Subject: Dev FAQ 1.0
Date: Wed, 30 Dec 1998 20:22:26 -0800
From: Jamin Frederick
Newsgroups: scea.yaroze.announce

Happy Holidays!

Yaroze Dev FAQ 1.0
September 1998

Compiled by Jamin Frederick
Net Yaroze Member SCEA
http://www.cse.psu.edu/~frederic

TABLE OF CONTENTS

SPRITES
1) How do sprites work?
2) What's the deal with video memory -- what's with the 4-bit, 8-bit, and 16-bit stuff?
3) What's (x,y), (mx,my), and (u,v) for in the sprite structure GsSPRITE?
4) Why are my sprites colored wrong?
5) Why do my sprite textures look misaligned, or shifted some, on the screen?
6) Why is my sprite losing pixels, especially along the vertical axis?
7) Do my sprites have to be in a special position within an image?
8) How can I cycle colors on my sprites?
9) How can I do pixel-by-pixel collision detection on my sprites?
10) What other methods of collision detection are there?

IMAGES
1) What is a texture page, and why does the Yaroze system use them?
2) Where do my images have to be aligned in video memory?
3) Can I have more than one image on a given texture page? Can I arrange several images next to each other within the texture page?
4) What's the max size of a sprite? Does it depend on the number of colors?
5) Where are the best places for CLUTs in video memory?
6) How do you set up transparency for a sprite / texture?
7) How do you set up translucency for a sprite / texture / polygon /model?
8) What do the different translucency settings mean?
9) Why do you have to specify translucency for GetTPage()?

BACKGROUNDS
1) How does the 16x16 pixel tiling work?
2) How do I make tiled backgrounds with sprites (bigger than 16x16 pixel tiles)?
3) How does the offsetting work -- I mean, how do you get sprites to scroll around in the background and stuff (2D)?
4) Why am I getting a "banding" effect with my background tiles, where a vertical line in the tile is smeared over a pixel?

3D GRAPHICS
1) How do I create a 3D object?
2) What is the 3D coordinate system of the playstation? I've looked everywhere in the manuals but couldn't find a simple description!
3) What is the direction of positive rotation around the x, y, and z axes?
4) What's the difference between RSD and TMD?
5) Are the coordinates different in RSD and TMD?
6) How can I see information on my RSD?
7) How can I see information on my TMD?
8) Can I save multiple models in one RSD file?
9) Can I save multiple RSD models to one TMD file?
10) What do the coordinates mean in rsdform (option -v)? What is "center"?
11) How do I assign parents to other objects?
12) How do I rotate an object?
13) How do you advance an object in the direction it's facing?
14) What does GsGetLs(), GsSetLs(), and Gs--- actually do?
15) What does gteMIMefunc() do?
16) What's a standard size for a model?
17) How small/big can I make my model?

18) I don't see my object! Where is it?
19) What's the .flg member in GsCOORDINATE2 for?
20) Is it true that the colors specified for an object's polygon is 24-bit, whereas images used for textures on models may only be 4-, 8-, or 16-bit?
21) I've tried to draw TMD lines or sprites as documented in the File Format document, but they don't seem to be working.
22) How do I draw lines in 3D?
23) I've also tried to use double sided polygons with no success.
24) I've set the 'fce' flag in the TMD structure to create double-sided polygons, but it doesn't work. What's wrong?
25) How can I mix 3D objects with 2D sprites?
26) How do TIMs get linked up with objects?
27) Do you have to texture map TIMs onto each individual polygon, as in RSDTool, or is there a way to "wrap" a texture around groups of polygons?
28) Why are my objects all "flat", even when I rotate them?
29) How do I convert a 3D coordinate to a 2D coordinate?

SOUND
1) How do I make sounds play? What programs have to be run?
2) I tried to test my sound FX with VABPLAY but it doesn't work. What's wrong?

EFFICIENCY
1) How can I make my games quicker, taking into account playstation hardware?
2) What are some suggestions to make my 3D graphics a little faster?

CODEWARRIOR
1) What can I do with Codewarrior that I can't do with Siocons?
2) What can I do with Siocons that I can't do with Codewarrior?
3) What's the best compiler setup?
4) What kinds of problems are there with Metrowerk's MWDebugIO library?
5) Why is PSComUtil failing?

YAROZE COMMUNITY
1) What Yaroze tutorials are available?
2) What are some useful general-purpose yaroze utilities?
3) Where is there some useful technical info?
4) Where are some other interesting sites?
5) Are there any chat sessions going on?
6) What yaroze contests are there? Who can participate?

MISCELLANEOUS
1) How do I convert from PAL to NTSC or NTSC to PAL?
2) How can I make my program detect NTSC or PAL an runtime?
3) How do I takes screenshots?

UNANSWERED

DEFINITIONS

In this FAQ, I would like to keep some definitions consistent so that there is no confusion about terminology. If you are adding something to this FAQ, please use the following conventions, thanks.

3D model
---
the information making up the 3D representation of a game object; it includes orientation of the primitives making up the object and their coloring attributes

CLUT

Color Look-Up Table. A color map for 4-bit and 8-bit sprites

display buffer (or display area or frame buffer)
---
portion of video memory that actually gets displayed when the playstation is running

image
---
a 2D picture that is typically created by an artist and used for sprites or model textures

main memory
---
memory used for normal program storage and operation; also the initial place for all incoming code and data when a yaroze project is downloaded

object
---
the arbitrary game object in a Yaroze game, which can have state information attached to it, including sprite and 3D model references

OT
--
ordering table, the thing that sorts primitives such as lines, sprites, triangles, and quads

sound memory
---
memory physically apart from main memory, used for storage and playback of sound

sprite
---
portion of an image that is used to display a game object

texture
---

portion of an image for mapping onto a model

texture page
---
one of 64 portions of video memory that are labeled according to the Playstation API

translucency
---
image or polygons appear "clear", meaning you can see through them, but still allowing original coloring to be seen

transparency
---
doesn't show up at all, like the black background pixels of a sprite

video memory
---
memory physically apart from main memory, used for image storage and blitting to display area of video memory

---

FILE FORMATS

Graphics:

RSD - intermediate model file
TIM - bitmap image file
TMD - Playstation format model(s) file

Sound:

DEF - instrument definition file
SEQ - Playstation music sequence file. Usually generated by SMF2SEQ.
VAB - soundwave data file containing VAGs and a DEF
VAG - individual soundwave data file. Generated by AIFF2VAG or WAV2VAG.
VB - soundwave data file generated by splitting a VAB
VH - soundwave data file generated by splitting a VAB

Utilities:

SIO - common SIOCONS batch file extension

ADDITIONAL SONY UTILITIES

Listed here are the utilities that are additional to the Net Yaroze package received in the mail, and their upgrade histories. They are distributed and supported by the "official" Net Yaroze authorities (SCEI, SCEE, and SCEA).

Rsdcat
v1.04
v1.06

Rsdlink
v3.7
v3.72

Tmdsort
v1.1
v1.25

Aiff2vag
v2.0
v3.1

KNOWN BUGS, ERRORS, OR TYPOS
---

1) In the green manual, RotMatrix(MATRIX *m, SVECTOR *r) should actually be RotMatrix(SVECTOR *r, MATRIX *m)

STARTUP / TROUBLESHOOTING

See the following:
http://www.netyaroze-europe.com/~jaycee/
http://www.netyaroze-europe.com/yaroze/problems/trblsht.htm

1) How do I use the memory card to store the default baud rate?

Information on this is available on the [SCEA] Yaroze web site in the

file BAUDRATE.ZIP located in the MISC section of the file area.

2) Why can't I see all the pixels on my TV screen? Part of the image is cut off on all sides.

Most TV’s do not display the entire image broadcast to them. The size of the border varies from set to set, but for 320x240 mode it can be as much as 27 pixels on all sides. For this reason, it is recommended that you not display information critical to the game in these areas. A good rule of thumb is to limit your game play to the center 266x200 pixels.

3) What can I do if I get "Out of environment space" when I try to build a project?

There are a couple of ways to increase your environment space, depending on your operating system. The easiest is to add "/e:4096" to the SHELL= line in your CONFIG.SYS file. If you do not have a "SHELL=" line in your config.sys file, you can add one. The format for the command is:

SHELL=<full path to command.com> /E:<memory size> /p

If your COMMAND.COM is at the root of your C: drive, the command would be:

SHELL=C:\command.com /E:4096 /p

You may also need to add the following line to your AUTOEXEC.BAT file:

SET GO32=DPMIStack 500000

After making these changes, you must reboot your computer before they will take effect.

4) Why is display from my Net Yaroze is completely grey and grainy / shudders?

You may be running an NTSC configured program on a PAL configured monitor. If you have the source, make sure that the program contains a SetVideoMode(MODE_PAL) command and does not contain a SetVideoMode(MODE_NTSC) command. If you only have the executable, you can pick up the 'Screen Mode Changer' (N!K/Napalm) from the Napalm hacker site (see link section) - it is able to change executables which contain a SetVideoMode(...) command. If, like me, you own an old Amiga monitor (1084/1084S/CM1884), an RGB to SCART cable should do the trick - forget your old Amiga lead!

5) I am trying to use graphic functions, but the screen locks / judders?

These kind of effects can occur when you haven't allocated enough GPU

packet space.

6) What does "BUS ERROR ON DATA LOAD" mean when I upload files to main memory?

a) You're loading code/data into main memory at an improper address. Make sure the address is from 0x80090000 to 0xFFFFFF00.

b) A bus error often occurs when you write words or double words to an odd address. Check you're not uploading to funny or uneven offsets.

YAROZE PLATFORM
---

1) What's this 0x00000000 stuff mean?

This is a hex value, which is 32 bits, meaning 8 hex digits corresponds to 32 binary digits (4 bits per hex digit). The PSX has a 32-bit address space, meaning all operations with memory are done with 8-digit hex values. Some other addresses are 4 hex digits = 16 bits, like device memory, but most things are done with 8-digit hex values.

2) What's the difference between uchar and ushort and ulong?

The "u" stands for unsigned, so any values you stick in the variable can't be negative. This gives you twice the numbers you can get from "signed" variables char, short, and long. Unsigned variables are ideal for memory addresses, though, since they are never negative -- uchar gives you 8 bits of storage, or addresses from 0x00 to 0xff, ushort gives you 16 bits of storage, or addresses from 0x0000 to 0xffff, and ulong gives you 32 bits of storage, or addresses from 0x00000000 to 0xffffffff.

3) Why do I have to load data at 0x80090000? What about those other addresses mentioned in the manual?

For the yaroze system, the only valid memory-mapped addresses (meaning if you stick stuff at these locations, it gets "mapped" to the actual playstation RAM) are 0x80000000 to 0x801fffff. This is exactly 2 megabytes, or 2,097,152 (2 * 2 ^ 20) bytes, to be exact. The other locations are not used on the yaroze. The only valid space for us is 0x80090000 to 0x801fff00 (1,507,072 bytes), since the rest is being used for the system. That's why data is usually started at 0x80090000, and added upwards. Your program may be loaded anywhere in this space, depending on where you tell your compiler to load it, but it's usually loaded up towards the top. There has to be enough space upwards for the

program itself (code and heap), plus the amount the stack will grow. The program's stack usually starts at the top of our usable space, which is 0x801fff00, and grows down until it collides with the loaded program (this is bad). That's why it's up to the programmer to figure out how much his stack will grow, and tell the compiler to place the program low enough in memory so that the stack doesn't end up running into it.

Typical Setup:

```
SYSTEM (OS) STACK (0x801FFFFF)
PROGRAM STACK (0x801FFF00)
  |
  |
  *

(hope the stack and heap don't collide!)

  *
  |
  |
PROGRAM HEAP
PROGRAM CODE (stays same size)
DATA N
  ...
DATA 3
DATA 2
DATA 1 (0x80090000)
OTHER OS GOODIES (0x80000000)
```

You can set the PROGRAM STACK and PROGRAM CODE in your compiler and the DATA 1 - DATA N in your yaroze batch file.

4) What's fixed point?

Fixed point is an alternative to floating point, and is quicker since it actually uses integer hardware and not any special floating-point hardware. It is actually a trick done with integers, and tends not to be quite as accurate as true floating point operations, so you have to watch it when rotating and scaling models, and make sure they're not getting distorted.

The Playstation uses 12-bit fixed point, meaning there's 12 bits of a short (16-bit) integer allocated to the decimal portion of a number (3 bits are used for the integer part, and 1 for the sign). The Playstation uses fixed point for the following cases:

a) Sprite rotation
b) Sprite scale
c) 3D normals

See the following web page for more info on fixed point:

http://www.scea.sony.com/net/yaroze/pages/scartier.html

5) How do I get C++ to work?

Take a look at these web sites:

http://www.netyaroze-europe.com/~c_graham/cplus.html
http://www.scea.sony.com/net/yaroze/pages/cblackwell2.html

6) Are there any naming conventions established for the Net Yaroze when using the memory card to store game information?

Please follow the conventions outlined below. While not mandatory, commercially released games expect save files to be in this format, and may behave unpredictably if they encounter a Memory Card with a file that is not.

File Names

=========

Use the following structure for file names.

Bytes Contents Notes

0 Magic Number Always 'B'

1 Region Japan: 'I' (*1)

North America: 'A'

Europe: 'E'

2-11 "NETYAROZE"

12-20 User/Public Use only non-0x00,

0x2a(*),0x3f(?) ASCII.

End with 0x00.

*1: None are checked by the system

Example: If the product code is SLPS-00001, the file name's first 12 characters are BISLPS-00001. Always add zeros to make the numerical portion 5 digits.

File Headers

==========

Put the following headers at the start of each file.

Bytes Contents Notes

2 Magic number Always 'SC'

1 Type See "Type Field" table below

1 Number of slots

64 Text name Shift JIS, (*1)

28 Pad

32 CLUT

128 Icon image (1) 16 x 16 x 4 bits

128 Icon image (2) Type: 0x12, 0x13 only

128 Icon image (3) Type: 0x13 only

128 x N Data Varies

*1: Non-kanji and primary standard kanji only, full-size 32 characters.

Type Field

========

Type Number of icon images (automatically replaced animation)

0x11 1

0x12 2

0x13 3

7) How do I read the controller?

The controllers are read automatically during the vertical blank interval.
All you need to do is provide a pair of variables to store the controller
state and register those variables with the Yaroze libraries. Like this:

```
volatile u_char *Cont0;
volatile u_char *Cont1;

GetPadBuf(&Cont0, &Cont1);
```

Once this has been done, the controllers can be referenced as Cont0 and

Cont1. For more information see Section 12, "Peripheral Devices Management," in the Yaroze User Guide.

8) In the sample programs I see ResetGraph(3). The manual does not document what the value of 3 is supposed to do.

ResetGraph(3) is essentially the same as ResetGraph(0) except that the current screen mode is maintained and the screen is not cleared.

9) Why does the screen flicker in interlace mode?

In interlace mode, the entire screen must be drawn between Vsync's. In NTSC mode this means that you must be able to draw the entire screen 60 times a second. If not, flicker will occur.

For more information see Section 5, "Frame Buffer Access," in the Yaroze User Guide.

ORDERING TABLE, SORTING, DISPLAY LOOP, ETC.
---

1) What is an Ordering Table?

An Ordering Table is a list of items to be drawn on the screen. The Ordering Table can also implement a simple Z-buffer algorithm to ensure that ‘closer’ objects are not drawn under objects that are ‘deeper’ into the screen. You can think of an Ordering Table as a series of transparent layers laid on top of the other. Any number of objects can be drawn on each of the layers. The layers are drawn from the back toward the front.

For more information see Section 6, "Integrated Graphics," in the Yaroze User Guide.

2) What is a Packet?

A packet is sometimes referred to as a primitive. It is the smallest unit that can be dealt with by the GPU.

For more information see Section 6, "Integrated Graphics," in the Yaroze User Guide.

3) In the sample programs, the packet area is defined as:

PACKET GpuPacketArea[2][PACKET_CNT*(20+4)];

What does this mean?

Four (4) bytes are used for the tag area of the packet. Twenty (20) bytes is the largest possible primitive packet, hence, (20+4) bytes per packet. You may use a smaller value than 20, like (n+4) where n is the size of the largest possible packet in your models. Information on the number and size of the primitives is reported when you run the RSD2TMD tool. Using these values, you can calculate the largest memory block taken by all the primitives in your world and use that area as the GS packet area.

4) How does priority work in the ordering table?

For each "primitive" that you "sort", you can give a priority of 0 to 2^MY_OT_LENGTH - 1, where 0 is the frontmost to the screen, and 2^MY_OT_LENGTH - 1 is backmost on the screen. So obviously, increasing MY_OT_LENGTH gives you more layers, or priorities, to work with.

If you sort primitives at the *same* priority, then the ones you sort first are drawn last, meaning they will appear frontmost (at that priority level) when displayed.

5) How does the display loop work? Do I sort stuff and update stuff every single cycle, or what?

You should sort your primitives each frame (with the appropriate SortXXX() functions), since the screen is entirely updated anew each time DrawOt() is called. However, this does not mean you have to update the other attributes of your game objects at each frame, which may not benefit from being updated as quickly as this. Alternatively, you can keep track of the number of times the screen was redrawn with a variable called LoopCnt, which can be incremented after DrawOt(), and only change game object attributes at certain intervals:

```
if(LoopCnt % Interval == 0)
  ..Do Object Update..
```

So by varying Interval, you can change how often you want your object to get updated -- big Interval for longer times, small Interval for quicker times. Note that LoopCnt can be incremented forever if it is unsigned -- it will just wrap around to 0 again after it hits its maximum.

SPRITES
---

1) How do sprites work?

Sprites are simply references to rectangular regions in video memory that are transferred to the "display" or "frame buffer" position in video

memory when they are "sorted" into the ordering table. So to make these images appear, you must load textures (also called bitmaps or TIMs) into the video memory first, and set up your sprites so that they reference portions of these textures. So what the playstation does is simply make use of the information stored in the sprite to "blit", or transfer, the pixels from one part of video memory to the other. The reason it is done this way is that copying from video mem to video mem is much faster than main mem to video mem.

2) What's the deal with video memory -- what's with the 4-bit, 8-bit, and 16-bit stuff?

TIM images can be specified as either 4-bit, 8-bit, or 16-bit (or 24-bit, but that will be covered later) meaning that there are 4, 8, or 16 bits of information, respectively, for each pixel in the image. However, all image colors on the playstation (besides the 24-bit true color picture mode) are displayed using 15 bits of information for the actual RGB values altogether -- 5 bits for R, 5 bits for G, and 5 bits for B. However, unsigned shorts are 16 bits, meaning there's one bit left over when storing a color -- and this is used for "translucency". This bit is '1' if the color is to be considered for translucency (the docs call it "transparency", but I reserve this word for something else). In other words, if a sprite or model texture using this color would like to be translucent, then the translucent color will be calculated. Otherwise, if the bit is '0', the color will not even be considered for translucency, even if the sprite or texture feels like being translucent. Note that there's a special case if each R,G,B value is 0, and then the remaining bit is considered as something totally different -- which I call "transparency", which is what is used to draw sprites without the black background. In this case, the 0 and 1 stand for the color black to be considered transparent or not when blitting the image.

So even though 16 bits are used for color, what about 4-bit and 8-bit TIMs? Well, obviously, 4-bit and 8-bit TIMs take up less space than 16-bit TIMs, which have a 16-bit value for each pixel, which is the actual color of the image pixel. 4-bit and 8-bit TIMs work differently, though. Instead of using their data bits for actual color, their data bits are used to reference a *table* of 16-bit colors (remember, I said all object colors are really 16-bit). This is where the term "CLUT" comes into play. With 4 bits, 16 different CLUT entries can be accessed, and for 8 bits, 256 different CLUT entries can be accessed. So 4-bit TIMs need to include a 16-value CLUT, and 8-bit TIMs need to include a 256-value CLUT. The CLUT values themselves are 16-bit, though, since they're referring to an actual color (well, actually 15-bit color...this is where some confusion sometimes arises). There are tools to edit the pixel positions and CLUT positions of the TIM image in video memory -- the TIM must include both the pixel information and the CLUT itself.

What about video memory? Well, the video memory is considered to be 1024 x 512 *pixels* -- where a pixel is an unsigned 16-bit color value. But the TIMs contained in video memory (when transferred with LoadImage())

are actually compressed, so that one "video memory pixel" takes up *two* 8-bit TIM pixels, and *four* 4-bit TIM pixels. But remember, these pixels represent CLUT look-up values, not actual 16-bit colors. So the TIM sprites are actually "squished" in video memory, before they get to have fun in their decompressed state over in the display buffer.

So what this means is that all of the sprites you see on the screen are actually 16-bit pixels, not 4-bit or 8-bit CLUT pixels -- so what actually happens when a blit occurs in the video memory with a 4-bit or 8-bit image, is that the playstation automatically references the TIM's pixel color by referencing the CLUT, and expands the pixel in its true color on the display buffer. Pretty neat, huh?

3) What's (x,y), (mx,my), and (u,v) for in the sprite structure GsSPRITE?

The vars x and y refer to the sprite's position on the screen when it is sorted to the OT. However, note that this x and y is with respect to the sprite's center, which is denoted by the vars mx and my. These vars describe the "center" of the sprite, which is the thing that gets placed at x and y on the screen. This can be used if you want to refer to some other part of the sprite as the center, instead of the upper left corner. Note that x and y can be negative, or even larger than the screen, but this does not cause an error. The OT compensates and automatically "clips" the sprite if it is partially or even totally off screen.

The u and v refer to the *pixel* location of the sprite's upper left corner with reference to its parent image (or texture page, or TIM). This can be considered as if the sprite is already expanded from its original 4-bit or 8-bit compressed image size. So if my sprite starts on pixel 20 over and on pixel 10 down in my TIM (just like it is painted, not worrying about compression), then you just specify (u,v) = (20, 10). The playstation takes care of the actual referencing in video memory, which takes into consideration the compression and the texture offset in video memory, via the information included in the TIM file.

4) Why are my sprites colored wrong?

a) Check the CLUT position in TimTool. All CLUTS must have unique positions in video mem; if any of them are the same, one CLUT will get loaded over another, using the color scheme of the other picture.

b) Check the image positions of all images within the video memory with TimTool. If any images are overlapping, you will get this effect. Make sure no images are hidden underneath another, either.

5) Why do my sprite textures look misaligned, or shifted some, on the screen?

Double check the alignment of the texture in TimTool. Sometimes TimTool

nudges the texture down some, messing up what you might think is the proper offset for the texture.

6) Why is my sprite losing pixels, especially along the vertical axis?

Try changing sprite scaling from 4096 to 4095.

7) Do my sprites have to be in a special position within an image? I heard they have to be on even coords or something.

You may get display problems if your sprites are not aligned on appropriate boundaries:

"Sprite data (or TIMs) should be horizontally aligned on 16-bit bounderies, i.e., 4-bit images should be on 4 pixel boundaries, and 8-bit images on 2 pixel boundaries."

"Fast Sprites (not scaled or rotated) must have even u co-ords, and even widths (Because the GPU renders them 2 pixels at a time). The v position is not limited however."

8) How can I cycle colors on my sprites?

Use MoveImage() to make a copy in video memory of the CLUT of the image that the sprite is using (to somewhere below the original CLUT), and between update frames, use MoveImage() to copy colors from the copied CLUT into the original CLUT, and the palette colors that are mapped to the sprite will give it an appearance of cycling, if it's set up right. Check out Jamin Frederick's palette library, which you can use to do this.

9) How can I do pixel-by-pixel collision detection on my sprites?

Typically, you can use just bounding boxes to see if sprites collide, but sometimes it is necessary to get down to the pixel level. One way to do this is to first see if the bounding boxes intersect, and if they do, then you can do a pixel-by-pixel collision detection on the intersection rectangle of the corresponding sprite images. The only problem is that these images are in video memory, unless you haven't written over the images that are in main memory when they originally were copied over from the com utility. In other words, we don't really have byte-by-byte access to the video memory for close comparisons, so we are forced to work with main memory. So once you get access to the intersection rectangles of the sprite images, all you have to do is compare corresponding pixels in the rectangles, one at a time, until you get a "hit" (non-black) pixel from one image with a "hit" pixel from the other image. However, this involves some rather tricky interpretation of the TIM image data in main memory.

10) What other methods of collision detection are there?

>From a newsgroup discussion:

"Have you considered whether a pixel-by-pixel collision detect is *really* necessary? It sounds so time expensive when you could be doing some freaky special effects or better AI or something instead... I've used gobs of weird hacks for collision detection in games, and I've always been able to find a way around pixel-by-pixel. I thought I'd list some of them off:

0. Make your game so intense no one has time to notice that collisions are pixel exact or not.

1. Simple shapes as hit regions - design the art so it fits well into shapes that are easy to mathematically check against, like rectangles, circles, and triangles.

2. Rotated simple shapes as hit regions - sometimes you've got objects that can rotate - rotating a rectangle and checking it against another rotated rectangle is still cheaper than pixel-by-pixel.

3. Stick Figures. Say you've got a figure like a character in a 2d fighter. Make a simple stick figure that mimics the shape and motion of the sprite, ie: a line for the head, a line for upper arm, lower arm, hand, two lines for the torso, etc. etc. Then to do hit detect with the other character in the game all you have to do is treat each line as a fat line and check them against the fat lines in the other character. This has the advantage of letting you know that the hand connected with the noggin.

4. Simple hierarchical shapes. Say you've got a top down view of the TOS U.S.S. Enterprise on the screen. Do your hit detecting on a circle and three rectangles, one for the two engines and one for the engineering hull, with the rectangles of course rotated according to the orientation of the ship on the screen.

5. Polygon shapes. Make a polygon outline that fits around the shape as tightly as you'd like so that game feels good, then check for intersection with collidable objects. If you design it right you can have one complicated polygon collision shape which you test against some trivial shape like a rectangle, but even still, poly-poly intersection is faster than pixel-by-pixel, unless your sprites are really tiny, in which case suggestion number 1. is probably the best IMO." -- Nick Porcino

---

IMAGES

---

1) What is a texture page, and why does the Yaroze system use them?

A tpage is a memory area in the frame buffer that stores images used for texturing a 3D model, or 2D sprite. They are used to make it easier to reference a portion of the frame buffer. A texture page ID is simply a form of shorthand for referring to a specific rectangular block of the frame buffer using a single byte rather than a complete set of coordinates.

This also allows the texture UV coordinates to be specified as 8-bit values that reference an offset from the start of the texture page, instead of 16-bit values that reference an offset into the frame buffer as a whole.

2) Where do my images have to be aligned in video memory?

They should be aligned on 16-bit boundaries. Try to ensure that your TIM stays within one of the 32 designated texture pages.

3) Can I have more than one image on a given texture page? Can I arrange several images next to each other within the texture page?

Yes, as long as you abide by 2).

4) What's the max size of a sprite? Does it depend on the number of colors?

Max size of a sprite is 256 x 256 pixels, no matter how few colors it is. One of the reasons is because sprites are referenced (u,v) via pixel offsets of texture pages (or TIMs), and since u and v are both unsigned chars, the biggest they can both be is 255, meaning sprite (u,v)s can only be from (0,0) to (255, 255).

Here's an informative snippet from a newsgroup:

"Sprites can be a maximum of 256 * 256. It's important to distinguish between the RENDERED width (the width you see on the screen) and the DATA width (the width of the image in VRAM). If you're using a 15-bit TIM, the rendered width is the same as the data width. For an 8 bit TIM, the rendered width is double the width of the data width, because the data is compressed into half the space. And for a 4 bit tim, the rendered width is 4 times the size of the data width, for similar reasons.

The size of a texture page is always 256x256 _rendered_ pixels, but in terms of VRAM this might mean 256, 128 or 64 VRAM words (16 bits each) depending on whether you're using 15bit, 8bit or 4bit tims.

When you're specifying UV coords, it's in rendered pixels, not VRAM words. Thus the maximum UV coords you can specify are always 255,255, and the

smallest is 0,0. If you want to display something bigger, you'll have to split it up into more than one sprite."

5) Where are the best places for CLUTs in video memory?

The typical convention is placement below the display buffers, since the area is too small to have any decent-sized sprites down there.

6) How do you set up transparency for a sprite / texture?

"Ensure that when you converted your bitmap to TIM format that you have enabled the transparency bit. This can be done by putting a checkmark in the appropriate boxes in TimUtil. Then, when setting up your GsSPRITE structure, make sure the transparency attribute is set correctly."

More specifically, each 15-bit color in a TIM has an additional bit left over in the 16-bit word. When this 15-bit color is black (meaning all 15 bits are 0), then the left-over bit determines whether the black is transparent or not when the sprite or texture is displayed on the screen (only the color black can be used for transparency on the Playstation). If it is 0, then the black is transparent (not drawn). If it is 1, then it is drawn as black. This tranlucency applies to all individual sprites or textures using the particular color (either direct or as a look-up value).

You do not have to specifically tell a sprite or texture to turn on transparency processing; setting the last color bit of black to 0 in the TIM as explained above is enough.

7) How do you set up translucency for a sprite / texture / polygon / model?

When a 15-bit color in a TIM is non-black (something other than all 15 bits zero), it is able to become a translucent color. This means that any sprite or texture using this color will show it as translucent only if the remaining bit left over (of the 16-bit color word) is 1, AND the translucency feature is turned on for the particular sprite or texture. Note that a sprite or texture also has one of four translucency settings that can apply when the translucency feature is turned on.

For sprites, the translucency flag and settings are contained in the attribute field of the GsSPRITE struct. For textures, the translucency flag (to make either the texture on the polygon or the polygon itself translucent) is within the TMD > PRIMITIVE > MyPolyPacket > Mode data block, and settings (to set the translucency level of just the texture) are within the TMD > PRIMITIVE > MyPolyPacket > TSB data block. These individual polygon settings can more easily be specified in a modeler tool such as RSDform, however. If you want to change model polygons dynamically, though, then you'll have to fool around with the TMD data

blocks in memory.

It seems that translucency must be applied on a sprite-by-sprite or polygon-by-polygon basis, so you must turn on each polygon to translucent in order to make an entire model translucent.

8) What do the different translucency settings mean?

 50% back + 50% polygon
100% back + 100% polygon
100% back - 100% polygon
100% back + 25% polygon

This is additive translucency which manipulates the RGB values per pixel based on the new image's RGB values. For example, if you use the "50% back + 50% polygon", then for each pixel in the display buffer (where the new image is being placed) the old pixel's RGB values are reduced by 50% intensity and are added to 50% of the intensitiy of the new image.

9) Why do you have to specify translucency for GetTPage()?

"You can use GetTPage to calculate the tsb section of a tmd primitive which does require the semi-transpency rate. Just & the result with 31 to get the Texture page number." -- Jim

---
BACKGROUNDS
---

1) How does the 16x16 pixel tiling work?

There is a special Playstation function called GsSortFixBg16(), which will automatically sort an entire background composed of 16x16 pixel tile images if you set it up right.

2) How do I make tiled backgrounds with sprites (bigger than 16x16 pixel tiles)?

This is a nifty trick. The ordering table accepts information in the sprite struct (GsSPRITE) that you give it as a *new* sprite, so that you don't have to maintain the same information in the struct within a screen update. So what you can do is keep changing the info in the sprite struct and sort it into the OT. So to make a checkered background, for instance, you just need a black square struct and a white square struct, and repeatedly sort them while changing their positions.

3) How does the offsetting work -- I mean, how do you get sprites to scroll around in the background and stuff (2D)?

It usually depends on the game, but here it goes...

First of all, let's draw a difference between a game object and a game sprite. A game object is something in the game that has a state and presence, but is not necessarily displayed. A sprite is just a representation of the object ON THE SCREEN, and always refers to the thing being displayed. So an object can easily have many sprites, each one representing different states of the object, or an object's current position on the screen.

Ok, so first you set up a world coordinate system, where all of the objects that you intend to have in the game are given a position in the world. So all the changes that you give to your objects are with respect to the world coordinates, NOT THE SPRITE COORDINATES. The reason for this is that the sprite coordinates can be calculated from the object's position in the world with respect to the SCREEN's position in the world. That's right -- you consider the "game screen" to have a position in the world as well, so that when you want to "scroll around" the world, you're just moving the screen's position around in the world, and the sprites that get displayed on your screen are just a result of them being around the screen when it gets close to them in the world.

So for instance, if you chose your world to start at (0,0) in the top left corner, then you could assign objects coordinates like (8,20) or (67,258) or (600,780) or even (4563,23453), depending on how big you want to make your world. Of course you're limited by how large your numbers can get, which for an unsigned long (32-bit) is 0 to 4,294,967,295. So just imagine that you have a bunch of (rectangular) objects positioned in your world like this (usually when you're referring to an object's world coord, it is the upper left corner of its rectangle, at least in this example) and think about a rectangular, 320x240 screen, moving and scrolling around in the world with them (again using the upper-left corner pixel as the world coord). Now to get them to be on the proper position on the screen, you want their sprite representations to be according to where the object is in reference to the screen in the world. If the object stays still and the screen moves, for instance, well then the sprite moves on the screen even though the object hasn't moved in the world at all. So what you have to do is set up a relation between the sprite and the screen. It's just:

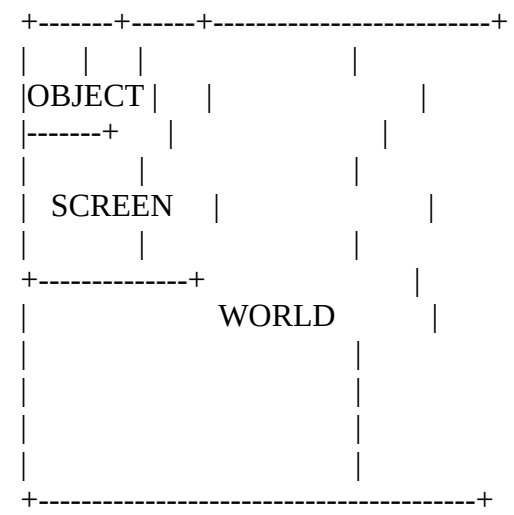
```
SpriteX = ObjectX - ScreenX
SpriteY = ObjectY - ScreenY
```

So if the object position gets bigger with the screen position fixed, then the sprite position gets bigger, and moves right and down on the screen. If the screen position gets bigger with the object position fixed, then the sprite position get smaller, and moves left and up on the screen. Now remember, this example is all in terms of the origin

of the world being to the upper left, and the origin of the objects and screen at the upper left of their bounding boxes as well.

When you give the OT a sprite to sort, you're giving it the sprite's SCREEN position. From experimentation I have found out that this does not mean it has to be from (0,0) - (SCREENW-1, SCREENH-1), but it can be any number at all, even negative! The Playstation compensates and does all the clipping for you, so that the formula above works fine when using SpriteX and SpriteY as the position to be be placed on the screen.

If you have trouble visualizing it, just start out with 1 object and the screen at the origin of the world, and imagining what happens to the sprite image on the screen when you try to move either one according to the formula:



4) Why am I getting a "banding" effect with my background tiles, where a vertical line in the tile is smeared over a pixel?

The individual tile textures of a background each have to be on an even u coordinate. If each sprite texture is not positioned on an even number in the video memory, you will get this effect. This can easily happen if you use gridlines to separate your tiles for easy editing, for instance.

3D GRAPHICS

---

1) How do I create a 3D object?

3D objects can be created using a variety of modeling and/or CAD programs. Any program that can output a 3D DXF file can be used. Once you have created a DXF file, you run the program DXF2RSD.EXE to convert it into a

format usable with the Yaroze.

For more information see Section 14, "Graphic Tools," in the Yaroze User Guide.

2) What is the 3D coordinate system of the playstation? I've looked everywhere in the manuals but couldn't find a simple description!

It's actually a right-handed coordinate system, meaning (with your right hand) +X cross +Y gives you +Z. The +X axis is to the right of the screen, +Y is down on the screen, and +Z is in towards (behind) the screen.

3) What is the direction of positive rotation around the x, y, and z axes?

Each positive rotation goes counter-clockwise when the positive axis is pointing at you. Again, it's the right hand rule: put your (right) thumb in the direction of the positive axis, and curl your fingers. The direction your four fingers point and curl is the direction that things get rotated about that axis.

4) What's the difference between RSD and TMD?

RSD is the file format used to convert from modelers, similar to DXF and other model formats. It is in ascii, making it easy to edit by hand. The info making up an RSD is actually four files, .rsd, .ply, .mat, and .grp. It's easy to mix up talking about RSD (which includes all four files) with just .rsd. A modeler converting to RSD will generate all four of these files. Also, an RSD refers to only one model, and the grouping information (.grp) in the model doesn't carry over to TMD.

TMD is the file format the playstation uses, and the data making up this file actually resides in memory when your Yaroze program is drawing objects on the screen. A TMD can have more than one model, and each model has its own set of primitives, which are made up of those model's own vertices and normals. Each primitive is described in detail as a triangle, quad, flat shaded, gourand shaded, colored, color gradated, or textured. All the combinations are described in the Sony online docs.

5) Are the coordinates different in RSD and TMD?

RSD and TMD both use floating point for coordinate x, y, and z of points, but they differ in the polygon normal representation. RSD uses a floating point number for each normal component, while TMD uses fixed point (12:3), with 4096 (2^12) representing one coordinate unit.

6) How can I see information on my RSD?

To view an RSD, use RSDTool. To get information on the vertex extents and "center" position, do rsdform -v mymodel.rsd. Follow this with > to output to a file.

7) How can I see information on my TMD?

To view a TMD, try this:
http://www.netyaroze-europe.com/~sevans/ftp/demos/yaroze/tmdview.ZIP

You can also get vertex information on the TMD when you are converting from the RSD with rsdlink, just do rsdlink -v mymodel.rsd or rsdlink -info mymodel.rsd. Follow this with > to output to a file.

8) Can I save multiple models in one RSD file?

Yes and no. An RSD contains only one model per file, so if you wish to put several models into one RSD with rsdcat, you are really only merging several models into one big one, and in effect, merging three coordinate systems into one.

9) Can I save multiple RSD models to one TMD file?

Yes. Unlike RSD files, TMD files can make references to several different models within the file. To put several RSD models together into one TMD, you can use rsdlink with several RSD file names as arguments (see top of p.149 in yellow manual).

10) What do the coordinates mean in rsdform (option -v)? What is "center"?

These coordinates refer to the offsets from the model's coordinate system. If you "translate" the model with rsdform, you are moving all the vertices within the fixed model coord system, and you can imagine a bounding box of all the vertices that comprise the primitives of the model moving around in space with respect to the model's fixed coordinate axis. The "center" is just a reference to the center of mass of the model, which can usually be specified within a modeler program. The playstation will rotate models around (0,0,0) of the coordinate system, *not* the "center", which can be a nonzero offset from (0,0,0).

When you save an object in a modeler and it is not placed in the middle of the world, for instance at (8, 10, 12), the "center" in your RSD will appear as (8, 10, 12). To get the model back to the origin (so that it rotates correctly) you need to translate back to (-8, -10, -12). However, many times you will want to use this information, for instance, if you have several objects and need to know their relative offset position, such as a helicopter and its blades. You would need to know how high the blades are in the world originally, so that you can offset them with

respect to the helicopter when you're actually drawing them in your program.

The origin of the model's coord system (0,0,0) is the point that is referred to when translating the model in your program. So if I give m model's translation vector (.t) in my program the value of (5, 2, 10), the model's origin will be at (5, 2, 10), and all the primitives will be drawn around this point.

11) How do I assign parents to other objects?

When initializing your object's coordinate system, instead of doing GsInitCoordinate2(WORLD, &MyObjectCoord), do GsInitCoordinate2(&MyParentCoord, &MyObjectCoord), assuming you already initialized &MyParentCoord with WORLD or some other coordinate system. Each argument is a pointer to type GsCOORDINATE2, which includes a matrix that describes the coordinate system you are using. So then, after the initialization is done, all of the offsets of your model are with reference to the parent coordinate system. For example:

```
GsCOORDINATE2 MyParentCoord;
GsCOORDINATE2 MyObjectCoord;

// the coords of an object with MyParentCoord are in the world
GsInitCoordinate2(WORLD, &MyParentCoord);

// the coords of an object with MyObjectCoord are w.r.t. the parent
GsInitCoordinate2(&MyParentCoord, &MyObjectCoord);
```

So if you had an GsDOBJ2 handler representing a tank, make its coord2 member point to MyParentCoord (the tank is the parent), and if you had a GsDOBJ2 handler representing the tank turret, make its coord2 member point to MyObjectCoord (the tank turret is the child). Now if the turret is at (0,0,0), it will be at the tank model's origin. You'll probably want to tranlate the turret to (0, -1000, 0) or so, depending on your model, and everywhere the tank moves (in the world), the tank turret will follow.

12) How do I rotate an object?

First you have to know that all models rotate about the origin (0,0,0) of your model's coordinate system (do rsdform -v mymodel.rsd). This means that however your primitives are placed in your RSD, they will rotate around this point. So if the "center" of your model (representing the actual center of your model) is not (0,0,0), translate it (with rsdform -t <x> <y> <z> mymodel.rsd) there first before you rotate it.

Assume there is an initialized GsCOORDINATE2 Coord structure (belonging to the object which I am rotating), and I want to rotate the object (RotX, RotY, RotZ) more than it was before. The model coordinates represented in the Coord.coord structure will be transformed from where they were the

last time, destroying the last position information:

```
MATRIX TempMatrix;
SVECTOR RotVector;

// this makes a vector
RotVector.vx = RotX;
RotVector.vy = RotY;
RotVector.vz = RotZ;

// this turns the angular displacement into a matrix, so that I can
//  multiply matrices
RotMatrix(&RotVector, &TempMatrix);

// multiply original coord matrix by "rotation" matrix, which changes
//  the position coordinates
MulMatrix0(&Coord.coord, &TempMatrix, &Coord.coord);

// object should be redrawn now since it changed
Coord.flg = 0;
```

Note: Your model may become distorted after several rotations if this method is used. "The problem with matrix concatenation is that precision errors accumulate and build up to produce scaling and shearing of your object. This happens very quickly for integer based matrices [which the playstation uses] but also occurs for eventually for floating point matrices." Try the following alternative to avoid this error accumulation:

```
GsCOORDINATE2 Coord;
SVECTOR RotVector = { 0, 0, 0 };
SVECTOR Position  = { 0, 0, 0 };
```

I want to rotate this object by 100 degrees around the Y, and move it to the position X=20, Y=30, Z= 50;

```
MATRIX TempMatrix;
SVECTOR RotVector;

RotVector.vy = 100 * 360 / 4096;  (4096 = 360 degrees)
Position.vx = 20;
Position.vy = 30;
Position.vz = 50;

// Creates the rotation matrix.
RotMatrix(&RotVector, &Coord.coord);

// Fills in the Translation part of the rotation matrix.
RotVector.coord.t[0] = Position.vx;
RotVector.coord.t[1] = Position.vy;
RotVector.coord.t[2] = Position.vz;

// Let the system know that this matrix has changed.
```

```
Coord.flg = 0;
```

13) How do you advance an object in the direction it's facing?

Assuming that your model starts out pointing in the +z direction, and you want to advance U units, and that your object has a GsCOORDINATE2 Coord struct:

```
SVECTOR StartVector;
SVECTOR CurrentDir;

// assume origin points exactly towards +z direction
StartVector.vx = 0;
StartVector.vy = 0;
StartVector.vz = ONE;

// multiply original orientation (start vector) by current orientation
//  matrix, to get the current direction vector
ApplyMatrixSV(&Coord.coord, &StartVector, &CurrentDir);

// add a multiple (units) of the current direction to the current
//  translation
Coord.coord.t[0] += (U * CurrentDir.vx) / ONE;
Coord.coord.t[1] += (U * CurrentDir.vy) / ONE;
Coord.coord.t[2] += (U * CurrentDir.vz) / ONE;

// the object should be updated now
Coord.flg = 0;
```

Note: You could use 1 instead of ONE=4096, and then not divide later on by ONE, but you'd get considerable error if you don't "pump up" your start vector to a suitable size because of the integer precision.

14) What does GsGetLs(), GsSetLs(), and Gs--- actually do?

GsGetLs() will create the Local to Screen matrix. It also checks the flg variable and recalculates the product of the parent matrices if necessary. This means the coordinate system hierarchy will contain the correct Local to World matrices. After the function call, TempMatrix will contain a matrix which will translate this coordinate system to Screen coords.

```
    GsGetLs(&Coord, &TempMatrix);
```

GsSetLs() sets up the Local -> Screen matrix in the GTE. Any subsequent GsSortObject() calls will use this matrix.

```
    GsSetLs(&TempMatrix);
```

"The Lw Matrix describes the transformation of the local coordinate system to the world coordinate system. The lihting subsystem of the PSX needs

this matrix to perform the calculations (probably for the transformation of the normal vectors of surface elements). The Ls Matrix transforms the local coordinates into viewing coordinates, where the origin is positioned at the Viewing Reference Point and the Z-Axis is pointing into viewing direction and the -Y-Axis is poiting into the viewing up vector direction. The Application of the Ls Matrix is perfomed juts before the perspective transformation is done. As you usually perform lighting and drawing of an object, you have this GsGetLws method that returns both matrices."

"Your 3D object is described in local coordinates, and must be transformed into World coordinates, and then finally into Screen coordinates before it can be drawn. GsGetLs creates a matrix which will perform these two transformations in one go. However, any lighting must be performed using the Local->World matrix. You set up the lighting in the GTE with the Local->World matrix.

If you're not performing any lighting, then you can ignore this matrix, but if you want to perform lighting, then you call GsGetLws() which will return both the Local->World and the Local->Screen matrices. You then set the GTE's lighting parameters with the Local->World matrix(using GsSetLightMatrix()), and the GTE's point transformation parameters with the Local->Screen matrix (using GsSetLsMatrix())."

"If you use the Local->Screen matrix as the Light Matrix, the lights will be relative to the view, as if you've got some lights attached to the camera. With the Dino demo, this doesn't really matter much. In most games the lights are fixed in relation to the world, not the camera, which is why the intermediate LW matrix is required."

15) What does gteMIMefunc() do?

Fast addition of one array of SVECTORS to another array of SVECTORS. With a scaling factor.

16) What's a standard size for a model?

Typical sizes are about 500.0 to 2000.0 in each dimension.

17) How small/big can I make my model?

If a model is too small (like 5.0-10.0 or so), the view will have to get very close to the model to see it, making viewing difficult and distorting textures.

If a model is too large, (>2000.0 or so), it may "blink out" when the view gets far away from it, since the view can only see up to 16 bits (-32768 to +32768) away. The model can be placed anywhere in the world, however, which is 32 bits (-2147483648 to +2147483647). If the model is small enough to begin with, though, it will shrink in size before this

happens.

18) I don't see my object! Where is it?

First, see if you're actually sorting it with GsSortObject4() -- of course, there's some stuff you have to do before you can actually sort it, like calling GsGetLws(), GsSetLightMatrix(), and GsSetLsMatrix(). You should also make sure the loading of the model was done right, too.

Second, see if the scale is big enough. Modelers tend to save out RSD models in the -2.0 to 2.0 range, which is pretty small for a standard playstation model. Scale it up with something like rsdform -s 512 512 512 mymodel.rsd, to give it a size big enough to see at standard viewing distances, which are 500.0 to 1000.0 units away.

Third, make sure the view is not too far away, and that your model is not too big, since you can only see up to 16 bits away with the view. (see #17)

19) What's the .flg member in GsCOORDINATE2 for?

The flg variable is used to indicate that the 'workm' matrix in the GsCOORDINATE2 is valid. The 'workm' matrix is identical to the 'coord' matrix for the top coordinate system in a heirarchical model, but in coordinate systems further down the tree it is the matrix you get when you multiply the 'coord' matrix and all this GsCOORDINATE2's parents' coord matrices together - thus creating a matrix which will translate this coord system's to WORLD coordinates. The flg variable is used to determine whether or not it should recalculate some or all these matrix multiplications (which is performed internally by the libraries), because if one matrix changes, all its children must change too.

20) Is it true that the colors specified for an object's polygon is 24-bit, whereas images used for textures on models may only be 4-, 8-, or 16-bit?

Yes.

21) I've tried to draw TMD lines or sprites as documented in the File Format document, but they don't seem to be working.

Even though they're documented, 3d lines and sprites cannot be used on the yaroze. See the next question.

22) How do I draw lines in 3D?

There is no library function for drawing 3D lines. What you need to do

is map the endpoints of the line to the screen and draw a 2D line (using the GsSortLine() function) connecting the points. If you are connecting the vertices of a 3D object with lines, you can use the function GsGetLs() to obtain a local-to-screen transformation matrix. This matrix could then be applied to the vertex coordinates to obtain the endpoints of the line.

23) I've also tried to use double sided polygons with no success.

Again, double-sided polys are not supported on the Yaroze. Try using the option on Dxf2Rsd to make double sided polys into two one-sided polys. See the next question.

24) I've set the 'fce' flag in the TMD structure to create double-sided polygons, but it doesn't work. What's wrong?

The 'fce' flag and double-sided polygons are not supported on the Yaroze system. However, when you are converting the DXF file to an RSD file, you can use the –both option to simulate double-sided polygons. This option works by creating two polygons (one for each side) for each polygon in the original model.

25) How can I mix 3D objects with 2D sprites?

The best way to do this is to create a single polygon for each 2D sprite and texture it with the sprite image. This method makes it easier to scale, rotate and otherwise fit the sprite into the 3D environment.

26) How do TIMs get linked up with objects?

When you do the rsdlink command, your TMD gets info about the textures you are using for the mapping to the object(s) from the TIM files referenced by your object. Make sure that if these textures change at all, such as tpage location or clut location, that you redo rsdlink.

27) Do you have to texture map TIMs onto each individual polygon, as in RSDTool, or is there a way to "wrap" a texture around groups of polygons?

There doesn't seem to be any tools out right now to do this, although the next version of RSDTool was supposed to incorporate this feature.

28) Why are my objects all "flat", even when I rotate them?

You forgot to set the coordinate system of your object, i.e., make your GsDOBJ2 object handler coordinate pointer (.coord2) point to a validly initialized GsCOORDINATE2 struct.

29) How do I convert a 3D coordinate to a 2D coordinate?

--SNIP--

```
void InitTransProj(void);
void TransProj(VECTOR *pos, short *x, short *y);

static GsCOORDINATE2 trans;

//
----------------------------------------------------------------------
//      init trans proj coord
//
----------------------------------------------------------------------
void InitTransProj()
{
        GsInitCoordinate2(WORLD, &trans);
}

//
----------------------------------------------------------------------
//      Trans Proj - convert 3d x,y,z to 2d screen x,y
//
----------------------------------------------------------------------
void TransProj(VECTOR *pos, short *x, short *y)
{
        MATRIX mat;
        VECTOR v;

        trans.coord.t[0] = pos->vx;
        trans.coord.t[1] = pos->vy;
        trans.coord.t[2] = pos->vz;
        trans.flg = 0;

        GsGetLs(&trans, &mat);

        ApplyMatrixLV(&mat, pos, &v);

        if(mat.t[2]) {
                *x = ProjectionDistance * v.vx / mat.t[2];
                *y = ProjectionDistance * v.vy / mat.t[2];
        } else {
                *x = 0;
                *y = 0;
        }
}
```

--SNIP--

Contributed by Steve Hunt <steve@itallnight.u-net.com>

SOUND
---

1) How do I make sounds play? What programs have to be run?

Say I want to use 3 .wav files, a.wav, b.wav, and c.wav. First you do "aiff2vag a.wav b.wav c.wav" to create .vag's out of the .wav's. Then you have to type "mkvab -f sounds.def a.vag b.vag c.vag -o sounds.vab" where sounds.vab is the output .vab and sounds.def is your definition file. What's that? It's a file you need to provide to make a vab. The vab can contain programs, and for each program, a tone. You set it up so that there's a .vag for each tone. You have to set this by hand via the .def file. To see other people's .def files for their programs, type "mkvab -r person.vab -o person.def" where person.vab is their vab, and person.def is the output .def that you want to look at. In the .def file is where you associate programs and tones. A program can have one or more tones, but the way it is usually set up is that there's a program for each sound, and each program only contains one tone, the .vag that you specify. Sometimes people use maybe two programs and say ten different tones, and maybe 4 tones in the first program and 6 tones in the second program, where each tone is associated with a vag. I don't know why, though. Maybe someone can answer this for me. Anyway, suppose you set up your .def so that program 0 has two tones, a.vag and b.vag, and program 1 has one tone, c.vag. So to play the sound, you do "SsUtKeyOn(vabid, prog, tone, note, fine, voll, volr);", where prog is the the program for the sound you want to play as specified in your .def file, such as prog 0, and tone is the tone of that prog as specified in your .def file, such as tone 0 or tone 1. The note is the pitch to play the sound. At 0, it's normal. Increasing the pitch is like going up the keys of a piano. The fine parameter is just fine pitch. And voll and volr is the left and right volumes. But what about vabid? Well, when you call SsVabTransfer(), you get returned a vab id. This is the vab that is being used for your sound calls. The vab got loaded into sound memory when you called SsVabTransfer(), where you have to specify the VH and VB type files that have been loaded into main memory from siocons. How did they get into main memory in the first place? Well, all you have to do is split up your vab you made, sounds.vab, with "vabsplit sounds.vab". Then you got the files sounds.vh and sounds.vb, which you just load into the psx memory via siocons. Using the syntax of SsVabTransfer(), you just transfer them into sound memory, and you're ready to make sound calls.

2) I tried to test my sound FX with VABPLAY but it doesn't work. What's wrong?

VABPLAY uses a .SEQ file to key the sounds and is used primarily for testing your game music. If you want to test your sound FX, you'll need to create a VAB containing the sound FX. Load the VAB in SPU using

SsVabTransfer() and save the return VabId. Set the main volume using SsSetMVol() and use SsUtKeyOn() with proper parameters (including the VabId) to play the sound effect. (Please refer to LibSound.c and LibSound.h in the game Survival for a simple implementation of playing background music and sound FX).

EFFICIENCY
---

1) How can I make my games quicker, taking into account playstation hardware?

a) Make use of the DCACHE in functions that you call very often; see Scott Evans' (SCEE) DCACHE article. Also, from a newsgroup post:

```
-------------------------------- SNIP -------------------------------
>
>   Anyone mind if I jump in here? Just taking up on your comment here,
>about putting the stack on the D-cache - in what cases would you want
>to do this, and when? Would you put in this code:
>
>        __asm__ volatile ("sw $29,(savesp)");
>        __asm__ volatile ("la $29,0x1f8003f0");
>
>    before a particular function call, and this:
>
>        __asm__ volatile ("lw $29,(savesp)");
>
> afterwards?
```

Yes, that looks right. The first bit saves the current stack pointer and loads in the new one, and the second bit restores the old stack pointer.

> For what sort of functions would I want to do this?

Well, for starters, as a general rule I wouldn't call any Yaroze library functions while your stack is on the D-Cache, which probably rules out a few functions that you want to speed up. Many of the library functions don't change the stack to the D-Cache, but some do to get extra speed. If your program is running with a D-Cache stack and you call a library function which resets the stack to the D-Cache too, your program will crash and burn because the new stack will overwrite the old one.

GsSortObject4 doesn't reset the stack (to my knowledge), but takes as a parameter a 'scratch' area to use for its intermediate workspace. If you've followed the sample code, you'll see that they use getScratchAddr(0) for this scratch area, which is a macro that points to the start of the D-Cache.

To be honest, I can't think of any obvious examples where using the D-Cache _as_a_stack_ would bring you a huge speed increase. But here's 3 reasons:

1) If you're writing a function that uses a lot of local variables (more than the number of registers available), then those variables will be allocated on the stack (and hence on the D-Cache), and therefore they'll go a bit faster.

2) If you are doing some major processing on a local array which is less than 1K, then having the stack on D-Cache will (generally) increase the speed of that function.

3) If you are doing a tree traversal (depth/breadth first, that sort of thing) which involves a lot of recursive function calls, then having the stack on D-Cache will be faster. The only proviso is to make sure that there aren't too many local variables and/or the tree is not too deep, or you'll overflow the D-Cache!

The D-Cache isn't a true cache in the usual sense of the word. A normal cache will _transparently_ store the most recently used lines of RAM to increase speed. The D-Cache is more like a really fast area of memory, but it's only 1K long. Thus it's up to the programmer to explicitly load and store parts of this memory, which is why most people set up their stack on it, because it gives an instant speed increase to local variable access.

If you want to process a global/static array, it's going to be stored on the heap and so you'll have to transfer it to D-Cache before you start, and transfer it back after you finish. This transfer overhead is only worth it if you're going to be accessing each element of the array more than twice. This is certainly the case if you're doing some image processing (like the flame/water effects).

The first heuristic of optimisation is to optimise the biggest timewaster. Back in the days when I was writing Unix database code, I managed to speed up a debugging function that was used twice in every function by a factor of 8. But since 90% of the time was spent preparing and parsing the SQL, the speed increase from the new function hardly made a dent in the performance. The lesson there is that you should concentrate on optimising the component which takes the longest time to complete.

If you want to time various parts of your code, use the VSync(-1) call or the Root counters. Run various important pieces of code in a loop a million times and see how many VSyncs each part takes. That will give you some idea of the proportion of time that code is taking."
-- James Russell, SCEE

---------- SNIP ----------

b) Declare variables static in functions that you call very often, so fewer local variables need to be pushed and popped.

c) Use lookup tables wherever possible, which are just arrays of pre-calculated numbers. There is no reason to do unnecessary CPU-intensive calculations when they have relatively the same output every time, for instance, Sin and Cos lookup tables.

2) What are some suggestions to make my 3D graphics a little faster?

a) First and foremost -- keep polygon count low. With the tools available this is sometimes difficult, since it seems modelers like to triangulate when saving out.

b) Make several versions of objects at different "resolutions", and sort higher polygon objects when those objects are closer to the camera, and lower polygon objects when they are farther away.

CODEWARRIOR
---

1) What can I do with Codewarrior that I can't do with Siocons?

a) Automatic makefiles - but other free Yaroze utils allow this
b) GUI coding - not suggested for PC, pretty choppy interface
c) GUI debug - very useful
d) FileServer - use MWDebugIO functions, which allow you to create, open, and close files on your PC; can be very unreliable; ARS and NiceARS (uses Pro Action Replay) is an alternative
e) Dead code stripping & call tree analysis.
f) Optimization that works
g) Understandable inline assembler syntax
h) Generate Playstation EXE format (for those who want that)

Here are some comments given by Craig Graham (creator of ARS):

"The yaroze version editor is buggy, yes. They sorted the IDE out on the Pro version about 6 months ago, no more dodgy characters and stuff. Perhaps if enough people mail them they'll do an IDE update for the yaroze compiler that provides the same fixes.

The debugger is a big step beyond the GNU one. The rest of the system is actually rather good. The compiler itself is very fast compared to GNU, and the code browsing stuff is excellent (better in many respects than the Microsoft IDE)."

And by Philip Gooch:

"
1) Compiling programs with multiple targets is a doddle - if you want to compile multiple versions of your code, say your trying out different things and you want to test the effects, this is easy. You can add all the different versions of your code to a project and quickly choose which ones to compile.

2) Similarly, it's easy to compile Debug and Optimised (with 4 levels of optimisation) versions of your code

3) Markers: if you want to quickly go to a particular point in your code, you can add a marker. Great for long and untidy source code!

4) Pop-up menus so you can quickly jump to all your functions.

5) Pop up menus so you can quickly open up your header files (great if you've forgotten what you've called or defined something as

6) Clicking and holding on a variable will let you jump to its definition

7) Colour syntax highlighting

8) Being able to dynamically download data from the host computer - great if you want to write games with lots of levels and graphics (downside is you need PsComUtil to run them)
"

Some other comments:

"I like the fact that I can compile and run my code with a single button press. I like the way the compiler points to erronous code in an editor window so it can be corrected there and then. But the main reason I stick with CW is the dynamic loading - load a TIM, copy to VRAM, load next TIM at same location etc etc. Saves a hell of a lot of memory. Now I have it, I can't live without it!"

2) What can I do with Siocons that I can't do with Codewarrior?

a) Download memory directly to a PC file (DSAVE[])

3) What's the best compiler setup?

One way to develop for Yaroze is to use Codewarrior to "make" your projects and the PSComUtil to download batch files, since it's mostly point and click. Many people have found the Codewarrior IDE very disturbing, though, so they choose to use an outside editor such as Microsoft DevStudio, or some shareware code editor. You can leave the files that you edit in the Codewarrior project, and just bring Codewarrior into focus and click "make" when you want to build your project to be downloaded to the yaroze.

Some other people have suggested forgetting Codewarrior altogether, using the makefile capabilities of MS DevStudio and integrating DJGPP(GCC) with it. See Steve Dunn's (SCEE) home page.

Also, check out the front end gnu programs that are available (in the utilities listing of this FAQ).

4) What kinds of problems are there with Metrowerk's MWDebugIO library?

The library seems to arbitrarily cause exceptions (crash your program) for unknown reasons (Metrowerks has been very quiet, and have not responded yet). You may want to check out ARS and NiceARS, which are libraries that allow quick I/O operations with the use of Pro Action Replay (European).

5) Why is PSComUtil failing?

a) If you get
"Connection Failed. Status -1
Transport Send Data. Status 101812
Transport Poll Rx Status. Status 101812
The PlayStation has generated a Hardware Interrupt exception at instruction address 0x0."

Probably means that Pscomutil is still resident in memory. Check your task manager and delete the process. For some reason, even if you exit PSComUtil, it still stays resident in memory sometimes.

YAROZE COMMUNITY
---

1) What Yaroze tutorials are available?

Ira Rainey's Sprite Tutorial (SCEE)
http://www.netyaroze-europe.com/~shadow/ftp/STUFF/Howto.pdf

James Chow's help docs (SCEE)
http://www.netyaroze-europe.com/~jaycee/

Peter Passmore's 3D Tutorial (SCEE)
http://www.netyaroze-europe.com/~middx_uni/ftp/tutorial.zip

Robert Swan's accompaniment to above tutorial (SCEE) - see other downloads too
http://www.netyaroze-europe.com/~middex2/ftp/comp_tut.zip

Jamin Frederick's Sprite Tutorial (SCEA)
http://www.scea.sony.com/net/yaroze/pages/ftp/jfrederick/onespr.zip

Nelson Santos' Ping - beginner game with docs (SCEA)
http://www.total.net/~nsantos/downloads/ping.zip

James Russell's Vectors and Matrices Tutorial (SCEE)
http://www.netyaroze-europe.com/~jruss1/matrix.html

Javier's Vector and Matrix Math (SCEE)
http://www.netyaroze-europe.com/~javier/

2) What are some useful general-purpose yaroze utilities?

**SCEE member sites:

Memory Viewer - graphically view Yaroze RAM, 0x80000000 to 0x801fffff
http://www.netyaroze-europe.com/~shaughnj/ftp/memview.zip

Analog PAD diagnostics - displays analog PAD values
http://www.netyaroze-europe.com/~shaughnj/ftp/analog.zip

DOS yaroze tools for people who don't like windows
http://www.netyaroze-europe.com/~yannick/

ARS (Action Replay File Server) - lets you do i/o with yaroze at 20x the speed; NOTE: you need Datel Action Replay for this
http://www.netyaroze-europe.com/~c_graham/areplay.html

NiceARS - ARS implemented in windows NT; NOTE: you need Datel Action Replay for this
http://www.netyaroze-europe.com/~steved/nice_ars_for_windows_nt.htm

RsdAnim - "...a PC (Win95) hosted keyframe animator for Playstation RSD format 3D models...."
http://www.netyaroze-europe.com/~c_graham/rsdanim.html

Binary Conversion Tool - converts data files to C source binaries
http://www.netyaroze-europe.com/~steved/tools/binary_conversion_tool.htm

Sound Effects Player - lets you test sounds on yaroze
http://www.netyaroze-europe.com/~steved/articles/sfx_player/
sound_effects_player.htm

C++ Library - wrappers around standard pslib function calls
http://www.netyaroze-europe.com/~steved/cpp_yaroze_library.htm

Yaroze Master - makes editing and using makefiles a snap on Win95/NT
http://www.netyaroze-europe.com/~deruiter/ym.zip

Lightwave to RSD converter - convert from LOB to RSD
http://www.netyaroze-europe.com/~Sig1LL/ftp/converters/pcpsx120.zip

Starfield Library - function calls to easily make a 3d starfield
http://www.netyaroze-europe.com/~rcutting/ftp/starfield/starfield.ZIP

GCC C++ - Win32 compiler for your C++ code, even if you don't have Codewarrior
http://www.netyaroze-europe.com/~frothy/ftp/gcc281.ZIP

Crossroads - freeware 3D file converter (Win32)
http://www.pnn.com/~rickhowd/xroads1.zip

Convert - freeware DOS audio file converter
http://www.netyaroze-europe.com/~nslaven/ftp/utils/convrt14.zip

PAK - compression utility for data uploads (incomplete?)
http://www.netyaroze-europe.com/~smithers/ftp/pak.zip

Linux Tools
http://www.netyaroze-europe.com/~respond/download.html

Amiga Tools
http://www.netyaroze-europe.com/~CACTUS/

Unix Tools
http://www.netyaroze-europe.com/~byz00002/unix/

Utilities for Atari ST sprites
http://www.netyaroze-europe.com/~sevans/demos.htm

TMD Viewer - includes depth cuing and object/viewpoint manipulation
http://www.netyaroze-europe.com/~sevans/ftp/demos/yaroze/tmdview.ZIP

CRNCHPLY - reduces redundant PLY data
CRNCHPL - same as above, but without DOS4GW.EXE
MEMEDIT - edit the contents of any file on your second memory card
DATAMAN - manages data download offsets
http://www.netyaroze-europe.com/~jruss1/#tools

Memory Card Dump
http://www.netyaroze-europe.com/~badchild/ftp/card.zip

Stereoscopic Vision
http://www.netyaroze-europe.com/~middx_uni/ftp/stereo.zip

Dump Address - creates .h and batch files for your Yaroze datafiles
Palette Ripper - extracts Cluts from PCX files. Useful for light and palette animation
Vertex Extractor - Extracts vertex and normal data from TMD files. Useful for animated 3D objects
http://www.netyaroze-europe.com/~IBEYOND/ftp/niftytools.zip

General GsBG Background Editor
http://www.netyaroze-europe.com/~shaughnj/ftp/bgedit.zip

**SCEA member sites:

VRAM and TIM viewer
http://ww1.scea.sony.com/net/yaroze/pages/ftp/kbender/vvram.zip

PSXsock - enables TCP/IP connection of Yaroze (Win95/NT)
http://www.scea.sony.com/net/yaroze/pages/ftp/jblack/PSXsock.zip

Unix Tools
http://www.scea.sony.com/net/yaroze/pages/chenrich.html

Psx IDE - front end to gcc for Win95/NT
http://yaroze-world.org/assets/images/psxide.zip

WAD Builder - compression utility for Yaroze data/code
http://yaroze-world.org/assets/images/wb.zip

Joystick Routines
Sprite Animation
Decompression
Starfield
(libraries and example code)
http://yaroze-world.org/html/ps_code.html

Card save module - functions to save to memory card
Font module - lets you use pretty, custom fonts
Menu module - lets you easily make a menu system
http://lucien.blight.com/~sauron/Net.Yaroze._Rocks_.My.World/

Address Arranger - automatically arranges your data addresses for downloading
Sprite Assembler - clips a sprite from a TIM and resaves it
TIM Manipulator - fiddle with specific TIM attributes
Screen Grabber - grabs the screen and stores as a file
http://www.scea.sony.com/net/yaroze/pages/scartier.html

PPTMDView - TMD/RSD viewer for Mac
http://www.scea.sony.com/net/yaroze/pages/ftp/wlee/PPTMDView_0.21.sit.hqx

Font Library - fonts and special effects
http://comradery.com/erico/cool/yaroze/fontdemo/fontdemo.zip

GsBG Library - lets you do tiled backgrounds
http://www.scea.sony.com/net/yaroze/pages/ftp/eolaughlen/bgdemo.zip

Card Save Library - lets you save to memory card
http://www.netyaroze-europe.com/~jaycee/

DMS - S3M module player for Yaroze
http://www.scea.sony.com/net/yaroze/pages/elee2.html

3) Where is there some useful technical info?

**SCEE member sites:

Scott Evans' Technical Notes
http://www.netyaroze-europe.com/~sevans/tech.htm)

Some intro demos and tidbits: dynamic TMDs, lines, sprites, gradients,
split screen (Robert Swan, Downloads section)
http://www.netyaroze-europe.com/~middex2/

Various technical info
http://www.netyaroze-europe.com/~mrfrosty/yze_resource/rce_index.html

Using DevStudio with GCC
http://www.netyaroze-europe.com/~steved/using_gcc_makefiles_in_ds.htm

General matrix and MIPS info, graphics links
http://www.netyaroze-europe.com/~javier/index.html

Shiny Toruses
http://www.netyaroze-europe.com/~Sig1LL/projects/second/howto.htm

3D Studio file format
http://www.netyaroze-europe.com/~Sig1LL/3ds/3ds.html

Motion Capture
http://www.netyaroze-europe.com/~Sig1LL/motion.html

TMD file format
http://www.netyaroze-europe.com/~Sig1LL/tmd/tmdform.html

Exception Handling
http://www.netyaroze-europe.com/~Sig1LL/ftp/source/EXCEP.ZIP

Macintosh help
http://www.netyaroze-europe.com/~daryl/

C++ for Yaroze
http://www.netyaroze-europe.com/~c_graham/cplus.html

**SCEA member sites:

NM.EXE
ClearImage()
Multiple CLUTs
Misc Tricks
Fixed Point
http://www.scea.sony.com/net/yaroze/pages/scartier.html

Using the Yaroze with C++

http://www.scea.sony.com/net/yaroze/pages/cblackwell2.html

C++ class examples
http://www.scea.sony.com/net/yaroze/pages/cblackwell4.html

Some programming tips
http://www.scea.sony.com/net/yaroze/pages/cblackwell3.html

Some answered FAQs
http://www.scea.sony.com/net/yaroze/pages/hchen.html

VSync() Diagram
http://www.scea.sony.com/net/yaroze/pages/wlee5.html

Tidbit on speeding up DOS SIOCONS programs
http://www.scea.sony.com/net/yaroze/pages/dminsterman.html

Macintosh resources
http://www.scea.sony.com/net/yaroze/pages/nporcino.html

Some info on SSUtKeyOn()
http://www.total.net/~nsantos/downloads/sfx.zip

Docs describing Yaroze API
http://www.total.net/~nsantos/downloads/yardoc.zip

4) Where are some other interesting sites?

Yaroze Demos Page (Nick Ferguson)
http://www.netyaroze-europe.com/~rookie1/demos/demos.htm
http://www.saqnet.co.uk/users/nickf/public_html/demos/demos.htm

Personal profiles of participating yaroze members
http://www.netyaroze-europe.com/~madmac/

Yaroze Game Reviews
http://www.netyaroze-europe.com/~mrfrosty/yze_resource/rce_index.html

Net Yaroze Times
http://www.geocities.com/TimesSquare/Alley/2200/

Club Yaroze
http://www.clubyarouze.com/cyarchive/

Underground Yaroze (no longer updated)
http://www.pacificcoast.net/~titan/

5) Are there any chat sessions going on?

Yaroze Chat in Auditorium on SCEA web page (every Saturday 9:00pm EST)

http://www.scea.sony.com/net/yaroze/aud.html

James Rutherford's chat page (still pending)
http://www.netyaroze-europe.com/~mrfrosty/chat/index.html

ICQ Chat Group
http://groups.icq.com/group.asp?no=17896

6) What yaroze contests are there? Who can participate?

UK Game Developer ’98 -- SCEE yaroze only
(http://www.gduk.co.uk/html/first.html)

Jeff Lawton's Yaroze Competition
http://yaroze-world.org

MISCELLANEOUS
---

1) How do I convert from PAL to NTSC or NTSC to PAL?

See:
http://www.netyaroze-europe.com/~jaycee/palntsc.html

2) How can I make my program detect NTSC or PAL an runtime?

According to George Bain:

```
if( CdSearchFile(&file, "\\DTL_S30.35;1") == 0)
  {
  printf("Boot file not found... MODE_NTSC enabled \n");
  SetVideoMode( MODE_NTSC );
  video_mode = GetVideoMode();
  }
else
  {
  printf("Boot file found... MODE_PAL enabled \n");
  SetVideoMode( MODE_PAL );
  video_mode = GetVideoMode();
  }
```

3) How do I takes screenshots?

There are two methods, described by the following newsgroup post:

"There are two main methods of grabbing screenshots which can be got from
the scea mirror under scee utilities (i think) Unfortunately, I think both
could do with being better programmed, so maybe theres a short thing i can

do tomight.

The first uses the files scrngrab.c/h and it basically printfs() the contents of a rectangle of video ram to the pc, which is picked up by logging the siocons output. A program is then run (log2raw) which strips the other crap out and leaves you with a .raw file which something like paint shop pro can convert to bmp. Plus points: fairly easy to set up, and easy to grab more than one image from the game without needing to reload it. Bad points: can be corrupted relatively easy during pc transfer. (A post by James Russell says there shouldn't be any interference using printfs but that isnt true. works fine at home, but maybe having 30 other pcs around the ones at uni explains why they cock up everysingle time).

second is screensht.c/h and uses storeimage to copy video ram contents to main memory, then dsave to transfer to PC. Use timtool to convert from tim format to bmp. Good points, hasnt corrupted on me yet. Bad points: harder to include, only get one screenshot (without extra work to it) have to quit program to grab pictures)

The first method takes around 20 seconds to load to pc, and the second takes 29 seconds at full speed..."

UNANSWERED

) Why does saving to DXF from modelers make so many more polygons? What can I do to avoid this?

) Is it more important that I make fewer objects or fewer polygons?

) Is there any reason to put several models within a TMD file instead of only one?

) How do the 24-bit images work?

) Can I do floating point?

) How are texture pages numbered, and what is their ID?

) Why doesn't LoadImage() work for large rectangles, close to the size of the video memory (1024 x 512)?

) Why is PSComUtil failing?

) Is there any differences in writing code for Codewarrior versus writing code for Siocons?

) What can I do with Siocons that I can't do with Codewarrior?

) Is there any way to read from a CD other than the Yaroze boot disk, and is it allowed?

) Can I run yaroze programs on a regular playstation? Do I still need to download?

) What are some cool 2D special effects?

) What are some cool 3D special effects?

) What is ARG and Pro Action Replay?

) How much is Pro Action Replay, and where do I get it?

) Is Pro Action Replay worth the money?

) Can I use the Game Shark instead of Pro Action Replay?

) What are some suggestions to make my 2D graphics a little faster?

) What are some suggestions to make my 3D graphics a little faster?

) Do all the translucency settings work right?

) Why would you want to "turn off" a primitive/object for display instead of just not sorting it?

) Are there any *simple* modelers out there, that don't triangulate like crazy?

) How about some simple Codewarrior (MWDebugIO) example code for creating, opening, reading, and writing files to the PC?

) How efficient is C++ on the playstation?

) How efficient is using 16x16 pixel tiles (with GsBG) as opposed to using sprites (with GsSPRITE) as backgrounds?

) How can I make my games quicker, taking into account playstation hardware?

) How do I rotate my model around a different pivot point, while keeping it centered (at (0,0,0)) on its local coordinate system?

) What's the difference between GsVIEW2 and GsRVIEW2 and how do they work?

) I want to have an object with several moving parts (a model for each part) and have each part move independently. What do I have to do to get this to work?