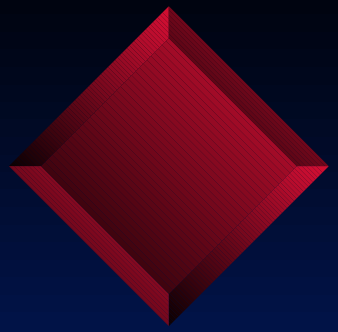

LIBGS

Fundamentals of GS
and
Inside LIBGS

Dan Burnash March 1996

PlayStation™

Fundamentals of LIBGS

- Capabilities of LibGS
  - Uses
  - Strengths
  - Weaknesses

2D Graphics

- Libgs 2D graphics
  - Background drawing using GsBG
  - GsBG describes a rectangular background surface which contains a scrollable, scaleable, and rotateable rectangle consisting of a series of textured subrectangles.

2D Graphics - cont.

- GsSortBg is slowest
- GsSortFastBg is faster
  - Background may not be rotated or scaled.
- GsSortFixBg16/32 are the fastest

- Gs supports sprite drawing
  - Using GsSPRITE

Sprites - cont.

- GsSortSprite is slowest
  - uses poly_ft4
- GsSortFlipSprite is faster
  - Uses a poly_ft4
  - supports horizontal and vertical flipping
- GsSortFastSprite is fastest
  - Does not support scaling or rotation

3D Graphics

- LibGS 3D graphics
  - Data formats used
  - Initializing the system
  - Setting the Viewpoint
  - Order Tables
  - Object Handling

Data Formats

- TMD - Defines 3D models
  - Sorted TMD - A faster form of TMD data. The data is sorted by packet type. This reduces icache misses.
  - All tmd data should be sorted using tmdsort.
- PMD - Defines preshaded 3D models
  - Contains preset double buffers for speed.

Init 3D System

- GsInitGraph(xres, yres, inter, dith, vram);
- GsDefDispBuff(x0, y0, x1, y1);
- GsInit3D();
- GsSetProjection(screen_z);

Setting the Viewpoint

- GsSetRefView2(&rview2);
  - takes viewpoint coords, reference point coords, and rotation angle.
  - Creates a rotation and translation matrix.
- GsSetView2(&view2);
  - Sets the view with a matrix defining rotation and translation.

Order Tables

- The Playstation commonly uses two linked lists(OT's) for rendering 2d primitives to the screen.
- There are two OT's so the gpu can use one as a draw list, while the other is being filled by the program.

Size of the OT

- Size of Order Table(OT) is
  - 1<< Z_RESOLUTION
  - where, 1 < Z_RESOLUTION <= 14

Object Initialization

- GsMapModelingData(tmd_ptr);
  - Maps TMD to a real address.
- GsLinkObject5(tmd, &object, obj_num);
  - links object handler to its TMD data.
- GsPresetObject(&object, addr);
  - Creates drawing primitives for object, which speeds up the processing.

Object Initialization - cont.

- GsInitCoordinate2(WORLD, &coord);
  - Init coordinate system.
  - Set to location of each object.
- LoadImage(&rect1,tim1.pixel);
  - loads tim data into vram.

Object Handling

- GsGetLw(object[i].coord2, &tmp_ls);
- GsSetLightMatrix(&tmp_ls);
- GsGetLs(object[i].coord2, &tmp_ls);
- GsSetLsMatrix(&tmp_ls);
- GsGetLws(coord, lw, ls);

The GsSort Functions

- GsSortObject3
  - processes pmd data and adds to OT
- GsSortObject4
  - processes tmd data and adds to OT
- GsSortObject5
  - processes tmd data and adds to OT
  - Uses preset packets for increased speed

3D Object Handlers

	Presort	Preset	Preshade	Workbase
GsDOBJ2	no	no	available	yes
GsDOBJ3	yes	yes	required	no. in data
GsDOBJ5	yes	yes	available	no. yes on subdivide

PlayStation™

Inside LibGS

- A closer look at the following functions:
  - GsSetProjection
  - GsInitGraph
  - GsDefDispBuff
  - GsInit3D
  - GsSetRefView2
  - GsSetView2

Inside LibGS - cont.

- A closer look at the following functions:
  - GsGetLw
  - GsSetLightMatrix
  - GsGetLs
  - GsSetLsMatrix
  - GsGetLws
  - GsSortObject5

GsSetProjection

- Sets the projection distance
  - Calls:
    - SetGeomScreen(h);

GsInitGraph

- Initializes the GS graphics system
  - Sets disp and draw structure members
  - Calls:
    - ResetGraph(0)
    - PutDrawEnv()
    - PutDispEnv()
    - InitGeom()
    - SetFarColor(0, 0, 0)
    - SetGeomOffset(0, 0)

GsDefDispBuff

- Defines the double buffers
- Calls:
  - GsSetDrawBuffClip()
    - Sets clip members of draw environment
      - Calls: PutDrawEnv()
  - GsSetDrawBuffOffset()
    - determines drawing offset
      - Calls: SetGeomOffset()

GsInit3D

- Inits the 3D system
  - Sets default lighting to normal
- Calls:
  - GsSetDrawBuffOffset()

GsSetRefView2

- Create a unit matrix including aspect ratio
- scale vpx, vpy, vpz, vrx vry, vrz
  - Right shift them until they fit in 15 bits

GsSetRefView2 - Rot/Trans

- Create a rotation matrix from GsRVIEW.rz
- Create an x rotation matrix using vp & vr
- Create a y rotation matrix using vp & vr
  - MulMatrix(unit_matrix, rz_rot_matrix)
  - MulMatrix(unit_matrix, rx_rot_matrix)
  - MulMatrix(unit_matrix, ry_rot_matrix)
- Apply the 32 bit translation to the matrix
  - ApplyMatrixLV()
- Store result as WSMatrix

GsSetRefView2 - cont.

- If the coord.super = WORLD
  - we are done
- If coord.super points to another coord
  - Continue transforming until we reach the WORLD

GsSetView2

- WS_Matrix = View2.view
- If view2.super = WORLD
  - we are done
- If view2.super points to another coord
  - Continue transforming until we reach the WORLD

GsGetLw

*(GsCOORDINATE2 *m, MATRIX *out)*

- If m->super is WORLD
  - if m->flg is 0
    - output matrix equals m->coord
  - if m->flg is not 0
    - work matrix is still valid, so
      - out equals m->workm

GsGetLw - cont.

*(GsCOORDINATE2 *m, MATRIX *out)*

- If m->super is not equal to WORLD
  - follow super until you reach the WORLD
  - follow logic on previous page to determine what output matrix equals.
  - loop from end of list to beginning, doing this:
    - ApplyMatrixLV(out, current_item->coord->t[0], tmp);
    - MulMatrix(out, current_item->coord);
    - out->t0, t1, t2 equals tmp->t0, t1, t2

GsSetLightMatrix

```
GsSetLightMatrix(MATRIX *mp)
{
        MATRIX  tmpmatrix;
        tmpmatrix = GsLIGHTWSMATRIX;
        PushMatrix();
        MulMatrix(&tmpmatrix, mp);
        PopMatrix();
        SetLightMatrix(&tmpmatrix);
}
```

GsGetLs

- GsGetLw(coord, outw);
- GsMulCoord2(&GsWSMATRIX, outw);

GsMulCoord2

- void GsMulCoord2(MATRIX * m1, MATRIX * m2)
- {
- VECTOR tmp;
- ApplyMatrixLV(m1, (VECTOR *) & m2->t[0], &tmp);
- MulMatrix2(m1, m2);
- m2->t[0] = tmp.vx + m1->t[0];
- m2->t[1] = tmp.vy + m1->t[1];
- m2->t[2] = tmp.vz + m1->t[2];
- }

GsSetLsMatrix

```
void    GsSetLsMatrix(MATRIX *mp)
{
        SetRotMatrix(mp);
        SetTransMatrix(mp);
}
```

GsGetLws

- GsGetLw(coord, outw);
- *outs = *outw;
- GsMulCoord2(&GsWSMATRIX, outs);

GsSortObject5

- Processes each polygon of object, transforming it into screen coordinates and adding it to the order table
  - Loops through polygon list
  - Calls appropriate function for each type of polygon(e.g. gouraud triangle, flat quad,...)

Polygon processing functions

- Each sub function called to process a specific type of polygon:
  - transforms polygon
  - determines polygon facing
  - determines Z distance to polygon
  - Does primitive specific processing
    - lighting, texturing, etc.
  - Adds drawing primitive to order table

The End

TM