User Guide

Software Development Tool

Table of Contents

About Net Yaroze

What You Need to Know

In order to get started with Net Yaroze, you should have experience ofC programming to a competent level and a knowledge of a 2D graphic creation/editing tool. In addition, at least a basic grasp of a 3D modelling packageand a sound creation/editing toolwould be help you get the best out of you Net Yaroze kit.

The Net Yaroze Manual Set

There are three books in the set of Net Yaroze manuals.

1. Start Up Guide

   An introductory booklet explaining the contents and requirements of the Net Yaroze Starter Kit. It also gives step by step instructions on setting up they Net Yaroze software on your PC and how to run Net Yaroze software on the system.

2. User Guide (this document)

   A reference manual providing details on making software for the Net Yaroze system.

3. Library Reference

   A manual listing and describing the functions and structures in the Net Yaroze libraries.

Additional Reading

Please see the Additional Reading list at the end of theStart Up Guide

1

System Overview

This chapter contains an overview of the Net Yaroze system, and an explanation and overview of PlayStation hardware.

Net Yaroze is a revolutionary project which enables anyone to create PlayStation applications by using a range of Net Yaroze development tools on a personal computer connected to a special Net Yaroze PlayStation. Net Yaroze applications can be posted on an exclusive Net Yaroze web site to be shared and enjoyed by other Members.

Net Yaroze System

The Net Yaroze system is designed to let you write, debug and test PlayStation applications on a personal computer (PC) linked to a special Net Yaroze PlayStation. The PC, which acts as a host machine, is linked via a dedicated serial cable to the PlayStation which runs the applications.

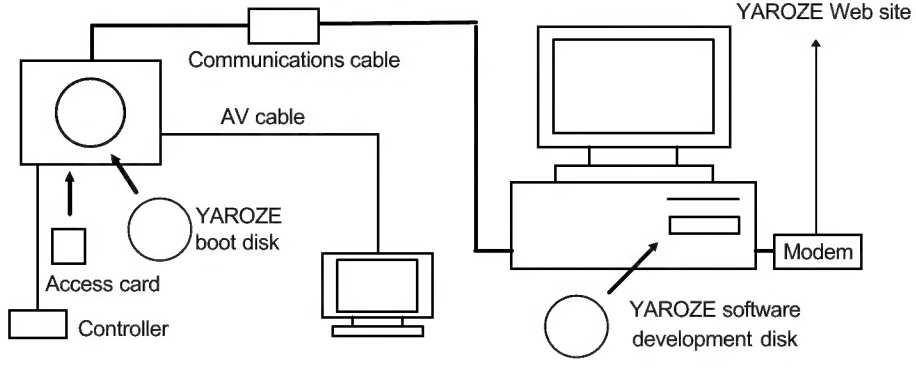


Figure: Net Yaroze System Set Up

First of all, you create a program using a variety of programming tools. Once you've written, compiled and linked it, you need to download it into the PlayStation using the console tool, SIOCOMS (discussed in Chapter 17), and then test it on the PlayStation.

Using the same procedure you can also download and verify any data which are needed by the application - such as audio files. (You need a Controller connected to the PlayStation to operate your program.)

IF your Net Yaroze host computer also has an Internet connection, not only can you post your Net Yaroze programs to the Web site very easily but also download and immediately execute those posted by other Members.

The Net Yaroze Web site

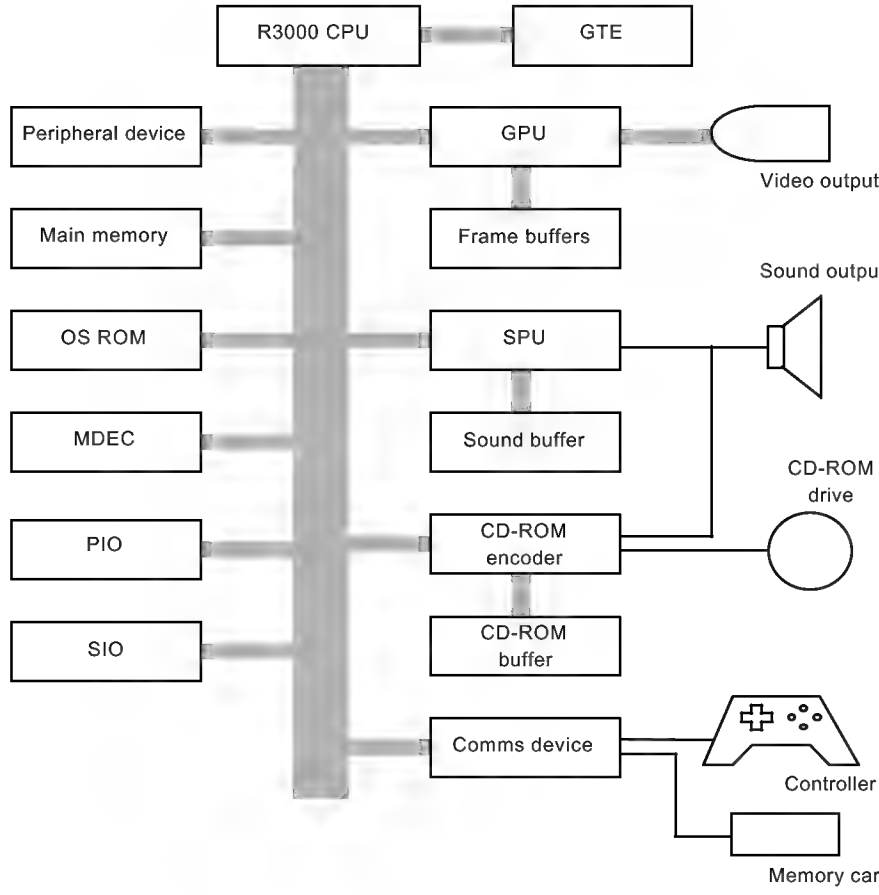
The Net Yaroze Web site is hosted by a server run by Sony and accessible via the Internet. The Net Yaroze Web site has features such as a mail forum, Member's home pages and access to documentation and program libraries. It enables the exchange of information between Members as well as the uploading and downloading of Net Yaroze applications. The site also provides the latest Net Yaroze related information, and support.

You'll need an Internet connection and a World Wide Web browser to access the Web site. For more details refer to the Start Up Guide

PlayStation Architecture

As is shown in the following diagram, the PlayStation system is centred on a 32-bit RISC CPU, and comprises a number of processors and devices dedicated to certain functions such as graphics and sound.

Figure : PlayStation Block



Glossary

GTE : Geometry Transfer Engine

GPU : Graphics Processing Unit

SPU : Sound Processing Unit

MDEC : Data Decompression Engine

PIO : Parallel Expansion port

SIO : Serial Expansion Port

The CPU and Its Peripherals

The PlayStation uses a custom CPU based on the R3000 (33 MHz) 32-bit RISC CPU (little endian).

The I Cache

The CPU reads instruction code in the logical memory space of the I cache at approximately 5 times the speed of main RAM. Instruction code that has been read once is stored in the I cache within the CPU, and can be re-executed without accessing the main memory. The I cache cannot be operated on from within programs.

The CPU is equipped with a 4K I cache. The logical memory space is divided into 4K units, these multiply-mapped onto the I cache.

The D Cache

The D cache employs a special structure called a scratch pad, and is mapped onto a 1K of logical memory space (0x1f800000~0x1f8003ff) which the program developer can freely access.

The General-Purpose Registers

There are thirty-two 32-bit general-purpose registers. The compiler assigns each to the specific uses shown in the table below. You must use registers in accordance with these assignments in thread database operations and development using the assembler.

Register No.	Macro (1)	Macro (2)	Assembler Assignments	
0	R_ZERO	R_R0	zero	0 fixed
1	R_AT	R_R1	AT	reserved for the assembler
2~3	R_V0~1	R_R2~3	v0~1	values returned by functions
4~7	R_A0~3	R_R4~7	a0~3	function arguments
8~15	R_T0~7	R_R8~15	t0~7	destroyed within functions
16~23	R_S0~7	R_R16~23	s0~7	saved within functions
24~25	R_T8~9	R_R24~25	t8~9	destroyed within functions
26~27	R_K0~1	R_R26~27	k0~1	reserved for the kernel
28	R_GP	R_R28	gp	global pointer
29	R_SP	R_R29	sp	stack pointer
30	R_FP	R_R30	fp	frame pointer
31	R_RA	R_R31	ra	return address

Table: R3000 General-Purpose Registers

The Return Address

The R3000 instruction set does not directly include the functionality of a subroutine. A subroutine call is replaced by a jump command which stores the return address in a register. The register that contains the return address can be assigned by the assembler, but in the case of the C compiler it is limited to general-purpose register No. 31.

The Stack

R3000 chip does not include a stack. As a result the compiler creates a stack by storing a pointer in general-purpose register No. 29. In addition, in order to utilise function frames (memory areas used for automatic variables and as working areas) efficiently, general-purpose register No. 30 stores a start address for the function working area (frame), referred to as the frame pointer. The value of this frame pointer is determined by the value of the stack pointer. The values of these frame and stack pointers are harmonised at module activation.

The Global Pointer

There is a register indirect mode used to access R3000 memory, the mode using symbols and a 16-bit offset. In order to work efficiently the compiler collects together up to 64K of variables in a block labelled 'bss session'. General-purpose register No. 28 stores the central address of the bss session and, using the mode specified above, the data stored in the bss session block is accessed with a single command. This address value is referred to as the global pointer. It does not change within a module.

The Main Memory

The PlayStation is equipped with 2 MB of main memory. Addresses are allocated in this memory starting from 0x0000 0000. This is called the 'physical memory space'.

The CPU memory space has 32-bit addresses and is called the 'logical memory space'. The physical memory space is mapped to 3 locations in the logical memory space. The CPU is not equipped with a virtual memory manager so the relationship which maps the two memory spaces described above is fixed.

Physical Memory	Logical Memory	Segment Name	I Cache
0x00000000~0x001fffff	0x00000000~x001fffff	ku	Available
	0x80000000~0x801fffff	k0	Not available
	0xa0000000~0xa01fffff	k1	Available

Table: Physical Memory and Logical Memory

The OS ROM

The PlayStation has 512K of ROM. The OS kernel and the boot loader are stored in this ROM to which access not permitted.

The DMA Controller

A DMA Controller is attached to the CPU. This carries out transmission of data between memory and devices in accordance with instructions from the CPU.

Graphics System

The PlayStation is equipped with a graphics processing unit (GPU). The GPU features CRTC functions for display on a screen, and high speed polygon drawing functions which work on the frame buffer.

The Frame Buffer

The GPU has a 1MB frame buffer. This frame buffer is composed of a two-dimensional address space (1024 x 512) made up of 16-bit pixel units. This memory space is managed by the GPU, and cannot be directly accessed from the CPU.

The Display

The GPU displays the contents of any rectangular area within the frame buffer, without modification, on a CRT display. This area is called the 'display area'. The following 10 types of screen mode are supported.

NTSC		PAL	
Interlaced	Non-interlaced	Interlaced	Non-interlaced
256(H) x 480(V)	256(H) x 240(V)	256(H) x 512(V)	256(H) x 256(V)
320 x 480	320 x 240	320 x 512	320 x 256
512 x 480	512 x 240	512 x 512	512 x 256
640 x 480	640 x 240	640 x 512	640 x 256
384 x 480	384 x 240	384 x 512	384 x 256

Table: Screen Modes (NTSC and PAL)

The GPU supports 2 modes with respect to the number of colours that can be displayed: a 15-bit direct mode (32,768 colours) and a 24-bit direct mode (full colour).

In the 15-bit direct mode 32,768 colours can be displayed simultaneously. This means that, in comparison to the 24-bit direct mode, the number of colours that can be displayed is limited; however, colour calculation within the GPU during drawing uses 24 bits and dither function to give a pseudo full colour display.

In the 24-bit direct mode 16,777,216 colours can be displayed simultaneously. However, only image data that has been transmitted to within the frame buffer can be displayed (still picture display), and execution of GPU drawing functions is not possible. The bit length is 24 bits per pixel, but you need to specify the frame buffer co-ordinates of the display location on the frame buffer on the basis of 16 bits per pixel. In other words, make 640 x 240 24-bit direct mode image data have the size 960 x 480 within the frame buffer.

Drawing Capabilities

The drawing functions supported by the GPU are as follows.

Name	Details
Polygon Drawing	4bit CLUT(16 colours/polygon) 8bit CLUT(256 colours/polygon) 16bit (32768 colours/polygon) Flat shading, Gouraud shading Texture mapping
Straight Line Drawing	Gradation is possible
Image Transmission	CPU ↔ frame buffer Frame buffer→ frame buffer
Others	Blending (semi-transparency), dithering, clipping

Table: GPU Drawing Functions

Polygon Drawing

The GPU is equipped with polygon drawing functions. The polygons it deals with are 3-sided and 4-sided figures. These are drawn by specifying the screen co-ordinates for each of the vertices of the figure. In addition to the shape, the GPU can specify their colour and/or texture, for example: 'texture mapping' (in which an image from any area of the frame buffer is pasted onto the polygon surface), 'flat shading' (solid colour paint out) and 'Gouraud shading' (gradation paint out, in which each pixel in a polygon is given a colour calculated from the colours assigned to the vertices).

Images and texture patterns pasted onto polygons are transmitted to the frame buffer before drawing the polygon. Data is stored in the frame buffer in 256 x 256 pixel pages, in which there can be as many as the available memory will allow. These 256 x 256 areas are called 'texture pages'. The size of a texture page in the frame buffer varies depending on the mode.

CLUT Processing

There are 3 colour modes for texture patterns, a 4-bit CLUT mode, an 8-bit CLUT mode, and a 15-bit direct mode.

In the 4-bit and 8-bit CLUT modes use a colour look-up table (CLUT). Colour look-up tables are lists on the frame buffer comprising 16 or 256 RGB values that represent the colours that will ultimately be displayed. Each RGB value is given a number on the frame buffer in order from left to right, and these numbers define the colour of each pixel in a sprite pattern. YOu can select CLUTs at the sprite level, which means each sprite can have its own independent CLUT.

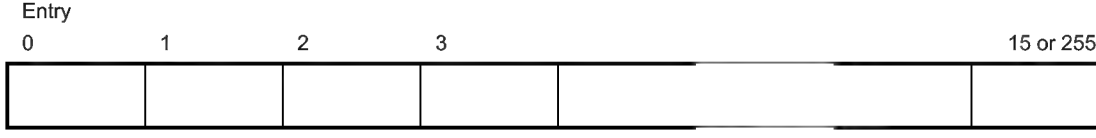


Figure: CLUT Structure

Each entry has the same structure as a single pixel under 15-bit direct mode. Therefore, one CLUT set is equivalent to 1 x 16 or 1 x 256 pixels of image data.

Sound System

The PlayStation sound system is made up of the Sound Processing Unit (SPU) and the CD-ROM decoder. Audio output from the CD-ROM decoder enters the SPU, is mixed with output from the SPU, and is transformed into the final audio output.

The CD-ROM Decoder

Data is read from the compact disc (CD), and then either reproduced directly as audio output or transmitted to the main memory.

ADPCM data provided by the CD-ROM XA, and CD-DA 16-bit PCM data is directly output as sound. Data read from the drive to the CD-ROM buffer is processed by the sound source within the CD-ROM decoder, and the resulting audio signal sent to the Sound Processing Unit via the mixer built into the decoder.

The SPU

The Sound Processing Unit (referred to as the SPU) has 24 voices and controls 512 K of memory, known as the sound buffer. Compressed waveform data is stored in the sound buffer, and this data is regenerated by the SPU at a sampling frequency of 44.1 kHz, in accordance with sound production commands from the CPU. The sound buffer is composed of a one dimensional memory space made up of 2 byte units which cannot be directly accessed from the CPU.

The sound buffer is also a working area for the reverb function (which operates as an effector) and a temporary buffer during transmission of sound data to the main memory. That is sound data that has been input from the compact disk or created by the SPU. The sound buffer enables data to be transmitted to the main memory without interrupting sound production.

There are facilities to alter the output of each voice: pitch change operations, envelope operations (attack, decay, sustain, release), and volume operations, for example.

Other Systems

In addition to those already described, the PlayStation is equipped with the following systems.

The GTE

The GTE is a co-processor that carries out the vector and matrix arithmetic operations essential for 3D graphics. The GTE supports fixed-point arithmetic, the results of its calculations can be incorporated, without modification, into drawing commands sent to the GPU.

The Data Decompression Engine

The data decompression engine carries out reverse DCT conversion calculation at high speed, and decompression of JPEG data and MPEG data (only for data compression within the frame). The data decompression engine is not part of the Net Yaroze system.

The Controller

The Controller is an interface that transmits the player's intentions to the application. In professional PlayStation development, in addition to the two connectors provided on the main body of the PlayStation, a larger number of Controllers can be connected using Multi-tap ports. Multi-tap ports cannot be used with the Net Yaroze system.

The Memory Card

The Memory card provides storage for PlayStation application data when the PlayStation is switched off. In professional PlayStation development, in addition to the two connectors provided on the main body of the PlayStation, a larger number of cards can be connected using Multi-tap ports and Multi-taps. Note, however, that Multi-tap ports cannot be used with the Net Yaroze system.

Expansion Ports

Two expansion ports are provided: one serial, the other parallel.

In a professional PlayStation development, the serial ports enable communication between two PlayStations via a Link cable. In the Net Yaroze system this serial port connects the Net Yaroze PlayStation to the your computer. Thus the PlayStation link facility is not available in the Net Yaroze system.

The parallel port is reserved for future expansion and cannot be used.

2

The PlayStation Development Environment

The PlayStation is equipped with the PlayStation operating system (OS). This OS was developed specifically for the PlayStation's R3000 CPU, and incorporates extremely novel concepts of which both the Professional Development System and the Net Yaroze system take advantage. This chapter provides an overview of the Professional PlayStation Development System and describes the strengths of the Net Yaroze system.

The Professional PlayStation Development System

The PlayStation OS was developed specifically for the R3000 chip used as the PlayStation's CPU, and features extremely novel concepts. The efficiency of program development depends to a large extent on the environment and services provided by the OS of the machine used. If a CPU and peripheral devices of can be guaranteed to be of a sufficiently high performance, the services provided by the OS can be utilised effectively. Thus application development can proceed rapidly and smoothly as there is no need to spend time considering how to stretch the hardware performance to its limits and the developer can concentrate on actual programming.

The design concept of the PlayStation OS was to provide the game program developer with an environment in which interrupt driven programs can be easily operated. Based on this concept, the kernel of the PlayStation OS is provided as a collection of R3000 and PlayStation hardware management services (subroutines).

Access to these services is provided via a set of library functions written in the programming language C. Not only does the use of C mean that source code can be more readily understood and maintained, but also that features such as the simplicity of block structure description and function calls can be used to good effect, and thus programming can be carried out very easily.

The Professional PlayStation Development System is based on the concepts described above. It is composed of a small-scale OS kernel provided with convenient interrupt processing and multitasking support functions; a hardware management library; and middleware consisting of a group of high level services such as a MIDI driver and a 3D graphics system. The minimum memory size required by the OS has been limited to 64K, and the OS has been designed so that the developer can have the maximum freedom possible under a CD-ROM system. SCE's R&D teams continue to expand the professional PlayStation library and middleware based on requests from professional application developers.

In return for providing a high degree of freedom, however, this environment requires a high level of design and development skills on the part of the developer. In practice, at the time of writing, there are more than 1600 functions in the C language function group of which the PlayStation libraries are composed. Within this group a number of options have been provided to carry out a single task (and moreover, each of these options has various pros and cons). Professional developers must select from within this function group the methods that are most suitable for the programs they are designing.

The Professional PlayStation Development System can be described as a collection of parts designed for use by specialists, selected more in the pursuit of diversity and freedom than of ease of arrangement, co-ordination or comprehension.

The Net Yaroze PlayStation Development System

The Net Yaroze PlayStation Development System has been designed with the aim of providing a simple and easily understood development environment by using a more easily understood subset the Professional PlayStation Development System. The Net Yaroze PlayStation Development System has the following strengths.

Programming in C

All of the services are provided as C functions, so programming can be carried out consistently using C.

Easy Utilisation of the Functions of the R3000 Chip

Carrying out interrupt processing procedures on the R3000 chip can be said to be complicated, but with the PlayStation OS these procedures are processed by the OS kernel on behalf of the CPU. In addition, a 'callback' system is provided as a means of informing the user of interrupts.

The Focus is on Vertical Synchronisation Interrupts

As it has been designed as a dedicated video game machine, its vertical synchronisation interrupt is the key for getting the most out of the PlayStation. A call back system comprising of a simple interface which enables direct coding of interrupt processing programs in C is provided as the key feature of PlayStation programs, in a form that is focused on vertical synchronisation interrupts.

The Production of Compact Programs

The Net Yaroze system connects the PlayStation and the development host using a serial interface, which has a high degree of convenience and connectibility. To compensate for slow data transmission speed, a weak point of serial interfaces, system programs such as graphics, sound, CD-ROM, and debugging monitor programs are all loaded into the main memory from the boot disk as a permanent library. Then the linker writes in the addresses to enable access to this permanent library from within application programs. As a result, application program size, and therefore downloading time, is kept to a minimum.

Parallel Processing Type OS

The PlayStation OS carries out processing by dedicated hardware in parallel with program execution by the CPU. In the Net Yaroze system, parallel processing functions are provided by placing polling at the centre of programming style.

3

Application Development Procedures

This chapter presents an overview of the method used to create PlayStation applications using the Net Yaroze system.

Data needed by applications, such as sound and graphics files, can be easily converted by the Net Yaroze system from standard file formats (.bmp files for graphics, for example) to PlayStation file formats.Also, as the development steps are based on standard C language development steps, no special sequence is necessary. A GNU C compiler and associated utilities are provided as programming tools.

For detailed descriptions of each of the items, please refer to the chapter describing the hardware, or the chapter describing the corresponding services.

The Flow of Program Creation

The Net Yaroze development steps are based on the standard steps used in C language development, so people who have had experience in development using C will find it familiar.

Source Code Creation

Source code should be made as a standard MS-DOS text file using any commercial editor designed for writing programs.

Compiling and Linking

Once you have written the source code, it must be compiled and linked to make an executable program. The Net Yaroze system provides a special library, a GNU C compiler, and various tools associated with the compiler.

Test Runs

The executable program can be tested on the Net Yaroze PlayStation. Using the console tool, SIOCONS described in Chapter 17, you download the executable program from your PC to the Net Yaroze PlayStation

Debugging

When your program fails to operate as it should, you can debug it either at source code level, or using the GNU debugger (gdb), provided with your Net Yaroze kit, which allows you to step through the program and see what's happening as it executes.

Using Makefile

The Makefile function is a convenient way to simplify the series of operations related to compiling and linking. The 'make' command, provided in the Net Yaroze system, allows high level maintenance of applications.

Making a Library of Useful Routines

You can increase development efficiency by making a library of frequently called routines and sub-routines that are jointly used by a variety of programs. In the Net Yaroze system, the following GNU utilities assist this process: 'ar' - makes libraries from object files, and 'nm' - provides details (such as the start address) of object symbols. (See Chapter 16 , the documentation with the GNU compiler on your Net Yaroze PC disk and commercially available documentation for details of GNU utilities.)

The Flow of Data Creation

Data needed within a PlayStation application can be broadly divided into three categories: 2D graphic, 3D graphic and sound. (Note that a fourth type, movie data, while available in the Professional PlayStation Development System cannot be used in the Net Yaroze system.)

There are special PlayStation data formats for each of these. The Net Yaroze system provides converters to change files from the standard formats in which they were created to PlayStation formats.

2D Graphic Data

2D graphic data used by the PlayStation consists of image data, which is used as sprite pattern data, and texture data. PlayStation format 2D graphics data is referred to as TIM data.

In the Net Yaroze system, there is a special converter to convert data created using any Windows or Macintosh painting tool to TIM data.

Data type:	TIM - image data (sprite pattern and texture)
Tools provided:	Converters to convert from existing formats such as BMP, PICT, (among others) to TIM format
Operating environment:	Windows, MS-DOS

3D Graphic Data

3D graphic data used by the PlayStation consists of modelling data used by 3D applications. PlayStation format 3D graphics data is referred to as TMD data.

In the Net Yaroze system, there is a special converter to temporarily convert data from DFX format - the standard 3D modelling format - into the artist-oriented RSD format file, and then to convert this file into the binary TMD format data that the library can deal with.

Data types:	RSD - modelling data (defines the shape of objects)
	TMD - modelling data (binary format)
Tools provided:	DXF → RSD format, RSD→ TMD format converters
Operating environment:	Windows, MS-DOS

Sound Data

Sound data used by the PlayStation consists of score data and sound source data. Score data is called SEQ data and sound source data is called VAB (or VAG data - VAG is the single-sound data, VAB a collection of VAGs). All of these data types can be dealt with directly using the sound services.

In the Net Yaroze system, a special converter makes AIFF data or standard MIDI data (SMF) which has been created using commercial sound tools into SEQ and VAB data.

Data types:	VAG - waveform (single-sound) data
	VAB - waveform (bank - collection of VAGs) data
	SEQ - Sequence data (score data)
Tools provided:	SMF → SEQ format, AIFF→ VAG format converters
Operating environment:	Windows, MS-DOS

Data Verification

The Net Yaroze system provides a viewer and player which, when called from the DOS prompt of your PC, will display sound and graphic data on your TV monitor via the Net Yaroze PlayStation. Thus you can verify sound and graphic data without writing an application to use them.

Using these it is possible to capture application run-time images.

The Link Between Data and Programming Tools

Thus to provide data files such as graphic and sound for PlayStation applications, you need to create and/or edit sound or graphic files in a commercial graphic or sound application and then use the Net Yaroze conversion utilities to change them to the appropriate PlayStation file formats. You can then use these files as part of a Net Yaroze application (when they will be loaded into main memory and accessed by the program code).

Programming Style

In this section, characteristic PlayStation application programming methods will be described, along with explanations of the terminology.

Basic Style

The following sequence describes the basic flow of graphics-processing in the PlayStation. However, in the PlayStation steps (1) to (3) are carried out simultaneously, enabling a continuous flow of new data to the screen.

Basic Style Steps:

(1) Taking data describing a 'world', which has been placed in the memory, a 'drawing command' list structure is formed.

(2) This 'drawing command' list is sent to the GPU which draws polygons in the frame buffer.

(3) The completed image drawn in the frame buffer is displayed on screen.

List structures are created in the main memory, and picture images in the frame buffer. Two groups of each of these are prepared. While the CPU is creating the first list, the second is being transmitted, and while the

GPU is drawing the first picture image, the second is being displayed (GPU image drawing and display are shown below).

Figure: The GPU Draws One Image and Displays Another Simultaneously

The problem with drawing a single image using 3D graphics is that it takes time. To overcome this problem, image creation is divided into two processes, carried out in parallel: drawing command list creation and polygon drawing.

Of the Basic Style Steps listed on the previous page, step (1) is executed by the CPU in accordance with the program stored in the main memory. Step (2) is carried out automatically by the DMA Controller, a hardware device dedicated to data transmission. Step (3) is carried out automatically by the image display part of the GPU. The CPU, and, therefore, the program, is not concerned with the details of the execution of steps (2)

and (3). The CPU is only involved in giving the dedicated hardware very small amounts of data such as the display location and the start address for data transmission. The result of this parallel processing is that the CPU can devote almost all of its time to creating drawing command lists.

The technique of keeping the previous image for display also helps to squeeze out more time for image creation. Japanese and US TV NTSC receivers display an image every 30th or 60th of a second (30 or 60 images are displayed in a second - known as 30 or 60 'frames per second'). In Europe, the display rate is 25 or 50 frames per second. During the time spent displaying one image, another is prepared for display. If an image in preparation is not ready when it comes time to display it, the one already on display is kept there until it is ready. Thus a continuous display is maintained on screen.

The CPU, DMA Controller and GPU control the operation between them thus: the CPU completes a drawing command list structure for one full screen and then, when the DMA Controller has finished transmitting the previous list to the GPU for drawing, the CPU instructs the DMA Controller to transmit its newly created list. At the same time, the drawn image is set up for display in the display part of the GPU.

So:

- CPU makes command list structure
- DMA Controller transmits list structure
- GPU draws one image while it displays another image

In this process there is a time limit. It is not possible simply to switch the displayed image at any time. If image data that is in the process of being displayed is exchanged with other data, image discrepancies and display hardware activity will appear on the screen as noise. Image data switching must, then, be carried out at a time when the act of displaying to the screen is finished (the GPU has finishedmaking the image on screen and the whole image is being shown). The period of time during which this condition is met is called the Vertical Line Return Interval, which recurs at fixed intervals 60 times a second in NTSC and 50 times a second in PAL. The start of each interval is communicated to the CPU as a vertical synchronisation interruption.

As a result, the CPU completes list creation, waits for the DMA Controller to complete transmission, and for the vertical synchronisation interruption. When these three conditions are met, it switches the display image in the GPU and sets up the address of the new list in the DMA Controller. Once this process is complete, the

CPU starts creating the next drawing command list. The state of the Controller is read, and based on this the new image is created. The repetition of this sequence is the basic style of PlayStation programs.

Double Buffering

Double buffering is a technique whereby two processes are carried out in parallel by setting aside two equivalent data storage areas (buffers) and switching these two buffers as appropriate, so that creation and transmission, or drawing and display can be carried out simultaneously. In standard PlayStation programs two drawing command lists called 'ordering tables' (OT) (refer to Z sorting, below) are maintained in the main memory. This speeds up image creation by enabling parallel processing. Similarly, two picture images are maintained in the frame buffer to guarantee time for picture creation.

Nonblocking Functions

Most of processes that are actually performed by dedicated hardware, such as graphics drawing and loading from CD-ROM can be carried out in parallel right up to program execution by the CPU. The functions that activate this kind of parallel processing are called nonblocking functions. A nonblocking function terminates as soon as the relevant processing requests have been recorded by the hardware or OS, and the actual processing is executed after the function has terminated.

In the Net Yaroze system, the completion of processing requested by means of nonblocking functions can be checked at any time by calling a special test function. This programming style, in which a program that makes a processing request explicitly tests for the completion of that processing, is called polling.

Z Sorting

When 3D objects are displayed on a 2D screen, a proper image is not obtained unless surfaces that should be obscured by other surfaces are not displayed. In the PlayStation this is achieved by means of a Z sorting algorithm.

Z sorting is a method of not displaying the surfaces that should be concealed by drawing each surface in order, from the furthest to the nearest in terms of distance from the viewing point. Taking into consideration drawing speed and the scale of the hardware, Z sorting is more appropriate than other methods (Z buffering, for example), but as with sorting in general, as the number of elements to be compared and exchanged increases, the time taken to sort and, in this case, display, dramatically increases. In the PlayStation, ordering tables (OT) and graphic drawing command list structures are used as the basis for the implementation of stable Z sorting.

An OT is an empty drawing command array. An OT contains, for each drawing command, its own size and a pointer that specifies the next drawing command. In the initial state (and in the cleared state), the pointers of each of the elements of an OT point, in an orderly fashion, to the adjacent element. So, element N points to element N+1. In the initial state an OT is physically an array, and logically, a single list (seeOrdering Tables, 6.5).

Every time a drawing command is created, taking the OT array element numbers as co-ordinate values on the axis perpendicular to the screen (the z-axis: where x = vertical, y = horizontal and z = depth), the new command is inserted, by means of a list operation, between the corresponding OT array element and the element to which that element points.

The creation of drawing commands is carried out independently of the co-ordinate value of the z-axis, but as a result of the list operations, they are formed into a single list. Moreover, all drawing commands are linked in between the OT element representing their co-ordinate value on the axis perpendicular to the screen and the following OT element. They therefore exist in a sorted state. (Using the example above, the new command is inserted in the list between N and N+1, N now pointing to the new command and the new command pointing to N+1.)

Thus, even if the number of drawing commands increases, the amount of time required for sorting does not greatly increase, and processing is more likely to be stable.

Following this, this list of linked drawing commands is sent to the GPU with the end that holds the greatest z-axis value first (i.e. the drawing objects furthest from the viewpoint first). As this is a simple operation, it is carried out by the dedicated hardware unit known as the DMA Controller. (The DMA Controller is hardware which executes Z sorting at high speed. It also carries out the initialisation of OTs at high speed.) By this means, the image is drawn from furthest to the nearest surface.

Critical Section

All interrupts can be inhibited by means of a software operation involving the use of the function, EnterCriticalSection(). The resulting state is called the critical section state. During critical section, the processing of vertical synchronisation interrupts (V-blanks), and also of various interrupts that are processed within the OS, are halted. In this situation most library functions operate abnormally. However, there are library functions, such as Exec(), that are guaranteed to operate normallyonly in the critical section state. By

calling the function, ExitCriticalSection(), the program leaves the critical section state and recognises interruptions.

A program is immediately in the critical section state, once the state has been activated.

Synchronisation with Vertical Synchronisation Interruptions

In order to display images created on the PlayStation on a screen without corruption, program operation must proceed in step with the timing of vertical synchronisation interrupts (V-blanks). ('V-blank' - all lines of an image have been drawn on the screen - see diagram).

Figure: Displaying an Image On Screen

The PlayStation OS is equipped with two methods for achieving V-blank synchronisation:

1. Vsync()
   When this function is called with 0 specified as the first argument, Vsync(0), it waits, returning only when a V-blank occurs. Explicit synchronisation of program execution with V-blanks can be achieved by calling this function in the main loop.

2. Callback -
   Using callback, any function can be executed so that it takes advantage of V-blanks. For details refer to the following section,CallbackFunction

Callback Functions

VSyncCallback() is a special function which recognises when a V-blank occurs. It is used to specify (via a parameter) a function to be called when this interrupt occurs.

A function specified in this way is called a callback function. A callback function is executed with the same scope as existed at the point when VSyncCallback() was called, so information can be held in common with the normal program via external variables. These variables must all fit into 4K: The stack uses a 4K memory area set aside within the OS code, so if the number of these variables used is greater than can be accommodated in this area, OS code is destroyed.

Callback functions are executed in the critical section state, that is, when interrupts are inhibited. Beware, the result of leaving the critical section state while within a callback function is not certain.

While a callback function is being executed, the main flow of the program (whatever was being executed when the V-blank occurred) is forcibly suspended for a temporary period. On return from the callback function, information related to the main flow which had been put to one side is reconstructed by the CPU, and execution of the main flow is restarted.

Processing of Vertical Synchronisation Interrupts within the OS

When a V-blank occurs, first of all communication with the Controller and Memory card is carried out. (Communication with the Memory card can only be brought about within a read() or write() function.) Callback functions are called after that. When Vsync() has been called from the main flow (causing the program to wait for synchronisation with a V-blank), and a V-bank occurs, communication with the Controller and the callback function call are carried out from within Vsync(). On return from the callback function, Vsync() terminates and the main flow restarts.

Multiple Vertical Synchronisation Interrupts

If execution of the callback function takes a long time, the next V-blank may occur while the callback function is still being executed. Interrupts are inhibited during execution of the callback function, so subsequent V-blanks are put into a stand-by state. Accordingly, as soon as return from the callback function occurs, this standby V-blank becomes valid and the callback function is called again.

Normally this phenomenon manifests itself as a delay in screen revision or in music reproduction, so is not desirable. Callback functions should take as little time as possible. To find out which part of the callback function is taking too much time, set a counter within the callback function. The Vsync(1) function, which returns the number of horizontal lines drawn (H-blanks) since it was last called, can be used as a rough

counter. (See below for timings of V-blanks and H-blanks.) Note that Vsync(1) can be used within VSyncCallback() despite VSyncCallback() being a critical section.

Note that the interruption standby mechanism can only store a single interruption. Therefore, even if two or more subsequent V-blanks occur during execution of a callback function, only a single interruption will be stored.

Timings of V-blanks and H-blanks:

V-blanks: 50 (PAL) or 60 (NTSC) vertical synchronisation interrupts per second
H-blanks: 311 per V-blank

Video Mode

In the library, the SetVideoMode() function declares current video signal mode- either PAL or NTSC. For PlayStation libraries, the video signal mode is set to NTSC as a default but by calling the SetVideoMode() function prior to any other function calls, all related libraries will check the mode specified and operate accordingly.

Application Environment

In this section stack pointer initialisation, standard start-up routines, and memory mapping executed by application programs are described.

Memory Mapping

In the Net Yaroze system, application memory mapping is as follows.

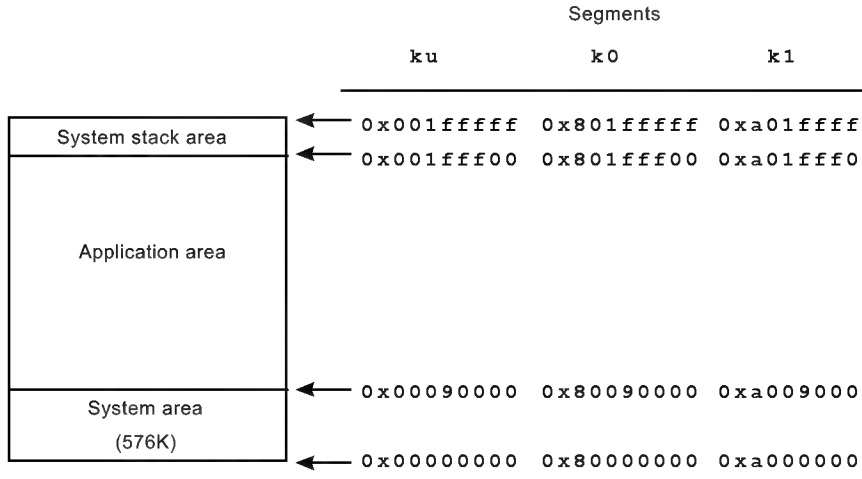


Figure: Memory mapping

The OS itself ('System area' on the above diagram) is allocated to the lowest address part of the memory, and the system stack, which is the OS operating space, is allocated to the highest address part. The rest of the memory is left free for applications.

Start-Up Routines

When an application is activated, a program called the start-up routine is executed before main() is called. The Net Yaroze standard start-up routine is provided as part of the Net Yaroze library.

This program carries out various procedures, including initialisation of global pointers and zero clearing of external variables which do not have initial values.

The Stack Pointer

The value used by the host program is used without modification. Where no explicit host program exists, the system stack is inherited as the pointer.

4

The Net Yaroze library is equipped with graphics, sound, Memory card and CD-ROM management services which are designed to make the most of the PlayStation's capabilities. It also provides the standard C language functions including mathematical functions This chapter gives a broad overview of these services.

Graphics Services

- Frame Buffer Access

  There are sets of services which access the frame buffer and set up the drawing and the display environments.

- Integrated Graphics

  Similarly, there are integrated graphics services which manage the 2D and 3D graphics created by external graphics tools.

Sound Services

The Net Yaroze libraries offer background playback of sound sequences which have previously been recorded as score data (MIDI type data).

Standard Services

Standard C language functions are provided.

- Standard C Functions

  These are a subset of the standard C language library, and include character functions, memory operation functions, character class tests, non-local jumps, and other utility functions.

- Mathematical Functions

  ANSI/IEEE754 standard mathematical functions are provided, including a package which provides floating point arithmetic using software.

Other Services

The PlayStation OS functions can be accessed via special API, CD-ROM management, and peripheral device management services

- Kernel Management

  An interface (API) between applications and the PlayStation OS is provided.

- CD-ROM Management

  There are services for reading image data, sound data and programs from the CD-ROM drive as well as regenerating CD-DA (digital audio) sound.

- Peripheral Device Management

  There are services for managing the peripheral devices such as the Controller and Memory Card, and for managing callbacks for interrupt processing.

File Organisation

This section describes the Net Yaroze library files.

- Header Files

  To use a service provided by the Net Yaroze library, in the source code you need to include a header file that defines the variables and functions used by that service.

  Net Yaroze library header files are organised as shown below. All of the services that particularly draw on PlayStation functions are defined in the file 'libps.h'. Always include 'libps.h' when creating programs.

File	Contents	Notes
abs.h	abs()	included in stdlib.h
asm.h	R3000 register definition	utilised using assembler(*)
assert.h	assert()	
convert.h	atoi(), atol(), etc. (type conversion)	included in stdlib.h
ctype.h	isupper(), toupper(), etc. (type evaluation)	
fs.h	macro definitions	internal use
libps.h	all services related to PlayStation functions	always include this
limits.h	C type limit macro definitions	
malloc.h	malloc(), etc.	included in stdlib.h
memory.h	memcpy(), etc. (memory operations)	included in strings.h
qsort.h	qsort()	included in stdlib.h
r3000.h	R3000 memory definition	utilised using assembler(*)
rand.h	rand(), srand(), etc. (random number generation)	included in stdlib.h
romio.h	-	internal use
setjmp.h	setjmp(), longjmp(), etc. (omit large areas)	
stdarg.h	va_start(), va_end(), etc. (variable arguments)	
stddef.h	type definitions	
stdio.h	standard I/O	
stdlib.h	standard functions	
string.h	strcpy(), etc. (character string operations)	identical to strings.h
strings.h	strcpy(), etc. (character string operations)	
sys/errno.h	errno and error definitions	
sys/fcntl.h	macro definitions	used by sys/file.h
sys/file.h	file I/O macro definitions	used by open(),close(), etc.
sys/ioctl.h	macro definitions	internal use
sys/types.h	type definitions	

Table: Net Yaroze Header Files

* Refer to the Net YarozeWeb site for information related to assembler programming.

- Library Files

  The Net Yaroze library file, libps.a, is automatically linked during compiling, so it needn't be explicitly referred to at any point.

Data Processing

The Net Yaroze library can process PlayStation graphics data and sound data which are in PlayStation format files without any modification to them. These PlayStation format files are created by converting data from standard file formats. (SeeCreating PlayStation Applicationsor the Net YarozeWeb site).

5

Frame Buffer Access

There are access services to the frame buffer, such as setting the drawing and display environments, and transmission of image data to the frame buffer.

These services can be divided the following three groups.

(1) Environment setting functions

(2) Frame buffer access functions

(3) Drawing control functions

Details of the Frame Buffer

Pixels

From a software point of view, a frame buffer is a 1024 x 512 two-dimensional address space composed of 16-bit pixels. The top left pixel has the coordinates (0, 0) and the bottom right pixel has the coordinates (1023, 511). Each pixel is made up of three 5-bit data items indicating RGB brightness values, each ranging from 0~31, and a semi-transparency flag.

The semi-transparency flag is only valid when the pixel is used as a texture.

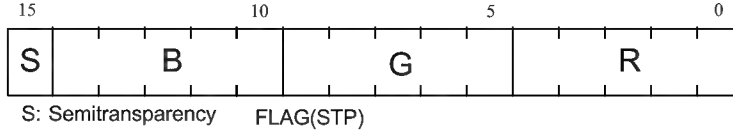


Figure: Pixel structure

Display Area

The part of the frame buffer that is displayed is a rectangular area known as the 'display area'. Based on the GPU display function, the display area can be selected as a pair of values ranging from 256 x 240 to 640 x 480 for NTSC and 256 x 256 to 640 x 512 for PAL. Interlace mode is on when the display height is 480 for NTSC or 512 for PAL.

Drawing Area

Drawing is limited to a rectangular area, referred to as the 'drawing area' within the frame buffer. The drawing area may be any size that can be accommodated in the frame buffer.

Environment Setting Functions

Information related to drawing as a whole, such as where drawing is to be carried out within the frame buffer and the starting point (offset) for drawing is referred to as the 'drawing environment'. In the same way, information related to display of the frame buffer, such as which part of the frame buffer to display is referred to as the 'display environment'. The drawing environment is set up using the GsInitGraph() function, and the display environment is set up using the GsDefDispBuff() function.

GsInitGraph()sets the following arguments:

x_res Horizontal resolution (256/320/384/512/640)

This sets the horizontal screen resolution, and the range of the display and drawing areas. Drawing is clipped to the specified range.

y_res Vertical resolution (240/480) for NTSC and (256/512) for PAL.

This sets the vertical screen resolution, and the range of the display and drawing areas. Drawing is clipped to the specified range.

intl Attributes

The interlaced display flag (bit0)

Non-interlaced display is set when 'bit0' is 0, and interlaced display when 'bit0' is 1.

Set 'bit0' to 1 when using a display height of 480 in NTSC or 512 in PAL.

The offset specification flag (bit 2)

Set 'bit2' to 0 so the GTE offset is used, and set it to 1 so it is NOT used. Usually, GPU offset tends to be used.

vram VRAM mode

When the VRAM mode is 0, 16 bits are displayed as 1 pixel, when it is 1, 24 bits are displayed as 1 pixel. When the VRAM mode is 1, only frame buffer access functions, such as the 'LoadImage()' function, are available. Other drawing functions, such as the GsDrawOt() function, are only available when the VRAM mode is 0.

GsDefDispBuff() can set the following arguments:

x0,y0 The top left coordinates of image buffer 0.

x1,y0 The top left coordinates of image buffer 1.

Image buffers '0' and '1' are rectangular areas on the frame buffer. The top left corners are (x0, y0) and (x1, y1), and the width and height are x_res and y_res, as specified by the GsInitGraph() function. One of these image buffers is used as the drawing area and the other as the display area. The buffers allocated for the drawing and display areas are swapped each time GsSwapDispBuff() is called. This allows the implementation of double buffering.

Frame Buffer Access Functions

Transmit data within the frame buffers, or between the frame buffers and the main memory using the following functions. Apart from these functions there is no way to operate directly on data in the frame buffers.

Function Name	Direction of Transmission
LoadImage()	main memory→ frame buffer
StoreImage()	frame buffer→ main memory
MoveImage()	frame buffer→ frame buffer

Table: Frame Buffer Access Functions

These functions are non-blocking functions, that is to say, they terminate as soon as the access requests have been registered by the system. Up to 64 access requests may be registered (this number is the total for all three functions). These functions will block, i.e. not terminate, while the processing requests cannot be registered.

Drawing Control Functions

Use the following functions to control drawing (specifically, the transmission of drawing commands to the GPU by the DMA Controller) and frame buffer access request processing.

Function Name	Action
GsDrawOT()	starts drawing
DrawSync()	waits for the termination of both drawing and the processing of frame buffer access requests, and ascertains state of progress.
ResetGraph()	stops drawing

Table: Drawing Control Functions

Drawing Control in Non-Interlaced Mode

In non-interlaced mode, there is a simple rule that image buffer switching is not carried out until drawing is completed.

After DrawSync(0) detects that drawing is complete, VSync(0) is executed. This delays the program until a vertical synchronisation interrupt occurs. At which point the image buffers are exchanged using GsSwapDispBuff().

```
while (1) {
  ....
  DrawSync(0);
  VSync(0);
  GsSwapDispBuff();
  ....
}
```

Interlaced Mode

In interlaced mode, the pixels with odd and even numbered vertical coordinates are displayed alternately every 1/60th of a second in NTSC, or every 1/50th of a second in PAL. This automatic double buffering is compulsory.

As a result of this characteristic, in interlaced mode the display area and the drawing area are perfectly stacked on top of each other, thus saving lots of frame buffer space. However, you should always remember that the switching of display areas occurs every 1/60th of a second in NTSC or 1/50th of a second in PAL, regardless of the workings of the program. Therefore, the program must switch the image buffers at exactly the same time.

Drawing Control in Interlaced Mode

In interlaced mode, due to the display mechanism used, calculation and drawing must always be completed in 1/60th of a second (in NTSC) or 1/50th of a second (in PAL). Therefore, it is essential that buffer switching cued by vertical synchronisation be carried out, irrespective of whether the drawing is complete. As a result, ResetGraph(3) is called following Vsync(0), without using DrawSync() at all.

```
while (1) {
    ....
    VSync(0);
    ResetGraph(3);
    GsSwapDispBuff();
    ....;
}
```

Kanji [Japanese Character] Fonts

There are two value bit mapped 16-bit x 16-bit kanji fonts stored in the PlayStation. You can use these to create messages.

6

Integrated Graphics

The integrated graphics service handles 3D and 2D graphics (using polygons, sprites, background surfaces).

The following services are supported as part of the integrated graphics service.

1. Hierarchical co-ordinate systems
2. Light source calculation (3 parallel light sources, depth cueing, and ambient light)
3. Automatic object partitioning (polygon sub-division) and semi-transparency processing
4. Viewpoint management
5. Z sorting
6. OT (ordering table) initialisation, hierarchical organisation, and compression
7. Frame double buffering
8. Automatic aspect ratio adjustment
9. 2D clipping, offset processing
10. Sprites / backgrounds / lines

In addition, data created using all the graphics tools can be handled without modification.

The 3D model format used by the integrated graphics service is called 'TMD'. This format stores the information required to define a 3D model (efficiently polygon vertices, surface normals, texture co-ordinates extremely) extremely efficiently.

The 2D graphics data format used by the integrated graphics service is called 'TIM', which stores format, image resolution, number of colours, colour lookup table (CLUT) information, along with pixel data.

For details of the TMD and TIM formats refer to Chapter 15, Graphics Tools

The Processing Sequence

The flow chart below describes the integrated graphics process drawing sequence.

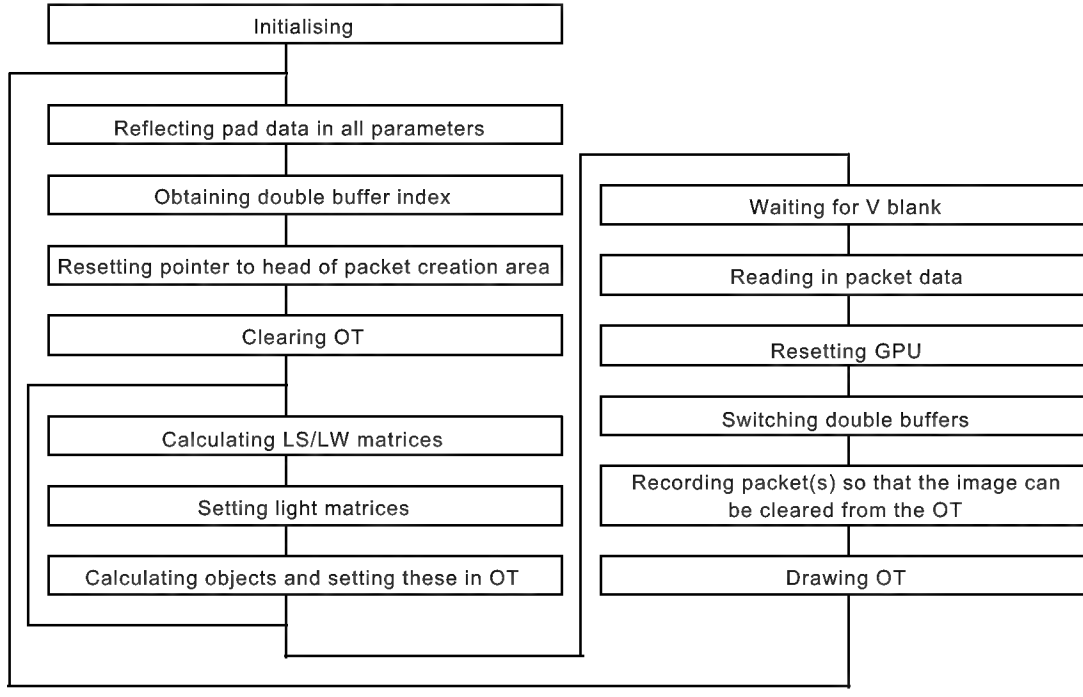


Figure: Drawing Processing in the Integrated Graphics Service

The drawing process sequence is as follows.

1. Initialise the drawing and display environmentsand variables to be used.(GsInitGraph, GsDefDispBuf)
2. Set parameters to reflect pad data.
3. Set the view point is set. (GsSetRefView2, GsSetView2)
4. Set up the packet creation working area. (GsSetWorkBase)
5. Clear the ordering table. (GsClearOt)
6. Calculate the LS/LW (Local Screen/Local World) matrices. (GsGetLs, GsGetLw)
7. Set up the LS/LW (Local Screen/Local World) matrices. (GsSetLightMatrix, GsSetLsMatrix)
8. After calculating co-ordinates, perspective conversion and light source, record the drawing packse are in the ordering table. (GsSortObject4)
9. Wait for a drawing completion and then wait for a V blank. (DrawSync, Vsync)
10. Swicth the double buffers. (GsSwapDispBuff)
11. Draw the ordering table. (GsDrawOt)
12. Return to step 2.

Graphics System Initialisation

Use the GsInitGraph() function to initialise the graphics sub-system, and set the screen resolution and interlace mode. As GsInitGraph() initialises various internal variables it must be called before the integrated graphics service is used.

Use the GsDefDispBuff() function to define the two rectangular areas of the frame buffer used during double buffering. This also initialises the clipping area and GPU offset co-ordinates so ther can be no drawing outside the drawing area. Cahnge these drawing environment attributes using the GsSetClip2D() and GsSetOrign() functions respectively.

The GsSwapDispBuff() function switches the double buffer, while GsGetActiveBuff() identifies the currebt drawing area.

The Viewpoint

As a flat, 2D TV screen cannot actually display 3D graphics, a 3D graphic has to be effected. This is done by choosing a viewpoint in front of a theoretical screen and projecting 3D space onto the theoretical screen.

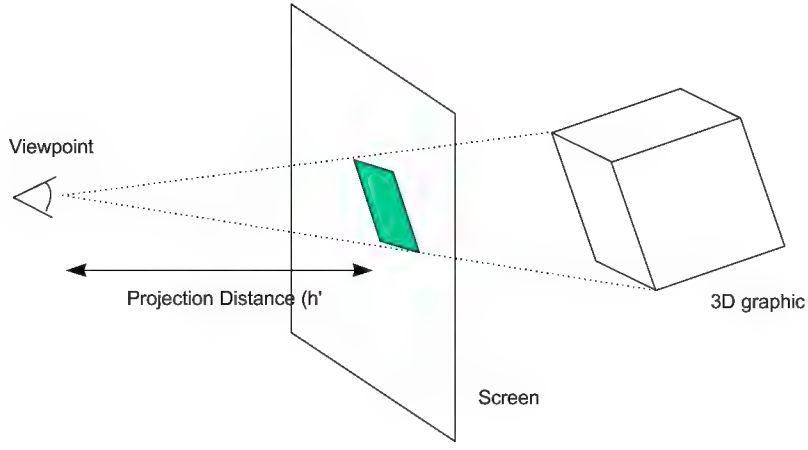


Figure: Effecting a 3D Image on a Screen

Therefore, in order to project images onto the physical screen, you need to establish a viewpoint and a theoretical screen.

Setting the viewpoint

Set the viewpoint by initialising the members of either a GsRVIEW2 or GsVIEW2 structure, and calling either the GsSetRefView2() or GsSetView2() function, respectively.

Both GsRVIEW2 and GsVIEW2 set the viewpoint but by different methods: GsRVIEW2 sets the co-ordinates of the viewpoint and a reference point, while GsVIEW2 directly sets up a matrix for conversion to the viewpoint co-ordinate system.

You can set up a hierarchical co-ordinate system along with GsVIEW2 or GsRVIEW2. For example, if you take the standard co-ordinate system as the world co-ordinate system, the result is an ordinary camera that captures an objective view. Alternatively if you take the standard co-ordinate system as the local co-ordinate system of an object, then the result is a camera that captures the subjective view of that object.

Setting the Theoretical Screen

Set the distance between the viewpoint and theoretical screen, the 'projection distance (h)', using GsSetProjection().

The Theoretical Screen

The height and width of the theoretical screen should be the resolution of the physical screen. For example, if the resolution is 640 x 480, the width of the theoretical screen is 640 and the height 480.

If the screen resolution does not comprise a regular dot configuration (that is, the horizontal to vertical ratio of the screen resolution is not 4 to 3), the vertical values are adjusted. For example, in the case of a 640 x 240 dot configuration, objects are displayed with half the origin vertical values, and the aspect ratio will appear to be the same as for a regular dot configuration.

The Projection Distance

Adjust the filed of view by altering the projection distance. The longer the projection distance, the narrower the field of view (this tends towards a parallel projection). Alternatively the shorter the projection distance, the wider the field of view which emphasises the impression of there being close and distant objects from the viewpoint.

Packets

A 'packet' (or 'primitive') is the smallest unit of drawing commands that can be dealt with by the GPU. In the integrated graphics service, use the GsSortObject4() function to create packets. However, when GsSortObject4() creates packets, the area occupied by them expands or contracts depending on the number and type of polygons. Therefore you need to set up a working area for packet creation beforehand which allows room for expansion. Use the GsSetWorkBase() function to allocate a working area, it returns a value

indicating how much of the working area is being used in packet creation. Note that double buffering can only work if there are two packet creation areas.

As the packet data used for drawing is located in the packet creation working area, drawing cannot be carried out correctly if this area is also being accessed by GsSortObject4().

Ordering Tables

When displaying 3D graphics, some polygons are hidden and some visible. An ordering table specifies polygons as hidden or visible by using a simple z sorting routine. Sorting the polygons in furthest first order ensures that closer polygons aren't covered by further polygons.

An ordering table (OT) is like a z axis 'ruler' in the memory, and each scale mark on this 'ruler' can possess any number of polygons.

Sorting is based on the average Z values of polygons, and is carried out by 'placing' polygons in the scale mark equivalent to their average Z value. The ordering table is transmitted to the rendering chip in scale mark order, thus hidden surfaces are cancelled out as drawing progresses.

GsOT

In the integrated graphics service, use a 'GsOT' structure to handle an OT. This structure contains a pointer to the OT itself (the org member) and a number of parameters indicating the attributes of the OT.

Select the resolution on the Z axis (or ruler markings) by using the member 'length'. This can be set at 14 levels ranging from 2 to the power of 1, to 2 to the power of 14.

GsOT_TAG

The actual OT is defined as an array of GsOT_TAGs. Each entry in the OT represents the single marks of the Z axis ruler. For example, if 'length' is 4, then the actual OT will be an array of 16 OT_TAGs (16 is 2 to the power of 4).

OT Initialisation

Use the GsClearOt() function to initialise an OT. GsClearOt() takes three arguments - 'offset', 'point' and 'otp'. 'otp' is a pointer to the OT handler. ('offset' and 'point' are described later.)

When an OT is initialised, it is in an empty state with no polygons linked to it. An OT must be initialised each time.

Multiple OTs

You can use multiple OTs at any one time. The GsSortOt() function can merge and sort multiple within one global OT. The representative Z value for the entire local OT is stored in the GsOT member 'point'. This is used when inserting and sorting the local OT into the global OT. Using multiple OTs allows the order of sorting to be controlled.

For example, you can sort in object units by preparing a local OT for each object. These local OTs are then sorted and merged together into a single global OT. This technique is extremely useful if the depth relation between objects (i.e. which objects are in front and behind) is already known. (For instance: when you are looking down from a helicopter on cars that are driving along a road you already know that the helicopter is in front of all the cars).

OT Compression

Using OTs increases the speed of sorting, but OTs consume a considerable amount of memory.

You can reduce an OT's memory consumption by making 'length' smaller. Doing this, though, reduces the sorting resolution, and can result in flickering polygons on screen.

However, when you know that the Z values of the polygons to be sorted are all higher than a certain value ('x') you can use offsetting to reduce the amount of memory consumed by OTs without reducing the resolution. To implement this, set the 'offset' parameter value in GsClearOt() function to the lowest value ('x'). Using this the OT does not use up any memory for the part of the table below the offset value ('x'), so reducing the amount of memory.

Co-ordinate Conversion & Light Source Calculation

Co-ordinate Conversion

In the PlayStation the vertex data for the objects to be displayed is handled using 3D co-ordinates, so images of an object seen from various viewpoints can be created simply by calculation. By doing this, the game's

viewpoint can be freely selected to constitute the player's viewpoint, or the viewpoint of one of the characters in a scene.

In order to display objects expressed in terms of 3D co-ordinates on a screen, 3D images need to be expressed as 2D images via a projection conversion. This projection conversion calculation takes the form of: multiplying by a 3 x 3 rotation matrix and adding of 3D parallel motion (translation) vectors. The result is then divided by the distance form the viewpoint in order to give perspective (so objects further away appear smaller).

For example, taking the co-ordinate system in which the object you want to display is situated at (Xw, Yw, Zw), and the co-ordinate system in which the theoretical screen is placed at (Xs, Ys, Zs), the following equation (Equation 1) expresses the projection conversion calculation.

Equation 1

$$\begin{bmatrix} Xs \\ Ys \\ Zs \end{bmatrix} = \begin{bmatrix} SW11, SW12, SW13 \\ SW21, SW22, SW23 \\ SW31, SW32, SW33 \end{bmatrix} \begin{bmatrix} Xw \\ Yw \\ Zw \end{bmatrix} + \begin{bmatrix} SWx \\ SWy \\ SWz \end{bmatrix}$$

* Swij is the world/theoretical screen conversion matrix.

If you want to display more than one object, each of which movable independently, you need to allocate a co-ordinate system (Xl, Yl, Zl) for each independent object. In the PlayStation these three types of co-ordinate system are referred to as follows.

(Xl, Yl, Zl)	Local co-ordinate system (a co-ordinate system possessed by an object and centred on that object)
(Xw, Yw, Zw)	World co-ordinate system (a co-ordinate system, fixed for a world in which objects are placed)
(Xs, Ys, Zs)	Theoretical screen co-ordinate system (a co-ordinate system fixed for a theoretical screen and centred on a viewpoint)

Vertex co-ordinates are usually expressed in terms of a local co-ordinate system, so the following conversions are necessary in order to display an object on a theoretical screen.

local → world → screen

Equation 2 (below) and Equation 1 (above), together achieve the projection conversion calculation. Equation 3 and simplified in 4 (both below) below represent them both together.

Equation 2

$$\begin{bmatrix} Xw \\ Yw \\ Zw \end{bmatrix} = \begin{bmatrix} WL11, WL12, WL13 \\ WL21, WL22, WL23 \\ WL31, WL32, WL33 \end{bmatrix} \begin{bmatrix} XL \\ YL \\ ZL \end{bmatrix} + \begin{bmatrix} WLx \\ WLy \\ WLz \end{bmatrix}$$

Equation 3

$$\begin{bmatrix} Xs \\ Ys \\ Zs \end{bmatrix} = \begin{bmatrix} SW11, SW12, SW13 \\ SW21, SW22, SW23 \\ SW31, SW32, SW33 \end{bmatrix} \begin{bmatrix} WL11, WL12, WL13 \\ WL21, WL22, WL23 \\ WL31, WL32, WL33 \end{bmatrix} \begin{bmatrix} XL \\ YL \\ ZL \end{bmatrix}$$

$$+ \begin{bmatrix} SW11, SW12, SW13 \\ SW21, SW22, SW23 \\ SW31, SW32, Sw33 \end{bmatrix} \begin{bmatrix} WLx \\ WLy \\ WLz \end{bmatrix} + \begin{bmatrix} SWx \\ SWy \\ SWz \end{bmatrix}$$

Simplifying this equations gives:

Equation 4

$$\begin{bmatrix} Xs \\ Ys \\ Zs \end{bmatrix} = \begin{bmatrix} SL11, SL12, SL13 \\ SL21, SL22, SL23 \\ SL31, SL32, SL33 \end{bmatrix} \begin{bmatrix} XL \\ YL \\ ZL \end{bmatrix} + \begin{bmatrix} Trx \\ Try \\ Trz \end{bmatrix}$$

The GsGetLs() function can find SLij and Tr. Also, as shown in Equation 5 (below), by multiplying the theoretical screen co-ordinate values by h/Zs, the image will appear to have been projected in parallel onto the theoretical screen, and perspective conversion applied (i.e. image size scaled according to distance from view).

Equation 5

$$Xss = Xs\frac{h}{Zs}$$

$$Yss = Ys\frac{h}{Zs}$$

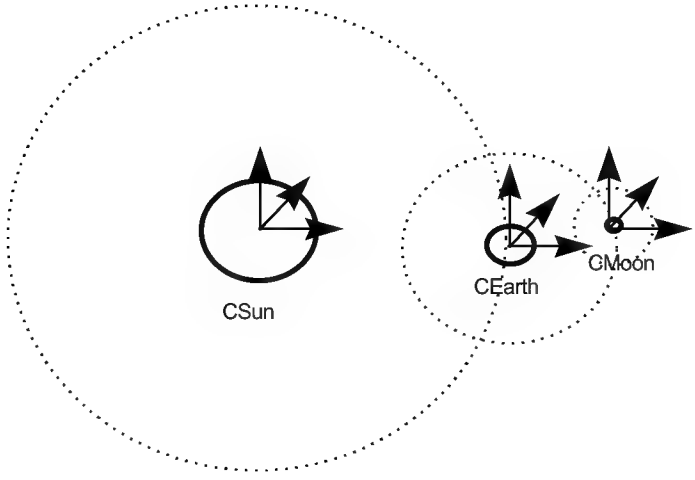
* 'h' is the distance (projection) from the viewpoint to the theoretical screen.

The GsSortObject4() function takes TMD data (which only contains the local co-ordinate system) and creates the polygon drawing packet for a polygon projected onto a theoretical screen. It does this by setting up the matrices in Equation 2 using the 'coord' member of the GsCOORDINATE2 function, the matrices in Equation 4 using the GsSetLsMatrix() function, and 'h' in Equation 5 using the GsSetProjection() function.

Hierarchical Co-ordinate Systems

Hierarchical co-ordinate systems involve the introduction of a layered tree structure into object co-ordinates. Consider, for example, a situation in which the earth is travelling in an orbit around the sun, and the moon is orbiting around the earth. When the sun forms the origin of the world co-ordinate system, it is far easier to describe the motion of the moon using an earth co-ordinate system which takes the position of the earth as its origin, rather than using the world co-ordinate system centred on the sun. In this situation, the best approach is to set up a pointer in the 'super' member of the GsCOORDINATE2 function referred to by the moon's object handler, and to set up the moon's co-ordinate system under the earth's co-ordinate system.

(See diagram, below.)




```
CSun {
        coord   ...
        super   WORLD;
}
CEarth {
        coord   ... /* describes orbital motion around the sun */
        super   CSun;
}
CMoon {
        coord   ... /* describes orbital motion around the earth */
        super   CEarth;
}
```

Figure: Hierarchical Co-ordinate Systems

Light Source Calculation

The light source calculation determines the contribution of light from each light source within the world. The PlayStation supports the following three methods for light source calculation.

(Note that the normal line vector or 'normal' is an imaginary line perpendicular to either the surface of a polygon or the polygon vertices (the points that define the position of the polygon). Similarly the Light Source Vector is an imaginary line that defines where the light in an on-screen scene is coming from.)

- Flat shading

  In this method each polygon has a single normal line vector from its surface (the 'flat' or 'face' normal). The colour and brightness of the polygon is determined by the size of the inner angle between this normal line and the light source vector.

- Gouraud shading

  In this method each polygon vertex (each point that is a 'corner' of the polygon) has its own normal line vector. The colour and brightness at each vertex is determined by the size of the inner angle between the vertex normal and the light source. The colour and brightness at each vertex is then interpolated to approximate the light source contribution across the polygon. This produces a smooth graduation of colour and brightness across the polygon.

- Depth cueing

  This method varies the colour and brightness of polygons depending on the distance from the viewpoint. This achieves a fogging effect by making the colour of a polygon more similar to the background colour, the further away it is from the viewpoint.

The description of light source calculation in this guide is based on the following model (see over).

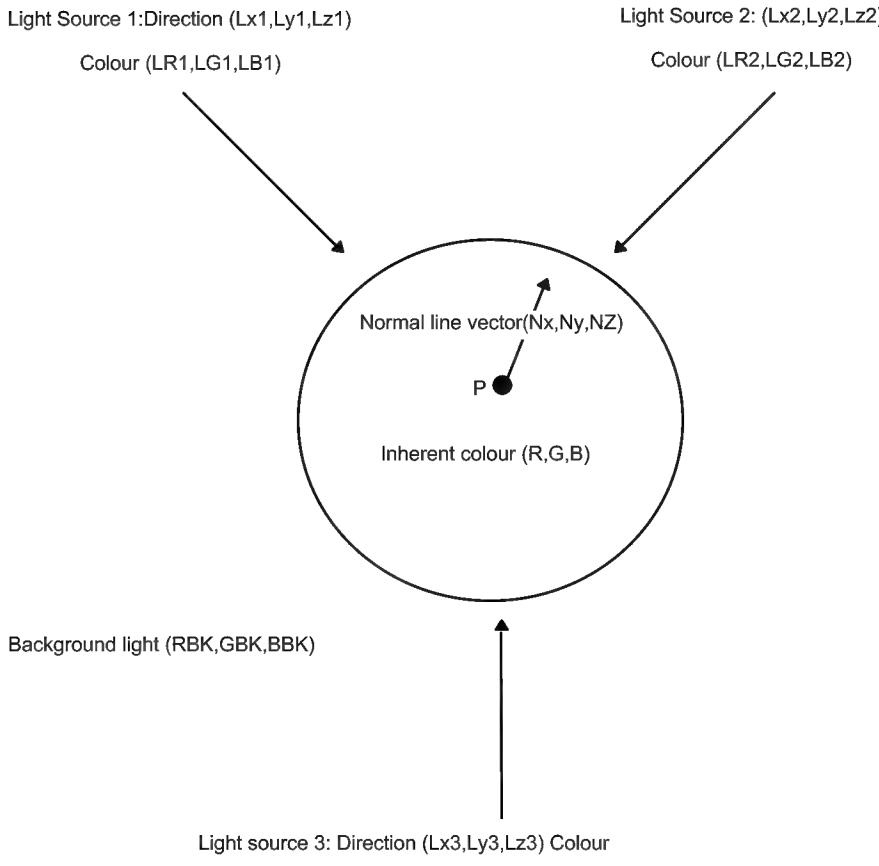


Figure: Light Source Calculation

In this figure, point P represents a point on the surface of an object. To calculate the light source contribution at point P the following terms are used:

(Nx, Ny, Nz) = The normal line vector at point P.

(R, G, B) = The inherent colour of the object.

(Lx, Ly, Lz) = The light source vector.

(LR, LG, LB) = The light source colour.

(RBK, GBK, BBK) = The background (ambient) light.

The following is an example of the light source calculation sequence in the PlayStation if three light sources are used. (RR, GG, BB) represents the final colour values used when point P is displayed.

(1) Convert co-ordinates ofthe normal line vector into the world co-ordinate system.

$$\begin{bmatrix} NWx \\ NWy \\ NWz \end{bmatrix} = \begin{bmatrix} WL11, WL12, WL13 \\ WL21, WL22, WL23 \\ WL31, WL32, WL33 \end{bmatrix} \begin{bmatrix} Nx \\ Ny \\ Nz \end{bmatrix}$$

(2) Obtain the inner product of the light source vector and the normal line vector.

Normal line vector (world)• Light source vector→ Light source influence (L)

$$\begin{bmatrix} L1 \\ L2 \\ L3 \end{bmatrix} = \begin{bmatrix} Lx1, Ly1, Lz1 \\ Lx2, Ly2, Lz2 \\ Lx3, Ly3, Lz3 \end{bmatrix} \begin{bmatrix} NWx \\ NWy \\ NWz \end{bmatrix}$$

(3) Obtain the light source influence colour by multiplying the inner product and the light source colour separately for each item.

Light source influence (L) x Light source colour (Lr, Lg, Lb)→ Light source influence colour (LI)

$$\begin{bmatrix} LIr \\ LIg \\ LIb \end{bmatrix} = \begin{bmatrix} Lr1, Lr2, Lr3 \\ Lg1, Lg2, Lg3 \\ Lb1, Lb2, Lb3 \end{bmatrix} \begin{bmatrix} L1 \\ L2 \\ L3 \end{bmatrix}$$

(4) Obtain the total influence colour from the environment by adding together the light source influence colour and the ambient colour.

Light source influence colour (LI) + Ambient colour (BK)→ influence colour (LT)

$$\begin{bmatrix} LTr \\ LTg \\ LTb \end{bmatrix} = \begin{bmatrix} LIr \\ LIg \\ LIb \end{bmatrix} + \begin{bmatrix} BKr \\ BKg \\ BKb \end{bmatrix}$$

(5) Obtain the theoretical screen colour by multiplying the inherent colour with the influence colour for each item separately.

$$\begin{bmatrix} RR \\ GG \\ BB \end{bmatrix} = \begin{bmatrix} R * LTr \\ G * LTg \\ B * LTb \end{bmatrix}$$

Use GsGetLw()to calculate the local world matrix. In order for this to be applied (as in the Wlij matrix in Equation 1, above) set it internally using the GsSetLightMatrix() function.

Set the light source vector and the light source using the GsSetFlatLight() function, and the ambient colour using the GsSetAmbient() function.

When all the above are set, use the GsSortObject4() function to make the light source calculation as shown accordance with the equations given above.

Packet Creation

The GsSortObject4() function makes the co-ordinate calculation, perspective conversion and light source calculation on polygon data defined within TMD data. The results are converted into drawing packets which are then added to an ordering table.

In the GsSortObject4() function, GsDOBJ2 handles the object data (TMD data). So that TMD data can be handled using GsDOBJ2, the TMD data is mapped to a real address using the GsMapModelingData() function, and then linked to GsDOBJ2 using the GsLinkObject4() function.

The GsSortObject4() function carries out the co-ordinate conversion with the local to screen matrix set using the GsSetLsMatrix() function. It then carries out the perspective conversion using the projection distance set using the GsSetProjection() function. Next, the GsSortObject4() function carries out the light source calculation with the local world light matrix set using the GsSetLightMatrix() function. After that the GsSortObject4() function creates a drawing packet, and then, using the Z values found during co-ordinate conversion, links the drawing packet to an ordering table.

Use the GsDrawOt() function to draw a drawing packet that has been linked to an ordering table. Use the DrawSync() function to detect when drawing is complete.

Objects

GsDOBJ2 is a handler to deal with objects (TMD data). Use the GsDOBJ2 members, 'coord2' and 'attribute' to operate on objects.

'coord2' is a pointer to a co-ordinate system GsCOORDINATE2. The position and orientation of the object are controlled by the setting the parameters of co-ordinate system GsCOORDINATE2 to which 'coord2' points. 'attribute' is the member which sets the object's attributes.

'attribute' sets the following of the object's attributes.

Light Source Mode

This determines the method of light source calculation.

Light Source Calculation OFF

This compulsorily cancels light source calculation.

Inhibit Display

This cancels packet creation for the whole of the object.

Automatic Partitioning

If turned on, this attribute causes all the polygons contained within the object to sub-divide. The number of sub-divisions, are 2 x 2, 4 x 4, 8 x 8, 16 x 16, 32 x 32 and 64 x 64. Sub-dividing polygons in this way reduces texture distortion and polygon fragmentation.

7

Sound

The sound service, for playing background and in-game sound data, is composed of the following function groups.

1. SEQ access functions to handle score data (that is SEQ data).
2. Individual sound setting functions to handle individual-sound sound effects..
3. Shared attribute setting functions to carry out essential settings to enable use of the sound service, and the setting attributes that are shared by all voices of the SPU.
4. Sound utility functions to change the attribute tables within VAB data at run-time, and to implement effects for allocated voices after 'KeyOn'.

Score Data

The data format for score data within the sound service is 'SEQ data format'.

The SEQ Data Format

An SEQ data format file is an SMF (Standard Midi File) format 1 file which as been converted for PlayStation use. SEQ is a format in which all MIDI data structure tracks and chunk data are merged in time order. Sixteen tunes can be reproduced simultaneously. The notes in the tunes are expressed as status (1 byte), data (the number of bytes is determined by the status byte) and delta time (variable length expression, maximum 4 bytes).

In SEQ uses format running status, with 'note on status' converted to 'note off status' and velocity 0. Within the MIDI standard, the following status data are supported in SEQ format.

- note on
- note off
- program change
- pitch bend

The following items among 'control change'.

- data entry(6)

- main volume(7)
- panpot(10)
- expression(11)
- nrpn data(98,99)

(Note that the number in brackets is the Controller number.)

MIDI Support

Setting VAB Attribute Data Using Control Change

NRPN data is defined so that you can use MIDI standard Control Change NRPN to set VAB attribute data.

When creating an SMF file using a sequencer, set VAB attribute data by sending attribute data as follows, in the sequence given.

bnH	99	data1	(NRPN MSB)
bnH	98	data2	(NRPN LSB)
bnH	06	data3	(Data Entry)

Setting Repetition within Tunes Using Control Change

The function that repeats part of a tune is achieved using NRPN data. The number of repetitions can be set within the range 0~126 (127 specifies an endless loop). The repetition symbol '||:' corresponds to Loop1, and the symbol ':||' corresponds to Loop2. Use loops any number of times within a tune, but not in a nested state. The following, for example, can't be used: (Loop1...(Loop1'...Loop2')...Loop2).

Attribute	data1(CC 99)	data2(CC 06)
Loop1(start)	20	0 127
Loop2(end)	30	

Table: Repetition using Control Change

Note 1

Depending on the sequencer used, even when data is entered in the correct order, if the same Delta Time is set, there is a possibility that the order will be switched during conversion to SMF and the data rendered invalid, so do not set identical Delta Times. Also when setting VAB attribute data, note that values become valid from the Key On immediately after Data Entry has been read.

Note 2

Set a repetition in a single location only within one tune (there is no need to set the repetition for each channel individually).

Sound Source Data

The data format defined for sound source data within the sound service is called 'VAB format'.

VAB Format

Waveform data (called 'VAG format' data) is created by sampling sounds such as piano or explosion sounds, and then compressing and encoding these sounds for PlayStation use. The VAB format is a sound source control data format that collects together this VAG format data into a whole, and constitutes the unit dealt with as a file at run-time.

A VAB file contains all of the sound source actually used in a particular scene (that is sound effects). Multitimbral sound (multisampling) is supported using hierarchical control.

A single VAB file can possess a maximum of 128 programs, and each of these programs can possess a maximum of 16 sound tone lists. Also, each VAB file can contain a maximum of 254 items of VAG data.

As it is possible for more than one tone list to refer to the same waveform, and to play a single waveform in more than one way.

Up to 16 VAB files can be utilised simultaneously.

For data format details, refer to the Net YarozeWeb site.

Function Execution Order

When you use sound service functions, use them in the following order.

1. Data open

   Execute the SsSeqOpen() function.

2. Essential processing

   After carrying out main volume setting, execute the essential processing.

   Data close

   Execute the SsSeqClose() function.

8

Standard C Functions

Net Yaroze provides a set of standard headers and functions that are broadly similar to normal C (as defined in Kernighan & Ritchie's definitive,The C Programming Language

Include Headers

File	Contents	Notes
abs.h	abs()	included in stdlib.h
assert.h	assert()	
convert.h	atoi(), atol(), etc. (type conversion)	included in stdlib.h
ctype.h	isupper(), toupper(), etc. (type evaluation)	
fs.h	macro definitions	for internal use only
limits.h	C type limit macro definitions	
malloc.h	malloc(), etc.	included in stdlib.h
memory.h	memcpy(), etc. (memory operations)	included in strings.h
qsort.h	qsort()	included in stdlib.h
rand.h	rand(), srand(), etc. (random number generation)	included in stdlib.h
setjmp.h	setjmp(), longjmp(), etc. (omit large areas)	
stdarg.h	va_start(), va_end(), etc. (variable arguments)	
stddef.h	type definitions	
stdio.h	standard I/O	
stdlib.h	standard functions	
string.h	strcpy(), etc. (character string operations)	identical to strings.h
strings.h	strcpy(), etc. (character string operations)	
sys/errno.h	errno and error definitions	
sys/fcntl.h	macro definitions	used by sys/file.h
sys/file.h	file I/O macro definitions	used by open(),close(), etc.
sys/ioctl.h	macro definitions	for internal use only
sys/types.h	type definitions	

Functions Supported

The following standard C functions are supported.

Function	Description
abs	calculates absolute values
atoi	converts character strings to integers
atol	converts character strings to long integers
bcmp	compares memory blocks
bcopy	copies memory blocks
bsearch	carries out binary searches
bzero	writes zeros to memory blocks
calloc	allocates the main memory
exit	causes programs to terminate normally
free	frees memory blocks that have been allocated
getc	obtains a single character from the stream
getchar	obtains a single character from the standard input stream
gets	reads in a character string from the standard input stream
isXXX	carries out character testing
labs	calculates absolute value of longs
malloc	allocates memory
memchr	searches for characters within memory blocks
memcmp	carries out comparison of memory blocks
memcpy	carries out copying of memory blocks
memmove	carries out copying of memory blocks
memset	writes specified characters to memory blocks
printf	formats output to standard output (stdout)
putc	outputs a single character to a specified stream
putchar	outputs a single character to stdout
puts	outputs a character string to stdout
qsort	carries out quick sorting

rand	generates random numbers
realloc	reallocates heap memory
srand	initialises the random number generator
strcat	concatenates one character string to another character string
strchr	searches for the position at which a specified character appears in a character string
strcmp	compares two character strings
strcpy	copies a character string
strcspn	returns the length of the first part of a character string that is composed entirely of characters not included in a specified collection of characters
strlen	finds the number of characters in a character string
strncat	concatenates a specified length of a character string to another character string
strncmp	compares two character strings with each other
strncpy	copies a specified length of character string
strpbrk	searches for the position at which a specified character first appears in a character string
strrchr	searches for the position at which a character included in a specified collection of characters first appears in a character string
strspn	searches for the first part of a character string that is composed only of characters that occur within a specified collection of characters
strstr	searches for the position at which a specified partial character string occurs
strtok	searches for a character string bounded by characters included in a specified character combination
strtol	converts a character string into a long
strtoul	converts a character string into an unsigned long
toascii	masks the 7th bit of an input value
tolower	converts characters to lower case
toupper	converts characters to upper case

9

Mathematical Functions

Floating-point Numbers

The Net Yaroze system includes a standard set of C mathematical functions that support IEEE 754 standard single-precision floating-point numbers ('float') and double-precision floating-point numbers ('double').

Because the PlayStation version of the R3000 CPU does not possess a floating-point arithmetic co-processor, the floating-point arithmetic package is implemented entirely in software.

Attribute	Specification
Size	4 bytes
No. of digits available	6 digits (decimal number conversion)
Overflow limit value (Largest number)	$2.0^{128} = 3.4e38$ *
Underflow limit value (Lowest Number)	$0.5^{126} = 2.2e-38$ *

Table: 'Float' DataType

Attribute	Specification
Size	8 bytes
No. of digits available	15 digits (decimal number conversion)
Overflow limit value (Largest Number)	$2.0^{1024} = 1.8e308$ *
Underflow limit value (Largest Number)	$0.5^{1022} = 2.2e-308$ *

Table: 'Double' Data Type

* Note: In scientific notation, 'e' means '10 to the power of' so, 2.2e-308 is 2.2×10^{-308}

Error Processing

Errors related to floating-point arithmetic are reported via 'events' in addition to the normal C method of setting external variables (such as 'math_errno').

Error Types

Where argument value ranges are stated explicitly in a function definition, during execution the function tests the range of its arguments to ensure they fit in the appropriate range. It reports a 'domain error' if a value is out of range.

When a function makes calculations or uses when arithmetic operators, and the result obtained exceeds the range of the data type, it reports a 'range error'.

Internal Processing when Errors Occur

For a function to report 'domain' or 'range' errors, set an external variable with the appropriate error code. In the case of the result of a calculation, the result is set to an unsigned infinite number value so that calculation can continue as far as possible (see table, below).

The function returns a NaN (Not a Number) value with a division by zero error.

	Float Data Type	Double Data Type
Positive infinity	0x7F800000	0x7FF0000000000000
Negative infinity	0xFF800000	0xFFF0000000000000
Positive NaN	0x7FFFFFFF	0x7FFFFFFFFFFFFFFF
Negative NaN	0xFFFFFFFF	0xFFFFFFFFFFFFFFFF

(Note that NaN is a bit pattern reserved so that the arithmetic subroutine can report an abnormality. The value cannot be assigned to variables.)

Error Variables

The mathematical functions use an external variable, math_errno to indicate error codes. An 'extern' declaration can be found in thelibps.h file for this variable.

The variable is initially zero, but when an error occurs, it is set to the value indicated by the macros EDOM or ERANGE (both of which are defined in the sys/errno.h file), depending on the type of error. Note that math_errno does not automatically reset itself, so you should explicitly set it to zero once your error handling is complete.

Functions Supported

Function	Description
pow	x^y - X to the power of Y
exp	exponential
log	ln(x) - natural logarithm of X
log10	$\log_{10}(x)$ - base 10 logarithm
floor	largest integer not greater than X
ceil	smallest integer not smaller than X
fmod	floating-point remainder of x/y, with the same sign as X
modf	splits X into integral and fractional parts
sin	sine
cos	cosine
tan	tangent
asin	$\sin^{-1}(x)$ - arcsine
acos	$\cos^{-1}(x)$ - arccosine
atan	$\tan^{-1}(x)$ - arctangent
atan2	$\tan^{-1}(y/x)$ - arctangent
sinh	hyperbolic sine
cosh	hyperbolic cosine
tanh	hyperbolic tangent
sqrt	square root
hypot	absolute value of a complex number
ldexp	$X \times 2^n$
frexp	splits X into a normalised fraction in the interval [½,1]
fabs	absolute value (macro)
printf2	formatted output to the console (supports 'float' and 'double' type arguments)
sprintf2	formatted output to an array (supports 'float' and 'double' type arguments)

10

Kernel Management

Management of PlayStation hardware, including the CPU takes the form of C language functions which jump to the kernel proper. These are as follows.

- Root counter control
- I/O control
- Module control
- Additional services

Root Counter Control

Counter functions are indispensable to game programs as they are used for time period management and timing adjustment. The 'root counter' is a single 16-bit counter that increases the count every 8 cycles of the system clock (about 0.24 micro seconds). Obtain the count value from within applications using the GetRCnt() funtion.

Counting

The root counter uses hardware to increase the count. As a result, counting continues regardless of software operations such as interrupt inhibitions.

Target Values

You can set a target value in the counter which, when reached clears the counter to zero and then continues counting.

Macros

Use the RCntCNT2 macro to specify root counters and macros, RCntMdNOINTR | RCntMdSP to specify a root counters operation. The root counter is notguaranteedto work with any other macros.

State Immediately After Activation

As a counter is either stopped or running freely after activation, initialise it using StartRCnt().

I/O Control

This service supports input from and output to files. The data structures and macros used in the I/O control service are defined in the 'sys/file.h' file.

I/O Control deals with all PlayStation file management. Net Yaroze has the usual file access functions, such as open() and close() as well as others for file searching, firstfile(), nextfile(), for example.

Block Size

Each device has a 'block size', which is its characteristic access data unit.

All data access is in multiples of this size, any fractional part of a value (i.e. a part of a value with does not reach the block size) is removed.

The Standard I/O Stream

File descriptors No. 0 and No. 1 are both treated as the standard I/O stream.

Immediately after activating the Net Yaroze system, serial device 'tty', which inputs and outputs one character at a time via the serial interface, is assigned to the standard I/O stream.

The Memory Card Driver (bu)

As memory cards are partitioned and used by more than one application, the only access to them is via the file system. Given this, the Memory card device 'bu' supports file searching functions. Deatils of 'bu' are listed in the table, below.

Item	Description
Device Name	bu
Physical devices	bu00: Memory card in slot 1 bu10: Memory card in slot 2
Functions supported	open, close, read, write, firstfile, nextfile, delete, rename, format
Block size	128 bytes

Table: The Memory Card Driver

Item	Description
Device name	buXY:filename 'filename' is a 21 character ASCIZ character string There is a colon (':') between 'buXY' and 'filename' 'XY' specifies the insertion slot Characters that can be used are the upper case English letters, numbers and the character '_'.
Directory	not used
Number of files	Between 1 and 15 for each card

File size	Multiples of 8K
	Specified at file creation. No automatic expansion using write().
	Example of specification: (this creates a 24K file)
	long size = 3;
	long fd = open(fname,O_CREAT|(size<<16));
	close(fd); /* always close immediately after creation */

Table: The Memory Card File System

The Serial Driver (tty)

This driver forms connection between your PC and Net Yaroze PlayStation that is essential to the Net Yaroze system. The ioctl() function sets the transmission speed.

Item	Description
Device name	tty
Physical device	serial interface
Functions supported	open, close, read, write, ioctl
Block size	1 byte

Table: The Serial Driver

Module Control Service

This is a basic service to load and execute modules.

Executable Files

The PlayStation executable file format is called the 'PS-X EXE' format, and all applications operating on the PlayStation must conform to this format. Executable files contain the information listed below.

Information within Executable Files

(a) Code and data linked to fixed addresses

(b) The execution start address

(c) gp register initial value

(d) The head address and size of the data area for data without initial values

Layout within Executable Files

Offset from Head of File	Block name	Contents	Size
0 bytes	header	(b), (c) & (d) in the list above	2048 bytes
2048	body of the program	(a) in the list above text section + data section for data with initial values	multiple of 2048 bytes

Table: Layout within Executable Files

Executable File Information Data Structure

The group of functions related to executable file management,(Load() and Exec()) operate on the basis of the information contained within the file. Use the Exec structure to access information within an executable file.

EXEC has the following structure.

```
struct EXEC {                       /* executable file information */
        unsigned long pc0;          /* execution start address */
        unsigned long gp0;          /* gp register initial value */
        unsigned long t_addr;       /* head address of text section
                                    + data section for data with initial values */
        unsigned long t_size;       /* size of text section
                                    + data section for data with initial values */
        unsigned long d_addr;       /* reserved for the system */
        unsigned long d_size;       /* reserved for the system */
        unsigned long b_addr;       /* head address of the data section for data
                                    without initial values */
        unsigned long b_size;       /* size of the data section for data without
                                    initial values */
        unsigned long s_addr;       /* stack area head address {for user
                                    specification}*/
        unsigned long s_size;       /* stack area size {for user specification}*/
        unsigned long sp,fp,gp,ret,base;    /* register evacuation address */
};
```

Handling Executable Files

The CdReadExec() function loads executable files from the PlayStation's CD-ROM disk. Use Exec() to then execute that file.

Additional Services

- InitHeap()

  InitHeap() initialises the heap area. You need to initialise the heap area before using the memory allocation function, malloc(). Usually, however, InitHeap() is called automatically, before main() is

executed so you can use malloc(). Only execute InitHeap() specifically when you are changing the heap area.

- FlushCache()
  This function flushes the R3000 I cache. Always execute this between reading in an executable file (using CdReadExec(), for example) and executing the file with Exec().

The Heap Area

This is a memory area dynamically controlled using malloc() and free(). A fixed area is allocated from within the start-up routine, based on the size of the application program.

11

CD-ROM Management

The CD-ROM management service deals with, among other things, reading files from the CD-ROM and playing CD-DA.

CD-ROM

Sectors

Digital data is recorded in a spiral pattern on the surface of a CD-ROM, just as on an audio CD. This digital data is controlled in processing units called sectors. One second's worth of digital area is divided into 75 sectors.

Each sector can be classified as an audio sector, a data sector, or an ADPCM sector, depending on what it is used for

Audio Sectors

The data in the audio sectors is recorded at 44.1kHz as digital stereo audio data (this is the normal speed for CD audio data). Reproduce audio sectors using the CdPlay() function. The PlayStation cannot read in these sectors as data, they go straight to audio output.

Data Sectors

These store user data. There are slight differences in the effective user area in a data sector depending on the which mode format you use, but the standard tends to be 2048 bytes (mode-1 format).

ADPCM Sectors

'ADPCM sectors' refers to sectors properly called 'real time sectors' or 'mode-2 form-2' sectors. ADPCM compressed voice data is stored in these sectors. These can be reproduced as voices in the same way as audio sectors. However, Net Yaroze does not handle ADPCM sectors.

Transmission Rate

The PlayStation CD-ROM, can rotate either at the standard speed or at double speed.

The standard speed is the same rotation rate as is used by ordinary CD players with a transmission rate of 150KB/sec, while double speed is twice that at 300KB/sec.

This means that in one second at standard speed, the PlayStation reads 75 sectors while at double speed, it reads 150 sectors in one second.

The File System

The PlayStation CD-ROM employs an ISO-9660 level 1 standard file system. The details of the file system are as follows.

Item	Content
File format	basename.ext;version 'basename' is up to 8 characters, 'ext' is up to 3 characters. Between 'basename' and 'ext' there is a '.' (period). Between 'ext' and 'version' there is a ';' (semicolon). Characters that can be used are the upper case English letters, numbers and the character '_'.
Directory format	basename 'basename' is up to 8 characters. No extension is possible. Characters that can be used are the upper case English letters, numbers and the character '_'.
Directory levels	Maximum 8 levels. The root directory has no name.
File arrangement	All sectors in a file are physically arranged in a consecutive series.
Block size	2K

Table: The CD-ROM File System (ISO-96601 level 1 standard)

However, the PlayStation only supports lists of files and directories that can be stored in one sector (2048 bytes). As a result, there are certain limiting criteria inherent in PlayStation. These are as follows.

Total number of directories	up to 45
Total no. of files per directory	up to 30

Table: Inherent PlayStation Limits in Relation to CD-ROMs

* The file and directory control data structure in ISO-9660 is variable length, so if many of the names are short, the number of directories and files that you can use is somewhat larger than stated above.

File Access

File Searching

To search for the location of the head of a file via the ISO-9660 file system, use the CdSearchFile() function. CdSearchFile() searches for the location of the file head from the absolute path of the file. The results of the search are stored in the CdlFILE structure, which has the following construction.

```
typedef struct {
    CdlLOC  pos;            /* file position */
    unsigned long  size;    /* file size */
    char    name[16];       /* file name (body) */
} CdlFILE;
```

In addition, the CdlLOC structure, which expresses the file position, has the following construction.

```
typedef struct {
    unsigned char minute;  /* minute (BCD) */
    unsigned char second;  /* second (BCD) */
    unsigned char sector;  /* sector (BCD) */
    unsigned char track;   /* track (void) */
} CdlLOC;
```

Reading Data Files

Use CdReadFile() to read data files from a CD-ROM. The following is an example of reading a data file from a disk.

```
CdlFILE fp;
if (CdSearchFile(&fp, "\PSX\SAMPLE\RCUBE.TIM") != 0)
    CdReadFile(fp, sectbuf, 2048);
```

Note that CdReadFile() executes asynchronously, and is a 'non-blocking' function that returns immediately. CdReadSync() detects the actual termination of execution.

Reading Executable Files

The CdReadExec() function loads executable files from the PlayStation's CD-ROM disk. Use Exec() (Kernal management function) to then execute that file.

Note that CdReadExec() executes asynchronously, and is a 'nonblocking' function that returns immediately. Use CdReadSync() is to detect the actual termination of execution.

Read Synchronisation

CdReadFile() and CdReadExec() execute asynchronously, and are 'non-blocking' functions that return immediately. Use the CdReadSync() function is used to detect the actual termination of execution of CdReadFile() and CdReadExec(). The CdReadSync() function returns the number of remaining sectors not yet read.

Reproducing CD-DA Audio Data

CdPlay() function reproduces audio from the CD-Rom, direct to audio output. .

12

Peripheral Devices Management

The peripheral devices service manages standard PlayStation peripherals, which, in the Net Yaroze system, includes Controllers and Memory cards.

Controller Management

Controllers are important devices that communicate the player's wishes to the application. Two Controllers can be connected to the PlayStation (eight when a Multi tap is used). In addition to the standard Controller, the PlayStation can handle other classes of Controllers, such as the Mouse, neGcon, Analog joystick and light gun. Net Yaroze supports up to two Controllers and ONLY the following classes of Controller; standard Controller, Mouse, neGcon and Analog joystick. (Note that Net Yaroze does NOT support Multi tap or light gun.)

Reading in Information From Controllers

The Net Yaroze library provides the GetPadBuf() function as a service for receiving input from Controllers.

Communication with Controllers is carried out at each vertical synchronisation interruption (V-blank), the result stored in buffers within the system. Pointers to these buffers can be initially obtained using GetPadBuf().

There are two Controller buffers corresponding to the two connection sockets, and the following data is stored in these buffers.

Bytes from the Head	Content
0	0xff: no Controller, 0x00: Controller connected
1	Upper 4 bits: terminal type Lower 4 bits: size of received data (1/2 the number of bytes)
2~	Received data (maximum 32 bytes)

Table: Receiving Buffer Data Format (1)

Controller Types and Data Received

In the Net Yaroze library, the Mouse, NegCon, and Analog joystick are supported as Controller types, in addition to standard Controllers.

The content of the buffers, which can be obtained using GetPadBuf(), differs depending on the category of Controller used, and has the following structure.

Terminal Type	Device Name	Byte	Content
0x1	Mouse	2nd byte	Not used
		3rd byte	2nd bit: right button 3rd bit: left button Bit values: 1 - button release, 0 - button push
		4th byte	Movement in X direction -128~127
		5th byte	Movement in Y direction -128~127
		6th~7th byte	Not used
0x2	neGcon Controller	2nd~3rd byte (a single 16-bit group)	Value of each bit: 1 - button release, 0 - button push Refer to the Net Yaroze Web site for button bit locations.
		4th byte	Analogue channel values -128~127
		5th~7th byte	Analogue channel values - 0~255
0x4	Standard Controller	2nd~3rd byte (a single 16-bit group)	Value of each bit: 1 - button release, 0 - button push Refer to figure, below, for button bit locations.
0x5	Analog joystick	2nd~3rd byte (a single 16-bit group)	Value of each bit: 1 - button release, 0 - button push Refer to the Net Yaroze Web site for button bit locations
		4th~7th byte	Analogue channel values - 0~255

Table: Receiving Buffer Data Format (2)

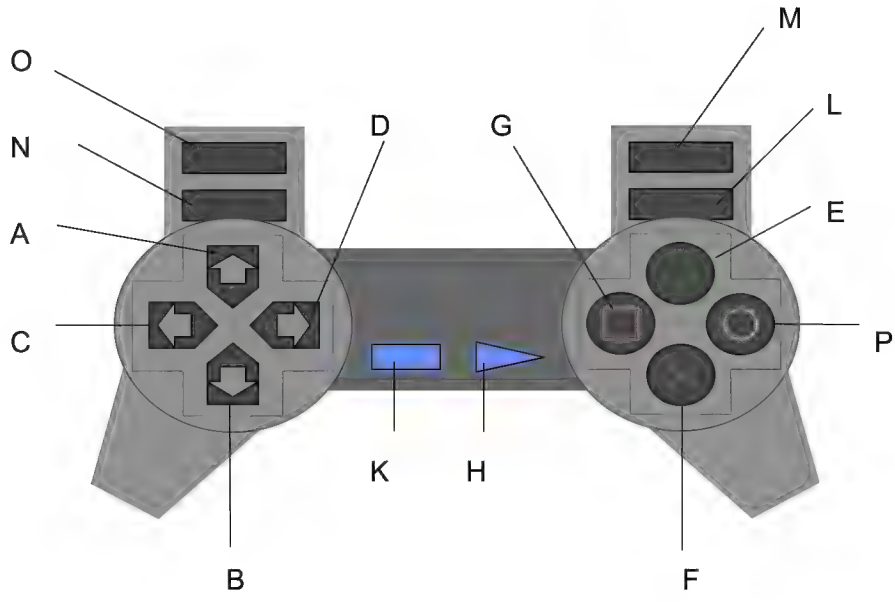


Figure: Bit assignments for the Standard Controller

Bit No.	Corresponding Button	Macro
15	C	PADLleft
14	B	PADLdown
13	D	PADLright
12	A	PADLup
11	H	PADstart
10		
9		
8	K	PADselect
7	G	PADRleft
6	F	PADRdown
5	P	PADRright
4	E	PADRup
3	L	PADR1
2	M	PADR2
1	N	PADL1
0	O	PADL2

Table: Standard Controller Bit Assignment

Memory Card Management

The Net Yaroze library supports the function, TestCard(), which checks for a of a Memory card inserted in the PlayStation. For details, refer to theLibrary Referencemanual.

13

Creating PlayStation Applications

This chapter presents an overview of process of creating a PlayStation application using the Net Yaroze system.

An application is composed of data and code.

- Code is the program that runs on the machine. The code is written using the standard GNU C development environment.
- Data is the 3D models, bitmaps and sound data that the code uses to generate application's output. While the PlayStation uses a set of proprietary dedicated file formats for this data, it can be converted easily from a wide range of commercial file formats.

Creating Data

There are three types of data used by a Net Yaroze PlayStation application. These are 2d graphics (often referred to as bitmaps), 3D graphics (often referred to as models) and sound data. The fourth type of data, for movies (often referred to as Streaming or Full Motion Video), is not available for use with the Net Yaroze system.

2D Graphics Data

With the PlayStation system 2D graphics data (bitmaps) provide the imagery for sprites and textures for 3D graphics. PlayStation uses TIM format. Take ordinary bitmaps, created using any of a wide range of Windows or Macintosh painting tools, and turn the into TIM files using the Net Yaroze tool: TIMUTIL For details on TIM data, refer to the Net Yaroze Members' Web site.

File Formats Supported

In the Net Yaroze system supports conversions to TIM format from the following file formats.

- Windows bit-mapped format (BMP)
- Macintosh PICT format (PICT)
- RGB format (RGB)

Tools

In the Net Yaroze system, the timutil tool (timutil.exe), running on Windows, converts files to TIM from existing formats.

For details of how to use this tool, refer to the Graphics Tools chapter.

Data Creation and Conversion

Use a painting tool which outputs data in one of the 2D file formats supported by the Net Yaroze system (BMP, PICT, RGB).

Data Verification

The Net Yaroze system has a TIM viewer (timv.bat), running on DOS, which operates on the Net Yaroze PlayStation. Using this, you can check the contents of a TIM file.

For details of how to use this tool, refer to the Graphics Tools chapter.

2D Tools List

Tool	Environment	Functions
timutil.exe	Windows	Converts data from BMP, PICT, RGB to TIM
timv.bat	MS-DOS	Displays TIM data on the PlayStation

Table: 2D Tools

3D Graphics Data

In the Net Yaroze system, there are a series of 3D data conversion tools. These convert models from the DXF format (a standard 3D modelling format) to an intermediate file format, the RSD format and then to the PlayStation TMD format. The intermediate RSD format is an ASCII based file which you can edit using a text editor. The final TMD file is a binary format which can be operated upon directly by the Net Yaroze library functions.

For details regarding RSD and TMD data, refer to the Net Yaroze Members' Web site.

File Formats Supported

DXF format is supported by virtually all commercial 3D modelling tools.

For details of the DXF format, refer to theAutoCad Reference Manualpublished by AudoDesk Ltd.

Tools

The following tools convert files from existing formats.

For details of how to use each of these tools, refer to theGraphics Tools chapter.

Tool	Environment	Functions
dxf2rsd.exe	MS-DOS	Converts DXF files to RSD files
dxf2rsdw.exe	Windows	Converts DXF files to RSD files
rsdlink.exe	MS-DOS	Converts RSD files to TMD files
rsd2dxf.exe	MS-DOS	Converts RSD files to DXF files
rsdcat.exe	MS-DOS	Links several RSD files into a single RSD
rsdform.exe	MS-DOS	Transformation and movement of RSD models

Table: 3D Graphic Tools

Data Creation and Conversion

Use a 3D modelling package which supports the DXF file format which you can convert, using the tools provided, into RSD format.

RSD is an artist oriented format mainly used to carry out texture mapping. It is composed of four different file types, as listed in the table, below.

File name	Content
*.RSD	File connection information
*.PLY	Polygon information
*.MAT	Material information
*.GRP	Group information

These files are all text files, which you can edit using a standard text editor.

The sequence of conversion must be DXT->RSD->TMD, whether the files need editing or not.

Data Verification

The Net Yaroze system has an RSD viewer which operates on the Net Yaroze which allows you to preview how a 3D model will be displayed and regenerated within the finished application.

For details of how to use this tool, refer to the Graphic Tools chapter.

Tool	Environment	Functions
rsdv.bat	MS-DOS	displays RSD data on the PlayStation

Table: 3D Graphic Data Verification Tools

Creating Sound Data

Sound data in the Net Yaroze system is divided into two types: the VAB and the SEQ format.

- VAB format files hold samples - these may include sound effects (explosions, for example) or music.
- SEQ format files store score data - information on when and how to play the samples in from the VAB file to replay a piece of music.

Create sound data in standard sound formats using standard sound tools and use the conversion tools supplied with the Net Yaroze system to convert them to VAB and SEQ format files.

File Formats Supported

In the Net Yaroze system, conversion of the following existing file formats into SEQ and VAB data is supported.

- Standard MIDI file format 1 format (SMF)
- AIFF (or alternatively 16-bit straight PCM waveform) - (formats for sample data)

Tools

Use the following tools to convert sound files existing formats.

For details of how to use each of these tools, refer to the Sound Tools chapter .

Tool	Environment	Functions
smf2seq.exe	MS-DOS	Converts standard MIDI files into SEQ data
aiff2vag.exe	MS-DOS	Converts AIFF or 16-bit straight PCM into VAG format
mkvab.exe	MS-DOS	Creates VAB data based on VAG data and attribute definition files
vabsplit.exe	MS-DOS	Divides VAB data into VH data and VB data

Table: Sound Tools

Data Creation and Conversion

Create sound data using commercial sequencing software or waveform editing software which can output data in one of the file formats supported by the Net Yaroze system.

Data Verification

In the Net Yaroze system has sound players which operate on the Net Yaroze PlayStation. Using this, you can check how converted sound data will reproduced and played form with in the finished application.

For details of how to use each of these tools, refer to the Sound Tools chapter.

Tool	Environment	Functions
sndplay.bat	MS-DOS	Reproduces SEQ data on the PlayStation
vabplay.bat	MS-DOS	Reproduces VAB data on the PlayStation

Table: Data Verification Tools

Linking Data with Programming Tools

Once in the appropriate PlayStation format (SEQ or VAB), sound files need no further processing. Thus development is rapid and efficient.

The Flow of Program Creation

The code part (program) of a Net Yaroze application is written in the computer language C using the industry standard GNU C development environment. This is a simple explanation of that process (See the Net Yaroze Additional Reading List at the end of theStart Up Guidefor further information.).

Creating Code

1. Write c code in a text file (called the source code and named 'name.c') using a text editor.
2. Compile the code. That is, convert the code into the native language of the PlayStation (often called object or machine code and named 'name.obj') using a special Compiler program.
3. Link the compiled program to the Net Yaroze library functions using the Linker. This produces the program (also known as the 'application') that you can run or 'execute' (called the executable and named 'name.exe') that can be run.
4. Run the executable code by downloading it into the PlayStation using the SICONS program.

This process is iterative. Often the compiler will find mistakes in the code - that is syntax errors. Which you have to find and correct before successfully recompiling and then linking the code.

Even syntactically correct compiled and linked code may not run as it is meant to. For example, it may run for a time and then stop (called a 'crash') or it may simply not do what the programmer intended. In this case, you need to edit source code to change the defective parts of the code and compile, linke and execute it again. These defects are often called 'bugs', the process of finding and removing them called 'debugging'. Programmers often use a piece of software called a 'debugger' to help them in this process. The Net Yaroze development environment has a debugger.

Makefile

Typically the process, options and sequence of operations for compiling and linking are quite complex. Also, as a complete program may consist of several files, changes in one file may require the recompilation of others. A makefile can simplify this process A makefile is similar to a batch file in DOS or a project file in other C development environments. A makefile is interpreted by the make utility.

Making a Library of Useful Routines

If you have a set of functions which perform a certain task, you may want to reuse them. The most efficient way to so this is to convert them into a library so any application programs can use them. This reduces development time and makes code cleaner and easier to read.

In the Net Yaroze system, the ar utility is a librarian and nm is an object symbol control tool.

Creating a Release Version

When you have successfully debugged the executable code, you need to remove any debug information from it to create a release version. The utility, 'strip', removes symbol information (used by the debugger) from the executable program.

Tools

Tool	Environment	Functions
gcc.exe	MS-DOS	Compiles .c (source code) files into .obj (object) files
		Links object files together to make an executable program
ld.exe *	MS-DOS	Links object files together to make an executable program
siocons.exe	MS-DOS	Serial console program
gdb.exe	MS-DOS	Debugger
strip.exe	MS-DOS	Removes symbol information
make.exe	MS-DOS	Interprets makefiles

* gcc (GNU compiler and linker) is sufficient to create the executable file (it can compile and link), although you may prefer to use the dedicated linker tool, ld.

14

Graphic Tools

The graphics tools consist of data converters and previewers. The data converters convert files created by commercial tools (.bmp, pict and dxf files for example) into PlayStation format data. The list below shows the types of dedicated graphic data used in the PlayStation. For details of each of the formats, refer to the Net YarozeWeb site.

- RSD data3D object data (initial conversion for artists)
- TMD data3D object data (second conversion for programmers)
- TIM data Image data (2D)

Using a PlayStation and TV monitor, you can verify conversions on your PC via previewers which operate under Windows or MS-DOS. This chapter briefly describes the functions of the main tools.

dxf2rsd, dxf2rsdw

These tools convert the standard file format for 3D objects (DXF format) into PlayStation format, RSD. Dfx2rsd is a DOS tool, dfx2rsdw is the windows version of the same tool.

rsdcat

This command combines multiple RSD files into a single RSD file.

rsdform

This command makes simple editing of RSD data (operations such as moving, expanding, and contracting objects).

rsdlink

This command converts RSD data into TMD data.

As TMD format is the data format used by PlayStation programs, PlayStation applications can process TMD files directly using the graphics library without further conversions.

rsdv

This is an RSD data previewer, displaying RSD data in a TV monitor using a PlayStation.

timv

This is a TIM data previewer, displaying TIM data in a TV monitor using a PlayStation.

timutil

This tool converts image data between the following formats. (Ordinary image data can be converted into TIM data using this tool.)

- Windows BMP
- Macintosh PICT
- Ordinary RGB
- PlayStation TIM

dxf2rsd.exe

This tool converts DXF files into PlayStation 3D object data files.

Usage

```
dxf2rsd [options] DXF-files
```

When a DXF file is supplied as the argument, the following four files are created:

- An RSD file (*.rsd),
- a polygon file (*.ply)
- a material file (*.mat)
- a group file (*.grp).

Note: You can: use wildcards in the argument, convert more than one file at a time, omit the file extension '.dxf'.

Options

-o targetname

Specifies the name for the RSD output file (where 'targetname' is the specified file name). The default is the input filename minus the .dxf extension in the current directory.

-col r g b
Specifies the colour of the model as a whole using RGB values (each in the range of 0-255). The default setting is grey (200 200 200).

-cf color-file
Specifies a colour table file which defines colours (where 'color-file' is the file name). This is almost always used in conjunction with the -cl option, detailed below.

-cl
Outputs a list of undefined colours to standard output. Also arranges polygons of the same colour and outputs them into the MAT file. The default is OFF. (See Example 2, below.)

-info
Displays to standard output information regarding the DXF input file. This gives the approximate size and number of polygons in the file. With this option no conversion is carried out.

-max n-poly
Specifies the maximum number of polygons that can be converted (where n-poly is the specified number). The default is 10000.

-quad or -quad1
Division of 4 vertex 3DFACE polygons into triangles is not carried out. This option can reduce the number of polygons in the model. The default setting is OFF.

-quad2 (threshold-value)
Adjacent pairs of triangles are formed into single quadrilaterals. The optional argument (here, 'threshold-value') must be a decimal number between 0.0 and 90.0. This controls the pairs of triangles that are combined by specifying a maximum angle of orientation difference (normal line vectors) between the triangles that can be combined. Any pair of triangles with an angle of orientation greater than the 'threshold-value' are not combined.

When the argument is 0.0, only triangles with exactly the same normal line vectors will be combined, when the argument is 10.0, a difference in orientation of up to 10 degrees is allowed. The default is 1.0. (See Example 4,below.)

-quad3
Triangles are created as quadrilaterals in which the 3rd and 4th vertices are identical to each other. Using this option all polygons are defined as quadrilaterals.

-s and -g
Smooth (Gouraud) shading is carried out. The default is OFF.

-e distance-value
Reduces the number of polygons by causing all spheres with a specified radius (here, 'distance-value') to be regarded as identical. (Note that the radius calculation is carried out after expansion or contraction using the -sc option, listed, below.)

-r
Reduces the number of normal lines by not producing any identical normal line. This is effective with flat shading. The default is OFF.

-n
Normal lines are not created. Use this option when there will be no light source calculation. The default is OFF.

-sc factor
Expands and contracts models by a specified scaling factor (here, 'factor'). Use decimal numbers, the default is 1.0.

-t x y z
Shifts the model left or right, up or down and back or forward ('x', 'y' and 'z') by the values specified. Use decimal numbers, the default is (0.0, 0.0, 0.0).

-auto
Shifts the model to the vicinity of the origin (x, y and z as zero) and expands or contracts the model, as appropriate, so that it fits within a cube with sides of 1000. The default is OFF.

-back
Reverses the normal lines of all polygons. The default is OFF.

-both
Creates all polygons as double-sided. The default is OFF.

-dup
Creates front and back polygons for each polygon, doubling the number of polygons. The default is OFF.

-nopl
Ignores POLYLINE data, converts 3DFACE data. The default is OFF.

-Y+Z, +Y-Z, +Y+Z, -Z-Y, -Z+Y, +Z-Y, +Z+Y
Specifies the co-ordinate system conversion method. This option specifies the labelling and direction of the co-ordinate axes which extend towards the viewer (here, the first value, the forward axis) and upwards (here, the second value, the up axis) as if you are looking at the modeller co-ordinate system from the front.

For example, '-Y+Z' indicates that the forward axis is the negative Y axis, and the up axis is the positive Z axis. The co-ordinate system referred to here is the co-ordinate system of the DXF file, which does not necessarily coincide with the physical screen of the modeller. The default is '-Y+Z'. This co-ordinate system is converted into the PlayStation co-ordinate system (-Z-Y) by the dxf2rsd tool. (In the PlayStation co-ordinate system the forward axis is the negative Z axis and the up axis is the negative Y axis.)

-v
Outputs to the output detailed information concerning the conversion. (See Example 1.)

Restrictions

The present package has the following restrictions.

- Only ASCII format DXF files are supported.
- Out of the DXF entities, only 3DFACE data and POLYLINE data are supported.
- Sometimes large POLYLINE polygons cannot be converted. Wherever possible use a modeller to convert POLYLINE data into 3DFACE data (triangles and quadrilaterals) before creating the DXF file.
- Occasionally quadrilaterals that have vertices which are not all on the same plane are not displayed correctly.
- The number of polygons that can be converted is influenced by the number of vertices created and the number of normal lines. As a general rule, think of this number as about 5000.

Supplementary Notes

- 3DFACE and POLYLINE are both data formats used to express DXF polygons. 3DFACE data represents single polygons (triangles and quadrilaterals) using four vertices, while POLYLINE data represents more than one polygon using connected line segments.
  For a given model, 3DFACE format files tend to be larger in terms of amount of data than POLYLINE, but also have superior compatibility. By contrast, using POLYLINE data reduces the amount of data but the degree of freedom of expression is large, and at times data cannot be successfully exchanged between different modellers. 3DFACE data can be directly converted into RSD data, but with POLYLINE data, triangulation must be carried out first. Sometimes triangulation is not successful (in which case a 'Fail to triangulate!' error message appears). Furthermore, even when triangulation is completed successfully, sometimes with POLYLINE data the orientation of some of the polygons ends up reversed. However, POLYLINE data created using the 3D Studio software package, referred to as 'POLYFACE MESH' data, is equivalent to 3DFACE data and thus have no conversion problems.

- The maximum number of polygons that can be realistically moved about as a single object on the PlayStation is about 2000. Use this number as a guide when creating model data.
- When conversion cannot be carried out using flat shadingdue to the large number of polygons involved, it may be possible to carry out conversion if you specify Gouraud shading.
- Each time the Y and Z co-ordinate axes are exchanged, or the positive and negative directions of each axis are switched, the front and back surfaces of the polygons are reversed.
- Depending on the modeller used, polygons may sometimes end up back to front, even when the data has been output as 3DFACE data. If all of the polygons are reversed, either change the co-ordinate system (e.g. +Y+Z), or use the '-back' option. When only some polygons are reversed, correct them using a modeller.
- Use modellers to create DXF data for conversion using dxf2rsd that can: output the whole model as 3DFACE DXF data, and reverse polygons individually. (However, data fromsome modellers can be converted even if these conditions are not met. Also, even when data cannot be converted directly, alter the DXF file via a dxf2rsd tool compatible modeller. - That is, if the DXF file has been made by a dxf2rsd-tool-incompatible modeller, open and re-save it in a modeller that is compatible with the dxf2rsd tool.)
- With large files use the -n option to create data without the normal lines.
- The -r option does not work with Gouraud shading (-s or -g options) and cannot be used with normal line MIMe (the latter because -r changes the the number and order of normal lines).
- The effects of the -quad2 option can be over-ridden. Use a modeller to specify different colours for each of the triangles in a pair and then specify the -cl option with -quad2 at conversion. (For example: `dxf2rsd -quad2 -cl DXF-files` [where 'DXF-files' are the files to be converted].)

Example 1: Example of dxf2rsd Output Using the '-V' Option

```
C:\> dxf2rsd -v -auto +Z+Y -quad -s foo
```

Note: Input file = foo.dxf, Output files =foo.rsd, foo.ply, foo.mat, foo.grp]

Input File (DXF)			Explanation
SIZE	40230 lines		No. of lines in DXF file
VERTEX	4320		No. of vertices in DXF file
POLYGON	1468 (estimate)		Estimated no. of polygons
	3-poly	1376	1376 triangles
	4-poly	32	32 quadrilaterals
	(>9)-poly	2	2 polygons with more than 10 sides
	polylines	2 (max size=32)	2 polyline figures (32 vertices)
RANGE	x: -1.015	+0.785	Min. value ... max value
	y: -2.533	+0.768	for Each axis. Y and Z axes
	z: -1.161	+0.625	are converted to 'PS' if necessary.
SCALE	302.870		Scaling factor
MOVE	(dx,dy,dz) = (34.788,267.255,81.207)		Amount of movement
MATERIAL	0		No. of coloured polygons
Output File (RSD)			Explanation
VERTEX	796		No. of vertices after conversion
POLYGON	1468		No. of polygons after conversion
	triangles	1436	1436 triangles
	quadrangles	32	32 quadrilaterals
RANGE	-272.477	+272.477	Min. value ... max value for
	-500.000	+500.000	axes after conversion
	-270.510	+270.510	
MATERIAL	0		No. of polygons with material
NORMAL	796		No. of normal line vectors created

Table: Example of dxf2rsd Output Using the '-V' Option

Example 2: Using Colour Information

Use the -c option to maintain colour that has been added in the modeller in the RSD file. Polygons in the resulting file will be modelled appropriately, so that parts that are coloured the same in the unconverted file will be allocated the same colour as each other in the RSD file.

Exact colour matching cannot be maintained because the unconverted DXF file only holds colour numbers rather than RGB values. To maintain colours, either edit use a text editor to edit the MAT file after conversion, or specify the colours before conversion by specifying a colour table file, as detailed below.

1. Output a list of undefined colours for the foo.dxf file, piped to a new file, foo.cl.

```
C:\> dxf2rsd -cl foo > foo.cl
```

2. Display the contents of the resulting file. (This is a list of unassigned colours).

```
C:\> edit foo.cl
183
40
253
0
8
```

3. Allocate the colours required against (in a list to the right of) the unassigned colour number as decimal RGB values. (RGB values must be in the range 0-255).

```
C:\> edit foo.cl
183   100 100 200
40     58  20  43
253    10 100  10
0     212  20 100
8       0 128 126
```

4. Run dxf2rsd with -cf option and the foo.cl file specified.

```
C:\> dxf2rsd -cf foo.cl foo
```

The newly created RSD file will have colours allocated in accordance with the colour table file.

Example 3: Converting Large Data

For data that is extremely detailed, use the -e option to make it less detailed by combining several vertices into single vertices . In the following example the number of vertices, the number of polygons and the number of normal lines, are reduced by regarding any two vertices separated by a distance of 100 or less as a single vertex. (Note that the distance is calculated on the basis of the scale after expansion.) Depending on the data, an appropriate distance value can reduce, the volume of data with virtually no change in shape.

Taking the file, big.dxf……

```
C:\> dxf2rsd -v -e 100 -sc 1000 big.dxf
```

Note: Input File = big.dxf, Output file = big.rsd, big.ply, big.mat, big.grp

Input (DXF)				Explanation
SIZE	134628 lines			
VERTEX	18982			
POLYGON	8618 (estimate)			
	3-poly	1746		
	4-poly	3436		
RANGE	x	-1.644	+1.545	Because the scale is
	y	-2.352	+0.000	small, it is
	z	-3.649	+3.993	multiplied by1000
SCALE	10000.000			
MATERIAL	0			
Output (RSD)				**Explanation**
VERTEX	1208			
POLYGON	2708 (68%reduced)			No. of polygons reduced by about 30%
	triangles	2708		
RANGE	x	-643.811	+1545.072	
	y	-2352.365	+0.000	
	z	-3649.154	+3992.687	

Table: Example Converting Large Data

Specify the '-r' option in addition to those above to reduce the number of normal lines.

Example 4: Forming quadrilaterals

The -quad2 option, combines neighbouring pairs of triangles into quadrilaterals. In the following example, all neighbouring triangles with an angle between them of five degrees or less are converted into quadrilaterals.

Taking the file, earth.dxf…

```
C:\> dxf2rsd -v -quad2 5.0 earth
```

Note: Input = earth.dxf, Output = earth.rsd, earth.ply, earth.mat,earth.grp

Input (DXF)				Explanation
SIZE	88158 lines			
VERTEX	8811			
POLYGON	2937 (estimate)			
	3-poly	2937		Originally 2937 triangles
RANGE	x	-4.000	+3.986	
	y	-3.997	+3.997	
	z	-4.000	+4.000	
MATERIAL	0			
Output (RSD)				**Explanation**
VERTEX	2231			
POLYGON	1686			1686 polygons in all
	triangles	43		43 triangles combined
	quadrangles	1251		1251 quadrilaterals formed
MATERIAL	0			
NORMAL	1686 (quad2<4.9870)*(see note below)			

Table: Example Forming Quadrilaterals

*Note: 4.9870 indicates the largest angle between pairs of triangles that were actually converted. This angle is less than or equal to the allowed angle (in this case 5.0 degrees). Given this, running the same dxf file through dxf2rsd and specifying '-quad2 4.986', would result in a smaller number of quadrilaterals formed.

dxf2rsdw.exe

This is a Windows application that converts the DXF format files output by a variety of commercial modellers into RSD format files for use on the PlayStation (the windows version of the DOS tool, dfx2rsd detailed above). This application reads a DXF file, then creates the following four files.

- An RSD file (*.rsd)
- A polygon file (*.ply)
- A material file (*.mat)
- A group file (*.grp)

Usage

Click on dxf2rsdw.exe from the Windows File Manager or Windows Explorer. With the application running....

1. Display the parameter window:
2. Select a DXF file using 'Open...' in the File menu.
3. Change these parameters as required and select either 'Convert...', or 'Save as...' from the File menu. The save file dialogue box is displayed.
4. Specify the name for the converted file in this box and format conversion begins.

Alternatively:

Specify the input file by dragging the DXF file from Windows Explorer or File Manager and dropping it into the 'dxf2rsdw.exe' window.

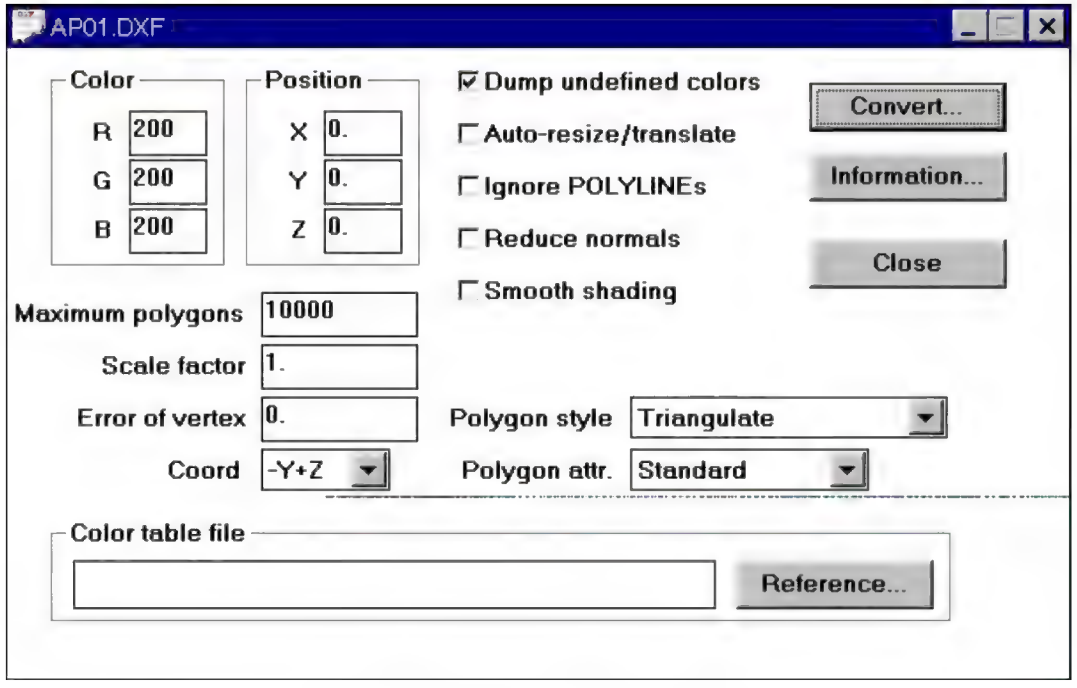


Figure: The Parameter Setting Window

Description of Each Item in the Parameter Setting Window

For each item the corresponding options in the MS-DOS version (dxf2rsd.exe) are noted - see these for an explanation of their function.

Overall Colour

The colour of the model as a whole is specified using RGB values (each being in the range 0-255).

Equivalent to the MS-DOS version '-col r g b' option.

Model Position

Moves the model. Specify the position.

Equivalent to the MS-DOS version '-t x y z' option.

Maximum Number of Polygons

Specifies the maximum number of polygon that can be converted.

Equivalent to the MS-DOS version '-max n-poly' option.

Expansion Ratio

Carries out expansion and contraction of the model. Specify the scaling factor as a real number.

Equivalent to the MS-DOS version '-sc factor' option.

Minimum Vertex Separation

Reduces the number of vertices and the number of polygons. Specify a radius, and all vertices lying within spheres with that given radius are regarded as identical. Note that the distance calculation is carried out after expansion or contraction using Expansion Ratio.

Equivalent to the MS-DOS version '-e distance' option.

Co-ordinate System

Specifies the co-ordinate system conversion method. This option specifies the labelling and direction of the co-ordinate axes which extend towards the viewer (the forward axis) and upwards (the up axis) when looking at the modeller co-ordinate system from the front.

For example, '-Y+Z' indicates that the forward axis is the negative Y axis, and the up axis is the positive Z axis. The co-ordinate system referred to here is the co-ordinate system of the DXF file, which does not necessarily coincide with the physical screen of the modeller. The default is '-Y+Z'. This co-ordinate system is converted into the PlayStation co-ordinate system (-Z-Y) by the dxf2rsd tool. (In the PlayStation co-ordinate system the forward axis is the negative Z axis and the up axis is the negative Y axis.)

Equivalent to each option on the MS-DOS version - see ' -Y+Z, +Y-Z, +Y+Z, -Z-Y, -Z+Y, +Z-Y, +Z+Y' in the DOS version information

Display Undefined Colours

See Notes, Using Undefined Colours below. (The default for this option is different to the MS-DOS version default. The default is ON.)

Equivalent to the MS-DOS version '-cl' option.

Auto Size Adjustment
Shifts the model to the vicinity of the origin and expands or contracts the model to a suitable size (so that it fits within a cube with sides of 1000).

Equivalent to the MS-DOS version '-auto' option.

Ignore POLYLINE
POLYLINE data is ignored and only 3DFACE data converted.

Equivalent to the MS-DOS version '-nopl' option.

Consolidate Normal Lines
Identical normal line vectors are not produced, so the number of normal lines are reduced. Particularly effective with flat shading.

Equivalent to the MS-DOS version '-r' option.

Gouraud Shading
Carries out smooth (Gouraud) shading.

Equivalent to the MS-DOS version '-s' and '-g' options.

Polygon Type
The following choices are available.

'Triangles'
Divides all polygons into triangles.

'Inhibit Triangulation'
Specifies that there should be NO division of 4 vertex 3DFACE polygons into triangles. Using this option the number of polygons in the model as a whole can be reduced.

Equivalent to the MS-DOS version '-quad' option.

'Form Quadrilaterals'
Adjacent pairs of triangles are formed into single quadrilaterals. The optional value, 'threshold-angle' must be a decimal number between 0.0 and 90.0. This controls the pairs of triangles that are combined by specifying a maximum angle of orientation difference (normal line vectors) between the triangles that can be combined. Any pair of triangles with an angle of orientation greater than the 'threshold-value' are not combined.

When the argument is 0.0, only triangles with exactly the same normal line vectors will be combined, when the argument is 10.0, a difference in orientation of up to 10 degrees is allowed. The default is 1.0.

'Quadrilaterals'
Triangles are created as quadrilaterals in which the 3rd and 4th vertices are identical to each other. Using this option all polygons can be defined as quadrilaterals.

Equivalent to the MS-DOS version '-quad2' option.

Polygon Attributes
The following choices are available.

'Standard
Creates polygon and normal line information in the DXF file without modification.

'Invert Normal Lines'
Inverts the normal lines of all polygons.

Equivalent to the MS-DOS version '-back' option.

'Double-Sided Polygons'
Makes all polygons into double-sided polygons.

Equivalent to the MS-DOS version '-both' option.

'Front and Back Polygons'
Backwards facing polygons are created for all polygons.

Equivalent to the MS-DOS version '-dup' option.

'Don't Create Normal Lines'
Normal lines are not created. Use this option when there will be no light source calculation.

When the amount of data is so large that conversion cannot be completed, this option is specified automatically.

Equivalent to the MS-DOS version '-n' option.

Colour Table File
Specifies a colour table file. Specify the file name in the file dialog box. Display the dialog box by clicking 'Browse...'. (SeeNotes, Using Colour Information,below).

Equivalent to the MS-DOS version '-cf color-file' option.

Convert
Carries out format conversion in accordance with the specified parameters.

Enter the name for the converted file in the save file dialog box. The default directory is the current directory.

This function is the same as 'Save As...' option in the File menu.

Information
Displays an information dialog box (shown below) containing details of the input DXF file, including the approximate size and number of polygons.

Equivalent to the MS-DOS version '-info' option.

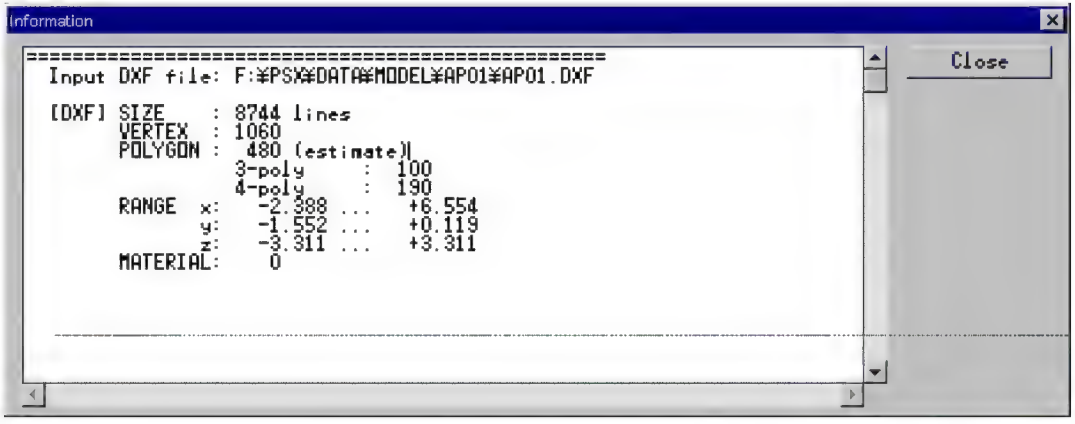


Figure: Information Dialogue Box

Note: To store the information in this dialog box, use the mouse select the required text and copy it to the Windows clipboard using Ctrl+C. Then paste it into any text file, using a suitable editor.

Close
Closes the present window.

This function is the same as 'Close' in the File menu.

Menu Bar

'Open...' Command ([File] menu)

Opens an existing DXF file. In the open file dialog box , specify the DXF file to open.

- The 'Save As...' Command ([File] menu)

Saves the DXF file being worked on under a new name. In the file save dialog box, enter a suitable name for the DXF file that is being worked on and save it. Conversion to RSD format is then carried out in accordance with the specified parameters.

Notes

Using Colour Information

To maintain colour that has been added in the modeller in the RSD file, establish and then specify the undefined colours by converting the same file twice; once the establish the undefined colours and once the specify the colours. Note that when specifying undefined colours, exact colour matching cannot be maintained because the unconverted DXF file only holds colour numbers rather than RGB values.

- Establish the undefined colours

1. Select 'Display Undefined Colours' in the conversion dialog box to get a listing the undefined colours. After conversion an undefined colours list is displayed in the dialog box shown below.

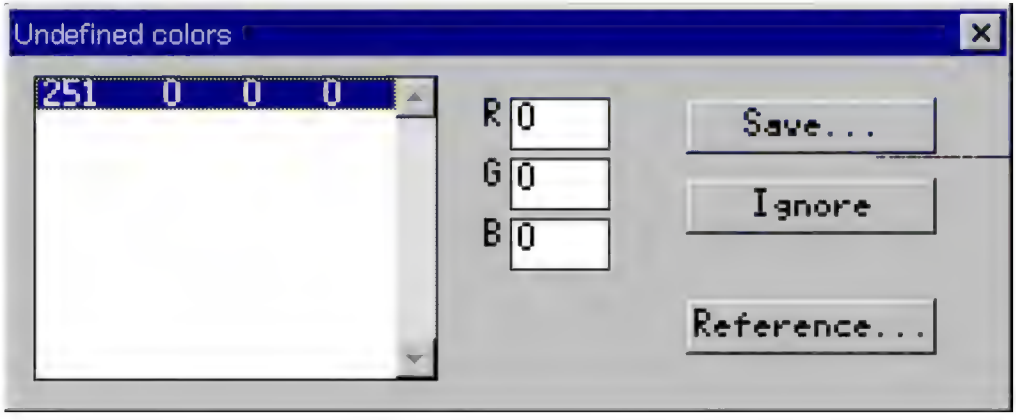


Figure: The Undefined Colours Dialogue Box

- Specify colours

Use the dialog box to specify colours for the list as follows.

1. Select a line in the table and edit the R, G and B fields to specify the required colour (using RGB values of between 0-255).

2. Click 'Save' and specify a name for the resulting colour table file in the file dialog box.

3. Redo the conversion specifying the name of this undefined colour file in the 'Colour Table File' option.

Colours in the newly created RSD file will be allocated in accordance with the colour table file.

For further restrictions and supplementary notes, seedxf2rsd.exe, example 2Using Colour Information.

rsd2dxf.exe

Converts an RSD format file into a DXF format file.

Usage

```
rsd2dxf [options] [ARG-files ...] RSD-files ...
```

The group of RSD format files supplied (name.rsd, name.ply, name.mat, name.grp, where 'name' is the file name) is converted into DXF format. The default file name for the output DXF file is the name of the input RSD file with the .dxf extension replaced with .dxf.

Wildcards can be used in the argument.

Options

-o targetname
Specifies the name for the dxf output file (where 'targetname' is the specified file name). The default is the RSD input filename minus the .rsd extension in the current directory. This option can only be used when there is only one input RSD file.

-r
Inverts and puts back to front all of the polygons in the DXF file created.

-h
Displays a list of all the options that can be specified and simple explanations of how to use these options.

-v
Displays information related to the conversion.

[ARG-files]

Specifies argument files which describe 'rsd2dxf' options and input filenames. Use this option when specifying a command argument exceeding 128 bytes.

Use a text file with the filename extension '.arg' which rsd2dxf interprets as an argument file. In the file, begin comment lines with '#'. '#', ensures that the comments are not interpreted by the tool.

Notes

- In the co-ordinate systems of the DXF files created, +X is to the right, +Y is downwards, and +Z is to the rear. These are the same as the default values taken by 'dxf2rsd' as the co-ordinate system of DXF files. As a result, when the DXF files created are read by a modeller, the model may have to be rotated the to suit the modeller's co-ordinate system.
- Information contained in the RSD files regarding normal lines, colours, and textures is discarded.
- If the output filename already exists, its content is overwrittenBEWARE: there is no warning of file overwriting.
- Argument files cannot be specified inside an argument file (they are ignored).
- Wildcards cannot be used in an argument file.

rsdcat.exe

Joins more than one RSD file into a single file.

Usage

```
rsdcat [options] [ARG-files ...] RSD-files ...
```

The RSD file sets supplied as the argument are joined together, and a new RSD file set created.

Wildcards can be used in the argument.

Options

-o targetname

Specifies the name for the output files (where 'targetname' is the specified file name). Using 'targetname', the resulting files are: targetname.rsd, targetname.ply, targetname.mat, targetname.grp). Default creates files with the names;'out.rsd', 'out.ply', 'out.mat' and 'out.grp' in the current directory.

-h
Displays a list of all the options that can be specified and simple explanations of how to use the options.

-v
Displays information related to the conversion.

[ARG-files]
Specifies argument files which describe 'rsdcat' options and input filenames. Use this option when specifying a command argument exceeding 128 bytes.

Use a text file with the filename extension '.arg' which rsdcat interprets as an argument file. In the file, begin comment lines with '#'. '#', ensures that the comments are not interpreted by the tool.

Notes

- One group is added to the group file (*.grp) for each input RSD file. Each group is given the name of the input rsd file (minus the .rsd extension) and consists of a list of polygons in the rsd input file.
- 'rsdcat' does not carry out compression of textures, vertices, normal lines or groups of information. As a result, the 'rsdcat' output file may need to be manually corrected.
- If the output filename already exists, its content is overwrittenBEWARE: there is no warning of file overwriting.
- Wildcards cannot be used in an argument file.

rsdform.exe

Transforms and moves 3D object data (artist-oriented RSD data).

Usage

```
rsdform [options] RSD-name
```

This command alters the form of objects by applying scaling, rotation and translation to RSD format object data. If more than one transformation operation is specified at the same time, the operations are carried out in the following order: scaling→ rotation → translation.

Specify the RSD file name as either 'file.rsd' or 'file' (without the .rsd extension).

Options

-o targetname
Specifies the name for the output files (where 'targetname' is the specified file name). The output name cannot be the same as the input file name. The default filename is 'a'.

-s x y x
Specifies scaling values for the x, y and z axes. Use 'float' type numerical values. A value of 1.0 leaves the scaling unchanged. Negative values create a reflection with respect to that axis.

-r x y z
Specifies angle of rotation around the x, y and z axes. Use float type numerical values. Positive values indicate clockwise rotation (see Supplementary Notes). The angle units are degrees by default. Specify the -rad option for the angles units as radians.

-t x y z
Specifies translation along the x, y and z axes. Specified as 'float' type numerical values.

-rad
Changes the unit for angle of rotation from the default (degrees) radians.

-l
Fixes the centre of the model for rotation and scaling. Specified together with -s and -r options.

-c

Share files if possible. (SeeSupplementary Notes).

-c[pmg]
Specifies PLY, MAT and GRP files, respectively, should be shared. Specify the files to be using a combination of the three letters - 'p' (PLY file), 'm' (MAT file) and 'g'(GRP file). Certain other options make sharing impossible(see Supplementary Notes). If this occurs there is a warning and no files are shared.

-k
Keeps the 'version' of the original file. Using this option, the result is @RSD940102, @PLY940102, @MAT940801 and @GRP940102.

-quiet
Stops transformation history being added to the output files. The default is OFF.

-v
Outputs detailed information concerning conversion to the standard output.

Supplementary Notes

- With RSD, four files (.rsd, .ply, .mat and .grp) form a single set. These files must be kept in the same directory, normally a dedicated RSD directory of a project directory.
- Sharing of files is possible under the following conditions.
- PLY files: when transformation specification is not carried out at all.
  MAT files: when mirror reflection relative to the original data has not been carried out.
  GRP files: these can usually be shared
  The PlayStation co-ordinate system is 16-bit. While this restriction does not occur with RSD, when RSD data is converted into TMD data using 'rsdlink', the model uses this system.
- 'Clockwise rotation' refers to the direction in which the fingers curl when the thumb of the right hand is extended to point in the positive direction of the axis of rotation.

Example 1: Basic usage

Expands the model by a factor of 10 in the X axis direction, and rotates it 45 degrees around the X axis. Specifies sharing when files can be shared.

Taking the file, cube.rsd.....

```
C:\> rsdform -v -s 10 1 1 -r 45 0 0 -c cube
```

Input RSD file (cube.rsd) Details					Explanation
SCALE		10	1	1	
ROATION		45	0	0	(degree) Content of transformation specification
TRANSLATION		0	0	0	
RANGE	x	+0.0	+1000.0	(centre) +500	Pre-transformation size
	y	-500.0	+500.0	+500	(max, min and central
	z	-500.0	+500.0	+500	co-ordinate values for each axis)
Output RSD file (a.rsd) Details					**Explanation**
FILE TRANSFORMATION		cube.ply	becomes	a.ply	
		cube.mat	becomes	cube.mat (shared)	This file can be shared
		cube.grp	becomes	cube.grp (shared)	This file can be shared
		cube.rsd	becomes	a.rsd	
RANGE	x	+0.0	+10000.0	+5000.0 (centre)	Post-transformation size
	y	-707.1	+707.1	+0.0	(max, min and central

Table: Example rsdform Basic Usage

Example 2: Carrying out Local Transformation

Example 1 with the addition of the -l option. This option doesn't change the position of the model but makes the centre of the model the central point for rotation and scaling.

Compare the results with those obtained in Example 1.

Taking the file, cube.rsd.....

```
C:\> rsdform -v -l -s 10 1 1 -r 45 0 0 -c cube
```

Input RSD file (cube.rsd) Details					Explanation
SCALE		10	1	1	
ROATION		45	0	0	(degree) Content of transformation specification
TRANSLATION		0	0	0	
RANGE	x	+0.0	+1000.0	(centre) +500	Pre-transformation size
	y	-500.0	+500.0	+500	The centre is the same as post transformation size
	z	-500.0	+500.0	+500	
Output RSD file (a.rsd) Details					**Explanation**
FILE TRANSFORMATION		cube.ply	becomes	a.ply	
		cube.mat	becomes	cube.mat (shared)	This file can be shared
		cube.grp	becomes	cube.grp (shared)	This file can be shared
		cube.rsd	becomes	a.rsd	
RANGE	x	-5400.0	+55000.0	+500.0 (centre)	Post-transformation size
	y	-707.1	+707.1	+0.0	The centre is the same as pre-transformation size
	z	-707.1	+707.1	+0.0	

Table: Local Transformation with rsdform

Example 3: Copying RSD files

Use the 'rsdform' command to copy and/or rename RSD files. Run it from batch files for convenience as shown in the example below.

Batch file contents:

```
@ECHO OFF
rsdform -o %2 %1
```

Example 4: Overwriting RSD files

The rsdform tool is designed so that it doesn't overwrite pre-transformation RSD files. However, it can be used to overwrite via batch files as shown below.

Batch file contents:

```
  @ECHO OFF
  SET ARGS=
  REM                                   reads all arguments
  :LOOP1
  IF "%9"=="" GOTO LABEL1
  SET ARGS=%ARGS% %1
  SHIFT
  GOTO LOOP1
  :LABEL1
  REM                                   converts data using the temporary name ' _tmp'
  rsdform -o _tmp %ARGS% %1 %2 %3 %4 %5 %6 %7 %8
  IF ERRORLEVEL 1 GOTO END
  REM                                   searches for the final argument (= input
filename)
  :LOOP2
  IF "%1"=="" GOTO LABEL2
  SHIFT
  GOTO LOOP2
  :LABEL2
  REM                                   overwrites the input file (%0)
  rsdfo rm -o %0 -quiet _tmp
  REM                                   deletes '_tmp.*' and completes the operation
```

```
DEL _tmp.*
:END
SET ARGS=
```

rsdlink.exe

Converts artist-oriented 3D object data files (RSD files) into an object data file (a TMD file) of PlayStation format.

Usage

```
rsdlink [options] rsd-names ...
rsdlink [options] rsd-names [options] rsd-names [options] ....
rsdlink [options] arg-files ...
```

Multiple RSD data files supplied as arguments are linked into a single TMD file. Specify scaling factors and translation values separately for each RSD data file, if required. (See Example 1)

If there are no file paths specified for the required RSD filenames in the argument, rsdlink searches the current directory first, followed by the '.\RSD' directory.

When there are a large number of arguments, supply these in an argument file. (That is a text file, listing the arguments and named with the filename extension: '.arg'). Note that an argument file name cannot be listed in another argument file. For arguments, the filename extension of RSD files ('.rsd') may be omitted, but the filename extension of argument files ('.arg') must not be omitted. (See Example 2)

Options

-o filename
Specifies the output filename. The default is 'a.tmd'

-s factor
Expands and contracts RSD data by a given factor from the next argument on (where 'factor' is the expansion/contaction factor). The specified scaling factor is applied to all RSD files appearing after this option is specified, unless another specification is made.

-sc factor

Rounds the scaling factor to 2 to the power of n (where 'n' is the value specified - here shown as 'factor') . The default is 1.0. The TMD format uses 16-bit integers, so set scaling factor so that the model can be dealt with within this co-ordinate system.

-t x y z

Translates RSD data from the next argument on. The default is (0 0 0). The specified translation is applied to all RSD files appearing after this option is specified, unless another specification is made.

-info

Gives detailed information about the object being converted, such as its type, vertex co-ordinate values and texture information. Details go to standard output. (seeExample 4)

-v

Gives detailed information about the conversion, such as the number of polygons. It also outputs the approximate size of each RSD in the PlayStation co-ordinate system (Range: vertex minimum and maximum values (x, y, z)), so position and size can be confirmed before the model is displayed on the PlayStation. Details go to standard output. (SeeExample 3)

Restrictions

- When the number of polygons in a single RSD data file is extremely large, sometimes linking is not possible. The maximum number of polygons in a file where linking is guaranteed is roughly 5000. However, the maximum number varies depending on the number of vertices and normal lines, and on the memory space available). This limit is only for individual RSD files, there is no maximum number of RSD files that can be linked, or maximum number of polygons in the TMD file created.

Note

- Recommendation: do all translation and scaling in RSD format, using:
  'dxf2rsd(-sc -t)' and 'rsdform(-s -t)'.
  This makes data that is more accurate data and easier to work with.

- Texture files (TIM files) referred to in '.rsd' files must be in the same directory as the RSD files or in the '..\TIM' directory. 'rsdlink' searches for TIM files in this order.

Example 1: Basic usage

```
C:\> rsdlink -v -o boxes.tmd box1 -s 2.0 -t 100 100 100 box1 box2 -t 200 -200
200 box3
```

In this example the following four objects are combined to form a single TMD file ('boxes.tmd')

box1	unscaled object
box1	box1 scaled to twicethe size and translated by (100 100 100)
box2	box2 scaled to twice the size and translated (100 100 100)
box3	box3 scaled to twice the size and translated by (200 -200 200)

Example 2: Collecting the arguments together in a file

When, as in Example 1, the argument is long, combine the elements of the argument into a simple text file saved as an argument file (with the filename extension '.arg'). Then specify this file as the argument.

1. Create the argument file: test.arg as a text file.

```
box1
-s 2.0 -t 100 100 100 box1 box2
-t 200 -200 200 box3
```

2. Use test.arg as an argument.

```
C:\> rsdlink -v -o boxes.tmd test.arg
```

Example 3: Example output with '-v' option

Taking the file, dino.rsd

```
C:\> rsdlink -v dino -s 100 box
```

Output		Details
1 - RSD		1st RSD ("dino.rsd")
RSD files	\PSXGRAPH\DATA\RSD\dino.ply, dino.mat, dino.grp	
TEX[0]	dino0.tim	Texture filenames
TEX[1]	dino1.tim	
TEX[2]	dino2.tim	
TEX[3]	dino3.tim	
TEX[4]	dino4.tim	
TEX[5]	dino5.tim	
POLYGON	2724	no. of polygons
VERTEX	1376	no. of vertices
NORMAL	2671	no. of normal lines
MATERIAL	2592	no. of materials
RANGE	(-180, -210, -1690) - (180, 580, 290)	maximum and minimum range values (x, y, z)

Table (Part 1): Example rsdlink Output

(See part 2, below.)

2 - RSD			2nd RSD ("box.rsd")
RSD FILES	\PSXGRAPH\DATA\RSD\box.ply, box.mat, box.grp		
POLYGON	12		
VERTEX	8		
NORMAL	12		
MATERIAL	1		
SCALE	128		scaling factor : rounded to 2
RANGE	(-6400, -6400, -6400) - (6400, 6400, 6400)		Maximum and minimum range values (x, y, z)
TMD			
OUTPUT TMD	a.tmd		Output filename
TMD HEADER		(12 bytes)	TMD file header size
OBJECTS	2	(56 bytes)	Total no. of RSDs
PRIMITIVES	2736	(65640 bytes)	Total no. of primitives
	12	Flat Coloured Triangles	
	136	Gouraud Coloured Triangles	Breakdown for each mode
	2434	Flat Textured Triangles	
	154	Gouraud Textured Triangles	
VERTICES	1384	(11072 bytes)	Total no. of vertices
NORMALS	2683	(21464 bytes)	Total no. of normal lines
Total File Size 98244 bytes			Output file size

Table (Part 2): Example rsdlink Output

(See part 1, above.)

Example 4: Example of output with '-info' option

Confirmation of the actual contents of the TMD data file created by the conversion can be obtained using the '-info' option.

```
C:\> rsdlink -info box
```

Output		Details
INPUT RSDS	1 object(s)	No. of objects in the TMD file
RSD[0]	"box"	Name of each RSD
TOTAL VERTICES	8	Total no. of vertices
TOTAL NORMALS	12	Total no. of normal lines
TOTAL PRIMITIVES	12	Total no. of primitives
Box		
FLAT TEX 3-POLY(0x24000507) LIGHT: ON = 0		
Vert-0: (-150, -150, -150) (#2)		
Vert-1: (150, -150, -150) (#6)		
Vert-2: (-150, 150, -150) (#0)		
Norm-0: (0, 0, -4096) (#0)		
UV 0-2: (0 0) (47 0) (0 47)		
Pixel mode : 4bit CLUT : (x y)=(0 480)		
Texture Page: 10 Texture No. : 0		
FLAT TEX 3-POLY(0x24000507) LIGHT: ON = 1		

For each primitive, as shown below, the following information is displayed:

- [the polygon number], flat or Gouraud shading (FLAT/GOURAUD),
- two-sided or one-sided polygons (TWO-SIDED), gradation (GRADATION),
- presence of absence of texture (TEX/NO TEX), semi-transparency ON/OFF(TRANS),
- primitive type (3-POLY, 4-POLY, LINE, SPRITE), primitive header hexadecimal display (0x...),

- light source calculation ON/OFF (LIGHT: ON/OFF).

Next, the co-ordinate values of each of the vertices of that primitive are shown as follows.

```
Vert-0: (-150,-150,-150) (#2)
```

'(#...)' being the vertex number used in the PLY file.

Normal lines are displayed in the same way:

```
Norm-0: (0,0,-4096) (#0)
```

(With RSD, normal lines are usually standardised to a size of 1 [in this case (0, 0, -1.0]), but with TMD the values shown above occur because the floating point number 4096 is calculated as having the value 1.)

After that material information such as the UV co-ordinates, colours, etc., of the textures are displayed.

rsdv.bat (RSD Previewer)

The RSD data previewer displays RSD data on a TV monitor using the Net Yaroze PlayStation. It is a DOS batch program.

Usage

```
C:\>rsdv  RSD_file
```

RSD data cannot be directly processed by the PlayStation. Thus the rsd first converts the RSD file specified (here 'rsd_file') into a TMD file using the rsdlink command. It then combines this TMD file and the TIM file(s) into a single file and transmits that file to the PlayStation. Finally, it transmits the RSD previewer program ('rsdview') to the PlayStation, and the previewer displays the file on the PlayStation's TV monitor.

Using the Tool

Use the PlayStation Controller to explore the model. (See Controller functions shown below.)

Note: Before using the rsdv tool, ensure the following:

- the PC and the PlayStation are connected by communications cable DTL-H3050,
- the Net Yaroze PlayStation boot disk is in the Net Yaroze PlayStation,

- the power is switched on.

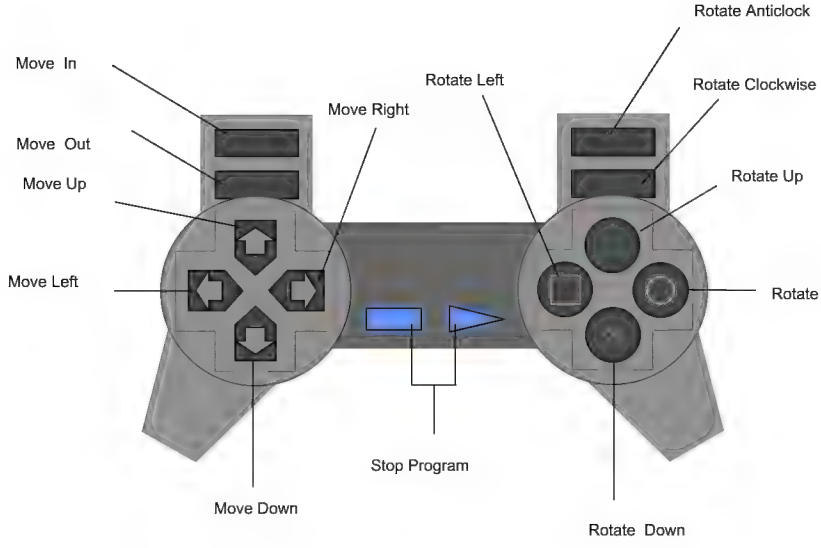


Figure: Controller Operating Method

timutil.exe (TIM utility)

This is a Windows application which converts files between each of the following bitmapped formats: PlayStation TIM, Windows BMP, Macintosh PICT, and ordinary RGB.

Usage

Click on timutil.exe from the Windows File Manager or Windows Explorer. With the application running....

1. Display the parameter window:

   Select a bitmap file using 'Open...' in the File menu.

2. Change these parameters as required and select either 'Convert...', or 'Save as...' from the File menu. The save file dialogue box is displayed.

3. Specify the name for the converted file in this box and format conversion begins.

Alternatively:

Specify the input file by dragging the RSD file from Windows Explorer or File Manager and dropping it into the timutil window.

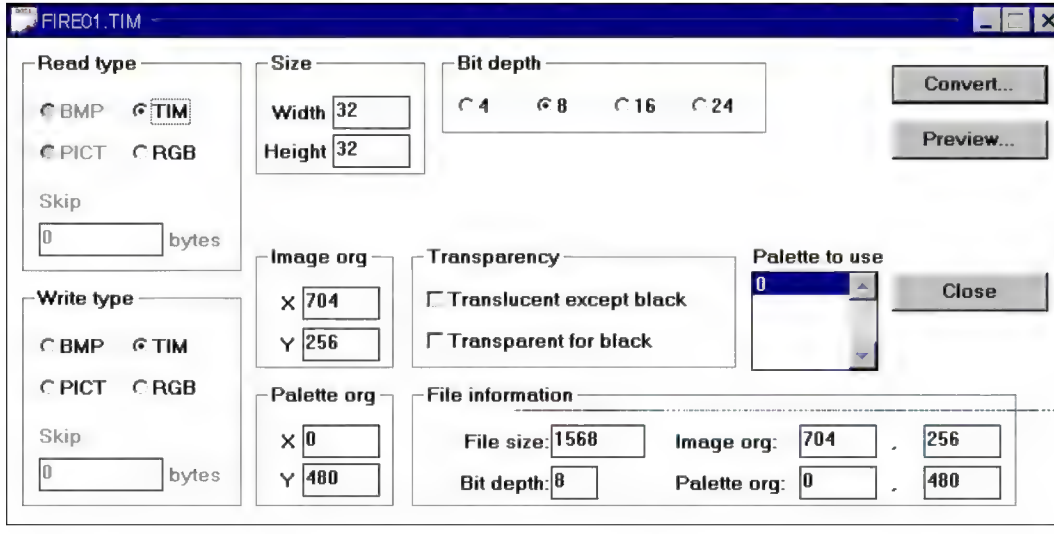


Figure: The Parameter Setting Window

Decription of Each Item in the Parameter Setting Window

Read Format

Shows the format of the input file. Where this is TIM, BMP, or PICT format, it is possible to select RGB format. (Select RGB and the TIM, BMP, or PICT header information is ignored, the file read as RGB data.)

'TIM'

Select when the input file is a PlayStation TIM format file.

'BMP'
Selected when the input file is a Windows BMP format file.

'PICT'
Selected when the input file is a Macintosh PICT format file.

'RGB'
Selected when the input file is a a TIM, BMP, or PICT format file.

Skip
Used to specify the number of bytes when the input file format is 'RGB'.

Write Format
Selects the file format for the converted (output) file.

'TIM'
Select this to convert to PlayStation TIM format.

'BMP'
Select this to convert to Windows BMP format.

'PICT'
Select this to convert to Macintosh PICT format.

'RGB'
Select this to convert to ordinary RGB format.

Skip
Used to specify the number of bytes when the output file format is 'RGB'.

Size
Used to specify the byte arrangement when the input file is 'RGB'. This information is essential for RGB image data interpretation. Reports an error if: the size of the image data (calculated from the values entered here and the value entered in the 'skip' box) is larger than the size of the input file. Displays a warning if the size of the image data is smaller than the size of the input file.

Image Origin

This sets the PlayStation frame buffer image origin co-ordinate values for a TIM output file.

Palette Origin

This sets the PlayStation frame buffer image palette (CLUT) origin co-ordinate in the TIM output file when the bit depth is 8 or less.

Transparency Control

Used to specify the transparency when the file format is TIM, and the bit depth is 16 or less. 'Semi-transparent Except Black' = for all palette entries which have at least one non-zero R, G, or B value, the transparency control bit is set to '1'.

'Make Black Transparent' = for all palette entries in which the R, G, and B values are all '0', the transparency control bit is set to '0'. If this is not selected, when the R, G, and B values are all '0', the colour will be an opaque black.

Bit Depth

Specifies the number of bits per pixel in the output file.

When the output format is BMP only the values 4, 8 and 32 can be selected, when it is PICT only 4, 8 and 16 can be selected, and when it is RGB only 24 can be selected.

Write Palettes

When the input file has more than one palette and the output file is TIM, this selects which palettes will be used in the converted (output) file. The palettes specified here are also used when the TIM is displayed.

Automate the Y co-ordinate setting of the palette origin. Using the File menu 'Setup...' command, set 'Reflect in Origin' set for 'Write palettes'. With this set, the Y co-ordinate value of the palette origin is automatically increased or decreased to match how far the selected entry is from the top of the write palette list.

Read File Information

Displays the size and pixel depth of the input file.

For an input file in TIM format, the 'image origin' is displayed. Also for an input file with a 'bit depth' is 8 or less, the 'palette origin' is displayed.

Convert

Carries out format conversion in accordance with the specified parameters.

Enter the name for the converted file in the save file dialog box. The default directory is the current directory.

This function is the same as 'Save As...' in the File menu

Display

Reads and displays the specified input file.

If the bit depth of the inout file is larger than the depth of the display you are using, the colours are approximated.

Close

Closes the current window.

This function is the same as 'Close' in the File menu.

The Menu Bar

'Open...' Command ([File] menu)

Opens an existing bitmapped file. In the open file dialog box , specify the file to open

In addition to PlayStation TIM, uncompressed format Windows BMP, and 32-bit mode files, the TIM utility supports ordinary RGB format and PICT format files that contain at least one bitmap (the bit depth for output is limited to 4, 8, 16 and 24 bits).

RSD format model data can also be selected. In that case all TIM files specified in the model data are opened.

The 'Close' Command ([File] menu)

Closes the current active window.

The 'Save As...' Command ([File] menu)

Saves the bitmap being worked on with a new filename. In the file save dialog box, enter a suitable name for the converted file and save it. Conversion is then carried out in accordance with the specified parameters.

The 'Save All Files...' Command ([File] menu)

Saves all bitmaps being worked on. . In the file save dialog box, enter a suitable name for the converted file and save it. Format conversion will then be carried out in accordance with the parameters set for each of the bitmaps. The filenames will be the same as the original filenames, except that the filename extensions are replaced by extensions appropriate to the output format.

The 'Setup...' Command ([File] menu)

Specifies the initial values displayed in the parameter setting window when a file is opened. (As shown below.)

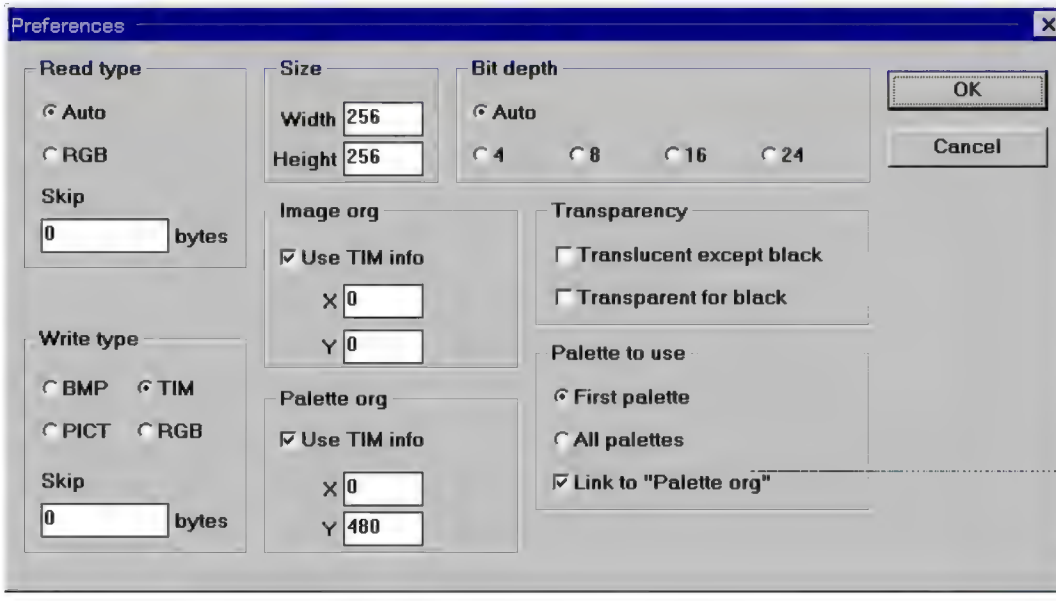


Figure: The Setup Dialogue Box

The Set Up Dialogue box - Notes

- When 'Read Format' is set to 'Same as Read File', the parameter setting window displays the file format details of the input file. When 'Read Format' is set to 'RGB', the format is set to RGB regardless of the type of file. The same applies to 'Bit Depth'.

- When the read format is not TIM or when 'Give TIM Info Priority' is not checked, the utility uses the values entered for 'Image Origin' and 'Palette Origin' as the conversion parameter default values.

- For an input TIM file with more than one palette (CLUT), there is a palette list to select the required palette(s). Set a default selection state for this list using the 'Write Palettes' options.

  If 'First One' is selected, only palette No. 0 is used. If 'Select All' is selected, all the palettes are selected used.

  If the 'Reflect in Origin' option (under 'Write Palettes') is checked and 'First One' is selected from the write palette list, the palette origin will be set to the Y co-ordinate value of the palette origin (of the input TIM file), plus 1.

- All parameters not described above are simply used as the initial values for each item in the parameter setting window.

- The items set up in this dialog box are stored when the TIM utility is shut down.

The 'TIM Arrangement...' Command ([Window] menu)

The 'TIM Arrangement' window indicates graphically the position of the currently open TIM format file(s) within the frame buffer (that is what is where TIM files are in video RAM).

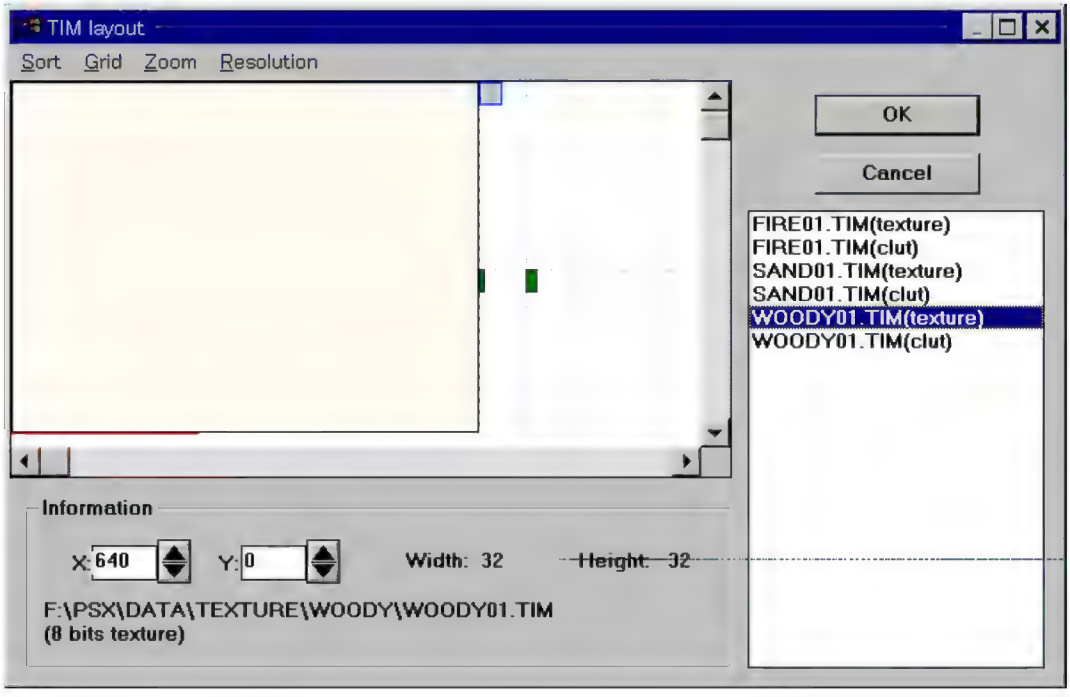


Figure: the TIM Arrangement Dialogue Box

The 'TIM Arrangement' window is roughly divided up into the following four areas:

A frame buffer image area,

An information area,

A selection list area,

A menu bar.

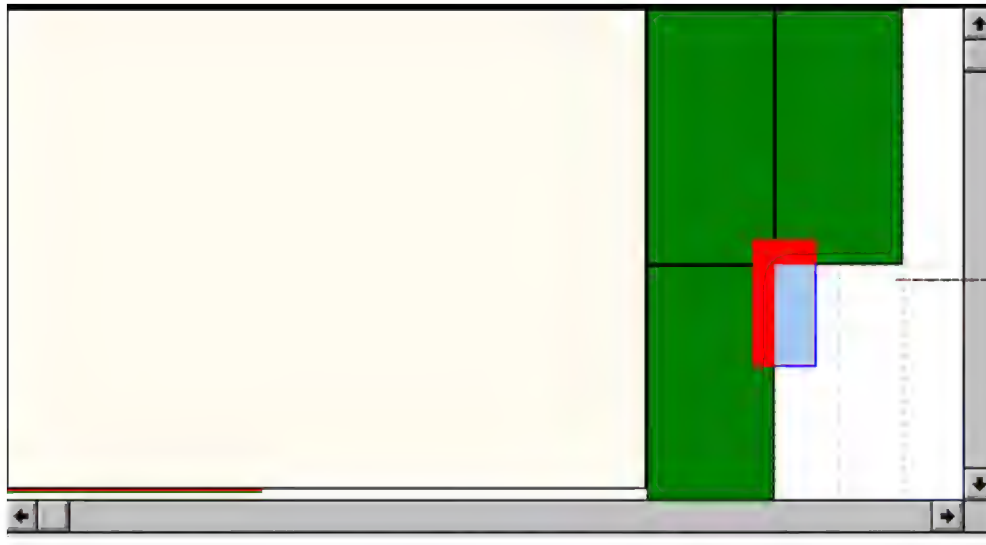


Figure: The Frame Buffer Image Area

This area shows the PlayStation frame buffer image (that is the files stored in video RAM).

In the figure above……..

- Green/red/blue rectangles to the right of the area represent the currently open TIM file images and palettes (CLUTs) at the x,y co-ordinates at which they are stored,
- The two white rectangles on the left represent the screen display areas,
- The dotted lines represent the texture page boundaries. (That is the x and y co-ordinates at which a texture page TIM file must start for the PlayStation hardware to display it correctly. Align the top and left sides of the rectangle with these boundaries.)
- Parts of image or palette rectangles which overlap other image and palette rectangles or protrude from the frame buffer area are coloured red, parts which don't overlap are coloured green (blue if selected).
- Using the left Mouse button, select and drag any rectangle around the frame buffer (hold down the 'Shift' or 'Ctrl' key for multiple select). Selected rectangles are blue.

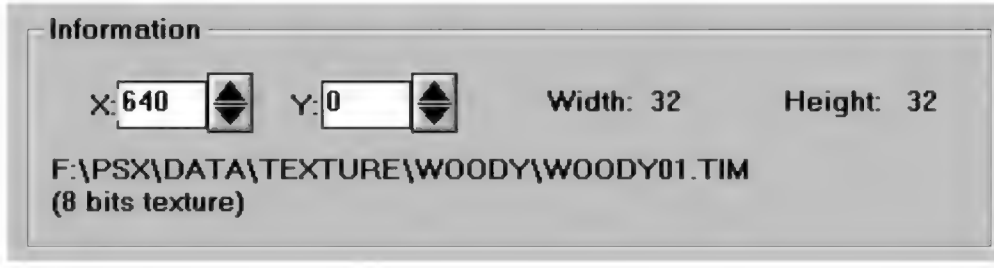


Figure: Information Area

This displays details of the currently selected image or palette: the rectangle origin co-ordinates (top left corner), the size of the file, its filename, and bit depth. When there is more than one file selected, no details are displayed in this information area.

Alter the origin co-ordinates either by clicking the up or down arrow button or by directly entering values in the text boxes.

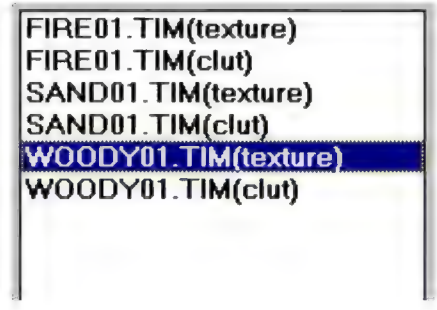


Fig: Selection List Area

This area shows a list of currently open images and palettes. The currently selected file (the light blue rectangle in the frame buffer image area) is highlighted. Click on a file on this list to change the selection (hold down the 'Shift' or 'Ctrl' key for multiple select).

Figure: Menu Bar

The 'Arrange' Menu

'Textures'
Aligns the top left corners of all textures page TIM file with the edges of the nearest texture page boundary. After rough arrangement using the Mouse, use this option to neatly align files.

'Palettes'
For NTSC graphics this vertically spreads out all palettes from the bottom edge of the current display area. (This option does not work for PAL where the display height is 256.)

The 'Grid' menu

'None'
Disables the grid function. Textures are arranged at the co-ordinates specified using the Mouse.

'Texture page'
Textures are moved from the co-ordinates specified with the Mouse, and aligned in rows at the nearest texture-page boundary. When 'XY' is selected, both X and Y co-ordinates are aligned with the texture-page boundary. When 'X' is selected, only the X co-ordinates, and when 'Y' is selected, only the Y co-ordinates are aligned in this way.

'Magnet'
Textures are moved from the co-ordinates specified with the Mouse, and aligned with the edge of the nearest texture area or display area. When 'XY' is selected both X and Y co-ordinates are aligned {with the texture-page boundary(sic)}. When 'X' is selected only the X co-ordinates, and when 'Y' is selected only the Y co-ordinates are aligned in this way.

The 'Zoom' Menu
Sets the scaling factor for image display in the 'frame buffer area'. It is easier to select and move small textures and palettes at some magnicication.

The 'Display Area' Menu
Changes the display resolution used on the PlayStation's display. When this changes, the range of the display area changes.

To display the details of the edited frame buffer image in the parameter setting window, press the 'OK' button. If 'Close' (on the System menu) or 'Cancel' are selected the image and palette movements will not take effect.

Notes

- The following file types cannot be input: compressed format BMP, JPEG, compressed PICT, 32-bit PICT, PICT files that do not contain at least one bitmap.
- The following files cannot be output: compressed format BMP, JPEG,compressed PICT files.
- The bit depth of output files is limited to 4, 8, 16 and 24 bits.
- Colour information is approximated when the bit depth is decreased from, for example, 16-bit data to 8-bit data,. In this situation, the colour system may deteriorate because the method of approximation is a colour map which assigns equivalents for R,G and B values separately, rather than through compression..
- As a rule, the input file cannot be overwritten. Overwriting is possible, however, when TIM format is used for both the input and the output format, and when the image origin or the palette origin have been changed.

timv.bat

This is a TIM (image data) previewer. It displays TIM data on a TV monitor using the PlayStation.

Usage

```
timv  TIM_file
```

Before starting this tool, ensure that the PC and Net Yaroze PlayStation are connected by the Net Yaroze communications cable, that the Net Yaroze PlayStation boot disk is in the PlayStation, and that the power is turned on. Also, timview only works when the baud rate is set to 9600 (the default baud rate with no memory card in the right-hand slot).

When this command is activated, the TIM file supplied as the argument is transmitted to the PlayStation. Next the TIM previewer (timview) is transmitted to the PlayStation, and the previewer starts. Use the Controller to operate the previewer with the Controller functions shown below.

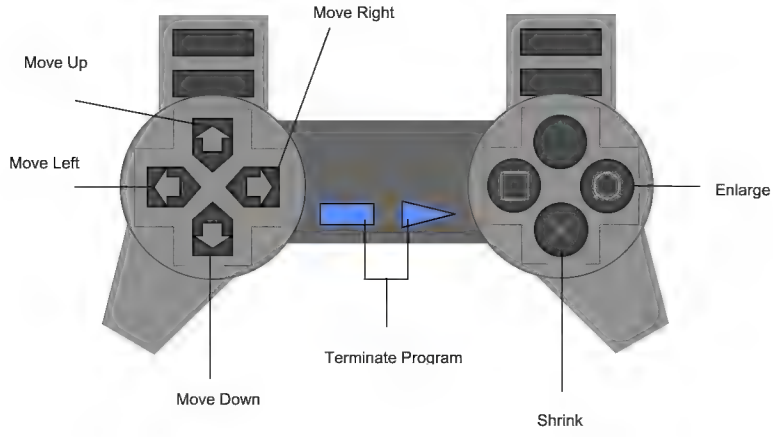


Figure: Controller Pad Operating Method

15

Sound Tools

The sound tools consist of data converters and players. The converters change sound data (waveform data and score data) created using commercial tools, into PlayStation format and construct and edit sound source data. The dedicated PlayStation sound data formats are listed below.

- SEQ data: Score data
- VAG data: Single waveform data
- VAB data: Sound source bank data
- VH data: Sound source data (attribute part)
- VB data: Sound source data (waveform part)

Using the players, you can verify PlayStation sound data on a TV monitor using the PlayStation. All of the sound tools work from MS-DOS. This section briefly describes the function of the main tools.

smf2seq.exe
This tool takes Standard MIDI File (SMF = Standard MIDI File) format-1 data created using commercial sequencer software (score creation editors), and converts it into SEQ data.

aiff2vag.exe
This tool takes 16-bit straight PCM data or AIFF (Audio Interchange File Format) data created using commercial waveform editing software, and converts it into VAG data.

mkvab.exe
This tool starts with VAG data and attribute definition files and creates VAB data.

vabsplit.exe
This tool splits VAB data into VH data and VB data.

sndplay.bat
This tool is an SEQ data player. It reproduces SEQ data on the PlayStation using the standard

sound source found on the Net Yaroze PlayStation boot disk.

vabplay.bat
This tool is a SEQ data player. It reproduces SEQ data on the PlayStation using a sound source created on the PC.

smf2seq.exe

Converts Standard MIDI File (abbreviated to SMF below) format-1 data into PlayStation score data files.

Usage

```
smf2seq [options] SMF-files ...
```

This tool creates a SEQ file (*.SEQ), (a SEQ file is a PlayStation score data file) from SMF files. You can specify more than one SMF file, they will all be converted at the same time. (You can omit the filename extension '.smf').

Options

-Q
Converts in 'Quiet mode': displays no warning messages.

-V
Converts in 'Verbose mode': displays a list of metaevents and control changes that were used in the SMF.

-B
Carries out compulsory deletion of bank changes that were used in the SMF.

Restrictions

The Net Yaroze sound service has the following restrictions.

- SMF format-0 is not supported.
- Files of 64K or more are not supported.

_ The control changes listed below are supported.

⇒ Bank Select(#0)

⇒ Data Entry(#6)

⇒ Main Volume(7)

⇒ Panpot(10)

⇒ Expression(11)

⇒ NRPN(98, 99)

aiff2vag.exe

Converts Audio Interchange File Format data (referred to below as AIFF), windows WAV Format data or 16-bit straight PCM data (without header(s)) into PlayStation waveform data files. All data must be either 16-bit straight or monaural waveform data.

Usage

```
aiff2vag [options] aiff/WAV-files...
```

Creates a VAG file (*.VAG) from an AIFF, WAV format or 16-bit straight PCM format file. (A VAG file is a PlayStation compressed waveform data file.) You can supply more than one AIFF file as the argument, they will be converted at the same time. (You can omit the filename extension '.aif ' or 'wav').

Options

-1
During encoding, waveform data with loops will be compulsorily encoded as a sound source without loops.

-L
During encoding, waveform data without loops will be compulsorily encoded as a sound source with loops.

-R fs

Specifies the sampling rate for 16-bit straight PCM data input. 'fs' is specified in hertz.

-E
Endian (byte strings order) conversion will not be carried out.

Restrictions

The Net Yaroze sound service has the following restrictions.

- Only 16-bit uncompressed data is supported. No support is provided for 4-bit, 8-bit or compressed format data.
- Only monophonic data is supported. When converting data from a stereo source, convert the channels separately.

mkvab.exe

Constructs sound source bank VAB data from attribute tables and VAG format waveform data which has already been created using aiff2vag.exe.

Usage

```
mkvab [option]  vag-files....
```

Options

-f def_file
Specifies the definition file (def_file) that creates the attribute table(s).

-r vab_file
Analyses a VAB file, and outputs attribute definition files.

-o out_file
Specifies the output file.

Restrictions

The Net Yaroze sound service has the following restrictions.

- The size of the VAB file and the size of each of the VAG files is calculated automatically, so the specified value is ignored.

Attribute Definition Files

Label	Value	Explanation
form	'VABp'	Format identifier
ver	7	Format version number
id	0	VAB id (usually 0)
fsize	0	File size (mkvab calculates this automatically)
ps	0~128	Total no. of programs in the VAB data
ts	0~2048	Total no. of tones in the VAB data
vs	0~254	Total no. of VAGs in the VAB data

Table: VabHdr (Definitions of Attributes that Affect the Whole VAB File)

Label	Value	Explanation
tones	4	No. of tones in the program
mvol	0~127	Program volume value
mpan	0~127	Program panning value

Table: ProgAtr (Definitions of Program Level (equivalent to instrument level) Attributes)

Label	Value	Explanation
prior	0~127	Tone priority level - the higher the value the higher the priority
mode	0,4	Setting this to 4 gives a reverberation effect. *
vol	0~127	Tone volume value
pan	0~127	Tone panning value
center	0~127	Centre note (in semitone units)
shift	0~127	Centre note fine tuning
min	0~127	Note limit minimum value
max	0~127	Note limit maximum value
pbmin	0~127	Maximum value for downwards pitchbend (in semitones)
pbmax	0~127	Maximum value for upwards pitchbend (in semitones)
ar	0~127	Attack rate
dr	0~15	Decay rate
sr	0~127	Sustain rate
rr	0~31	release rate
sl	0~15	sustain level
prog	0~127	no. of program that tone belongs to
vag	0~254	the VAG in use

Table: VagAtr (Definitions of Tone Level Attributes)

* To obtain a reverberation effect, the reverberation must be set using a separate sound function.

Supplementary Notes

You can set the ADSR (envelope) rates - that is, the rates for attack, decay, sustain and release - individually, and specify linear and exponential function curves for the change over time. You can also set the sustain level.

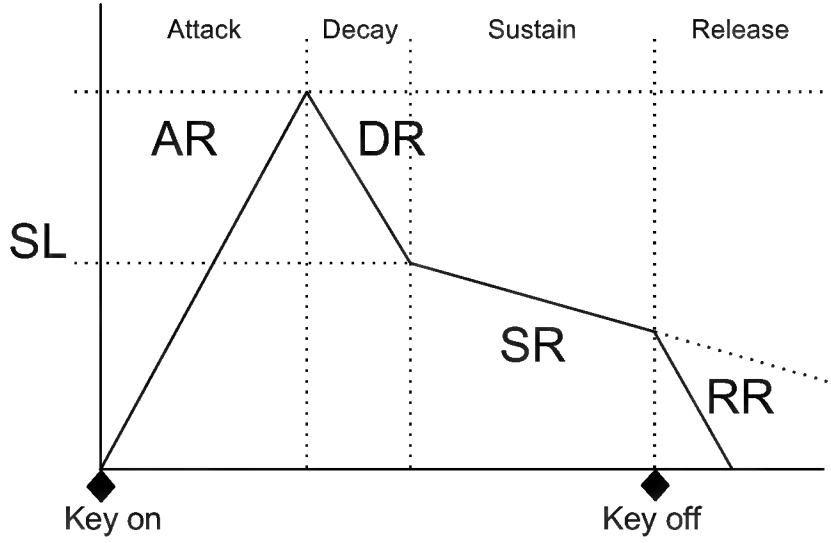


Figure: ADSR Concept Diagram

vabsplit.exe

Splits PlayStation sound source bank (VAB) data which has been created using 'mkvab.exe' into an attribute table part (VH) and a waveform data part (VB). VH data must be present in the main memory during reproduction of the music, whereas VB data need not be present in the main memory once it has been transmitted to the SPU.

Usage

```
vabsplit vab-files...
```

Splits vab data into a PlayStation sound source bank attribute table part and a waveform data part. You can specify more than one VAB file as the argument, they will be converted at the same time. (You can omit the filename extension '.vab'.)

There are not options or restrictions associated with this tool.

The Sound Players

Running Sound Players

When you installed the Net Yaroze system on your PC, the batch files, 'seqplay.bat' and 'vabplay.bat' were installed in the command directory, and example sound source and sampled score data files in the 'data' subdirectory (including 'sample1.seq' used in the example below).

With the Net Yaroze system set up (the PC and the PlayStation both switched on and connected via the connector cable, the boot disk in the PlayStation), type the following command from the 'data\sound' directory.

(This loads standard sound source data files (STD0.VH,STD0.VB) from the boot disk and plays the specified SEQ file.)

```
C:\> seqplay sample1.seq
```

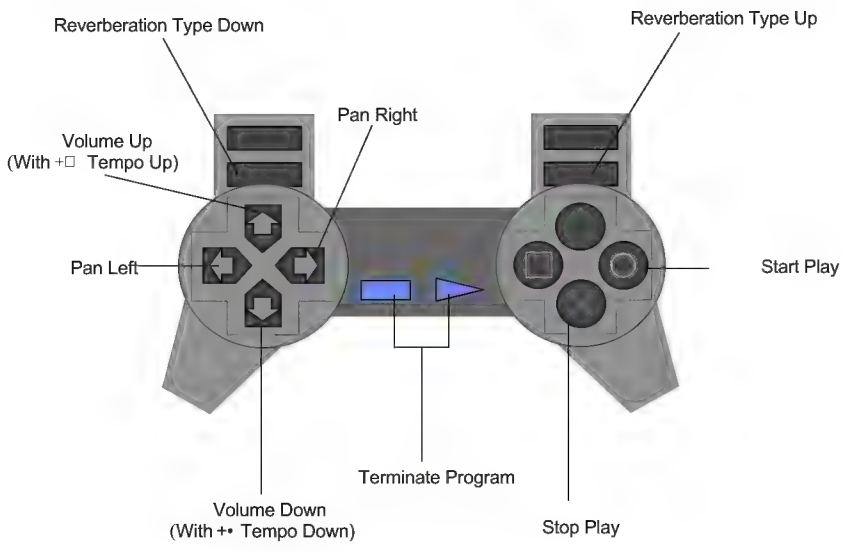
You can play sound data using a sound source set other than the standard sound source data file by transmitting the data via the serial port. Use the following command.

```
C:\ > vabplay my.seq my.vh my.vb
```

Operating the Sound Player

If you have activated the program correctly, the PlayStation will play music and output an image to the TV screen, similar to one shown below.

Use the Controller functions shown in the diagram below to manipulate the sound.



Sound Source Data Specifications

Sampled sound source data can be used in the same way as MIDI sound source data. For details, refer to the Net Yaroze Web site.

16

Programming Tools

In the Net Yaroze system, a GNU C compiler and associated utilities are provided as programming tools.

Compiler	gcc.exe
Linker	ld.exe
Debugger	gdb.exe
Librarian	ar.exe
Maintenance utility	make.exe
Symbol information remover	strip.exe
Object Controller	nm.exe
Assembler	as.exe, etc.
Others	size.exe

In this chapter only the most frequently used tools ('gcc', 'ld', 'strip' and 'make') are discussed. For details of the other commands, please refer to the documents belonging to the GNU C compiler, or to related commercially available documents. (See Additional Reading List at the end of the Start Up Guide)

The Compiler'gcc'

The gcc compiler creates object and executable files from C source files.

Filename Extensions and Actions

gcc calls different tools to link and compile the input C source files. The tools called depend upon the file extension (that is its suffix, the '.c' part of :filename.c). See the list in the table below.

Note that files with filename extensions that the compiler cannot recognise are treated as object files, and the linker is called.

Extension	Tools Called and Order of Calling
.C	C pre-processor→ C compiler → assembler → linker
.I	C compiler → assembler → linker
.CC	C pre-processor→ C++ compiler → assembler → linker
.CPP	C pre-processor→ C++ compiler → assembler → linker
.II	C++ compiler → assembler → linker
.S	Assembler → linker
others	Linker

Table: Tool Calling Order

Also, when you use gcc, you can specify different options to halt compilation at different points: at the pre-processing, linking, assembling or compiling stage.

The output object file or executable file name depends on the compiler options you have specified.

Usage

To execute gcc, type the following command at the MS-DOS prompt.

```
gcc [-option] <source files>...
```

[-option]

Specify options after a hyphen ('-'). Remember that the compiler is case sensitive (that is '-O' is not the same as '-o'). When specifying more than one option, separate each with a space.

<source files>...

After the options, leave a space, then specify the name of the source file.

Examples (using -c or -o options)

- To compile a source file called 'test.c' and create an object file ('test.o'), use the following command:

```
gcc -c test.c          // Creates the object file 'test.o' from test.c.
```

- To compile 'test.c' and create the executable file 'test', use following command:

```
gcc -o test test.c      // Creates the executable file 'test' from test.c.
                           Note that you must specify the executable file
                           name (here 'test').
```

- To compile more than one source file ('test1.c', 'test2.c' and 'test3.c') to create three corresponding object files ('test1.o', 'test2.o' and 'test3.o') in one command:

```
gcc -c test1.c test2.c test3.c
```

Options

There are number of options for 'gcc'. Only the most essential are described below. Refer to the GNU C compiler documents, or to related commercially available documents for further details.

Compiler Specific (gcc) options

- Default (no options)

  With no options specified, 'gcc' calls the linker and creates an executable file. With no name specified for the output file, the default is 'a.out'.

```
gcc test.c                  //output to a.out
or
gcc test.c -o temp3.exe     //output to temp3.exe
```

- Execute the pre-processor only (-E)

  The '-E' option specifies that only pre-processing be carried out. (That is there is no compiling and no linking.) With no output filename specified, the result will be displayed to standard output (the computer screen).

```
gcc -E test.c                  //Output to screen
or
gcc -E test.c > test.pre       //Output to file test.pre
```

- Output Assembler Code (-S)

  The '-S' option specifies processing up to compilation. (That is, there is no linking.) Specify a C file

as the source file to output an assembler file. With no name specified for the output file, the default is the same as the input file name but with the extension '.s'.

```
gcc -S test.c                  //output to test.s
or
gcc -S test.c > temp1.s        //output to temp1.s
```

- Output an Object File (-c)

  The '-c' option specifies, processing up to creation of an object file (.o). (That is there is no linking.) With no name specified for the output file, the default is the same as the input file name but with the extension '.o'.

```
gcc -c test.c                  //output to test.o
or
gcc -c test.c -o object1.o     //output to object1.o
```

- Support ANSI (-ansi)

  The '-ansi' is specified, the compiler supports all ANSI standard C programs.

```
gcc -ansi test.c               //take and ansi C file and output to a.out
```

- Create Debugging Information (-g)

  The '-g' option specifies inclusion of debugging information in the executable file. Always specify this option if you are going to use 'gdb', the GNU debugger.

```
gcc -g test.c -o test          // output to test - with debugging information
```

Optimisation Options

These options tell the compiler to improve the efficiency of the executable files that it creates

Note: You cannot use optimisation options with the debugger option (-g).

- Don't Optimise (-O0)

  The '-O0' option specifies no optimisation. This is the default setting so is usually not used.

- Standard Levels of Optimisation (-O, -O1, -O2, -O3)

  The '-O', '-O1', '-O2' or '-O3' options specify levels of optimisation, (from -O, which is minimum optimisation, to -O3, which is the maximum level of optimisation).

```
gcc -O2 test.c
```

Linker Options

The following options control the linker.

- Don't Link the Standard Library (-nostdlib)

  By default, the Net Yaroze standard library is linked into the application. Use the '-nostdlib' to tell the compiler not to link the standard library. With this option, only the file(s) specified are passed to the linker.

```
gcc -nostdlib test.c        //the standard library is not included in the
                            executable
```

- Library Specification (-l<libname>)

  This option specifies the name of the library (where '<libname>' is the required library) you want to be linked. (Note that there is not space between the option and the library name.) this option to include your own libraries that you have created using the librarian utility.

```
gcc -lmylib.lib test.c      //link the library 'mylib.lib' to 'test.c'
```

- Linker Option Specification (-Xlinker <linker option>)

  You can use options supported by the separate linker ('ld', discussed later) with the gcc compiler by specifying the option: '-Xlinker'. However, if the linker options you wish to specify contain spaces, you must describe the options separately, calling '-Xlinker' for each option and any output file.

```
gcc -Xlinker -Map -Xlinker mapfile test.c  //causes the creation of map file
```

Note that '-Xlinke'r is called to specify the option and called again to specify the output file.

or

```
gcc -Xlinker -Ttext -Xlinker 80140000
```

General Options

The following options control 'gcc' as a whole.

- Warning Messages (-W, -Wall)

  Use the '-W' or '-Wall' option to get warning messages from the compiler in response to various events.

```
gcc -W test.c
```

or

```
gcc -Wall test.c            //set maximum warning level
```

- Macro Definition (-D<NAME>, -D<NAME=VALUE>)

  Use the '-D' option to specify a macro where '<NAME>' is the name of the macro that you want to use and, if appropriate, <VALUE> is a numerical value assigned to it. For example:

  Specify the Macro 'DEBUG'

```
gcc -DDEBUG test.c
```

or

  Specify the Macro 'DEBUG' as '0'

```
gcc -DDEBUG=0 test.c
```

- Remove a Macro Definition (-U<NAME>)

  Use the '-U' option to specify the macro you wish to remove where '<NAME>' is the name of the macro you wish to remove. For example:

  Remove the Macro 'DEBUG'

```
gcc -UDEBUG test.c
```

- Display Detailed Information (-v)

  Use the '-v' option to get information form the compiler on the way in which each tool is activated.

```
gcc -v test.c
```

-o Warning

You can only specify one output name using the '-o' option. Thus only use this option when you are compiling to make an executable file out of the input file.

For example:

```
gcc test1.c test2.c test3.c -o test
```

This compiles and links the three source files (test1.c, test2.c, test3.c) into the one executable file, 'test', as required.

However, the same compile command with the '-c' (don't link) option (as below), causes the compiler to compile each source file independently and write each output file to 'test', subsequent ones overwriting previous ones Thus only the results of compiling the last file (test3.c) are saved in 'test'.

```
gcc -c test1.c test2.c test3.c -o test
```

The Linker 'ld'

'ld' produces a single application (executable file), rejoining divided subprograms.

Usage

To execute 'ld', input the following command at the MS-DOS prompt.

```
ld [-o <output>] <obj files>.... [-option]
```

Specify the object files that you want to link as arguments.

[-o <output>]

Use '-o' immediately after 'ld' to specify an output file name where '<output>' is the required output name. Ensure there is a space between '-o' and the filename.

[-option]

Options are preceded by a hyphen ('-'). Separate multiple options with spaces.

Only the most commonly used options are describe below.

Options

- Standard Output Specification (-o <output filename>)

  Use '-o' immediately after 'ld' to specify an output file name where '<output filename>' is the required output name. The default output filename (without this option) is 'A.OUT'.

```
ld -o test test.o       //outputs to 'test.o'
```

- Symbol Definition (-defsym <symbol=expression>)

  After '-defsym', specify the symbol ('symbol') you wish to define and its value ('expression') joined by a '=' (equals sign). Separate '-defsym' and the symbol with a space.

```
ld -o test test.o -defsym DEBUG=1      //outputs to 'test.o' and defines
                                       'DEBUG' as 1
```

- Mapfile Creation (-Map <mapfile> or -M)

  Use '-Map'to make the linker output all external singles the mapfile name specified (where '<mapfile>' is the specified mapfile name). Alternatively use '-M' to display mapping information to standard output (screen)

```
ld -o test test.o -Map mapfile         //outputs object file to 'test.o'
                                       and mapping information to 'mapfile'
```

or

```
ld -o test test.o -M                   //outputs object file to 'test.o' and
                                       mapping information to screen
```

- Symbol Display (-y <symbol>)

  Use '-y' to display information regarding the object file (where 'symbol' specifies the information required).

```
ld -o test test.o -y DEBUG     //outputs object file to 'test.o'
                               and displays debug information to screen
```

strip (Symbol Information Remover)

The 'strip' is a utility removes symbolic information from the executable file.

Symbolic information fulfils an important role during program creation, but is not needed when the application is released and widely used by Net Yaroze members. As 'strip' removes symbolic information, it reduces the size of an executable file.

Usage

To execute 'strip', enter the following command at the MS-DOS prompt.

```
strip  [-option]  <file>
```

Options are preceded by a hyphen ('-'). Separate multiple options with spaces.

After the options, leave a space, then specify the name of the executable file.

For example, to remove symbol information from the file 'test', use the following command.

```
strip test
```

For details of the options, refer to the GNU C compiler documentation and, related commercially available documents.

The Maintenance Utility 'make'

'make' is a maintenance utility that automates program construction and reconstruction, that is compiling and linking of c source files into object and/or executable files.

The make utility is very useful as you can use it to reconstruct only the elements of the program that need to be reconstructed. To take a simplified example: you have three source code files: test1.c, test2.c and test3.c which you have already compiled and linked. Then you find that you need to change test3.c. Using the make utility, you can change recompile and link only the altered file, rather than all three, thus speeding up your program construction time. While compile times may not be significant in this simple example, in a large project there can be many source files, header files and complex dependencies of source files on header files. Thus waiting for compile and linking can be time consuming.

The make utility works in conjunction with a 'Makefile'. This is an ordinary text file which describes a program's construction sequence and dependency relationships. The make utility decides whether a program needs to be reconstructed by examining the time stamps of target file and the source files on which the program depends. Generally, if just one of the source files is newer than the target file, it reconstructs the target file. Thus, in addition to speeding up the process of program construction, it simplifies the process. You don't have to remember which files you been changed since the last project build, and which files depend on them.

The command 'make' calls the makefile.

Usage

To execute 'make', type the following at the MS-DOS prompt.

```
make [-f makefile] [options] <target>...
```

Options are preceded by a hyphen ('-'). When specifying more than one option, separate them by spaces.

After the options, specify the file that make must work on (here, '<target>') . If no target file is specified, make works on the first file it finds in the makefile.

By default, make looks for a makefile called 'makefile' in the directory that you are currently in. However, 'make' can use any named text file using the '-f' option.

The most widely used options are described below. For other options, refer to the GNU C compiler documentation or related commercially available documents.

Options

- 'Makefile' Filename Specification (-f <filename>)

  Use '-f' to specify a makefile (where '<filename>' is the required name). The default filename is 'makefile'.

```
make                        //assumes existence of 'makefile'
or
make -f test1.mak           //runs the makefile called 'test1.mak'
```

- Ignore Errors (-i)

  When you specify '-i', the makefile continues execution even when commands return an error status

```
make -i all                //continues to run the makefile, regardless of
                           errors
```

- Display Debugging Information (-d)

  When you specify '-d', 'make' debugging information is displayed.

```
make -d all                //displays debugging information
```

- Inhibit Display (-s)

  Specify '-s' so 'make' executes silently, not displaying the commands it is running.

```
make -s all                //'make' executes silently
```

- Question (-q)

  Use '-q' to instruct 'make' to return the status of makefile processing (returns '0' if the target file has been updated, '1' if not).

```
make -q all                //returns the status of makefile processing
```

Makefile

A 'Makefile' is a standard text file which describes a program's construction sequence and dependency relationships. It is managed by means of explicit rules known as 'dependency rules', and 'implicit rules'.

A target file can be any file that a the compiler can create automatically, an object file for example. Target files usually have files on which they depend ('dependencies'). For example, a C source file that includes a C header file depends on that header file. If the header is changed in any way, then the source file needs recompiling, the new target file will then incorporate the changes that have been made to the header file.

Target files also have commands. These tell the 'make' utility how to create the target file, when it is created or updated.

By specifying both the dependencies and the commands for all target files you are interested in, you tell 'make' both when and how to update target files. So 'make' works out which files to update, making program update a lot simpler.

When building a target file, 'make' first of all searches for the dependency rules for that target file. If there are no dependency rules, it constructs a target file using implicit rules.

Dependency Rules

Specify dependency rules in the makefile in following way.

```
Targetfile.exe:    <depfile1> <depfile2> <depfile3> <depfile4>
                   command
                   ...
                   command
```

Specify the target file (here 'targetfile.exe'), followed by a colon (:) and its dependency files (here '<defile1>' and '<depfile2>', etc).

Ensure that there is a space between each dependency file's filename. Note that a long list of dependency files can be split over many lines using the backslash, as shown below:

```
Targetfile.exe:    <depfile1> <depfile2> <depfile3> <depfile4>\
                   <depfile5> <depfile6> <depfile7> <depfile8>
```

Dependency files can be either C or C++ source files or header (.h) files.

After the dependency files, list the commands used to create the target file. You need to list them in the order of execution, one per line, with a tab at the beginning of each line.

Note that the command lines must begin with a single tab as any spaces cause 'make' to report errors which aren't really there.

How It Works

```
Targetfile.exe:    <depfile1> <depfile2> <depfile3> <depfile4>
                   command
                   ...
```

```
command
```

The relationships between each field, and the resulting actions are as follows.

- The first line specifies that the file 'targetfile.exe' depends on '<depfile1>' '<depfile2>' '<depfile3>' '<depfile4>'.
- If any of the dependency files is newer than the target file, or if the target file does not exist, 'make' executes the commands that follow and reconstructs the target file.
- If no dependency files are listed, 'make' always reconstructs the target file.
- If any of the dependency files does not exist, 'make' tries to reconstruct the missing dependency file(s) before executing the commands to create the target file.
  However, if 'make' cannot find the rules that define how to reconstruct the necessary file(s), it stops and reports an error.

Examples of Dependency Rules

Example 1

```
main.o:     main.c main.h
            gcc -c main.c
```

Example 2

```
main.exe:   main.c main.h
            gcc -o main.exe main.c
```

In example 2, 'main.exe' is dependent on 'main.c and 'main.h'. If either of these files is newer than 'main.exe', or if 'main.exe' does not exist, the command 'gcc -o main.exe main.c' will be executed to create or renew the target file 'main.exe'.

Example 3

```
main.exe:   main.c main.h
            gcc -c main.c                 // create the object file main.o
            ld -o main.exe main.o         // create the executable file
```

In this example, two commands construct 'main.exe'

Example 4

```
clean:
```

```
del *.o
```

In example 4, the 'target file' is called 'clean' and depends on no other files. The action is the

MS-DOS command, <del *.o>, which deletes all filenames with .o extension.

This example shows that you can use 'make' as a simple action-labeller: here it doesn't create a target file at all, it just specifies that the name 'clean' refers to the action 'delete all files with .o extension'.

Implicit Rules

If there are no commands to construct the target file, 'make' searches for implicit rules that define how to construct the target file.

These implicit rules are general rules that define how to create one type of file from another, for example, how to convert an '.ASM' file into an '.EXE' file.

Implicit rules take the following form.

```
.<source extension>.<target extension>:
                                        command
                                        ...
                                        command
```

'target extension' specifies the filename extension of the target file, and 'source extension' specifies the filename extension of the file(s) which the target file is made from. On the following line there is at least one command that constructs files that have the target filename extension.

For example:

```
.c.o:
        gcc -c $@ $<
```

In this example 'make' creates files with the extension '.o' from files with the extension '.c' using the 'gcc' command. (See below for the meanings of "$@"and "@<").

Command Searching

'make' carries out its search for commands in the following order.

1. The current directory
2. The directory specified by PATH

If a specified command is not an 'EXE' or 'COM' file, or if it is a 'BAT' file, 'make' calls 'COMMAND.COM' to execute the command or batch file. As a result, MS-DOS commands such as 'CD' and 'DEL' can be used within makefiles (as shown inExample 4, above).

Command Prefix

You can use the '@' prefix when specifying commands in dependency and implicit rules. It tells make not to display the command.

For example:

```
@ld -o test test.o
```

If the '-s' option is not specified, 'make' normally displays the command it is executing. To prevent this, add '@' to the beginning of the command to be executed.

Macros

A macro is a symbol that represents a piece of text. Macros are a form of shorthand-labelling.

Macros take the following form.

```
Macro_name = Macro_text
```

The name of the macro ('name') and the text that defines its content ('text') are joined by an equals sign ('='). Upper case and lower case letters are not distinguished.

If a macro definition refers to another macro, the macro referred to is expanded on use. 'Make' are expands macros used in rules immediately.

You can redefine macros at any time.

When a macro appears, its contents are replaced by the character string defined by 'text'. A defined macro can be referred to in the following form.

```
${macro_name}
```

Example 1

```
C_FLAGS = -O2 -DDEBUG
...
.c.o:              // implicit rule for constructing .o files from .c files
gcc ${C_FLAGS} -c $@ $<
```

In this example, '$(C_FLAGS)' in the 'gcc' command line will be replaced by '-O2 -DDEBUG'.

Example2

```
PROJECT = main                                    // name of executable
OBJECT_FILES = main.o pad.o tim.o tmd.o           // list of object files
LINKER = -Xlinker -Ttext -Xlinker 80100000        // linker option
${PROJECT}: ${OBJECT_FILES}
gcc ${LINKER} ${OBJECT_FILES} -o ${PROJECT}
```

Predefined Macros

Predefined macros all begin with the dollar symbol ('$'), and can be used instead of filenames in dependency or implicit rule command lines.

The table below shows examples of Predefined Macros.

Macro	Content
$@	name of target file
$?	Updated dependency file
$<	first dependency file

Table: Examples of Predefined Macros

Example1

```
LINKER = -Xlinker -Ttext -Xlinker 80100000        // linker option

main: main.o pad.o tim.o tmd.o
```

```
gcc  ${LINKER) -o $@                    // equivalent to: gcc ${LINKER)
                                        -o main
```

Directives

You can use the following directives.

Directive	Content
define <variable>	Set up 'variable'
endef	Remove the 'variable' set up using 'define'
ifdef <variable>	Test if 'variable' has been set up. On 'true' execute the next line. On 'false' jump to 'else' or 'endif'.
Ifndef <variable>	Test if'variable' has not been set up. On 'true' execute the next line. On 'false' jump to 'else' or 'endif'.
Ifeq (A,B)	Test if A and B are equal. If equal execute the next line. If not equal jump to 'else' or 'endif'.
Ifeq "A" "B"	(Same as above)
ifeq 'A' 'B'	(Same as above)
ifneq (A,B)	Test if A and B are not equal. If not equal execute the next line. If equal jump to 'else' or 'endif'.
Ifneq "A" "B"	(Same as above)
ifneq 'A' 'B'	(Same as above)
else	Execute the next line if the immediately previous 'if....' statement condition clause is not actioned.
Endif	End the 'if....' statement condition clause(s)
include <file>	Include the file 'file'

Table: Makefile Directives

Comments

'make' treats a line with a hash symbol at the beginning (#) as a comment.

Example

```
# whole line comment: main.exe only depends on main.c
main.exe: main.c
```

Line Division

If a command is too long to be contained in a single line, you can spread it over more than one line by adding a backslash symbol ('\') to the end of the line to be continued.

Example 1

```
main.exe:  main.c header1.h header2.h \
           header3.h header4.h
```

Example 2

```
OBJECT_FILES =     file1.c file2.c file3.c\
                   file4.c file5.c file6.c file7.c
```

17

The Console Tool

SIOCONS is a DOS console tool that allows you to download programs and their associated data to the Net Yaroze Members' PlayStation and execute them.

An Overview of SIOCONS

SIOCONS is a user interface front end program that operates the Net Yaroze Member's PlayStation. Its operating environment requirements are as follows.

Operating machine type:	IBM-PC compatible computer
Operating environment:	Net Yaroze Members' PlayStation
Operating OS:	DOS Version 5 or DOS Version 6,
	Windows 3.1 (DOS window), Windows 95 (DOS box)
Driver required:	ANSI.SYS

Usage

```
siocons [-pport address,IRQ] [-Bbaud rate] [auto file]
```

1. Set up your PC system so that SIOCONS can function properly. Check that 'ANSI.SYS' is included in your CONFIG.SYS file, and if it is not, edit your CONFIG.SYS file and reboot your PC.
2. Check that your PC and PlayStation are connected using Communications cable.
3. Insert the Net Yaroze boot disk in your Net Yaroze Member's PlayStation, and switch on. If the Access card is not in the PlayStation at this point, insert it into Memory card slot 1.
4. Start SICONS on you PC by typing:

   C:>**siocons -<option1>** (where '<option1>' is the required parameter, if any is required)

Options

-pport- address,IRQ	specifies communication port address and IRQ setting
-Bbaud rate	specifies communication rate

auto file specifies an autoexecution batch file

Example

```
siocons -p0x3f8,4 -B115200
siocons batch1
```

Operating Method

With SIOCONS, normal keyboard input is sent to the PlayStation, and characters sent from the PlayStation are displayed on your PC monitor. (So, the C function <printf>, called by a program running on the PlayStation, will output to the SICONS console on the PC). However, function keys and cursor movement keys are processed locally and are not sent to the PlayStation.

Monitor Commands

You can use certain PlayStation monitor commands with SICONS. These are listed below.

Command	Function
DW/DH/DB	Hexadecimal memory content display
SW/SH/SB	Hexadecimal memory content substitution
FW/FH/FB	Continuously write to memory
DR	Hexadecimal register content display
SR	Hexadecimal register content substitution
GO	Execute program.
DIS	Disassemble memory contents
AC/DC/SC	Save user defined commands
AD/DD	Save device drivers
HELP/?	Display help messages
RDB	Transfer to GNU debugging monitor mode
DIR	Display a list of files in directories
CD	Change or display current directory

READ	Copy data (file→ memory)
WRITE	Copy data (memory→ file)
REN	Rename file
DEL	Delete file
FORMAT	Format file
LOAD	Read in PLAYSTATION EXE (specify filename)
EXEC	Execute PLAYSTATION EXE (specify filename)
WAR	Copy data (memory→ waveform memory)
WAW	Copy data (waveform memory→ memory)
VAR	Copy data (memory→ frame buffer)
VAW	Copy data (frame buffer→ memory)
PLAY	Play DA (specify track)
BAUD	Set communication speed
CB	Display colour bar
CLS	Clear console screen

Table: SIOCONS PC Monitor Commands

Local Commands

There are a number of local commands with SICONS Execute these by pressing function keys at the SIOCONS command prompt.

[F1] Display help messages.

[F2] Input a DOS command and execute this command.

[F3] Input the name of a batch file and execute this file.

[F4] Input the name of an object file and download this file.

[F5] Input the name of an operation log file and start logging operations.

[F8] Toggle the local line editor function ON/OFF.

[F9]→[F4] Input the name of a binary file and download this file.

[F9]→[F5] Stop logging operations.

[F10]→[F2] Terminate the SIOCONS program.

[F10]→[F4] Input a filename and upload a memory image.

Editing Functions When Executing Local Commands

Depending on the command involved, you may need to input filenames when on executing a local command. In this situation, you can edit the command line using the following keys.

Left Arrow, Ctrl-S	Move cursor to the left.
Right Arrow, Ctrl-D	Move cursor to the right.
Up Arrow, Ctrl-E	Move back through history buffer.
Down Arrow, Ctrl-X	Move forward through history buffer.
Ctrl-G	Delete character at cursor position.
Ctrl-Y	Delete line. When referring to history buffer, escape from history buffer.
Ctrl-K	Delete from cursor position to end of line.
TAB	Move back through history buffer until the current input line matches the head part.
Ctrl-J	Move forward through history buffer until the current input line matches the head part.
ESC	Push current line into history buffer without executing it.

Downloading and Executing Files

Downloading Programs

To download a program you have created onto the PlayStation, press the [F4] key at the SIOCONS command prompt. When you do this, the 'Load[x]' command prompt is displayed. Enter the executable filename at this prompt. Here you can specify 'PLAYSTATION EXE' format executable files (however, the filenames need not end in .exe).

For example

```
Load[1]: main
```

Note that you need to download the data used by any program before the program can run.

Downloading Data

To download data you have created to the PlayStation, press [F9] followed by [F4] at the SICONS command prompt. When you do this, the 'Dload[x]' command prompt is displayed. At this prompt, specify the filename and the hexadecimal address pair where that file will be loaded into main memory. More than one file and address pair can be specified.

For example

```
Dload[2]: wheel256.tim 80190000
```

This example loads the binary TIM file (where 'whee1256.tim' is the file name) into main memory, starting at address 0x80190000

For example

```
Dload[3]: giulieta.tmd 801a0000
```

This example loads the binary TMD file into main memory (where 'giulieta.tmd' is the file name), starting at address 0x801a0000.

Program Execution

To execute a program that has been downloaded to the PlayStation, execute the monitor command 'go' from the SIOCONS command prompt. (You can specify the execution start address using the 'go' command, but this is not usually necessary).

Terminating SIOCONS

To terminate SIOCONS execution, press the [F10] key, followed by the [F2] key, at the SICONS command prompt. Alternatively, force-quit it by pressing [Esc].

Auto-execution

SIOCONS has a simple auto-execution facility: this enables many commands to be executed in sequence without you having to type them all in each time. You can do this by preparing a simple text file (batch file) containing a list of keyboard input commands which will be read and executed in sequence, very much like

batch files in the MS-DOS operating system. Having created the text file, you can run that entire set of commands by a single command to SIOCONS.

To use the auto-execution facility, press the [F3] key at the SIOCONS command prompt. When you do this, the 'Auto[x]' command prompt will be displayed. At this prompt, specify the name of the auto-execution file you have created. The file you specify will be read and the commands executed in sequence.

[Note that the auto-execution facility executes the first line of the batch file unconditionally, so ensure that the SICONS command prompt is displayed when you use this facility; i.e. before running a batch file, make sure that the Net Yaroze Member's PlayStation is in its responsive ready-to-download state.

For example

```
Auto[3]: batch1
```

Local Commands with Batch Files

To execute a SICONS local command from within a batch file, first write the word 'local', then the local command and its arguments, as follows.

```
local local-command [argument....]
```

You can also use the following local commands.

Commands	Action
help	Display help messages.
dos <argument>	Execute the DOS command supplied as the argument.
auto <batchfile>	Execute the batch file supplied as the argument. Nesting of batch files up to 16 levels is allowed.
dload <filename address>	Specify a filename and address pair as the argument, to download a binary file.
load <filename>	Specify a filename in the argument, to download and executable file.
log [filename]	If the name of an operation log file is supplied as the argument, operation logging will begin. If there is no argument, operation logging will terminate.
dsave <filename address size>	Specify a combination of filename, address and size as the argument and a memory image will be uploaded into the file.

beep	Sound the buzzer
pause	Wait for keyboard input. Continue when a suitable key is pressed
echo <string>	Display the character string supplied as the argument.
sleep <second>	Pause for the number of seconds specified in the argument.
wait-prompt	Wait for monitor prompt output.
auto-again	Re-execute a batch file from the start.
quit	Cause the SIOCONS program to terminate.

Table: SIOCONS Local Commands

Examples of Batch Files

The following example batch file does several things. The PlayStation is reset, then a TIM file and a program are dwnloaded, keyboard input is waited for, then the program is executed.

Example

```
local dload wheel256.tim 80190000
local load program 80080000          // download program
local beep
sr sp 801ffff0
local echo Type Any Key to Execute Program
                                     // print text to SIOCONS console
local pause                          // wait for key press on the PC
go 80080100
```

Another way of executing batch files is to specify them at the MS-DOS prompt as an argument to SIOCONS, as shown below.

```
siocons  <autoexecution-file-name>
```

User Guide
Software Development Tool

- This product is sold on a membership agreement basis to Members of Yaroze, which is operated by Sony Computer Entertainment Inc.
- The PS symbol, 'PlayStation' and 'Net Yaroze' are trademarks of Sony Computer Entertainment Inc.
- Company and product names recorded in/on this product are generally trademarks of each company. Note that in/on this product the symbols '® 'and 'TM' are not used explicitly.

Published February 1997

Written and produced by :
Sony Computer Entertainment Inc.
Akasaka Oji Building
8-1-22 Akasaka, Minato-ku, Tokyo, Japan 107
Enquiries to: Network Business Project
E-mail:ny-info@scei.co.jp
TEL:+81 (0) 3-3475-1711

Sony Computer Entertainment Europe
Waverley House
7-12 Noel Street
London W1V 4HH, England
Inquiries to: The Yaroze Team
E-mail: yaroze-info@scee.sony.co.uk
TEL:+44 (0) 171 447 1616 / +44 (0) 7000 YAROZE

Sony Computer Entertainment America
919 E. Hillsdale Blvd., 2nd Floor
Foster City, CA 94404, USA
Inquiries to: The Yaroze Team
E-mail: yaroze@interactive.sony.com
TEL:+1-415-655-3600