PlayStation Sampler Disk

Specification for Playable Game Segments

SONY

COMPUTER
ENTERTAINMENT

Revision 3.0
4/15/99

Overview

This document sets out the requirements for generating a playable game demonstration to be included in a PlayStation sampler. This document includes a specification of the details involved in generating a demonstration, a list of materials, which must be submitted with the demonstration, and an **important technical checklist**. If you are submitting just video or graphics, please see the sections on Menu Content and Video and Graphics Specifications. *If you do not adhere to the published spec, if you do not include necessary materials, or if you do not complete the steps on the technical checklist, your submission will likely either not be included in the sampler or be delayed until it meets the specifications.*

Data can be submitted on a PC or Mac format cd-rom, zip cart, jaz cart or diskette if it will fit. Mail submissions to the attention of your account manager, **Sony Computer Entertainment America, 919 East Hillsdale Blvd., 2nd Floor, Foster City, CA 94404**. Files may also be directly FTP'd or emailed. Please contact SCEA for the current information on FTPing or emailing files.

Production Requirements

Demonstration Modes

Your application will be launched in one of two modes: interactive, or non-interactive. In either case, your application must, in every scene, menu, or screen, **terminate** if the rectangular **'SELECT' button is pressed** on any attached control pad.

Interactive Mode

This is the normal, playable demonstration of your game. In this mode that game players will get a feel for the gameplay and experience that your game offers. If your game contains separate levels/bouts/races/etc, then you should allow the player to either complete the game segment, or die/crash/fail trying. If your game is not segmented in this manner, then you should allow them to play long enough to get a basic feel and understanding of the game (a few minutes), and then your application should time out (preferably after a timeout warning). Your application should also terminate if there is no input from the user within the specified amount of time. All pause/configuration screens must also timeout accordingly.

Non-Interactive Mode

If the sampler has a kiosk mode (some samplers have this some do not) and is left alone for a minute or two, it enters an attract loop **-** alternating playback of promotional video and non-interactive applications (selected interactively). If your application is launched in this mode, then your application should run in an auto-play mode in order to demonstrate game play. As in the interactive mode, it should either terminate at the end of a game segment, or timeout within the specified time. If the nature of your application makes auto-play infeasible, then a video demo will suffice. In this case, your application should ignore the specified timeout, and terminate after one pass through your demo segment. If your non-interactive application is _just_ a movie, and there are no other title, feature, "coming soon," or other such screens, then playback will be handled by the sampler shell in order to reduce 'dead' time. Of course, demonstration of cool game features and scenarios is greatly encouraged, and overuse of pre-rendered footage is equally discouraged.

Full Product Information

In order to encourage gamers to buy your game, it is recommended that you put up a features screen at the end of your demonstration, detailing the scope and/or features of your full application. Providing a release date for your game may be problematic. If your schedule slips or if the demo is used after the release date the information may become stale. We won't be able to correct this problem unless you submit a new demo.

Disk Usage

Each game is allocated **40MB**. This space reflects the entire allowance for code, data, movies, and DA for both your interactive and non-interactive game demonstration. While you are allowed to fill this space with whatever you choose, you are greatly encouraged to keep your CD usage to a minimum, as we can use extra space for cool 'secrets' and the like. If your demonstration uses a lot of DA, please consider cutting down the size by including only one 'music' track, and/or by looping smaller sections of your DA tracks. **If your demo is unable to fit within 40MB, get in touch with SCEA** and we will try to arrange more space.

Loading & Program Start-up

Immediately upon beginning execution, your program should post a Title/loading screen, complete with the legal copy for your game. Clearly, in a kiosk environment, load times should be kept to an absolute minimum. Optionally, a loading progress bar or animation, or even a little game (as in Ridge Racer) is a great way to avoid having your game abandoned by the impatient customer.

Menu Content

As each playable demo is selected from the menu application, you will need to provide some content for use in the menu. Please provide a game logo in any Photoshop compatible format. The title of your game will be on the sampler interface. Please provide the exact title of your game as you wish to see it on the menu (provide correct capitalisation). The ESRB rating for each game is also displayed on the sampler interface. Please provide the correct ESRB rating for your game. If it has not yet been rated, provide RP.

The menu also **requires a controls screen**, or a list of the controls for your playable demo, or both. You may present the game controls within the demo. Please make sure that these controls are easy to access and correctly match the real controls for the game demo. If you are going to create a controls screen, the PlayStation Controller graphic is available from SCEA.

The design of the sampler menu depends on the specific disc. If your demo is going on a CD that has video in the menu interface you may provide 45-second video clip of game footage. Since this footage will not be playing back in a full screen, but on a screen within the screen, you may wish to 'zoom in on the action.' This footage can be submitted in an uncompressed digital format (as a set of 24-bit frames, or as a quicktime or .avi movie), or as an edited beta-sp tape, or as raw footage with a set of time codes for the intended cuts. If no footage is provided, then we'll just make some up (if this isn't acceptable, submit footage!). If your demo is going on a CD that uses game thumbnails in the interface and you wish to provide the thumbnail, please submit this in any Photoshop compatible format or we will create a thumbnail for you.

Required Menu Content Checklist

Please verify that all of the following are included with your submission:

- ☐ A logo for your game or demo
- ☐ The game’s ESRB rating
- ☐ The exact title of the game or demo as you wish to see it on the sampler menu
- ☐ A Controls screen or written controls for playable demos
- ☐ A Title screen if none exists in your video or playable demo

Failure to provide any of these items will result in phone-tag.

Video and Graphics Specifications

Videos that are intended to be stand-alone demos should be **fully edited and include sound**. They should begin with a title screen, include an ESRB rating if appropriate and any copyright information that you require. Please do not include the PlayStation logo unless your video was previously produced video segment (e.g. a TV spot or a merchandising video). The recommended limit on the length of a video is one and a half minutes. Please contact SCEA if you plan to significantly exceed this limit. Audio should be provided as 44khz, 16bit, stereo. Any captured game footage should be de-interlaced. If it is not de-interlaced we will correct the problem; but the resulting video will be lower in quality. Please make sure that your content is centered and that odd or varying black margins do not exist around your video. Letter box format is OK. Any readable text should be located in the title safe area of the screen. If you are sending your video on tape, higher quality tapes produce the best resulting PlayStation video after the capture and compression process.

Video footage should be submitted as one of the following:

1) A set of uncompressed graphic files, 640X480 or 320X240
2) An uncompressed 24-bit, 640X480 or 320X240, 15fps quicktime movie
3) An uncompressed 24-bit, 640X480 or 320X240, 15fps avi movie
4) A beta-sp format tape
5) Audio should be 44khz, 16bit, stereo (lower quality will work but won’t sound very good)
6) A PlayStation format compressed video.

Graphics can be submitted in any Photoshop compatible format. If you are providing a full screen graphic, please provide a 640X480 or 512X384, 72 dpi, Photoshop compatible file. The high-resolution files that are required for printing are not necessary for NTSC display. If you are having trouble with the size of your files, reduce the resolution to 72 dpi.

Technical Specification

Overview

The way the demo disk actually works is as follows. When the PlayStation boots up at power on, the launcher program is loaded from CD into main RAM and runs. The launcher is loaded to and runs within the 32K bytes of RAM right above the PlayStation kernel's 64K of RAM (80010000 - 80018000).

The launcher then launches the menu program, which allows the user to select and play the various games, and do whatever other activities are provided. Once the user has chosen your playable demo, your executable (.EXE) will be loaded from CD; the BSS segment will be cleared, and your program will be executed. While your program is running, the launcher will still be in RAM, so your program _must_ not write to memory between 0x80001000 and 0x80018000. Since you are probably used to not touching memory inside the kernel's space, it is anticipated that altering your code to avoid corrupting the launcher will probably only require that you re-link your code with an org address 32K bytes higher than before. Hopefully, losing this 32K of RAM will not require a lot of changes to your game code. If you are pushed for RAM, you may consider checking the size of the stack you are using - the default is 32K, which is pretty big. The launcher keeps its own small stack inside it's 32K, so you don't need to worry about corrupting it's stack; your stack (in standard configuration) is in the top 32K of memory. With the launcher loaded, and your program running, the main RAM looks like:

`0x80000000 - 0x80010000`	PlayStation Kernel RAM space
`0x80010000 - 0x80018000`	launcher program and stack
`0x80018000 - 0x80....`	Your demo code and data
`0x801f8000 - 0x801fffff`	Your stack (assuming the default size and position assigned by libsn)

In order to maintain sanity, **your demo must live within its own directory on the CD**, which will contain the data files used by your demo and any other information it needs, with the exception of any DA audio, which will be a separate track on the CD. You are encouraged to keep the number of files you use to a minimum, as some other pieces of code on the disk may use CdSearchFile, with its limitations of around 40 directories with about 30 files in each. If your application uses a number of files anywhere near this limit, then your application will likely cause other applications to fail (in which event, the applications are prioritised).

The launcher runs your program using the kernel call Exec(), and so your playable demo must be a standard PlayStation .EXE file. **Critically, in order for your playable demo to accept arguments, and return control to the launcher properly, you must link your code with the provided startup.obj, a replacement for the startup code in 2mbyte.obj and 8mbyte.obj.** This start-up module does not clear the bss and set up the heap, because if your playable demo does this, it may overwrite the launcher. The only problem is that code linked with startup.obj will not run except when launched from a harness program. A simplified version of our application launcher is provided with this distribution so that you can test your demo from within such a harness.

In addition to the memory and start-up restrictions described above, your program must do its initialisation, and tear down, as below. This small fragment of code is essentially a harness for a program that will return properly to the launcher and also leave the various PlayStation subsystems in a usable state.

```
#ifdef LINKED_STARTUP             /* If we have linked startup.obj */
int main( int argc, int* argv)  /* launcher will pass argc, argv to you. */
#else
int main()                        /* Plain old main instead. */
#endif                            /* PSX doesn't like argc, argv in a main prog */
  {
   ResetCallback();     /* Clear all of the CD callbacks. */

   EnterCriticalSection();               /* if your program uses Sony’s memory allocator */
   InitHeap(__heapbase, __heapsize);
   ExitCriticalSection();

   CdInit();            /* Re-initialise the CD subsystem. */
   PadInit(0);          /* Initialise the pads. */
   ResetGraph(0);       /* Cold boot the GPU. */
   SetGraphDebug(0);    /* Turn GPU debugging off. */

   /* Now you can do any other startup you need to */
   /* And the code from here on is your own        */

     .              .                      .
     .              .                      .
     .              .                      .
     .              .                      .
   /* Little Johnny has been killed by the giant spiders from Mars, so.. */
```

```
     /* This is the end of the program now. */

     /* reverse any callback initializations */
     /* stop and close any events */
/* failure to do these almost reliably causes crashes in the launcher or in other demos,
    and will equally reliably involve another revision of your demo */

     StopCallback();      /* Stop the CD callbacks. */
     PadStop();           /* Stop pad reading. */
     ResetGraph(3);       /* GPU warm reset */
     return (0);          /* This is necessary too. */
    }
```

The launcher will pass the standard C variables argc and argv to your program. However, argv will not be a ragged array of characters, it will actually just point to an array of integers; argc will be four, as your demo will be passed four pieces of information. **The contents of argv will be: the mode of the demo (interactive or non-interactive), the time-out you will use to stop the playable demo, the track index of the game's first DA track, and the sector # for the directory containing your app.** We will use ensure that your DA tracks are stored sequentially beginning at the specified track number. DA track numbers are 1 indexed. If the game does not use DA, arguments four will be zero and can be ignored. If your game does use DA, you will need to specify which files are to be DA tracks and in what order, as part of your submission form. While you are free to use whatever mechanism you choose to locate your files on the cd, the sector # for your application's directory is provided for speed, as your file positions can be hard coded as offsets from this value.

You can figure out the mode, time-out, position, and DA information using the following code:

```
#define INTERACTIVE                 0
#define NON_INTERACTIVE             1


#ifdef LINKED_STARTUP                  /* If we have linked startup.obj */
int main( int argc, int* argv)       /* launcher will pass argc, argv to you. */
                                       /* argv is actually an array of integers. */
#else
int main()                             /* Plain old main instead. */
#endif                                 /* PSX doesn't like argc, argv on its own */
  {
   int mode;                           /* Demo mode */
   int timeout;                        /* timeout in seconds. */
   int first_DA_track;                /* First DA track */
   int sector_offset;                  /* location of application directory */


   /* All the usual startup stuff etc.  */

   #ifdef LINKED_STARTUP
   mode = argv[0];
   timeout = argv[1];
   first_DA_track = argv[2];
   sector_offset = argv[3];
   #endif

   /* And on with the action */
```

If you have any additional technical questions, or require further assistance, please get in touch with your account manager, or Lifelike Productions for technical matters.

SCEA Demonstration Submission Form

(submit with demonstration)

Game Title:
Publisher:
Developer:
Project Supervisor (Name, Phone, E-mail):

Primary Technical Contact (Name, Phone, E-mail):

Style of Game (e.g. shoot-em-up, racing, etc.):

Game Function: Playable? Auto-Play for Non-Interactive? FMV only?

Controller / Peripheral Support (final game & this demo):

Size of EXE on CD (bytes):
Size of Data on CD (bytes):

Build Instructions (alternatively, submit a .cti or .ccs file with your data):

Estimated Final Release Date:
Any Other Comments or Issues:

Playable Section Technical Checklist
(MUST BE SUBMITTED WITH DEMONSTRATION)

Please verify all of the following technical details before submitting your demonstration:

- ☐ Demo linked with startup.obj.
- ☐ Demo executable and data lives within its own directory.
- ☐ Demo requires less than 40MB unless more space has been approved by SCEA
- ☐ Demo orged at 0x80018000 or above.
- ☐ Demo does not write to memory in range 0x80010000 → 0x80018000
- ☐ Demo does not initialize or use the memory card.
- ☐ Demo does not use the SetMem function.
- ☐ Demo does startup and shutdown as specified above
- ☐ Demo behaves according to the specified mode (interactive/noninteractive)
- ☐ Demo programs using DA pay attention to argv[3] in order to determine the appropriate DA tracks.
- ☐ Demo removes any callbacks and event handlers before closing down
- ☐ Demo can be quit in either mode with 'select' key at any point, on any screen.
- ☐ Demo falls out of main in order to exit.
- ☐ Demo terminates at the end of a game segment or after the specified timeout.
- ☐ Interactive demo will never sit without input for more than the specified timeout.
- ☐ Demo is configured for NTSC.
- ☐ Demo is not in the middle of DMA when closing down.
- ☐ Demo clears the reverb buffer in SPU RAM before closing down
- ☐ On a debug station, demo is tested to indefinitely launch successfully from the provided launcher, and return control to the launcher. (This loop should be verified for an absolute minimum of 10 iterations in both interactive and noninteractive modes).
- ☐ Demo does not depend on the state or contents of RAM (other than BSS), data cache, VRAM, SPU RAM, CD sector buffer, or I cache at startup
- ☐ Stack pointer has been set properly in your .EXE header to the value that you require (you can use setsp program included with this distribution to modify this)
- ☐ printf() inserted at the start of main() showing arguments received by the launcher
- ☐ printf() inserted at the end of main() to mark end of demo execution

Playable Section Submission Checklist

Please verify that all of the following are included with your submission:

- ☐ Data is submitted on a PC or Mac format cd-rom, zip cart, or jaz cart. Mail submissions to the attention of your Sony representative, **Sony Computer Entertainment America, 919 East Hillsdale Blvd., 2nd Floor, Foster City, CA 94404**. Files may also be directly FTP'd or emailed. Please contact SCEA for the current information on FTPing or emailing files.
- ☐ A playable demo disk used for debug station tests is submitted. (This may be the same as your submission disk if you include copies of all of your .xa, .da, and interleaved data as form1 files).
- ☐ File order and types are explicitly specified, preferably by including the .ccs or .cti file you used to build the playable demo that you tested in order to pass the technical checklist above.
- ☐ Title screen, controls screen or written control instructions, ESRB rating, and game logo are required. Screens should be in a common uncompressed PC, Mac, or PlayStation graphic format, and the video footage should be submitted as a set of uncompressed graphic files, an uncompressed 24-bit, 15fps quicktime movie, an uncompressed 24-bit 15fps avi movie, or on a beta-sp format or better tape. Videos should be fully edited with sound. In all cases, graphic data should be submitted at the highest production values available.