert_0000 / 4);
M(Norm_0000 / 4);
M(ImagePrim_0000 / 4); /* 1D envmap tex */
0;

```

```

0;
0;
0;
/* 1D envmap */
B(2); B(0); B(4); B(0); /* 1D: texmode, (none), material, (none) */
/* 2D reflection */
B(0); B(0); B(0); B(0);
B(0); B(0); B(0); B(0);
/* 2D refraction */
B(0); B(0); B(0); B(0);
B(0); B(0); B(0); B(0);
M(CoordSect_0000 / 4);
*****

Explanation of parameters)

1D envmap tex : Link to image primitive section of texture for
                1-D mapping
                (The head of the section is specified in the example)
1D texmode    : Texture mode(2 is specified in the example)
1D material   : Material(Refer to the following)

1-D mapping materials are designed as follows in the example texture
files.

    0: blue to black
    2: brass
    4: gold
    6: red
    8: gradation blue (test)
   10: mono (test)
   12: all blown (test)
   14: gradation blown (test)

2. 1-D mapping + 2-D mapping
   (reflection and refraction: 16-bit texture for both)
*****
PolyPrimHdr_0000:
    14; /* hdr size */
    M(Poly_0000 / 4);
    M(Vert_0000 / 4);
    M(Norm_0000 / 4);
    M(ImagePrim_0000 / 4 + 0); /* for 1D envmap */
    M(ImagePrim_0000 / 4 + 3); /* reflection tex */
    0; /* reflection clut */
    M(ImagePrim_0000 / 4 + 6); /* refraction tex */
    0; /* refraction clut */
    /* 1D envmap */
    B(2); B(0); B(4); B(0); /* 1D: texmode, (none), material, (none) */
    /* 2D reflection */
    B(2); B(1); B(120); B(0); /* texmode, abr, reflect rate, (none) */
    B(0); B(0); B(0); B(0);
    /* 2D refraction */
    B(2); B(1); B(120); B(0); /* texmode, abr, refract rate, (none) */
    B(0); B(0x50); B(0x80); B(0x78); /* (none), r, g, b */
    M(CoordSect_0000 / 4);
*****

3. 1-D mapping + 2-D mapping
   (reflection and refraction: 8-bit texture for both)
*****
PolyPrimHdr_0000:

```

```

14; /* hdr size */
M(Poly_0000 / 4);
M(Vert_0000 / 4);
M(Norm_0000 / 4);
M(ImagePrim_0000 / 4);          /* for 1D envmap */
M(ClutImagePrim_0000 / 4);      /* reflection tex */
M(ClutImagePrim_0000 / 4 + 3);  /* reflection clut */
M(ClutImagePrim_0000 / 4 + 6);  /* refraction tex */
M(ClutImagePrim_0000 / 4 + 9);  /* refraction clut */
/* 1D envmap */
B(2); B(0); B(4); B(0);        /* 1D: texmode, (none), material, (none) */
/* 2D reflection */
B(1); B(1); B(120); B(0);      /* texmode, abr, reflect rate, (none) */
B(0); B(0); B(0); B(0);
/* 2D refraction */
B(1); B(1); B(120); B(0);      /* texmode, abr, refract rate, (none) */
B(0); B(0x50); B(0x80); B(0x78); /* (none), r, g, b */
M(CoordSect_0000 / 4);
*****

```

Explanation of parameters)

```

reflection tex  : Link to image primitive section of texture
                  for reflection
reflection clut : Link to image primitive section of texture CLUT
                  for reflection
refraction tex  : Link to image primitive section of texture
                  for refraction.
refraction clut : Link to image primitive section of texture CLUT
                  for reflection
/* 2D reflection */
2D texmode      : Texture mode
2D abr          : Semi-transparency rate
2D reflect rate : Reflect rate
/* 2D refraction */
2D texmode      : Texture mode
2D abr          : Semi-transparency rate
2D refract rate : Refract rate
2D r, g, b      : Specification of the base color
                  (subject to change in future)

```

4, 8, and 16-bit can be used for 2-D mapping texture. However, since CLUT is read every time for every polygon in 8-bit color, when 8-bit color is used and two polygons are written by one driver(as a case of both reflection/refraction or texture + refraction), drawing is not efficient. In the case, sharing CLUT makes drawing efficient.